A Cross-Layer Power Analysis and Profiling of Wireless Video Sensor Node Platform Applications

by

Tejas Shah

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved May 2014 by the
Graduate Supervisory Committee:

Martin Reisslein, Chair
Jennifer Kitchen
Michael McGarry

ARIZONA STATE UNIVERSITY

December 2014

ABSTRACT

Wireless video sensor networks has been examined and evaluated for wide range of applications comprising of video surveillance, video tracking, computer vision, remote live video and control. The reason behind importance of sensor nodes is its ease of implementation, ability to operate in adverse environments, easy to troubleshoot, repair and the high performance level. The biggest challenges with the architectural design of wireless video sensor networks are power consumption, node failure, throughput, durability and scalability. The whole project here is to create a gateway node to integrate between "Internet of things" framework and wireless sensor network. Our Flexi-Wireless Video Sensor Node Platform (WVSNP) is a low cost, low power and compatible with traditional sensor network where the main focus was on maximizing throughput or minimizing node deployment. My task here in this project was to address the challenges of video power consumption for wireless video sensor nodes. While addressing the challenges, I performed analysis of predicting the nodes durability when it is battery operated and to choose appropriate design parameters. I created a small optimized image to boot up Wandboard DUAL/QUAD board, capture videos in small/big chunks from the board. The power analysis was performed for only capturing scenarios, playback of reference videos and, live capturing and real-time playing of videos on WVSNP player. Each sensor node in sensor network are battery operated and runs without human intervention. Thus to predict nodes durability, for different video size and format, I have collected power consumption results and based on this I have provided some recommendation of HW/SW architecture.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Wireless sensor networks have received a noteworthy interest from the research community in recent years due to its ability to capture video at distributed video sensor nodes and transmitting the video through multiple wireless hops to sink nodes [2, 5, 24, 33, 43, 46, 49]. Wireless sensor network is a collection of nodes where each node consists of a radio transceiver with an antenna, a microcontroller to interface with sensors and a battery or another energy source. Wireless video sensor network has been used in broad field of applications such as video surveillance [21, 29, 47], video tracking [7, 10], remote live video and control [16, 28] and computer vision [8, 12]. There are some features of wireless video sensor networks where lot of research has been done such as multisensor image fusion [6, 34], image and video compression techniques [27, 38], lightweight operating system and middleware [9, 13, 30] and resource allocation strategies [19, 20, 37, 43]. A wireless video sensor node platform is a sensor platform that considers video as a basic element just as a data sources such as heart monitoring machine inside a human body connected wired or wirelessly to the network within the "Internet of Things" framework. There are some resource constraints that come with wireless sensor node design such as power consumption, throughput, cost and node failure. Power consumption is an important factor to ensure that the node remains for long life time. Power consumption of a sensor node depends on wide range of device choices such as power source type, component selection(camera, segment length, video resolution, codec, container), network management algorithms etc. Thus to ensure that sensor node survives for long time, it has to provide most of power modes (On, Ready, Doze, Sleep, Idle, Hibernate) if

not all of them. Second important resource constraint is throughput of the sensor node. The throughput of a node is defined as number of video frames per second received by the sink node from the source node [43]. Third important resource constraint is reliability of data. The reliability of data through sensor nodes needs to be ensured or network might not provide data in time needed in some applications such as video tracking. The networks ability to deal effectively with node failure, making sure that data is delivered in time will ensure reliability of data transfer through the sensor network.

Flexi-WVSNP design is a video sensor node and the three core components of this design are 1) Dual WiFi-Zigbee radio for video streaming 2) Middleware for controlling dual radio and 3) Hardware-Software module architecture. Flexi-WVSNP is a gateway node between sensor network and "internet of things" framework. This sensor platform is designed to be low-power, low-cost and compatible with the existing wireless sensor network. Current video delivery architecture, eg: transmitting the video over Real-Time Streaming Protocol(RTSP) are not adaptable with the wireless sensor networks while the "internet of things" framework requires different approaches to be able to integrate with the wireless sensor networks. There are some approaches that will be able to connect internet of things and wireless sensor networks: 1) Make video acquisition and delivery universal by using HTTP segmented video streaming approaches. 2) Lower the cost of platform by using open-source tools and open it for further research. 3) Make use of highly portable HW/SW architecture to benefit from changing off-the-shelf components.4) For fast application re-purposing use flexible power source. 5) For sensor network management, data-fusion and transmission use our low coupling but highly cohesive dual wireless radio approach. The paper is organized as follows: In section II, I talk about existing approaches on measuring power consumption, throughput, and cost of node. The difference between flexi-wvsnp

and existing wvsnp is described in section III and in section IV; I talk about challenges with wireless video sensor nodes. Challenges that I had to deal with in thesis are described in section V. Test-bed description and data collection methodologies are described in section VI and VII respectively. Data representation is presented in VIII, recommendations from the data presented in IX and conclusion and future works are provided in X.

Chapter 2

RELATED WORKS

There are some existing papers that have performed power measurement and some talks about power measurement and management in sensor networks. Here I will talk about the methods that I reviewed for performing power measurement and then talk about my work which is first of a kind to focus on video power on sensor nodes.

In [22], they have surveyed different approaches to measure the energy in wireless communication devices and have also provided analysis of each approaches by providing their advantages and disadvantages. Energy consumption of any electrical device is calculated by the product of current (I), voltage (V) and time (t). To calculate energy, voltage and time can be calculated directly, but there is no way direct way to calculate current. Therefore in this paper they have surveyed different current measuring techniques and then they have made recommendation on which technique to use according to the requirements. Here are the approaches: 1) Shunt resistor: Resistor is placed in series in the electric circuit. The current draw is same across the whole circuit and therefore the current draw across the resistor is just V/R. The advantage of using shunt resistor is that it is the easiest method to use. If the voltage over the resistor is too large, it might cause malfunctioning of the device. To avoid this side effect, voltage across the resistor has to be low and thereby the resistor value has to be low. The major disadvantage of this approach is its inability to measure high dynamical signals. If the current has high dynamic range, then the accuracy for low current will be insufficient. 2) Voltage to Frequency Conversion: This method is an extension to shunt resistor method. The goal of this method is to

extend the dynamic range of shunt resistor to improve the accuracy for low current. In this method, shunt resistor is connected to a new block "voltage to frequency conversion". The current can be measured here with high dynamics, in other words low current can be measured with high accuracy as well as high current. The main advantage of this method is that it provides with high dynamics. 3) Inductor: The current measurement using inductor method comes from current clamps used in heavy engineering. Voltage is induced in the inductor by the the electric field around the electric line. By sampling the induced voltage, current can be determined. The disadvantage of this method is that this method requires calibration after the energy of an electrical device is measured. And secondly, although inductor method supports high sampling rate, it would be less beneficial as it is highly susceptible to noise. 4) Coulomb Counter: There are two capacitors in this method which are charged and discharged in turn. When a capacitor is discharged, the discharge current provides the energy. Since the capacity of both capacitors is known, time required to discharge the capacitor implies current. The disadvantage of this method is that, low current results in low frequency and thus low temporal resolution. The temporal resolution for coulomb counter method depends on current draw and since the current draw is not constant, the temporal resolution is also not constant.

In [32], the authors have presented a low-cost experiment for performing power measurements for wireless sensor networks and they have also provided the calibration and validation of this experiment. In this setup, clamp-on current probe is used to collect current measurements but they have also shown the shunt resistor technique. The current probe is clamped around the power supply. Here, power supply and the output voltage through the current probe are sampled and this is the linear function of current through the clamp. The biggest drawback of current probe method is that the results obtained using this method are very low-profile.

In [23], the authors have presented Sensor Node Management Device(SNMD) which they have used in their WSN testbed to perform energy measurement of the sensor nodes. SNMD used here is not to monitor actual deployed sensor network as it uses a wire-based infrastructure which is only available in test bed infrastructures. SNMD helps protocol and network evaluation in estimating and enhancing the network and node's lifetime. In this approach also shunt resistor method is used to collect current measurement and thereby calculates energy consumption. In [26], a scalable power observation tool (SPOT) is presented to measure the energy and power consumption of sensor node over four decades of dynamic range or temporal resolution of microseconds. In this method also shunt resistor is used to perform current measurement. In [18], Avrora simulation tool is used to see if energy measurement can be measured accurately in wireless sensor network and then the results obtained from the tool are compared with the results obtained from experiment using SANDbed.

Chapter 3

DIFFERENCES BETWEEN FLEXI-WVSNP AND EXISTING WVSNPS

There are some difference between existing WVSNPs and flexi-WVSNP that I will talk about in this section. Flexi-WVSNP design is not a platform rather it is a video sensor node which can stream video using both WiFi and Zigbee [43]. This dual-radio system is capable of reconciling with other zigbee sensors and it grants a gateway access for sensor to internet thru WiFi. The existing sensor platform design are either targeted for vast applications making it a general purpose architectures, or they are targeted for a specific narrow applications. The general purpose application platform design tries to cover all peripherals as well as pcb modules that might be needed for an application [43]. These type of platform design usually experiences a high power consumption, lot of under-utilized modules and huge cost of manufacturing them. The narrow application platform designs provides with high performance when used for application for which the platform is designed for but on other side since they are targeted for specific application, it makes them over-customized [43]. Hardware modules in narrow application platform design are quite dependent on each other to meet timing and cost constraints of the main application. Therefore hardware design will be inflexible most of time and would require re-design for any change in application load or onsite requirements. Since software modules are dependent on each other, they wouldn't be reusable for any change in other software or hardware modules [43].

Flexi-WVSNP design attempts to engage hardware and software design but on the other side tries to keep them away to avoid binding as in narrow application platform designs. This design is not targeted for any specific application, rather it

is very adaptable and cost flexible to be able to cover a low surveillance system to remote monitoring system [43]. The general WVSNP architecture works on the facts that 1) no need of speculating any application and 2) removes the need of designing the complete WVSNP initially. Hardware and semiconductor keeps on evolving with new research and therefore power reduction depends on these important elements added for a precise application. These evolution in hardware and semiconductor processes drives the design of flexi-WVSNP. Due to evolution in semiconductor processes, processor selection is the most important initial selection. SoC modules should be such that they can be controlled without needing any support of other modules of SoC from active state to power off. SoC should also have hardware accelerators which are needed for video capture, encoding and streaming. If any other capabilities are needed, they can be fulfilled by using flexible connections. In my case I am using iMX6 Solollite, Wandboard dual/quad boards which has low power capabilities, modules that can be controlled from active state to power off and wandboard boards do have hardware accelerators for video capturing, encoding and streaming. Besides hardware, another important factor that motivates flexi-design is software architecture. The software now-a-days is changing continuously while open source software is advancing at an exponential rate. Therefore if flexi-design were to be tied to a current hardware and software architecture, then this would fail to comply with adaptability required to achieve low power and cost for WVSNP.

Chapter 4

CHALLENGES WITH WIRELESS VIDEO SENSOR NODE

There are some challenges that has to be dealt with when video is introduced into wireless sensor node. In [41], the authors talk about issues and contingency with wireless video sensor node and then they discuss the key research issues with video sensor network such as camera coverage, network architecture and data processing and communication of low-power video. Here are the issues discussed in the paper: 1) Emergency Saving Control Strategy- Video sensor nodes are used in environment complicated applications where battery change is not possible and the power consumption of these node is higher than the traditional sensor. Therefore the biggest question is how to conserve the limited battery energy to try and extend the nodes lifetime. 2) Network Architecture of Video Sensor Nodes- Traditional sensor networks based on cluster structure are not capable of processing image and media data, plus processing and transmission of data from the network will consume lot of network resources. Therefore the authors are proposing to use isomorphic or heterogeneous video sensor nodes to build network that will be able to perform powerful functions and also provide with excellent performance. 3) Real-time transmission- There are some requirements in delay and synchronization of video data transmission. Currently the processing capacity of video sensor nodes is limited as well as broadband resources. They are proposing to do research to design a reliable transport protocol to be able to use in video sensor node and also conserve energy and can extend the life of network. 4) Collaborative Processing Technology of Video Sensor Nodes- The coordination of sensor nodes is necessary for high consistency. The perception and communication range of single sensor is limited as well as a single node cannot pro-

cess large data. Therefore there is lot of collaboration needed. 5) Storage and Search of Video Information- Video sensor nodes capture lots of visual information in the environment and therefore the data is very large. How to store the large amount of data and how to search for the interested information. 6) Deployment and Coverage of Video Sensor Nodes- The traditional sensor networks deployment and coverage are based on directional perception model which doesn't apply to the video sensor nodes which already has directional perception. They are proposing for a new directional perception model upon which a new video sensor network is designed to be able to save energy as well as do reliable monitoring.

In [44], the authors are analyzing how the algorithm designed for traditional sensor networks behave in the video sensor networks. They are analyzing the algorithm that integrates the coverage and routing problem. Their results suggest that the sensor network doesn't provide expected results in coverage of monitoring areas because of the way camera captures the data. They have discussed the difference between traditional network and video sensor network that leads to such a result and have also given some ideas on how to design protocols for video sensor networks. They have analyzed a situation in which camera nodes focus on one plane and their results shows that the application aware routing protocol behaves very differently in video sensor network than the traditional network due to the unpredictable focus of the camera.

One of the approaches, which is found to be suitable in Flexi-WVSNP is to reply on duty cycling and dual-radio as a means to fit video acquisition into power saving algorithms of sensor networks. Since the video is big and there are issues with storing it as described in [41], we decided to explore segmented videos to fit in video sensor networks. The existing segmented protocols like HLS and MPEG-DASH are neither power friendly nor sensor friendly, therefore we have an alternative to these

protocols, WVSNP-DASH that we are trying to explore if it can help to lower the power consumption. There are two video representations (BIG and SMALL) that we have for WVSNP DASH. BIG and SMALL video segments can be 2 sec, 5sec and 10sec long. Similar to HLS and MPEG-DASH where during the streaming if the buffering time of segment is large, then the next video segments will be streamed at lower bit-rate, in WVSNP-DASH the streaming segments are switched between BIG and SMALL video segments. HLS segments can only be encoded in h264 as iPhone can only play h264 encoded videos and MPEG- DASH segments can be encoded only in h264 and webm, while for WVSNP-DASH it can be encoded in h264, webm, ogg and any other encoders. Several earlier node designs in literature have generally focused on novel protocols, better buffer algorithm, better compression schemes. In this thesis an exhaustive empirical evaluation shows that a lot of design parameters can be identified throughout the architecture design, by observing different stages of the segmented video capture and, storage and transmission flow. This enables architecture fine tuning parameters ranging from discoveries of avoidable power consumption waste points in the flow, better choice of tools and libraries to use, video encoder parameters and other transcoding elements like muxers, SW only versus HW accelerators, etc. This thesis reveals many of these in chapter 6 and summarizes them as shown in figure 3 to 67.

Chapter 5

CHALLENGES TACKLED DURING THESIS

There were many challenges I faced during the thesis, but during the course of finding solutions to these challenges I read and learned lot of new concepts from my research online and also by talking to my advisor. Some of the challenges includes booting bootloader and kernel successfully, setting up proper NFS and boot parameters to avoid errors while mounting the file-system, getting gstreamer to capture video from USB camera and segmenting to obtain HLS segments.. I started working with beagleboard at first by using demo bootloader (uboot), kernel (uImage) and rootfs. After succeeding in this process, I cross-compiled uboot and uImage using source code and compiler from linux-linaro and replaced this new files with the default files on the sd-card. While playing around with the board, I broke the board, therefore from here, instead of using the real board I started emulating the beagleboard board using QEMU. To emulate beagleboard, a hardware pack is required which contains uboot, kernel and rootfs packed together in a file. After initial success in booting the default hardware pack, I changed uboot source code by changing uboot prompt for a start. This compilation failed, but after failed attempt of help from linux-linaro and forum onlines, I started using yocto. Yocto project provides a wide range of hardware and device emulation through QEMU (a quick emulator to emulate hardware board and devices). Yocto project also provides with source code for uboot and kernel just as linux-linaro does. Just as linux-linaro, initial boot worked properly for yocto also, but after changing the source code, compilation failed for yocto also. I encountered same error in yocto also and failed in resolving errors here also due to lack of support. During this time, I came across Freescale's iMX6 board (Sololite,duallite for

smart devices), which were capable of performing what was required for my thesis. Therefore I started using these boards with LTIB (Linux Target Image Builder) which also provides source code for uboot and kernel and is also compilation tool just as linux-linaro and yocto for preparing binary file.

With the setup described above, sdcard needs to be removed every-time there is a change in bootloader or kernel source code or if for ex: a video file captured from USB camera connected to the board is saved on the rootfs. To avoid this problem, I used NFS/TFTP. By starting NFS/TFTP server and with board connected to internet via ethernet cable, board doesn't get uboot and kernel from the sd-card instead it gets from the server, and rootfs is also mounted from the server itself. Since rootfs is mounted from the server, it is also accessible from the laptop and thus any file saved during board run is accessible instantaneously without removing the sdcard. To avoid errors while kernel tries to mount rootfs, boot parameters have to be set perfectly as given. The board(iMX6 Sololite) doesn't come with a in-built camera, therefore I had to use a USB camera to perform video capturing from the board. Since the board didn't had camera, the boards default kernel configuration didn't had camera plugins selected for the camera. The biggest challenge here was to figure out which plugins I need for camera to work and also want the plugins to successfully be compiled. After posting question on Freescale's open source community and with some back-and-forth emails, I was successfully able to install gstreameer with all dependencies and was able to also get camera capturing the video.

Chapter 6

TEST BED

This section describes how the reference video and segments were generated, parameters of reference videos, codecs used to encode those videos and the test cases that were used to measure the power consumption, computation speed and throughput. All the results were conducted using ASU reference video(without any video). The videos shows people walking by at different places on ASU campus. It was encoded into two video options: The first video option is "BIG" video where the resolution of the video is 640 x 360, bitrate of 500 kbps and framerate of 30 fps, and the second video option is "SMALL" video where the resolution of the video is 320 x 180, bitrate of 150 kbps and framerate of 15 fps. Now, these videos were segmented into MPEG2 TS for HLS and MP4 for WVSNP-DASH player. Each video files (HLS and MP4) were created by segmenting the orginal video into 2, 5 and 10sec segments using ffmpeg. Besides this, MPEG-DASH stream with ISOBMF files were created using MP4Box from "BIG" video option. For MPEG-DASH, video segments in size of 5se, 10sec and 15sc are available. To obtain power consumption result, tests were performed on different platforms as below:

- *Windows 7 64 bit, Macbook Air Mid 2012 with i5-3427U 1.8 GHz processor and 4 GB RAM*: Google Chrome (version 32) was used for WVSNP playback. While for HLS, JWPlayer 6 with the HLSProvider plugin running was used for playback and MPEG-DASH was played back on DASH-JS player- both were ran on same Google Chrome.

- *Ubuntu 13.10 64 bit, Dell OptiPlex 360 with Intel Core2Duo E7300 2.66GHz processor and 2 GB RAM*: Google Chrome (version 32) was used for WVSNP playback. While for HLS, JWPlayer 6 with the HLSProvider plugin running was used for playback and MPEG-DASH was played back on DASH-JS player- both were ran on same Google Chrome.

The power consumption results were performed on server side using Wandboard-quad board which runs mongoose server and serves all players and videos. The results were performed for VOD (playback) with different segment size(2sec, 5sec and 10sec), codecs using the BIG video and SMALL video option for HLS, WVSNP and MPEG-DASH of videos served with mongoose webserver running on Wandboard-quad board. For MPEG-DASH playback, there is 15sec segment files also available besides the above mentioned segment files. The above results were taken with board and client (laptop) connected to the router under WiFi and therent connection. Here are the comparison graphs presented in the next chapter with consumption results taken for above defined different scenarios.

- WVSNP vs HLS Playback (2sec, 5sec, 10sec) with Ethernet connection on Mac
- WVSNP vs HLS Playback (2sec, 5sec, 10sec) with Ethernet connection on Windows
- WVSNP vs HLS Playback (2sec, 5sec, 10sec) with Wifi connection on Mac
- WVSNP vs HLS Playback (2sec, 5sec, 10sec) with Wifi connection on Windows
- WVSNP vs HLS Playback (2sec, 5sec, 10sec) with Wifi connection on Ubuntu using BIG and SMALL video option
- MPEG-DASH (10sec) with Wifi connection on Ubuntu

After VOD measurements, LIVE capturing and real-time playing measurements were performed on quad core board. LIVE video was captured with different segment size with H264 encoding using the BIG video and SMALL video option of WVSNP and HLS. USB webcam was used for live-capturing by putting it in front of monitor running ASU reference video in loop. The camera is enabled for capturing using ffmpeg which encodes the video using x264enc(H264) software encoder and segments the captured stream on-the-way. For HLS, ffmpeg performed segmentation and it also creates m3u8 file using HLS muxer. While for WVSNP, instead of using ffmpeg segmenter, ffmpeg command is ran in a unix script loop for each segment (i.e. camera shuts down after each segment is captured and is turned on again for the next segment to be captured). This is necessary for WVSNP because WVSNP-DASH player requires each file in the WVSNP syntax file to be final, which current version of ffmpeg segmenter is not providing this. Although the power consumption of WVSNP is lower than HLS as will be shown in the next section, the results could be much better since ffmpeg keeps on turning on and off camera in case of WVSNP, while for HLS it is a capturing continously. Here are the test cases for LIVE case:

- WVSNP vs HLS LIVE (2sec, 5sec, 10sec) under Ethernet connection on Mac
- WVSNP vs HLS LIVE (2sec, 5sec, 10sec) under Etherent connection on Windows
- WVSNP vs HLS LIVE (2sec, 5sec, 10sec) under Wifi connection on Mac
- WVSNP vs HLS LIVE (2sec, 5sec, 10sec) under Wifi connection on Windows
- WVSNP vs HLS LIVE (2sec, 5sec) under Wifi connection on Ubuntu

Besides these results on quad core board, some other results were performed on Wandboard-dual board. All power consumption results were performed while cap-

turing the video with different cameras(USB and CSI camera) facing monitor running BIG reference video using dual board of different segment size (2sec, 5sec and 10sec) and encoding those segments with different codecs. A comparison is made between HLS and WVSNP with HLS using MPEG2 container and WVSNP using MP4 container. Since ffmpeg currently doesn't support hardware encoder for iMX6 processors, results were only taken using software encoder. Nextly, a comparison between USB camera and CSI camera is shown by capturing WVSNP videos with ffmpeg commands using these two camera with H264(libx264) encoding. Next, WVSNP videos were captured using CSI and USB camera with gstreamer commands and encoded using hardware and software H264 encoder. Based on this, different comparisons were made with CSI and USB camera and, hardware and software H264 encoder. Next, WVSNP videos were captured using USB camera and CSI camera with videos encoded using hardware and software MPEG4 and, hardware and software Jpeg encoder. Different comparisons plots were created such as comparison between software and hardware encoder on same camera, a comparison between same encoders on different cameras.

Chapter 7

DATA COLLECTION METHODOLOGIES

There are several software tools that can help to measure the power consumption on board(iMX6 sololite, wandboard-dual) and mobile PC under ubuntu. For the tools to work on mobile PC, it requires the device running on the battery. One of the tools, that performs power consumption on mobile PC is called PowerTOP [45]. PowerTOP is a linux tool which helps to diagnose problems with power consumption and power management [45]. Besides this diagnostic tool, powertop also has an interactive mode through which user can observe different power management settings that were not enabled for this linux distribution. If powertop is summoned without any arguments, it starts by default in interactive mode. After invoking, it reports which components are expected to be consuming more power, from software applications to active peripherals in the system. There are several options available to run powertop, but one of the important one is "calibrate mode". Powertop can track the power consumption and system activity of the device, when the device is running on a battery. It starts reporting the power estimates after getting enough measurements [45]. To get authentic measurements, calibrate mode can be used to enable the calibration cycle. The calibrate cycle will go through USB activities and workloads. The drawback of this tool is that it only shows power consumption after it takes five minutes of measurements.

One of the other approach is measuring process/device power consumption using "powerstat" tool [17]. This tool monitors the system for 10 sec and keeps on doing it, until it captures 48 samples in around 480 sec. While it is collecting data, it provides with the following output fetched from the kernel: time- time it started

monitoring, user- shows the cpu usage of processes started by current user, nice- if a applications is running while powerstat is running and requires lot of cpu time , then this value shows up, sys- shows the cpu usage by kernel, idle- the time cpu is in low power mode or idle state, IO (IO wait)- after cpu sends a signal to a hardware to ask if something needs to done, cpu waits for the hardware to reply given that cpu has nothing else to do , run- shows current running process, ctxt/s (context switch)- shows the number of times cpu paused and resumed running processes per second, IRQ/s- IRQ is the signal used by hardware devices to talk with the CPU to get the work done. IRQ per second here shows the number of irq request received from the hardware, fork- a fork is a method used by currently running process in which it makes a copy of itself (this shows the number of process forked), exec- it is function of kernel which replaces the current running processes with the new one(if the number here says 10, then 10 processes are changed with the new ones in this particular period), exit- the process terminates by itself after completing its jobv and lets the kernel know by this command (the number here shows the processes that exited during this time) and watts- shows the power [17]. The last value "watt" is the most important information and it shows the current power consumption rate (energy unit per second) [17]. After collecting all the samples, it shows "average", "minimum" and maximum" value of each above mentioned field and at the very end shows the "average" power consumption plus the standard deviation.

Another alternative is PowerAPI which is a software library to monitor the energy consumed by the processes [3, 35, 36]. This approach doesn't require any kind of external arrangement to measure the energy consumption whereas the above two software tools requires the device under test to be operating with battery connected to be able to measure any power or energy [3]. This is fully software-based approach wherein the energy analytical models outlines the consumption of different hardware

components such as CPU, memory to make an estimate. In PowerAPI, each module in their architecture corresponds to measuring unit for each hardware unit. The objective of PowerAPI is to present simple and efficient way to measure the energy consumption of the device [3, 35, 36]. It is very simple because user just has to provide the process name that needs to be monitored and refresh period (every how many seconds the results should be displayed). It is also very efficient because user can select which modules they want in their library to be able to monitor corresoponding hardware device [3, 35, 36].

Here are some techniques that have been used to measure power consumption on embedded boards. In the first technique, they talk about power management in the linux kernel [14]. The presentation describes building blocks of power management on embedded boards and have shown, how each building blocks can be used to manage power. The building blocks are: suspend and resume, CPUidle, Runtime power management and, frequency and voltage scaling. The suspend operation will take the system into the lowest power state being supported by the board [14]. Arm architecture has a assembly code included in the kernel which implements CPU specific "suspend and resume". The "enter-state" function calls CPU specfic suspend-ops functions for any kind of power management application and it also executes suspend and resume functions for the device. And, userspace can be used to start the execution of this function. Runtime power managment means that all hardware components are turned off which will not be used in the near future from the perspective of userspace. CPUidle puts the idle CPU into low-power states. Current CPUs have several sleep states giving different power savings according to the wakeup latencies. To manage idle, CPU has a feature named "dynamic tick" feature, which allows to remove periodic ticks to save power and also to know when the next event is listed for smarter sleeps. Frequency and voltage scaling building block can be set using cpufreq

kernel infrastructure. CPUfreq drivers allows cpu to be set to one frequency. To offer dynamic frequency scaling, cpufreq core needs to tell the drivers of target frequency. Now, to decide which frequency to set within the cpufreq policy, cpufreq governors are used. The cpufreq governors are performance, powersave, ondemand, conservative and userspace. Performance sets frequency to highest frequency and Powersave sets frequency to lowest frequency among the borders of scaling-min and scaling-high frequency. Userpace governor allows the user to set the frequency of the cpu, while Ondemand sets the frequency depending on the current usage. Conservative governor sets the frequency of the cpu just as Ondemand, except that the behavior of conservative governor is different. In case of conservative, the frequency increases or decreases gradually depending upon the load rather than increasing to highest value when there is any kind of load. Another method used to measure power consumption is using scripts. These scripts contains commands to measure power and then records them into files [1].

For measuring power on wandboard-dual board, I am using Pico Oscilloscope, current clamp and oscilloscope probe. Pico oscilloscope captures the data and shows it on picoscope software on the laptop. I choose using Pico oscilloscope instead of traditional oscilloscope, because it is cost-effective, it is very small, light weight and very reliable. Initially, I tried measuring current by connecting a breadboard wire from the probe to test points on the board while having connected current probe around the breadboard but this didn't work because test points weren't working to measure current. In second case, breadboard wire was connected from probe to the test point(only measuring voltage) andy by putting current clamp around the power cord. Current measurements were not accurate in this case because of insulation of power cord. So, I stripped the Vin(5V) of the power cord and connected a breadboard wire in between. I placed current clamp around this newly added breadboard wire,

therefore when I plug power cord and start the board it gives the current measurement for the board. This method of measuring current can be dangerous because it might burn the board if there is any loose contact, but it is very reliable method with the tools I have.

If I had time, then instead of stripping the wire and adding a breadboard wire, I would have used Aim TTI's Aim I-prober 520, a new type of current probe. This current probe can measure the current in pcb track. It can also work with any oscilloscope capable of displaying current from DC to 5 MHz and 10mA to 20A. This current probe uses a small fluxgate magnetometer which allows to measure field at a very precise point. Fluxgate magnetometer allows field and thereby current flowing in the track to be displayed. This probe would allow me to avoid all the hassle I went through for measuring the current by stripping wires and adding a peeled breadboard wire.

Chapter 8

DATA REPRESENTATION

In this section, I will show the power consumption results performed for different scenarioes as discussed in chapter6(Test Bed). Before going into results, here is a figure showing the setup used to perform power consumption results.

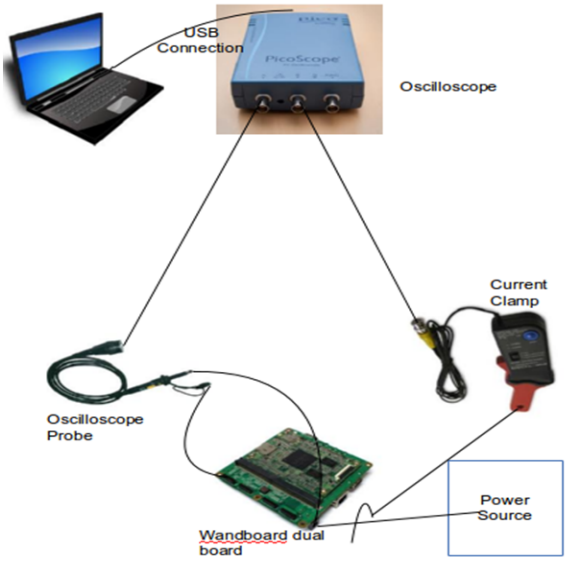

Figure 8.1: Power Measurement Setup

The three main components of measuring power consumption are the oscilloscope, current clamp and oscilloscope probe as shown in the picture. One side of each probe and clamp are connected to the oscilloscope to measure current and voltage and in turn, oscilloscope is connected to laptop to view the current and voltage measurements and save it on the laptop. To measure current, clamp is placed around the wire connected to the board for powering it up, while voltage is measured by connecting probe to the 5V jack on the board.

From the boot time until a video capture is complete, there are four stages. All measurements that will be presented in this section are captured with these four stages. Here is a power measurement figure showing all the four stages for a measurement using wandboard-dual.

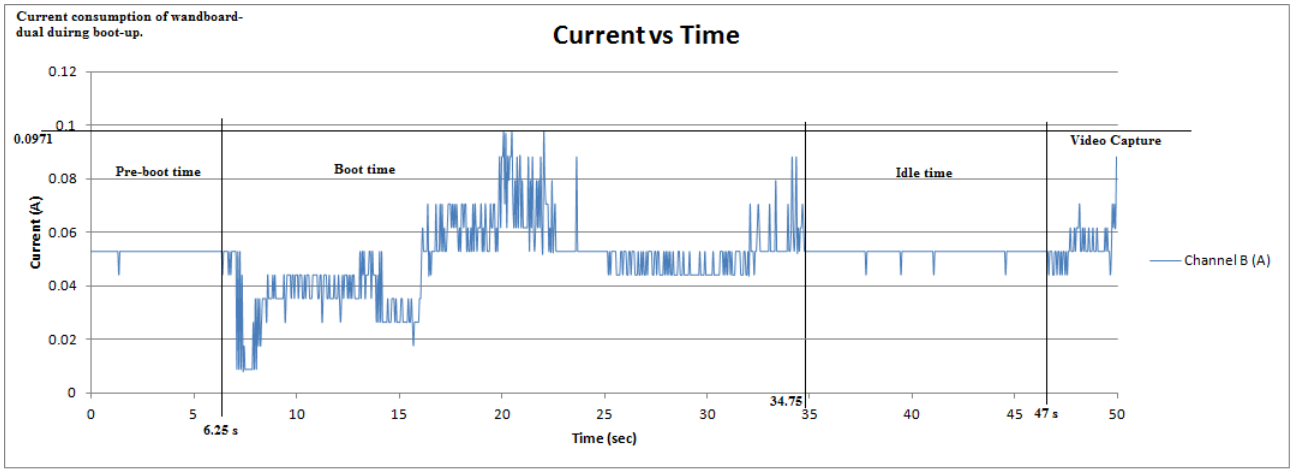

Figure 8.2: Wandboard Boot Time

The first stage is pre-boot time: during this time the board could be in idle state as in here or could be in shut-down. The second stage is boot-time: at six seconds, the board boot-up starts by pressing the "Reset" button on the board and this process takes 28 seconds to complete. The third stage is idle time: during this stage, the board is booted up waiting in idle state at login window to be operated. And, the final stage is video capture time: at 45 seconds, the video capture is started either by running a script or by running a command from Putty screen. After looking at the current consumption of the board at boot-up time for each measurement, I have made an assumption that the pre-boot time, boot-time and idle time will remain same, while the video capture time will be different in each measurement. Therefore from here on, in each measurement result, the video capture current consumption will be showed.

First, here is a current consumption comparison of capturing videos in 2sec, 5sec and 10sec segments for HLS and WVSNP. These video capturing were performed on USB camera facing a monitor running BIG reference video using ffmpeg commands and encoding was performed in H264 using libx264 encoder. There are some differences between HLS and WVSNP: 1) In HLS, as the video segments are being created, an extended M3U(.m3u8) playlist file is created which contains the metadata of the video segments created. While in WVSNP, no playlist file is created or required, making it hassle free to play segments. 2) In HLS, all the video segments are dependent on the first video segment. Now, if the first segment is missing, all video segments after that are unplayable. While, in WVSNP, all video segments are independent of each other. Therefore WVSNP segments are playable even though when some video segments in between are corrupted or missing. 3) HLS only supports MPEG2-Part1 transport stream container, while WVSNP supports all containers such as mp4, webm, MPEG2 etc. Here mp4 container was only used for WVSNP video storage, but for future work other containers should be tried for WVSNP videos because power consumption and size of the segments with other containers might be lesser than mp4 container.

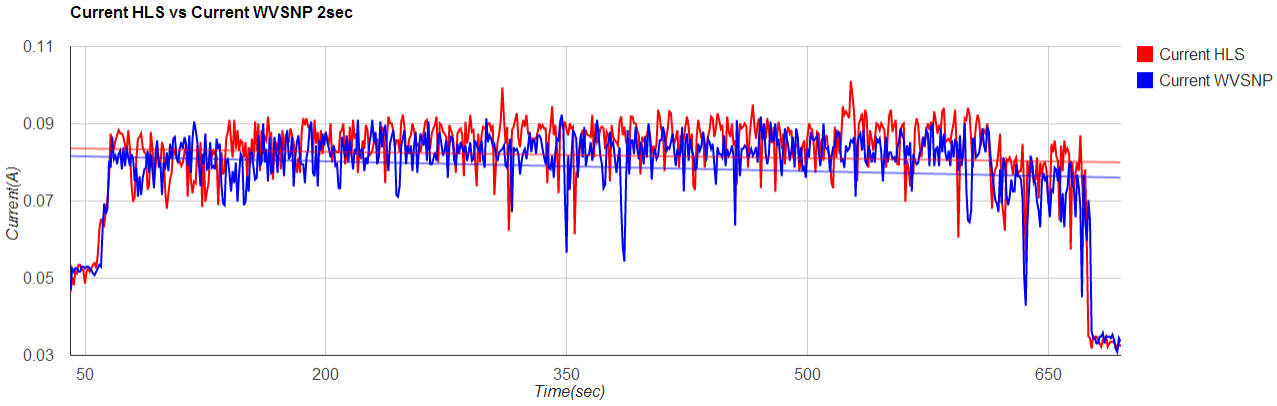

Figure 8.3: HLS vs WVSNP (2sec)

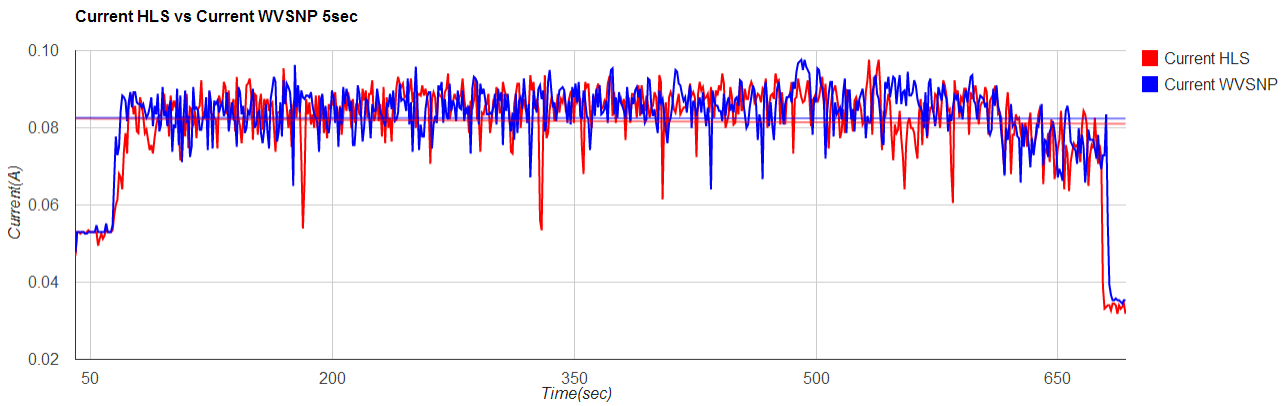

Figure 8.4: HLS vs WVSNP (5sec)

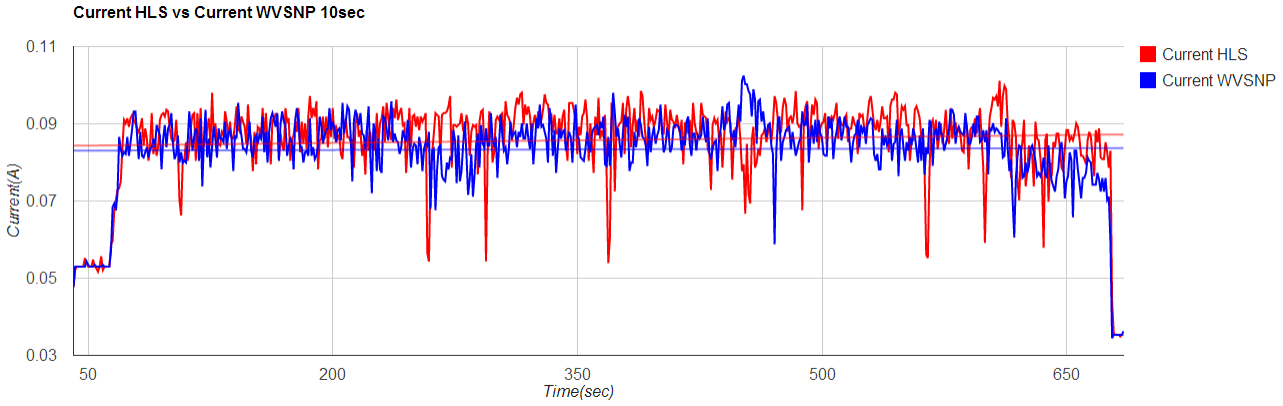

Figure 8.5: HLS vs WVSNP (10sec)

The first 10 sec of all the above plots shows the idle-time and then capturing of the video starts at 1 minute. The current consumption of capturing a WVSNP video is lesser than capturing a HLS video is visible easily for 2sec and 10sec case, while for 5sec the current consumption is same for half time of the capture and then HLS starts performing little bit better than WVSNP.

Now lets look at the current consumption of capturing WVSNP video using CSI camera and USB camera. The WVSNP videos were captured in 2sec, 5sec, 10sec segments and 10min(full) using CSI and USB camera facing a monitor running BIG reference video. These videos were captured using ffmpeg commands and encoding

was performed in H264 using libx264 encoder. Below are the plots showing current comparison between two cameras:

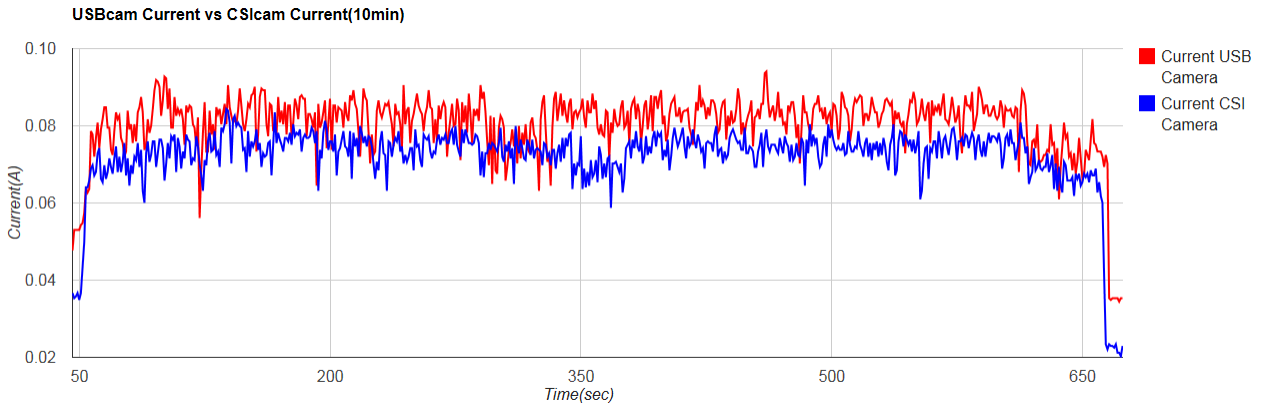

Figure 8.6: USBcam Current vs CSIcam Current (10min)

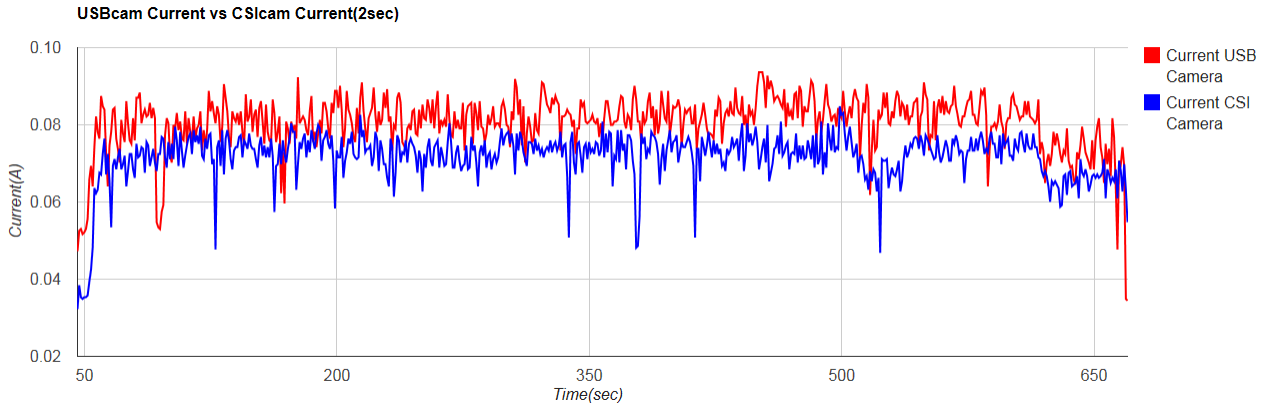

Figure 8.7: USBcam Current vs CSIcam Current (2sec)

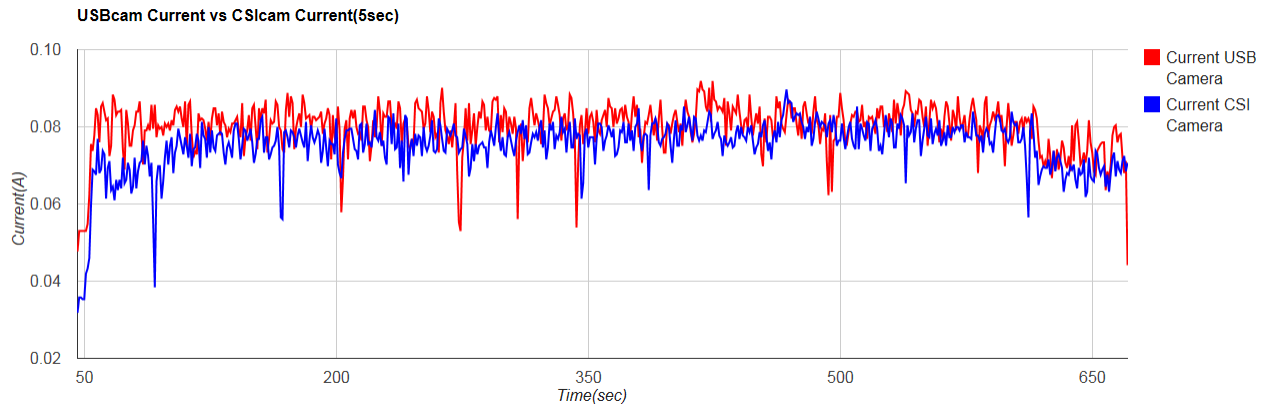

Figure 8.8: USBcam Current vs CSIcam Current (5sec)

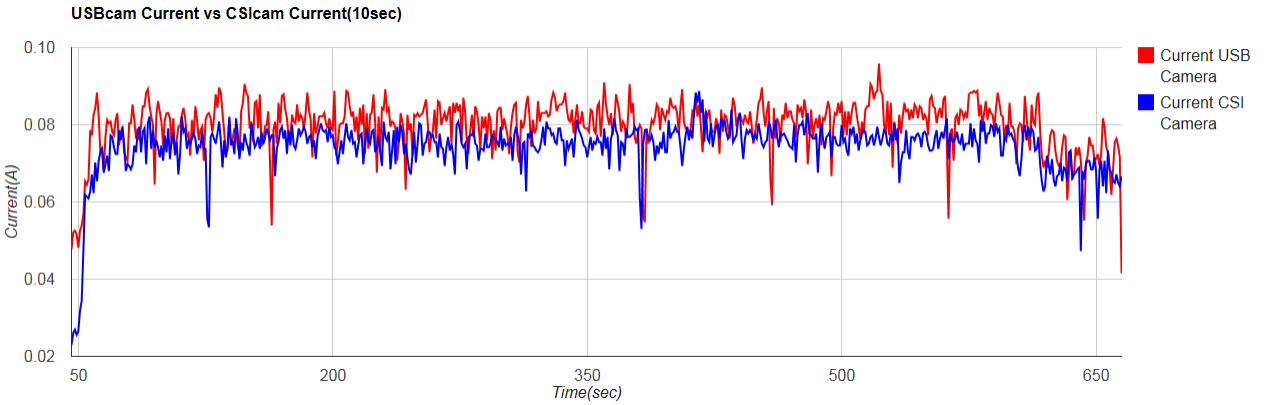

Figure 8.9: USBcam Current vs CSIcam Current (10sec)

The "logitech" webcam was used as USB camera and it connects to the wandboard-dual board through USB peripherial and "wandcam" camera was used as CSI(camera serial interface) camera, which connects to the board via flex cable. The expected result here is that CSI camera will consume lesser current than USB camera. Since the USB/UVC stack has many components, the raw data collected by USB camera has to pass through all components which increases the processing time and besides this there is a software hand-off of raw data on the board for video encoding to software/hardware encoder which increases the current consumption. While CSI stack is smaller, therefore the raw data travels faster for video encoding, plus there

is no data hand-off required like USB camera since there is already a direct path created on board to send data from camera module to the encoders. It can be seen from above plots that video captured using CSI camera has current consumption lesser than that of usb camera as expected. The most substantial difference between current consumption can be noted for the comparison plot of 10min and 2sec case. There is no evident difference between plots of current consumption comparison of segmented video and full video in this case because ffmpeg has a capability of segmenting the videos while the capturing is happening without turning off the camera. Besides this, there are lots of peaks and valleys in segmented video capture plots because of turing ON/OFF of the camera as here, while the plots of full video are expected to not have many peaks and valleys, but here they are present. The difference in overall graph of current consumption comparison between hardware and software encoder will become very evident in gstreamer video capturing because gstreamer doesn't have capability of segmenting a video while it is being captured. Thus, a loop capture is used wherein gstreamer command turns ON the camera, captures the video for 2sec,5sec or 10sec segments and turns OFF the camera again and keeps on repeating the capture.

The following results are current consumption comparison of capturing video encoded by software and hardware H264 through USB camera and CSI camera. The first set is a comparison of video capturing through CSI camera using hardware and software H264 encoder. The capturing is performed using gstreamer commands and encoding is performed using x264enc(software encoder) and vpu encoder (codec=6, hardware encoder).

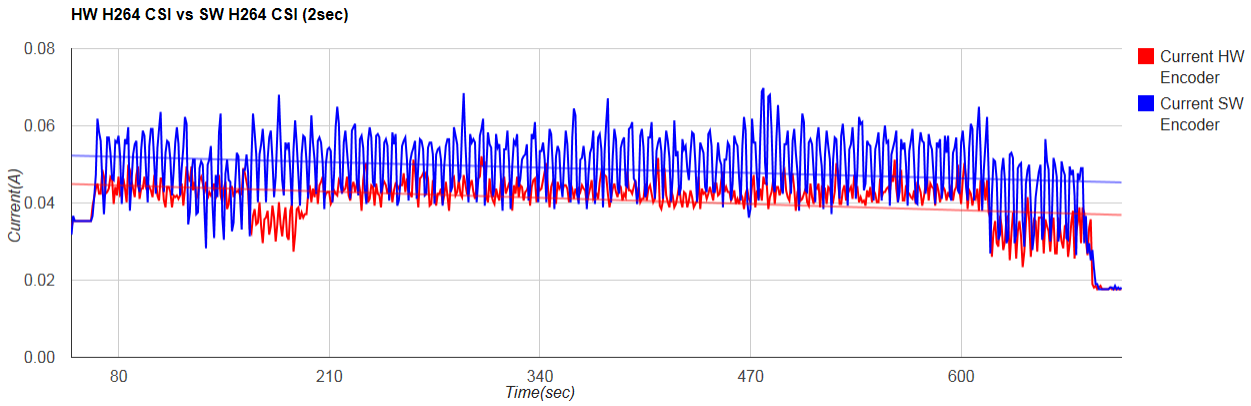

Figure 8.10: Current Consumption Comparison of Video Capturing through CSI-cam Using Hardware and Software H264 (2sec)

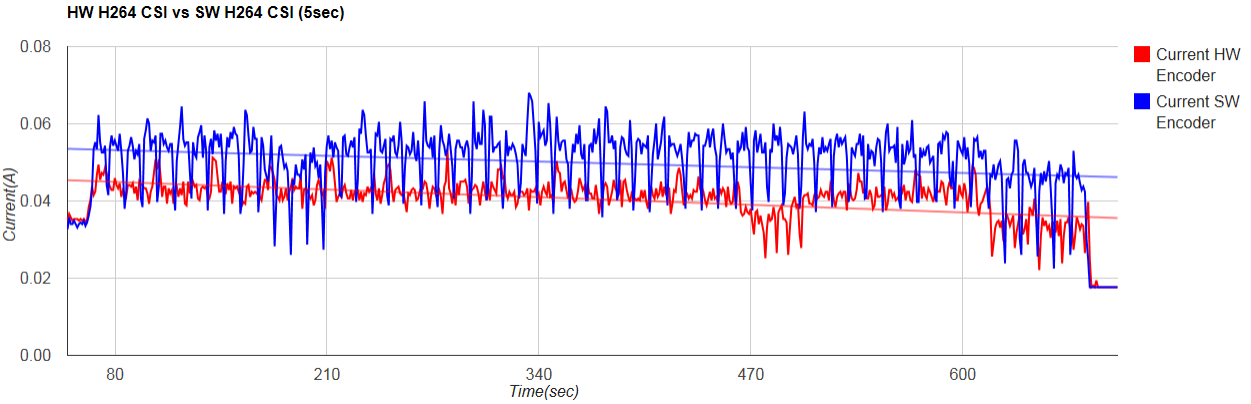

Figure 8.11: Current Consumption Comparison of Video Capturing through CSI-cam Using Hardware and Software H264 (5sec)

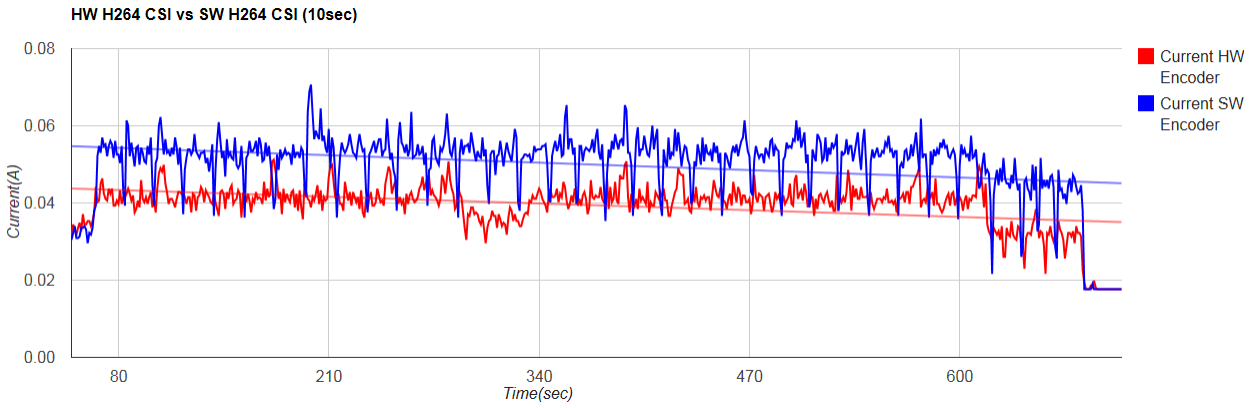

Figure 8.12: Current Consumption Comparison of Video Capturing through CSI-cam Using Hardware and Software H264 (10sec)

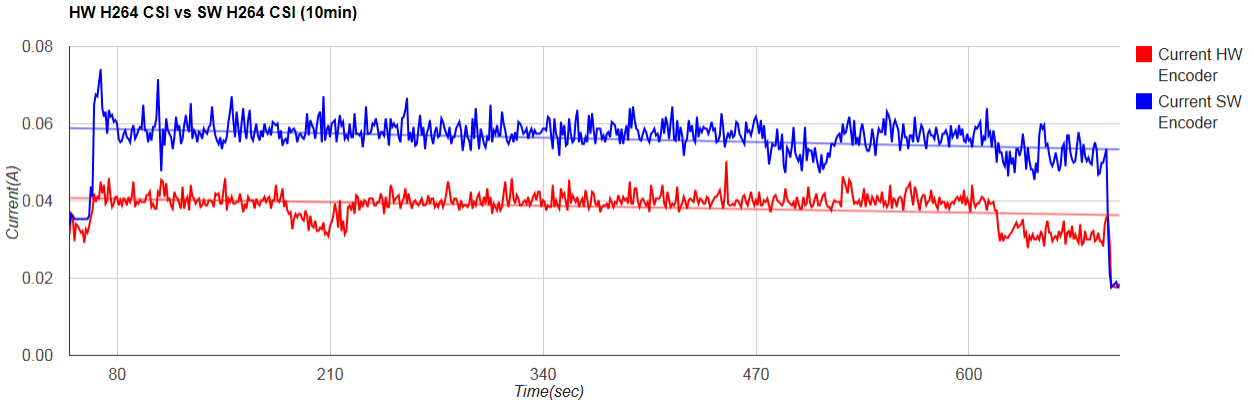

Figure 8.13: Current Consumption Comparison of Video Capturing through CSI-cam Using Hardware and Software H264 (10min)

The plot shows that the current consumption of video captured using hardware encoder is lesser than the software encoder in all 4 cases. It can be seen from the current consumption plots of segmented video captures that there are lots of highly significant peaks and valleys. These are due to the turning ON/OFF of the camera as mentioned before. While for full video, there are less significant peaks and valleys which are not due to any camera turning ON/OFF, they would be due to the processing of the data. Another important factor to note here is that, the gap between the current consumption of video encoded through hardware and software encoder

gets bigger with increasing segment size. As soon as the segment size increases, the processing in a loop increases making the hardware encoder more efficient and thereby making it less current consuming. While for software encoder, as soon as the processing in a loop increases, the average current consumption also increases. Hardware encoding is performed on dedicated processor such as GPU for processing video data while software encoder uses board's CPU and therefore the load on CPU is higher when using software encoder and in turn makes current consumption higher than hardware encoder. Dedicated processors are more efficient because they rely on accelerated, per-function instruction which are job-specific. GPU are highly parallel by definition and video encoding gets benifited a lot by parallelization. Therefore significant difference in current consumption can be seen between hardware and software encoder.

The second set of results are the comparison of video capturing through USB camera and encoding using hardware and software H264 encoder. The capturing is performed using gstreamer commands through USB camera which is placed facing a monitor playing BIG reference video. The video encoding is performed using x264enc(software encoder) and vpu encoder (codec=6, hardware encoder).

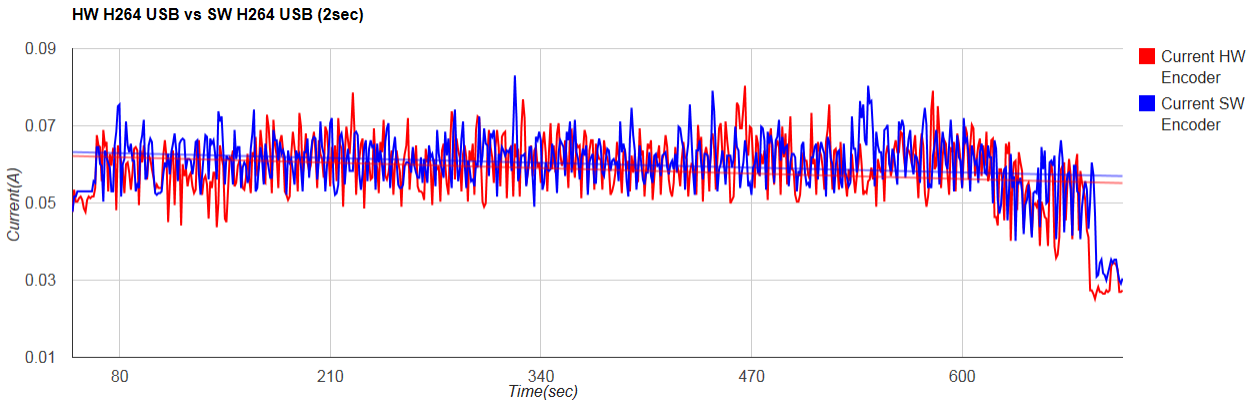

Figure 8.14: Current Consumption Comparison of Video Capturing through USB-cam Using Hardware and Software H264 (2sec)

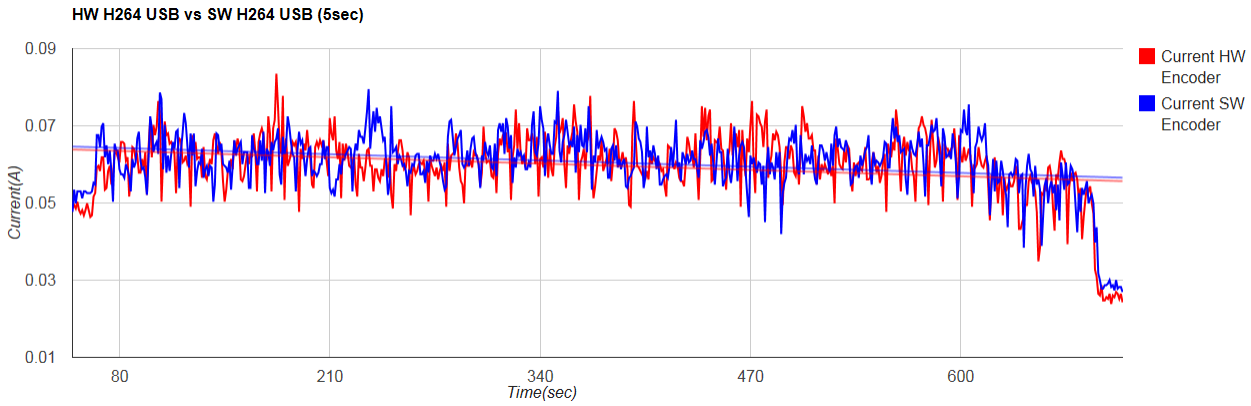

Figure 8.15: Current Consumption Comparison of Video Capturing through USB-cam Using Hardware and Software H264 (5sec)

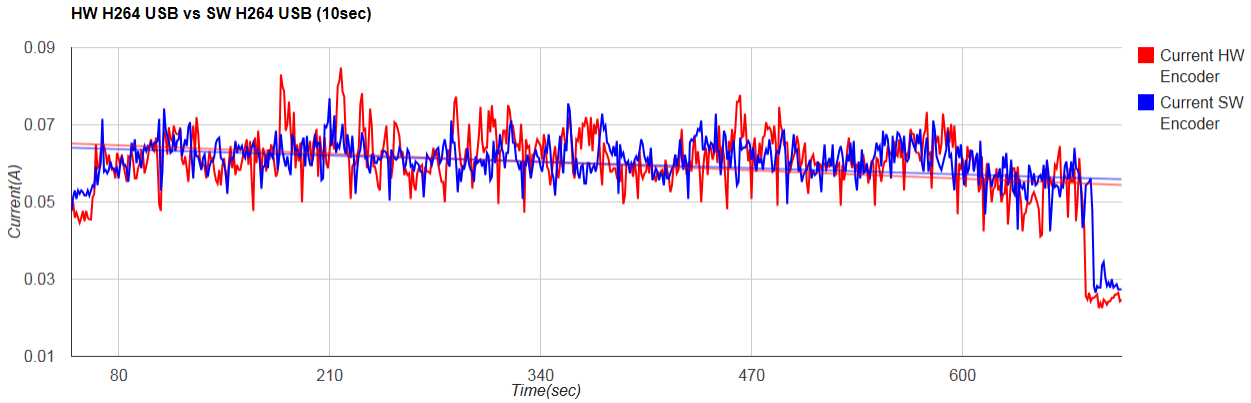

Figure 8.16: Current Consumption Comparison of Video Capturing through USB-cam Using Hardware and Software H264 (10sec)

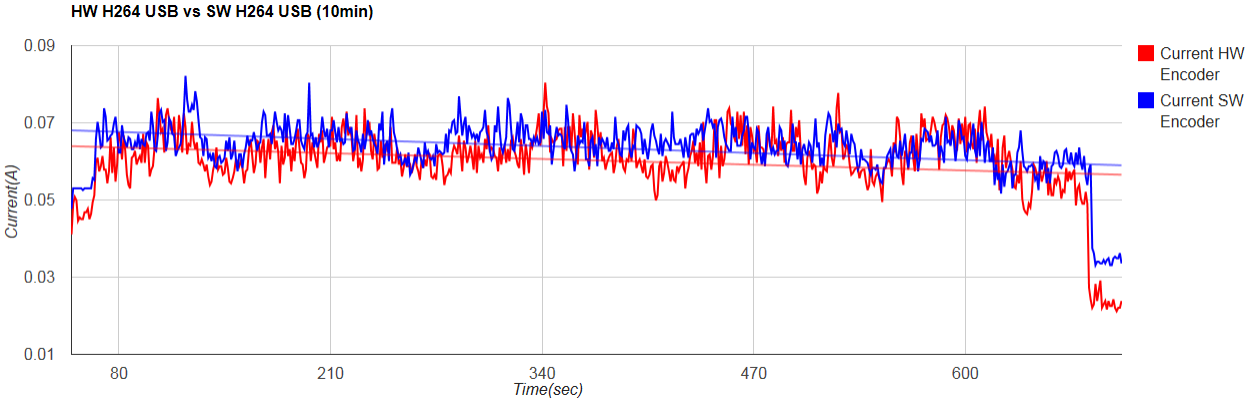

Figure 8.17: Current Consumption Comparison of Video Capturing through USB-cam Using Hardware and Software H264 (10min)

Since USB cameras are power hungry, a upward shift in the current consumption plots can be seen in all of the cases here compared to the CSI camera plots. There is some kind of compression technique applied to the raw data while going through the USB stack, due to which there is no apparent difference seen between the current consumed by hardware or software encoding . The current consumed by USB camera is such that it is shielding the efficiency of hardware encoding. The plot shows that the current consumption of video capturing using USB camera and encoding with hardware encoder is lower compared to encoding with software encoder for 2sec, 5sec and 10min case. For 10sec, the current consumption of hardware encoder is higher initially and then lowers down from half way till the end. Another important factor to note is that the difference in current consumption is very substantial for 10min case (current consumped by hardware encoder is considerably lower than software encoder) , while for segmented video capture the difference in current consumption of hardware and software encoder is very less as mentioned above.

The third set is the current consumption comparison of video captured using CSI and USB camera and encoded by hardware H264 encoder. The capturing is performed using gstreamer commands through CSI and USB cameras, placed infront of monitor running BIG referenc video. The video is encoded using vpu encoder with codec = 6.

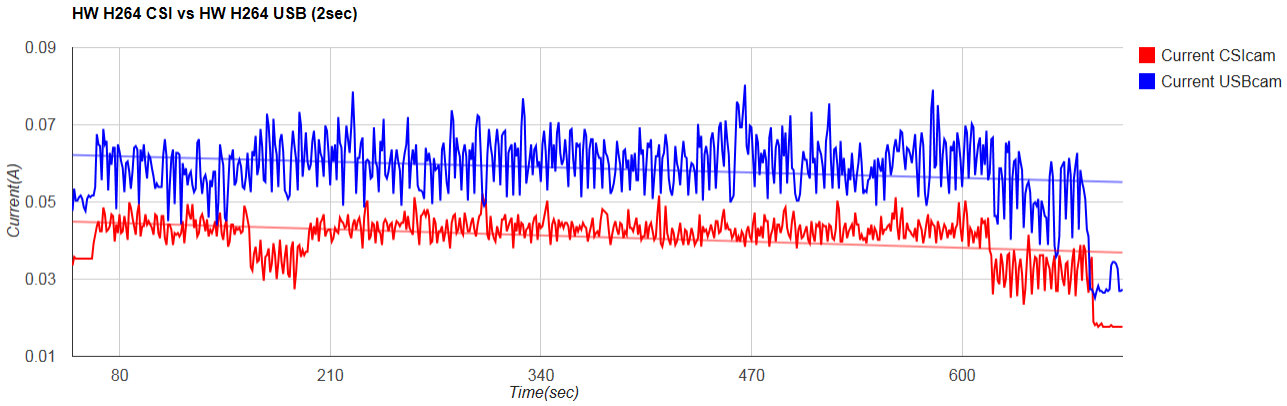

Figure 8.18: Current Consumption Comparison of Video Encoded Using Hardware H264 and Captured Using CSI and USBcam (2sec)

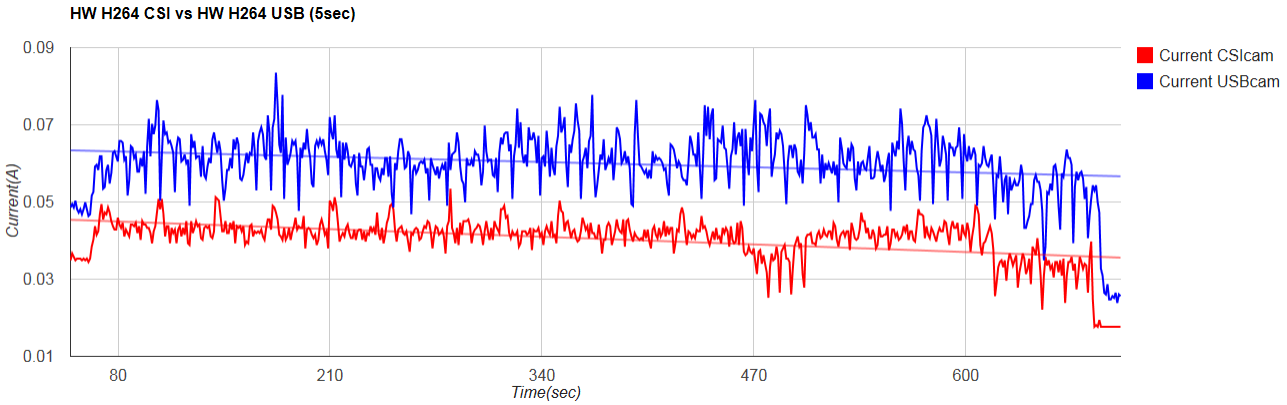

Figure 8.19: Current Consumption Comparison of Video Encoded Using Hardware H264 and Captured Using CSI and USBcam (5sec)

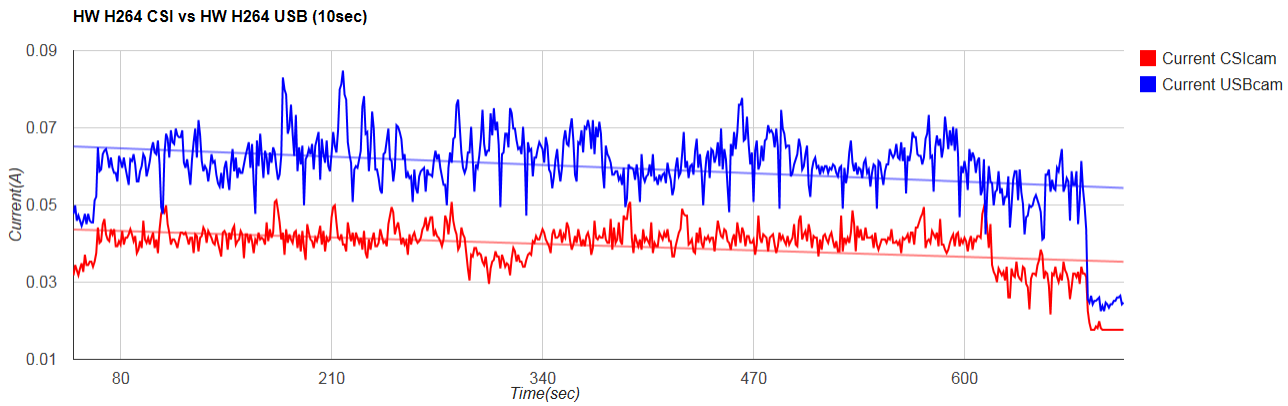

Figure 8.20: Current Consumption Comparison of Video Encoded Using Hardware H264 and Captured Using CSI and USBcam (10sec)

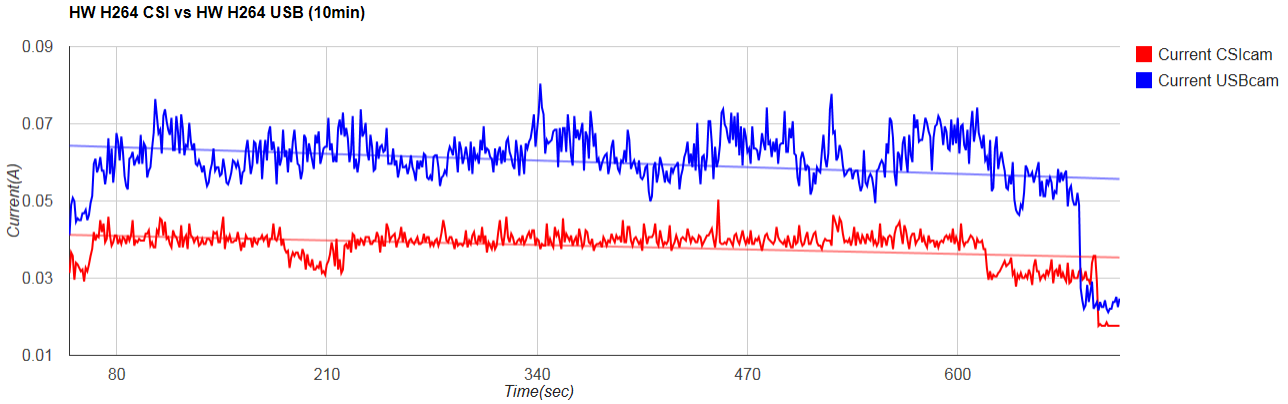

Figure 8.21: Current Consumption Comparison of Video Encoded Using Hardware H264 and Captured Using CSI and USBcam (10min)

It is very clear from the above plots that video captured using CSI camera and encoded with hardware encoder consumes very less power compared to video captured using USB camera also encoded with hardware encoder. It can be seen that the current consumption of video encoded with hardware encoder is getting more efficient(i.e. it is consuming less current as the segment size increases) and this gap is most apparent for the full video case. As mentioned earlier, USB camera does a software hand-off of raw data to hardware encoder, while CSI camera has a direct

path to the hardware encoder and therefore there is a big gap in the average current consumption.

The last set of data here is the current consumption comparison of video captured using CSI and USB camera and encoded by software H264 encoder. The capturing is performed using gstreamer commands through CSI and USB cameras, placed in-front of monitor running BIG reference video. The video is encoded using x264 encoder.

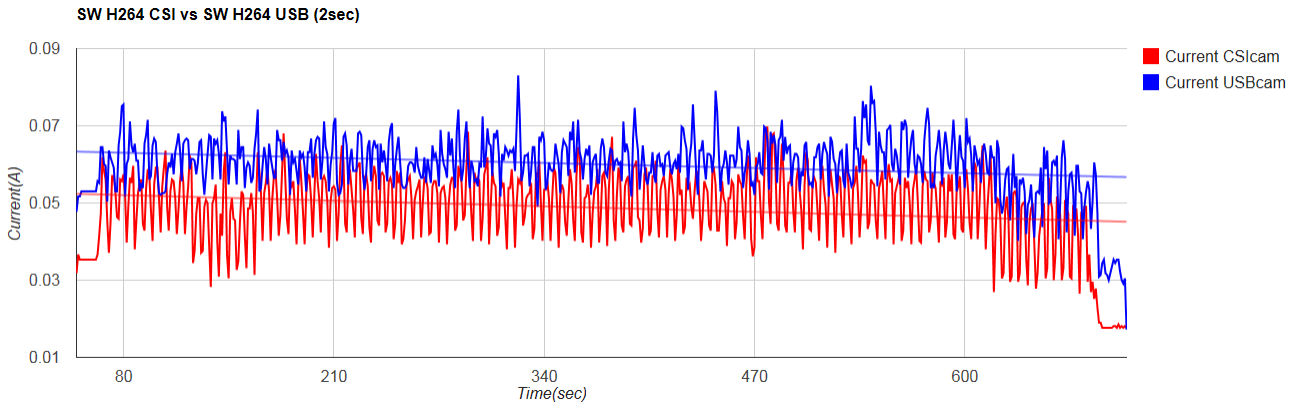

Figure 8.22: Current Consumption Comparison of Video Encoded Using Software H264 and Captured Using CSI and USBcam (2sec)

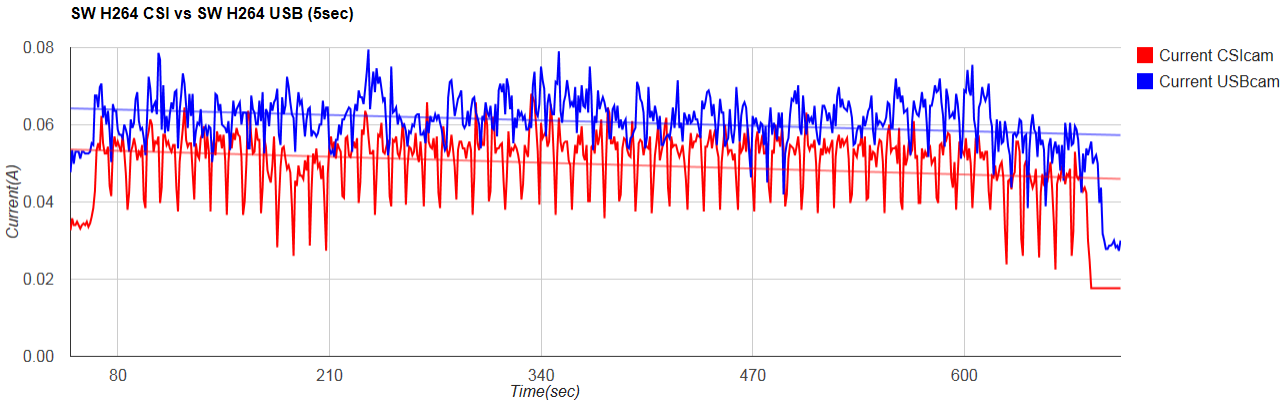

Figure 8.23: Current Consumption Comparison of Video Encoded Using Software H264 and Captured Using CSI and USBcam (5sec)

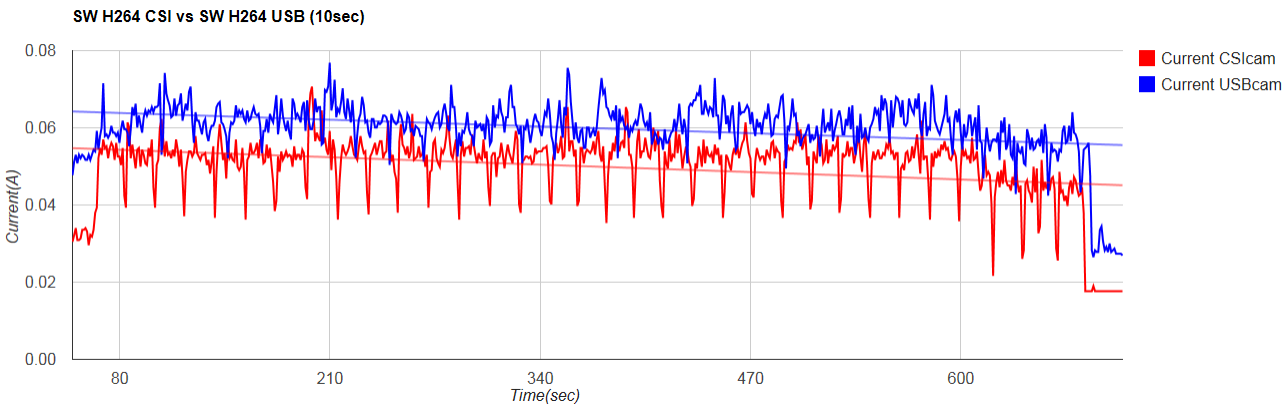

Figure 8.24: Current Consumption Comparison of Video Encoded Using Software H264 and Captured Using CSI and USBcam (10sec)

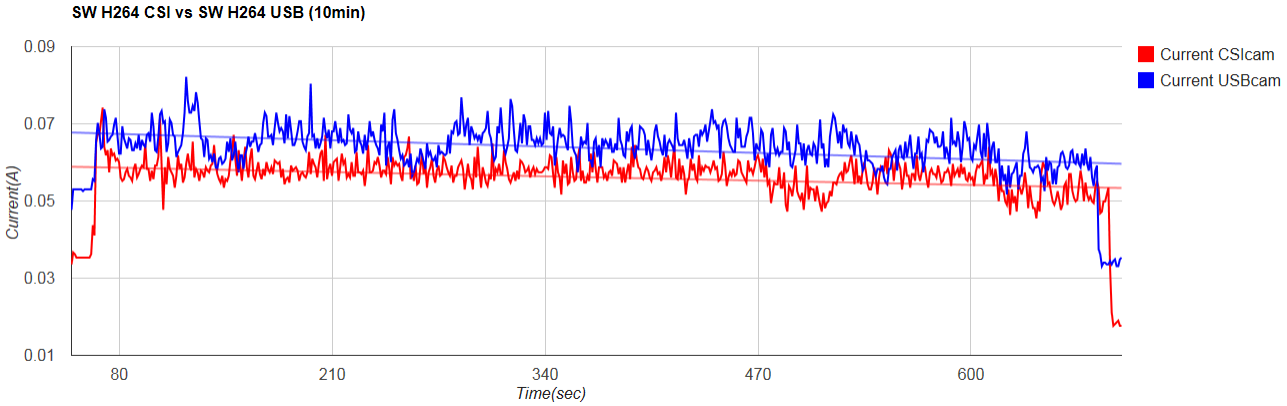

Figure 8.25: Current Consumption Comparison of Video Encoded Using Software H264 and Captured Using CSI and USBcam (10min)

The above graphs shows that the current consumption of video capturing through CSI camera is less current consuming than USB camera. The current consumption graph of video capturing through USB camera with software encoding hasn't shifted a bit from the graph with hardware encoding, while the graph of CSI camera has shifted upwards by 1mA. As mentioned earlier, USB camera has do a software hand-off and is very power hungry, therefore the efficiency of hardware encoder

is not visible and the average consumption is similar. For CSI camera, the raw data from CSI camera does a hardware hand-off to the software encoder on the board, but it is not that efficient as hardware encoding.

.

After different comparisons with H264 encoders, there were some results collected with other encoders. The next set of results are the comparison of current consumption of capturing video through jpeg and mpeg4 hardware and software encoder using USB and CSI camera. The first encoder is a avi (Audio Video Interleaved) encoder. The videos were captured using gstreamer command and encoded using jpegenc(software encoder) and vpuenc(codec=0, hardware encoder).

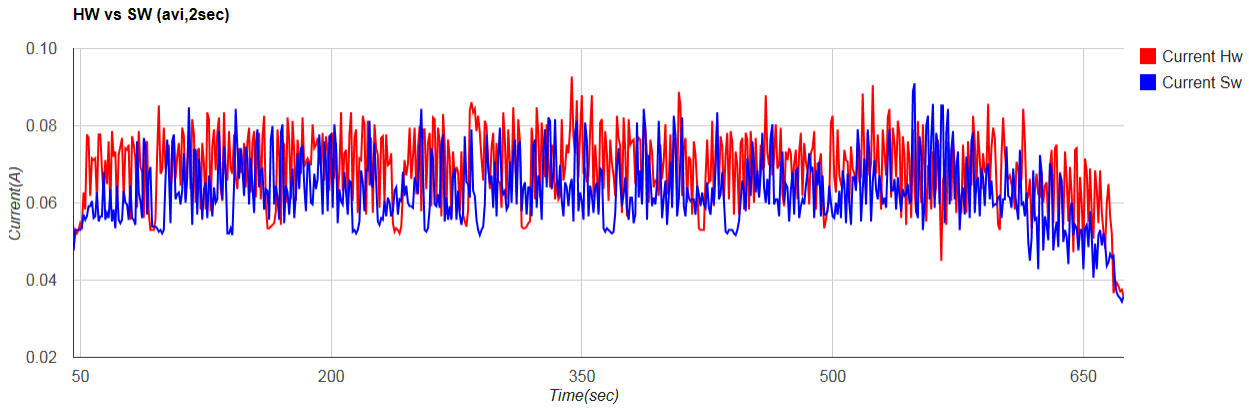

Figure 8.26: Current Consumption Comparison of Capturing Video through Hardware and Software avi Encoder (2sec)

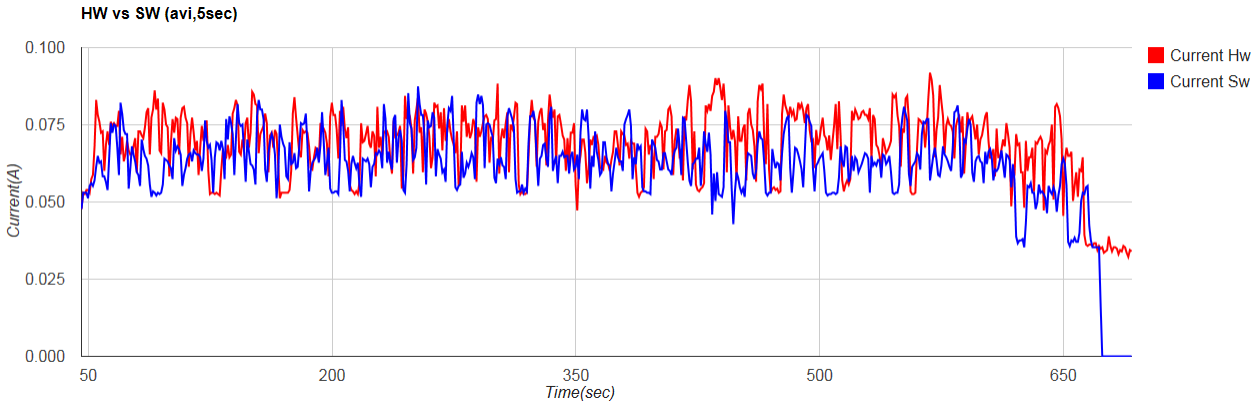

Figure 8.27: Current Consumption Comparison of Capturing Video through Hardware and Software avi Encoder (5sec)

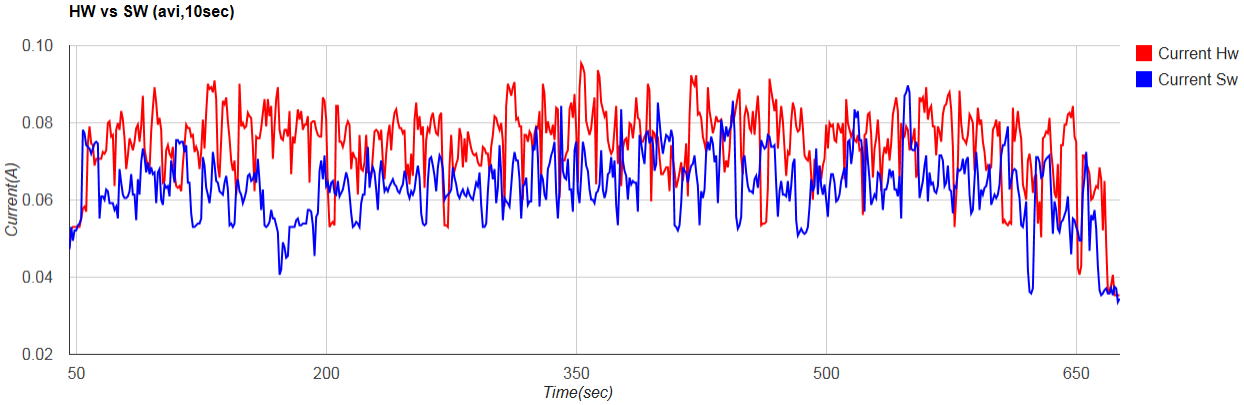

Figure 8.28: Current Consumption Comparison of Capturing Video through Hardware and Software avi Encoder (10sec)

The current consumption of video captured through hardware encoder is expected to be lower than that of software encoder because hardware-based encoders have dedicated,limited-purpose processors such as GPU to run a specific algorithms while the general-purpose processors which runs software encoders are designed to perform several other functions. Here for avi encoder, the video encoded by hardware encoder consumes more current compared to software encoder. The results are not as expected because the encoding performed by hardware avi encoder is not as efficient as it is with software encoder. The second encoder for current consumption

comparison between hardware encoder and software encoder is mpeg4. The videos were captured using ffmpeg commands and encoded with mpeg4(software encoder) and vpuenc(codec=12, hardware encoder).

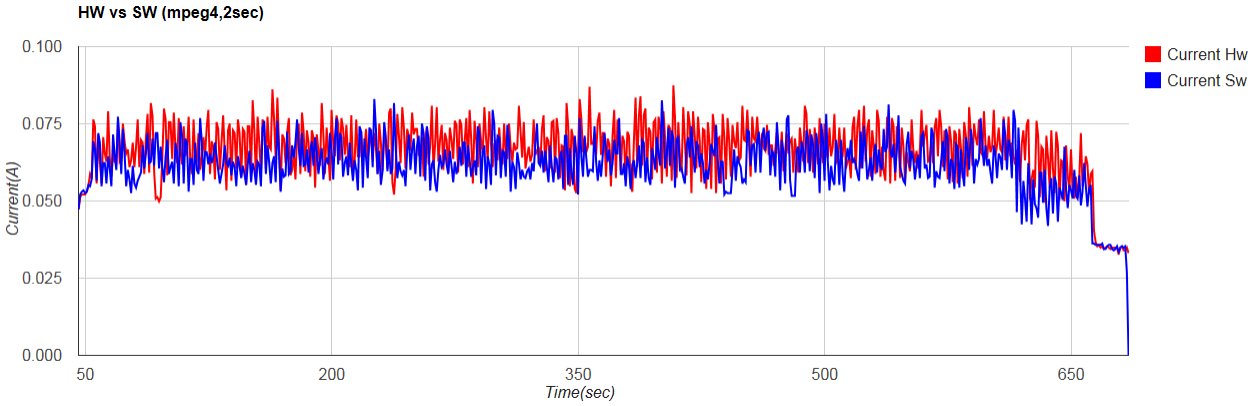

Figure 8.29: Current Consumption Comparison of Capturing Video through Hardware and Software mpeg4 Encoder (2sec)

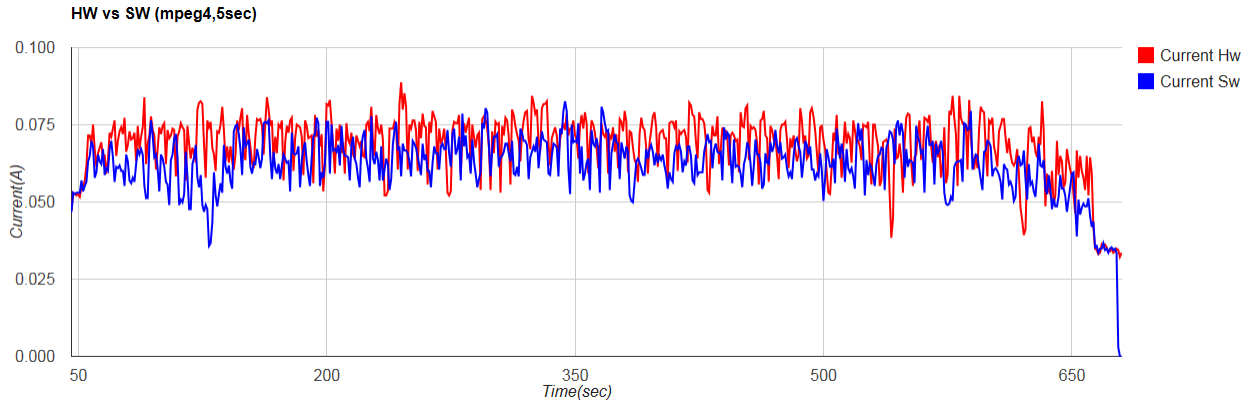

Figure 8.30: Current Consumption Comparison of Capturing Video through Hardware and Software mpeg4 Encoder (5sec)

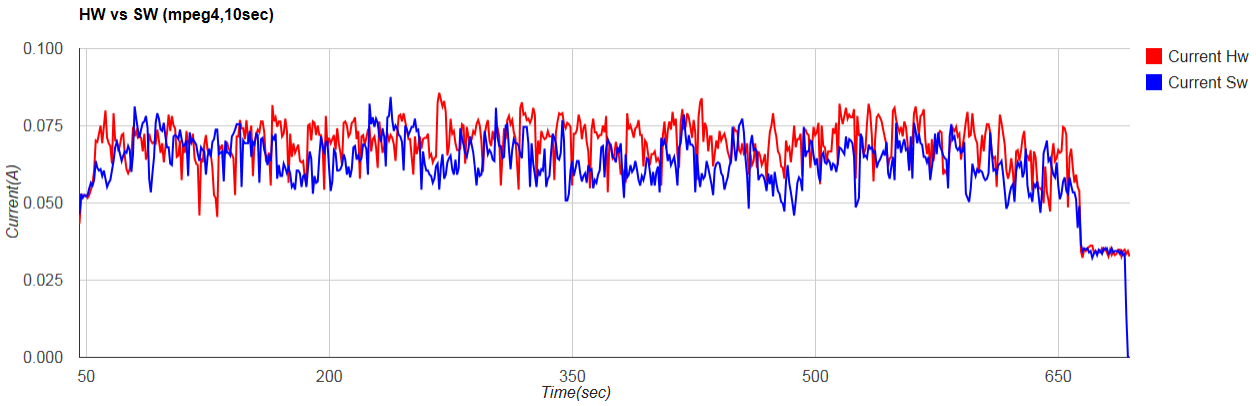

Figure 8.31: Current Consumption Comparison of Capturing Video through Hardware and Software mpeg4 Encoder (10sec)

The current consumption of hardware mpeg4 encoder is higher than software mpeg4 encoder just as above with avi encoder. The results of both encoders are not good as were with H264, where video encoded by hardware encoder consumed lesser current than software encoder in all different scenarios. For future work, there is retrospect required on why hardware encoders with USB cameras are not performing efficiently. The comparison between hardware and software encoder was also performed with CSI camera but for 2sec segments only, to see if encoders behave the same as with USB camera. Here is the comparison of hardware vs software encoder for avi.

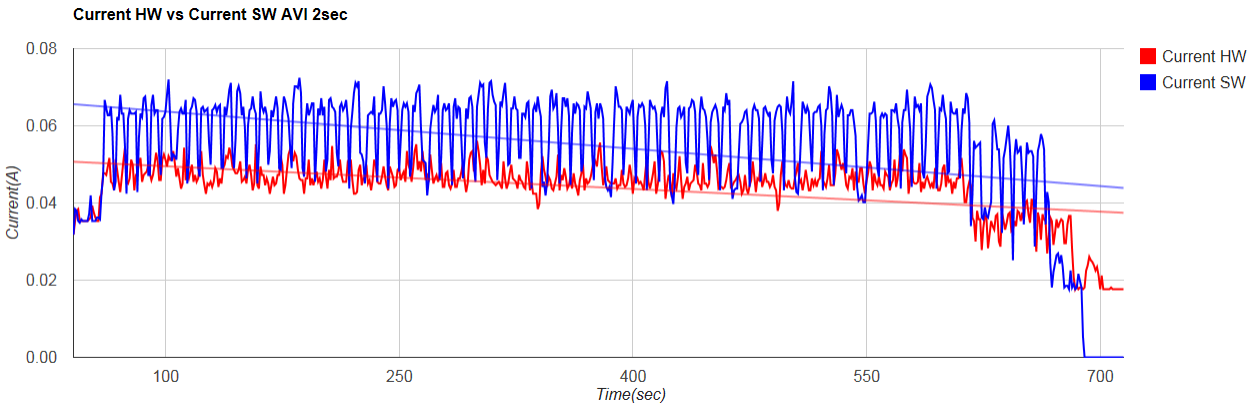

Figure 8.32: Current Consumption Comparison of Capturing Video through Hardware and Software avi Encoder (2sec)

The current consumption of hardware encoder is lower compared to software encoder for avi, and the results are very encouraging compared to the results obtained with USB camera. Although the hardware encoder has out performed the software encoder, the gap between the current consumption of these encoders with CSI camera capture is not substantial as it was with H264 encoder. It should be noted that these results have shifted 2mA below the results of USB camera. One of the reason, this results looks better is due to the use of CSI camera. Since the CSI camera's stack is smaller and there is a direct path from camera module to the GPU , the data gets faster for the processing. Secondly, the software encoding of CSI raw data hasn't changed a bit except for shifting downward, while hardware encoding of CSI raw data is very efficient than hardware encoding of USB raw data. These are the two reasons for good results. The second encoder for comparison of hardware and software encoder is mpeg4.

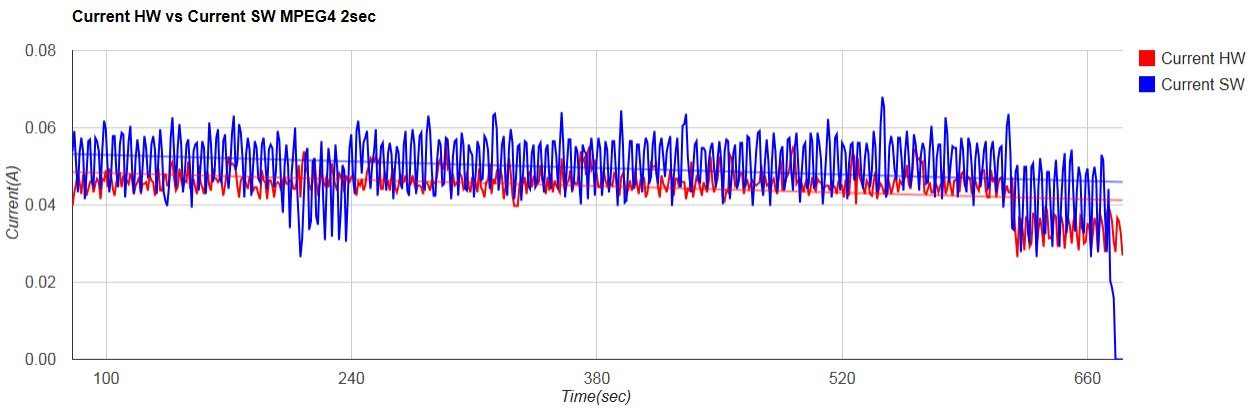

Figure 8.33: Current Consumption Comparison of Capturing Video through Hardware and Software mpeg4 Encoder (2sec)

The current consumption of capturing video with CSI camera and encoding with hardware encoder is lower than that of with software encoder for mpeg4 case. Just as for avi case, here also the software encoding of CSI raw data is very similar to USB raw data, except that this has shifted down. While the current consumption graph of hardware encoding of CSI raw data is very suppressed and efficient compared to the graph of hardware encoding of USB raw data. Here also, the current consumption has shifted down by more than 2mA.

The next set of results are the playback of HLS and WVSNP videos under different browsers and different network scenarios on JWplayer and WVSNP player. As mentioned in the TEST BED section, before taking results, a reference video was created and then these videos were transcoded into HLS and WVSNP videos. The first set of results are playback of WVSNP and HLS videos on Mac. On Mac systems, by default Safari web-browser uses HLS and so when playback is performed of HLS videos on WVSNP player, the web-browser starts using native HLS and since it is properitary I don't know what processing is happening at the backend. Therefore for fair comparison, HLS is played on JWplayer6 on Chrome web-browser on Mac. WVSNP videos were played on WVSNP player on chrome web-browser. For LIVE

capturing and real-time playing to work, the client and board needs to be connected. Since DHCP would provide different IP address everytime the board tries to connect after a reboot, a router with a static IP is set. And a board is also set with a static IP address. In the first case, board is connected to the router via ethernet and client is connected via wifi to the router.

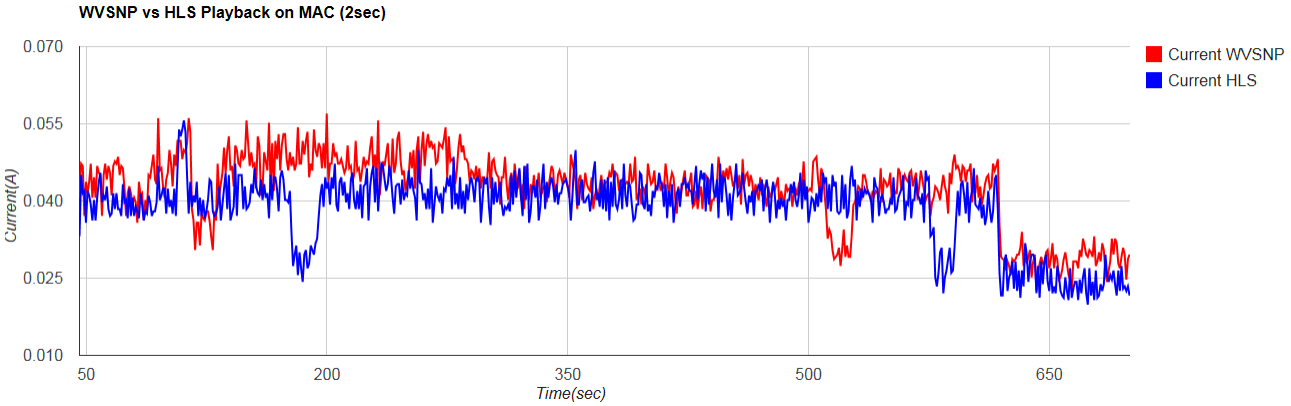

Figure 8.34: Current Consumption of Playback of WVSNP and HLS Videos on Mac (2sec, Ethernet)

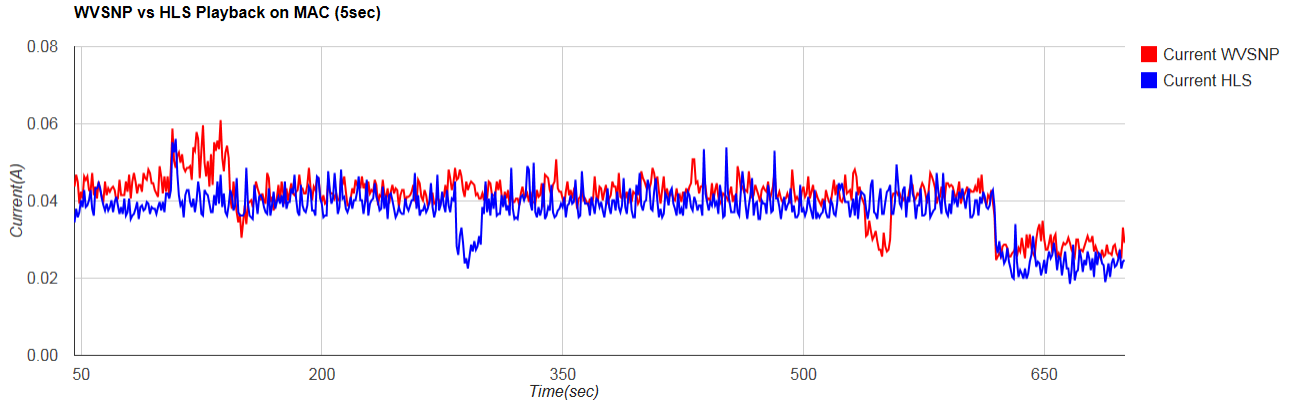

Figure 8.35: Current Consumption of Playback of WVSNP and HLS Videos on Mac (5sec, Ethernet)

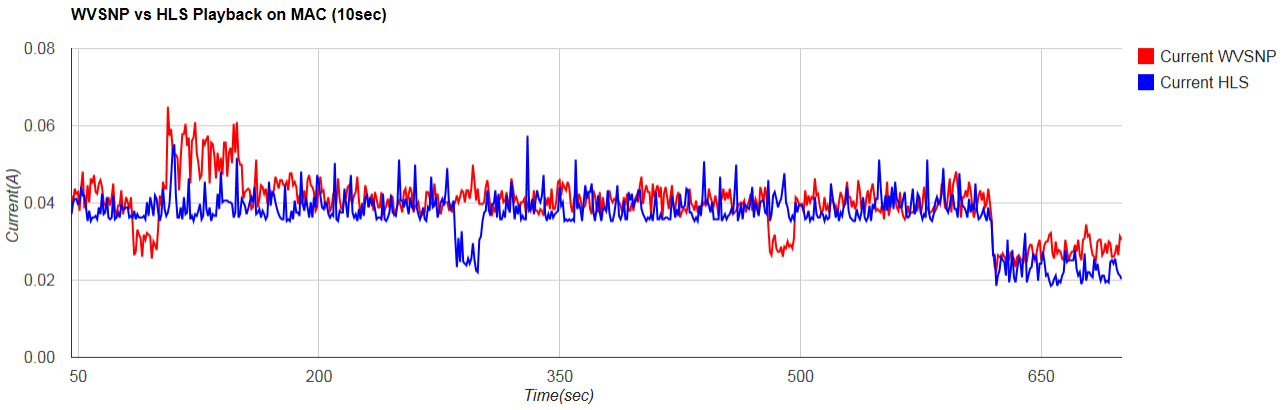

Figure 8.36: Current Consumption of Playback of WVSNP and HLS Videos on Mac (10sec, Ethernet)

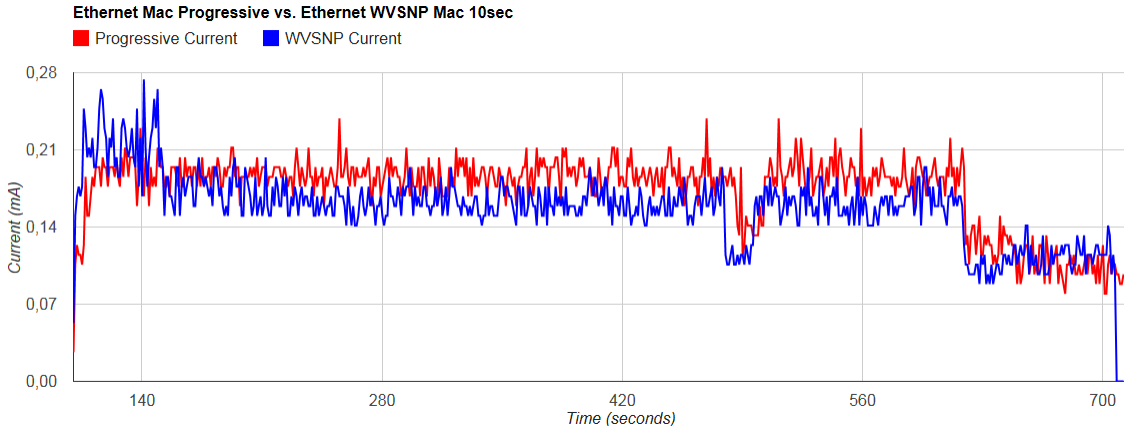

Figure 8.37: Current Consumption of Playback of Progressive WVSNP Video on Mac (Ethernet)

The current consumption of WVSNP is higher than HLS during the first five minutes of the plot, because WVSNP buffers all the segments while it is playing. While in HLS, it buffers some segments while it is playing the video, then it stops buffering and then starts again. But after the WVSNP buffering is complete, it is consuming lesser current than HLS in all three cases. The last plot shows the current consumption comparison of progressive(i.e. it is a playback of direct 10 minute video) WVSNP video and 10 sec segmented video. It can be seen from this graph that

progressive download video consumes more current than segmented video and other segmented videos results also does better than progressive download video. The next set of results are playback on WVSNP and HLS videos on Windows. As above, board is connected to the router via ethernet and client is connected via wifi to the router.

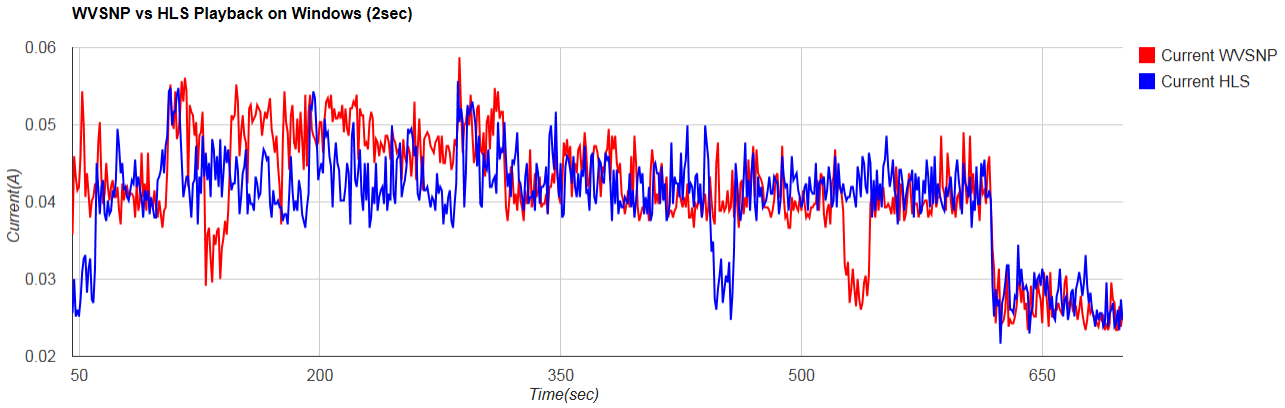

Figure 8.38: Current Consumption of Playback of WVSNP and HLS Video on Windows (2sec, Ethernet)

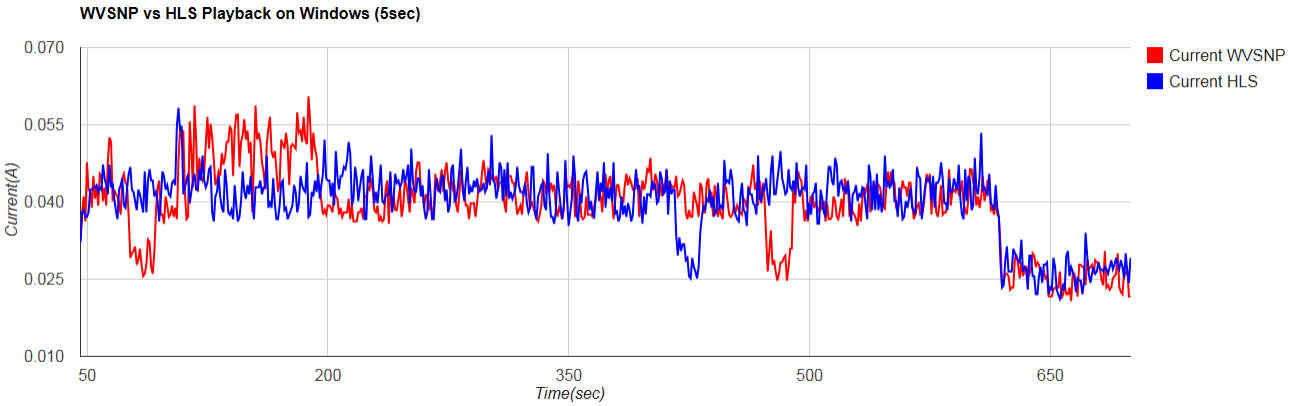

Figure 8.39: Current Consumption of Playback of WVSNP and HLS Video on Windows (5sec, Ethernet)

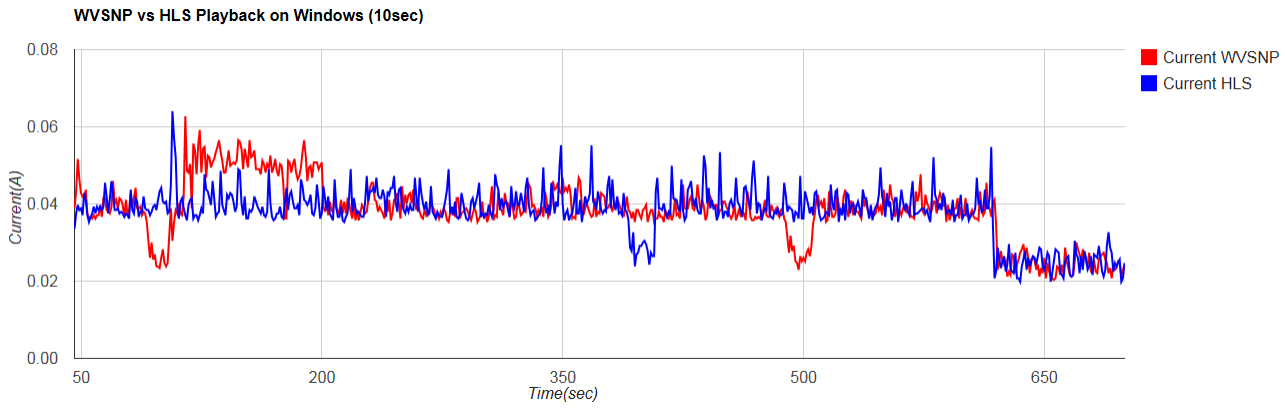

Figure 8.40: Current Consumption of Playback of WVSNP and HLS Video on Windows (10sec, Ethernet)

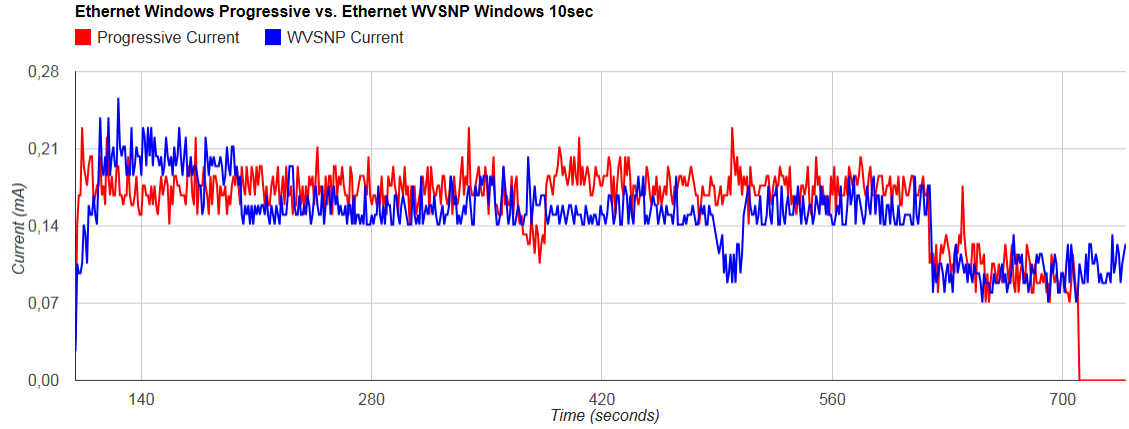

Figure 8.41: Current Consumption of Playback of Progressive WVSNP Video on Windows (Ethernet)

As above, the current consumption of WVSNP is higher than HLS during the first few minutes of the capture because WVSNP buffers all the segments during that time and then the current consumption reduces below HLS. For HLS, all segments are not buffered as WVSNP, here some initial segments are buffered, then it stops buffering and resumes it again after some pause. The current consumption graph for WVSNP and HLS are very close compared to the Mac. The last plot is the current consumption comparison of progressive download and WVSNP 10 second playback.

It can be seen that the current consumption of progressive download is higher than WVSNP segment videos playback as above. The next set of results are playback on WVSNP and HLS videos on Mac, but this time board and client both are connected to the router via wifi.

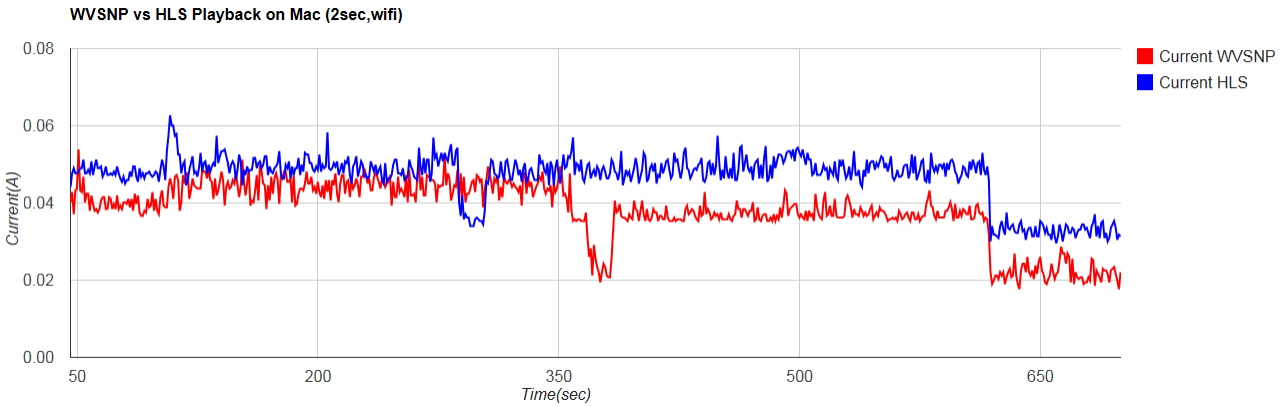

Figure 8.42: Current Consumption of Playback of WVSNP and HLS Video on Mac (2sec, Wifi)

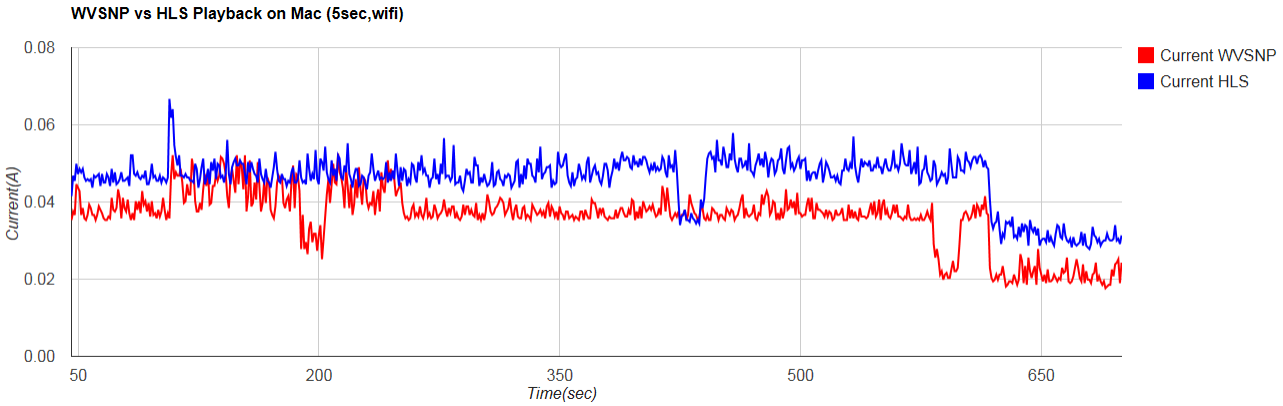

Figure 8.43: Current Consumption of Playback of WVSNP and HLS Video on Mac (5sec, Wifi)

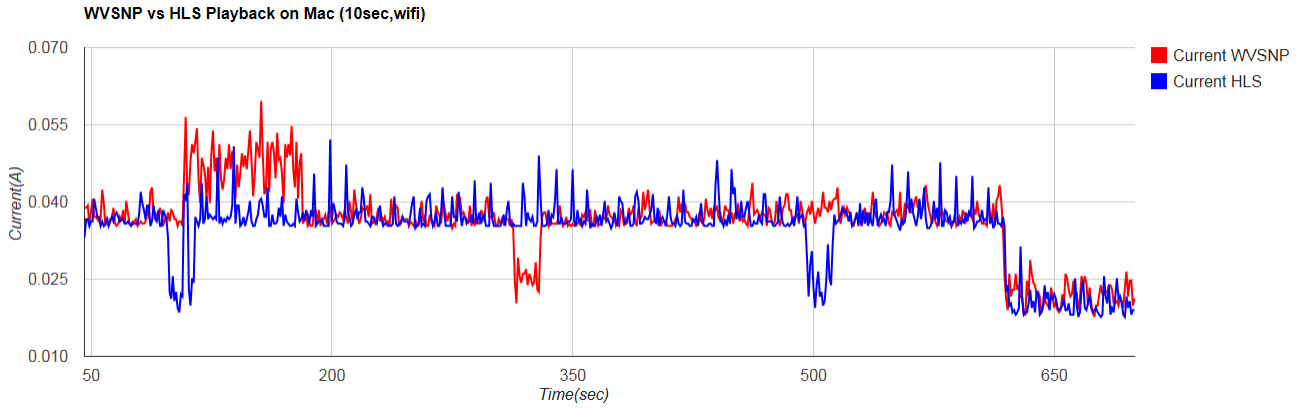

Figure 8.44: Current Consumption of Playback of WVSNP and HLS Video on Mac (10sec, Wifi)

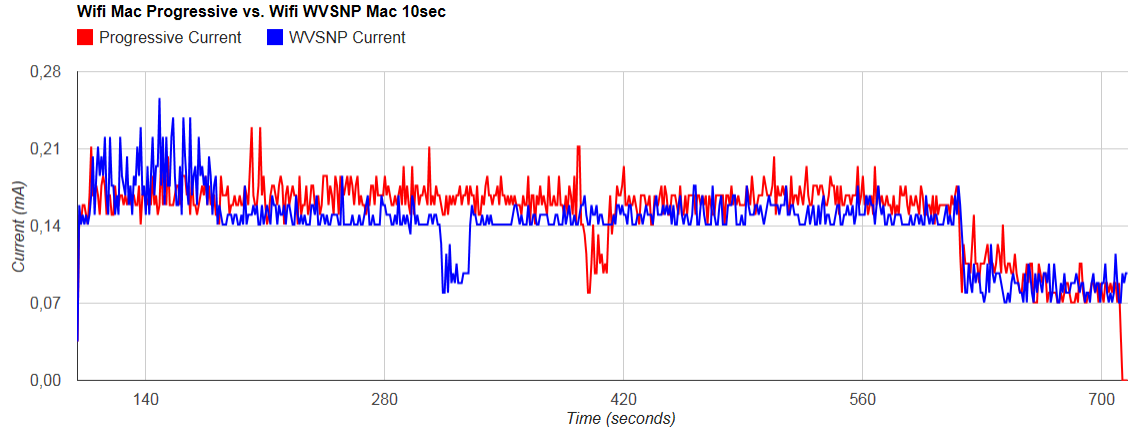

Figure 8.45: Current Consumption of Playback of Full WVSNP and 10sec Segmented WVSNP Video on Mac (Wifi)

The current consumption of playing WVSNP video is lower over the whole capture than HLS in both 2sec and 5sec case, while in the 10sec case, the current consumption is higher only during the buffering time and then current consumption is again lower than HLS. And, last plot is comparison of progressive download and WVSNP 10 second playback on Mac. Again, the current consumption of segment video playback is higher during the buffering time and then the consumption remains lower than progressive download.

The last set of results are playback of WVSNP and HLS videos on Windows and here board and client like above are connected to the router via wifi.

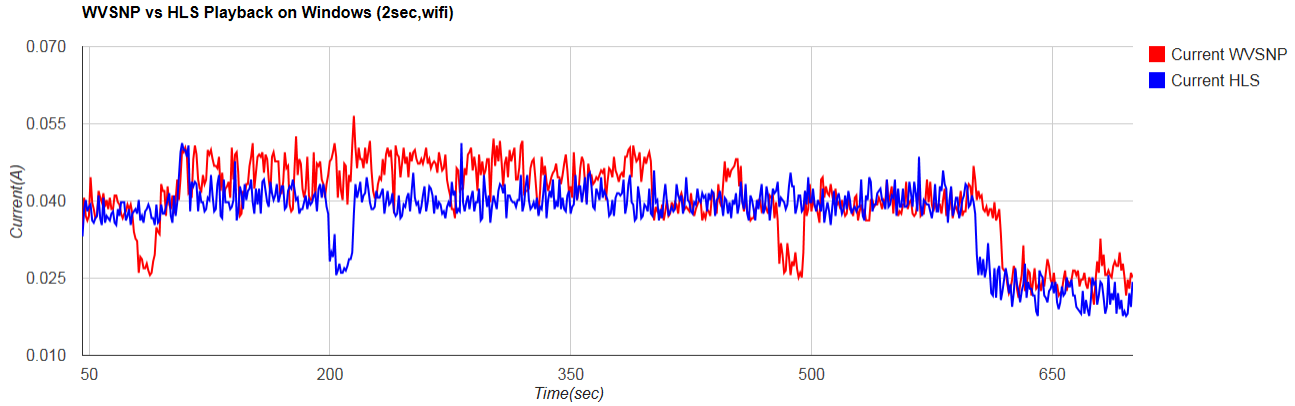

Figure 8.46: Current Consumption of Playback of WVSNP and HLS Video on Windows (2sec, Wifi)

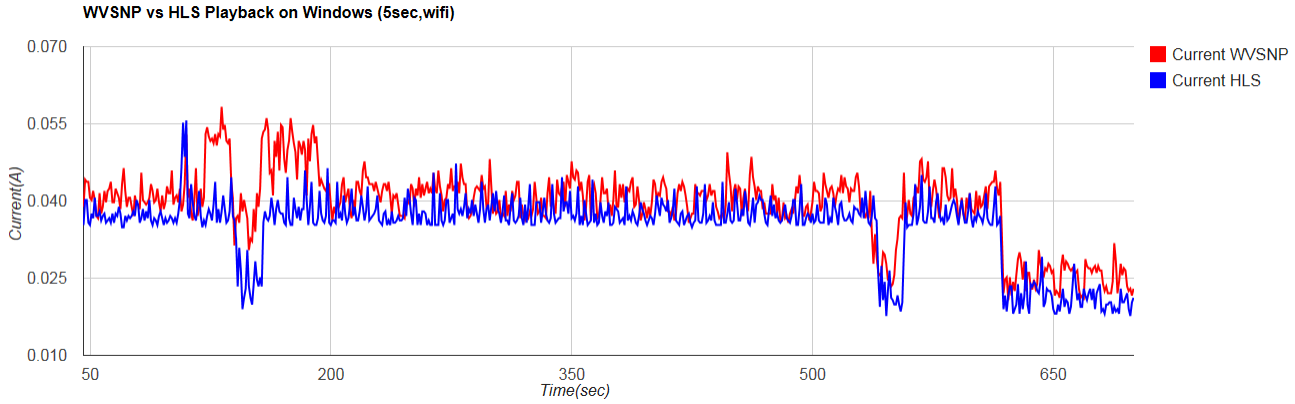

Figure 8.47: Current Consumption of Playback of WVSNP and HLS Video on Windows (5sec, Wifi)

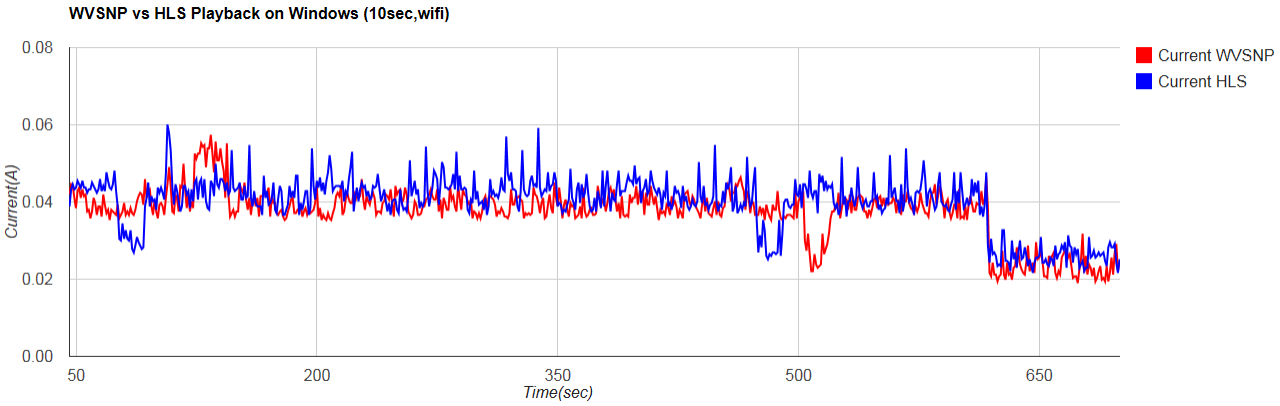

Figure 8.48: Current Consumption of Playback of WVSNP and HLS Video on Windows (10sec, Wifi)

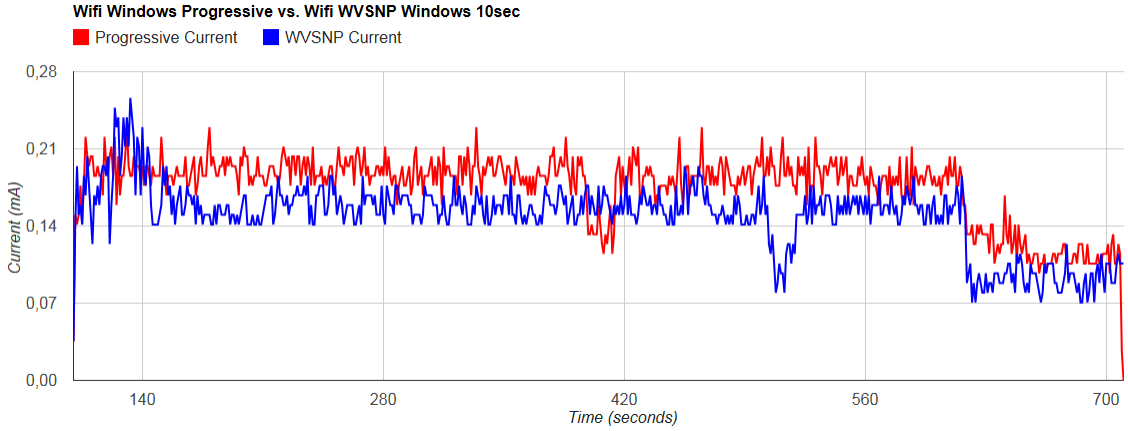

Figure 8.49: Current Consumption of Playback of Progressive WVSNP Video on Windows (Wifi)

The current consumption of WVSNP is little bit higher than HLS in 2sec and 5sec, but the graph follows very closely to HLS over the capture. While in 10sec case, the current consumption is only higher during buffering time and then again it is lower than HLS over the capture. And the last plot shows the comparison of WVSNP progressive download and WVSNP 10 seconds playback on Windows.The current consumption of WVSNP 10 second segments playback is lower than progressive download over the capture.

Here are the results of live capturing plus real-time playing of WVSNP and HLS videos on Windows and Mac. In the first case, the board is connected to the router through ethernet and client is connected through WifI to the router. For live-capturing plus real-time playing case, the results were taken for 5sec case only. The encoding is done using ffmpeg libraries and as before in playback, HLS videos are played back on Jwplayer and WVSNP videos are played on WVSNP player. The LIVE capturing is performed by placing USB camera in front of a monitor continously running a BIG reference video.

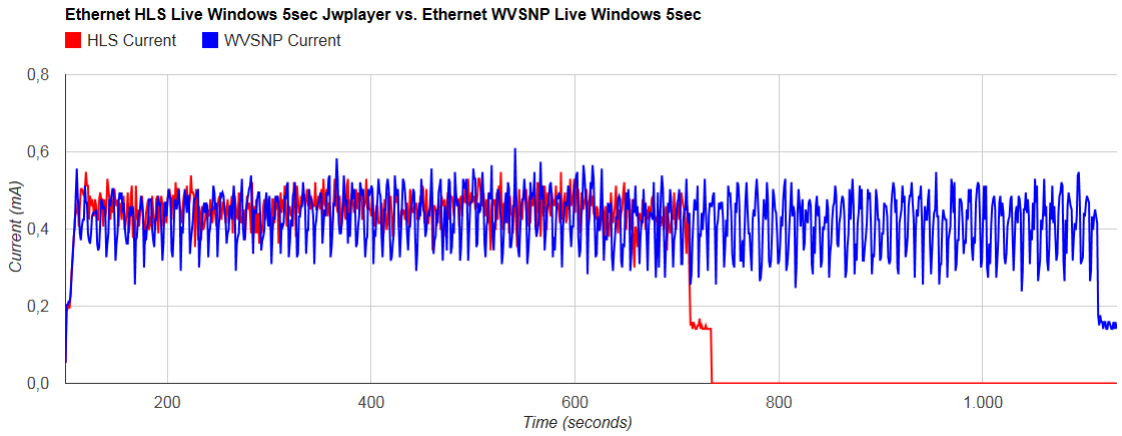

Figure 8.50: Current Consumption of Live Capturing + Real-Time Playing of WVSNP and HLS Video on Windows (Ethernet)

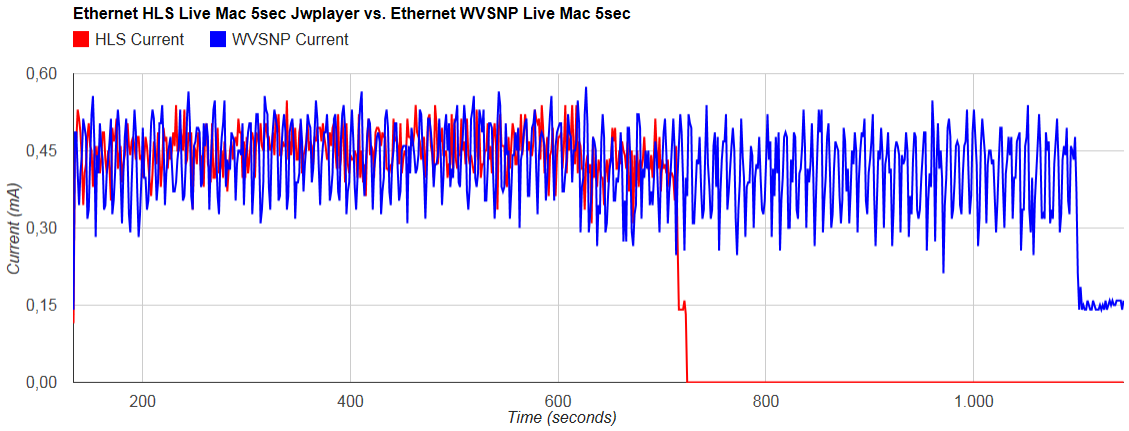

Figure 8.51: Current Consumption of Live-Capturing + Real-Time Playing of WVSNP and HLS Video on Mac (Ethernet)

Before WVSNP player starts playing back the videos, it waits for three segments to be ready. While playing back the videos, player keeps on playing the video until the next video segments get before the last segment is completed playing. If the current segment is completed and next one is not eady yet, then player waits again for three segments before resuming playback again. For HLS playback, the JWPlayer starts playback as soon the first segment is ready and in case if again the segments are not ready, it buffers one segment and resumes playback again. In case of HLS, the video encoding was fast enough to have videos ready before the current segment completed playing, therefore the live-capturing and real-time playing gets completed almost on time. While for WVSNP case, due to slow encoding, next segments are never ready, therefore the player keeps on pausing to get three segments before starting playback again. Another important factor to note here is that, although WVSNP and HLS are using ffmpeg libraries, HLS is using ffmpeg's capabilities of segmenting the video while capturing is on. For WVSNP, ffmpeg loop capture is used wherein the each time the camera is turned ON, capturing performed and again the camera is turned OFF. Due to loop capture, plenty of time is wasted in turning ON/OFF

of the camera. The current consumption of live-capturing and real-time playing of WVSNP and HLS videos on Windows and Mac are almost same. It is very hard to see from these plots if WVSNP current consumption is higher or lower than HLS, therefore here are those two plots again with trendline, and this time the plot only shows the comparison until HLS capture.

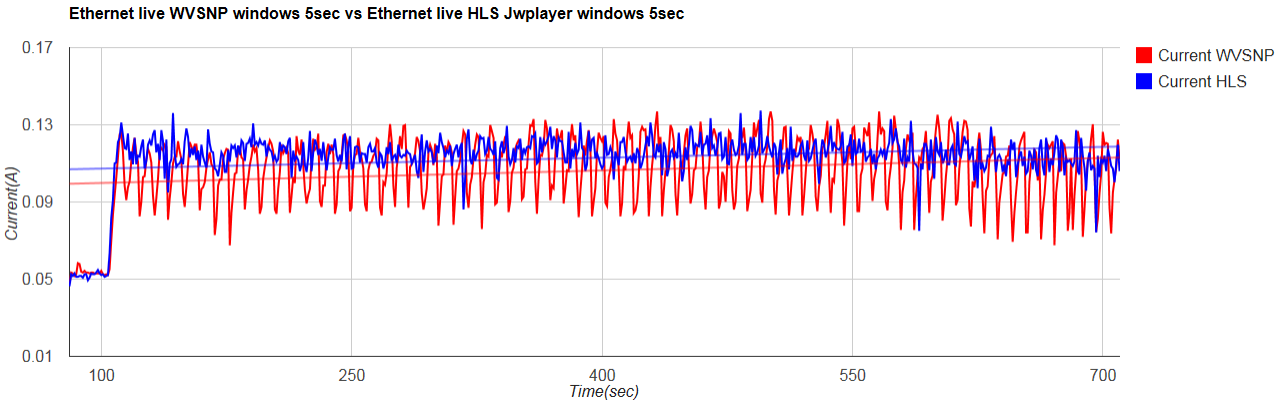

Figure 8.52: Current Consumption of Live Capturing + Real-Time Playing of WVSNP and HLS Video on Windows with Trendline (Ethernet)

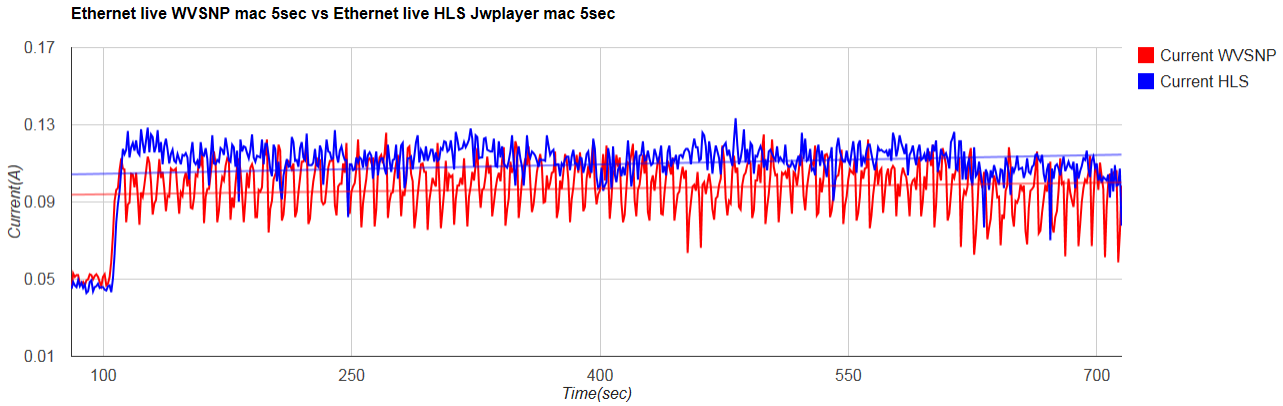

Figure 8.53: Current Consumption of Live-Capturing + Real-Time Playing of WVSNP and HLS Video on Mac with Trendline (Ethernet)

It is now very clear from this graph that the current consumption of WVSNP live-capturing plus real-time playing on Windows and Mac is lower than HLS.

After live-capturing plus real-time playing in ethernet connection case, here are the

results of live-capturing plus real-time playing of WVSNP and HLS videos on Windows and Mac for Wifi connection case. The encoding is done using ffmpeg libraries and, board and client are connected to the router via wifi.

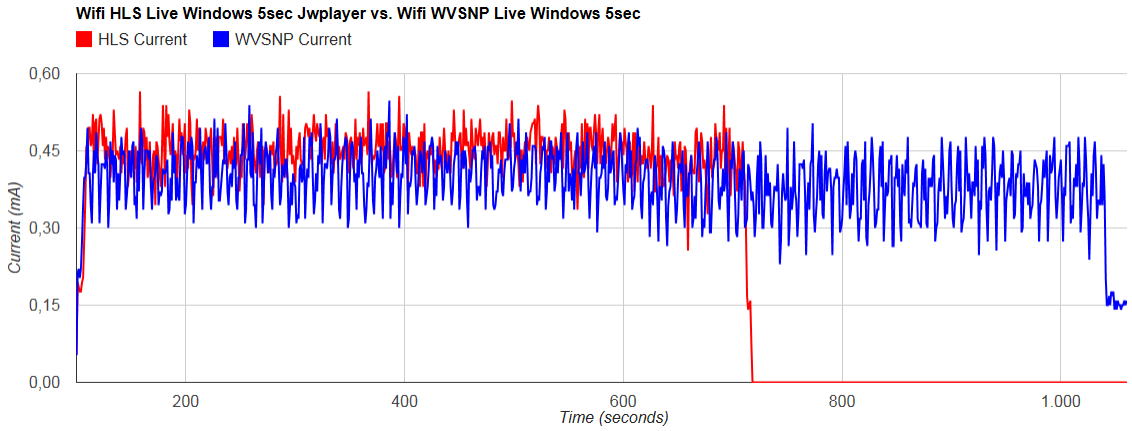

Figure 8.54: Current Consumption of Live-Capturing + Real-Time Playing of WVSNP and HLS Video on Windows (Wifi)

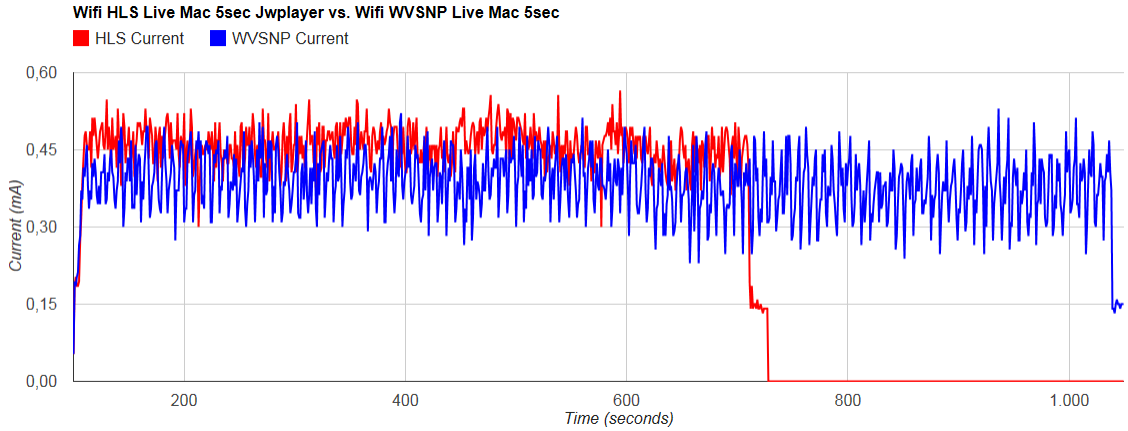

Figure 8.55: Current Consumption of Live-Capturing + Real-Time Playing of WVSNP and HLS Video on Mac (Wifi)

The results are very clear for wifi connection case compared to ethernet case above. Here, the current consumption of live-capturing and real-time playing of HLS videos is higher than WVSNP videos over the capture.

After playback and, live-capturing and real-time playing on Windows and Mac, here are some results performed on Ubuntu. All of the results performed until now were all with BIG video option, so in ubuntu for both playback and, live capturing and real-time playing, results were performed with BIG and SMALL video option. The difference between SMALL and BIG video option is in the resolution, frames per second and bitrate. It is expected that videos of SMALL option will consume less current when playing back or live capturing. For all ubuntu tests, board and the client are connected to the router via Wifi connection. The WVSNP videos were played on WVSNP player on chrome web-browser and HLS videos were played on JwPlayer6 on chrome web-browser. First, here are the current consumption comparison of playing back a WVSNP and HLS video of BIG option.

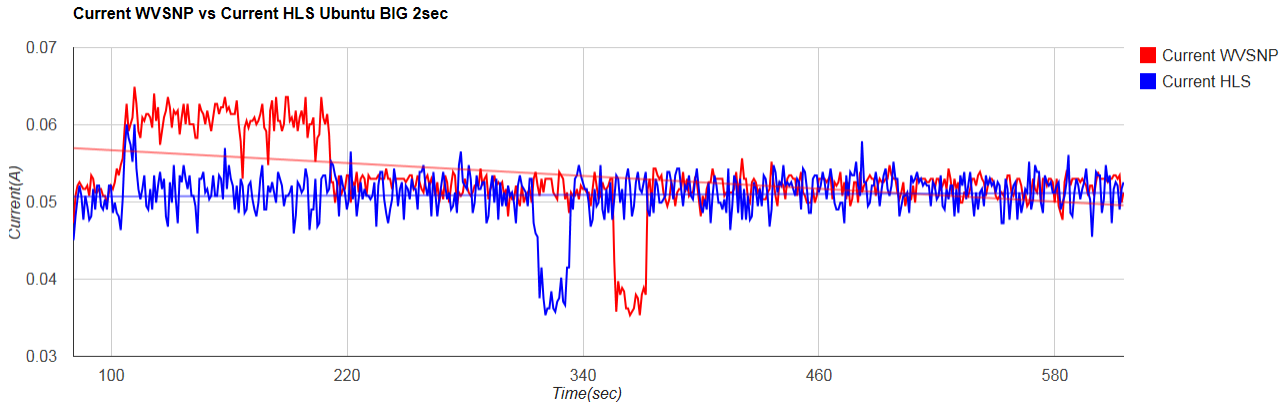

Figure 8.56: Current Consumption of Playbacking HLS and WVSNP BIG Videos on Ubuntu (2sec)

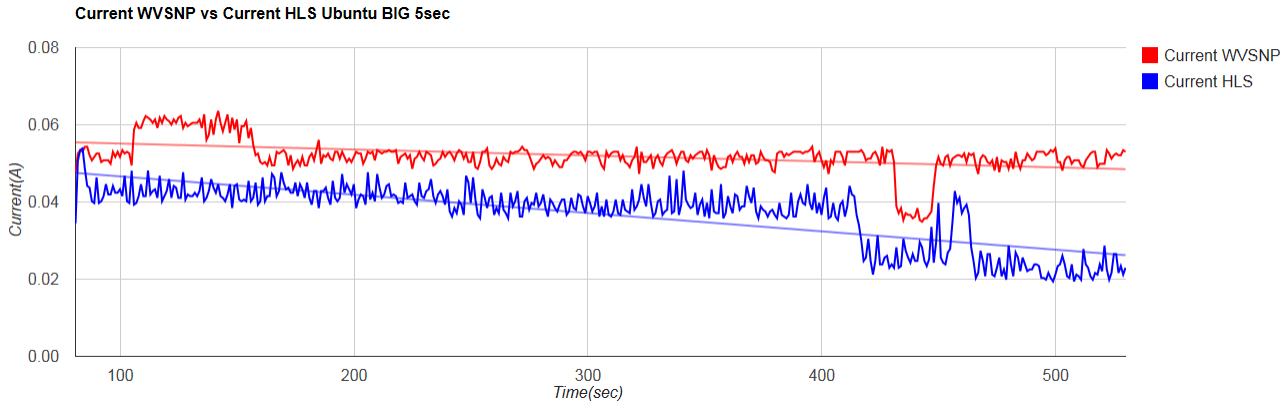

Figure 8.57: Current Consumption of Playbacking HLS and WVSNP BIG Videos on Ubuntu (5sec)

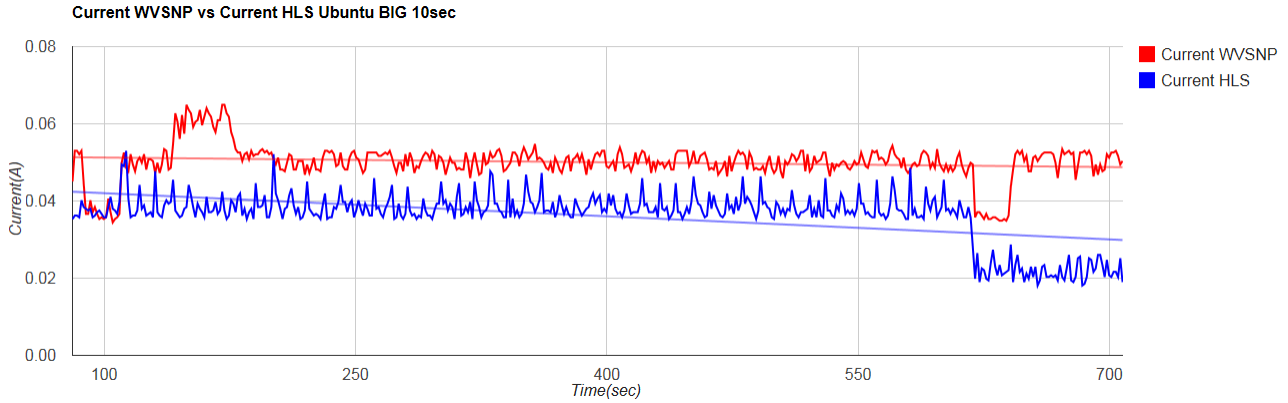

Figure 8.58: Current Consumption of Playbacking HLS and WVSNP BIG Videos on Ubuntu (10sec)

As for the playback case for Windows and Mac, here for Ubuntu also, WVSNP current consumption is higher at the beginning since it buffers all the segments while playing it. The current consumption of WVSNP is very high compared to HLS for 5sec and 10sec case. While for 2sec, the current consumption of WVSNP is higher than HLS at the start and then it reduces down below the current consumption of HLS. Since Ubuntu operating system is very power consuming, it has impacted a lot on WVSNP videos. The WVSNP player is required to be more efficient for playback

on Ubuntu to reduce the current consumption. Now, here are the graphs of playing back WVSNP and HLS of SMALL option.

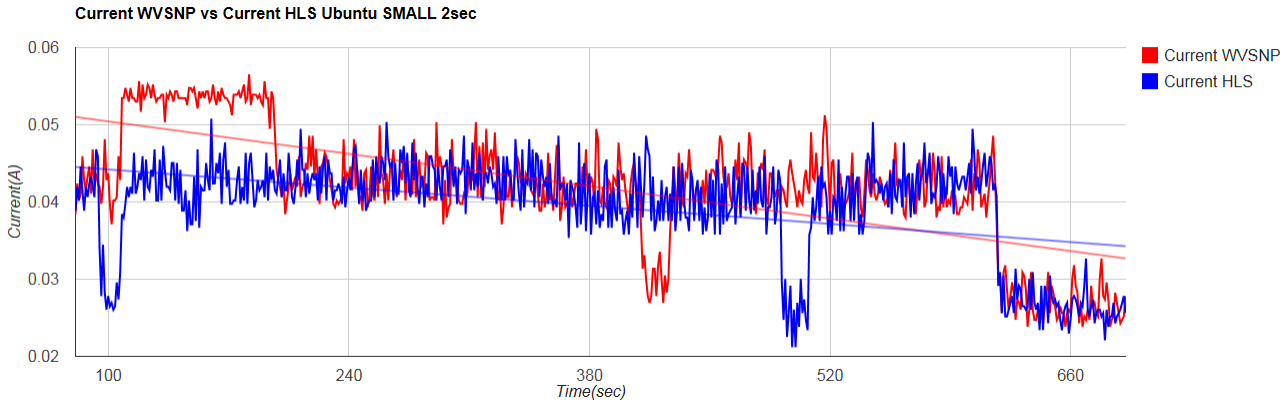

Figure 8.59: Current Consumption of Playbacking HLS and WVSNP SMALL Videos on Ubuntu (2sec)

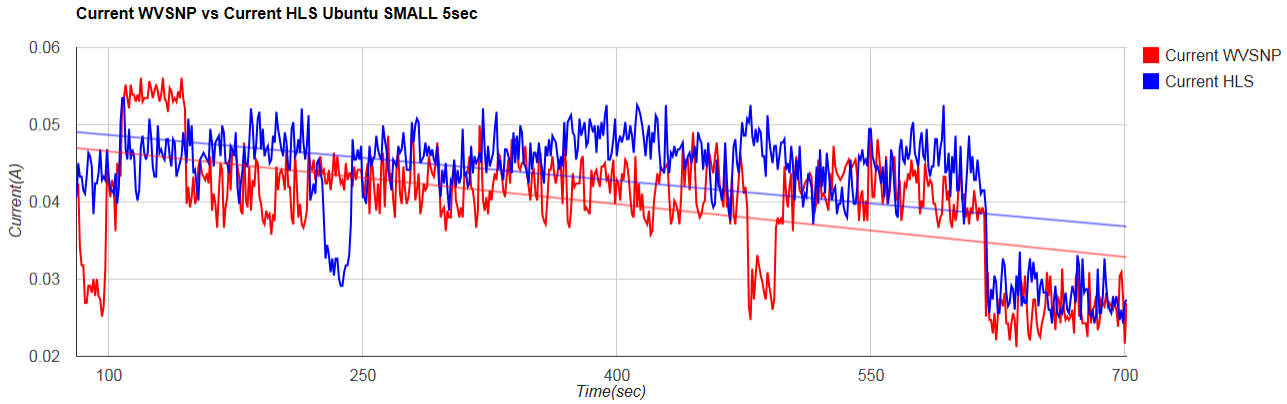

Figure 8.60: Current Consumption of Playbacking HLS and WVSNP SMALL Videos on Ubuntu (5sec)

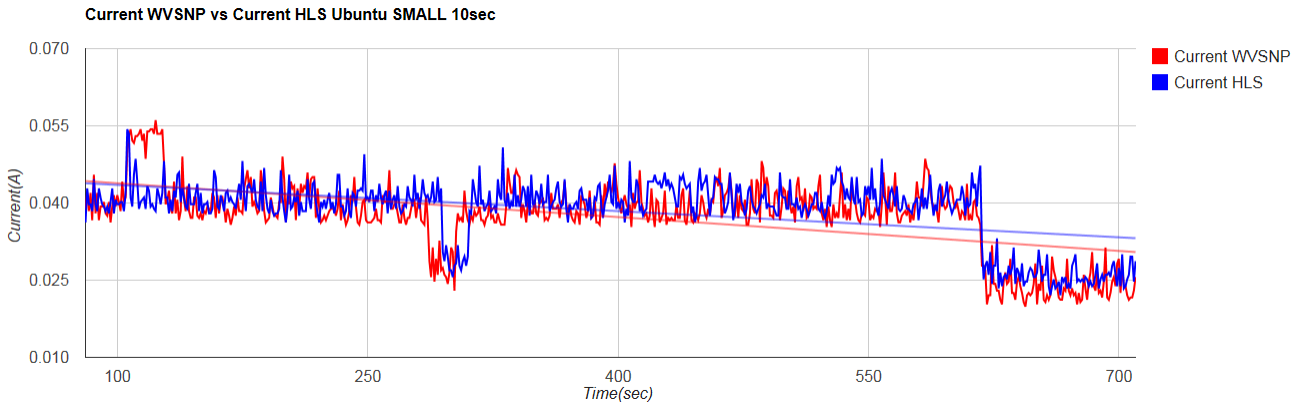

Figure 8.61: Current Consumption of Playbacking HLS and WVSNP SMALL Videos on Ubuntu (10sec)

The results with SMALL option were better than BIG option for all segment option. For 5sec and 10sec case, the current consumption of WVSNP is lower than HLS over the whole capture. While for 2sec, WVSNP consumes more at the start and then starts consuming less than HLS by the end of the capture just as for BIG 2sec playback.

Here are some live capturing results that were performed for both BIG and SMALL video option. The first set are the live capturing and real-time playing of BIG option. The capturing is performed using ffmpeg commands and encoding is done through H264 encoder.

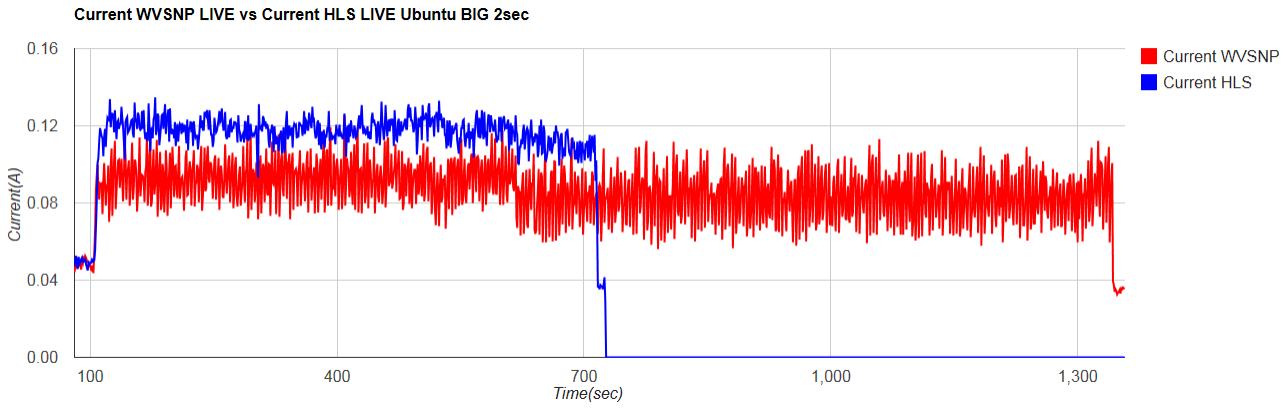

Figure 8.62: Current Consumption of Live Capturing and Real-Time playing of BIG WVSNP and HLS Videos on Ubuntu (2sec)

The current consumption of HLS is higher than WVSNP for the capture. It can be noted that HLS completes both live capturing and real-time playing much faster than WVSNP. As described earlier, ffmpeg commands were used for live capturing, but here ffmpeg's capabilities of segmenting videos while capturing were used to avoid turning ON/OFF of the camera for every segment. Secondly, JwPlayer is used for real-time playing the HLS videos. The player starts playback as soon as the first segment is available, and if during the playback, next segment is not ready yet, it will wait for it to be ready and then resumes it again. For WVSNP, ffmpeg loop capture is used, therefore camera is turned ON/OFF for each segment capture. Since camera is turned ON/OFF for each capture, it takes extra time to get segment ready for playback. Secondly, WVSNP player is designed to wait for 3 segments before a playback can start. If the player completes playing the current segment before the next segment is ready, the player again waits for 3 segments to be ready, before resuming the playback. This is the second reason, WVSNP has taken way more time than HLS to complete. Now, here is the live capturing and real-time playing of WVSNP and HLS videos of SMALL option.

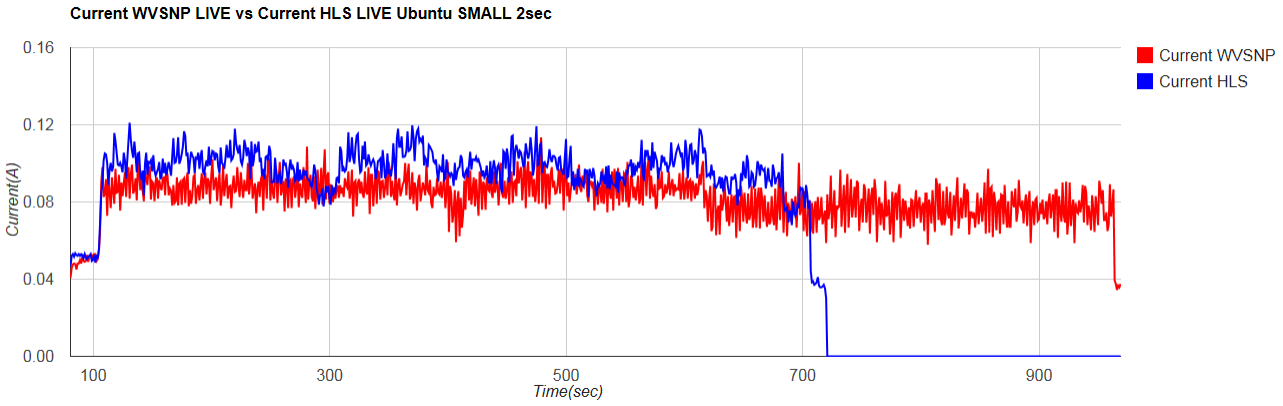

Figure 8.63: Current Consumption of Live Capturing and Real-Time Playing of SMALL WVSNP and HLS Video on Ubuntu (2sec)

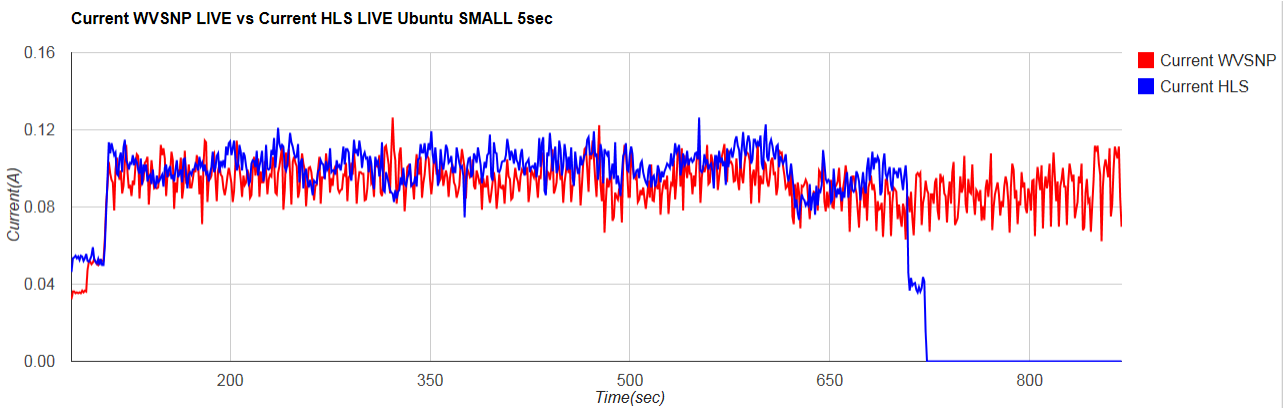

Figure 8.64: Current Consumption of Live Capturing and Real-Time Playing of SMALL WVSNP and HLS Videos on Ubuntu (5sec)

First, live capturing and real-time playing of WVSNP video with SMALL option completes earlier than WVSNP video with BIG option. Since the resolution and frames per second of WVSNP videos are lower with SMALL option, the encoding became faster and therefore it completed little earlier. Here also, the current consumption of WVSNP is lower than HLS in both 2sec and 5sec cases. Now, lets look at the comparison of live capturing and real-time playing of WVSNP video of BIG option with WVSNP video of SMALL option and HLS video of BIG option with HLS video of SMALL option.

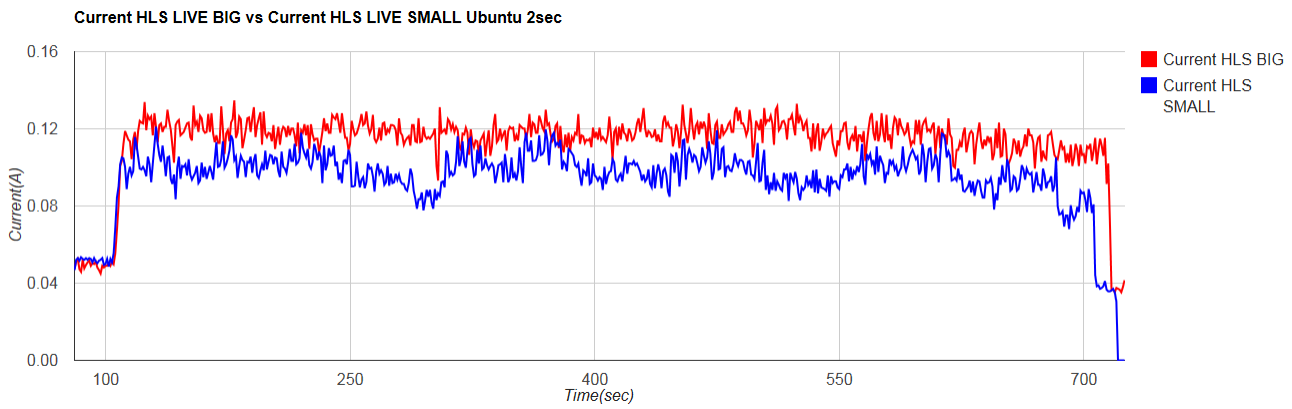

Figure 8.65: Current Consumption of Live Capturing and Real-Time Playing of BIG WVSNP Video and SMALL WVSNP Video (2sec)

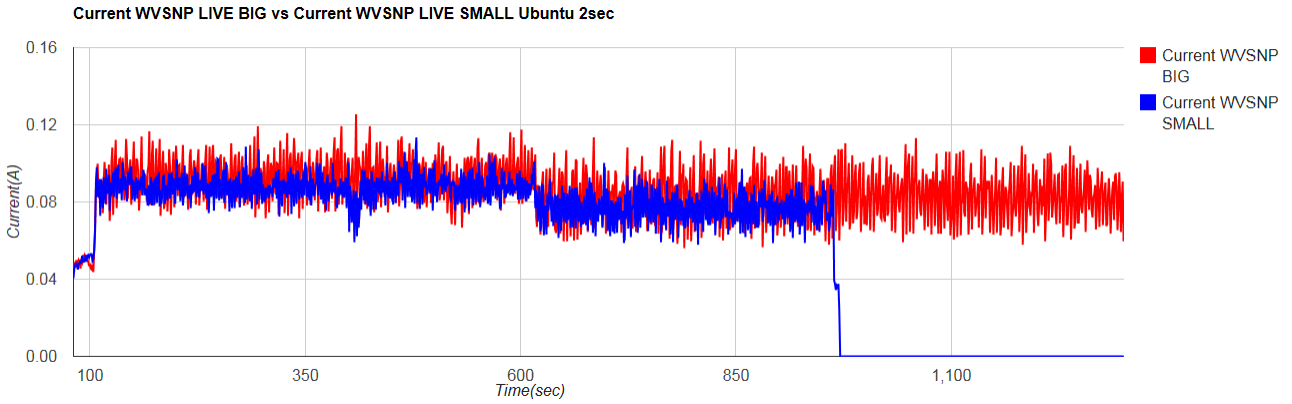

Figure 8.66: Current Consumption of Playbacking HLS Video on Ubuntu and Playbacking HLS Video on Windows (5sec)

The current consumption of WVSNP video with BIG option is higher than WVSNP video with SMALL option as expected and same follows for HLS. It can be seen from the WVSNP comparison plot, that SMALL option videos completes earlier than BIG optionas talked before. For HLS, SMALL video option completes just a second earlier than BIG option, but there is a substantial difference in the current consumption.

The last result shows the current consumption comparison of playing back a 10sec segments of MPEG-DASH with WVSNP 10 sec segments on Ubuntu. Here,

MPEG-DASH videos were played on DASH player on chrome and WVSNP videos on WVSNP player also on chrome. Currently, since live capturing is not working for MPEG-DASH, only playback results are shown.

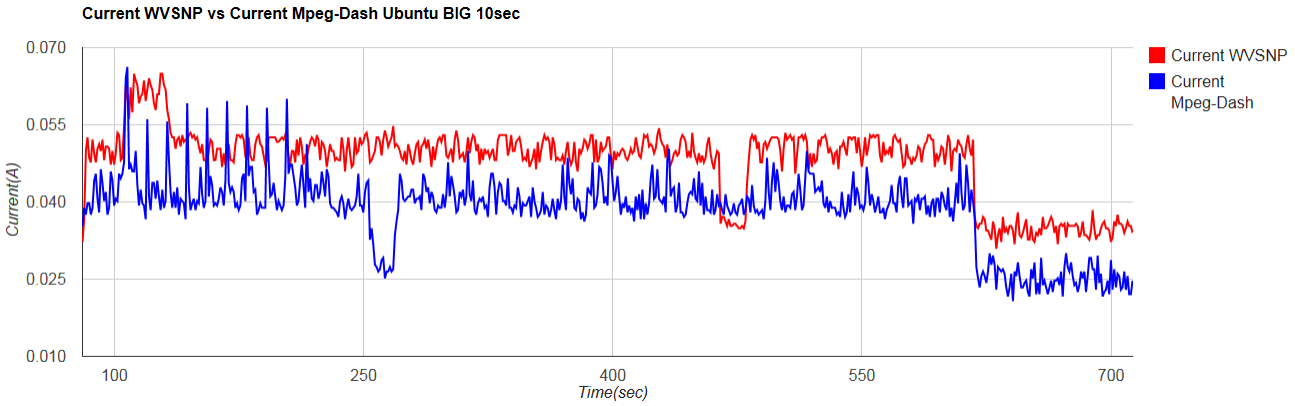

Figure 8.67: Current Consumption Comparison of Playbacking WVSNP Video and MPEG-DASH Video(10sec)

The current consumption of playing BIG WVSNP video is higher than BIG MPEG-DASH videos. For MPEG-DASH, mp4box is used to create videos, while for WVSNP ffmpeg was used to create videos. This would be one of the reasons of MPEG-DASH performing better than WVSNP over the whole capture.

The results shown here were taken under same setup but at different times and different temperature. Some of the results were taken in lab environment where the temperature is same over the whole day and some were taken by me at my home where temperature is different depending upon time of the day. To ensure reliability of the data, I took same result at lab and then repeated it at home at different times of the day. All the results of the test performed were found to be same. This shows that there is no impact of temperature or time of the day on the device and thereby the obtained results. For a next step, I will be taking one of the best result from above and use a more sensitive current clamp to perform measurement at different times of the day to ensure the accuracy of the measurement.

Chapter 9

RECOMMENDATION

There are some recommendations that I would like to make based upon the research work I performed.

- Currently, gstreamer is capable of capturing video encoded with both software encoder and hardware encoder, while ffmpeg is only capable of capturing video encoded with software encoder. Ffmpeg has a capability of segmenting the videos while it is being captured without any loops and scripts, while gstreamer requires scripts to capture in segments. Furthermore, gstreamer is not as power efficient as ffmpeg while capturing the videos. Since ffmpeg is very beneficial, there is some research required in creating a wrapprer for ffmpeg which would make it possible to grab data encoded by hardware encoder through ffmpeg also.

- The results for hardware and software encoder comparison for avi and mpeg4 with video captured using USB camera were not as expected. The power consumption of encoding video using software jpeg and mpeg4 encoders through USB camera is somehow performing better than power consumption of encoding video using hardware jpeg and mpeg4 encoders. Therefore, there is a need to look at the code of mpeg4 and avi software encoders to figure out problems and make it more efficient.

- The live capturing and real-time playing of HLS completes way before WVSNP completes. These happens mainly due to the two reasons: 1) Since ffmpeg segmenter is used for HLS capture, but ffmpeg loop capture is used for WVSNP

capture. 2) The player used for HLS playback buffers only one segment and starts playback. In case, if the playback picks up with live-capture, the HLS player only buffers 1 segment and resumes again. While WVSNP player buffers for 3 segments and therefore, if playback picks up with the live-capturing here, the WVSNP player buffers next 3 segments before resuming again. Therefore there is a need for a standalone capture application created, which will speed up video capture file preparation during capture. So a 10 minute live video capture doesn't take 15 minutes to capture and playback. These lost compuation time can be used for *L*ive capture Parallelization, *t*hreaded capturing and file preparation, *H*W capture.

- It is recommended to use WVSNP than HLS because it consumes lesser current for all cases for live capturing and real-time playing and playback. Besides this, HLS is limited to only one container (MPEG2-TS), while WVSNP videos can be encapsulated into MP4, AVI, MPEG2-TS and others. WVSNP is easier to implement for server node and different OSes as it is playable on different browsers(Chrome, Windows Internet Explorer, Firefox), while HLS is either playable on Safari or on Chrome. WVSNP is backward compatible, while HLS might not work with older version of browsers. And lastly, WVSNP doesn't require any type of browser modification.

- The current consumption of USB camera is higher than CSI camera for any hardware or software encoders. Since USB camera stack is big, the data goes through lots of block and it also has a software hand-off of the data on the board. While CSI camera's stack is smaller and it has a direct path from the camera module on the board to the encoders. Therefore, CSI camera is highly recommended camera.

- There are two types of CSI(camera serial interface) camera's available(Parallel,Serial). Here, serial camera was used for measurements. Some power consumption results are required to be taken with parallel camera to draw a comparison between them. This comparison will make it clear on which camera is efficient among CSI camera's.

- The measurements were performed with three different segment length(2sec, 5sec and 10sec). Based on the results, 10sec segment is the recommended segment length. Since there is lots of fluctuations in the bandwidth, and with this if small segment sizes are used, there will be lots of flickering seen in the playback. Flickering occurs because adaptation of increase or decrease in video quality is never smooth due to the small segment size. With bigger segment size, if the video quality or segment length changes, adaptation will be smooth before the next segment is played. And, with bigger segment size, the coding efficiency also increases. Therefore 10sec is a recommended segment size.

- Progressive download consumes more power than segmented videos playback. Therefore segmented videos are recommended to be used than using a big video. Here is the reasoning: A progressive is a big file in a node. Now this big file needs more energy to just open it before reading. While WVSNP requests HTTP server to send small segments. These segments are not required to be open, they just need to be sent. With progressive download, the file is opened and a small portion of a file(byte-by-byte) is sent. Now, if more portion(bytes) are required, the files has be opened over and over again and has to seek the position it was at some known point in the file. Therefore this opening and closing of big files and remembering the position of known point in the file consumes power.

- There are some implications with using BIG and SMALL video option. The difference between BIG and SMALL option is in the resolution, framerate and bitrate of the video. It is recommended to use BIG video option for better quality if the bandwidth is high, while SMALL video option is better for use if the bandwdith is lower.

- There are some segment length effects on LIVE and VOD. With the increase in the segment length, the number of segments also decreases and due to this the current consumption graph starts to settle down at a value although there are peaks and valleys here also. With smaller segment size, the number of segments are higher and therefore if a segment is played, the current consumption rises but even before it settles, the segment is completed playing and so graph goes down and this keeps on repeating till the end. While in case of LIVE video, with the increase in segment size, HLS current consumption doesn't changes, but WVSNP current consumption gets little bit higher.

- There are many effects of BIG and SMALL video option on LIVE and VOD. In case of VOD, with BIG option the current consumption of WVSNP is higher than HLS with a huge gap between them. Now with SMALL video option the current consumption of both WVSNP and HLS has shifted down and the gap has been reduced to micro amps. And, the current consumption of WVSNP has been lower than HLS with SMALL video option. In case of LIVE, with BIG video option HLS completes very early than WVSNP and the current consumption of both are higher. With SMALL video option, HLS completes at the same time while WVSNP completes earlier than before since with lower video quality the encoding and playback got easier to cope with.

- Since LIVE capturing is not possible with MPEG-DASH for rightnow, there is only a comparison made between MPEG-DASH, WVSNP and HLS with playback of 10sec segments. And this result shows that MPEG-DASH outperforms both WVSNP and HLS. The results to look at will be the live-capturing of WVSNP, HLS and MPEG-DASH.

Chapter 10

CONCLUSION AND FUTURE WORK

The power analysis was performed to predict the node's durability when it is battery operated and based on this, some recommendations were provided. To perform power analysis, I created a small optimized image to boot the Wandboard DUAL/QUAD board and perform capture and playbacks. There are three types of measurement taken where some were taken with Wandboard dual board and some with quad board. First, playbacks were performed of WVSNP and HLS and MPEG-DASH using quad board which runs a moongoose webserver and it serves all players and videos. The playback results were taken when board connects to a router via Ethernet or Wifi connection. Second, live-capturing and real-time playing of WVSNP and HLS videos were performed for BIG and SMALL video option also using quad board. And, finally there were some results performed that were only capture, and these were performed on dual board. The results where videos were captured were performed using gstreamer or ffmpeg commands. Based on this results, here are some recommendations: 1) It is recommended to use ffmpeg instead of gstreamer since it is power efficient. A wrapper is required so that ffmpeg would be able to encode raw data using hardware encoder. 2) After performing results with USB and CSI camera, I found that CSI camera should be used, because the capturing is very efficient and besides that camera module on the board has a direct path to the GPU for processing. 3) After comparing both hardware and software H264, it is highly beneficial to use hardware H264, since it suppresses the current consumption manifold. 4) Full video(without segments) as well as video in segments were played back and based on that it is recommended to use segments because progressive download consumes very

high power compared to segment playback. 5) And lastly, BIG and SMALL video options are both important as they provide a adaptive streaming options in case of LIVE capture and VOD. If network bandwidth is high, it is still recommended to use SMALL video option, since it consumes lesser power and it would avoid buffering issues.

Although the power consumption obtained with serial CSI camera were exceptionally good, I would like to look at the results obtained with parallel CSI camera, since it might be even more efficient. Secondly, I have used H264 encoder for encoding the videos for both HLS and WVSNP and the encoding has been better than other encoders, but it would be very interesting to see if the newer version i.e. H265 encoder is suitable for WVSNP power requirements.

An exciting future research direction is to examine visual wireless sensor networks in combination with modern access networks, such as fiber-wireless networks [4, 11, 25, 31, 48]. Importantly, in the context of embedding of visual sensor networks in access networks, and the larger Internet of Things, it will be important to carefully consider the characteristics of the video traffic [39, 42]. Quality of service mechanisms will likely be needed to ensure timely delivery of the video data over the multiple networks hops [40, 15].

REFERENCES

[1] "Android power management on i.mx6dq/dl", `https://community.freescale.com/docs/DOC-93884/version/1` (2012).

[2] Akyildiz, I. F., T. Melodia and K. R. Chowdhury, "A survey on wireless multimedia sensor networks", Computer networks **51**, 4, 921–960 (2007).

[3] Aurlien Bourdon, R. R., Adel Noureddine and L. Seinturier, "Powerapi: A software library to monitor the energy consumed at the process-level", (2012).

[4] Aurzada, F., M. Levesque, M. Maier and M. Reisslein, "Fiwi access networks based on next-generation pon and gigabit-class wlan technologies: A capacity and delay analysis", (2013).

[5] Bachir, A., M. Dohler, T. Watteyne and K. Leung, "Mac essentials for wireless sensor networks", Communications Surveys Tutorials, IEEE **12**, 2, 222–248 (2010).

[6] Blum, R. S. and Z. Liu, *Multi-sensor image fusion and its applications* (CRC press, 2005).

[7] Bouaziz, S., M. Fan, A. Lambert, T. Maurin and R. Reynaud, "Picar: experimental platform for road tracking applications", in "Intelligent Vehicles Symposium, 2003. Proceedings. IEEE", pp. 495–499 (2003).

[8] Bramberger, M., R. Pflugfelder, B. Rinner, H. Schwabach and B. Strobl, "Intelligent traffic video sensor: Architecture and applications", in "Proceedings of the Telecommunications and Mobile Computing Workshop", (Citeseer, 2003).

[9] Campbell, J., P. B. Gibbons, S. Nath, P. Pillai, S. Seshan and R. Sukthankar, "Irisnet: an internet-scale architecture for multimedia sensors", in "Proceedings of the 13th annual ACM international conference on Multimedia", pp. 81–88 (ACM, 2005).

[10] Chen, P., S. Oh, M. Manzo, B. Sinopoli, C. Sharp, K. Whitehouse, O. Tolle, J. Jeong, P. Dutta, J. Hui, S. Schaffert, S. Kim, J. Taneja, B. Zhu, T. Roosta, M. Howard, D. Culler and S. Sastry, "Experiments in instrumenting wireless sensor networks for real-time surveillance", in "Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on", pp. 3128–3133 (2006).

[11] Coimbra, J., G. Schütz and N. Correia, "A game-based algorithm for fair bandwidth allocation in fibre-wireless access networks", Optical Switching and Networking **10**, 2, 149–162 (2013).

[12] DIAS, F., P. CHALIMBAUD, F. BERRY, J. SEROT and F. MARMOITON, "Embedded Early Vision systems: implementation proposal and Hardware Architecture", (Cognitive System for interactive sensor(COGIS 2006), 2006).

[13] Dong, W., C. Chen, X. Liu and J. Bu, "Providing os support for wireless sensor networks: challenges and approaches", Communications Surveys & Tutorials, IEEE **12**, 4, 519–530 (2010).

[14] Electrons, F., "Power management", `http://free-electrons.com/doc/power-management.pdf` (2011).

[15] Fidler, M., "Survey of deterministic and stochastic service curve models in the network calculus", Communications Surveys & Tutorials, IEEE **12**, 1, 59–86 (2010).

[16] Friedman, J., D. Lee, I. Tsigkogiannis, S. Wong, D. Chao, D. Levin, W. Kaiser and M. Srivastava, "Ragobot: A new platform for wireless mobile sensor networks", in "Proceedings of the First IEEE international conference on Distributed Computing in Sensor Systems", pp. 412–412 (Springer-Verlag, 2005).

[17] Gayan, "Powerstat: Power consumption calculator for ubuntu linux", `http://www.hecticgeek.com/2012/02` (2012).

[18] Haas, C., J. Wilke and V. Stöhr, "Realistic simulation of energy consumption in wireless sensor networks", in "Wireless Sensor Networks", pp. 82–97 (Springer, 2012).

[19] He, Z., Y. Liang, L. Chen, I. Ahmad and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints", Circuits and Systems for Video Technology, IEEE Transactions on **15**, 5, 645–658 (2005).

[20] He, Z. and D. Wu, "Resource allocation and performance analysis of wireless video sensors", Circuits and Systems for Video Technology, IEEE Transactions on **16**, 5, 590–599 (2006).

[21] Hengstler, S., D. Prashanth, S. Fong and H. Aghajan, "Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance", in "Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on", pp. 360–369 (2007).

[22] Hergenroder, A. and J. Furthmuller, "On energy measurement methods in wireless networks", in "Communications (ICC), 2012 IEEE International Conference on", pp. 6268–6272 (2012).

[23] Hergenroeder, A., J. Wilke and D. Meier, "Distributed energy measurements in wsn testbeds with a sensor node management device (snmd)", in "Architecture of Computing Systems (ARCS), 2010 23rd International Conference on", pp. 1–7 (2010).

[24] Hill, J. L., *System architecture for wireless sensor networks*, Ph.D. thesis, University of California (2003).

[25] Hossen, M. and M. Hanawa, "Network architecture and performance analysis of multi-olt pon for ftth and wireless sensor networks", Int. J. Wireless Mobile Netw **3**, 6, 1–15 (2011).

[26] Jiang, X., P. Dutta, D. Culler and I. Stoica, "Micro power meter for energy monitoring of wireless sensor networks at scale", in "Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on", pp. 186–195 (2007).

[27] Karlsson, J., T. Wark, P. Valencia, M. Ung and P. Corke, "Demonstration of image compression in a low-bandwidth wireless camera network", in "Proceedings of the 6th international conference on Information processing in sensor networks", pp. 557–558 (ACM, 2007).

[28] Kogut, G., M. Blackburn and H. Everett, "Using video sensor networks to command and control unmanned ground vehicles", Tech. rep., DTIC Document (2003).

[29] Kuo, C., C. C. Chen, W. C. Wang, Y. C. Hung, E. C. Lin, K. Lee and Y. Lin, "Remote control based hybrid-structure robot design for home security applications", in "Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on", pp. 4484–4489 (2006).

[30] Liu, H., T. Roeder, K. Walsh, R. Barr and E. G. Sirer, "Design and implementation of a single system image operating system for ad hoc networks", in "Proceedings of the 3rd international conference on Mobile systems, applications, and services", pp. 149–162 (ACM, 2005).

[31] Maier, M., N. Ghazisaidi and M. Reisslein, "The audacity of fiber-wireless (fiwi) networks", in "AccessNets", pp. 16–35 (Springer, 2009).

[32] Milenkovic, A., M. Milenkovic, E. Jovanov, D. Hite and D. Raskovic, "An environment for runtime power monitoring of wireless sensor network platforms", in "System Theory, 2005. SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium on", pp. 406–410 (2005).

[33] Misra, S., M. Reisslein and G. Xue, "A survey of multimedia streaming in wireless sensor networks", Communications Surveys Tutorials, IEEE **10**, 4, 18–39 (2008).

[34] Nakamura, E. F., A. A. Loureiro and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications", ACM Computing Surveys (CSUR) **39**, 3, 9 (2007).

[35] Noureddine, A., A. Bourdon, R. Rouvoy and L. Seinturier, "A Preliminary Study of the Impact of Software Engineering on GreenIT", in "First International Workshop on Green and Sustainable Software", pp. 21–27 (Zurich, Switzerland, 2012).

[36] Noureddine, A., A. Bourdon, R. Rouvoy and L. Seinturier, "Runtime monitoring of software energy hotspots", in "Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering", ASE 2012, pp. 160–169 (ACM, New York, NY, USA, 2012).

[37] Pudlewski, S. and T. Melodia, "A distortion-minimizing rate controller for wireless multimedia sensor networks", Computer Communications **33**, 12, 1380–1390 (2010).

[38] Rein, S. and M. Reisslein, "Low-memory wavelet transforms for wireless sensor networks: a tutorial", Communications Surveys & Tutorials, IEEE **13**, 2, 291–307 (2011).

[39] Reisslein, M., J. Lassetter, S. Ratnam, O. Lotfallah, F. Fitzek, S. Panchanathan *et al.*, "Traffic and quality characterization of scalable encoded video: a large-scale trace-based study, part 1: overview and definitions", Arizona state university, dept. of electrical engineering, technical report (2003).

[40] Reisslein, M., K. W. Ross and S. Rajagopal, "Guaranteeing statistical qos to regulated traffic: The multiple node case", in "Decision and Control, 1998. Proceedings of the 37th IEEE Conference on", vol. 1, pp. 531–538 (IEEE, 1998).

[41] Ren, X. and Z. Yang, "Research on the key issue in video sensor network", in "Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on", vol. 7, pp. 423–426 (2010).

[42] Seeling, P. and M. Reisslein, "Video traffic characteristics of modern encoding standards: H. 264/avc with svc and mvc extensions and h. 265/hevc", The Scientific World Journal **2014** (2014).

[43] Seema, A. and M. Reisslein, "Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a flexi-wvsnp design", Communications Surveys Tutorials, IEEE **13**, 3, 462–486 (2011).

[44] Soro, S. and W. Heinzelman, "On the coverage problem in video-based wireless sensor networks", in "Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on", pp. 932–939 Vol. 2 (2005).

[45] Van De Ven, K. A., Arjan and A. Yates, "Power top", (????).

[46] Vieira, M., C. Coelho, J. da Silva, D.C. and J. da Mata, "Survey on wireless sensor network devices", in "Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference", vol. 1, pp. 537–544 vol.1 (2003).

[47] Yin, H., C. Lin, B. Sebastien and X. Chu, "A novel secure wireless video surveillance system based on intel ixp425 network processor", in "Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling", WMuNeP '05, pp. 62–69 (ACM, New York, NY, USA, 2005).

[48] Yu, X., Y. Zhao, L. Deng, X. Pang and I. Tafur Monroy, "Existing pon infrastructure supported hybrid fiber-wireless sensor networks", in "National Fiber Optic Engineers Conference", pp. JTh2A–32 (Optical Society of America, 2012).

[49] Zhang, Q., W. Zhu and Y.-Q. Zhang, "End-to-end qos for video delivery over wireless internet", Proceedings of the IEEE **93**, 1, 123–134 (2005).