University of Bremen

Faculty of Mathematics and Computer Science

Computer Graphics and Virtual Reality Research Group

Master's Thesis

Design and development of hardware and software for wireless Head-Mounted Displays

by Dmitry Galkin
(matriculation number: 2780410)

Supervisors:
Dr. Gabriel Zachmann, *University of Bremen*
Dr. Marc Herrlich, *University of Bremen*

Thesis submitted in partial fulfillment of the requirements for the
Digital Media Master of Science degree at the University of Bremen.

2014 - 2015

Statutory Declaration

I hereby declare that I have written the enclosed Master's thesis by myself, I have not used sources or means without respective declaration in the text. Any thoughts from others or literal quotations are clearly marked and references are given. The Master's thesis was not used in the same or in a similar version to achieve an academic grading or being published anywhere else.

…..	…..
date	(signature)

Abstract

With the introduction of a new head-mounted displays models on the consumer market, the interest to those devices raised dramatically in the recent past. Although only a few wireless head-mounted display models are available today, there is a certain range of applications where those are required. Existing solutions that allow using conventional head-mounted displays for such applications wirelessly are either very expensive or too cumbersome.

This thesis is aimed to overcome those drawbacks with a new solution of a distributed design. Wired HMD is being connected to a compact, low-cost, portable computer that displays a real-time compressed video stream, transferred from a computer performing the actual rendering of an HMD-enabled application. A prototype according to the proposed design was developed and evaluated.

KEYWORDS: WIRELESS HEAD-MOUNTED DISPLAY, HMD, OCULUS RIFT, BANANA PI, VIDEO STREAMING, H.264, USB OVER IP.

Table of Contents

1. Introduction

"A display connected to a digital computer gives us
a chance to gain familiarity with concepts
not realizable in the physical world.
It is a looking glass into a mathematical wonderland."
(Sutherland, 1965, p. 506). Two years before the first HMD has appeared.

While first head-mounted displays appeared almost fifty years ago, as a part of collaboration between Ivan E. Sutherland - computer graphics and virtual reality pioneer (McCracken, 2013) and Bell Helicopter company - one of the first manufacturer of rotor-crafts (Bell Helicopter Textron Inc., 2014), they used to be limited to military and research use for a long time (Lowood, 2015).

The first head-mounted display worn by a pilot of a helicopter showed video from a servo-controlled infrared camera mounted beneath the helicopter. The camera moved with the pilot's head, both augmenting his night vision and providing a level of immersion sufficient for the pilot to equate his field of vision with the images from the camera. The display was so heavy that a suspension system was required to hold it (Lowood, 2015).

One of the first commercially available head-mounted display for gaming and entertainment purposes - Forte VFX-1, was shown on the Consumer Electronics Show only in 1994. With sales stared in 1997, it's took roughly thirty years for HMDs to become more widespread and available (Silicon Classics, 2012).

Nowadays we may observe a rapidly growing popularity of head-mounted displays with the introduction of new and affordable models primarily on the consumer entertainment market. According to market research agencies, the HMD market will grow up to 3.8 Billion USD in the year 2018 (Kzero Worldwide, 2014) and up to 12 Billion USD by the year 2020 (Markets and Markets, 2014).

Some of the newly appearing head-mounted display models are also announced to be wireless, one of them is Neo PRO by Light&Shadows. According to an advertisement, the HMD runs an Android operating system and any Windows or Linux application can be streamed remotely to the head-mounted display (it3D Magazine, 2014, p. 45).

1.1 Motivation

Although in many applications the head-mounted display is worn by a seated user or by a user who does not need a freedom of local movement, there are also use cases when user has to move in space while wearing a HMD. A number of such examples, demonstrating the need for wireless HMDs, is examined below.

During the experiment, conducted at Max Planck Institute for Biological Cybernetics, where gait parameters in Virtual Environments were analyzed, a head-mounted display was attached to the laptop located in participants' backpack (Mohler, Campos et al. 2007). The weight of the backpack reached 7.36 kg and HMD used (eMagin Z800) had a weight of 0.24 kg. The results of the experiment showed that users walk significantly slower in the VR, compared to real-world: “This appears to be due to both the weight of the HMD and backpack and the smaller vertical field of view” (Mohler, Campos et al., 2007, p.4).

A clinical research conducted in Royal Adelaide Hospital in Australia, where head-mounted displays were used for monitoring in general anesthesia, was performed also with laptop (hand-held) contained in backpack. The backpack was worn by anesthesiologists performing procedures in operating room and HMD attached to the laptop superimposed the patient's vital signs (Liu, Jenkins et al., 2010).

Research was conducted to evaluate if anesthesiologists can spend more time looking at the patient and less to the conventional patient monitor with a use of head-mounted displays. The weight and dimensions of the equipment were pointed among the major limitations: “the weight and bulk of the head-mount and backpack equipment was a major concern for participants. Technological improvements to superimposed information displays should result in smaller and less obtrusive devices” (Liu, Jenkins et al., 2010, p.1037).

A paper about Huge Immersive Virtual Environment (HIVE), that was build for a research on spacial perception and cognition at the Miami University, does not point the size or the weight (9.8 kg) of the designed portable setup as a limitation or a drawback (Waller, Bachmann et al., 2007). Setup included a rendering computer, HMD and a number of auxiliary devices mounted on a wearable frame as shown on the figure 1.1 below. Designed system allowed conducting experiments on distance perception in virtual reality environments within a large (570 m^2) physical space.

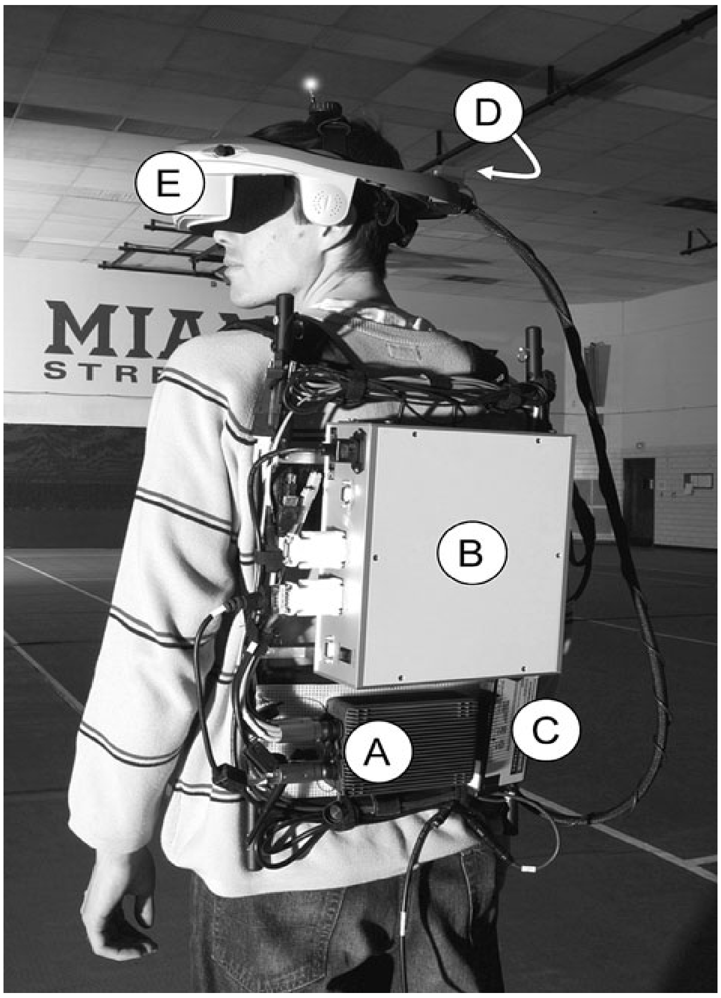


Figure 1.1: HIVE users wear a 9.8 kg backpack on which is attached their rendering computer (A), video control unit (B), and associated power supplies (C). An inertial magnetic orientation sensor (D) is attached to users' head-mounted display (E). (Waller, Bachmann et al., 2007, p.8)

The research results from the University of Virginia, however, demonstrate that at least the visual perception in real world is directly affected: "when people are tired or encumbered by wearing a heavy backpack" (Proffitt, 2006, p.120). Results state that participants who made estimations of distance while wearing a heavy backpack judged the targets to be significantly further away, than the participants who made the same estimations without the backpack (Proffitt, 2006). That raises a question, if a previously described setup with backpack utilized in HIVE could actually affect the results of experiments conducted.

Another example is from the Vienna University of Technology, where users wearing a head-mounted display had to move within a room. In the Institute of Software Technology and Interactive Systems HMDs were utilized for a collaborative, educational augmented reality application with six people located in one room (Csisinko & Kaufmann, 2011). Considering the optical user tracking system, the number of cables used is at least equivalent to the number of displays. As cables lied on the floor they got tangled as users started to move which led to the restriction of movement and interaction possibilities: “A major hindrance for practical usage of such a setup was the number of HMD cables.” (Csisinko & Kaufmann, 2011, p.1). The authors of the paper proposed their solution for making the HMDs wireless, which is described later in the section 2.3 of the following chapter.

The applications of wireless head-mounted displays may also include various fields, for instance ARVIKA project suggests the use of mobile Augmented Reality system with HMDs for production, service and assembly (Friedrich, 2003). Or HMDs can even be used for a computer-aided surgery operations (Birkfellner, Figl, Huber et al., 2002).

Many other examples can be found in different domains of head-mounted display applications. In fact, even when HMDs are used for entertainment and users don't need much of a freedom to move around, the would still prefer using HMDs with no wires attached to their computers (Oculus VR, 2013). All that demonstrates the motivation to make head-mounted displays wireless, lightweight and portable at the same time.

1.2 Requirements

The aim of this work is therefore to design a system that would allow using a conventional wired head-mounted display wirelessly and to build a prototype to make an initial evaluation and ensure feasibility of the designed system.

While topic is not restricted to a specific type of the head-mounted displays and their applications, binocular HMDs with wide field of view are considered as the target class of devices. Those are normally used for Virtual Reality applications with computer-generated images.

Requirements for a wireless head-mounted display can be defined as follows:

1. Head-mounted display should operate autonomously with no wired connections to the stationary sources of power or to a computer that runs respective HMD-enabled application.
2. The operating range of the wireless HMD should allow its use at least within the same room (min. 100 m^2) where computer is located.
3. The performance of the wireless HMD from the user's point of view preferably should not differ from its wired version. Performance here implies latency, screen resolution and screen refresh rate.
4. The wireless HMD should be lightweight and portable. The weight of extra hardware utilized with HMD should be estimated by at most few hundred grams. The setup should not require the use of big bags or backpacks for portability.
5. The costs for hardware and software used to make head-mounted display wireless should not exceed 100 EUR.

While requirements (1) and (4) are justified by the highlighted examples from the Motivation section of this chapter, the explanation of the requirements under numbers (2), (3) and (5) follows.

As room sizes and operating ranges were not always specified in the previously described use cases of wireless HMDs, an operating range within at least 100 m^2 room is proposed. Taking into account the fact, that an average office room has a size of approximately 25 m^2 in Germany (Voss, 2011), that should presumably cover most of the potential use cases.

The requirement (3) basically defines that it is desired to keep the performance of the wireless HMD as similar to its wired version as possible. Preferably. the user should not notice any difference when HMD is connected via wires or if it operates wirelessly and can be used portably. HMD latency, screen resolution and screen refresh rate can be defined as major factors, if those parameters differ significantly for wireless head-mounted display (when compared to its wired version) the HMD may become unsuitable for some applications or even completely unusable if end-to-end latency will be too high (Brooks, 1999).

The requirement (5) was added as consumer and entertainment head-mounted displays appearing nowadays become relatively inexpensive. At the moment of writing, the Oculus Rift Development Kit 2 costs 350 USD and according to P. Luckey - CEO of the Oculus VR, the release version should not be significantly more expensive and have a price in range of 200 – 400 USD (Chacos, 2014). In the upcoming chapter, where several existing wireless HMDs solutions are reviewed, the prices of the used extra wireless hardware start from approximately 300 EUR. Therefore, a low price level (100 EUR) is defined among other requirements.

2. Related Work

This chapter gives an overview of existing wired head-mounted displays and known solutions that allow using HMDs wirelessly. Also describes related video interfaces, wireless standards and existing technologies that can be applied to turn a conventional wired head-mounted display into wireless one.

2.1 Wired HMD Interfaces

Modern binocular head-mounted displays with wide field of view, like Oculus Rift Development Kit 1 (DK1), Development Kit 2 (DK2) or Sony Morpheus have a video connection provided via one of the common wired digital interfaces: Digital Visual Interface (DVI) or High Definition Multimedia Interface (HDMI). Wired data connection with a computer is normally provided via the Universal Serial Bus (USB) to deliver the input from embedded tracking sensors (Oculus VR, 2013.; Goradia, Doshi & Kurup, 2014).

New entertainment head-mounted displays and their prototypes are also expected to have embedded sound system, like for instance Oculus Rift Crescent Bay prototype, that can use HDMI for multichannel audio delivery (Walton, 2015). Oculus Rift Development Kit 2 HMD also has a separate connection to synchronize the frequency of the IR sensors of emitter and receiver for the depth tracking system. Same Oculus Rift DK2 also features a single USB port hub to attach other devices for use with HMD, like a Leap Motion sensor (Goradia, Doshi & Kurup, 2014).

Only a relatively few HMDs with analog video interfaces can still be found in production nowadays. Carl Zeiss Cinemizer is one of them, however it has both analog video and digital HDMI interfaces available at the same time (Carl Zeiss, 2015).

Taking into account this information, the goal of making a wireless connection with HMD can also be interpreted as a goal of making a digital video (DVI or HDMI) and data (USB) interfaces wireless. Suitable technologies are reviewed in the following sections.

2.2 Existing Wireless Video Technologies

Technologies and standards available as of today allow establishing connections using aforementioned digital interfaces of head-mounted displays wirelessly with an extra hardware and software.

One of the relatively new WirelessHD (also known as UltraGig) standard suggests a wireless implementation for high-definition video and audio content streaming with data rates up to 25 Gbit/s (WirelessHD Consortium, 2008). Although the standard was finalized in 2008, there are only two devices were found at the moment of writing: DVDO Air and Sharp VR-WH1, both operating in the 60GHz band (Wi-Fi IEEEE 802.11ad compliant) and priced over 200 EUR (WirelessHD Consumers, 2015).

Among other wireless multimedia standards are: Intel Wireless Display (WiDi, Intel, 2011), Wireless Home Digital Interface (WHDI LLC, 2011) and Miracast (Wi-Fi Alliance, 2012).

Intel Wireless Display (WiDi v.4.2) main features (Intel WiDi, 2011-2015):

- Full HD (1920 x 1080) video support up to 60 FPS;
- Hardware-accelerated video encoding (H.264);
- High-bandwidth Digital Content Protection (HDCP) 2.x support;
- Up to 6-channel audio;
- Low latency mode (< 150ms, but 60 FPS not supported in this mode);
- 2.4GHz and 5Ghz Wi-Fi support (with compatible receiver);
- Compatibility with Miracast standard.

Wireless Home Digital Interface (WHDI v.2.0) main features (Cnet, 2010):

- Uncompressed Full HD resolution support with 60 Hz refresh rate;
- 3D video modes support with 30 Hz refresh rate;
- HDCP 2.x support;
- Up to 6-channel audio;
- Latency < 1ms.

Miracast main features (Wi-Fi Alliance, 2012):

- Full HD video support;
- Up to 6-channel audio;
- Uses Wi-Fi Direct connection;
- Standard does not define the maximum latency.

2.3 Existing Wireless HMD Solutions

With a growing popularity of head-mounted displays and their applications, a large number or solutions to make wired HMD wireless appeared in the recent past. In most cases, proposed solutions rely on the technologies and standards described in the previous section.

The paper "Cutting the Cord: Wireless Mixed Reality Displays" describes a solution where Zinwell ZWD-2500 WHDI devices, powered by Li-Pol batteries, were used to provide wireless connection with see-through head-mounted displays (Sony Glasstron and eMagin Z800) and wireless hand-held displays in laboratory and educational environments. Designed wireless setups for mixed reality applications, as shown on the figure 2.1 below, allowed collaboration of 6-8 users located within one room (Csisinko & Kaufmann, 2011).

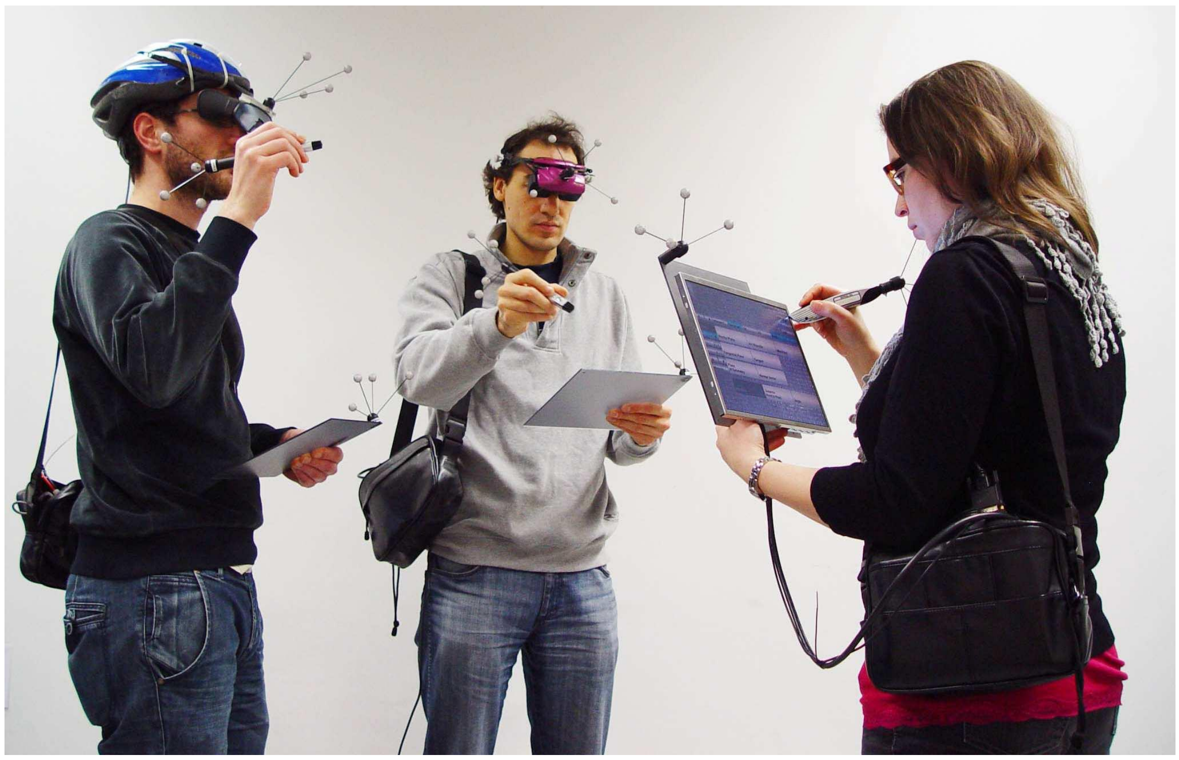


Figure 2.1: Wireless TFT display and two wireless HMDs used by students (Csisinko & Kaufmann, 2011, p. 8).

The Research Group of Cognitive Neuroinformatics of the University of Bremen used Oculus Rift DK1 HMD with a HD1080 Wireless Video Link device from Sensics, which is optimized for use with Sensics head-mounted displays and comes with an embedded battery. The device is also certified to comply with the WHDI standard and operates in 5GHz band (Sensics Inc., 2011-2013). The applications of the setup include VirtuSphere and research on spatial cognition.

W. Steptoe, Virtual Environment and Computer Graphics researcher at the University College London, used an Asus Wavi WHDI device in combination with external battery pack to make a modified Oculus Rift DK1 wireless, which also delivered sufficient performance for a developed augmented reality application (Steptoe, 2011-2015).

While all of the described solutions are suitable for using a head-mounted display wirelessly, they share a common drawback - high hardware price. The specialized solution from Sensics is priced 2000 USD (Sensics Inc., 2011-2013), the device from Zinwell - 500 EUR (Aliexpress, 2015) and the most inexpensive – Asus WAVI approximately 250 EUR (Amazon, 2014). This does not include the associated costs for rechargeable battery packs, HMDs and other hardware used. Although these costs can be totally affordable in some circumstances, they may seem too high, if compared with the price of the HMDs used. EMagin Z800 HMD was last priced 549 USD (Oled-Info, 2006) and Oculus Rift DK1 was sold for 300 USD (Chacos, 2014).

Secondary drawbacks, applicable to Zinwell and Asus WHDI devices, are the dimensions (241x171x29mm for Asus; 181x145x33mm for Zinwell), wearable weight (1.2kg for Asus; 0.8kg for Zinwell) and power consumption (24 Watts for Asus; 15 Watts for Zinwell). The WHDI device from Sensics meanwhile is significantly smaller (60x90x20mm), lighter and has an embedded battery that allows up to 3 hours of autonomous operation. Comparing to previously exposed “backpack” solutions these points obviously cannot be considered drawbacks, as devices are relatively small and portable.

2.3.1 Intel WiDi and Miracast for HMDs

As the Intel WirelessDisplay compliant devices were not previously used with head-mounted displays, a number of tests was performed in order to evaluate the potential for use of such devices with HMDs.

For that purpose a Netgear PTV3000 Push2TV (Netgear, 1996-2015) dual band WiDi adapter was used with Oculus Rift DK1 HMD attached to it. A laptop (Samsung 530U4C) with Intel N6235 wireless network adapter was used as a source of video input. While setup performed significantly better when operating over the 5GHz wireless band, it still failed to deliver acceptable performance. The delay that appeared between the changes on the laptop screen and the display attached to WiDi adapter can be clearly seen on the video recording attached in the Appendix section (9.9 WiDi Latency). Finally, WiDi is restricted to use only by laptops equipped with wireless network adapter from Intel and a limited set of Intel Core series processors. Desktop solutions were not available at the moment of writing.

The Miracast compliant devices were not tested as numerous negative feedbacks from the users were found (Bott, 2014). Video recordings available clearly demonstrate that those devices are not suitable for use in applications where low system latency is crucial for system performance. One of the tests shows latency of at least 165ms (Paine, 2014).

2.4 Video Modes and Interfaces

To estimate the amount of data transferred over the digital video interfaces used with head-mounted displays a short overview was conducted. The table 2.1 below reflects the bit rates of raw video and H.264 (baseline profile) compressed video for common video modes supported by modern HMDs. Assuming that color space is RGB (4 : 4 : 4) and color depth is 8 bits/color as a default setting for modern computer graphic cards and monitors (Yurek, 2011).

Screen Resolution (2D) (pix x pix), (aspect ratio)	Display Refresh rate (Hz)	Uncompressed bit rate (raw RGB video, Gbit/s)	H.264 baseline compressed bit rate* (YUV420 video, Gbit/s)
1280 x 720 (16 : 9)	60	1,33	0.00790
1280 x 800 (16 : 10)	60	1,47	0.00877
1920 x 1080 (16 : 9)	60	2,99	0.01777
1920 x 1080 (16 : 9)	75	3,73	0.02221
2560 x 1440 (16 : 9)	75	6,64	0.03949

Table 2.1: Bit rates overview for common high definition video modes (Forret, n.d.; Thomas, 2010-2014).

The table 2.2 below shows maximum data throughput for the common digital video interfaces (Anthony, 2013.; Denke, 2010).

*significantly depends on the chosen color space, compression rate, encoding settings and video content. Calculated with Video bitrate calculator (Thomas, 2010-2014).

Video Interface (version)	Maximum (video) data rate (Gbit/s)	Maximum (video) data rate including overhead** (Gbit/s)
DVI-D (single link)	3,96	4,95
HDMI (v. 1.0)	3,96	4,95
HDMI (v. 1.3)	8,16	10,2
HDMI (v. 2.0)	14,4	18

Table 2.2: Data rate throughput capabilities of common video interfaces (Anthony, 2013.; Denke, 2010).

From the table 2.1 can be inferred that even for a 1280 x 720 @ 60Hz high-definition resolution an effective bandwidth of 1.33 Gbit/s is required to transfer the uncompressed raw video stream. The table 2.2 demonstrates, that all the common interfaces, including DVI-D that first appeared in the year 1998, are capable of transferring HD video content of at least Full HD resolution over wires.

2.5 Wireless Network Standards

An overview of existing local area wireless computer network standards was conducted in order to estimate the capabilities for transferring video streams over such networks. The table 2.3 below collates the basic parameters of a several, common today wireless network standards.

**the overhead is created by extra encoding to provide higher skew tolerance, reduce electromagnetic interference. For DVI and HDMI interfaces a Transition-minimized differential signaling (TMDS, a form of 8b/10b encoding) is used (Digital Display Working Group, 1999).

Wireless Standard	Operating frequency band (GHz)	Max channel bandwidth (MHz)	Max number of streams (MIMO)	Max theoretical data rate* (Mbit/s)
Bluetooth v 4.0	2.4	2	N/a	24
Wi-Fi IEEE 802.11n	2.4	20	4	450
Wi-Fi IEEE 802.11n	5	40	8	600
Wi-Fi IEEE 802.11ac	5	160	8	1000
Wi-Fi IEEE 802.11ad	60	2160	N/a	6750

Table 2.3: Common wireless standards bandwidth and transfer capabilities (IEEE 802.11, 1997-2014).

The transfer data rates present in the table 2.3 are a theoretical maximum according to respective standards specifications. Depending on a set of circumstances including interference, distance between the sender and receiver, maximum allowed signal strength and some other those values can differ (Hummel et al., 2007).

Considering the information from the tables 2.1 and 2.3, the 2.4GHz and 5GHz bands wireless networks cannot provide sufficient data transfer rates to transport raw video content with HD resolutions (1280 x 720 or higher). Only Wi-Fi IEEE 802.11ad wireless networks operating in 60GHz band can theoretically provide transfer rates that would be sufficient for Full HD (1920 x 1080) video modes.

The IEEE 802.11ad wireless standard is a relatively new one, it appeared in the year 2010 and at the moment of writing, only a limited number of compliant devices was available. Among those is a Nighthawk X6 wireless router from

* Legacy wireless 802.11 networks are half-duplex. The bandwidth is shared by both incoming and outgoing data streams (Duarte, Sabharwal et al., 2012).

Netgear (Netgear, 1996-2015). Besides its 300 EUR price (Amazon, 2015), the device support only 1000 Mbit/s local area network connection, thus making it also unsuitable for transferring raw high definition video in real-time.

Another fact, that should be considered when IEEE 802.11ad wireless networks are used, is that at 60 GHz band the signal cannot penetrate walls and obstacles as good as on lower frequency bands (Smulders, 2003). Therefore, one of the requirements for 60 GHz band Wi-Fi is to have a clear line-of-sight between the sender and the receiver.

While other wireless network standards are not suitable for raw high definition video streams transfer, they do provide enough theoretical bandwidth to transmit H.264 compressed video streams, with resolutions also higher than Full HD (1920 x 1080), as can be seen from the tables 2.1 and 2.3.

2.6 Wireless USB Connections

Since most of the head-mounted displays require a Universal Serial Bus connection with a computer to transfer the input from tracking sensors, a way to establish it wirelessly is needed. Some of the WiDi and WHDI compliant devices, including mentioned before Asus WAVI (ASUSTeK, 2015), allow connecting one or two USB devices remotely. The Miracast standard does not specify such options in its current revision.

To provide wireless USB connections a Wireless USB (also known as W-USB or Certified Wireless USB) standard compliant devices can be used. The W-USB specification of version 1.0 suggests maximum data throughput rate of the wired USB v.2.0 (480 Mbit/s) within a range of 3 meters and 110 Mbit/s rate up to 10 meters distance between the sender and receiver (USB Implementers Forum Inc., 2006). As the caring frequencies standard suggests use of a wide range from 3.1 GHz to 10.6 GHz.

While the first version of W-USB standard appeared in the year 2005 and version 1.1 was released in 2010, W-USB certified devices are still rare and also expensive (Oxford, 2011). One of the possible reasons is that Intel, being among the major contributors of W-USB, has dropped the development of respective ultra-wideband chips used by W-USB devices in 2008 (Fleishman, 2008).

2.7 HMD USB Usage

In order to estimate the actually used USB data throughput by HMD a test with a USBlyzer application was performed (Usblyzer, 2006-2014). Oculus Rift DK1 HMD was connected to a USB 2.0 port of a desktop computer (specifications can be found in the Appendix section 9.1) with the original USB 2.0 cable that came along with the HMD.

The detailed study of the USB Properties of the HMD obtained via USBlyzer showed that, while the device also complies with USB v.2.0 specification, the current connection speed was only 12 Mbit/s, which corresponds to the USB specification of versions 1.0 and 1.1 (Compaq, Intel et al., 1998). Full set of extracted USB properties can be found in the Appendix section (9.2 USB Usage Analysis).

A number of HMD-enabled demo applications, including “Tuscany” (Oculus VR, 2013), and “Alone in the Rift” (Oculus VR Forums, 2013) were used for the test while USB communication was captured by USBlyzer.

The analysis of the recorded communication showed that the connection speed always remained the same (12 Mbit/s) and that transfer buffer always had a constant size with 62 bytes of data. Stable rate can be explained by the constant refresh rate of the tracking sensor (Desai et al., 2014). The size of the transferred tracking information will therefore remain the same for any application or use conditions of the same HMD.

2.8 Related Work Summary

Summarizing the results of the analysis of the related technologies, existing standards and solutions as well as tests performed with respect to the initially stated requirements, several conclusions can be made:

1. The WirelessHD compliant devices can presumably be used for head-mounted displays, however at the moment of writing those devices were not widely available and expensive. Considering the requirements for the additional costs of the wireless HMD those devices cannot be used.
2. The Wireless Home Digital Interface (WHDI) compliant devices can deliver sufficient performance for wireless use of head-mounted displays, as proved by a number of solutions reviewed in the section 2.3. The devices of WHDI standard cannot however satisfy the requirements on the additional costs and/or the portability for the wireless head-mounted displays.
3. The Intel Wireless Display (WiDi) and Miracast devices can satisfy the stated requirements on the added costs and portability, however tests

performed with compliant device from Netgear demonstrated a high, clearly-visible level of latency. The Miracast devices were not tested due to negative feedbacks and test results found.

4. The raw (RGB) uncompressed video data bit rate for high definition resolutions and common display refresh rates requires at least 1.3 Gbit/s of effective data throughput. Some video modes (like 2560 x 1440 @ 75Hz) would require more than 6 Gbit/s of data throughput. While many wired video interfaces can transfer those data rates in real-time, modern wireless network standards specify significantly lower data transfer capabilities, that are, nevertheless, suitable for compressed video data.
5. The Wireless USB (Certified Wireless USB or W-USB) devices are expected to deliver sufficient performance for use with head-mounted displays since, according to the tests conducted, the USB v.1.0 (v.1.1) connection speeds are sufficient. (Tested with the Oculus Rift DK1, data rates for other HMDs can be different). The W-USB devices are also rare and expensive at the moment of writing (Oxford, 2011).

Can be concluded, that an ordinary combination of the reviewed standards and technologies cannot be used to make a head-mounted display wireless and satisfy all the requirements that were previously defined in the Introduction chapter at the same time.

3. Solution Proposed

This chapter describes a proposed solution that is expected to satisfy initial requirements. The solution is based on the assumption that a head-mounted display has two interfaces, one for video signal and one for USB connection. Other interfaces as described before in section 2.1 of the Related Work chapter are not considered, as of now as they are only present on several head-mounted display models. The goal therefore is to establish a wireless video transfer and a wireless USB interface connections with the head-mounted display.

3.1 Video Transfer

Considering the maximum transfer rate limitations of today's wireless standards and the raw video bit rate at high definition resolutions, along with extra requirements on the wireless head-mounted display portability and pricing, as was described in previous chapters, it is proposed to transfer a compressed video instead of a raw video stream over the IEEE 802.11 wireless networks to a secondary computer with HMD attached.

In this case, to create a compressed video stream either a hardware or a software screen capturing device is needed. After the frame is captured, a video stream should be created and compressed. For compressing it is suggested to use one of the advanced video coding standards nowadays – H.264, also known as MPEG-4 Part 10, Advanced Video Coding (AVC) (Marpe, Wiegand, & Sullivan, 2006).

The particular choice for video coding standard is determined by a number of factors:

1. The encoding and decoding processes should be performed in real-time with high definition resolution and adding minimum of extra latency (estimated by the order of milliseconds or tens of milliseconds) as in many virtual reality applications the system usability is latency-depended (Brooks, 1999).
2. The encoded video stream should provide (visually) lossless picture quality when decoded and shown on the HMD.
3. The encoding should be preferably done with a free library due to overall costs limitation.

3.2 Overall System Design

A distributed system design with a Server and a Client computers is proposed. Besides a computer that would normally have a wired HMD connected and performing the rendering tasks, a second, portable computer will be used with HMD connected to it instead.

The computer performing the rendering where the wired HMD would have normally been attached to is named a Server computer further on. The portable computer with HMD interfaces connected is named a Client computer further on. The data transfer between the Server and Client can be performed over the wireless 5GHz 802.11n Wi-Fi network.

With suggested design a portable Client computer with HMD attached can be carried by the user as long as it is powered via an external battery pack and located in range of Wi-Fi network stable connection.

An overall scheme of the proposed design with tasks performed by the Server and Client computers can be found on the figure 3.1 below.

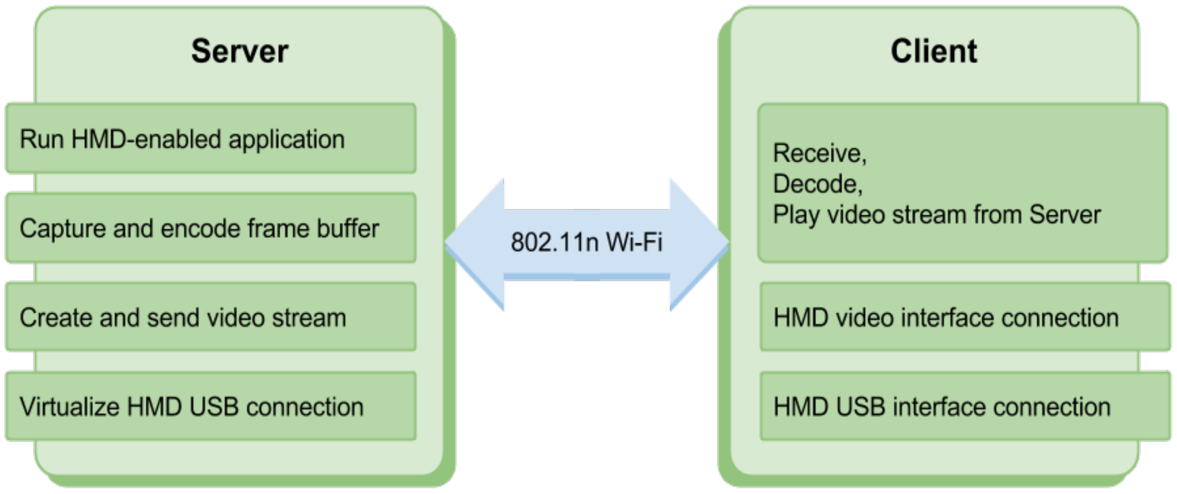

Server
Run HMD-enabled application
Capture and encode frame buffer
Create and send video stream
Virtualize HMD USB connection
802.11n Wi-Fi
Client
Receive,
Decode,
Play video stream from Server
HMD video interface connection
HMD USB interface connection

Figure 3.1: Main tasks performed by used computers according to the proposed system design.

The tasks performed in real-time on the Server computer within the system can be than defined as follows:

1. Execute and render the HMD enabled application, in a same way as that is done with a wired head-mounted display;
2. Capture the frame buffer;
3. Compress the captured frame to reduce its size;
4. Push the compressed frame to the video stream (point-to-point streaming);

5. Receive the HMD sensors input data for the rendered application from the Client computer.

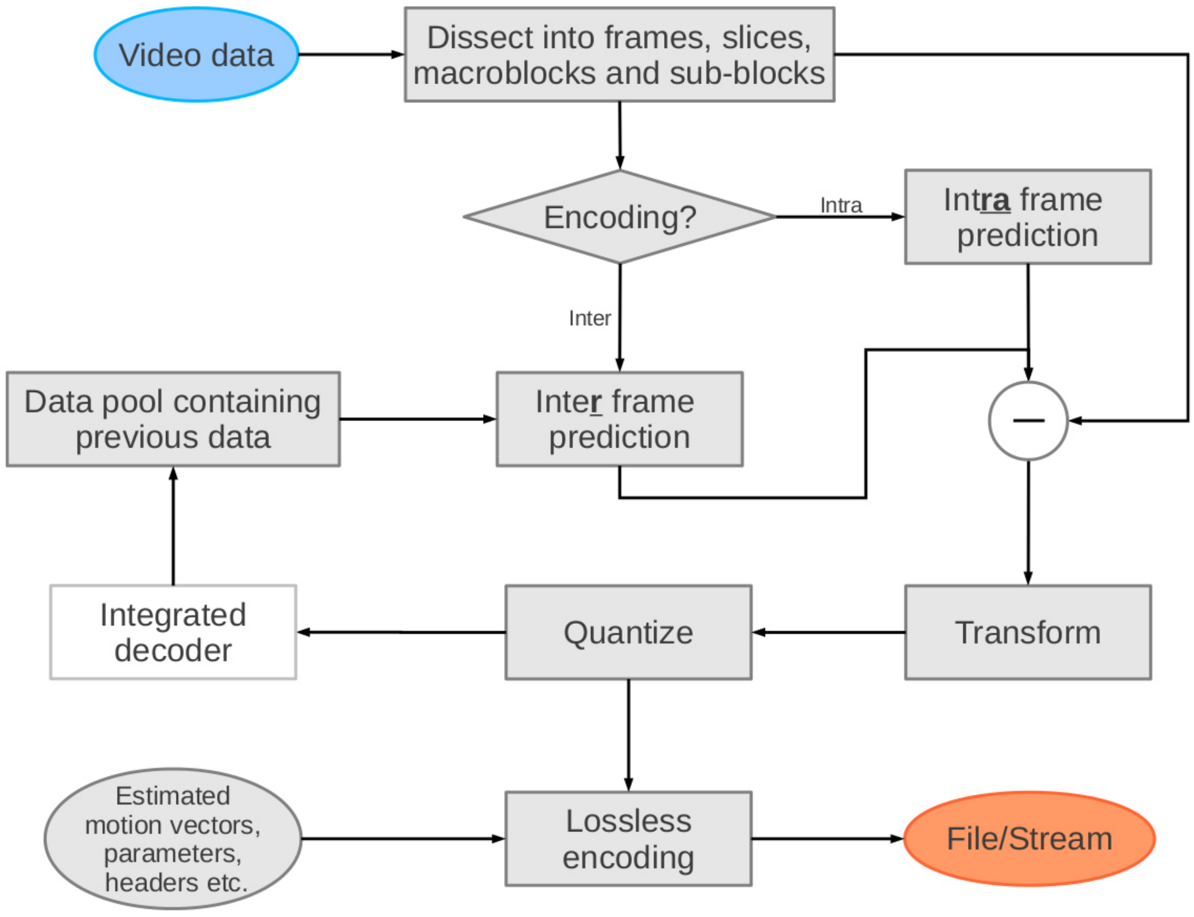
The tasks performed in real-time on the Client computer within the system can be defined as follows:

1. Pass the HMD sensors data to the Server computer;
2. Receive the video stream from the Server computer;
3. Decode and play the received video stream (on the HMD attached);
4. Forward the HMD sensors data to the Server computer.

3.3 H.264/AVC Video Coding Standard

H.264/MPEG-4 AVC is a block-oriented motion-compensation-based video compression standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC JTC1 Moving Picture Experts Group (MPEG). The H.264 standard can be viewed as a family of standards composed of different profiles. A specific decoder can decode at least one, but not necessarily all of the existing profiles. The decoder specification describes which profiles can be decoded (Marpe, Wiegand, Sullivan, 2006).

The general H.264 encoder flowchart can be found on the figure 3.2 below. The video data is first dissected down to blocks. A prediction is created by either inter or intra frame prediction for each block. The prediction is subtracted from the original data to obtain the residual data. This data is then transformed, quantized and encoded. Accompanied by parameters and headers this results in (optionally lossless) video file or video stream. In addition to that, there is an integrated decoder. This integrated decoder is used to assure that both encoder and decoder have the same data for the inter frame prediction used for compression (Hermans, 2012).



Video data
Dissect into frames, slices, macroblocks and sub-blocks
Encoding?
Intra
Intra frame prediction
Inter
Data pool containing previous data
Inter frame prediction
−
Integrated decoder
Quantize
Transform
Estimated motion vectors, parameters, headers etc.
Lossless encoding
File/Stream

Figure 3.2: General H.264 encoder flowchart (Hermans, 2012).

The H.264 standard defines a sets of capabilities, which are referred to as profiles, targeting specific classes of applications. These are declared as a profile code and a set of constraints applied in the encoder, which later allows a decoder to recognize the requirements for decoding of that specific video stream.

There are three basic profiles: Baseline, Main, High and more than 10 profiles that originate from those and differ mostly by the supported features, applications and restrictions. As the designed system should perform with minimal additional latency, the Baseline and Constrained Baseline profiles are the most suitable for use, since they provide minimum encoding and decoding times (Hermans, 2012).

The Constrained Baseline in fact incorporates a number of common features from Baseline, Main and High profiles which makes it more flexible in configuration. It's also the most commonly used profile for video conferences and mobile applications as the Baseline profile is natively supported by all decoders. Information on used H.264 encoding software and its configuration follows in the Implementation chapter.

Besides high compression rates and an option to preserve lossless video quality, the H.264/AVC is well supported by a large variety of hardware - GPU capabilities can often be utilized for encoding or decoding H.264 video (Halldal, 2012). However common mobile CPUs with ARM architecture as used in many modern smartphones cannot normally allow real-time decoding of H.264 streams with high frame rate and high definition resolutions, unless that's a solution like ARM Neon (Reddy, n.d.).

The H.264/AVC standard had a number of changes since the initial release (Wiegand, Sullivan et al., 2003). At the moment of writing, the most recent version is number 20 (appeared in April 2013).

3.3.1 H.264 Hardware Support

With the rising popularity of on-line games nowadays (Persistence Market Research, 2014), the amount of people watching others play in real-time has also increased dramatically (SuperData Research, 2014), thus video capturing, compressing and streaming are becoming more common tasks. As many games are demanding applications on its own, extra tasks to capture, compress and stream the game play in real-time to a service like twitch.tv (Twitch.tv, n.d.) are now being given to the GPU by Nvidia and AMD graphics card manufacturers.

In the middle of 2013 Nvidia launched a series of portable devices named Shield, that operate according to the following principle. The actual game is running on the desktop workstation with a (capable) graphics card and the game-play is streamed in real-time over a 5GHz wireless network to either a small gaming console (Shield Portable) or to a gaming tablet (Shield Tablet). User can enjoy a gaming PC performance, but on a light and portable device. Now Nvidia additionally allows to use user's desktop workstation as a part of Nvidia Grid cloud and thus deliver rendering as a service for Shield devices in any place with broadband Internet access (Nvidia, 2015).

Technology utilized by Shield system was first introduced in 2012 with Kepler based GeForce GTX 6xx gaming graphic cards. It is also available on Nvidia Quadro graphic cards. The technology is called NVENC and designed for real-time H.264 video encoding. The video stream encoding in this case is done by an extra SIP hardware and allows to process up to 480 frames per second with Full HD (1920 x 1080) resolution with a baseline H.264 profile for high-end graphic card models (GeForce GTX 680 and better). While, entry-level graphic cards do not support NVENC, the mid-level cards like GTX 650 can deliver at least 100 FPS at Full HD resolution (Mohapatra, 2014).

AMD now offers a similar technology called Video Coding Engine (VCE) which can be found as a part of some APUs and GPUs produced by AMD (Ihab, 2014). With Core series processors of the second generation (Sandy Bridge and newer) Intel also introduced a Quick Sync technology. It is designed to encode video in a real-time with low latency (Intel Quick Sync, 2013). In this case the encoding or decoding is performed with use of SIP core that's a part of the integrated GPU (Intel HD series). This one is used by the previously described Intel Wireless Display technology.

Summarizing, there is a number of hardware-accelerated technologies available with modern graphic cards, that can be used on the Server computer side only, as those are either desktop- or laptop-ready solutions.

3.4 USB Data Transfer

In the previous chapter a Certified Wireless USB standard was described and a test on the USB usage by Oculus Rift DK1 head-mounted display was conducted. Test demonstrated that the actual USB connection speed and data throughput do not exceed 12 Mbit/s.

Since the Certified Wireless USB devices were not widely available at the moment of writing and taking into account the requirements on the overall pricing and portability of a wireless HMD, it is proposed to emulate the presence of the USB connection on the Server computer with a virtual USB driver and send the USB I/O messages via TCP/IP payload from the Client computer.

This approach allows to avoid the use of extra hardware and to connect the USB port of the head-mounted display to the Client computer, therefore minimizing costs.

3.5 Wireless Network

In the sections 2.4 and 2.5 of the Related Work chapter an overview of video modes used with head-mounted displays and of the capabilities of modern wireless network standards was given. The 5GHz IEEE 802.11n Wi-Fi network is suggested for use as it provides sufficient capabilities for transferring H.264 encoded video streams with high definition resolution (including Quad HD, 2560 x 1440) and high frame rate (75) as well as suitable for forwarding the USB head-mounted display data (up to 12 Mbit/s) from the Client computer to the Server.

The IEEE 802.11n compliant network adapters are also relatively inexpensive and common, which makes them a suitable choice considering the requirements for designed system. The 5GHz spectrum is preferred over the 2.4GHz due to higher channel width, transfer speeds, output power, less noise and interference (Geier, 2014). However, most of the devices available are dual-band and support both 2.4GHz and 5GHz bands at the same time.

3.6 Proposed Solution Summary

Summarizing here the most important points of the proposed design for wireless head-mounted displays:

1. There is no modification of the head-mounted display hardware, the HMD remains wired, but being connected instead to a secondary, portable computer (Client).
2. The frame buffer of the Server computer that performs the rendering of the HMD-enabled application is captured with a frame rate that is equal or higher, than the HMD screen refresh rate is.
3. The captured frame buffer information is being encoded using H.264 AVC video coding standard.
4. A video stream of encoded video is created and sent to the Client computer over the 5GHz IEEE 802.11n Wi-Fi network.
5. The Client computer receives the video stream, decodes and plays it on the attached head-mounted display.
6. The USB data from the HMD is forwarded to the Server computer being encapsulated into network packets payload over the same 5GHz wireless network.

3.7 Design Limitations

While the design proposed is expected to meet most of the requirements defined initially, it clearly won't improve the head-mounted display performance, at least in a sense of latency.

In Virtual Reality, latency is widely recognized as a key source of disorientation and disbelief, so it is crucial to keep it as low as possible. The end-to-end latency of the HMD is the time it takes between user's head moved to a new orientation and the correct image arrived on retinas of the eyes (LaValle, 2013).

The design proposed involves a number of extra, time consuming operations that introduce additional latency. The video should be compressed on the Server computer, streamed, decoded and played on the Client computer with a head-mounted display attached.

According to one source (Xinreality, 2015), the end-to-end latency of a wired Oculus Rift Development Kit 1 HMD is normally in a range of 50-60ms. According to Oculus VR Developer forums, it is at least 42ms (Oculus VR forums, 2013). A number of sources of different years, state that the latency under 100ms is generally not perceivable by human (Card, Robertson, Mackinlay, 1991.; Miller, 1968.; Myers, 1985).

Meanwhile, according to J. Carmack - CTO of Oculus VR, for virtual reality systems the end-to-end latency should not exceed 20ms: “The end-to-end latency of 20ms of a virtual reality system is generally imperceptible. The total latency of 50ms will feel responsive for the user, but still subtly lagging.” (Sterling, 2013).

If 20ms is considered as the target end-to-end latency of a VR system, the use of HMD like Oculus Rift DK1 would already introduce more latency than that, especially when used wirelessly. Steve LaValle, researcher at the Oculus VR, however argues, that the latency problem has been nearly resolved with modern prediction techniques: “During ordinary game play, even with some fast head motions, simple prediction techniques accurately predict 20 to 40ms” (LaValle, 2013). Can be expected, that a newer Oculus Rift HMDs with advanced prediction should perform noticeably better when used wirelessly comparing to Development Kit 1 or any other “slower” HMDs.

4. Implementation

This chapter describes how a prototype of the design proposed in the previous chapter was implemented - which hardware and software was utilized, what alternative options were considered and solutions found for the problems that occurred.

4.1 Prototype Software

Similar to the section 3.2 of the Solution Proposed chapter which described the overall system design, the following scheme (figure 4.1) demonstrates software proposed to run on both Client and Server computers in order to perform their tasks. Particular choice for each software is explained in the upcoming sections.

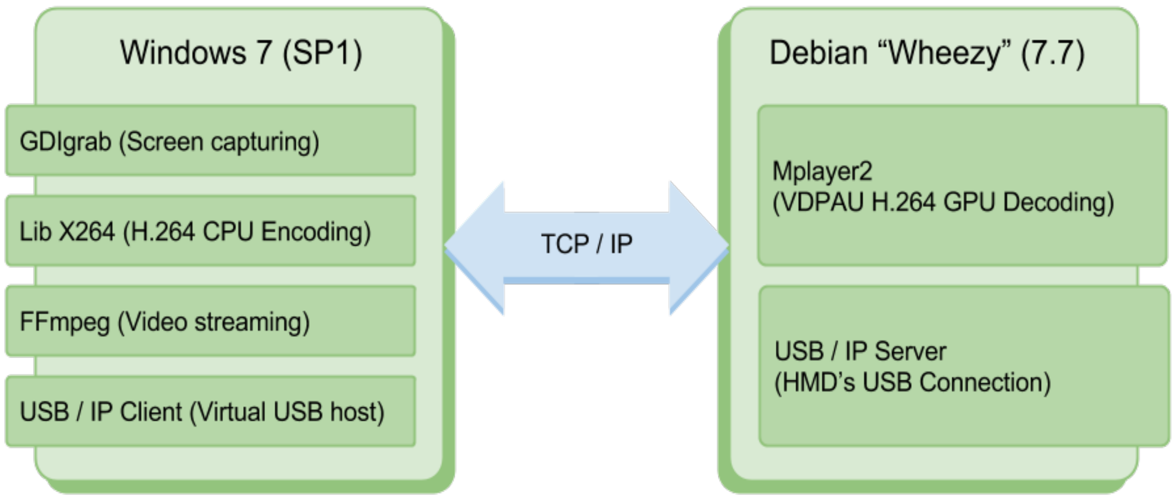

Windows 7 (SP1)
GDIgrab (Screen capturing)
Lib X264 (H.264 CPU Encoding)
FFmpeg (Video streaming)
USB / IP Client (Virtual USB host)
TCP / IP
Debian "Wheezy" (7.7)
Mplayer2
(VDPAU H.264 GPU Decoding)
USB / IP Server
(HMD's USB Connection)

Figure 4.1: Overview of the software for the Server (left) and Client (right) computers.

4.2 Server Software

Since most of the HMD-enabled applications today are only run under Windows operating systems (according to Oculus VR Share repository), the Server computer should run one of the Windows family OS. For the prototype development a Windows 7 Professional N with Service Pack 1 (64 bit version) was used, newer operating systems or another editions of Windows 7 can potentially be used, however compatibility was not tested.

4.2.1 Screen Capturing

There are various ways exist to perform software frame buffer capturing on the Server computer running Windows operating systems: via Graphics Device Interface (GDI), via DirectX and via Windows Media API (Palem, 2006).

While for some applications it might not make a big difference which one to use, it is better to select the one that already has an interface to work with the streaming software. As FFmpeg software was selected for video streaming (more in the following section 4.2.3 Video Streaming), there is no interface exist for use with Windows Media API and therefore either GDI or DirectX would be a better option.

The GDIgrab software that uses Graphics Device Interface was selected as it can be used directly from FFmpeg streaming software and as it allows to specify either the whole desktop screen for capturing or only a specific application window.

Among the alternative solutions a Screen Capture Recorder to Video (Screen Capture repository on GitHub) was tested which also can be used by FFmpeg. It is an open source software which works via DirectShow multimedia framework of the Microsoft DirectX SDK or Microsoft Windows SDK.

The major problem encountered with Screen Capture Recorder was a frame dropping, the application can drop a series of frames, especially when used with high frame rates and with limited computation resources available. This can sometimes also lead to the video stream being stopped or introduce a significant extra delay for the system.

Another, less important drawback of the Screen Capture Recorder is that some elements of the Windows operating system interface were not captured by it, like the "Start" button for instance. As GDIgrab performed without aforementioned drawbacks it was given a preference over the Screen Capture Recorder. It should be mentioned, that all the tests were performed with Windows Aero User Interface turned off for a better performance as suggested by a project description.

4.2.2 Video Encoding

As H.264 video encoding standard is widely supported today, there is a large number of software and hardware implementations available. One of them - Lib x264 was used, it is a free and open source software encoder developed by the VideoLAN organization.

Lib x264 is widely used by media companies in many projects including FFmpeg streaming software, which has it included in all builds by default and thus can be used immediately (CoreCodec, Inc., 2013). Compared to other software encoders it provides a number of features that are not yet supported by QuickTime, Nero Digital or MainConcept encoders. This allows to utilize Lib x264 in a wide range of use cases including real-time video encoding for low-latency streaming (Streaming Media, 2008.; Superuser community, 2011).

Lib x264 is well known due to numerous awards received and video codecs comparisons won (Vatolin, Kulikov & Arsaev, 2012). Also for using psycho-visual enhancements that aimed to increase the subjective video quality of the encoded video.

According to a user test, Lib x264 compared with technologies utilizing hardware capabilities like Nvidia NVENC and Intel Quick Sync, shows close levels of application's performance when used on the state of the technology hardware and capturing a game-play with a constant bit rate and 60 FPS (Scan Computers, 2014). It also delivers the best visual picture quality according to the test. Resulting frame rates, however depend on the hardware and encoding presents used, therefore other tests can show different results.

4.2.3 Video Streaming

As was previously mentioned, to perform the video streaming from the server computer FFmpeg project was chosen. FFmpeg is a free and open source video streaming and transcoding software offered under the GNU GPL license. It contains a number of libraries that can also be used for the development of transcoding, streaming and playback applications (FFmpeg, 2015).

FFmpeg was chosen as it is allows flexible stream configuration and works with many screen capturing devices and interfaces. It has support for a chosen GDIgrab screen capturing and Lib x264 video encoder. Also allows point-to-point streaming, i.e. does not requires a streaming server to run on the network. At the moment of writing, FFmpeg does not officially support Nvidia NVENC or Intel Quick Sync hardware H.264 encoders, however patches for Linux and Windows can be found in development and expected to be released soon (FFmpeg developers mailing list, 2014.; 'ffmpeg_libnvenc' repository, 2015).

Among the alternatives an Open Broadcasting Software (OBSproject) and FFsplit (ffsplit.com) were also considered. While Open Broadcasting Software (as of v.0.637 beta) allows to utilize Intel Quick Sync or Nvidia NVENC hardware encoders (or Lib x264 as well), it has a serious drawback - the minimum stream buffering time cannot be set under 60ms. This adds respective extra latency. It also lacks a point-to-point streaming option and therefore a streaming server should be set up, which potentially adds even more latency.

FFsplit streaming software is in fact a convenient interface to FFmpeg. It requires FFmpeg to be installed and allows using presents, save configurations, preview output and a number of other features that are commonly used by users doing Internet video streaming. As those features are not obligatory required for the developed prototype, the FFmpeg was given a preference.

4.2.4 Virtual USB Connection

In the sections 2.6 and 2.7 of the Related Work chapter an overview of the Certified Wireless USB standard and USB data rate usage test by Oculus Rift DK1 head-mounted display were conducted.

The test results demonstrated that the amount of data transferred does not exceed the 12 Mbit/s, i.e. USB 1.0/1.1 speeds. Since a 802.11n Wi-Fi network was proposed for use (Section 3.5 of the Solution Proposed chapter), there should be theoretically enough bandwidth to transfer both encoded video to the Client computer from Server and USB data of the head-mounted display from Client to Server computer. The 802.11n 5GHz Wi-Fi standard defines transfer speeds up to 600 Mbit/s when using multiple data streams (MIMO).

With this assumption, the USB I/O messages can be encapsulated into TCP/IP packets and transferred between the Server and Client computers while HMD will be connected to the Client computer USB port.

The USB/IP project (usbip.sourceforge.net) was used for creation of a such virtual USB connection over the wireless network. It is free and open source (GNU GPL) software that includes an application with a virtual USB device enumerator driver for Windows operating systems and a second application for Linux with kernel driver module.

The USB/IP project originally suggests a naming convention where Server is a computer that provides physical connection with a USB device. To avoid confusion, it is named a USB Host computer further on or also a Client computer as before. And the Server computer is the one that utilizes the actual remote USB device. Figure 4.2 below demonstrates the USB/IP software design (Hirofuchi, Kawai et al., 2005).

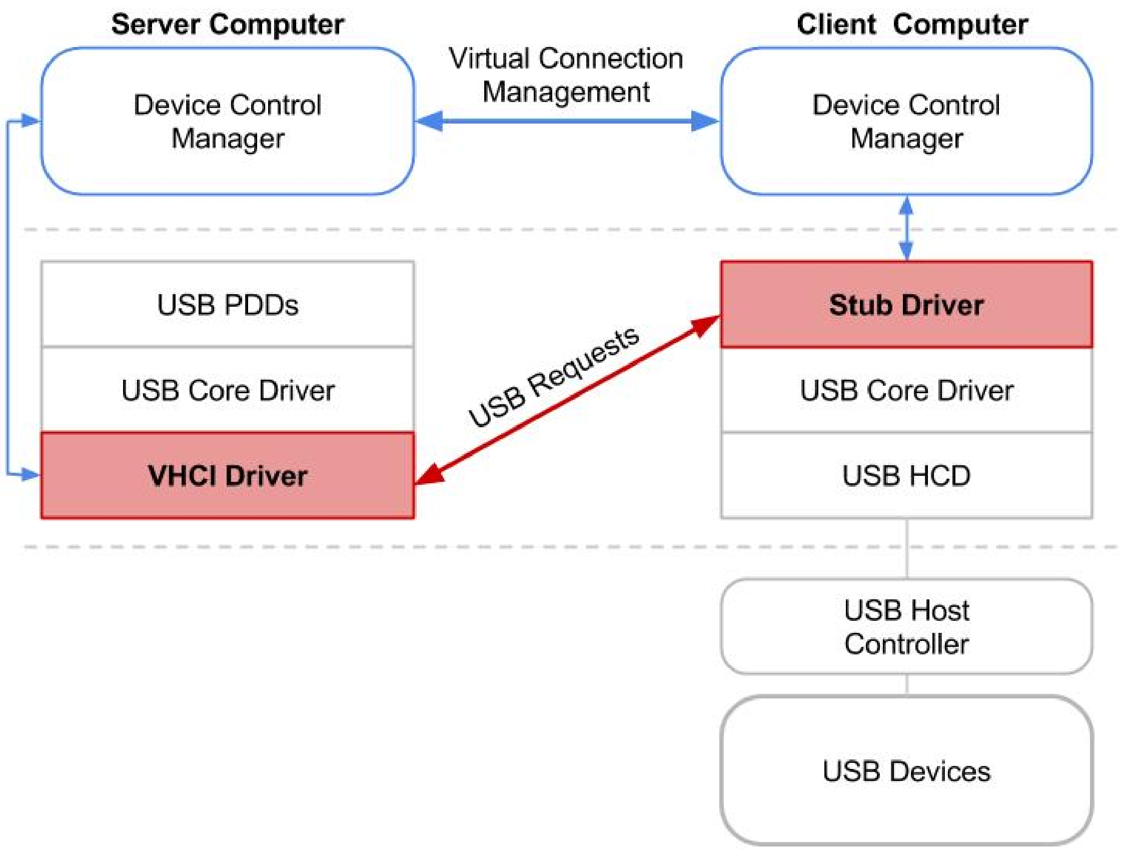


Figure 4.2: USB/IP project design overview (Hirofuchi, Kawai et al., 2005).

The USB device is physically attached to the Client computer with a stub driver that encapsulates the USB data into network packets payload and sends them to the Server computer running a Virtual Host Controller Interface (VHCI) driver. There are also two Device Control Manager applications that are used to establish and configure USB connections between the Client and a Server.

The USB/IP application and driver for Windows initially did not work as expected and therefore were rebuilt from the sources. The Windows VHCI driver source was modified to avoid Windows Stop Error (BSOD) on the disconnection of a remote Human Interface Device (HID) USB device. The managing application was rebuilt with no modifications done, as the USB/IP discussion forum suggested due to “usbip_recv_op_common” error (USB/IP Open Discussion, 2011).

The modified source and new builds can be found in the Appendix section (9.3 USB/IP for Windows). New driver and application were successfully tested on two computers running different versions of Windows 7 operating system (Professional N x64 with SP1 and Enterprise x64 with SP1).

As the driver was rebuilt, it wasn't digitally signed as the original one that is distributed. With the default Windows 7 security policy the non-signed drivers can still be installed, but cannot be used unless the system is running in testing mode. Therefore a testing mode was enabled (Techspot, 2009).

Among the alternatives to USB/IP a VirtualHere project (virtualhere.com) was considered. It is a proprietary, but free for personal use software serving the same purpose of connecting a USB device over the TCP/IP networks.

With a similar software architecture, the VirtualHere provides own graphical user interface for managing the connection with a USB device and automatic remote USB device discovery from the Server. It runs completely in user space on the Client computer, which makes it more convenient to setup and run initially.

The performance of VirtualHere for real-time use with head-mounted displays, however, can be rated unsatisfactory. Number of tests performed with two demo applications used before (“Alone in the Rift”, “Tuscany”) demonstrated lags and freezing of picture when HMD is being rotated with USB connection working over the wireless 802.11n network and a direct wired video interface connection with a Server computer. Described problems occurred only periodically, but with a frequency of up to several times per minute making it hard to use the setup. Same tests performed over the wired 1 Gbit/s network connection between the Server and Client computers demonstrated unacceptable performance of the head-mounted display in the similar way. More on testing follows in the Evaluation chapter.

4.3 Client Software

4.3.1 Client Operating System

As was previously mentioned in the Proposed Design chapter, the Client computer should receive, decode and play the video stream from the Server and transfer the USB data from the attached head-mounted display to the Server computer.

A Debian “Wheezy” Linux was chosen as operating system for the Client computer. The choice for OS is partially determined by the used prototype hardware (described in the upcoming sections of this chapter). Debian is being the third most popular Linux distribution at the moment of writing, and two leading distributions are in fact also based on Debian (Distrowatch, 2015).

Debian is one of the best community driven and supported Linux OS nowadays with a wide range of hardware architectures supported and numerous drivers and applications available from official and unofficial software repositories. Debian, as most Linux distributions, is free and open source, full set of included software licenses can be found on the official Debian online resources (debian.org).

4.3.2 Video Playing

To decode and play the video stream received from the Server computer an Mplayer2 - free and open source video player was chosen (not to be confused with Mplayer media player of the second version).

The core advantages of Mplayer2 are flexibility, extensive command-line user interface and support of a wide range of video output drivers as well as file and video stream formats (mplayer2.org).

The Mplayer2 was chosen as it allows fine-tuning and configuration of video files and streams played and also allows to utilize hardware capabilities for decoding. This is important, as the latency can be reduced by deactivation of all buffers and utilization of hardware-accelerated decoding.

4.3.3 Client USB Connection

To allow the remote use of the USB connection with HMD it is required to run the USB/IP project application and a kernel driver module on the Client computer.

The required package 'usbip' for all major hardware architectures can normally be found in “main” Debian software repositories and no special configuration for installation was required, the software needs only a few other packages to be installed.

4.4 Prototype Hardware

4.4.1 Server Hardware

The detailed specifications of the Server computer used for development and evaluation can be found in the Appendix section (9.1 Server Computer). There are a few special hardware requirements applicable for the Server computer:

1. Multi-core Intel x86_64 family CPU (Intel Core i5/i7 or AMD FX 8000+ series are recommended);
2. Wireless network adapter supporting 802.11n 5GHz 300 Mbit/s (or higher) Wi-Fi or wired 1000BASE (1 Gbit/s) network adapter for use with external wireless access point or router hardware;
3. Graphics card with (sufficient) capabilities for running HMD-enabled application;
4. Disk space and amount of RAM should be sufficient to run Windows operating system of a choice.

4.4.2 Client Hardware

Clarifying the general requirements, previously described in the Introduction chapter, the hardware requirements for the Client computer can be specified as follows:

1. Compact size, low weight and portable, preferably single-board design;
2. At least one digital video interface (HDMI or DVI) supporting target HMD native resolution and screen refresh rate;
3. At least one USB 1.1 interface to connect the HMD;
4. An embedded 802.11n 5GHz wireless network card or a suitable extension port;
5. Sufficient CPU and GPU capabilities to receive over network and process the H.264 encoded video stream with native HMD resolution and frame rate equal or higher than the native HMD refresh rate is;
6. Low power consumption for use with portable battery pack for at least three hours;

Here and further on Oculus Rift Development Kit 1 (DK1) is considered as the target HMD and it is used for development operations. The native DK1 screen resolution is 1280 x 800 and refresh rate is 60Hz. Although many head-mounted displays for VR applications have comparable specifications and interfaces and can also be used, some models may require different hardware to be utilized respectively.

Taking into account the requirement on the extra costs of the developed prototype a few of the single-board computers were tested for the role of a Client computer, including Raspberry Pi model B and a similar, yet more powerful Banana Pi.

While both provide GPU hardware-acceleration for H.264 encoding or decoding, only Banana Pi computer allowed to achieve acceptable level of system performance. Raspberry Pi computer failed to decode and play H.264

high-definition content with more than 30 frames per second rate, even when its GPU was operating at a doubled frequency (500 MHz), CPU being over-clocked by 57% and RAM by 25% with active cooling system installed.

The technical specifications of Banana Pi computer can be found in the table 4.1 below.

Model	Banana Pi (Allnet version)
CPU	Cortex-A7 (ARMv7 NEON, Dual Core, Allwinner A20 SoC), 1000 MHz
GPU	Mali400MP2 (Dual Core, Allwinner A20 SoC), 500 MHz
RAM	1 GB DDR3 (shared with GPU)
Main interfaces	USB 2.0 x2; HDMI; RCA video out; 3.5mm audio out; RJ45 LAN;
Dimensions (mm)	92 × 60 x 22
Weight (gr.)	48
DC power input (V)	5
Power consumption (Watt, max.)	6

Table 4.1: Banana Pi single-board computer technical specifications (bananapi.org).

According to the specifications, the Allwinner A20 system on a chip of Banana Pi is capable of real-time encoding or decoding H.264 (high-profile) video with Full HD (1920 x 1080) resolution at the rate of 60 frames per second (Allwinner Tech, 2015), which is sufficient for the Oculus Rift DK1 HMD.

The figure 4.3 below gives an overview of interfaces and board design of the Banana Pi computer:

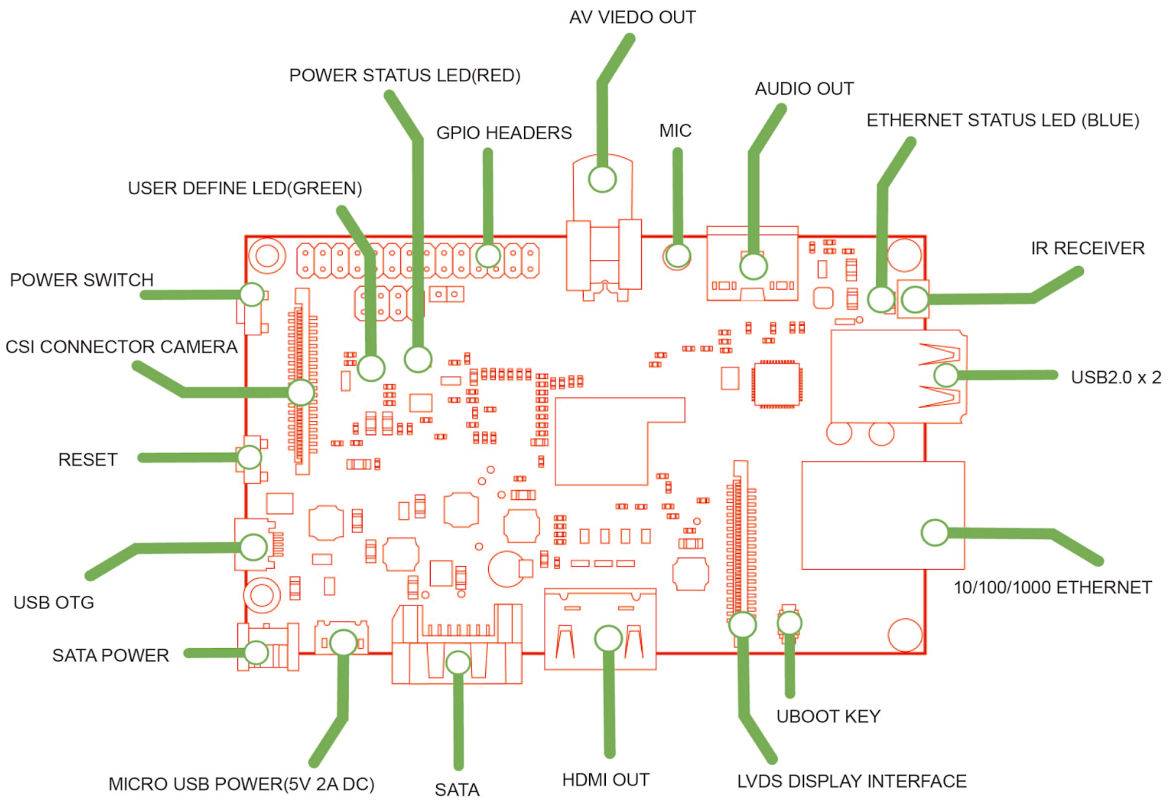


Figure 4.3: Banana Pi single-board computer interfaces overview (bananapi.org).

As can be seen, Banana Pi offers a wide range of interfaces including HDMI and two USB 2.0 ports that allow connecting a head-mounted display and one more device. The Banana Pi has no wireless network adapter embedded, therefore to provide a wireless connection an external dual band USB Wi-Fi adapter was used, its specifications can be found in the table 4.2 below.

Model	Asus USB-N53
Operating bands	2.4 GHz; 5 GHz
Connection speed (Mbit/s, max.)	300 (2.4 GHz); 300 (5 GHz)
Antennas	2x internal
Wi-Fi standards supported	802.11 a/b/g/n
Interface	USB 2.0
Dimensions (mm)	96 x 26 x 12
Weight (gr.)	18

Table 4.2: USB 2.0 Asus Wi-Fi adapter technical specifications (ASUSTeK, 2011).

To provide autonomous power supply of the Banana Pi and the HMD attached to it, a portable rechargeable battery with two USB A-type plugs was used. Since both Banana Pi and Oculus Rift DK1 HMD used require 5V DC input, a single power source can be used. The technical specifications for the used battery pack can be found in the table 4.3 below.

Model	EasyAcc Power Bank PB 10000
Capacity	10 000 mAh @3.7V / 37 Wh
Battery cell type	Lithium-Polymer
Charging input	5V, 1500 mA
Outputs	5V 1000 mA; 5V 2000 mA
Output connectors	2x USB A-type plug
Dimensions (mm)	140 x 73 x 19
Weight (gr.)	252

Table 4.3: Rechargeable battery technical specifications (Easyacc, Inc, 2013-2015).

To attach the Banana Pi to the battery a common micro USB cable was used. In order to attach the Oculus Rift HMD to the battery an adapter cable with 5.1mm barrel DC connector was used.

4.5 System Configuration

4.5.1 Server Configuration

During the screen capture on the Server computer the Windows Aero interface was disabled and GDIgrab maximum frame rate was set to 90 FPS.

The Lib x264 parameters were configured as follows in the table 4.4 below. Some of the options are in fact provided via FFmpeg interface, more detailed description is available on the Linux encoding website (Linux Encoding, n.d.).

Option	Meaning
-preset ultrafast	Enables configuration for faster encoding, but lower compression rate.
-tune zerolatency,fastdecode	Enables optimizations for faster encoding and decoding, to minimize added latency. (Disables any lookahead features).
-pix_fmt yuv420p	Specifies to use the yuv420p color space for better compression.
-qscale 1	Specifies the quality of the encoded video. With setting=1 best and visually lossless quality is achieved.
-me_method zero	Disables the motion estimation to minimize latency at the price of higher bit rate.
-g 4	Specifies that every forth frame is an I-frame. A trade off for lower latency and good compression rate.
-vsync drop	Specifies not to duplicate frames for keeping constant frame rate.
slices=20	Specifies the number of slices to be used in parallelized encoding.
intra-refresh=1	Enables Periodic Intra Refresh that can replace key frames by using a column of intra blocks that move across the video from one side to another. Keeps bit rate more constant and reduces the latency.
no-chroma-me=1	Disables “Chroma” subpixel refinement for lower latency.
sliced_threads=1	Optimization for faster encoding on multi-core CPUs, allows to run thread per slice (instead of thread per frame).

Table 4.4: Configuration settings used for Lib x264.

The H.264 encoding profile and level are selected automatically by the encoder depending on the parameters used. Here, a Constrained Baseline level 4.1 is selected.

FFmpeg options were configured as follows in the table 4.5 below.

Option	Meaning
-fflags +nobuffer	Reduces the latency by disabling the optional buffering.
-vcodec libx264	Specifies to use Lib x264 as the encoder.
-f mpegts tcp://192.168.123.5:12345	Specifies the transport stream muxer (mpegts); transport protocol (tcp); destination IP address and port (192.168.123.5:12345), i.e. those of the Client computer.

Table 4.5: Configuration settings used for FFmpeg.

As was previously specified in the section 4.2.4 Virtual USB Connection, the driver and the USB/IP application were rebuilt. To use the unsigned USB/IP driver, Windows 7 - operating system of the Server computer, was configured to run in a testing mode. It allows using drivers that were not digitally signed.

The USB/IP managing application needs an IP address of the Client computer and a bus ID to connect remotely to a head-mounted display. As the bus ID of the remote USB device can be obtained via scan function of the USB/IP software itself, a Python script was written that performs the scanning of the remote host by a specified IP address and connects if any attached USB device found.

To ease the start of all software on the Server computer, the same Python script was extended further and incorporated both USB/IP software and FFmpeg with Lib x264 and GDIgrab parameters. The script should be called with two arguments: IP address of the Client computer and the port that is used for incoming video stream. Both arguments are positional. Script can be found in the Appendix section (9.4 Automation on Server).

4.5.2 Client Configuration

As was already specified, a Debian “Wheezy” (v.7.7) Linux distribution for ARM architecture was chosen and installed on the Banana Pi computer with a minimum set of software packages.

In order to enable full compatibility with Banana Pi hardware, a custom kernel based on a stable mainstream version 3.4.90 and a driver needed to utilize Mali400 GPU H.264 encoding and decoding capabilities was installed. Development was done by an open source community (Sunxi Community, 2015).

As a desktop environment an LXDE (Lightweight X11 Desktop Environment) was used as being one of the fastest graphical environment solutions for Linux (lxde.org). The actual need for running an X-server (Window system display server) on Banana Pi was determined by the use of Video Decode and Presentation API for Unix (VDPAU), which is accessed by media players.

VDPAU interface allows utilizing Client computer's capabilities to decode H.264 video content. The 128Mb of the shared RAM of Banana Pi were dedicated for use by the GPU via the FEX configuration to ensure sufficient space for graphical desktop environment operation and high definition video playback. The average RAM usage when the OS is completely loaded (with LXDE) was about 52 Mb out of 874 Mb available in total.

While the wireless network was configured on the Client computer, a Bug #34872 in Linux Kernel (kernel.org) was encountered, which prevented normal use of the 5GHz bands. The Sunxi kernel was cross compiled from the sources with a new configuration, where Linux cfg80211 driver was built as a module which resolved the problem.

The Wi-Fi regulatory domain was after configured to meet the regulations in Germany, this allowed normal operation of wireless network adapter in both passive (managed) and active (master) modes in 5GHz Wi-Fi bands. The network was configured via "/etc/network/interfaces". The automatic power management for wireless network adapter was disabled to ensure stable Wi-Fi connection.

The network interface mode was also configured as "auto" and "managed" to allow a hot-plug of the USB network adapter and automatic connection to the "WirelessOculusRift" Wi-Fi network, where name comes from the used HMD model. The managed Wi-Fi operation mode defines that Banana Pi will act as a client and connect automatically to a network with "WirelessOculusRift" SSID and obtain an IP address via DHCP. To ensure that IP address always stays the same, a respective DHCP address reservation by network adapter MAC address can be configured.

To decode and play the video stream from Server computer an Mplayer2 was used. The VDPAU interface was configured to use installed Sunxi driver via environmental variable "VDPAU_DRIVER=sunxi", therefore hardware-acceleration is used for any supported type of content. Mplayer2 was configured respectively to utilize VDPAU. Other configuration options for Mplayer2 can be found in the table 4.6 below:

Option	Meaning
-nocache	Disables stream caching buffer.
-noidx	Skips rebuilding of index file, since a “file” is a video stream this option is used.
-nosound	Disables sound, so player does not seek a sound stream.
-nodouble	Disables double buffering to reduce the latency by one frame.
-benchmark	Records the stream information and enables player optimizations for faster video playback start.
-fs	Runs player in a full screen mode.
ffmpeg://tcp://0.0.0.0:12345?listen	Specifies the stream to play. “0.0.0.0” IP address stands for “localhost”; “12345” - port to listen; “tcp” defines transport protocol.

Table 4.6: Configuration settings used for Mplayer2.

To automate the video stream playback start with the start of the Client computer, a Python script that is launched on the start of the X-Server of the Debian OS was written. It launches the Mplayer2 with configuration as specified in “mplayer_config.txt” file located on the /boot/ system partition. In case the stream is being dropped or stopped, the Mplayer2 is restarted after a 5 seconds timeout. The configuration is being read on every player launch, this allows changing configuration on-the-fly and also from Windows operating system as /boot/ partition has file system accessible from any operating system (FAT). Script is available in the Appendix section (9.5 Automation on Client).

In a similar manner, the USB/IP software is launched automatically by a Python script with the system start (via init script added to /etc/init.d/) and reads the “hmd_usb_id.txt” file from /boot/. File has only a single line with USB device identification code, which allows to determine whether HMD USB port is attached and bind the device on connection for remote use on the Server computer. The USB ID should be defined once for HMD before the first start. Script can also be found in the Appendix section (9.5 Automation on Client).

To load the USB/IP kernel driver module automatically during the system start the “/etc/modules” file was modified. To enable the autostart of the USB/IP service a new shell script was created in /etc/init.d/ folder.

4.6 Implementation Summary

Summarizing the most important points of this chapter:

1. Server computer runs HMD-enabled application. GDIgrab with FFmpeg and Lib x264 software to capture, encode and stream the visual information from the Server's screen to the Client computer.
2. Server enables a software wireless access point if Wi-Fi network adapter is available or connects to an external wireless access point or router device via wired local area network.
3. Server connects to the USB of head-mounted display via USB/IP software with VHCI driver.
4. Client computer is running Debian OS on Banana Pi single-board ARM computer with external USB network adapter and all head-mounted display interfaces attached.

5. Client computer automatically connects to the wireless network on system start, launches Mplayer2 to listen to the incoming video stream and runs USB/IP software to bind to the head-mounted display USB.
6. Both Client computer and used head-mounted display are powered via an external rechargeable battery.

The developed prototype running "Tuscany" HMD demo application (Oculus VR, 2013) can be seen on the figure 4.4 below:

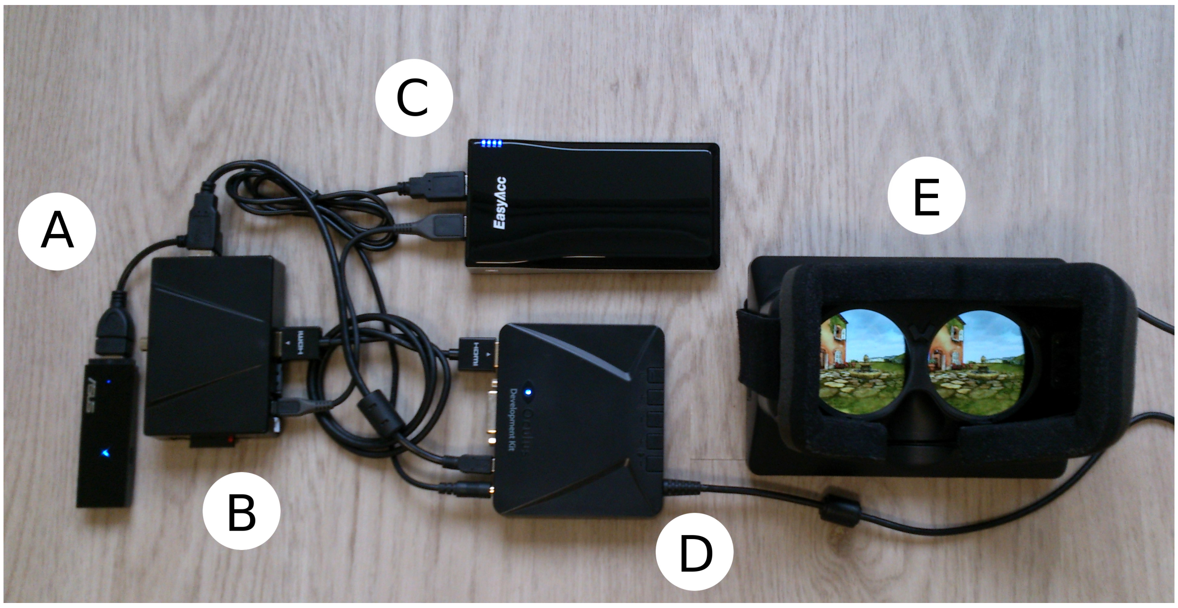


Figure 4.4: Wireless prototype running demo application. (A) USB Wi-Fi adapter; (B) Client computer with plastic case; (C) Battery pack; (D) HMD interfaces box; (E) HMD.

5. Evaluation

5.1 Evaluation Criteria

In the Introduction chapter a number of requirements was stated for the wireless version of a head-mounted display. The main objective was to design a system and to build a wireless version of the head-mounted display that would perform as possibly close as the wired version of the same HMD in the sense of visual quality, latency and usability from the user perspective. Requirements also apply on portability and pricing.

A number of measurable criteria was used to evaluate how similar the performance of the wireless prototype built comparing to the original wired head-mounted display. Those criteria are:

1. Additional (streaming) latency. Introduced by all the extra operations performed, including screen capturing, video compression, video streaming on the Server. Video receiving, decoding and playing on the Client computer. Wireless network communication and data transfer between Server and Client.
2. Frame rate and screen refresh rate. Although, the HMD-enabled applications can be rendered with frames per second rate of 100 or higher, most of the conventional (non-stereo) monitors deliver the vertical refresh rates of 60 Hz. The Oculus Rift DK1 HMD used for the prototype built also has the (vertical) refresh rate of 60 Hz. Thus, it is important that the video stream played on the Client computer with HMD attached had the frame rate at least as high as the display refresh rate to allow the user to perceive video smoothly (Bakaus, 2014).

3. Screen resolution. The screen resolution for the HMD attached to the Client should be native, i.e. the maximum supported with no scaling applied. For Oculus Rift DK1 it is 1280 x 800 pixels, which corresponds to the 16:10 screen aspect ratio.

There is also a number of secondary criteria that were evaluated as they reflect the fulfillment of the initial requirements and potentially the user acceptance:

1. Autonomous operation. The prototype should operate with battery for at least three hours within the Server's computer same room.
2. Added weight and size. Those are the values added by the use of extra hardware on the Client side.
3. Added costs. As all the software used to build the prototype is free, there are no extra costs added with it. The cost of operating system used on the Server as well as any third-party software used on the Server and the Server hardware are not counted as those costs also occur always when a wired version of the HMD is used. Therefore, only the costs added by extra hardware used for Client computer were estimated.

5.2 Streaming Latency Estimation

As the rendering latency of the Server computer should not be affected by the system designed as long as CPU is not running with full load, it is not going to be measured. The overall, end-to-end system latency was not measured since only the streaming latency is the one that was actually introduced by the system design and it allows to compare the performance of the prototype built with the wired HMD version.

The streaming latency was estimated according to the following methodology:

1. The HMD and a Client computer are placed near the conventional LCD monitor of the Server computer in a way that both HMD screen and LCD screen can be observed from one point of view.
2. The Oculus Rift DK1 HMD lenses are detached to provide a better screen view.
3. A Digital Single Lens Reflection (DSLR) camera (Nikon D80) is placed on a tripod in front to capture both HMD and LCD screens operation simultaneously.
4. The wireless HMD system is launched and a high precision timer application (Engineforce, 2013) is run on the Server computer.
5. The timer is started with a 1ms time update interval.
6. The DSLR set to manual mode and the shutter speed is configured to 1/1000 (1ms).
7. A picture of the timer application (visible on both Server's computer monitor and HMD) is taken.
8. Streaming latency is determined as the difference between the timer state on the Server's monitor and on the HMD screen.

Picture (figure 5.1) below demonstrates the setup and the process of estimation.

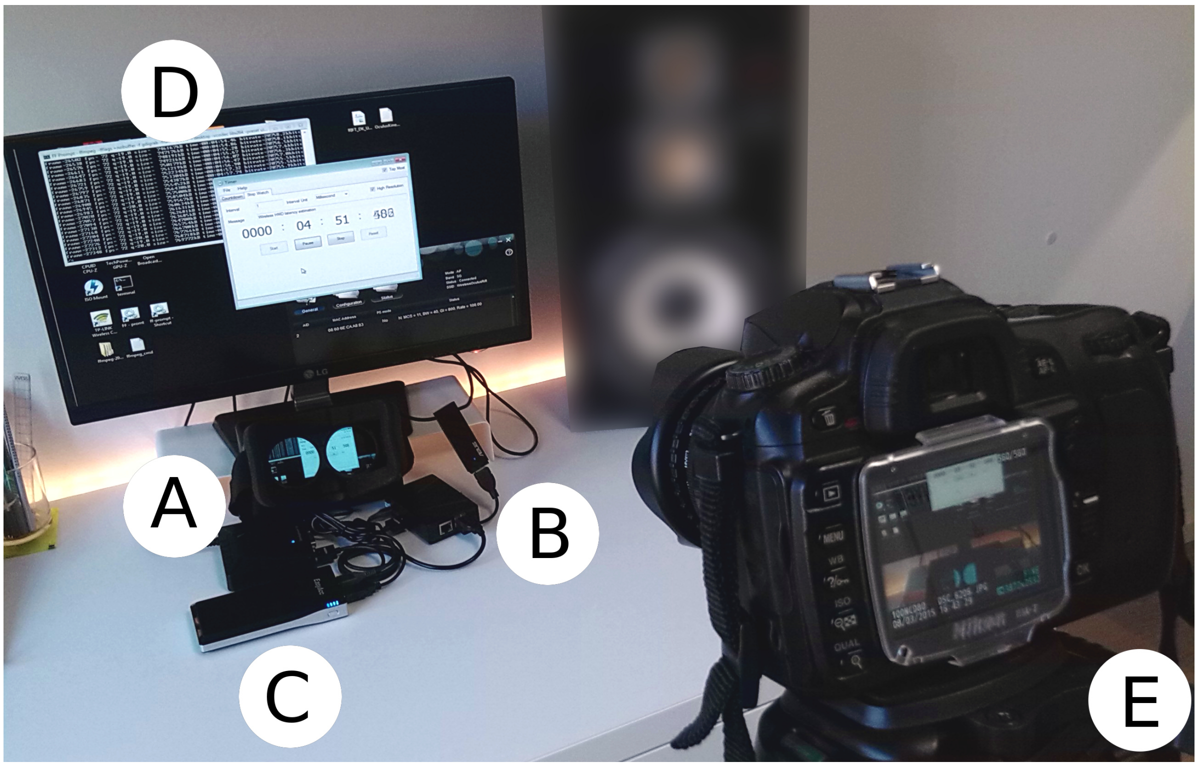


Figure 5.1: Setup for estimation of the added latency of a wireless prototype: (A) Oculus Rift DK1 HMD (lens detached) with connectors box; (B) Banana Pi Client computer with USB Wi-Fi adapter; (C) EasyAcc battery pack; (D) Server computer running timer application; (E) Nikon D80 DSLR camera on a tripod.

The wireless network for the test was created with a software access point on the Server computer. TP-LINK TL-WDN4200 Wireless Dual Band 802.11n compliant USB Adapter was used along with ASUS USB-N53 software. The specifications of the Server computer used can be found in the Appendix section (9.1 Server Computer). The distance between the Client computer and the Wi-Fi adapter of the Server computer was approximately 2 meters with no obstacles in-between.

The latency measurements according to the described methodology were performed 10 times in total, respective pictures can be found in the Appendix section (9.6 Latency Estimation). A 64-bit static FFmpeg build from

17.11.2014 (ffmpeg.zeranoe.com) was used during the estimation. Estimation results can be found in the table 5.1 below.

Test	Streaming latency (ms)	Amount of frames per second	Wireless connection speed (Mbit/s)
1	89	72	162
2	94	72	162
3	99	72	162
4	91	72	162
5	83	72	162
6	92	72	162
7	100	72	162
8	89	73	108
9	98	73	108
10	106	73	108
Avg.	94.1	72.3	145.8

Table 5.1: Prototype streaming latency estimation results.

The average latency is 94.1ms (Standard Deviation $\sigma = 6.707$). Can be seen that the frame rate during the tests stayed on the same level with an average of 72.3 FPS and the Wi-Fi connection speed noticeably varied ($\sigma = 26.084$) with an average of 145.8 Mbit/s.

When the connection was active, but no video or USB data was transferred, the speed remained on the levels of 270-300 Mbit/s, i.e. maximum possible with the used hardware. Such speed variation can be partially explained by the fact that a software access point was used. When wireless access point (TP-LINK Archer C5) was used during the development procedures, the connection speed remained more stable and stayed on the same levels.

Should be mentioned, that this method of latency estimation does not take into account a number of factors:

1. The added latency will vary with the change of frame rate. Running system with a higher frame rate should decrease the latency. The maximum possible frame rate depends on all components of a system, i.e. Server and Client computers capabilities and the Wi-Fi network used.
2. The input latency of the LCD monitor and of the HMD. While those values are not affected by the designed system, the latency estimation may show error. During the actual latency tests the monitor with an Advanced High Performance In-plane switching (AH-IPS) LCD matrix with 60 Hz refresh rate was used (LG IPS237L BN). The Oculus Rift DK1 HMD screen also supports a maximum refresh rate of 60 Hz and has an A-Si TFT/TN matrix (Innolux HJ070IA-02D). According to Display Lag Database (displaylag.com) the first has an input lag of 9ms, but the input latency of the used HMD remains unknown.
3. The wireless network connection state. The wireless signal strength, network quality as well as the transfer speed and network packet loss values vary depending on the location, hardware used, actual distance between the Server and Client computers, amount of other networks located on the same channel (Hummel et al., 2007). As transport done by the TCP/IP protocols, the user won't see visual artifacts, distortions or errors with the increase of packet loss, but the latency will increase with increasing percentage of packets being lost.

5.3 USB/IP Performance Estimation

To ensure that USB/IP software provides sufficient performance for wireless USB connection between the HMD and Server computer a number of tests was performed. The video interface of the head-mounted display was connected via an HDMI cable with Server computer during the test to avoid any confusion with the performance of a wireless video connection. The USB/IP software package of version 0.1.7-3 was used on the Client computer and latest source available in USB/IP repository (USB/IP Project, n.d.) was compiled for use on Server computer as mentioned earlier in section 4.2.4 of the Implementation chapter.

Two HMD-enabled demo applications were used for the testing: "Alone in the Rift" and "Tuscany". No human-visible delay was experienced even when abrupt head movements were performed. No issues, similar to those with VirtualHere software occurred, as was previously described in the section 4.2.4 Virtual USB Connection. The distance between the Client computer and the USB Wi-Fi adapter of the Server computer was approximately 2 meters during the tests with no obstacles in-between. Operation over longer distances did not introduce visible problems.

The paper about the USB/IP project also exposes results that USB/IP software can provide an effective data throughput of 28 MB/s (224 Mbit/s) over capable LAN (Hirofuchi, Kawai et al., 2005), which covers the needs of the HMD used (12 Mbit/s) with a significant reserve. While for the developed prototype the maximum connection speed is limited to 300 Mbit/s (the maximum capability of the USB Wi-Fi adapter used, ASUSTek, 2011) and maximum effective throughput can be significantly lower due to TCP/IP transport overhead and higher packet loss over Wi-Fi, it is still expected to meet the requirements in conditions similar to those during the tests.

For an extra comparison a test with a Belkin Wireless USB Hub F5U302 was performed. While device is not certified as a W-USB standard compliant, its technical specifications suggest an operating range up to 10 meters with up to full USB v.2.0 transfer speed (480 Mbit/s) (Belkin International Inc., 2007).

Oculus Rift DK1 HMD USB was connected to the Wireless Hub and previously used demo applications ("Alone in the Rift", "Tuscany") were run on the Server (i.e. Client computer was not used neither for wireless USB, nor for wireless video interface connection). The Wireless Hub with HMD attached was positioned approximately 2 meters away from its USB radio adapter attached to the Server computer.

The performance of the HMD connected over the wireless HUB can be rated as unacceptable. During a 5 minutes of demo application run the connection with wireless USB Hub was lost two times which resulted in error messages that HMD is not being connected. The reconnection took up to 15 seconds and the application remained unusable meanwhile. Similar to the performance of a previously tested VirtualHere software, the picture was freezing periodically with a frequency of up to several times per minute.

Can be concluded, that USB/IP provides sufficient performance for a used head-mounted display, unlike the tested VirtualHere software or a Belkin Wireless USB Hub.

5.4 Other Criteria Estimation

5.4.1 Refresh rate, frame rate, screen resolution

The head-mounted display used for development (Oculus Rift DK1) has a screen aspect ratio of 16:10, native resolution of 1280 x 800 pixels and a 60 Hz refresh rate (more specifications can be found in the Appendix section 9.7).

Many of the previously reviewed devices (WHDI and WiDi) in the Related Work chapter do not actually support the resolutions with 16:10 aspect ratio. This is confirmed by the users of Asus Wavi (MTBS forums, 2013), by the specification of Zinwell ZWD-2500 (Zinwell, 2009) and also during the tests of a Netgear PTV3000 device. All of those devices however support the 60 Hz screen refresh rate, which is also not always sufficient. For instance, head-mounted displays used in a previously described "Cutting the Cord: Wireless Mixed Reality Displays" paper, were in fact operating with a reduced refresh rate at 72Hz instead of maximum supported 85Hz (Csisinko & Kaufmann, 2011).

Banana Pi single-board computer that was used as a Client can be configured to operate with resolutions up to 1920 x 1200 pixels of both 16:9 and 16:10 aspect ratios with refresh rates up to 75Hz, thus being compatible with Oculus Rift Development Kit 1, Development Kit 2 (Full HD @75Hz) and potentially many other head-mounted displays. Running displays in non-native resolution normally results in stretching of the picture and running with a lower refresh rate results in screen being more blurry (Bakaus, 2014).

One more parameter that is important due to system design is a frame rate per second of the video played on the Client computer. To ensure the picture smoothness, the video frame rate should correspond to the screen refresh rate or be higher (Bakaus, 2014). This also applies to the rendering frame rate of the HMD-enabled application running on the Server computer.

As previously described in the section 4.5 System Configuration, the maximum rate was configured to 90 frames per second, which is 50% higher than the screen refresh rate of the used head-mounted display. Number of tests demonstrated that the actual frame rate with the chosen software configuration on the Server computer varies. A drop of frame rate can be observed with the rapid change occurring between the series of pictures, therefore a higher average rate is needed to ensure that FPS does not go down under the 60 frames.

The reason for frame drop is the used combination of Lib x264 parameters, as frame duplication is prohibited (by the “vsync drop” setting) and “intra-refresh” is enabled at the same time. With intra-refresh setting the key frame is “spread” over several frames and the image is “refreshed” by a wave of blocks that moves across the video from one side to another (Diary of an x264 Developer, 2010). This setting reduces the latency, but also makes the bit rate more constant. In combination with rapid picture change, where bit rate increases, this results in temporal frame drop, this can be observed for instance, when the HMD-enabled application is started or application's scene is changed.

5.4.2 Battery operation, dimensions, weight, costs

To estimate the autonomous battery operation time of the prototype developed a single test was performed. The system was started and a “Tuscany” (Oculus VR Inc., 2013) HMD-enabled demo application was run. Total running time before shutdown was 3 hours and 54 minutes. The running time may noticeably vary depending on the brightness setting of the HMD.

To estimate the portability of the prototype the added weight and dimensions of all components were measured. Results can be found in the table 5.2 below. The components utilized can fit altogether with the interfaces box of the Oculus Rift DK1 into a small belt bag.

Component	Length (mm)	Width (mm)	Height (mm)	Weight (g. ±1)
Client computer (Banana Pi with plastic case, SD card)	96	64	27	79
Wi-Fi adapter for the Client computer (ASUS USB-N53 with extension cable)	92	24	12	30
Battery pack (EasyAcc PB 10000)	140	73	18	252

Table 5.2: Weight and dimension of the prototype components.

Total weight added by the wearable components is 361 g. - less, than the weight of the used HMD (380 g.). The costs of the utilized components of the developed prototype at the moment of writing are collected in the table 5.3 below. Amazon.de on-line marketplace was used for price references.

Component	Price, incl. Shipping. (EUR)
Banana Pi single-board computer + SD card	37,22
Plastic case for Banana Pi	9,29
Wi-Fi Adapter ASUS USB-N53	24,95
Battery Pack EasyAcc PB 10000	21,99
Cables (HMDI, micro USB, USB to 5.4mm barrel DC)	Approx. 10
Total:	103,45

Table 5.3: Prices of the prototype components (Amazon.de).

5.5 Evaluation Results

The prototype built was evaluated by a number of criteria that allow to make conclusion, if the suggested system design can be used and if any limitations are applicable. Discussion of the results and conclusions follow in the next chapter. The summary of the Evaluation chapter can be found below:

1. The added (streaming) latency introduced by the wireless connection of the head-mounted display was estimated ten times in total and on average it is 94.1 ms. ($\sigma = 6.707$).
2. The performance of USB/IP software was compared to a similar software solution VituallyHere and also with a hardware solution (Wireless USB Hub, Belkin F5U302). USB/IP solution operates better than any of those tested.
3. The screen refresh rate and resolution of the wireless head-mounted display correspond to its wired version and its capabilities. The frame rate of the video stream played on the Client computer stays higher than the screen refresh rate (60) of the head-mounted display.
4. The operation time of the prototype was 3 hours and 54 minutes during the test with a 37Wh battery.
5. Weight and dimensions of the prototype components were estimated. The prototype can easily fit into a small belt bag.
6. The price of the components used to build a prototype at the moment of writing is approximately 103 EUR.

6. Summary

In this thesis a system that allows using wired head-mounted displays as wireless was designed and a prototype according to that design was developed. Number of requirements were taken into consideration, including the following:

- HMD should operate autonomously with no wires connected to the computer within the same room;
- The wireless HMD performance should resemble the wired version as close as possible;
- The wireless HMD should be light and portable, with no big bags or backpacks needed for its use;
- The costs for extra hardware and software utilized with HMD should not exceed 100 EUR.

The prototype was after evaluated by a number of parameters that allow to say whether the requirements were satisfied or not and what kind of limitations apply.

Before the system was designed, an overview of existing wireless head-mounted display solutions was done. In most cases the WHDI devices were used to connect the HMD wirelessly to the computer running an HMD-enabled application. The common drawbacks of those solutions are: relatively high price of the hardware; bulky size and noticeable weight; only partial compatibility with the HMDs used.

The system design was therefore intended to overcome those problems and was based on the commonly available technologies nowadays. The review of the capabilities of modern wireless networks and data rates of high-definition compressed and non-compressed digital video signals was done. This basically led to the suggested approach with a transfer of a compressed video that can played on the small, portable, low-cost computer with HMD attached to it.

The primary motivation of this work was the fact that solutions used in projects that require wireless head-mounted displays are often not well accepted by the users and directly affect the results of experiments. A common wireless HMD solution in those projects would be a “laptop in backpack” with HMD attached. While it might seem as a good and simple approach, a number of use cases, analyzed in Motivation section, clearly demonstrate drawbacks of such approach.

6.1 Limitations

The major drawback of the designed system and of a prototype developed is obviously the extra latency added. While not all of a today's wired head-mounted displays are actually good enough for Virtual Reality applications due to high latency, their wireless use with the developed system would only impair their performance. Therefore, developed system in its current state is not well-suited for use in VR, especially if application requires user to perform rapid head movements.

This judgment is based on the fact that the end-to-end latency of the VR system is recommended to be below 20ms (Sterling, 2013), which is several times less than the latency added (avg. 94.1ms for the developed prototype) and in fact also at least two times less than the end-to-end latency of the systems that use a wired HMD like Oculus Rift DK1 (42-60ms). User studies with various VR and AR applications would have allowed to make a better evaluation of the developed system, as user's perception of the performance may be different.

Another limitation comes from the fact that the developed prototype was only tested with one head-mounted display - Development Kit 1 from Oculus VR. Different HMDs may require additional customization and respective configuration to be done. Also, some HMDs may require a 3D display modes and a higher screen refresh rates (80-120Hz) which are not supported by the Client computer (Banana Pi) used for prototype development, therefore different hardware should be used for such cases.

Finally, the designed system has limited scalability for use in collaborative environments. This limitation is in fact determined by the capabilities of the used wireless network (5GHz 802.11n). Although several networks can coexist on one Wi-Fi channel, the performance may be significantly affected in such case. With 20 channels available in most countries in Europe (radio-electronics.com), the use of more than 20 similar prototypes/systems most likely won't be possible within one physical location. Should be mentioned, that limitations like a maximum power output also apply for some channels depending on the local regulations. This may affect the performance in certain circumstances and limit the amount of usable channels even further. Similar limitation, however, applies to any wireless standard that relies on open, unlicensed frequencies including WHDI.

6.2 Future work

With the suggested design concept the system can still be significantly improved with a use of better hardware, new software and optimizations. The Banana Pi Client computer used in this project can be replaced by a similar, but more powerful solution like Odroid C-1 single-board computer. Combined with real-time operating system this may noticeably reduce the video decoding time. And specialized H.264 decoding hardware solution would be even a better option.

The encoding of the video on the Server side can also be performed by the GPU with the use of technologies like Nvidia NVENC with higher frame rates. The respective FFmpeg patches were already in development at the time of writing. This would allow to decrease the encoding time on the Server.

The network communication can be optimized on the levels of operating system and also on the streaming software levels. The network buffers can be reduced in size and encoding can be performed with a constant bit rate, so that data would “fit” better into the network packets payload.

6.3 Conclusions

While the prototype developed may be not a very good option for virtual reality applications, it proofs the concept of the distributed system design where visual information is compressed on the computer rendering HMD-enabled application and transferred for a playback to a portable, low-cost computer with the head-mounted display attached, over a wireless network.

Although, no user studies or tests with other models of head-mounted displays were conducted, the current performance of the system might be sufficient for some applications. One of the possible examples is the augmented reality applications where extra latency estimated in the order of tens or hundreds of milliseconds does not affect the user performance.

In the previously described in Motivation section clinical evaluation, the monocular HMDs were used to superimpose patient's vital signs over anesthesiologist's field of view (Liu, Jenkins et al., 2010). In the paper “Clinical Implementation of a Head-Mounted Display of Patient Vital Signs” (Liu, Jenkins, & Sanderson, 2009) the details on the system implementation are given. It was stated that the system operated with a “clinically insignificant delay of 1 - 2 seconds” (Liu, Jenkins, & Sanderson, 2009, p.4.).

In applications like the one described, where visual information is mostly represented by numbers and small delay does not make it outdated or inappropriate, prototype developed in this thesis can presumably be used in its existing state. For virtual reality applications, as previously mentioned, further work oriented towards latency reduction should be done. The use of “faster” HMDs would also help to reduce the end-to-end latency and make solution suitable for wider range of applications.

Considering the other requirements, the prototype developed has a significantly smaller wearable weight (only 361 g.) and dimensions when compared to any of the solutions exposed in the Motivation (1.1) or Existing Wireless HMD Solutions (2.3) sections, except for one – Sensics Wireless Video Link. The battery operation time of the prototype is almost 4 hours with the used 37Wh rechargeable battery, which approximately corresponds to the battery operation time of an average laptop similar to those used in the reviewed solutions.

Finally, the extra costs of the developed prototype (not including the price of the wired HMD itself) are only a bit higher than 100 EUR, which makes it undoubtedly the most affordable solution among any of those reviewed.

7. Acknowledgements

I would like to thank my supervisors, Dr. Gabriel Zachmann (Computer Graphics and Virtual Reality Research Group, University of Bremen) and Dr. Marc Herrlich (Digital Media Research Group, University of Bremen) for their advice and support as well as for the Oculus Rift Development Kit 1 head-mounted display provided for the development of a wireless HMD prototype.

8. References

Aliexpress online marketplace (2015). Zinwell ZWD WHDI. Retrieved March 17, 2015, from http://www.aliexpress.com/item/ZINWELL-Wireless-Video-Transmitter-1080p-Video-Transmitter-Receiver-ZWD-2822-Wireless-Home-Digital-Interface-Free-Shipping/1472484088.html

Allwinner Technology (2015). Allwinner A20 SoC Features. Retrieved from http://www.allwinnertech.com/en/clq/processora/A20.html

Amazon.de online marketplace (2014). Asus Wavi WHDI. Retrieved from http://www.amazon.de/Asus-Wireless-Audio-Video-Interaction/dp/B004W7GZWO/ref=pd_sxp_f_pt

Amazon.de online marketplace (2015). Netgear X6 WiFi Router. Retrieved from http://www.amazon.de/Netgear-Nighthawk-R8000-100PES-Wireless-Beamforming/dp/B00SWEEYJ4/ref=sr_1_1?ie=UTF8&qid=1426552510&sr=8-1&keywords=netgear+x6

Anthony, S. (2013, September 4). HDMI 2.0 released. *Extremetech*. Retrieved from http://www.extremetech.com/computing/165639-hdmi-2-0-released-18gbps-of-bandwidth-allowing-for-4k-60-fps-32-audio-channels

ASUSTeK Computer, Inc. (n.d.). ASUS WAVI delivers 2-way HD PC content streaming. Retrieved March 17, 2015, from http://www.asus.com/Multimedia/WAVI/

ASUSTeK Computer, Inc. (2011). Asus USB-N53 Wireless Dual-band Adapter. Retrieved from http://www.asus.com/Networking/USBN53/

Bakaus, P. (2014, May) The Illusion of Motion. Retrieved from http://paulbakaus.com/tutorials/performance/the-illusion-of-motion/

bananapi.org. (n.d.). Banana Pi Single-board computer. Retrieved March 17, 2015, from http://www.bananapi.org/p/product.html

Belikin International, Inc. (2007). Belkin Wireless USB Hub F5U302. Retrieved from http://www.belkin.com/support/dl/p75427_f5u302_man%204-07.pdf

Bell Helicopter Textron Inc. (2014). Bell Helicopter History. Retrieved from http://www.bellhelicopter.com/Company/AboutBell/History/History.html

Birkfellner, W., Figl, M., Huber, K., Watzinger, F., Wanschitz, F., Hummel, J., Hanel, R., Greimel, W., Homolka, P., Ewers, R., & Bergmann, H. (2002, August). A Head-Mounted Operating Binocular for Augmented Reality Visualization in Medicine—Design and Initial Evaluation. *IEEE Transactions on Medical Imaging, 21(8)*, pp. 991-997.

Bott, E. (2014, July 24). Projecting your PC or mobile device with Miracast: How well does it work? Retrieved from http://www.zdnet.com/article/projecting-your-pc-or-mobile-device-with-miracast-how-well-does-it-work/

Brooks, F. P. (1999, November). What's Real About Virtual Reality? *Computer Graphics and Applications, IEEE 19(6)*, pp. 16-27. doi:10.1109/38.799723

Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991, April 28). The information visualizer: An information workspace. Proc. ACM CHI'91 Conf., pp. 181-188.

Carl Zeiss (2015). Cinemizer OLED Product Information. Retrieved March 17, 2015, from http://www.zeiss.com/cinemizer-oled/en_de/product-information.html

Chacos, B. (2014, September 5). Final Oculus Rift pricing, hardware teased as Gear VR reveals Oculus-ready interface. *PCWORLD*. Retrieved from http://www.pcworld.com/article/2602906/final-oculus-rift-pricing-hardware-teased-as-gear-vr-reveals-oculus-ready-interface.html

Compaq, Intel, Microsoft, NEC. (1998). Universal Serial Bus Specification 1.1. Retrieved March 17, 2015, from http://mprolab.teipir.gr/vivlio80X86/usb11.pdf

CoreCodec, Inc. (2013). Lib x264 Adopters, x264Licensing. Retrieved from http://x264licensing.com/adopters

Csisinko, M., & Kaufmann, H. (2011, Apri 6). Cutting the Cord: Wireless Mixed Reality Displays. Proceedings of Virtual Reality International Conference (VRIC 2011), Laval, France.

debian.org. (n.d.). A free operating system (OS) for computer. Retrieved from https://www.debian.org/

Desai, P. R., Desai, P. N., Ajmera, K. D., & Mehta, K. (2014, July 4). A Review Paper on Oculus Rift - a Virtual Reality Headset. *International Journal of Engineering Trends and Technology (IJETT), 13(4)*, pp. 175-179.

Denke, K. (2010, March 10). HDMI Cable Speed & Features Explained. *Audioholics*. Retrieved from http://www.audioholics.com/audio-video-cables/hdmi-cable-speed

Diary of An x264 Developer. (2010). x264: the best low-latency video streaming platform in the world. Retrieved from http://x264dev.multimedia.cx/archives/249

Digital Display Working Group. (1999). Digital Visual Interface 1.0. Retrieved from http://www.cs.unc.edu/~stc/FAQs/Video/dvi_spec-V1_0.pdf

Display Lag Database. (n.d.). Retrieved March 17, 2015, from http://www.displaylag.com/display-database/

Distrowatch.com. (2015). Top Ten Linux Distributions. Retrieved from http://distrowatch.com/dwres.php?resource=major

Duarte, M., Sabharwal, A., Aggarwal, V., Jana, R., Ramakrishnan, K., Rice, C., & Shankaranarayanan, N. K. (2012). Design and Characterization of a Full-duplex Multi-antenna System for WiFi networks. Department of Electrical and Computer Engineering, Rice University. AT&T Labs-Research, Florham Park, NJ 07932. Retrieved from http://arxiv.org/pdf/1210.1639.pdf

easyacc.com, Inc. (2013-2015). EasyAcc Power Bank. Retrieved from http://www.easyacc.com/power-bank/p99-easyacc-xtra-12000mah-power-bank-with-4-usb-ports.html

Engineforce. (2013, October 7). Free Countdown Timer and Stopwatch Timer. A high resolution Countdown Timer and Stop Watch in .NET. Available from http://sourceforge.net/projects/countdowntimer/

Goradia, I., Doshi, J., & Kurup L. (2014, September 2). A Review Paper on Oculus Rift & Project Morpheus. *International Journal of Current Engineering and Technology, 4(5)*, pp. 3196-3200.

FFmpeg. (2015). FFmpeg multimedia framework. Retrieved from https://www.ffmpeg.org/about.html

FFmpeg developers mailing list. (2014, December). [FFmpeg-devel] ffmpeg nvenc. Retrieved March 17, 2015, from https://ffmpeg.org/pipermail/ffmpeg-devel/2014-December/166745.html

ffmpeg_libnvenc. (2015). NVENC Support for FFmpeg. Available from https://github.com/agathah/ffmpeg_libnvenc

FFSplit. (n.d.). A freeware program that allows to capture or record desktop. Retrieved from http://www.ffsplit.com/

Fleishman, G. (2008, November 5). Another blow for UWB: Intel drops ultrawideband development. *Arstechnica*. Retrieved from http://arstechnica.com/uncategorized/2008/11/another-blow-for-uwb-intel-drops-ultrawideband-development/

Forret, P. (n.d.). Video bitrate calculator. Available from http://web.forret.com/tools/video_fps.asp

Friedrich, W. (2003). ARVIKA - Augmented Reality for Development, Production and Service. Siemens AG, Automation and Drives, Advanced Technologies and Standards.

Geier, E. (2014, September 30). Tips for Assigning Wi-Fi Channels. Retrieved from http://www.windowsnetworking.com/articles-tutorials/wireless-networking/tips-assigning-wi-fi-channels.html

Halldal, M. F. (2012). Exploring computational capabilities of GPUs using H.264 prediction algorithms. Retrieved from http://home.ifi.uio.no/paalh/students/MagnusHalldal.pdf

Hermans, A. (2012). H.264/MPEG-4 Advanced Video Coding. Retrieved from http://tcs.rwth-aachen.de/lehre/Komprimierung/SS2012/ausarbeitungen/H264-MPEG4.pdf

Hirofuchi, T., Kawai, E., Fujikawa, K., & Sunahara, H. (2005). USB/IP - a Peripheral Bus Extension for Device Sharing over IP Network. FREENIX Track: 2005 USENIX Annual Technical Conference, pp. 47-60.

Hummel, K., Adrowitzer, A., Hlavacs, H. (2007). *Measurements of IEEE 802.11g-Based Ad-Hoc Networks in Motion. Wireless Systems and Mobility in Next Generation Internet. Lecture Notes in Computer Science, 4396*, pp. 29-42.

IEEE. (1997-2014). IEEE 802.11 Standards. Retrieved from http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm

Ihab, A. (2014, February 19). Introducing the Video Coding Engine (VCE). Retrieved from http://developer.amd.com/community/blog/2014/02/19/introducing-video-coding-engine-vce/

Intel. (2011). Intel Wireless Display, A visibly smarter experience. Product brief. Retrieved March 17, 2015, from http://www.intel.com/Assets/PDF/prodbrief/323116.pdf?wapkw=%28widi%29

Intel. (2013). Intel Quick Sync Video. New Microarchitecture for 4th Gen Intel Core Processor Platforms. Retrieved from http://www.intel.de/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf

Intel. (2015). Intel Wireless Display and Pro Wireless Pro. Retrieved March 17, 2015, from http://www.intel.com/content/www/us/en/architecture-and-technology/intel-wireless-display.html

kernel.org. (n.d.). The Linux Kernel Archives. Retrieved March 17, 2015, from https://bugzilla.kernel.org/show_bug.cgi?id=34872

Kzero Worldwide. (2014). VR HMD and Input System Device Revenue Forecasts 2014 – 2018. Retrieved from http://www.kzero.co.uk/blog/vr-hmd-and-input-system-device-revenue-forecasts-2014-2018/

LaValle, S. (2013, July 12). The Latent Power of Prediction. Retrieved from https://www.oculus.com/blog/the-latent-power-of-prediction/

Lib x264. (n.d.). Free software library and application for encoding video streams into the H.264/MPEG-4 AVC format. Retrieved from http://www.videolan.org/developers/x264.html

Linux Encoding. (n.d.). x264 FFmpeg Options Guide. Retrieved March 17, 2015, from https://sites.google.com/site/linuxencoding/x264-ffmpeg-mapping

Liu, D., Jenkins, S. A., Sanderson, P. M., (2009, September 4). Clinical Implementation of a Head-Mounted Display of Patient Vital Signs. International Symposium on Wearable Computers, pp. 47-54.

Liu, D., Jenkins, S. A., Sanderson, P. M., Perry, F., & Russell, W. J. (2010, April). Monitoring with Head-Mounted Displays in General Anesthesia: A Clinical Evaluation in the Operating Room. *Society for Technology in Anesthesia, 110(4)*, 1032–1038. doi:10.1213/ANE.0b013e3181d3e647

Lowood, H. E. (2015). Virtual reality. In *Encyclopædia Britannica.* Retrieved from http://www.britannica.com/EBchecked/topic/630181/virtual-reality-VR/253103/Early-work

LXDE. (n.d.). Lightweight X11 Desktop Environment. Available from http://lxde.org/

Markets and Markets. (2014). Head-Mounted Display Market by Products, Components, Applications & Geography - Global Analysis and Forecast to 2020. Retrieved from http://www.marketsandmarkets.com/Market-Reports/head-mounted-display-hmd-market-729.html

McCracken, H. (2013, April 12). A Talk with Computer Graphics Pioneer Ivan Sutherland. *Tecnologizer at Time Inc.* Retrieved from http://techland.time.com/2013/04/12/a-talk-with-computer-graphics-pioneer-ivan-sutherland/

Meant To Be Seen 3D [Msg 2] (2013, June 13). Wireless Rift DIY Guide. Message posted to http://www.mtbs3d.com/phpbb/viewtopic.php?f=140&t=17710&start=40

Miller, R. B. (1968). Response time in man-computer conversational transactions. Proc. AFIPS Fall Joint Computer Conference, 33, pp. 267-277.

Mohapatra, S. (2014). Detail overview of NVENC encoder API. GPU Technology Conference. Retrieved from http://on-demand.gputechconf.com/gtc/2014/presentations/S4654-detailed-overview-nvenc-encoder-api.pdf

Mohler, B. J., Campos, J. L., Weyel, M. B., & Bulthoff H. H. (2007). Gait parameters while walking in a head-mounted display virtual environment and the real world. IPT-EGVE Symposium, pp. 1–4.

MPEG - Moving Picture Experts Group. (n.d.). A working group of ISO/IEC. Retrieved from http://mpeg.chiariglione.org/

Mplayer2. (n.d.). Media player for Linux. Retrieved from https://launchpad.net/mplayer2

Myers, B. A. (1985, April). The importance of percent-done progress indicators for computer-human interfaces. Proc. ACM CHI'85 Conf., pp. 11-17.

Neo, senso, node and core: A new HMD and interactive device by Light&Shadows (2014), *it3D Magazine*, *(01)*, 45-46.

Netgear, Inc. (1996-2015) AC3200 Nighthawk X6 Tri-Band WiFi Router product description. Retrieved from http://www.netgear.com/home/products/networking/wifi-routers/R8000.aspx

Netgear, Inc. (1996-2015). Netgear Push2TV PTV3000 product description. Retrieved March 17, 2015, from http://www.netgear.com/home/products/connected-entertainment/wireless-display-adapters/PTV3000.aspx

Ngo, D. (2010) Home devices to connect wirelessly via WHDI. *Cnet*. Retrieved March 17, 2015, from http://www.cnet.com/news/whdi-2-0-to-coexist-with-wi-fi/

Nvidia. (2015). Nvidia Shield. Retrieved from http://shield.nvidia.com/

OBS Project. (n.d.). Open Broadcaster Software. Free, open source software for live streaming and recording. Retrieved from https://obsproject.com/

Oculus VR, Inc. (2013). Oculus Rift Development Kit 1 Instruction Manual v1.1. Retrieved from http://static.oculus.com/sdk-downloads/documents/Oculus_Rift_Development_Kit_Instruction_Manual.pdf

Oculus VR Share. (n.d.). HMD-enabled applications sharing platform. Retrieved from https://share.oculus.com/category/all

Oculus VR, Inc. (2013). Alone in the Rift Demo. Available from https://forums.oculus.com/viewtopic.php?t=1539

Oculus VR, Inc. (2013). Oculus Tuscany Demo. Available from https://share.oculus.com/app/oculus-tuscany-demo

Oled-Info. (2006). eMagin reduces price of the Z800 3D Visor to 549$. Retrieved from http://www.oled-info.com/emagin/emagin_reduces_price_of_the_z800_3d_visor_to_549

Oxford, A. (2011, August 19). What happened to Wireless USB & HDMI? - The Ultra Wideband tech we're still waiting for. *Techradar*. Retrieved from http://www.techradar.com/news/computing/whatever-happened-to-wireless-usb-hdmi-994212

Paine, S. (2014, February 19) Miracast Latency Test ASUS Vivotab Note 8 + Actiontec. Retrieved from https://www.youtube.com/watch?v=zrPMAEosnGo

Palem, G. (2006, September 19). Various methods for capturing the screen. Retrieved from http://www.codeproject.com/Articles/5051/Various-methods-for-capturing-the-screen

Perfect prediction, Rift improvement idea [Msg 8] (2013, November 27). Oculus VR Development forums. Message posted to https://forums.oculus.com/viewtopic.php?t=5018

Persistence Market Research (2014). Rising Popularity of the Global Gaming Market. Retrieved from http://www.persistencemarketresearch.com/article/global-gaming-market-size.asp

Poole, I. (n.d.). Wi-Fi / WLAN Channels, Frequencies, Bands & Bandwidths. Retrieved March 17, 2015, from http://www.radio-electronics.com/info/wireless/wi-fi/80211-channels-number-frequencies-bandwidth.php

Proffitt, D. R. (2006). Embodied Perception and the Economy of Action. *Perspectives on Psychological Science*, *1(2)*, pp. 110-122.

Reddy, V. G. (n.d.). ARM NEON technology introduction. Retrieved March 17, 2015, from http://www.arm.com/ja/files/pdf/AT_-_NEON_for_Multimedia_Applications.pdf

Scan Computers. (2014 November 22). 60FPS Livestream Encoding in OBS: x264 vs Intel QuickSync vs Nvidia NVENC. Retrieved from https://www.youtube.com/watch?v=Z6uaPD_5r4w

Screen Capture. (n.d.). A free open source windows "screen capture" device and recorder. Available from https://github.com/rdp/screen-capture-recorder-to-video-windows-free

Sensics, Inc. (2011-2013). Wireless Video Link product description. Retrieved March 17, 2015, from http://sensics.com/additional-options-2/low-latency-hd1080-wireless-video/

Silicon Classics. (2012, November 17). Forte VFX1 Virtual Reality Headgear. *Silicon Classics*, *5.* Retrieved from http://www.youtube.com/watch?v=J0n5B3fl-bU

Smulders, P.F.M. (2003). 60 GHz radio: prospects and future directions. Proceedings Symposium IEEE Benelux Chapter on Communications and Vehicular Technology, Eindhoven, pp. 1-8.

Steptoe, W. (2014). AR-Rift. Retrieved March 17, 2015, from http://willsteptoe.com/post/66968953089/ar-rift-part-1

Sterling, B. (2013, February 24). John Carmack's latency mitigation strategies. Retrieved from http://www.wired.com/2013/02/john-carmacks-latency-mitigation-strategies/

Streaming Media. (2012, May 8). Sorenson Squeeze 8: x264 vs. MainConcept. Retrieved from https://www.youtube.com/watch?v=Hq-P-eNnxZQ

Sullivan, G. J., Marpe, D., & Wiegand, T. (2006, August). The H.264/MPEG4 Advanced Video Coding Standard and its Applications. Standards report. *IEEE Communications Magazine*, pp. 134-143.

Sunxi Community (2015). Sunxi Linux Kernel. Available from http://linux-sunxi.org/Linux_Kernel

Sunxi Community (2015). Sunxi Mali 400 Driver. Available from http://linux-sunxi.org/Mali_binary_driver

SuperData Research (2014, April). eSports: Digital Games Brief. Retrieved from http://gallery.mailchimp.com/a2b9207999131347c9c0c44ce/files/SuperData_Research_eSports_Brief.pdf

Superuser Community. (2011, November 7). Video Conversion done right: Codecs and Software. Retrieved from http://blog.superuser.com/2011/11/07/video-conversion-done-right-codecs-and-software/

Sutherland, I. E. (1965). The Ultimate Display. Proceedings of IFIP Congress, pp. 506-508.

Tangled [Msg 1] (2013, July 24). OCULUS VR General Discussion forums. Message posted to https://forums.oculus.com/viewtopic.php?t=3047&p=45725

Techspot (2009). How to Install & use unsigned drivers in Windows Vista/7 x64. Retrieved from http://www.techspot.com/community/topics/how-to-install-use-unsigned-drivers-in-windows-vista-7-x64.127187/

Thomas, A. (2010-2014). Video Bitrate Calculator. Dr. Lex website. Available from http://www.dr-lex.be/info-stuff/videocalc.html

TP-Link. (n.d.). TP-LINK AC1200 Wireless Dual Band Gigabit Router (Archer C5). Retrieved from http://www.tp-link.com/en/products/details/?model=Archer+C5

TP-Link. (n.d.). TP-LINK N900 Wireless Dual Band USB Adapter (TL-WDN4200). Retrieved from http://www.tp-link.com/en/products/details/?model=TL-WDN4200

Twitch.tv. (n.d.). World's leading video platform and community for gamers. Retrieved from http://www.twitch.tv/

USB Implementers Forum, Inc. (n.d.). Introducing Certified Wireless USB from the USB-IF. Retrieved March 17, 2015, from http://www.usb.org/developers/wusb/About_WUSB_FINAL5.pdf

USB/IP Open Discussion Forum. (2014, September 22). Windows Client v.0.2.0.0 Problems. Message posted to http://sourceforge.net/p/usbip/discussion/418507/thread/7ff86875/

USB/IP Project. (n.d.). A general USB device sharing system over IP network. Available from http://usbip.sourceforge.net/

usblyzer.com. (2006-2014). USBlyzer - USB Protocol Analyzer and USB Traffic Sniffer. Availabe from http://www.usblyzer.com/download.htm

Vatolin, D., Kulikov, D., & Arsaev, M. (2013, March). Eighth MPEG-4 AVC/H.264 Video Codecs Comparison. Moscow State University Graphics & Media Lab. Retrieved from http://www.compression.ru/video/codec_comparison/h264_2012/

VCEG - Video Coding Experts Group. (n.d.). ITU-T SG 16 standardization on visual coding. Retrieved from http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/default.aspx

VirtualHere Ltd. (2010-2015). VirtualHere Software. Retrieved from www.virtualhere.com

Voss, J. (2011). *Revisiting Office Space Standards, 11(11)*. Retrieved March 17, 2015, from http://www.thercfgroup.com/files/resources/Revisiting-office-space-standards-white-paper.pdf

Waller, D., Bachmann, E., Hodgson, E., & Beall A. (2007, November). The HIVE: A huge immersive virtual environment for research in spatial cognition. *Behavior Research Methods, 39(4)*, p. 835.

Walton, J. (2015, January 13). Oculus Demos Crescent Bay and VR Audio. Anandtech. Retrieved from http://www.anandtech.com/show/8876/oculus-demos-crescent-bay-and-vr-audio

WHDI LLC. (2011). About WHDI Special Interest Group. Retrieved March 17, 2015, from http://www.whdi.org/About

Wi-Fi Alliance. (2012). Wi-Fi CERTIFIED Miracast. Retrieved March 17, 2015, from http://www.wi-fi.org/discover-wi-fi/wi-fi-certified-miracast

Wiegand, T., Sullivan, G. J., Bjøntegaard, G., & Luthra, A. (2003, July 7). Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on circuits and systems for video technology, 13(7).*

WirelessHD Consortium. (n.d.). About the WirelessHD. Retrieved March 17, 2015, from http://www.wirelesshd.org/about/

WirelessHD Consortium. (2015). For Consumers. Retrieved March 17, 2015, from http://www.wirelesshd.org/consumers/product-listing/

Xinreality. (2015, February 21). Oculus Rift DK1 Specifications. Retrieved 17 March 2015 from http://xinreality.com/index.php?title=Oculus_Rift_DK1#Specifications

Yurek, J. (2011, October 8). The difference between 'color gamut' and 'bit depth'. Retrieved from http://dot-color.com/2011/10/08/the-difference-between-color-gamut-and-bit-depth/

Zinwell (2009). Zinwell HD Wireless ZWD2500 Specifications. Retrieved from http://www.zinwell.com.tw/Products/pdf/ZWD-2500_User%20Manual_20090217.pdf

9. Appendix

Files and folders (/) specified below can be found on the DVD disk attached.

9.1 Server Computer

The specifications of the computer that was used for development, testing and latency estimations are located in a table below:

CPU	AMD FX-8350; 8 Core@4.0GHz; (Turbo mode: 8 Core@4.5GHz)
RAM	Dual-channel DDR3 1600MHz, 8Gb
GPU	Nvidia GTX460 SE, 1Gb 256bit
SSD	SanDisk SDSSDP12 Sata 3, 128Gb
OS	Microsoft Windows 7 N SP1, 64bit

9.2 USB Usage Analysis

USBlyzer output is located in */Server/hmd_usb_usage.txt*

9.3 USB/IP for Windows

Driver source is located in */Server/usb_ip/driver/*

Builds are available in */Server/usb_ip/build/*

9.4 Automation on Server

Autostart script located at */Server/start.py*

9.5 Automation on Client

USB/IP autostart script is located in */Client/usb_ip/start.py*

Mplayer2 autostart script is located in */Client/mplayer/start.py*

9.6 Latency Estimation

Pictures taken for latency estimations can be found in */Latency* folder.

9.7 Oculus Rift DK1 Specifications

Display	7" LCD @60 Hz
Resolution	1280 x 800 (16:10)
Optics	one aspheric acrylic lens per eye (7X)
Interaxial distance	63.5 mm
Tracking	3dof rotational
Tracking frequency	1000Hz
End-to-end latency	50 - 60ms
Field Of View	Typical monocular: 99° H; binocular : 106° H Default rendering (SDK v.2.5): 114.5°H, 125.5°V
Weight	380 grams
Release date	March 29, 2013

9.8 Client Computer Image

Image of the memory card (zip compressed) of the Client computer is located at */Client/system_image.zip*

9.9 WiDi Latency

Video recording (30 FPS) of the test performed is located in */Latency/WiDi* folder.