

# Cloud Enabled 3D Tablet Design for Medical Applications

Vishwa Kiran S<sup>1</sup>, Ramesh Prasad<sup>2</sup>, Thriveni J<sup>1</sup>, Venugopal K R<sup>1</sup>, L M Patnaik<sup>3</sup>

<sup>1</sup>Department of Computer Science and Engineering,  
University Visvesvaraya College of Engineering,  
Bangalore University, Bangalore 560 001, India

<sup>2</sup>Chaos Labs, Senior Member IEEE

<sup>3</sup>Indian Institute of Science, Bangalore, India  
Email:[vishwakirana@pushkala.in](mailto:vishwakirana@pushkala.in)

**Abstract**—The prime objective of any technological innovation is to improve the life of people. Technological innovation in the field of medical devices directly touches the lives of millions of people; not just patients but doctors and other technicians as well. Serving these care givers is serving humanity. Growth of Mobil Devices and Cloud Computing has changed the way we live and work. We try to bring the benefits of these technological innovations to the medical field via equipment which can improve the working efficiencies and capabilities of the medical professionals and technicians. The improvements in the camera and image processing capabilities of the Mobile Devices coupled with their improved processing power and an infinite processing and storage offered by Cloud Computing infrastructure opens up a window of opportunity to use them in the specialized field like microsurgery. To enable microsurgery, surgeons use optical microscope to zoom into the working area to get better visibility and control. However, these devices suffer from various drawbacks and are not comfortable to use. We build a Tablet with large stereoscopic screen allowing glasses free 3D display enabled by cameras capable of capturing 3D video and enhanced by an image processing pipeline, greatly improves the visibility and viewing comfort of the surgeon. Moreover using the capabilities of Cloud computing, these surgeries can be recorded and streamed live for education, training and consultation. An expert sitting in a geographically remote location can guide the surgeon performing the surgery. All vital parameters of the patient undergoing surgery can be shown as an overlay on the Tablet screen so that the surgeon is alerted of any parameter going beyond limit. Developing this kind of complex device involves engineering skills in hardware and software and huge amount of investments in terms of time, resources and money. To accelerate the development, we make use of open source hardware and software and demonstrate how we can accelerate the development using these open source resources.

**Keywords**—3D Video; stereoscopic display; Tablet; Android; Multicore Processor; System Design; Mobile Cloud Computing; Open Source.

## I. INTRODUCTION

The mobile devices of today are more powerful than the computers of few years ago. Powered by multi core processors and fast RAMs, these devices allow complex computations to be performed on the device itself. This allows us to build applications and devices which not just replace the existing

manual devices but add more features and capabilities making it more useful and easy to use. We leverage the power of these mobile devices for medical application like microsurgery. Microsurgery involves operating on objects too small to be seen with the naked eye. Traditionally microsurgery has been done using optical microscopes. The microscope allows a magnification from 2.5x to 20x and uses fiber-optic for shadow free illumination. The operator gets parallel viewing capability using the optics provided by the Galilean system. However these devices have limited working distance, limited field of view, poor brightness and suffer from distortions. All these drawbacks make it difficult and uncomfortable to use. These equipment are known to cause fatigue during prolonged use [1][2]. These problems can be resolved using a Tablet with stereovision and a stereoscopic display, which lets the operator view the object in 3D on a very large, high-resolution display without the need for special glasses. Various optical and illumination distortions can be compensated by image processing algorithms. This greatly increases the capability to manipulate the object and reduces stress on the operator. Moreover innovative value adds like recording and streaming of the surgery, visual overlays etc can be provided on this platform. This 3D Tablet has applications in various other fields where micromanipulations are done like diamond cutting.

In this paper we describe the design of a 3D Tablet, which can be used in application described. In section II we describe the key requirements of the Tablet, hardware architecture and the software architecture of the Tablet. In section III, we discuss the selection criteria for the development board and describe the features of the board selected. In section IV, we describe the implementation flow and finally we conclude in section V.

## II. ANALYSIS

In this section we analyze the requirements and develop the hardware and software architecture of the 3D Tablet.

### A. Requirements

The 3D Tablet must be a real-time video play system with enough processing power to do video processing such as change of video with textual and graphical information

overlay. The system should have a display screen of size 12"-15" in size and the whole system can be considered as a tablet of that dimension. All powers should be through the board. The peripherals that derive power are 4 LED lights of 3W each, camera sensors (upto 4 cameras) and the display screen. The system supports 1080p (Full HD) video frames at 60 frames per second, which may not be compressed. The system preferably runs a high level operating system.

Based on these requirements, next we develop the Hardware and Software architecture along with the specifications.

### B. Hardware Architecture

The Fig. 1 shows the hardware architecture of the device-

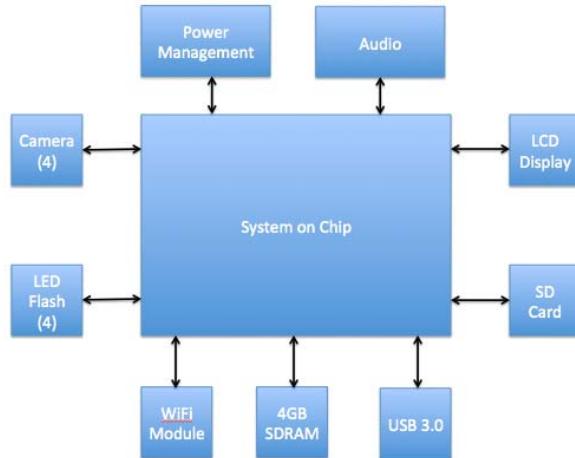


Fig. 1. Hardware Architecture

The following is a brief description of each component-

1. System on Chip (SoC) – The SoC would be the main processing unit comprising of CPU, video processing and Image Signal Processing Units. It must have enough processing capability to handle 4-channel 1080p 30 fps video.
2. Memory – The memory must be able to store upto 8 1080p frames and must have enough bandwidth to allow realtime processing of the video frames.
3. Camera – There would be 4 camera units each capturing 1080p 30 fps video each. The cameras would be adjusted to allow stereoscopic (3D) video to be captured.
4. LED Flash – To allow illumination of the object.
5. WiFi – WiFi module will provide network connectivity to the Tablet.
6. USB 3.0 – USB 3.0 would provide high-speed peripheral connectivity to the Tablet.
7. SD/MMC Card – SD/MMC Card will provide storage for the Operating system kernel and file system.
8. LCD Display – 12/15 inch full touch capacitive LCD display of resolution 2560 x 1600 pixels and ~247 pixel per inch density. It would be stereoscopic

display, which would render 3D videos, which can be viewed without glasses.

9. Audio – Audio would provide the audio capture and playback capability to the device.
10. Power Management – Power management would provide the battery charging capability and support power saving mechanisms of the SoC/OS.

### C. Software Architecture

The following are the software components of the device-

1. Camera module – The camera module would manage calibration and synchronization of the Image sensors for optimal stereoscopic capture. It would also control camera orientation, zoom and perform Image processing for stabilization. It would be controlled from the software running on CPU.
2. Camera Interface Driver – This module would provide camera control and image capture.
3. Camera Image Processing – Camera Image Processing like white balance etc.
4. Cloud Processing APIs – Middleware APIs which would allow image to be processed in the cloud.
5. 3D Image Processing – Algorithms for 3D Image Processing.
6. Display Image Processing – Typical Display Image processing like Gamma correction, resizing, overlay etc.
7. Display Interface Driver – Driver for displaying the processed Image on the LCD display.
8. Operating System – OS would provide the driver model, task scheduling, memory management, networking etc.

## III. SYSTEM DESIGN

In this section we discuss the design of the system based on the requirements and the specifications listed in the last section. We have deliberately chosen open source hardware and software components for designing the system. This has the benefit of ease of availability, flexibility to customize and large community support.

### A. Hardware

The selected hardware is Qualcomm Snapdragon 800 SoC based board. The following are the high level specifications of the SoC [3]-

1. Capture, playback and display in UltraHD video (with four times 1080p pixel density)
2. Dual Image Signal Processors for Snapdragon Camera support simultaneous camera/video, ultrafast 640 Megapixel/second capture speeds and computational camera
3. HD multichannel audio with DTS-HD and Dolby Digital Plus for enhanced audio
4. Higher display resolutions (up to 2560x2048) and Miracast 1080p HD support
5. Upgraded architecture for sustained peak performance in a mobile power profile

6. Quad core Krait on 800 CPU for advanced multitasking and multithreaded application support
7. LP-DDR3 memory for high performance and low latency
8. Updated Adreno 330 GPU delivers up to 50% increase in graphics performance
9. Support for advanced graphic and compute APIs, including OpenGL ES 3.0, DirectX, OpenCL, Render script Compute and Flex Render
10. Support for popular middleware and game engines like Unity, Epic, and Unigine
11. Hexagon DSP enables ultra-low power operation for a variety of applications like music playback, enhanced audio and advanced editing applications
12. Hexagon DSP now supports floating point calculations, dynamic multithreading and expanded multimedia instructions for enhanced low power performance
13. New USB 3.0 for ultra-fast transfers
14. 802.11ac Wi-Fi for peak Wi-Fi performance

The recommended board is Dragonboard based on Snapdragon 800. The following are the specifications of the board [4]-

- This DragonBoard Development Kit incorporates quad core high performance up to 2.3 Ghz Krait CPUs, a powerful multi-media engine with best in class 3D graphics, best in class programmable multimedia DSP, 1080p HD support, Wi-Fi, Bluetooth, USB connectivity, SATA, GPS, etc.
- The standalone CPU board is a complete, production ready, System-on-Module (SoM) enabling device maker to quickly and easily incorporate the SoM into their packaging. Just add the applications and go!
- Snapdragon 8074 quad core processor
- 2GB of LPDDR3 RAM, 16GB of eMMC
- Wi-Fi, Bluetooth, GPS
- HDMI out and qHD LCD with capacitive multi touch
- Adreno 330
- Hexagon DSP V5
- Qseven form factor (SoM)
- Mini ITX form factor (carrier board)
- Shipping with Android 4.2/kernel 3.4 preloaded
- Large numbers of Connectors like JTAG, USB, HDMI etc.

The SOM board can be used as is in the new hardware design thus greatly reducing the development time and complexity.

#### B. Software

The selected software stack is Android. Dragonboard ships with Android 4.2 and supports all the peripherals and SoC features. The following are the reasons for this selection-

1. Vendor support, hence reduces development time and efforts
2. Rich APIs and powerful open source stack

3. Large number of open source applications and drivers are available
4. Built on secure and robust Linux kernel which has excellent networking and driver support
5. Large and vibrant community

Fig. 2 shows the software architecture of the Android platform. The lowest layer of the stack is the Linux kernel layer and the device drivers. Above that are the native libraries, which are written in C. The native libraries provide implementation of many framework components like graphics, database, graphics etc. The native libraries also provide Hardware Abstraction Layer (HAL) for connecting the higher level frameworks with the underlying drivers and hardware. Alongside the native libraries is the Android runtime which provides an environment for the Java applications to run. The runtime environment consists of the Java Virtual Machine (JVM) which runs the Java byte codes and the core Java libraries which are essentially the Java packages. The layer above native libraries is the Application Framework layer, which is written in Java and communicates with the native layer via Java Native Interface (JNI). Above the Application Framework layer are the applications, which are written in Java using the framework APIs.

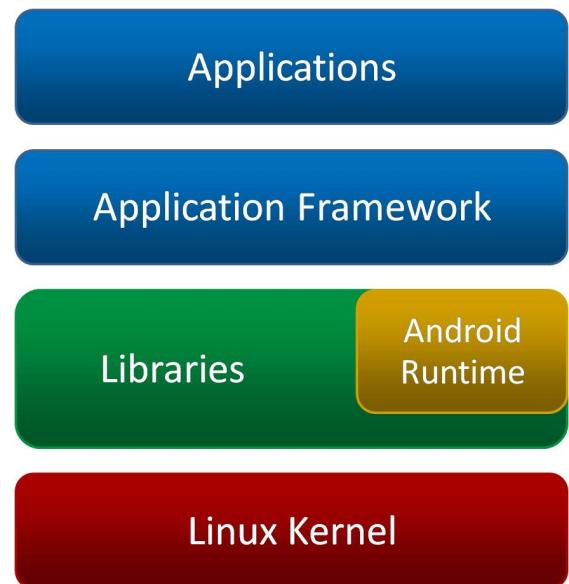


Fig. 2. Software Architecture

As we provide custom camera, codecs and display, we give a brief overview of the components in Android which needs to be modified.

To manage camera, Android provides camera HAL. We provide a custom HAL implementation which talks to the camera processing firmware via kernel device driver. The image processing pipeline for 3D video is implemented in the firmware. The kernel device driver is a generic pipe for passing the video data from the hardware to the userspace driver. User space driver implements the camera and processing pipeline control and implements the camera HAL. For audio and video,

Android provides a multi-media framework called Stagefright. Like any other multi-media framework, Stagefright provides a pipeline based architecture for building applications. Stagefright also allows integration with custom hardware codecs which are Openmax IL compliant. To support the compression of 3D video data, the codecs are implemented in the firmware and provide OMX IL interface for integration with the Stagefright. The display part of the Android is pretty complex as it uses a hardware composer with triple buffering for rendering the frame [5]. Qualcomm provides the driver for the 3D video support and allows stereoscopic displays to be supported on the Dragonboard.

One more important aspect of Android framework is ION memory allocator. The main goal of Android's ION subsystem is to allow for allocating and sharing of buffers between hardware devices and user space in order to enable zero-copy memory sharing between devices. This allows the different devices with different memory handling capabilities like scatter-gather, physically contiguous pages in memory etc. to work seamlessly with the userspace [6].

#### IV. IMPLEMENTATION

This section describes the implementation approach for meeting the objectives defined earlier. As mentioned the Dragonboard is used for development. It consists of two parts – SOM Module and Carrier Board. The SOM module has the SoC, RAM and some peripheral interfaces. The carrier board has multiple peripheral interfaces and connectors. This approach simplifies the design and the SOM board can be used as is in production design. The carrier board can be modified as per the requirement. The SOM is connected to the carrier via Edge connector.

Next we describe the major interfacing and processing functionality that has been implemented.

##### A. Camera Interface

As can be seen in the figure 2, the SOM module has 2 camera interfaces (2 in SOM board diagram).

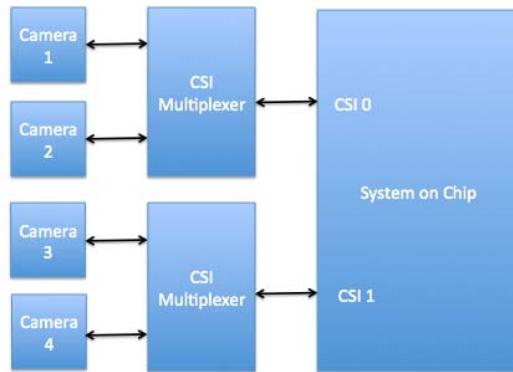


Fig. 3. CSI Multiplexing

The interface specifications for these two camera interfaces are as follows-

1. 2x MIPI CSI v1.0
2. 4 Lane

##### 3. 8 MP at 30 fps per CSI port

The carrier board has additional CSI port header, which can be populated. However it is connected to the CSI 2 port. As we need support for 4 cameras where 2 cameras would be active at a time, the Carrier Board is modified to build in additional camera interface and a multiplexing mechanism to enable/disable a set of cameras. All 4 cameras are connected to the CSI 0/1 ports with module selection pins for multiplexing. The Fig. 3 shows how the multiplexing would work-

##### B. Display Interface

As can be seen in the functional block diagram of figure 2, the Carrier module has 3 display interfaces – 2 DS1 and 1 HDMI. The interface specifications for these two DS1 interfaces are as follows-

1. 2x MIPI DS1
2. 4 Lane
3. Upto 2560x2048 resolution

There are various options for the stereoscopic glasses free LCD display, which support MIPI DS1 and can be which can be used in the Tablet. An appropriate display based on the availability would be used during production. During testing a 3D TV was used.

##### C. Data Flow Architecture

The overall data flow of the proposed system is as shown in the Fig. 4 [7]

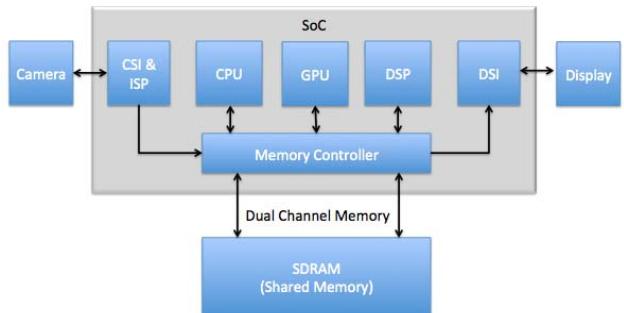


Fig. 4. Dataflow Architecture

The external SDRAM is shared amongst the various computing units. The processing algorithm is broken down between CPU, DSP and GPU tasks based on the characteristics of the processing and its suitability to the specific processor type. The frame data on which this computing unit work would reside in the shared memory and command and data pointers would be passed between them to enable processing. This ensures that the expensive data copy is avoided. Each compute unit has its own cache/On Chip Ram in which a part of this data is copied for processing. However the complete frame is never copied and the efficiency is maintained.

The GPU on Snapdragon 800 supports OpenGL ES 2.0/3.0. The data pointers and processing is specified via the API calls.

The DSP on Snapdragon 800 defines a set of C APIs for various processing task via which the data pointers can be passed to the DSP.

Snapdragon 800 has multiple processing units, namely CPU, GPU and DSP. Each of these processing units are optimized for a specific type of processing and differ in their instruction set and programming model. This heterogeneous computing allows the programmer to use a single programming model and abstracts the differences of the various computing units, while still leveraging their power to the fullest. Next we describe heterogeneous processing options on Snapdragon 800.

#### D. Approach to heterogeneous computing

Qualcomm supports various technologies for heterogeneous processing (processing on various cores of different types).

The following are the options and limitations-

1. Multicore Asynchronous Runtime Environment – It's a C++ library that helps developers to harness the benefits of Multicore SoC. However, it supports only homogenous cores like the Krait ARM 9 cores.
2. OpenCL – Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs) and other processors. However the OpenCL implementation for Snapdragon currently supports only CPU and GPU. It does not support DSP.

Hence due to the limitations described above, the developer must manually do the task of partitioning the algorithm, scheduling it across various computing units, synchronizing and control manually.

#### E. Bandwidth Computation

Given the real time nature of the device and the large volume of data to be processed, the bandwidth of the various busses and the processing unit becomes very critical. We will calculate the bandwidth required in the various data paths to ensure that the proposed solution meets the requirement.

##### 1. Data Source bandwidth

Assuming that we are capturing data in YUV 4:2:0  
 $16 \text{ bits per pixel} * 1920 * 1080 * 16 * 60 = 1.986 \text{ Gbps}$   
(Giga Bits Per Second)

##### 2. Camera Interface Bandwidth

The camera interface bandwidth is 8 MP @ 30 fps per camera interface. This translates to 3.8 Gbps, which is much larger than the required 1.986 Gbps. The two ISP in Snapdragon would control the transfer to data to the SDRAM memory. Qualcomm provides the BSP for the same.

##### 3. Memory bandwidth

The system memory specs are- 2xLPDDR3 SDRAM, 32 Bit Wide, 800 MHz. This gives us a theoretical bandwidth of 102 Gbps. Assuming 50% efficiency of reads and writes, we get 50 Gbps bandwidth. Assuming 4 reads and 4 writes of memory for the

data, we need a bandwidth of approximately 16 Gbps. This gives us a margin of 3x.

One of the main goals of optimization of video processing algorithm was to increase the memory efficiency by minimizing-

- a. Read/Write,
- b. DRAM refresh latency alignment and grouping of data
- c. Cache misses

#### 4. Processor bandwidth

There are multiple cores in the CPU-

- a. ARM x4 – 1.8 GHz
- b. DSP – 600 MHz
- c. GPU – 450 MHz

The overall clock cycle assuming 1.8 GHz CPU frequency is 8.25 GHz. Assuming an efficiency of 70%, the total cycle available is 5.7 GHz. This gives us a processing bandwidth of 46 CPU cycles per pixel or 23 CPU cycles per byte. This is a simplistic calculation where the parallelism of the cores has not been considered.

#### 5. Display bandwidth

DSI supports speeds of up to 800 Mbps per lane, totaling to 3.2 Gbps per lane for Dragonboard, which has 4 lanes. The required display rate assuming 60 Hz refresh rate for display would be 1.986 Gbps. Assuming 20% margin for sync, blanking etc, we need 2.3 Gbps. Hence we have sufficient margin for the display interface.

#### F. Cloud Architecture

The figure below shows the network architecture of the system to transport stereoscopic 3D live content from the Tablet to the cloud and then to multiple audience.

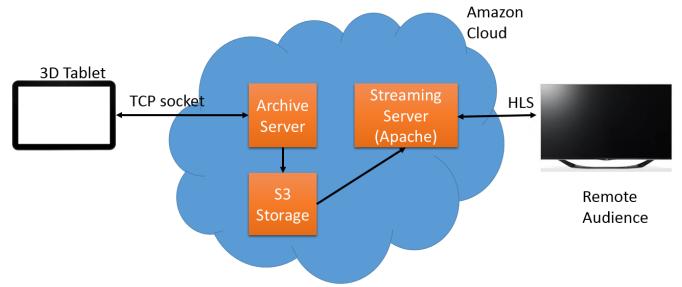


Fig. 5. Cloud Architecture

The tablet compresses the 3D video data using 3D extension of the HEVC [8]. This compressed data is used for local storage and archival/streaming in cloud. The Tablet application makes use of the cloud connector component to make connection to the Archival server on the cloud. The cloud connector essentially establishes a TCP socket connection to the Archival server to push the video data. The Archival server is a custom Java server which accepts an incoming socket connection on port 9000 and post authentication and authorization receives the compressed 3D video data. It then segments this video data in 10 seconds Transport Stream (.ts) files and updates the associated playlist

file (.m3u8). The playlist file along with the ts file used for HTTP Live Streaming (HLS). HLS allows standard HTTP servers like Apache server to be used for streaming live video data [9]. The video segments and the playlist files are stored on Amazon S3 storage.

The audience wishing to view the livevideo can connect to the HTTP server using client software capable of playing 3D HEVC video over HLS. The player requests for the playlist file and parses the links for the ts files, downloads and plays back the ts files. It also keeps requesting the updated playlist file with the new ts entries. As the stream is live, a playlist contains maximum three segments of ten seconds each.

As there is very small number of audience connected to the server at given point of time we don't have to worry about the scaling of the system. A single Apache web server on m3.medium EC2 instance is sufficient for the given use case. The capability built for cloud enablement is used for backing up the recordings of the video of the microsurgery and to provide a live view to remote audience such as medical students who can learn by observing the real life microsurgeries. Live stream can also be used to get expert opinions from a remote surgeon who could be thousands of miles away but could still see the operation in a natural 3D mode as if being present there. This can help the expert provide valuable advice to the surgeon who is actually performing the surgery.

#### G. Cloud APIs

To provide deep cloud integration to avoid the overheads of copying the voluminous 3D video data of extremely high resolution, and latency caused by moving it across the Android framework layer, we built the cloud APIs in the lowest layer of the Android Middleware itself. All the complexities of setting up the connection and data transfer are abstracted in the API cloud communication layer. These APIs are added in the Android Open Source Project (AOSP) and a custom Android image is built which is used on the Dragonboard [10].

As shown in Fig .2, the software architecture of the Android platform, the lowest layer of the stack is the Linux kernel layer and the device drivers. Above that are the native libraries, which are written in C. The layer above that is the

Application Framework layer, which is written in Java and communicates with the native layer via Java Native Interface (JNI). Above the Application Framework layer are the applications, which are written in Java using the framework APIs. As can be seen, more we go up, more is the overhead involved in handling data. So to achieve the best possible performance while handling massive 3D video data, we must restrict ourselves to the native layer.

The cloud APIs are implemented in the native layer using C/C++ and have been optimized to minimize buffer copies. The APIs leverage the other middleware libraries to communicate with the cloud.

## V. CONCLUSION

The paper describes the design of 3D Tablet that has application in the varied fields. As we have seen the real-time requirements are stringent and the amount of processing to be done pushes even the best CPUs available today to the limit. We have also shown how the use of open source hardware and software can help in drastically reducing the time, effort and cost involved in the development.

We have also demonstrated how deeply integrating the cloud capabilities into the Tablet can improve the performance of the mobile cloud computing and provide valuable services.

## REFERENCES

- [1] Corneal Endothelial Transplant: (DSAEK, DMEK & DLEK) - Thomas John - Boydell & Brewer Ltd, 2010
- [2] Principles and practice of periodontal microsurgery, Leonard S. Tibbets, Dennis Shanelec - International Journal of Microdentistry, 2009
- [3] Qualcomm website "<http://www.qualcomm.co.in/snapdragon>"
- [4] Dragonboard website "<http://mydragonboard.org>"
- [5] Android Developer website [www.developer.android.com](http://www.developer.android.com)
- [6] <http://lwn.net/Articles/565469/>
- [7] David Katz, Rick Gentile, "Embedded Media Processing", Elsevier, 2005
- [8] Data, Muller, K. et al. 3D High-Efficiency Video Coding for Multi-View Video and Depth Data, , *IEEE Transactions on Image Processing*, Vol.22 , Issue: 9, September 2013
- [9] HTTP Live Streaming - <https://tools.ietf.org/html/draft-pantos-http-live-streaming-14>
- [10] Android Platform Developer Guide "<https://source.android.com>"