بسم الله الرحمن الرحيم

PPU College of Engineering and Technology
The Home of Competent Engineers and Researchers

College of Engineering and Technology

Electrical and Computer Engineering Department

Communication and Electronics Engineering

Bachelor Thesis

Graduation Project

"Sending TV channels over Wi-Fi Networks"

Project Team

Mohammad Hasan Banat

Wisam Akram Nasriya

Ja'far Naser Masharqa

Project Supervisor

Dr. Murad Abusubaih

Hebron - Palestine
2012-2013

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

PALESTINE POLYTECHNIC UNIVERSITY

Graduation Project

"Sending TV channels over Wi-Fi Networks"

Project Team

Mohammad Hasan Banat

Wisam Akram Nasriya

Ja'far Naser Masharqa

According to the system of the College of Engineering and Technology, and to the recommendation of the Project Supervisor, this project is presented to Electrical and Computer Engineering Department as a part of requirements of B.Sc. degree in Electrical Engineering – Communication and Electronic Engineering.

Project Supervisor signature

Dr. .....

Testing Group signature

Dr. ..... Dr. .....

Department Headmaster signature
Dr. Ramzi Qawasmeh

.....

Hebron-Palestine

جامعة بوليتكنك فلسطين
الخليل – فلسطين
كلية الهندسة والتكنولوجيا
دائرة الهندسة الكهربائية

"Sending TV channels over Wi-Fi Networks"

محمد حسن بنات وسام أكرم ناصرية
جعفر ناصر مشارقه

بناء على نظام كلية الهندسة والتكنولوجيا واشراف ومتابعة المشرف المباشر على المشروع ومتابعة أعضاء اللجنة الممتحنة، تم تقديم هذا المشروع الى دائرة الهندسة الكهربائية والحاسوب ، وذلك استكمالا لمتطلبات درجة البكالوريوس في تخصص هندسة الاتصالات والالكترونيات.

توقيع المشرف

توقيع اللجنة الممتحنة

--- --- ---

توقيع رئيس الدائرة

DEDICATION

To all who helped, stand with, support and guide us from the beginning until we reach this stage.

To our Supervisor:
Dr. Murad Abusubaih

To our families, friends and whom we love.

ACKNOWLEDGMENT

We would like to extend our special thanks to everyone who has helped us in this work.

We would like to express our deepest appreciation and thanks to our supervisor Dr.Murad Abusubaih for his valuable guidance and support. His cooperation and detailed approach along with his encouragement were the major factors to get this work. It was a great pleasure to work under his supervision through these semesters.

Thanks to our teachers.

Special Thanks to:

Eng. Omar Abusaif

Eng. Sami Al-Salameen

For their guidance and support throughout this project.

Also our thanks to every person helped us in this project.

Abstract

Every day humans find new methods or create modern technologies to solve the problems that appear in the world. This aims to make the life easier using available resources. In this project, we try to apply a new idea that enables people to watch TV on their mobile devices using Wi-Fi technology. This will be achieved by streaming the video signal from a digital receiver over a Wi-Fi network. This requires a separation of the audio and video signals.

Throughout this project, we will make availability for the users to use this system for watching TV channels and for surfing the internet at the same time on the same network, this makes our system more interesting and more helpful for users. We will solve all the challenges associated with the processing of the TV signal at the sender side and the displaying of the received signal at the Wi-Fi receiver side.

الملخص

دائما ما يفكر الانسان في استغلال المصادر المتاحة من حوله استغلالاً أمثلاً، و يسعى للإستفادة مما يحيط به من أجهزة و وسائل لتقديم ما هو جديد لنا، ومن حظنا الوافر هو انتشار أجهزة و وسائل إتصال فردية كالهواتف المحمولة المتنقلة و الهواتف الذكية و أجهزة الحواسيب الشخصية و غيرها من الأجهزة التي تدعم تقنية الواي فاي، و التي بدورها انتشرت في كل مكان من حولنا، لذلك نحن الآن أمام ثورة تكنولوجية جديدة تقتحم عالم الترفيه المنزلي؛ و هذا ما دعانا لفكرة عمل هذا المشروع و التي سنطمح من خلالها الى جعل هذه الهواتف و الحواسيب المحمولة أجهزة تلفزة لا سلكية تقوم بعرض برامج القنوات التلفزيونية على سطح شاشة المستخدم، علاوة على ذلك ستبقى امكانية تصفح الانترنت متاحة على جهاز المستخدم في الوقت نفسه لننقل تلك الشاشات من الخيال العلمي إلى أرض الواقع و إلى تجربة حية و منخفضة التكلفة نستطيع شرائها و الإستمتاع بالمشاهدة أينما نزلنا.

لذلك تقوم فكرة مشروعنا على أخذ إشارتي الصوت و الصورة معا من جهاز مستقبل التلفاز و إرسالهما لا سلكياً عبر تقنية الواي فاي بنظام يتكون من مجموعة من المكونات المادية التي يتم توصيلها و التحكم بها برمجيا بلغات برمجة معينة تناسب مكونات النظام كل على حدى ضمن مرحلة النقل للإشارتين مع المحافظة على امكانية الوصول الى الانترنت خلال مشاهدة القنوات، و سيتم من خلال هذا المشروع حل جميع الاشكاليات المتعلقة بمعالجة الاشارة قبل الإرسال لتسهيل إمكانية العرض على جهة المستقبل.

CONTENTS

List of Figures:

List of Tables:

List of Abbreviation:

TV	Television
Wi-Fi	Wireless Fidelity
A/V	Audio/Video
DTV	Digital Television
NTSC	National Television System Committee
PAL	Phase Alternating Line
USB	Universal Serial Bus
IEEE	Institute of Electrical and Electronics Engineers
WLAN	Wireless Local Area Network
LAN	Local Area Network
AP	Access Point
OFDM	Orthogonal Frequency Division Multiplexing
CCK	Complementary Code Keying
IP	Internet Protocol
RF	Radio Frequency
DVD	Digital Versatile Disc
RCA	Radio Corporation of America
O.S	Operating System
RAM	Random Access Memory
PCM	Pulse Code Modulation
PAM	Pulse Amplitude Modulation
ADC,A/D,A- to- D	Analog-to-Digital Converter
DAC,D- to- A	Digital-to-Analog Converter
ICs	Integrated Circuits
STB	Surface Transportation Board
Sync	Synchronous
VCR	Video Cassette Recorder
VBI	Vertical Blanking Interval
CVBS	Color, Vision, Blanking, Synchronization
SECAM	System Essentially Contrary to American Method
PC	Personal Computer
HI Devices	Human Input Devices
NRZI	Non Return To Zero Invert
ISDN	Integrated Services Digital Network
ASF	Advanced Streaming Format
HTML5	Hypertext Markup Language 5
CDN	Content Delivery Network
RTSP	Real Time Streaming Protocol
RTMP	Real Time Messaging Protocol
HTTP	Hypertext Transfer Protocol
UART	Universal Asynchronous Receiver-Transmitter
RXD	Received Data
TXD	Transmitted Data
GND	Ground

RTS	Request to Sender
CTS	Clear To Send
SPI	Serial Peripheral Interface
SCLK	Serial Clock
CISC	Complex Instruction Set Computer
MCU	Micro processing Unit
SRAM	Static Random Access Memory
DMA	Direct Memory Access
FTDI	Future Technology Devices International
BMOS	Bulk Only Mass Storage
API	Application Programming Interface
ABI	Application Binary Interface
POSIX	Portable Operating System Interface for Unix
LED	Light Emitting Diode
VLDO	Very Low Drop Out
LVTTL	Low Voltage Transistor Transistor Logic
3GPP	3rd Generation Partnership Project
AVI	Audio Video Interleave
FPS	Frames Per Second
MBEG4	Moving Picture Expert Group-4
RM	Real Media
VOB	Video Object File
WMV	Windows Media Video File
EMAC	Ethernet Media Access Controller
MAC	Media Access Controller
URL	Uniform Resource Locator
DHCP	Dynamic Host Configuration Protocol

Chapter one

Introduction

Chapter contents

1.1 Overview

This chapter outlines the main idea of the system about streaming combined audio and video signals via Wi-Fi technology. Moreover, it will discuss the most important challenges, objectives, needed technology, and previous studies related to the system.

1.2 Project Motivation

As we see in the last few years there is a dramatic trend for the new technology devices around the world which includes laptops, smart phones, notebooks, and PDA's. Those new devices which are Wi-Fi enabled are almost in every home.

The motivation of this project therefore, is to make use of this technology to enable TV watching over mobiles, laptops and PDA's without needing extra charges. This system provides user enhancements to watch TV alone in his room without needing to settle with family, meaning multi TV at the same time in one house.

Wireless communication is needed to solve difficulties and problems caused by wired networks. It is suitable for small areas and indoor buildings. This applies for Wi-Fi technology which works with no physical wired connection between sender and receiver using radio frequency (RF); a frequency within the electromagnetic spectrum associated with radio wave propagation. This makes implementation more flexible because we don't have to deal with wires for video streaming. The system is also appropriate for limited spacing building more than traditional TV's. Moreover, we expect that the system is cheap enough to have a good chance to compete traditional TV by achieving this project in the future and make it available in markets. So this will lead to huge potential in the market.

1.3 Project Challenges

Despite of the good advantages for wireless communication systems, there are some challenges that may affect system performance. These challenges may come from the surrounding environment.

One of the most important challenges faces the wireless communication systems is the data rate which is a very important consideration in data communications. Data rate is how fast data can be sent in bits per second over a channel. Data rate depends on three factors:

1) The bandwidth available.
2) The level of the signals.
3) The quality of the channel (the level of noise).

On the other hand, in this project we use digitalization of composite signals audio and video, applying this operation needs large number of bits per sample for the large bandwidth of video signal, and increasing samples of signal may reduce the reliability of our designed system. Furthermore, there are other problems related to latency (delay) which is made of four components: propagation time, transmission time, queuing time and processing delay.

Video signal streaming over Wi-Fi involves many problems related to compression and the surrounding environment.

- The problem for video streaming as a result of compression:
  - Highly variable bit rate.
  - Inter-frame data dependency.
  - Some frames are more important than others.
- Video over wireless challenges, system is wireless so normally video streaming face these problems :
  - Interference, path loss.
  - Limited number of channels in unlicensed bands.
  - Channel characteristics constantly change (dynamic).
- Medium access non-deterministic (802.11 is originally designed for data).
- Inter-stream synchronization.
  - Between audio rendered at remote speakers and video.
  - Between one video stream and multiple audio streams.
- High probability of packet loss.[1]

In general, the system faces the following challenges:

1) Data Rate Requirement.
2) Processing and digitizing TV signals.
3) Separating audio and image.
4) Developing software for displaying videos over laptop devices.

Furthermore, the band width of the requirement for the audio and video signals is different and the synchronization between audio and video is another challenge.

1.4 Project Objectives

This project aims to achieve the following objectives:

1) Solve the limitation of spaces in homes by using one laptop instead of many TVs.
2) Mobility in watching TV by enabling people watching TV everywhere at home.
3) Saving time by doing different tasks at the same time on the laptop.
4) Design video streaming over Wi-Fi.
5) Develop a system of reduced cost.
6) Develop a system that does not incur modification at the digital TV receiver side.
7) Develop a system which provides people more comfortable life and some kind of privacy.
8) Saving money by allowing TV watching in any room without needed to buy a TV for every single room.

1.5 Project Idea

The approach we intend to follow for designing and implementing the system is depicted in figure 1.1.

The idea is to take the TV signal from the digital TV receiver and process it to make it suitable for transmission through the Wi-Fi Technology. This is done by combining the audio and video signal from the output A/V interface of the digital TV receiver by wired cables to circuits which do analog to digital conversion. We used a small device called Easy Cap which is capable of receiving audio and video signals together and produce digital signal .The signal will be delivered then to a device for transmission over Wi-Fi Technology. This will be achieved by developing a device capable of sending the signals over Wi-Fi Technology to a mobile device that displays it on the screen.

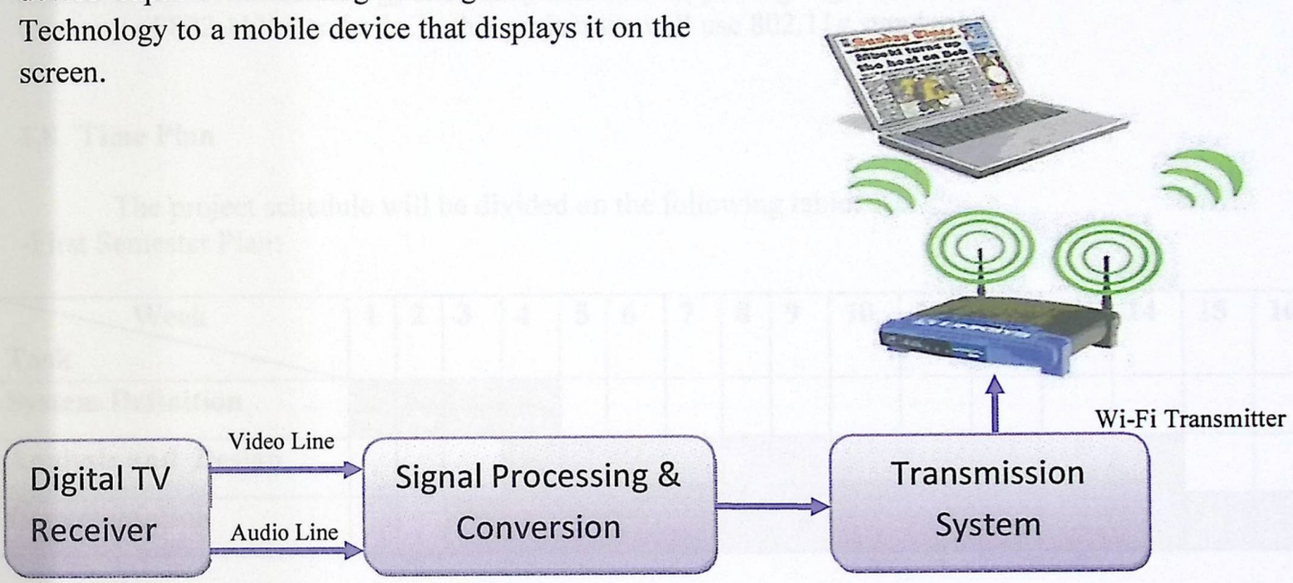


Figure 1.1: Depiction of the system

1.6 Related Work

The previous studies related to the idea of this project is (wireless Technology Personality Television) [2] .The idea depends on wireless headphone which use ZigBee technique to control transmitting of audio signal from TV to head phone and transferring audio signal between stations. However, the system we intend to develop is different as it deals with both audio and video signals. Moreover, the system is comprised of additional parts related to the processing and transmission of signals.

Another related project is (Free Space Laser Video Transmission) [3]. The main idea of this project is to design and implement a laser communication system which can be used as wireless communication system to transmit and receive video in free space.

By comparison, in this project the operation is different and more complex. That is because we transmit both audio and video signals from digital video receiver to EasyCap then transmit it via Wi-Fi designed device to laptop which is actually represent wireless TV with high quality.

1.7 Needed Technology

In this project we will use the Wi-Fi technology as wireless media communication to transmit output signals taken from EasyCap audio video adapter between digital TV receiver and the mobile side. Wi-Fi technology has many features supporting high data rate and availability and so on many of 802.11X standards. In this project we will use 802.11g standard.

1.8 Time Plan

The project schedule will be divided on the following table:
-First Semester Plan:

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
System Definition																
Analysis and Design																
Documentation																

-Second Semester Plan:

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
System Definition																
Analysis and Design																
System Implementation																
Testing and Analysis																
Documentation																

1.9 Estimated Cost of components

The following table shows the estimated cost for each component which is used in the designed system.

Components	Number of units	Cost
EasyCap USB 2.0 Audio/Video Adapter	1	40$
GESBC-9302E Board	1	200$
Wi-Fi router	1	30$
USB cable	1	2$
A/V Signal Distributor	3	5$
A/V Cable	1	3$
Total cost		280$

Chapter Two

Background

Chapter contents

2.1 Overview

This chapter outlines the main idea of the system about streaming the combined audio and video signal via Wi-Fi technology. Moreover, it will discuss the component of the system, and needed technology.

2.2 Wi-Fi Technology

In this project we will use the Wi-Fi technology as wireless media communication to transmit output signals which taken from EasyCap audio video adapter between digital TV receiver to the laptop side. Wi-Fi technology is used because it has many properties for user including data rate, easy implementation .In this project we will use the 802.11g technology.

Wi-Fi is the name of a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections. A common misconception is that the term Wi-Fi is short for "wireless fidelity," however this is not the case. Wi-Fi is simply a trademarked term meaning IEEE 802.11x.

The Wi-Fi Alliance, the organization that owns the Wi-Fi (registered trademark) term specifically defines Wi-Fi as any "wireless local area network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards".

Initially, Wi-Fi was used in place of only the 2.4GHz 802.11b standard; however the Wi-Fi Alliance has expanded the generic use of the Wi-Fi term to include any type of network or WLAN product based on any of the 802.11 standards, including 802.11b, 802.11a, dual-band, and so on, in an attempt to stop confusion about wireless LAN interoperability.

Wi-Fi works with no physical wired connection between sender and receiver by using radio frequency (RF) technology, a frequency within the electromagnetic spectrum associated with radio wave propagation. When an RF current is supplied to an antenna, an electromagnetic field is created that then is able to propagate through space. The cornerstone of any wireless network is an access point (AP). The primary job of an access point is to broadcast a wireless signal that computers can detect and "tune" into. In order to connect to an access point and join a wireless network, computers and devices must be equipped with wireless network adapters. [4]

2.2.1 Wi-Fi 802.11g

In July 1999, the IEEE 802.11g subcommittee was tasked to extend the 2.4-GHz unlicensed spectrum to data rates faster than 20 Mbps which achieve suitable data rate for this project. The resulting 802.11g standard was ratified in June 2003. The 802.11g standard provides optional data rates of up to 54 Mbps, and requires backward compatibility with 802.11b devices to protect the substantial investments in today's WLAN installations [5].

The 802.11g standard includes mandatory and optional components. It specifies OFDM (the same technology used in 802.11a) and CCK as the mandatory modulation schemes with 24 Mbps as the maximum mandatory data rate, but it also provides for optional higher data rates of 36, 48, and 54 Mbps [5].

In general 802.11 g has many features which encourage us to use it in the project as following features.

1) It is an extension to 802.11b.
2) It also operates in the 2.4 GHz band.
3) It supports up to 54Mbps data exchange rate.
4) It is backward compatible with 802.11b. All devices supporting 802.11b can operate on 802.11g networks at a lower speed, i.e., 11Mbps.
5) It also offers high-speed access to data at up to 300 feet from base station.

2.3 Wireless ADSL Modem Router

A router is a device that forwards data packets between computer networks, creating an overlay internetwork. A router is connected to two or more data lines from different networks. When a data packet comes in one of the lines, the router reads the address information in the packet to determine its ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey. Routers perform the "traffic directing" functions on the Internet. A data packet is typically forwarded from one router to another through the networks that constitute the internetwork until it reaches its destination node.

The most familiar type of routers are home and small office routers that simply pass data, such as web pages, email, IM, and videos between the home computers and the Internet. An example of a router would be the owner's cable or DSL modem, which connects to the Internet through an ISP. More sophisticated routers, such as enterprise routers, connect large business or ISP networks up to the powerful core routers that forward data at high speed along the optical fiber lines of the Internet backbone. Though routers are typically dedicated hardware devices, use of software-based routers has grown increasingly common [6].

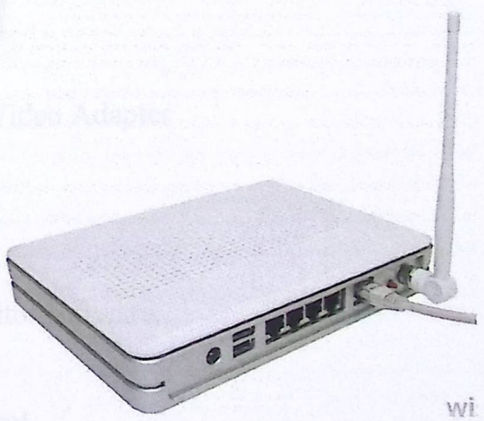


Figure 2.1: Wi-Fi enabled Router

2.4 EasyCap video Capture with audio

2.4.1 Overview

The EasyCap series USB 2.0 Video Grabber, they can capture High-quality Video and Audio file direct by USB 2.0 interface without sound card. However, the installation is very simple and the external power is unnecessary. Solution for laptop, the manufactures have enclosed the professional video editing software Ulead Video Studio 10.0 SE DVD then provide best editing function for you. Video Studio is video-editing software that makes editing your movies as fun as shooting them. The new Video Studio Movie Wizard helps novice users finish stylish movies in only three steps. Share finished projects on DVD, tape, the Web, and mobile devices. High-speed rendering and real-time performance mean less time waiting and more time creating. By the way, you can create many special effect and clip video files...etc [7].

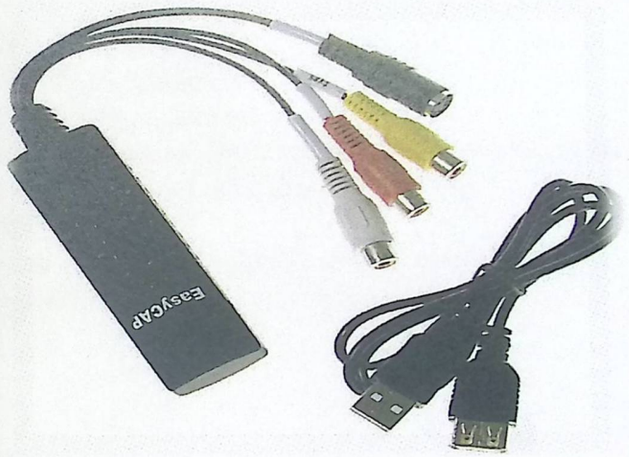


Figure 2.2: EasyCap USB 2.0 Audio-Video Adapter

2.4.2 Key Features

1) Include professional and easy to learn & used video editor software.
2) Popular USB 2.0 interface and not need other power.
3) Capture audio and video though USB 2.0 interface.
4) Support brightness, contrast, HUE, and saturation control.
5) The dimension suitable that is easy to carry.
6) Could capture audio without the sound card.
7) High plug & play.
8) Support for all formats: record in DVD+/-R/RW, DVD+/-VR, and DVD-video.
9) Applying to internet conference / net meeting.

2.4.3 Specification

1) Complies with universal serial bus specification Rev. 2.0
2) Supports NTSC, PAL, and video format.
3) Video input: one RCA composite, one S-Video.
4) Audio input: stereo audio (RCA) mm.
5) Dimension (L) 88 mm × (w) 28mm ×(H) 18mm.
6) USB bus power.
7) Supports high quality resolution.
    i. NTSC: 720 ×480 @ 30 fps
    ii. PAL: 720 ×576 @ 25 fps

2.4.4 System Requirements

1) USB: Compliant USB 2.0 free port.
2) OS: Windows XP, Vista, Seven.
3) CPU: Pentium3 800 MH_z or above.
4) HD: 600 MB of available hard drive space for program installation file, format support 4GB+ hard drive space for video capture and editing.
5) Memory: 256 MB of RAM.
6) Display: Windows - compatible display with at least 1024×768.
7) Sound card compatible Windows-sound card.
8) Digital signal [7].

2.5 Digitalization Process

A digital signal is a pulsed signal. It is depicted as discretely variable (on- off) against the analog signal which is continuously variable.

Digital signal is represented in many ways. It is represented by two voltages. Here 0 volts represent 0 in binary and +5 volts representing 1 in binary. A digital signal has the following characteristic:

1) Holds a fixed value for a specific length of time.
2) Has sharp, abrupt change.
3) A present number of values allowed.

Each pulse (on \off) is known as binary digit (bit) the number of bits transmitted per second is the bit rates of the signal [8].

2.5.1 Digitization

The process of converting analog signal in to digital signals is called digitization. It involves the modulation process called pulse code modulation .With digital transmission system, the quality of the system can be improved .The digital systems in comparison with the analog provides a better switching interface, enhance easier multiplexing and produce clear signals. To convert analog signals to digital signals, a coding system called (PCM) is used. The process of digitization generally involves four steps as shown in Fig 2.3.

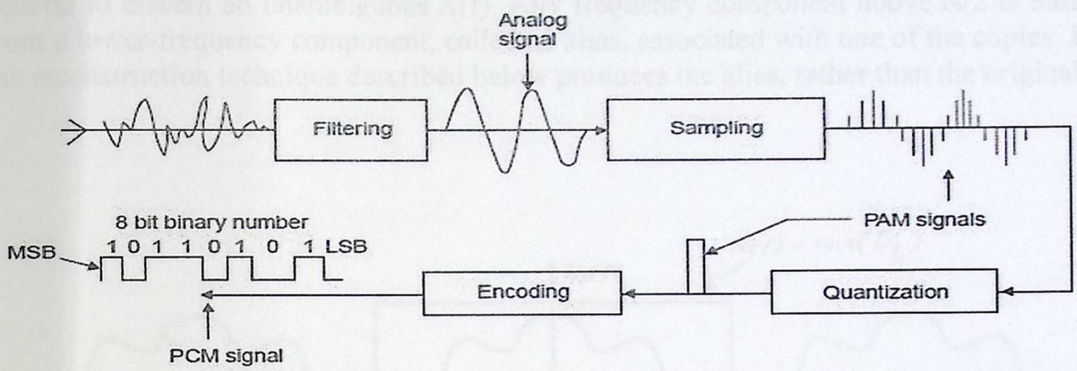


Figure 2.3: The process of digitization

2.5.2 Sampling

The sampled signal from analog signal should carry sufficient information so as to receive at the receiver with minimum distortion. The higher sampling rate contains sufficient information but increase in rate extends the bandwidth required for transmitting the sampled signal. By principle, the sampled signal can be ideally recovered exactly when $fs = 2fm$; Where fs = sampling frequency, fm = highest frequency of the band limited signal.

An analog signal that varies quickly must be sampled more frequently than an analog signal that varies slowly. The sampling period Ts is the spacing between two adjacent samples, i.e., seconds per sample.

The sampling rate or frequency fs is the number of samples per second (Hz). Sampled values of an analog signal s(t) is Ts = 1 / fs

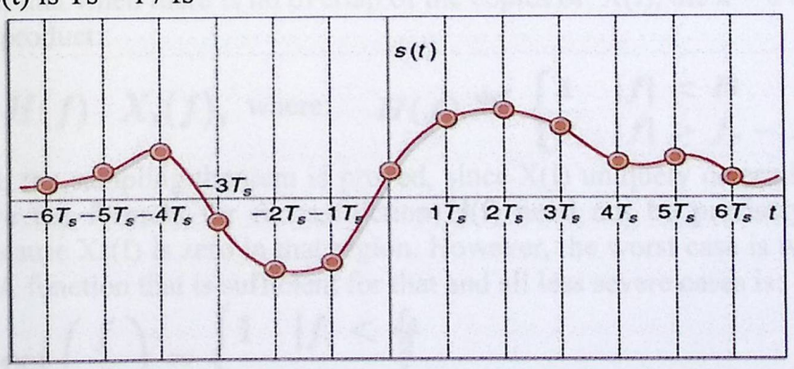


Fig 2.4: sampled values of an analog signal s(t)

The Poisson summation formula shows that the samples, x(nT), of function x(t) are sufficient to create a periodic summation of function X(f). The result is (T=1/fs):

$$X_s(f) \stackrel{\text{def}}{=} \sum_{k=-\infty}^{\infty} X(f - kf_s) = \sum_{n=-\infty}^{\infty} \underbrace{T \cdot x(nT)}_{x[n]} e^{-i2\pi nTf} \qquad (2.1)$$

Which is a periodic function and its equivalent representation as a Fourier series, whose coefficients are x[n]. This function is also known as the discrete-time Fourier transform (DTFT). As depicted in Figures 2.5, copies of X(f) are shifted by multiples of fs and combined by addition.

If the Nyquist criterion is not satisfied, adjacent copies overlap, and it is not possible in general to discern an unambiguous X(f). Any frequency component above fs/2 is indistinguishable from a lower-frequency component, called an alias, associated with one of the copies. In such cases, the reconstruction technique described below produces the alias, rather than the original component.

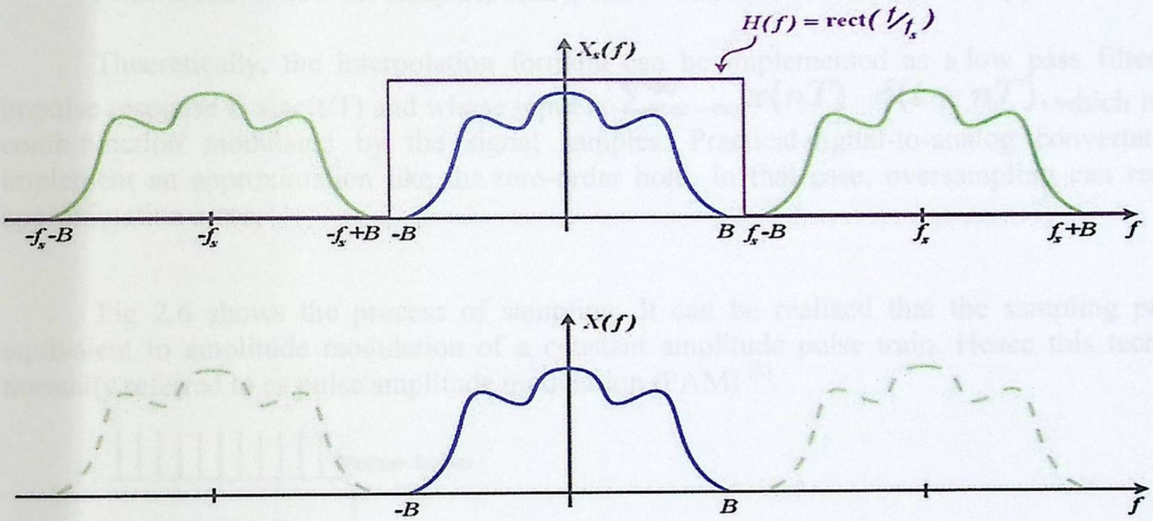


Fig 2.5: Spectra of discrete-time Fourier transform (DTFT) signal.

It is apparent that when there is no overlap of the copies of X(f), the k = 0 term of Xs(f) can be recovered by the product:

$$X(f) = H(f) \cdot X_s(f), \text{ where: } H(f) \stackrel{\text{def}}{=} \begin{cases} 1 & |f| < B \\ 0 & |f| > f_s - B. \end{cases} \qquad (2.2)$$

At this point, the sampling theorem is proved, since X(f) uniquely determines x(t). All that remains is to derive the formula for reconstruction. H(f) need not be precisely defined in the region [B, fs − B]because Xs(f) is zero in that region. However, the worst case is when B = fs/2, the Nyquist frequency. A function that is sufficient for that and all less severe cases is:

$$H(f) = \text{rect}\left(\frac{f}{f_s}\right) = \begin{cases} 1 & |f| < \frac{f_s}{2} \\ 0 & |f| > \frac{f_s}{2}, \end{cases} \qquad (2.3)$$

where rect( ) is the rectangular function. Therefore:

$$X(f) = \operatorname{rect}\left(\frac{f}{f_s}\right) \cdot X_s(f) \tag{2.4}$$

$$= \operatorname{rect}(Tf) \cdot \sum_{n=-\infty}^{\infty} T \cdot x(nT)\, e^{-i2\pi nTf} \tag{2.5}$$

$$= \sum_{n=-\infty}^{\infty} x(nT) \cdot \underbrace{T \cdot \operatorname{rect}(Tf) \cdot e^{-i2\pi nTf}}_{\mathcal{F}\left\{\operatorname{sinc}\left(\frac{t-nT}{T}\right)\right\}}. \tag{2.6}$$

The inverse transform of both sides produces the Whittaker–Shannon interpolation formula:

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \cdot \operatorname{sinc}\left(\frac{t-nT}{T}\right), \tag{2.7}$$

, which shows how the samples, x(nT), can be combined to reconstruct x(t).

Theoretically, the interpolation formula can be implemented as a low pass filter, whose impulse response is sinc(t/T) and whose input is $\sum_{n=-\infty}^{\infty} x(nT) \cdot \delta(t-nT)$, which is a Dirac comb function modulated by the signal samples. Practical digital-to-analog converters (DAC) implement an approximation like the zero-order hold. In that case, oversampling can reduce the approximation error.

Fig 2.6 shows the process of sampling .It can be realized that the sampling process is equivalent to amplitude modulation of a constant amplitude pulse train. Hence this technique is normally referred to as pulse amplitude modulation (PAM) [8].

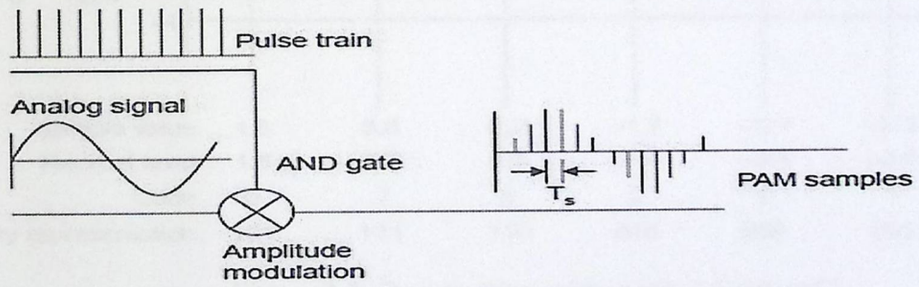


Figure 2.6: Generation of PAM samples

In PAM sample generator is an amplitude modulator. In simple version, it contains a AND gate. The signal to be converted is fed to one input of the AND. Pulses at the sampling frequency are applied to the other input of the AND gate to open it during the required time intervals. Thus, the output of the gate consists of pulses at the sampling rate, equal in amplitude to the signal voltage at each instant. The pulses are then passed through a pulse shaping network which gives the flat tops.

2.5.3 Quantization & Encoding

Instead of sending a pulse train capable of continuously varying one of the parameters, the PCM technique produces a series of binary code which represents the approximate amplitude of the signal sample at that instant. This process of approximation is called signal quantization. Thus, the combined operations of sampling and quantizing generate a quantized PAM wave form.

The quantizing process is as follows, let the signal $m(t)$ is to be quantized. Identify the signal excursion (negative peak to positive peak) which is ranges from VL to VH. Divide the total range into M equal intervals. In the center of each of these steps, we locate quantization levels. If the sampled amplitude is in a particular interval, the nearest value is chosen and the corresponding binary digit is approximated as equivalent to the sampled amplitude. The quality of the approximation may be improved by reducing the size of the steps, thereby increasing the number of allowable levels. Fig 2.7 shows the process of quantization and encoding [8].

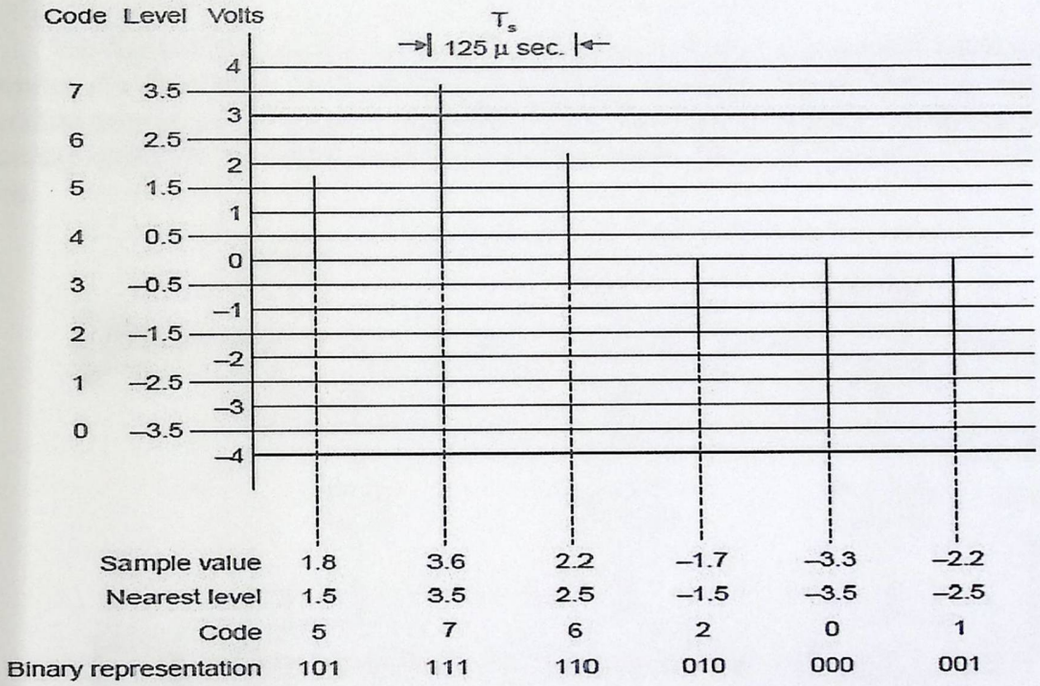


Figure 2.7: Quantization with number levels of 8

In the fourth step of the digitization process, the quantized samples are encoded into a digital bit stream (series of electrical pulses). In practical system, 256 levels can be used to obtain quality of signal. This results in the output of the encoding as 8 bit number. This binary number (or 8 bit word) is transmitted over the network as a series of electrical or optical pulses.

In the case of 256 (0 to 255) levels, the digital encoder recognizes the 255 different voltage levels of the quantized samples, then converts each level into a string of eight bits (1*s* and 0*s*). The series of pulses is called a digital bit stream [8].

In computers, encoding is the process of putting a sequence of characters (letters, numbers, punctuation, and certain symbols) into a specialized format for efficient transmission or storage. Decoding is the opposite process the conversion of an encoded format back into the original sequence of characters. Encoding and decoding are used in data communications, networking, and storage.

The terms encoding and decoding are often used in reference to the processes of analog-to-digital conversion and digital-to-analog conversion. In this sense, these terms can apply to any form of data, including text, images, audio, video, multimedia, computer programs, or signals in sensors, telemetry, and control systems. Encoding should not be confused with encryption, a process in which data is deliberately altered so as to conceal its content. Encryption can be done without changing the particular code that the content is in, and encoding can be done without deliberately concealing the content.

The figure 2.8 show the components in coding system, it's in general consist of Encoder and Decoder ,the first stage is encoding which taking the video signal from its source, then video encoding encompasses digitizing and compressing video that is originally in analog form. And video decoding encompasses decompressing an encoded video file, and displaying the video in an analog form.

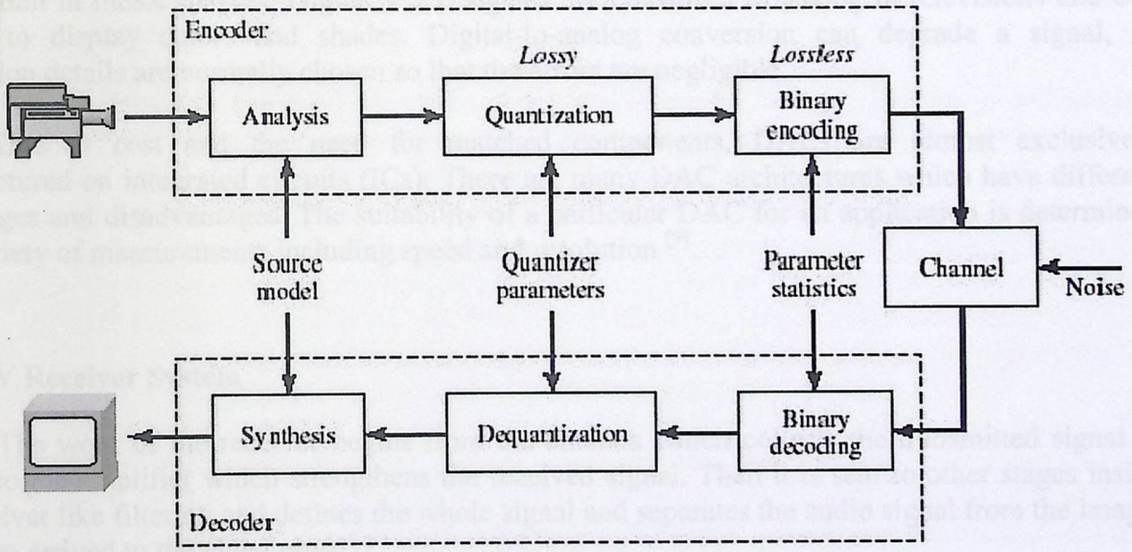


Figure 2.8: Components of coding system

2.6 Analog-to-Digital Converter

An analog-to-digital converter (abbreviated ADC, A/D or A to D) is a device that converts the input continuous physical quantity to a digital number that represents the quantity's amplitude. The conversion involves quantization of the input, so it introduces a small amount of error. The inverse operation is performed by a digital-to-analog converter (DAC). Instead of doing a single conversion, an ADC often performs the conversions ("samples" the input) periodically. The result is a sequence

of digital values that have converted a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal. An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number proportional to the magnitude of the voltage or current. However, some non-electronic or only partially electronic devices, such as rotary encoders, can also be considered ADCs.

The digital output may use different coding schemes. Typically the digital output will be a two's complement binary number that is proportional to the input, but there are other possibilities. An encoder, for example, might output a Gray code [9].

2.7 Digital-to-Analog Converter

In electronics, a digital-to-analog converter (DAC or D-to-A) is a device that converts a digital (usually binary) code to an analog signal (current, voltage, or electric charge). An analog-to-digital converter (ADC) performs the reverse operation. Signals are easily stored and transmitted in digital form, but a DAC is needed for the signal to be recognized by human senses or other non-digital systems.

A common use of digital-to-analog converters is generation of audio signals from digital information in music players. Digital video signals are converted to analog in televisions and cell phones to display colors and shades. Digital-to-analog conversion can degrade a signal, so conversion details are normally chosen so that the errors are negligible.

Due to cost and the need for matched components, DACs are almost exclusively manufactured on integrated circuits (ICs). There are many DAC architectures which have different advantages and disadvantages. The suitability of a particular DAC for an application is determined by a variety of measurements including speed and resolution [9].

2.8 DTV Receiver System

The work of the receiver begins from the antenna which collects the transmitted signal to pass it to the amplifier which strengthens the received signal. Then it is sent to other stages inside the receiver like filtering and defines the whole signal and separates the audio signal from the image, the video arrived to the video plug.

- What does a DTV receiver do?
  - Receives digital TV broadcasts from a cable, satellite or terrestrial network.
  - Decodes them.
  - Outputs them to a television or other display device.
    - This display device is usually a television.
    - May do other things.
    - E.g. execute applications included with the broadcast.

- The part of the receiver that actually receives the analog signal and converts it into a stream of bits.
  - Two main components, the tuner and the front end.
  - These may be integrated into a single package.
  - Usually specific to cable, satellite or terrestrial networks.
  - Some may support both analog and digital signals.
- The tuner.
  - Receives a signal on the frequency specified by the rest of the STB.
  - Demodulates the signal.
  - Turns the analog signal into a digital bit-stream.
  - The front-end.
  - Performs first level of error correction.
  - Removes the first level of packetization in the stream.
  - Outputs an MPEG-2 transport stream in digital format [10].

2.8.1 Analogue Video Signal

A video signal is an electrical voltage that fluctuates over time. These fluctuations are interpreted by a TV set while an electron beam is zigzagging across the screen to draw a picture, line by line. This is also known as scanning, and a single line as a scan line. The shape of the signal fluctuations tell the TV how bright a certain spot on the screen should be (luma or Y) and what color it should have (Chroma or C).

When a pulse of a certain shape in the signal occurs, the TV knows a line has ended and that the beam should travel back to the other edge of the screen to start drawing a new one. This is called horizontal retrace; the pulse is known as horizontal sync. While it is traveling back the beam is switched off so you don't see the trace. During the part of the video signal at which this is happening is called horizontal blanking or horizontal delay. For each video line coming into a TV, only about 5/6th is actually being drawn and about the first 1/6th of the total time is spent traveling back to the starting position. During the horizontal blanking, another synchronizing signal is included called color burst. It contains information needed to decode the color in the part of the line you do see. A video source like a VCR produces an endless string of video lines [11].

A specially patterned video line, vertical sync, indicates that a picture is finished. At this point the beam is located at the bottom of the screen and has to travel up again to be able to begin drawing the next picture. With the electronics of the first TV's this would take some time. This time interval is known as the vertical retrace time. During this time, analogue video line signals will keep coming in at regular intervals from the TV tuner or the VCR, but the beam is switched off, so these lines are not drawn on the screen. In the digital age, all kinds of extra information like teletext are digitally encoded in these video lines. This group of lines is known as the vertical blanking interval area (or VBI abbreviated) and the encoded data as VBI data. The lines that result in the picture lines are called active scan lines.

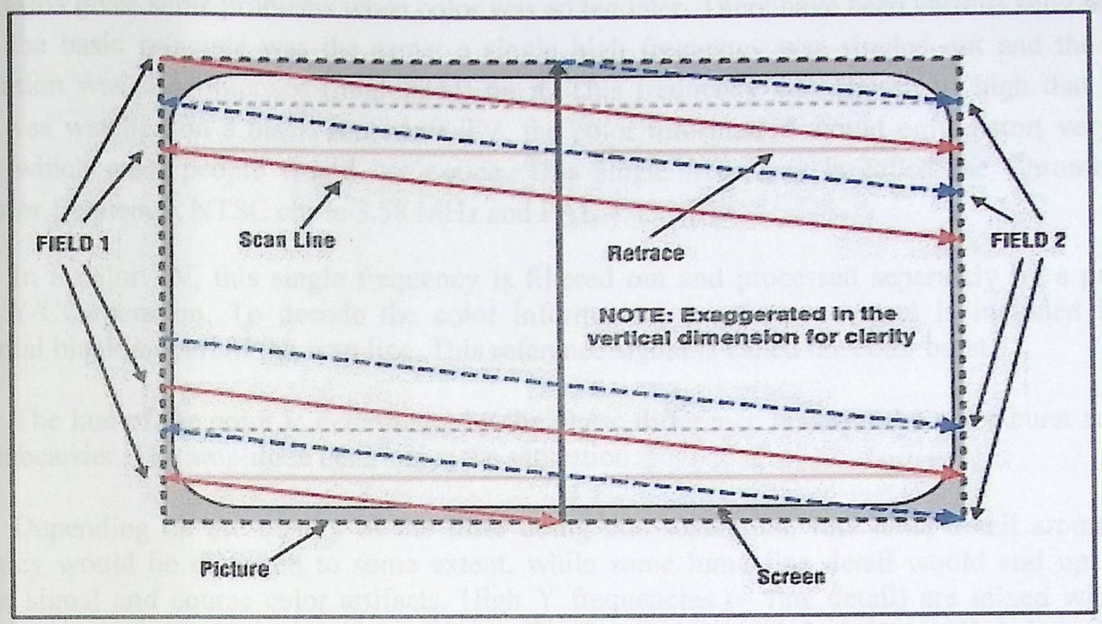


Figure 2.9: Interlaced scanning system

Fig. 2.10 shows the waveform of a NTSC scan line. In the active part of the signal, a low signal results in the scan line being black while a high signal results in it being white. Value in between results in a grey line. The signal in the example would have resulted in a horizontal white line being drawn. To have a small black dot in the middle of this line, the signal would have to dip rapidly to black level and up again. The faster the dip, the smaller the dot would be. This brings us to another point: to show fine detail, the signal needs to be able to change rapid. Rapid changes in a signal equal high frequencies, since frequency is another word for changes per time unit.

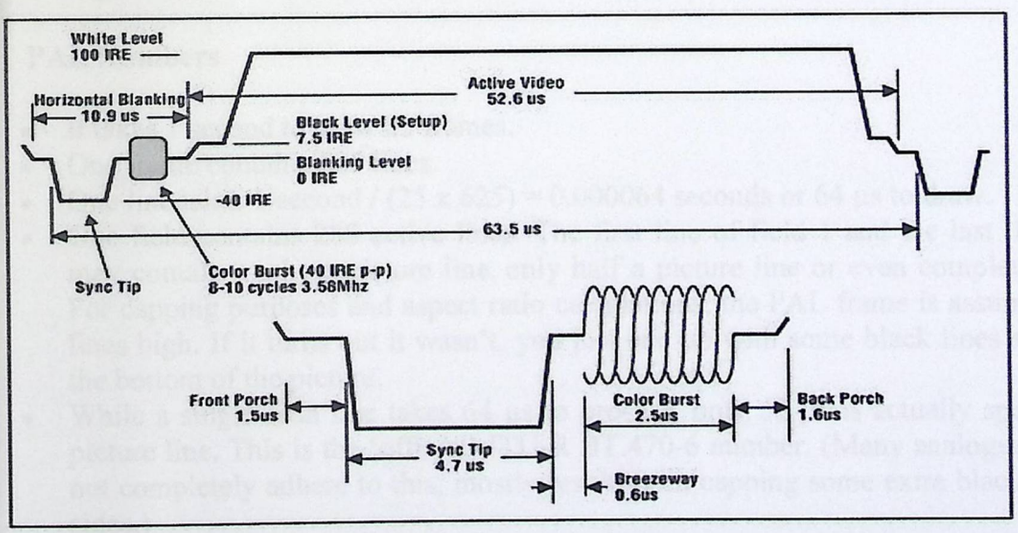


Figure 2.10: A typical waveform of a NTSC composite video signal

This gives some problems when color was added later. There have been various solutions for it, but the basic principle was the same: a single high frequency was singled out and the color information was superimposed (modulated) on it. This frequency was chosen so high that if the signal was watched on a black and white TV, the color information would only distort very fine details which most people would not notice. This single frequency is called the Chrominance subcarrier frequency. NTSC chose 3.58 MHz and PAL 4.43 MHz.

In a color TV, this single frequency is filtered out and processed separately by a process called Y/C separation. To decode the color information, a reference signal is included in the horizontal blanking part of the scan line. This reference signal is called the color burst.

The hue of the color is determined by the phase difference between the color burst and the color subcarrier. The amplitude determines the saturation.

Depending on the quality of the filter doing this separation, fine luma detail around this frequency would be damaged to some extent, while some luma fine detail would end up in the chroma signal and course color artifacts. High Y frequencies (= fine detail) are mixed with low chroma frequencies (color crosstalk = rainbow effects) and vice versa (luma crosstalk = dot crawls).

2.9 NTSC and PAL TV Systems

2.9.1 The 625/50 (PAL) and 525/60 (NTSC) timings

Depending on the frequency of the electricity mains in a country, two scan line timings have been developed: 625 lines/frame, 50 fields per second and 525 lines/frame, 60 fields per second. These are often (incorrectly) referred to as PAL and NTSC timing respectively, referring to the TV systems adopted by Germany and America. (Actually PAL and NTSC refer to the system used for color encoding, and nations with PAL 525/60 and NTSC 625/50 systems DO exist [11].

- **PAL Numbers**

  - It takes 1 second to draw 25 frames.
  - One frame contains 625 lines.
  - One line takes 1 second / (25 x 625) = 0.000064 seconds or 64 µs to draw.
  - One field contains 288 active lines. The first line of field 1 and the last line of field 2 may contain a whole picture line, only half a picture line or even completely be black. For capping purposes and aspect ratio calculations, the PAL frame is assumed to be 576 lines high. If it turns out it wasn't, you just end up with some black lines at the top and the bottom of the picture.
  - While a single scan line takes 64 µs to process, only 52 µs is actually spent drawing a picture line. This is the 'official' ITU-R BT.470-6 number. (Many analogue sources will not completely adhere to this, mostly resulting in capping some extra black pixels at the sides.)
  - The Chrominance subcarrier frequency is 4.433,6.18,7.5 MHz.

➢ NTSC Numbers

- It takes 1 second to draw 29.97 frames.
- One frame contains 525 lines.
- One line takes 1 second / (29.97 x 525) = 0.000063556 seconds or 63.556 µs to draw.
- One field contains 243 active lines. The first line of field 1 and the last line of field 2 may contain a whole picture line, only half a picture line or even be completely black. Most often, these two lines will contain only half picture information. For some reasons this has caused a lot of confusion about the active area of the NTSC frame, while this did not happen for PAL. The NTSC frame consists of 486 lines. It contains at most 2x242.5 = 485 lines of picture info. However, often only 484 lines are complete. Virtually all capture devices capture only 480 lines. Presumably that is because by capping 480, you can avoid both the first line of field 1 and the last line of field 2.
- For capping purposes always use 480 lines. For some aspect ratio calculations use 486 lines, and regard 480 as cropped from that. Forget the about the other numbers.
- While a single scan line takes 63.556 µs to process by a TV, only 52.6555 µs is actually spent drawing a picture line. This is the 'official' ITU-R BT.470-6 number, which is often used by people dealing with digital video. There are many standards for NTSC/M, all giving a slightly different number for the active part of a scan line.
- Assume 52.6555 µs if you need a number for calculations. If the analogue source used a different number, again this will result in capping some extra black pixels at the sides or cropping a bit off.
- The Chrominance subcarrier frequency is 3.579, 5.45 MHz

➢ SECAM

SECAM is the French TV system. It follows the same timing as PAL above, but encodes the color differently once the Video decoder is set to the SECAM video standard.

➢ PAL/M

This is a TV system used in amongst others Brazil. It uses NTSC timing, but PAL/M color encoding once the Video decoder is set to the PAL_M video standard.

➢ PAL-60

Modern European equipment can handle both 625/50 and 525/60 timings. It cannot handle NTSC color encoding. So if you use European 'NTSC playback' capable equipment to play back an American NTSC tape or DVD, they will produce a signal that retains the NTSC timing, but has both the color encoding and the color carrier frequency converted to PAL. Modern European TV's understand this hybrid signal and show the video. Some game consoles also produce a PAL-60 signal instead of true PAL.

To capture this, you need a device that understands PAL-60. Almost all devices do. Unfortunately, many drivers won't give you this option. If your device/driver has the option, set the Video decoder to the PAL_60 video standard [11].

2.10 USB Overview

Universal Serial Bus (USB) is a hardware interface for low-speed peripherals such as keyboards, joysticks, scanners, printers, telephony devices, and MPEG-1 and MPEG-2 digital video. The electrical design of USB limits the maximum cable length to 5 meters for full-speed devices and up to 3 meters for low-speed devices. Up to 5 hubs can be connected in series to allow for an extended cable run. Up to 127 devices (theoretical limit) can be attached to a single host computer. The hot-swap capability allows a USB device to be plugged in or unplugged without turning the system off. USB devices may be plugged into a USB receptacle on the PC, into a multi-port USB hub or into a USB device that also functions as a hub for other devices.

➢ Data Transfer

The Full-speed USB 1.1 maximum bandwidth of 12 Mbits/sec is equivalent to a data transfer speed of 1.5 Mbytes/sec with Hi-speed USB 2.0 dramatically increasing this to 480 Mbits/sec. A Low-speed rate of 1.5Mbits/sec is used for HI (Human Input Devices) such as keyboards, joysticks etc. Control, Interrupt, Bulk and Isochronous transfer modes are supported. A USB device indicates its speed by pulling either the D+ or D- line high to 3.3 volts. The pull-up resistors at the device end are also used by the host or hub to detect the presence of a device connected to the port.

Data signals are transmitted on a twisted pair labeled D+ and D− collectively using half-duplex differential signaling to combat the effects of electromagnetic noise on longer lines. D+ and D− usually operate together; they are not separate simplex connections. Transmitted signal levels are 0.0-0.3 volts for low and 2.8-3.6 volts for high. A NRZI (Non Return to Zero Invert) encoding scheme used to send data with a sync field to synchronize the host and receiver clocks [12].

➢ USB Power

The USB bus supplies 5V DC regulated power (maximum 500mA) through each port on pins 1 and 4 (pins 1 & 5 on mini-USB). These pins are longer than the data pins to ensure that the power connections mate first and un-mate last. Low-power devices that might normally require a separate AC adapter can therefore be powered via the USB cable, eliminating the need for associated AC adaptors. Bus-powered hubs derive all their power from the USB bus. Powered-hubs are powered from their own AC adapter and provide better power distribution to downstream devices. Port-switching USB hubs isolate all ports from each other so that a faulty device will not cause all other the devices on the same bus to also fail. The specification provides for no more than 5.25 V and no less than 4.35 V between bus positive and negative [13].

➢ **Plug & Receptacle Pin outs**

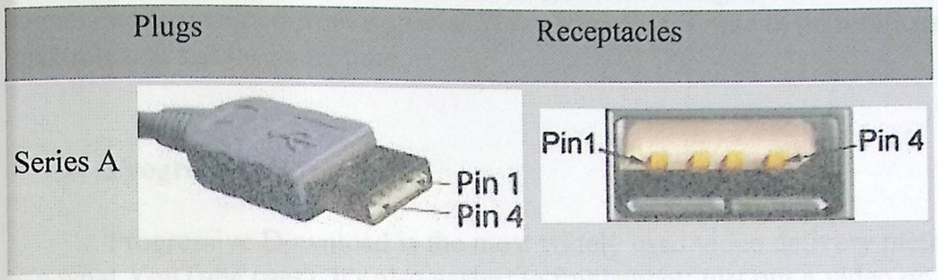


Figure 2.11: USB Series A Pin outs

➢ **Standard USB Pin out & Cable Color Code**

Table 2.1 Standard USB Pin out & Cable Color Code

Pin	Wire Color	Function
1	Red	V BUS (+5V)
2	White	D-
3	Green	D+
4	Black	Ground

2.11 Video Streaming

Streaming video is content sent in compressed form over the Internet and displayed by the viewer in real time. With streaming video or streaming media, a Web user does not have to wait to download a file to play it. Instead, the media is sent in a continuous stream of data and is played as it arrives. The user needs a player, which is a special program that uncompress and sends video data to the display and audio data to speakers. A player can be either an integral part of a browser or downloaded from the software maker's Web site [14].

Major streaming video and streaming media technologies include Real System G2 from Real Network, Microsoft Windows Media Technologies (including its NetShow Services and Theater Server), and VDO. Microsoft's approach uses the standard MPEG compression algorithm for video. The other approaches use proprietary algorithms. (The program that does the compression and decompression is called the codec.) Microsoft's technology offers streaming audio at up to 96 Kbps and streaming video at up to 8 Mbps (for the NetShow Theater Server). However, for most Web users, the streaming video will be limited to the data rates of the connection (for example, up to 128 Kbps with an ISDN connection). Microsoft's streaming media files are in its Advanced Streaming Format (ASF).

Streaming video is usually sent from prerecorded video files, but can be distributed as part of a live broadcast "feed." In a live broadcast, the video signal is converted into a compressed digital signal and transmitted from a special Web server that is able to do multicast, sending the same file to multiple users at the same time.

- **Progressive Download**

Progressive Download is the most widely used video delivery method by far (in part because it's what YouTube uses). It's also easiest to implement: just put a video on your web server and point your player to the URL. Once a user hits play, the player immediately starts downloading the file. The player will start video playback as soon as it has enough data to do so, but it will continue to download until it has received the whole file (hence the *progressive*).

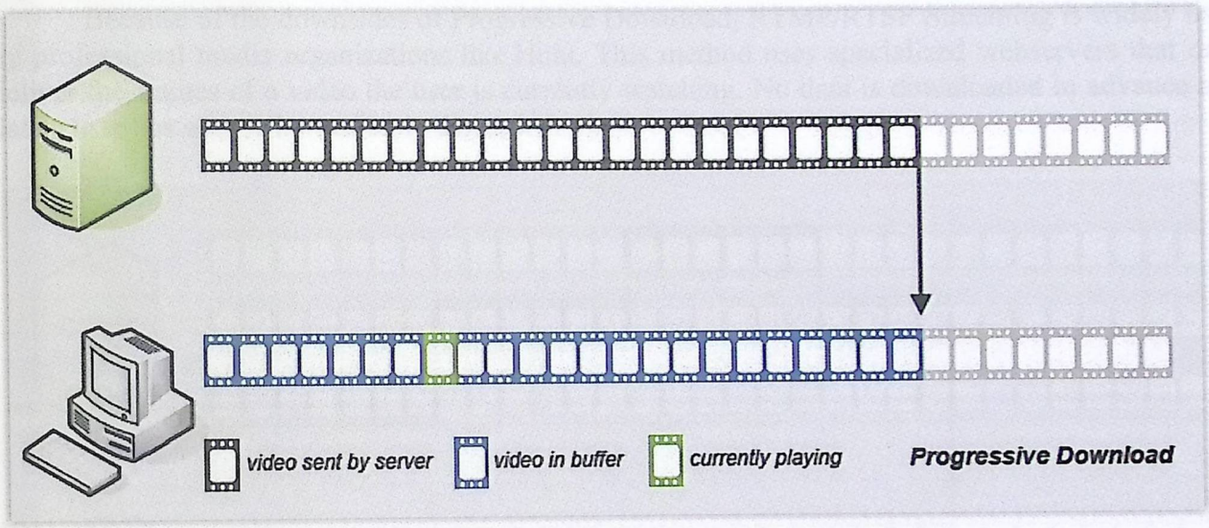


Figure 2.12: Progressive download

Progressive Download is supported by Flash, HTML5 browsers, the iPad/iPhone and Android. On the server side, every regular webhoster support downloads, as does every CDN (Content Delivery Network; webhosters that special in large-scale delivery). In most cases (Flash needs a small server module), it is possible to seek in a player to a not-yet-downloaded part of the video.

At that point which the player is be requested to play the video, the player re-downloads the video, starting at the seek offset instead of at the beginning. We call that feature *pseudo-streaming*.

The simplicity of Progressive Download also has its downsides. For one, bandwidth is wasted on data downloaded but not watched. Consider a user watching a ten minute video. They may leave the page after having watched only one minute of the video, but at that point the other nine minutes have already been downloaded. This means that the publisher has paid to transfer nine times as much data as the user actually watched - an expensive proposition on a large scale.

Another downside is the inability to change the quality of the video mid-stream: once the download starts, the video quality is locked. After switching a player to full screen, you generally see a blurry video, because it was intended to be watched at a much smaller size. Or, when you watch video on an iPad, your connection may switch from Wi-Fi to 3G. Playback then stutters, because the download speeds are much lower on 3G.

In sum, Progressive Download works fine for short clips (a few minutes). For longer videos, the downsides start to impact playback too much. Plus, live streaming is not possible, as there's no downloadable file.

- **RTSP/RTMP Streaming**

Because of the downsides of Progressive Download, RTMP/RTSP Streaming is widely used by professional media organizations like Hulu. This method uses specialized webservers that only deliver the frames of a video the user is currently watching. No data is downloaded in advance and data a user has seen is immediately discarded.

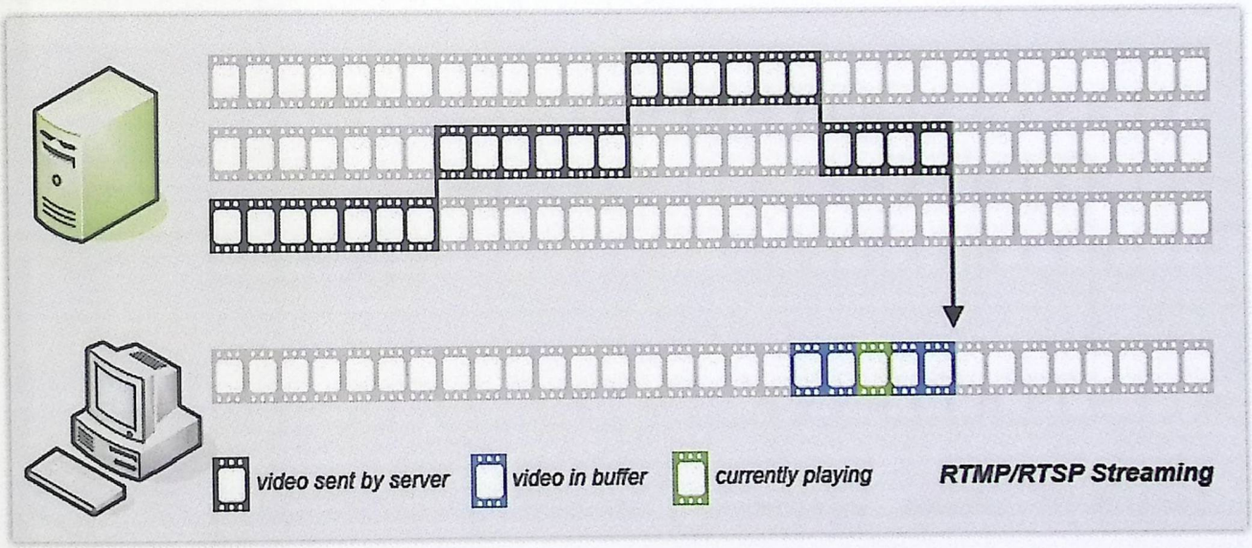


Figure 2.13: RTMP/RTSP Streaming

The most widely solution is used is RTMP (Real Time Messaging Protocol), the streaming protocol of Flash. It is supported by servers such as FMS and Wowza and most CDNs (but not by regular webhosters). HTML5 does not include a dedicated streaming protocol, nor does the iPad/iPhone. Android has support, for RTSP (Real Time Streaming Protocol). Unfortunately, RTSP is not widely supported by servers and CDNs.

This lack of support, especially on the server side, is the biggest drawback of RTMP/RTSP Streaming. Most publishers do not want to maintain expensive, dedicated servers to stream their videos. Additionally, the dedicated protocols (RTMP and RTSP) are often blocked by corporate firewalls.

On the plus side, RTMP streaming can change video quality mid-stream. This allows for optimal playback quality in the full screen and Wi-Fi/3G scenarios described above. However, if the connection speed drops below the minimum bandwidth needed for the video, playback will be continuously interrupted. Unlike progressive download, users cannot pause a video and wait for enough data to download to ensure smooth playback.

In sum, RTMP/RTSP Streaming works great even for long-form or live video. It has specific server and protocol requirements, which makes it less accessible and adds significant complexity and cost as compared to Progressive Download.

- **Adaptive HTTP Streaming**

5Adaptive HTTP Streaming is a fairly new streaming format. It attempts to join the merits of RTMP/RTSP Streaming (bandwidth efficiency, quality switching) with those of Progressive Download (no special servers or protocol needed). Adaptive HTTP Streaming works by storing your videos on the server in small fragments (a few seconds each). The player then glues these fragments together into a continuous stream.

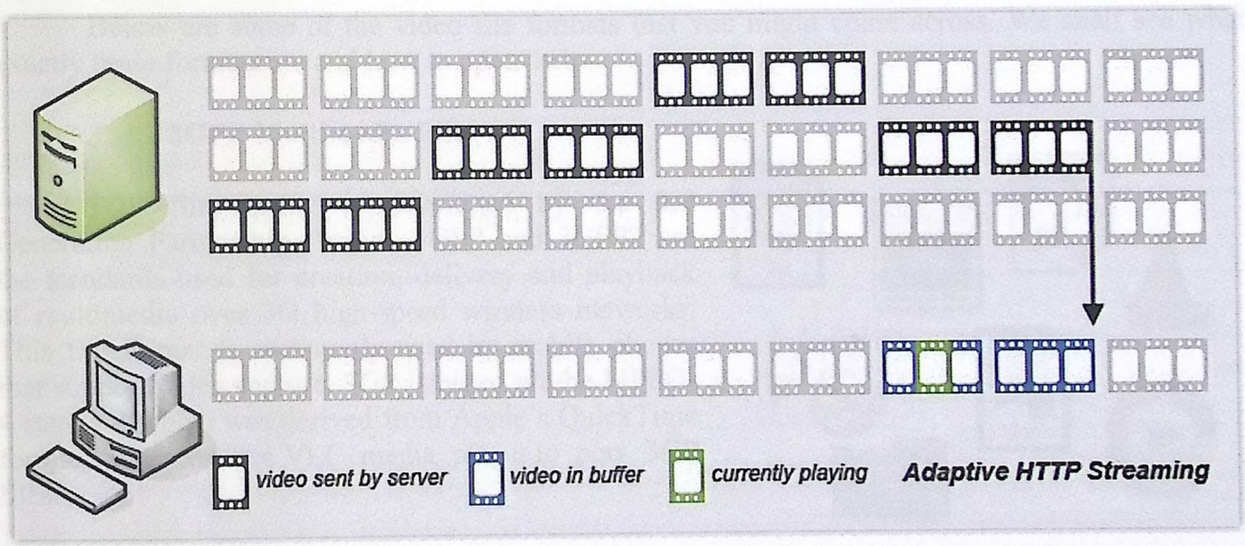


Figure 2.14: Adaptive HTTP Streaming

At present, Adaptive HTTP Streaming is supported by both Flash and the iPad/iPhone. Android supports it as of version 3 and support in HTML5 is currently under development. Since Adaptive HTTP Streaming leverages standard webservers, it is supported by webhosters and CDNs alike.

Although Adaptive HTTP Streaming eliminates many of the downsides of RTMP/RTSP streaming and Progressive Download, it still has issues of its own, the biggest being the lack of standardization. Because it is a new technology, there is no single, widely used implementation. The most popular is currently Apple's HLS (HTTP Live Streaming), which is supported by the

iPad/iPhone and Android 3.0. However, both Adobe and Microsoft have competing offerings (Zeri & Smooth) and the MPEG consortium is working on a standard named DASH.

It's also worth noting that none of the Adaptive HTTP Streaming implementations work with regular MP4 files. They all require your files to be converted from a regular MP4 into a specific fragmented format. Apple, Microsoft and Adobe each supply a tool for this, but support for these formats doesn't exist in regular video editors and transcoding tools.

In summary, while Adaptive HTTP Streaming will likely become the single video streaming method over time, the technology is still *fragmented* (no pun intended) and ecosystem support is only beginning to arrive [15].

2.12 Video Formats and Software Applications

There are numerous video formats that we come across daily. There are some common formats like 3GP, AVI, MPEG, RM and WMV. Besides these common file formats, there are many formats which might confuse you. It would be good if we know at least some of the common video formats and software applications which can be used to open them.

Below are some of the video file formats that you might come across. We shall see what exactly those formats are and how to open them.

- **3gp – 3GPP Multimedia File**

3GP file format is developed by the 3rd Generation Partnership Project. 3GPP and 3GPP2 are the standards used for creation, delivery and playback of multimedia over 3G high-speed wireless networks. This file format is commonly used by mobile phones that support video capture. 3GP is based on the MPEG-4 standard which was derived from Apple's QuickTime format. You can use VLC media player to play 3GP files.

Figure 2.15: video formats

- **asf – Advanced Systems Format File**

ASf or Advanced Systems Format File is developed for Microsoft. This format is used for streaming multimedia files which contains text, graphics, sound, video and animation. ASF file are mostly Windows Media Audio and Windows Media video files. ASF file just specifies the structure of the audio or video stream but not the encoding method. Microsoft Windows Media Player can be used to play .asf files.

- **avi – Audio Video Interleave File**

AVI file format is developed by Microsoft and stores data that can be encoded in a number of different codec's. You can use codec's like DivX to encode AVI files. AVI files can be played on multiple players like VLC; Windows Media Player and the player should support the codec in which the file was encoded.

- **mov – Apple QuickTime Movie File**

QuickTime Movie format is developed by Apple. it is a common multimedia format often used for saving movies and other video files that are played on the Internet. You can use Apple QuickTime Player to play .mov files

- **mp4 – MPEG-4 Video File**

MPEG-4 file format is a standard developed by the Moving Picture Experts Group which is used in many mobiles and video players (MP4 Players) these days. The MPEG-4 video format uses a separate compression for audio and video tracks. The video is compressed with MPEG-4 video encoding and the audio is compressed using AAC compression. You can play MP4 files using VLC media player.

- **rm – Real Media File**

Real Media is a file format developed by Real Networks. Real Media contains both video (Real Video) and audio (RealAudio) information and is usually used for streaming media files over the internet. rm files can be played using the Real Player.

- **vob – DVD Video Object File**

VOB file type is mainly associated with 'DVD Video Movie File'. A Video Object usually contains several streams multiplexed together: Video, Audio and Subtitles. You can use VLC media player to play VOB files.

- **wmv – Windows Media Video File**

Windows Media Video is one of the major video file formats that you encounter every day. The Windows Media file contains video encoded with Windows Media Video codec and audio encoded with Windows Media Audio codec. You can use Microsoft Windows Media Player to play .wmv files.

2.12.1 Factors for Choosing the Suitable Video Format

A. File size & quality

1. *Digital Storage Space* - To calculate the amount of storage space you will need for a project, digital video requires approximately 200 MB per minute of footage, or roughly 12 GB per hour. Of course this varies according to your recording device and the quality it is set to record at.

2. *Frames per Second* – The standard for FPS is 29.97, increasing the FPS allows for more images per second thus a smoother image. Decreasing FPS will make the video a bit choppy and not nearly as smooth.
3. *Video Bitrate* - Bitrate is a measurement of the number of bits that are transmitted over a set length of time. Your overall bitrate is a combination of your video stream, audio stream & metadata in your file with the majority coming from your video stream. The higher the bit rate the better the quality the bigger it will be.
4. *Resolution* – this is the number of pixels present in the images of the video. This determines whether your video is standard definition or high definition. The higher the resolution the clearer the image the bigger the file.

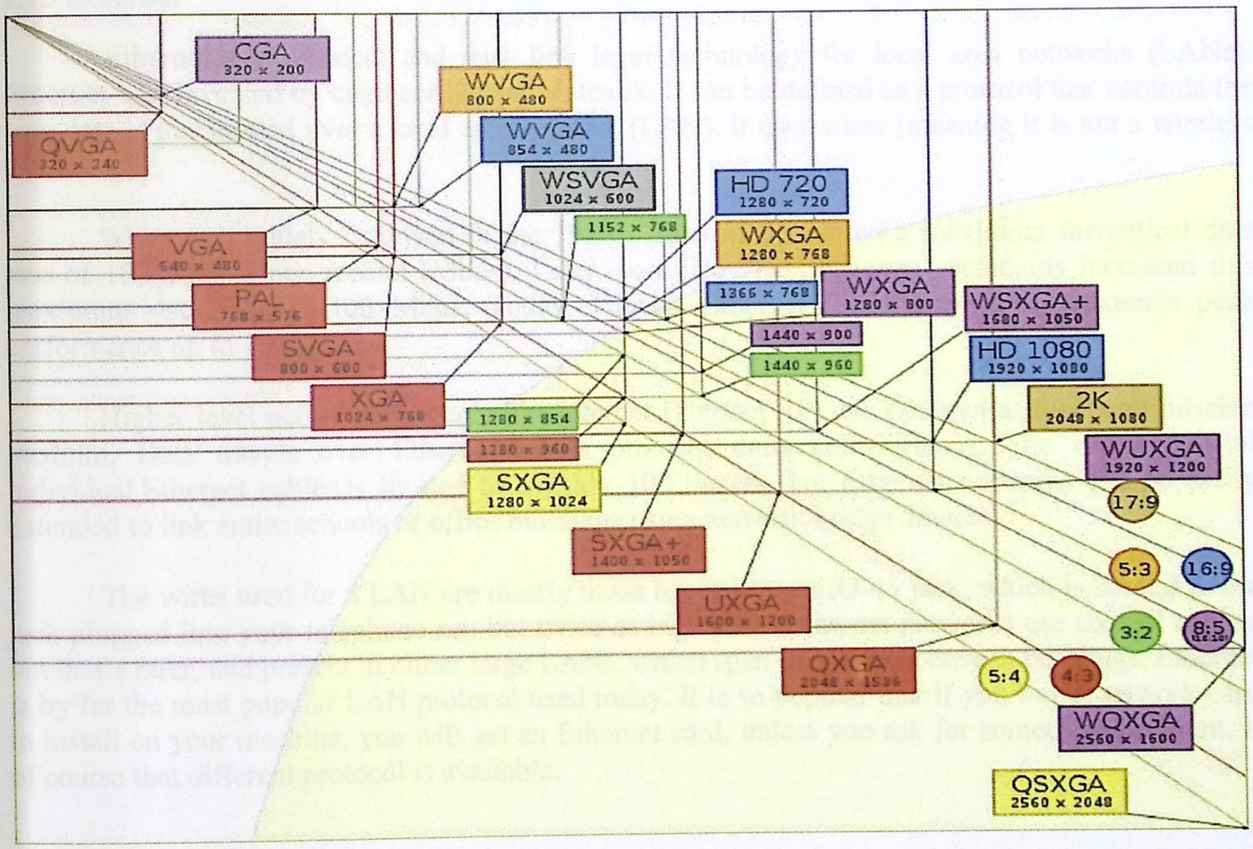


Fig 2.16: Resolution – The number of pixels present in the images of the video.[16]

B. Media of Transmission: deciding what codec you want to use depends on your means of transmitting the video so others can view it. This could be an external hard drive, jump drive, via email or uploading to a social media website/blog.

C. Choosing codec:

1) Check the requirements from the internet sites you wish to upload your video file. (Much of the internet supports a MP4 container with a H.264 codec)

2) Is your video meant for mobile applications? If so, select a codec that supports all mobile devices (ex. Apple does not support flash videos).

3) Is your video going to be embedded in another application? What other applications will typically do is take the video and apply their own codec to the video so that it will be playable within their application but that does not mean the application will understand all types of codec.

2.13 Ethernet

Ethernet is a physical and data link layer technology for local area networks (LANs). Ethernet was invented by engineer Robert Metcalfe. It can be defined as a protocol that controls the way data is transmitted over a local area network (LAN). It uses wires (meaning it is not a wireless technology).

When first widely deployed in the 1980s, Ethernet supported a maximum theoretical data rate of 10 megabits per second (Mbps). Later, so-called "Fast Ethernet" standards increased this maximum data rate to 100 Mbps. Today, Gigabit Ethernet technology further extends peak performance up to 1000 Mbps.

Higher level network protocols like Internet Protocol (IP) use Ethernet as their transmission medium. Data travels over Ethernet inside protocol units called frames. The run length of individual Ethernet cables is limited to roughly 100 meters, but Ethernet networks can be easily extended to link entire schools or office buildings using network bridge devices.

The wires used for a LAN are mostly those headed by an RJ-45 jack, which is similar to the jack plugged into your telephone set, but twice as big. Some Ethernet networks use coaxial cables, but that's rarer, and present in rather large LANs, which span over areas between buildings. Ethernet is by far the most popular LAN protocol used today. It is so popular that if you buy a network card to install on your machine, you will get an Ethernet card, unless you ask for something different, if of course that different protocol is available.

2.13.1 Ethernet Port

An Ethernet port is an opening on computer network equipment that Ethernet cables plug into. These ports are alternatively called jacks or sockets. Ethernet ports accept cables with RJ 45 connectors.

- **Ethernet Ports on Computers:**

Most computers include one built-in Ethernet port for connecting the device to a wired network. (A notable exception, the Mac Book Air, does not but allows connecting an Ethernet dongle to its USB port.) A computer's Ethernet port is connected to its internal Ethernet network adapter.

- **Ethernet Ports on Routers:**

All popular broadband routers feature Ethernet ports. An uplink port (also called WAN port) is a special Ethernet jack on routers used specifically for connecting to a broadband modem. Wireless routers include a WAN port and typically four additional Ethernet ports for wired connections.

- **Ethernet Ports on Consumer Electronics:**

Many other types of consumer gadgets now also include Ethernet ports for home networking. Examples include game consoles, digital video recorders and even some newer televisions.

Chapter Three

Conceptual Design

Chapter contents:

3.1 Overview

In this chapter, we present the system design concepts for both hardware and software including system operation, system entities, and detailed system diagram.

3.2 System Block Diagram

The general block diagram is shown in figure 3.1.The audio and video signals are firstly fed to the transmission system.

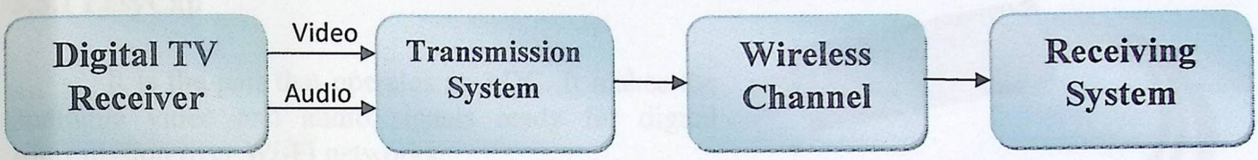


Figure 3.1: General block diagram

The transmission system is shown in figure 3.2.The video and audio signals are transferred to serial data and fed to the microcontroller. The main transmission unit is the Wi-Fi transceiver.

Figure 3.2: Transmission System block diagram

The receiving system is illustrated in figure 3.3 .It is mainly comprised of Wi-Fi adapter and a software for displaying the video information.

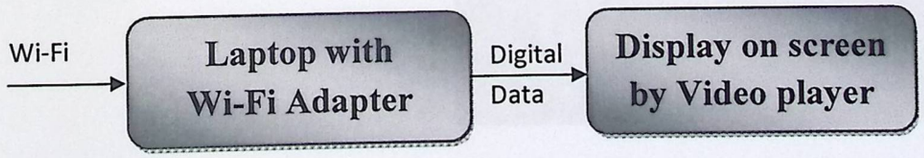


Figure 3.3: Receiving System block diagram

3.2.1 The challenge

A challenging point is the interfacing of EasyCap with the micro-processing unit. Normally, this device is connected to the computer and it needs a special driver with software to read the digital data from the USB port and display the video on the PC screen. However, in this project, it is required to receive the data from the EasyCap at the micro-processing unit and then deliver it to a distant PC or smart phone over Wi-Fi network.

3.3 System Entities

3.3.1 EasyCap

It is the part that operates as ADC. It makes the analogue video and audio signals ready for digital transmission over Wi-Fi networks.

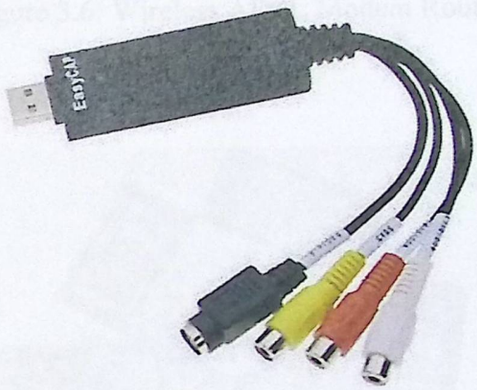


Figure 3.4: EasyCap Audio Video device

3.3.2 GESBC-9302E

It is a micro processing unit on a single integrated circuit containing a processor core, memory, and programmable input / output peripherals. It performs USB interfacing to make the data ready for transmission over Ethernet.

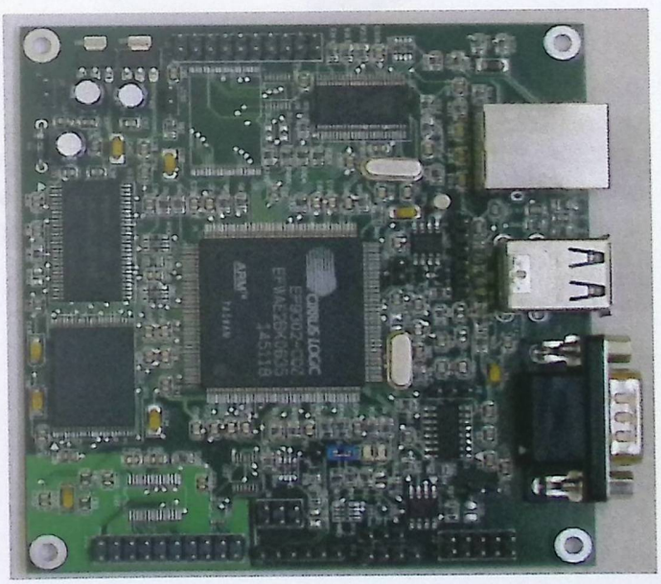


Figure 3.5: GESBC-9302E

3.3.3 Wireless ADSL Modem Router

It is used to transmit the data wirelessly over the Wi-Fi network. This will make us able to receive the transmitter video data, and also surf the internet at the same time.

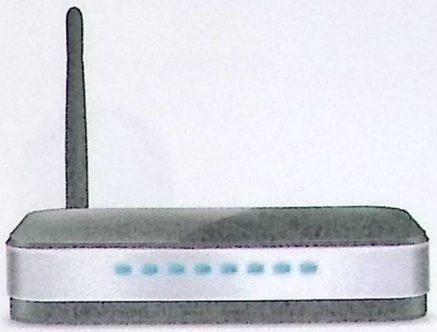


Figure 3.6: Wireless ADSL Modem Router

3.3.4 Receiving Devices

Receiving devices are devices that have build-in support for Wi-Fi; a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections.

With the advancement of technology, many of communication devices or gaming consoles now have built in wireless capabilities, all it takes is just some clicks and you will be connected to the Internet.

Figure 3.7: Wi-Fi Enabled Devices

It is so convenient that now you don't have to be constrained at home just to use the Internet. As long as you are equipped with a Wi-Fi enabled device such as a personal computer, video game console, mobile phone, MP3 player or personal digital assistant, you can connect to Internet anywhere as long as it is a Wi-Fi hotspot.

3.4 Detailed Diagram

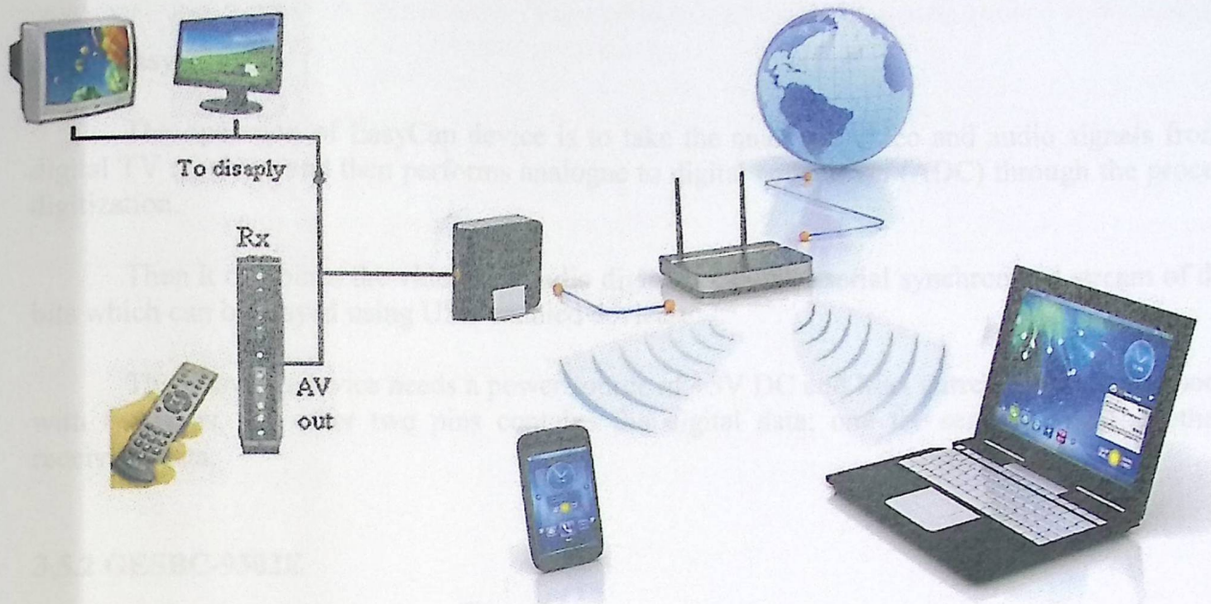


Figure3.8: Detailed diagram

The diagram of figure 3.8 shows the system in a detailed view. Starting from the digital TV receiver, the signal is driven to the TV to display the received satellite data. The signal at the same time is taken to be as input to the designed device in analogue form.

The signal then derived to the EasyCap device to make the process of conversion of the analogue A/V signals to digital stream of bits which is transferred serially to the USB port. The serial data should be made ready to be sent through a Wi-Fi transmitter. This is done by using the micro-processing unit to perform preparing and delivery of digital data.

After that, the signal is transferred wirelessly using Wi-Fi transmitter, access point or wireless router to the other side of transmission; the receiving devices. At this side, data is received at every Wi-Fi enabled device. This allows watching, editing, and storing this video data for future usage.

The receiving devices should have the suitable programs to display the video, or store and edit. These programs are unique for the different devices according to the type and operating system of that device.

3.5 System Operation

In this section, more information about the system operation is provided.

3.5.1 EasyCap

The operation of EasyCap device is to take the analogue video and audio signals from the digital TV receiver, and then performs analogue to digital conversion (ADC) through the process of digitization.

Then it combines the video and audio digital data into a serial synchronized stream of digital bits which can be played using USB enabled devices.

The EasyCap device needs a power source of +5V DC and Max current of 500mA associated with two pins, the other two pins contains the digital data; one for sending and the other for receiving data.

3.5.2 GESBC-9302E

The GESBC-9302E is a micro-processing unit that makes processing operations on the incoming digital serial data to interface between the USB dongle and the Wi-Fi router, performing serial interface to make the data ready for sending through Wi-Fi network.

3.5.3 Wireless ADSL Modem Router

This device provides an internet gateway and the communication between the designed system and the other receiving devices i.e. Laptops, Smart Phones, etc…

3.5.4 Receiving Devices

After the data is streamed through the Wi-Fi device, the data will be transferred to the receiving device which is Wi-Fi enabled. Those receiving devices are used to read the data and do some operations to play this data as a live video; this can be done after connection with the Wi-Fi transceiver with a prober signal quality.

It is a must for those receiving devices to have the needed software to be able to communicate and read the data to make it ready for human use. This can make it possible to play, edit, or even store the received data.

3.6 Software Design

The software design contains brief description about the programs and programming languages that are used in this project. Programming is an essential step to control and enabling the used devices and equipments to do their job correctly.

Software design divided into three categories depending on the order and place of the hardware that needs to program. Starting with the programming the GESBC-9302E to interface the EasyCap with microprocessor. Then we need to program the micro-processing unit to enable it for taking the data from EasyCap and deliver it to Ethernet adapter. Finally, at receiving devices it should be software or an application to handle the received data over the Wi-Fi network and displaying the video on the screen.

3.6.1 Interface of EasyCap to GESBC-9302E

In this part we describe three different interfaces for communication implemented in the GESBC-9302E module. One of the three interfaces has to be chosen.

3.6.1.1 UART

- Classical RS232 interface (after voltage conversion) with full 9-pin serial port.
- For ordinary communication we need necessarily just 5 pins – RxD, TxD, GND (RTS and CTS have to be pulled down to the ground through resistors).
- LVTTL (3,3 V) voltage levels, but TTL (5 V) tolerant.
- Baud rate is settable from 300 till 1 MBd (after reset is always set to 9600 Bd).
- On the Fig 3.9 is shown UART reception waveform (the upper part) and the transmission waveform.

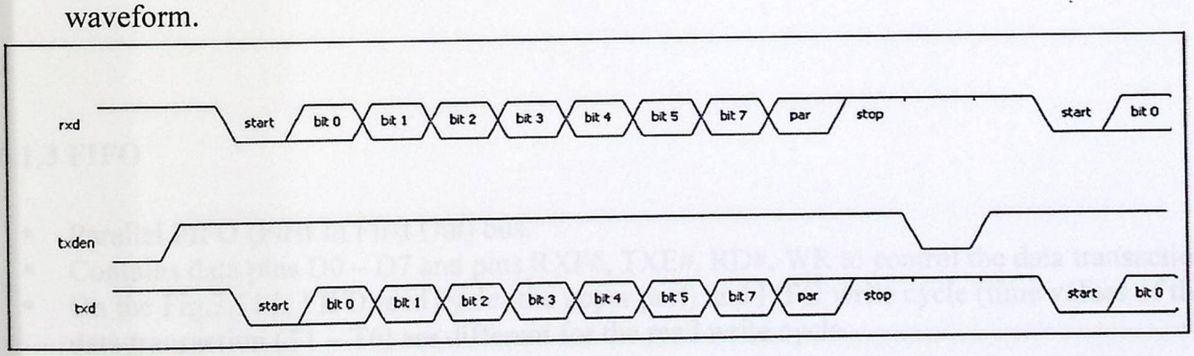


Figure 3.9: Sequences on the pins during reception and transmission of one byte

3.6.1.2 SPI

- Uses four pins four pins SCLK (Serial Clock), SDI (Serial Data Input), SDO (Serial Data Output), CS (Chip Select).
- Can operate with the clock (SCLK) frequency up to 12 MHz.
- Differs from other implementations, because it uses 13clock sequence to transfer a single byte of data.
- On the Fig 3.10 is demonstrated slave data read cycle (the upper part) and slave data write cycle.

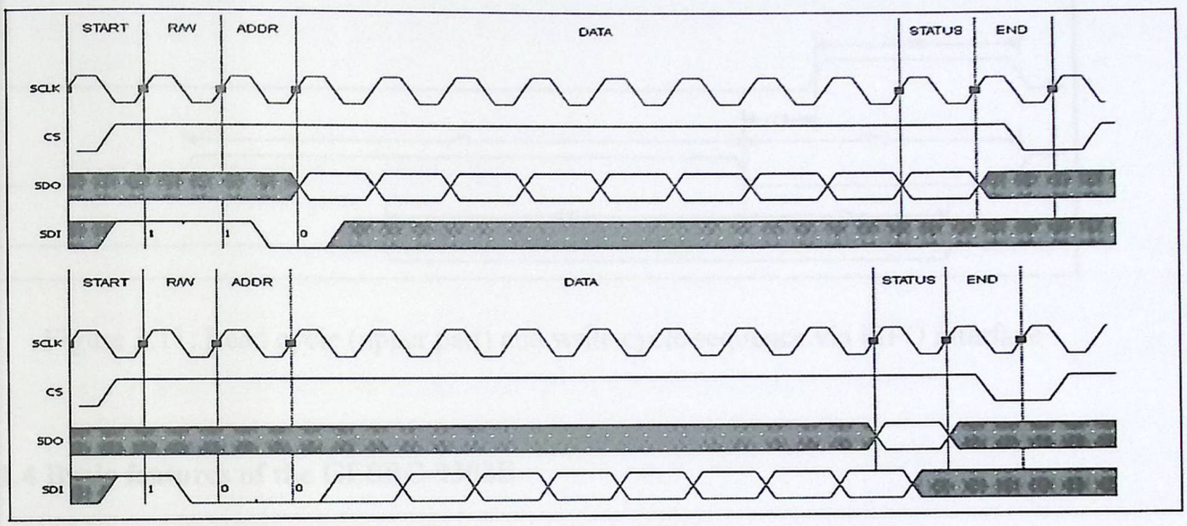


Figure 3.10: Slave data read cycle (upper part) and slave data write sequence on the SPI bus

3.6.1.3 FIFO

- Parallel FIFO (First In First Out) bus.
- Contains data pins D0 – D7 and pins RXF#, TXE#, RD#, WR to control the data transaction.
- On the Fig.3.11 is FIFO read cycle (the upper part) and FIFO write cycle (time values of the data transaction (T1 – T6) are different for the read write cycle.

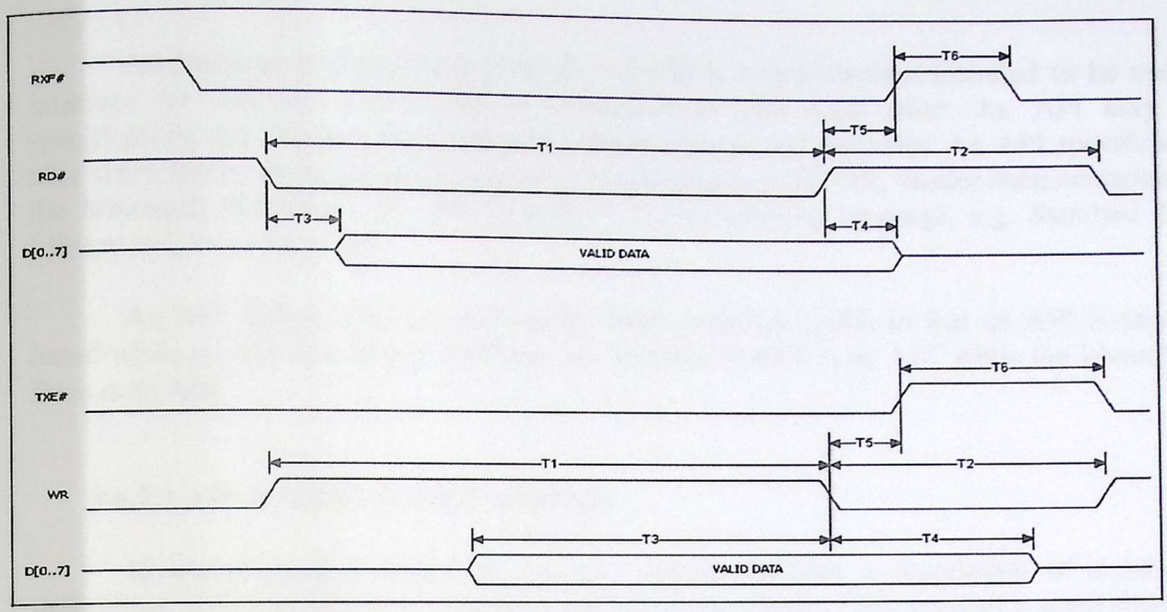


Figure 3.11: Read cycle (upper part) and write cycle sequence via FIFO interface

3.6.1.4 Basic features of the GESBC-9302E

The GESBC-9302E micro-processing board uses the Cirrus Logic EP9302 as the core processor on this development board. The top-level features of EP9302 processor are the following:

- ARM920T RISC Core Processor.
- 200 MHz / 200 MIPS Performance.
- 16 Kbyte Instruction Cache.
- 16 Kbyte Data Cache.
- 100 MHz System Bus.
- 16 bit SDRAM Interface (Up To 4 Banks).
- 16 bit SRAM / FLASH / ROM Interface.
- Serial EEPROM Interface./
- 10 / 100 Mbps Ethernet MAC.
- Two UARTs.
- Two-port USB Host.
- SPI Port

3.6.2 Programming Module

An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. An API specification can take many forms, including an International Standard such as POSIX, vendor documentation such as the Microsoft Windows API, the libraries of a programming language, e.g. Standard Template Library in C++ or Java API.

An API differs from an application binary interface (ABI) in that an API is source code based while an ABI is a binary interface. For instance POSIX is an API, while the Linux Standard Base is an ABI.

3.6.2.1 API in object-oriented languages

In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods (or more generally by all its public components hence all public methods, but also possibly including any internal entity made public, like fields, constants, nested objects, enemas, etc.).

The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class interface). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions.

More generally, one can see the API as the collection of all the kinds of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a technical detail of how the behavior is implemented.

For instance: a class representing a Stack can simply expose publicly two methods push() (to add a new item to the stack, and pop() (to extract the last item, ideally placed on top of the stack.

In this case the API can be interpreted as the two methods pop() and push(), or, more generally, as the idea that one can use an item of type Stack that implements the behavior of a stack: a pile exposing its top to add/remove elements. The second interpretation appears more appropriate in the spirit of object orientation.

This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java language and Lisp (programming language) API include the interface Serializable, which is a marker interface that requires that each class that implements it should behave in a serialized fashion. This does not require to have any public method, but rather requires that any class that implements it to have a representation that can be saved (serialized) at any time (this is typically true for any class containing simple data and no link to external resources, like an open connection to a file, a remote system, or an external device).

Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarily determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation.

In this sense, in object-oriented languages, the API defines a set of object behaviors, possibly mediated by a set of class methods.

In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API in JavaDoc notation.

The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use.

3.6.3 Data processing and preparing for Ethernet interface:

At the micro-processing unit commands in C++ language and libraries should be installed on it to enable handling the data format that comes from USB port and rearranging it in packets for transferring it over the Ethernet port.

The first step is to take the serial data from EasyCap and process it to enable streaming it. This was done by studying the drivers of EasyCap and analysis them to build a library contains C++ commands for controlling, synchronization and session initiating. The full code is attached in the appendix.

The second step is to deliver the digital data from the board to the Ethernet line. The data arranged in packets to transfer it over the network. This job is done by build-in Ethernet adapter, but we should define important parameters like data transmitting port and method to generate the IP address.

3.6.4 Video Displaying Module

To receive the video and watch it, we need a video player to do the job. The software that will be used in this project is a video player with advanced options. It is called "VLC media player".

The software should be configured to define the source of streaming, this can be done by entering the IP address of the device and its port number as we program it previously, see the following figure.

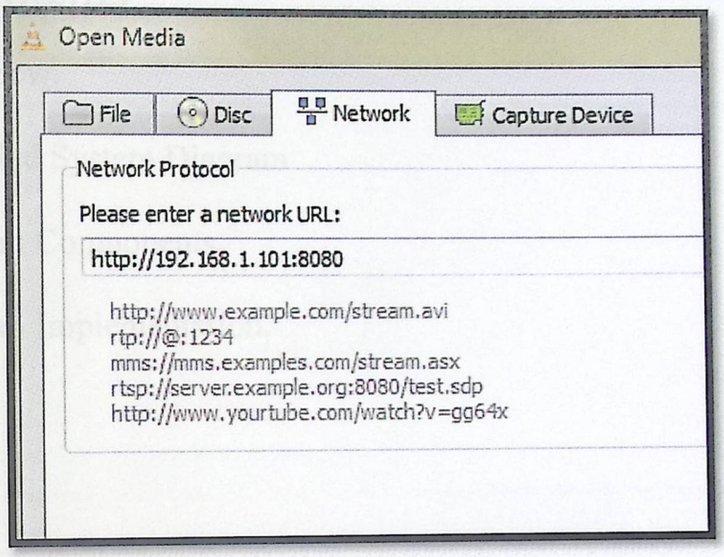


Figure 3.12: VLC Video Player advanced options

Chapter four

Detailed System Design

Chapter contents

4.1 Overview

Through the last three chapters we study and create a complete image about the system which will be designed.

Firstly, we create the general idea of the system, by searching and collecting information about TV systems, available technologies and hardware to be used in the project .Then we study each available part in details.

In this chapter we will make a complete design of the system by making connection between each part of the system to get the required system design, and show pin's connection in each part with the required program and software to communicate.

4.2 Detailed System Diagram

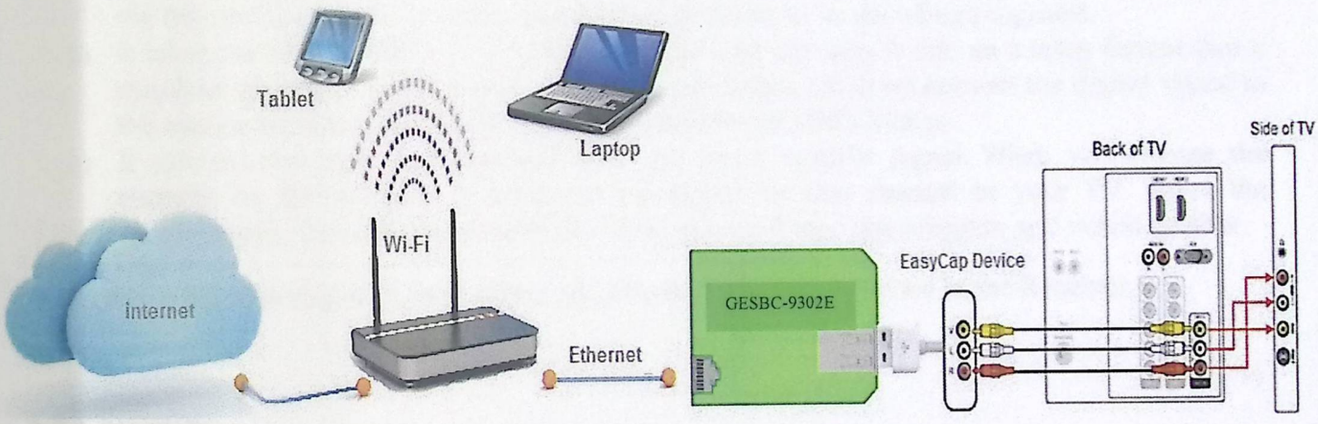


Figure 4.1: Detailed System Diagram

In figure 4.1 we see the intended system in detailed view, which specifies the system connection between system components.

The signal is derived to the TV to display the received satellite data and at the same time the signal is taken to be as input to the designed device in analogue form.

The signal then derived to the EasyCap device to make the process of conversion of the analogue A/V signals to the USB port. The serial data should be made ready to be sent through a Wi-Fi router.

After that, the signal is transferred wirelessly using Wi-Fi router to the other side of transmission; the receiving devices. At this side, data is received at every Wi-Fi enabled device. This allows watching, editing, and storing this video data for future usage.

The receiving devices should have the suitable programs to display the video, or store and edit. These programs are unique for the different devices according to the type and operating system of that device.

4.3 System Components

4.3.1 Digital Satellite Receiver

The end component in the entire satellite TV system is the **receiver**. The receiver has four essential jobs:

Figure 4.2: Digital Satellite Receiver

1) It decrypts the encrypted signal: In order to unlock the signal, the receiver needs the proper decoder chip for that programming package. The provider can communicate with the chip, via the satellite signal, to make necessary adjustments to its decoding programs.
2) It takes the digital MPEG-2 or MPEG-4 signal and converts it into an analog format that a standard television can recognize: In the United States, receivers convert the digital signal to the analog National Television Systems Committee (NTSC) format.
3) It extracts the individual channels from the larger satellite signal: When you change the channel on the receiver, it sends just the signal for that channel to your TV. Since the receiver spits out only one channel at a time, you can't tape one program and watch another.

In our system the signal is fed to the system through the A/V interface in the Receiver.

4.3.2 EasyCap

The EasyCAP USB 2.0 Video Adapter with Audio, it can capture High-quality video and audio file direct by USB 2.0 interface without sound card. However, the external power is unnecessary. This device is responsible for converting the analog Audio/Video signal to digital form to be ready for transmission over USB port.

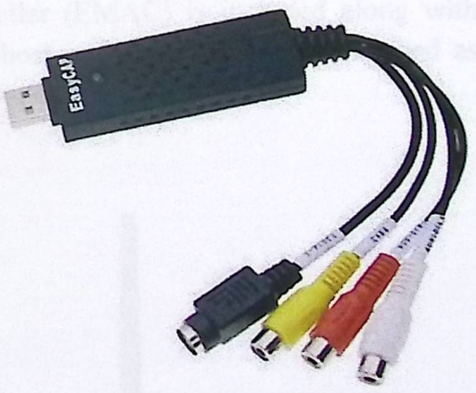


Figure 4.3: EasyCap Audio Video device

4.3.3 GESBC-9302E

The GESBC-9302E is a low cost compact sized micro-processing unit based on Cirrus Logic EP9302 processor. With a large peripheral set targeted to a variety of applications, the GESBC-9302E is well suited for industrial controls, digital media servers, audio jukeboxes, thin clients, set-top boxes, point-of-sale terminals, biometric security systems. The heart of the GESBC-9302E is the EP9302 which is the one in a series of ARM920T based processors. The ARM920T microprocessor core with separate 16 Kbyte 64-way set-associative instruction and data caches is augmented by the coprocessor.

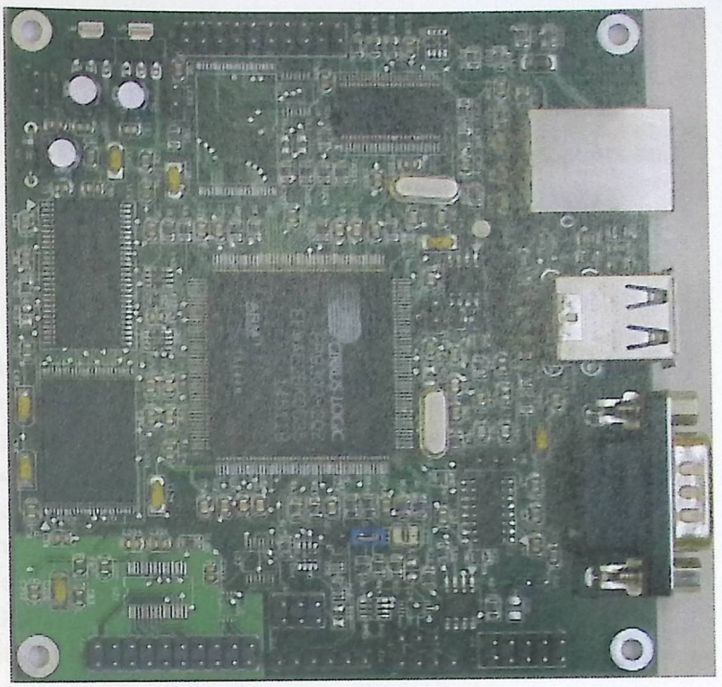


Figure 4.4: GESBC-9302E Micro-processing unit

The EP9302 is a high-performance, low-power RISC-based device built around a single ARM920T microprocessor core. The ARM920T on the EP9302 functions with a maximum operating clock rate of 200MHz and a power usage between 100mW and 750mW (dependent upon clock speed). The ARM core operates from a 1.8V supply while the I/O operates at 3.3V. A high performance 1/10/100 Mbps Ethernet Media Access Controller (EMAC) is included along with external interfaces to SPI and I2S Audio. A two-port USB host and two UARTs are included as well.

4.3.4 Wireless ADSL Modem Router

This device is commonly used to provide access to the Internet or a computer network. It does not require a wired link, as the connection is made wirelessly, via radio waves. It can function in a wired LAN (local area network), in a wireless-only LAN (WLAN), or in a mixed wired/wireless network, depending on the manufacturer and model.

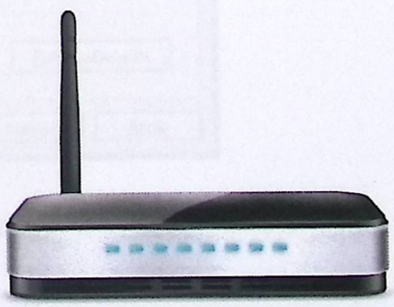


Figure 4.5: Wireless ADSL Modem Router

In our Project it is used to transmit the data wirelessly over the Wi-Fi network. This will make us able to receive the transmitted video data, and also surf the internet at the same time.

4.4 System Implementaion

Here we describe the GESBC-9302E working environment and familiarizes its components and functionality.

4.4.1 Assembly and Connections

In order to use the GESBC-9302E the user must first assemble and connect the peripherals to the GESBC-9302E, as described in the following procedure.
1) Place the GESBC-9302E on a static free surface.
2) Make sure all of the jumpers are in the factory default position.
3) Connect 5V regulated power supply to the board.
4) Connect null modem serial cable between GESBC-9302E UART 1 and PC/terminal serial port.
5) Launch a terminal emulator, such as HyperTerminal, or minicom, on the PC configured to connect to the serial port of the GESBC-9302E. Configure the serial port with the following parameters: 57600 bits per second, 8 data bits, no parity, 1 stop bit, no flow control.

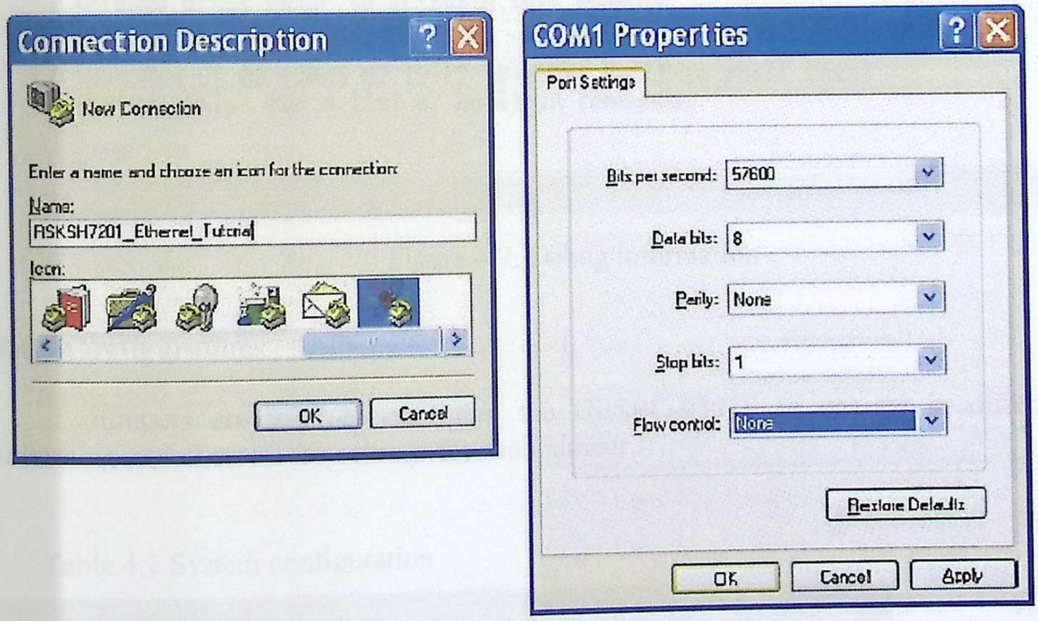


Fig 4.6 Hyper Terminal program settings

4.4.2 Operation

The startup procedure for the GESBC-9302E is straightforward. First, the connection of the power harness is required. Second, the null modem serial interface cable must be connected to the UART1 connector. It is recommended that all other cables be tested to determine they are properly seated. A few seconds after applying power to the GESBC-9302E, debug information will be displayed on the terminal program. The following figure shows what this should look like.

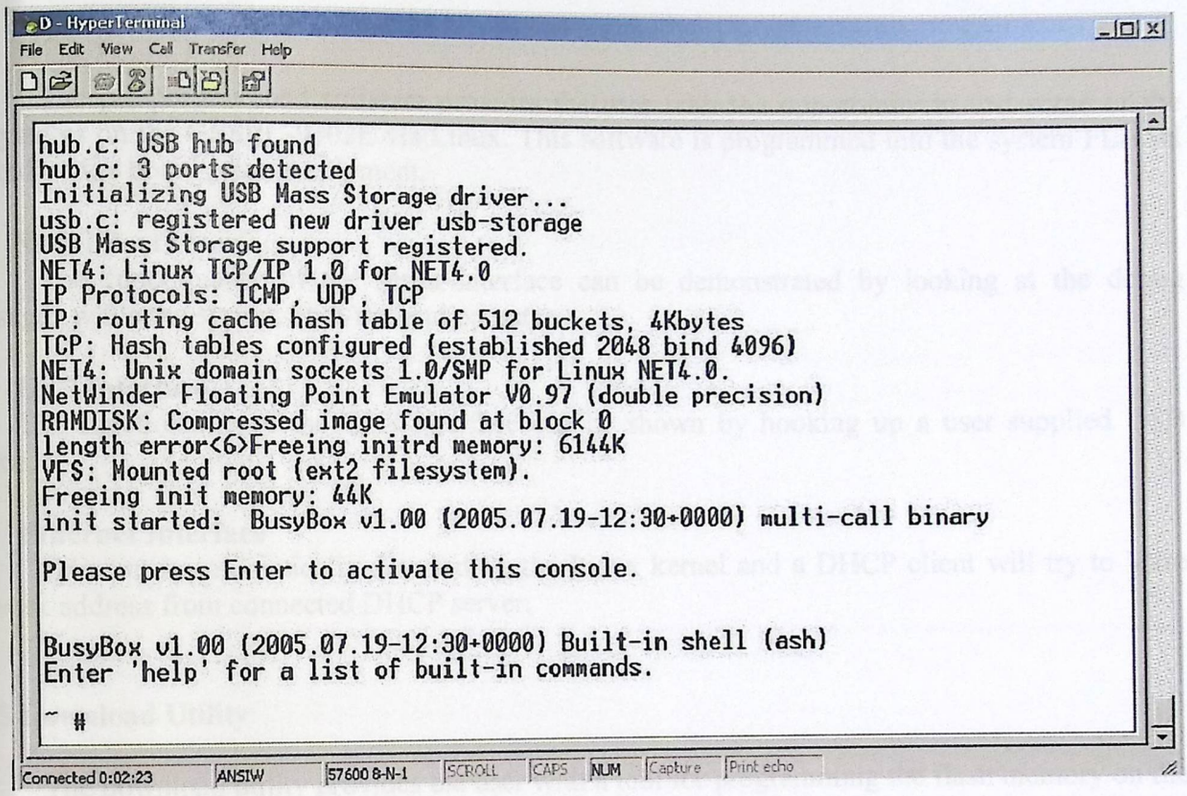


Figure 4.7: Debug information

4.4.3 Configurations

Jumpers are used to configure the GESBC-9302E to operate in different mode. The following table lists all the settings for each jumper.

Table 4.1 System configuration

Jumper	Description
2	Boot mode: connect pin 1 and 2 - serial boot connect pin 2 and 3 - flash boot
SW1	Reset switch header
LED3	Power indicator LED header

4.4.4 Software Description

This provides information regarding the software that is shipped with the GESBC-9302E Board. The software included with the board is Linux with a few test applications and network utilities. The Linux software provides the user with the ability to test some of the subsystems on the GESBC-9302E board. The download utility provides a means to program a binary image into the flash memory on the GESBC-9302E.

- **GESBC-9302E Linux Code**

The pre-programmed software provides the user with the opportunity to test some of the subsystems on the GESBC-9302E via Linux. This software is programmed into the system FLASH located on the board prior to shipment.

- **Serial Port Interface**

The functionality of the serial interface can be demonstrated by looking at the debug messages while the system boot-ups and operates.

- **USB Interface**

The functionality of the USB interface can be shown by hooking up a user supplied USB device.

- **Ethernet Interface**

Ethernet is automatically detected by the Linux kernel and a DHCP client will try to lease network address from connected DHCP server.

4.4.5 Download Utility

The download utility provides the user with a tool for programming the flash memory on the GESBC-9302E with a binary image. The following procedure will allow in-circuit programming of the flash memory via the EP9302 processor:

1) Power the board off.
2) Connect null-modem serial cable to UART1.
3) Set jumper 2 to connect pin 1 and 2 (JP2 factory default is pin 2 and 3).
4) Run download utility (assuming download.exe located in same directory as binary image) download binary_image_filename.bin
5) "Waiting for board to wake up…" message is displayed.
6) Power the board on.
7) Messages displayed regarding erasing, then programming the flash.
8) "Successfully programmed binary_image_filename.bin" message displayed upon programming completion.
9) Power the board off.
10) Install jumper on pin 2 and 3 of JP2.
11) Power the board on.

4.4.6 Loading Linux Kernel and root File System

The Redboot boot-loader provides two ways to load Linux kernel and file system into FLASH memory, by serial connection. After power on the GESBC-9302E board, the following message should be shown on the terminal console on the host PC connected to the GESBC-9302E board.

```
+FLASH configuration checksum error or invalid key
EP93xx - no EEPROM, static ESA, or RedBoot config option.
No network interfaces found
RedBoot(tm) bootstrap and debug environment [ROMRAM]
Non-certified release, version v2_0 - built 07:39:29, Dec
18 2004
Platform: Cirrus Logic EDB9301 Board (ARM920T) Rev A
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.
RAM: 0x00000000-0x04000000 available
FLASH: 0x60000000 - 0x60400000, 16 blocks of 0x00040000
bytes each.
RedBoot>
```

It's possible to use bootp of Redboot to acquire network address automatically. For situation it is not available, the following procedure can be used to configure a static IP address for the SBC.

```
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis load ramdisk
>> fis load zImage
>> exec -r 0x800000 -s 0x300000
>>
Boot script timeout (1000ms resolution): 1
Use BOOTP for network configuration: false
Gateway IP address:
Local IP address: 192.168.1.2
Local IP address mask: 0.0.255.255
Default server IP address:
DNS server IP address:
Set eth0 network hardware address [MAC]: true
eth0 network hardware address [MAC]: 0x00:0x00:0x00:0x00:0x80:0x21
GDB connection port: 21211
Force console for special debug messages: false
Network debug at boot time: false
Update RedBoot non-volatile configuration - continue (y/n)? y
... Erase from 0x60780000-0x60781000: .
... Program from 0x03fbe000-0x03fbf000 at 0x60780000: .
RedBoot>
```

The Redboot FLASH file system must be initialized in order to store data in the FLASH file system. The following procedure is used to initialize the Redboot FIS.

```
RedBoot> fis init
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
```

```
Warning: device contents not erased, some blocks may not be usable
... Erase from 0x607c0000-0x60800000: .
... Program from 0x03fbf000-0x03fff000 at 0x607c0000: .
RedBoot>
```

4.4.6.1 Load Root File System

The default configuration of GESBC-9302E is using part of SDRAM as RAM disk for Linux root file system. The RAM disk image must be stored in the on-board FLASH memory and loaded by Redboot for the Linux kernel. The image must be loaded into dynamic memory before it can be stored in the on board FLASH memory. To load the ramdisk file to SDRAM, enter the following commands at the terminal console,

```
load -v -r -b 0x800000 -h tftp_host_ip ramdisk_file_name
```

where

-v : verbose

-r : binary format

-b : base address in memory

for serial port download, the command is,

```
load -v -r -b 0x800000 -m ymodem
```

Immediately after entered the above serial download command, start Y-Modem transfer on the terminal program, the download process should start. The above commands will load ramdisk file into on board SDRAM. To store the image into non-volatile FLASH memory, use the following command,

```
fis create -b 0x800000 -l <ramdisk_file_length> ramdisk_file_name
```

where

-b : is the memory base address

-l : is the ramdisk size. It can be calculated by subtracting the end address from the base address from the Redboot response when loading the ramdisk file. The ramdisk_file_name can be any arbitrary name. To verify the image has been stored correctly in the FLASH memory, use the following command,

```
fis list
```

4.4.6.2 Load Linux Kernel

The kernel image must be loaded into dynamic memory before it can be stored in the onboard FLASH memory. To load Linux kernel, issue the following command at the terminal console connected to the GESBC-9302E board, The host computer should have *tfpt* server running and Linux kernel and file system file stored in the *tftp* root directory.

```
load -v -r -b 0x80000 -h host_ip_address kernel_image_name
```

where

-v : verbose

-r : binary format

-b : base address in memory

The above command will load Linux kernel image file into on board SDRAM. To store the image into non-volatile FLASH memory, use the following command,

```
fis create -b 0x80000 -l <kernel_image_length> kernel_image_name
```

where

-b : is the memory base address

-l : is the kernel image size. It can be calculated by subtracting the end address from the base address from the Redboot response when loading the kernel image file. The kernel_image_name can be any arbitrary name. To verify the image has been stored correctly in the FLASH memory, use the following

command, `fis list` , multiple kernel images or root file systems can be stored in the on-board FLASH memory when memory space permits.

4.4.7 USB Configurations

The GESBC-9302E provides two USB host connections. The EP9302 USB host controller is configured for two root hub ports and features an integrated transceiver for each port. The EP9302 integrates two USB 2.0 Full Speed host ports. These ports are fully compliant to the OHCI USB 2.0 Full Speed specification (12 Mbps). The controller complies with the OHCI specification for USB Revision 1.1. The USB ports are brought out by a standard double deck USB type A connector.

Widespread adoption of Linux in embedded devices translates to a demand for a robust debugger and development tools that are delivered at the highest level of integration. Embedded Linux development tools include full, royalty free source code of the embedded Linux kernel, bundled with a state of the art debugger, cross tool chains, and productivity tools, all integrated in a modern easy to use graphical environment. Available on Linux hosts, Products are positioned to deliver complete debugging coverage of device drivers and applications utilizing threads and shared libraries, from a single debug connection over high speed Ethernet.

The Arriba embedded Linux distribution is guaranteed to be fully tested for compatibility with the Arriba source debugger and build environment. Together, they provide developers with a comprehensive environment to rapidly and reliably bring products based on embedded Linux to market.

Use following steps to use USB drive as root file system,
1) Press CTRL-C to stop any Redboot script when power up the board.
2) At Redboot command prompt, load kernel image from FLASH memory using following command, `fis load zImage` or load it from TFTP server using following command, `load -r -v -b 0x80000 -h <TFTP server IP address> <kernel image file name>`
3) Use the following command to boot system use USB drive as root file system.

```
a) 2.4.21 kernel with USB root file system patch
        exec -c "console=ttyAM0 root=/dev/sda1"

b) 2.6.x kernel
        exec -c "console=ttyAM0 root=/dev/sda1 rootdelay=15"
```

Depending on the USB drive and system configuration, the delay time can be adjusted to obtain best result. All GESBC-93xx boards come with pre-build kernel that has USB storage option enabled. If you build your own kernel please make sure the USB storage and SCSI are configured in order to use USB drive.

The driver of EasyCap device now can be downloaded as a code in .h file format to the unit to have the ability to read the net data of the device. This makes the EasyCap work properly so we can deal with the data it produce in the digital form.

4.4.8 Data Delivery inside GESBC-9302E

Configuration of the micro-processing unit GESBC-9302E and downloading libraries for EasyCap device on it should be done to complete the project. This will enable the micro-processing unit to recognize EasyCap device and the instructions for controlling data flow from it which leads to make the incoming serial data is meaningful for the GESBC-9302E. After the data received from serial USB port of the EasyCap device, the micro-processing unit delivers it to Ethernet adapter. The Ethernet adapter divides the data to small pieces and then arranges them in frames that contain the source and destination addresses. When the frames are ready, the adapter sends them to the network over Ethernet cable.

4.4.9 Ethernet Configurations

The GESBC-9302E is shipped with support for a complete Ethernet interface. The EP9302 contains a MAC subsystem that is compliant with the ISO/TEC 802.3 topology for a single shared medium with several stations. The Media Access Controller (MAC) within the EP9302 supports 1/10/100 Mbps transfer rates and interfaces to industry standard physical layer devices. The GESBC 9302E is shipped with the ICS1893AF 100Base-X / 10Base-T Transceiver device which, along with a RJ45 connector provides the physical layer interface.

4.4.10 Receiving Devices Configurations

4.4.10.1 Steps for connecting to Wi-Fi Network

It only takes a few steps to learn how you can connect your device to the Wireless Network.

Step 1: Turn on the WIFI on your device and browse for available wireless networks around you.

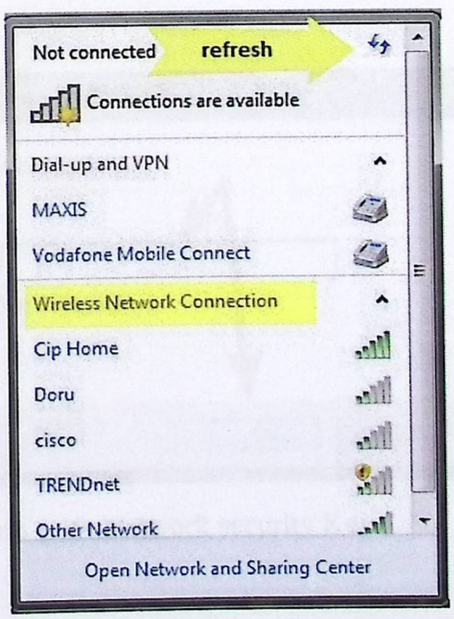


Figure 4.8: Networks available screenshot

A window with available network connections will open. As you can see from the screenshot below, the list is split by the type of available network connections. At the top you have dial-upand virtual private network (VPN) connections, while at the bottom you have a list with all the wireless networks which Windows 7 has detected. To refresh the list of available networks, click on the button highlighted in the screenshot in Figure 4.8.

You can scroll down through the list of available networks. If you leave your mouse cursor over a network for a second, you will see more details about it. Windows 7

will show the following: network name, signal strength, the type of wireless security used (if any) and its Service Set identifier (SSID).

Step 2: Identify the name of your wireless network (SSID) and click to connect to the wireless network.

Once you decided on which network to connect to, click on it. If you plan to use that network in the future, make sure you check the box that says *'Connect automatically'*. This way, when you start your laptop next time, in the same area, it will automatically connect to this wireless network without requesting any manual intervention. Next, click on the *Connect* button.

Figure 4.9: Network Identification

Step 3: Once you click Connect, you will usually be prompted for the password.

NOTE: Public networks usually do not require password.

You will be asked to enter the security key. Ask the administrator of the network for the wireless security key or, if you are in your own home network, take it from the control panel of your router. If you are in a public place, it is best to check the *'Hide characters* 'box so that other people don't see what you are typing. Then type the security key and click on *OK*.

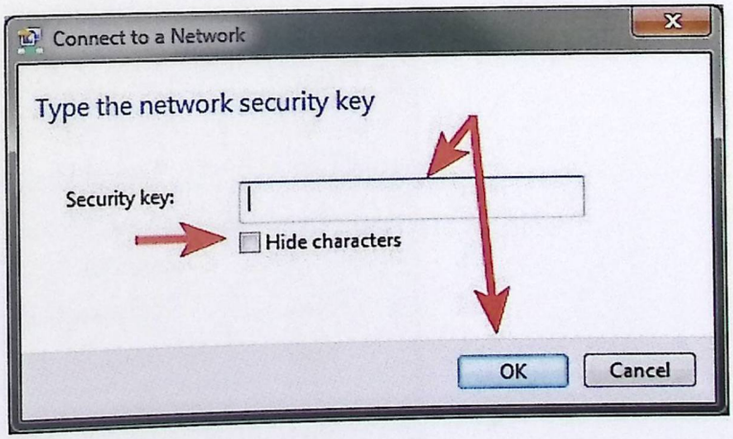


Figure 4.10: Network security Key

Step 4: Once you have successfully connected to a Wi-Fi network with your device, it will automatically connect to that network the next time you are in range.

Step 5: Before connecting to wireless network we must know:

If you are public Wi-Fi, connect to the wireless network with the strongest signal strength. The more green bars you see on the signal strength, the better connection speed you will get. If you are using your home wireless network, make sure your router is not positioned behind a thick wall or near microwave oven.

There are three basic types of encryption: WEP, WPA and WPA2. It is best to use WPA2 as it is the most secured encryption. A "**hidden**" wireless network is a network whose router has been configured not to broadcast the SSID.

4.4.10.2 Video Player Configurations

To display the channels and watching them you could use video player with advanced options related to the networks. In this project we use a free video player called "VLC media player".

After downloading the software freely from the internet and installing it on a computer or any Wi-Fi enabled devices; there are steps to configure the player before you can play and watch the channels:

1) Check the IP address of the designed device. This could be done by checking the routing table in your router which our designed device is connected to.

2) Open the player and then open the menu "media" on the top bar of player. Then click on "Open Network Stream" as shown in figure 4.8.

3) After that a window is appear asking you to enter the network URL. Enter the IP address for the designed device that you got before in step 1 with port number 21211 separated by colon.

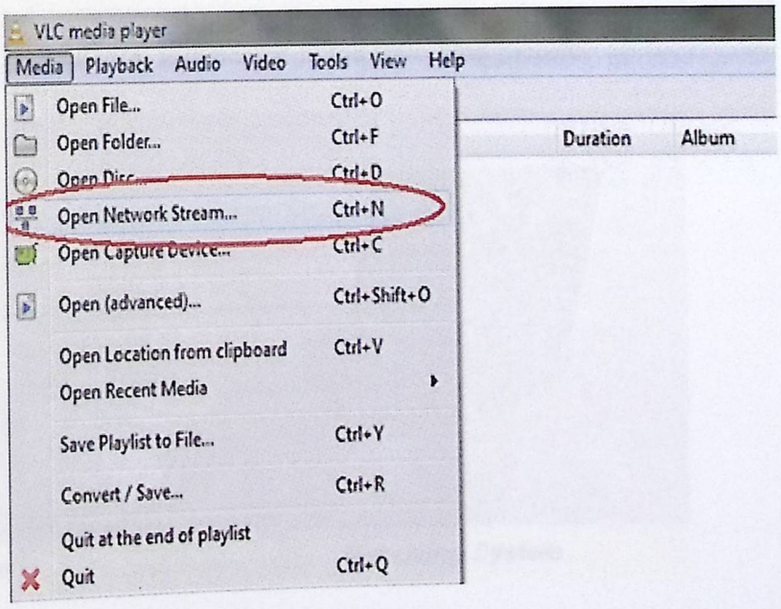


Figure 4.11: VLC Player "media" menu

4) Finally, click on play, wait a seconds and enjoy watching TV channels on your Wi-Fi enabled device or even any computer connected to the network.

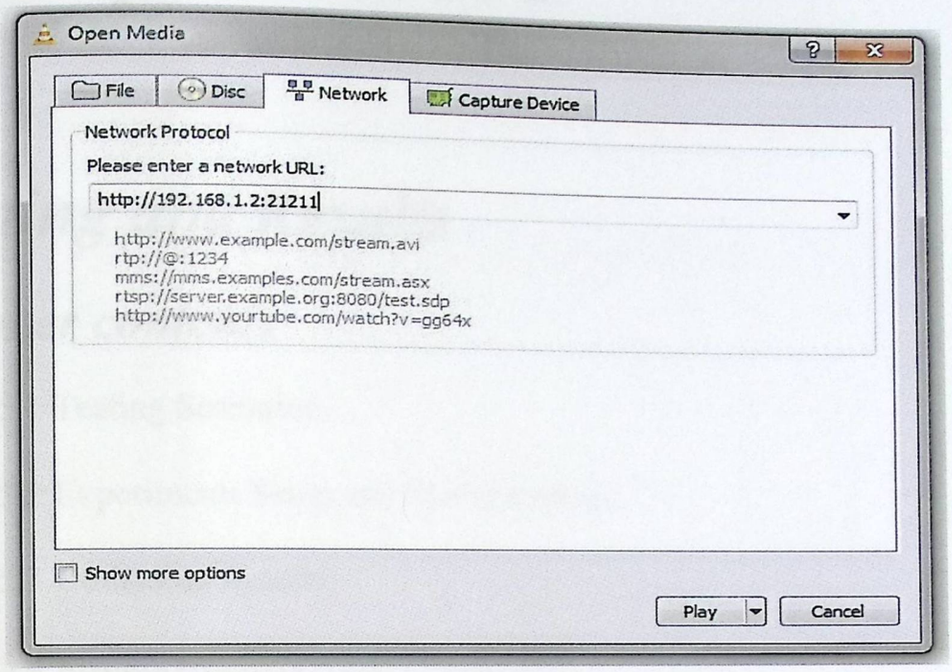


Figure 4.12: IP address and port configurations

4.4.11 Final Real Implementatiom

The system after the overall implemntation is depected in the following picture.

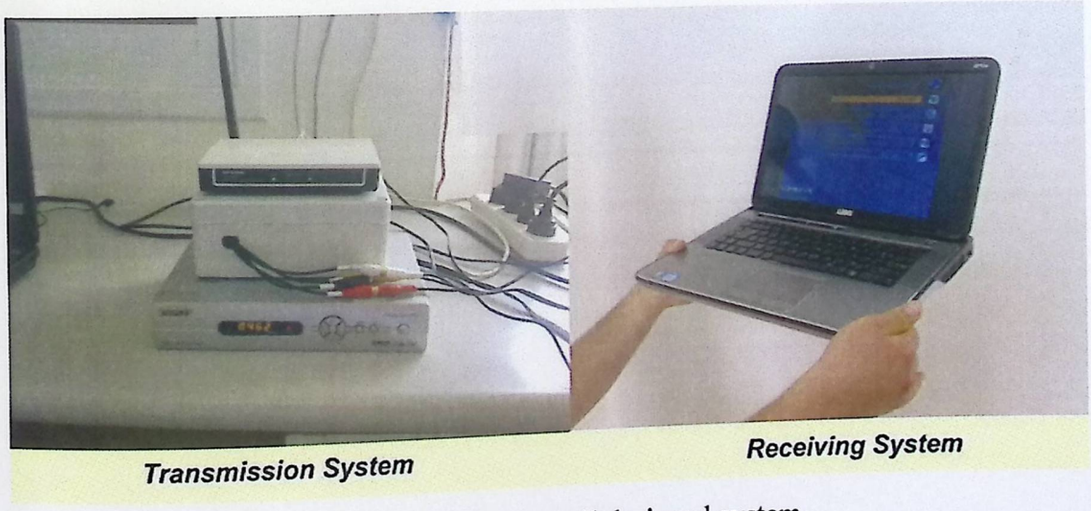


Figure 4.13: The overall designed system

Chapter Five

Testing and Results

Chapter contents

5.1 Testing Scenarios

There are many scenarios for testing the system. These scenarios are done according to different cases including number of served devices, quality of video, environment conditions and the distance between the transmitter and the receiver. So the testing scenarios will be as the following:

1) Increasing number of receiving devices while watching the quality and performance of the designed system.

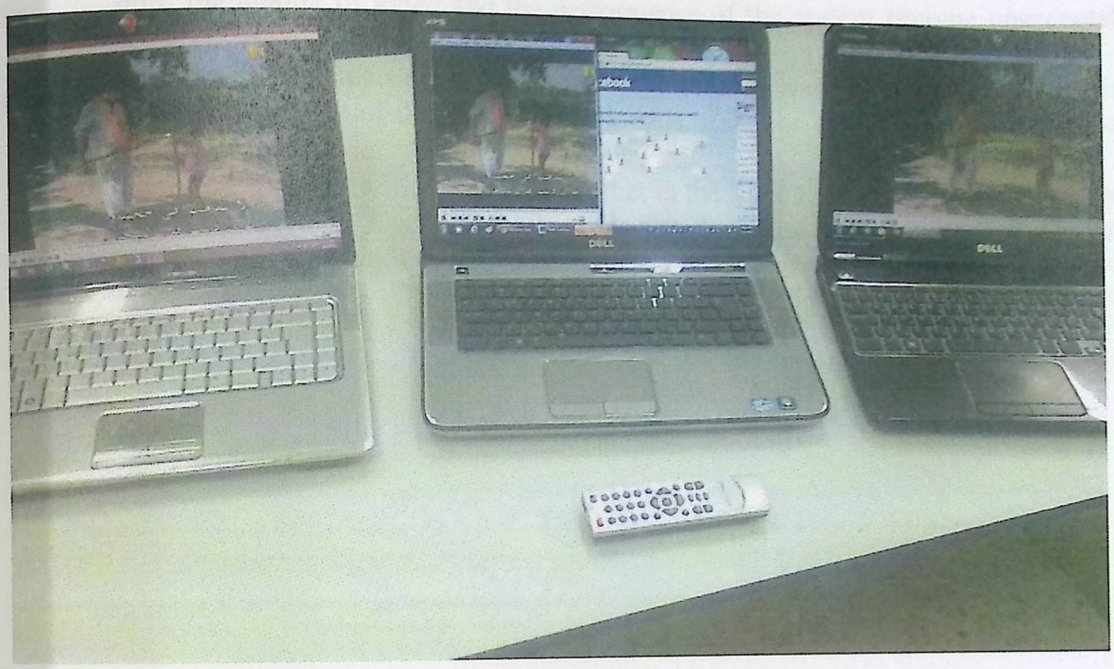


Figure 5.1: Multiple Receiving Devices

2) Changing the distance from access point or router and write down the results.
3) Making additional traffic on the network and observing the effect on the watching TV channels at the receiving devices.

5.2 Experiment Setup and Configurations

After powering up the designed device and starting watching TV channels on single receiving device we shall make some experiments and take the results. The experiments and their detailed procedures as the following:

1) While the first receiving device is playing the TV channels, play the channels on other device and notice the difference. Increase the number of receiving devices by one each time until the quality of video and the performance of the system become unacceptable. Then write down the results.
2) Play the TV channels near the transmitter, then start walk away with the receiving device until the coverage of the Wi-Fi network ends. Examine the effects of the distances on the system performance.
3) Start downloading a file or use the internet in the same time with watching TV channels on receiving devices. Use different speeds for download and watch what happen for the performance of the designed system and then take the results.

5.3 Collecting Results

Table 5.1 shows the results in first experiment by adding more received devices to watch TV channels on the same network.

5.3.1 System Performance Measures

The evaluation of system performance is based on the standards we define in the next points.

1) Very good is rated as (10) this is the ideal case of transmission, in this case the video signal is very clear and with no problems.
2) Good performance is rated as (7-9); the quality of this grade is shown in figure 5.2.

Figure 5.2: Good Video Quality

3) Acceptable performance is rated as (5-6); the quality of this grade is shown in figure 5.3.

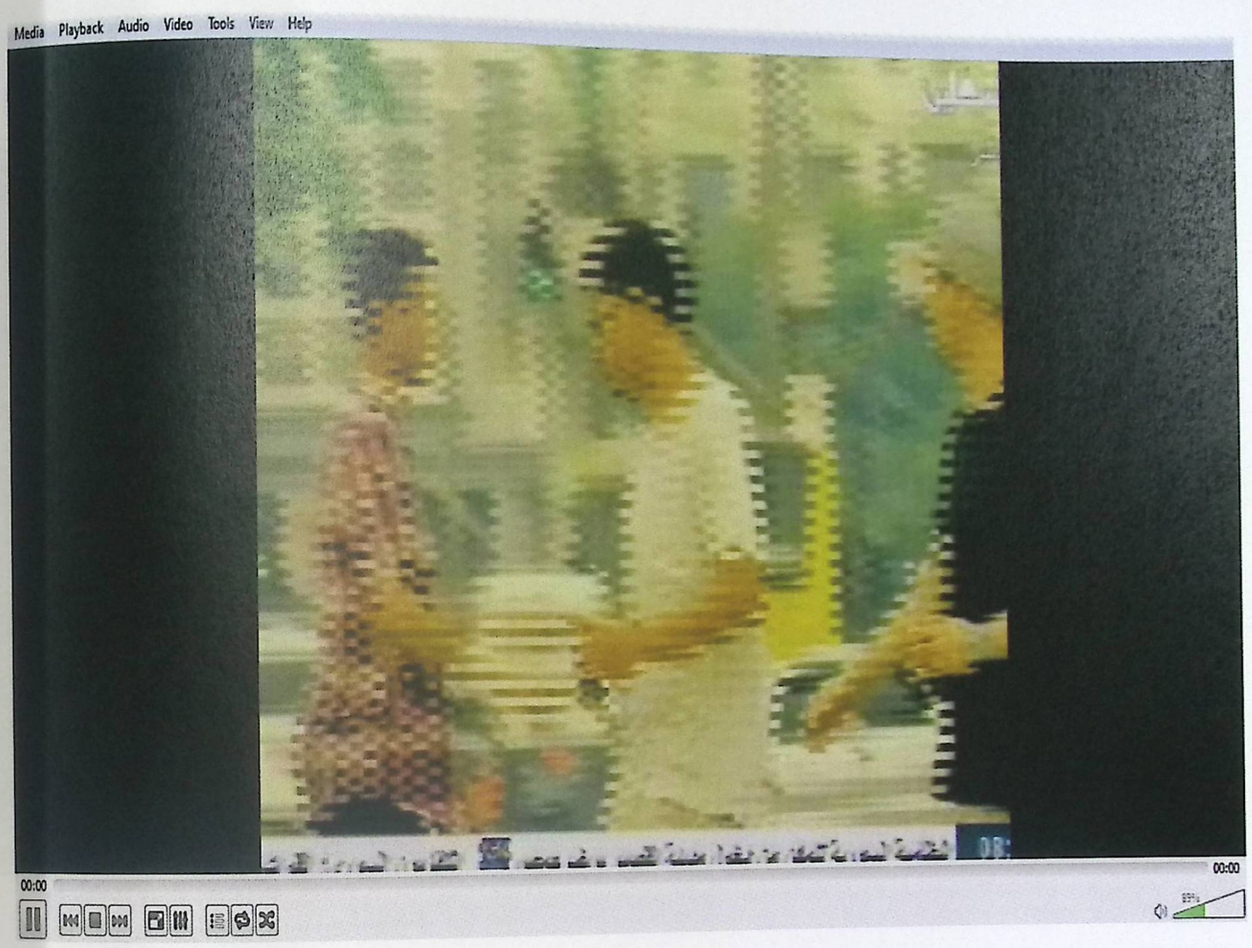


Figure 5.3: Acceptable Video Quality

4) Bad performance is rated as (less than 5) ; the quality of this grade is shown in figure 5.4.

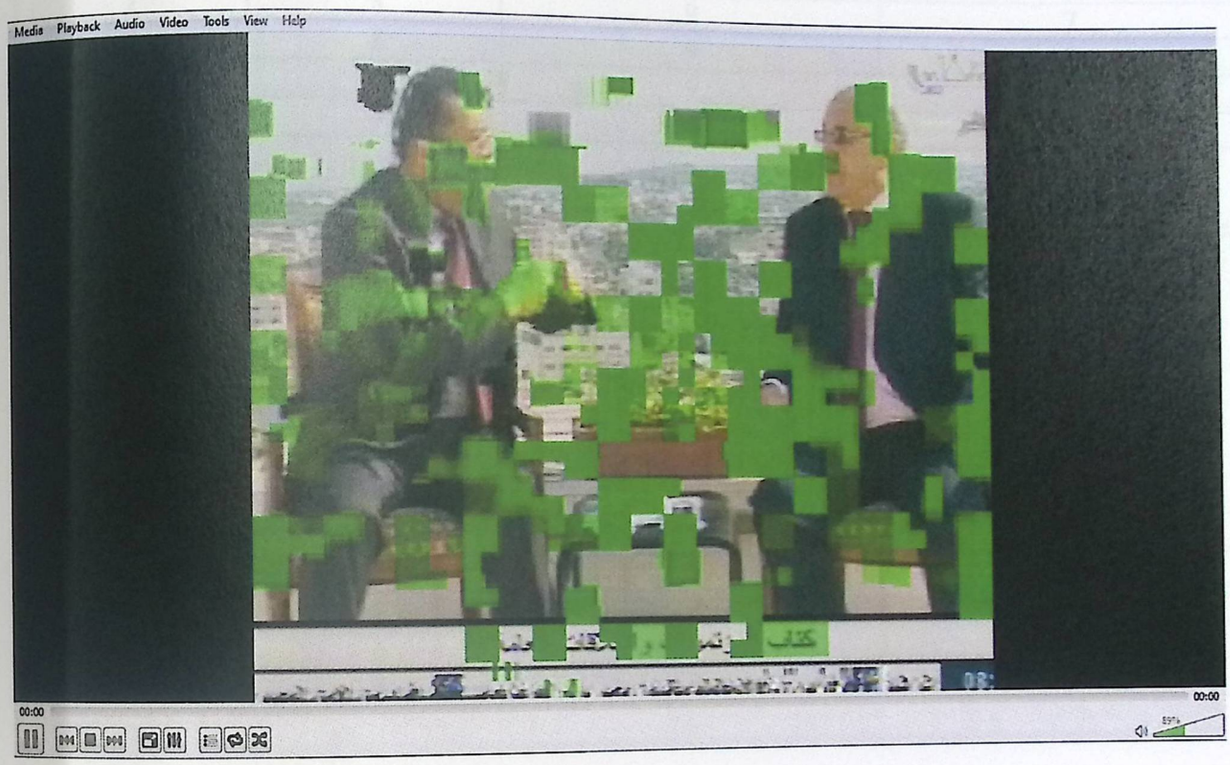


Figure 5.4: Bad Video Quality

5.3.2 Testing Results Tables

Table 5.1: Experiment one results

Number of receiving devices	System Performance	Picture Quality	Sound Quality
1	8	9	7
2	8	9	7
3	8	9	7
4	8	9	7
5	7	8	6

Table 5.2 shows the results in second experiment at different distances between the receiver and the transmitter.

Table 5.2: Experiment Two results

Distance between the Tx and Rx (m)	System Performance	Picture Quality	Sound Quality
0.5	9	9	8
2	8	9	7
4	8	9	7
8	8	8	7
12	6	6	6
16	5	5	5

Table 5.3 shows the results in third experiment at different speeds of downloading a file and the performance of the system.

Table 5.3: Experiment Three results

Downloading speed (K Bytes/sec)	System Performance	Notes
30	9	The effective bandwidth of the Wi-Fi network is approximately 25 Mbps; this will serve our system and internet access with no problems.
50	9	
70	9	
90	9	
110	9	

5.4 Results Evaluation and Discussions

From the previous section we build an idea about the overall performance of the system, this is determined by the quality of the received video, processing delay and synchronization between picture and audio of the transmitted video.

5.4.1 Delay Analysis

After the analysis of results we conclude that the system has an acceptable delay between the traditional TV and our system. This caused by the processing in the hardware, starting from signal processing and conversion into digital form in the EasyCap device. Another factor of delay is the traffic on the network that has queuing problems.

The overall delay in the implemented system is estimated by 3 to 5 seconds. This delay doesn't affect the performance on a single receiving device.

5.4.2 Synchronization Analysis

Analysis and testing of the system shows that, there are no synchronization problems in the received video between the sound and pictures but sometimes the sound is played first and then the picture at the beginning of watching.

5.4.3 Quality Analysis

The experiments on the quality of the video show that the quality of the received video decreased mainly by increasing the distance between the receiving devices and the transmitter, another factor is the obstacles in the path of the signal of transmitter. The quality is not affected by the number of users. Also the audio signal is affected by noise, resulted from digitization process.

Chapter Six

Conclusions and Future work

Chapter contents

6.1 Project Summarization

The main idea of this project is to use Wi-Fi technology to send TV channels from a digital TV receiver to mobile devices. The work in this project is started from studying the A/V signals and analysis them. Then, converting these signals to digital form using analog to digital conversion circuits which exist in one device called "EasyCap".

The data after conversion to digital is processed by micro-processing unit and then sends it by Ethernet port to any router or access point which has Wi-Fi infrastructure.

At receiving devices, it should be a software or application that accepts the video data from the network and display it on the screen.

6.2 Conclusion

Finishing this project was a dream for us, because many problems had faced our team across each step of work. We learned a lot of things, especially in electronics and technology sides and how to deal with any problem and solve it even if it looks impossible to be solved.

Working as team through these two semesters taught us skills to work with other people and how to share the knowledge that one has with other members of project team.

Essential thing that we have faced is the difficulty of getting the required components and codes for the used devices because many of the devices are commercial and the source code is not free.

After we implement our system, we conclude that our system can handle video streaming and internet surfing perfectly. Also, number of receiving devices doesn't affect the quality of received video because the data rate supported by Wi-Fi 802.11g is very suitable.

Performance of the designed system is affected by the power of the Wi-Fi signal on receiving devices.

6.3 Future work

There are possible ideas for improvement our designed system in future. This could be done by us or by anyone want to make our system more efficient and reliable. Some of these ideas are listed on the following points:

- Stream the TV channels over internet, to allow access in more places.
- Add the ability to control the TV receiver from the receiving device.
- Make an enhancement on the quality of video.
- Make our own software to run video on mobile devices.

Appendices

Appendix A

Data Sheets

EasyCAP003 USB2.0 Video Grabber
EasyCAP006 USB2.0 Video Grabber with Audio
DC60+ USB2.0 Video Adapter with Audio
Quick Installation Guide

Overview

- The **EasyCAP series USB 2.0 Video Grabber**, they can capture High-quality video and audio file direct by USB 2.0 interface without sound card. However, the installation is very simple and the external power is unnecessary. Solution for laptop, we have enclosed the professional video editing software Ulead Video Studio 10.0 SE DVD then provide best editing function for you. Video Studio is video-editing software that makes editing your movies as fun as shooting them. The new Video Studio Movie Wizard helps novice users finish stylish movies in only three steps. Share finished projects on DVD, tape, the Web, and mobile devices. High-speed rendering and real-time performance mean less time waiting and more time creating. By the way, you can create many special effect and clip video files...etc.

Notice: EasyCAP003 USB2.0 Video Grabber needs sound card to capture audio file.

Key Features

- Include Professional and easy to learn & used video editor software: Ulead Video Studio 10.0 SE DVD
- Popular USB 2.0 interface and not need other power
- Capture Video & Audio though USB 2.0 interface (EasyCAP003 model needs sound card to capture audio.)
- Support Brightness, Contrast, Hue, and Saturation control
- The dimension suitable that is easy to carry
- Could capture audio without the sound card
- High plug & play
- Support For All Formats: record in DVD+/-R/RW, DVD+/-VR, and DVD-Video.
- Applying to internet conference / net meeting

Specification

- Complies With Universal Serial Bus Specification Rev. 2.0.
- Supports NTSC, PAL, Video format
- Video input: One RCA composite, One S-Video.
- Audio input : Stereo audio (RCA)mm
- Dimension (L)88mm x (W)28mm x (H)18mm
- USB bus power
- Supports high quality video resolution

  NTSC: 720 x 480 @ 30fps

  PAL: 720 x 576 @ 25fps

- Supports Windows XP, Vista

System Requirements

- **USB:** Compliant USB 2.0 free port
- **OS:** Windows XP, Vista
- **CPU:** Intel Pentium 4 or higher
- **HD:** 1 GB of available hard drive space for program installation,4 GB+ hard drive space for video capture and editing
- **Memory:** 256MB of RAM (512MB or above for editing)
- **Display:** Windows-compatible display with at least 1024x768
- **Sound card:** compatible Windows-sound card

Package Contents

- EasyCAP003(or EasyCAP006, or DC60+) USB2.0 Video Grabber
- USB Cable
- Quick Installation Guide
- CD-ROM (included driver and the professional video editor software)

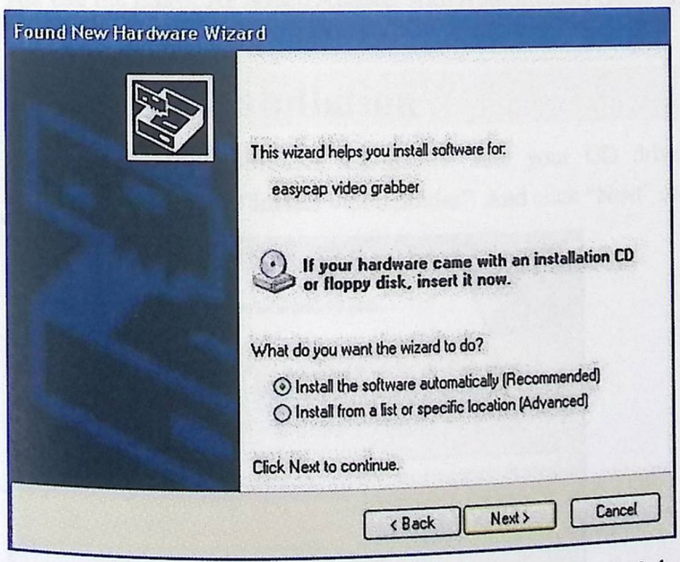
Notice: EasyCAP003 USB2.0 Video Grabber has no USB Cable.

Hardware Installation

Note: If you have any antivirus software enabled, please disable it during the installation of the software.

Before you first connect the EasyCAP series USB 2.0 Video Grabber to your computer, please insert the "Software CD-ROM" into your CD drive.

Select the option "Install Driver".



If prompted select "No, not this time" and select "Next". Select the option "Install the software automatically" and click "Next".

A windows may appear about the "easycap video grabber" not passing the windows logo testing, select "Continue Anyway".

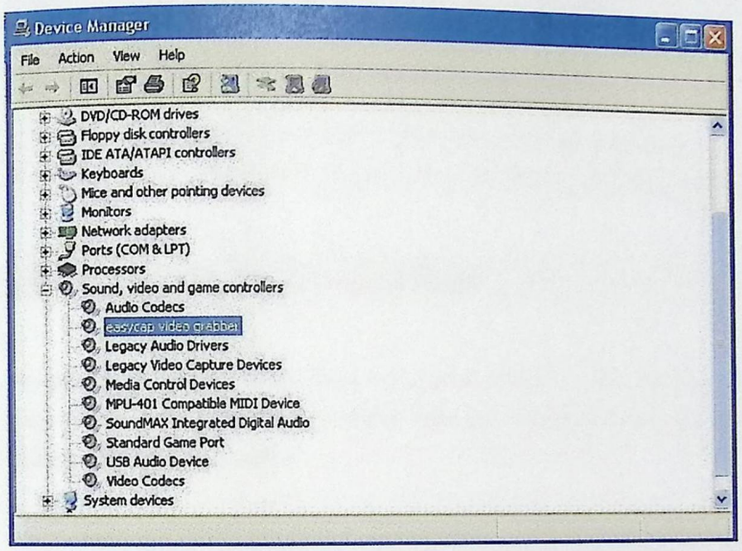
When "Completing the Found New Hardware Wizard" window appears click the "Finish" button

to complete the driver installation.

Note: If any further components are found please repeat the above process until you receive the message "Found new hardware : Your hardware is installed and configured use".

You will now need to check that The drivers are installed correctly. Connect the EasyCAP Video Grabber to your computer, Right Click on My Computer and Left Click on properties. Click on Hardware tab and then Device Manager.

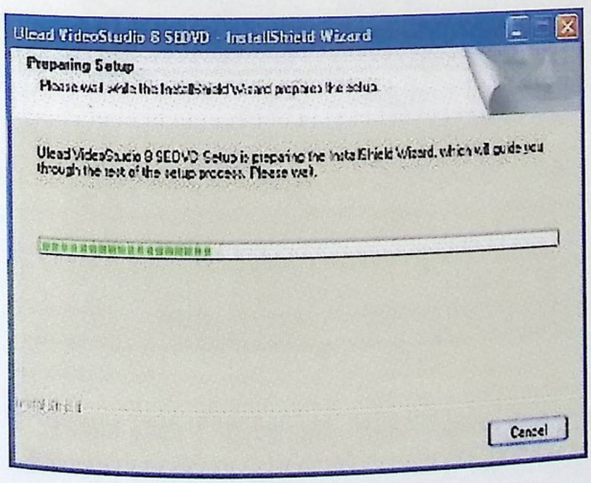
In the Device Manager click on the "Sound, video and game controllers" correctly you should see the "easycap video grabber". If it has a Yellow mark next to it then this means that the driver is not installed correctly. You will need to remove the driver and disconnect the EasyCAP Video Grabber from the computer and reconnect is to install the driver again.



Notice: You should see the "easycap video adapter" on the "Sound, video and game controllers" if you use the EasyCAP Video Adapter with Audio(Model DC60+).

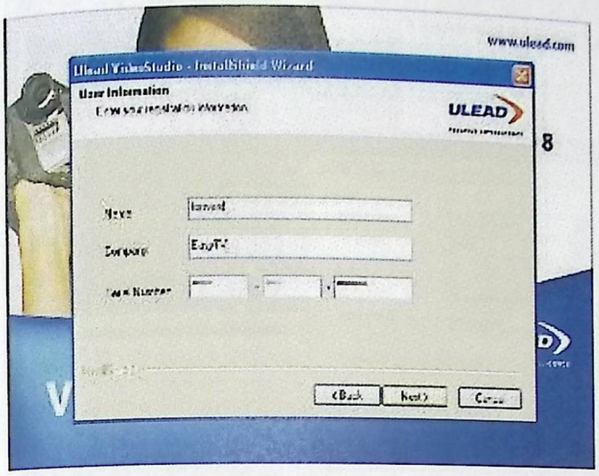
Software Installation

Please insert the "Software CD-ROM" into your CD drive, Select your language for the installation and click "Install Video Studio". And click "Next" or/and "Install".



You need enter user name and serial number (you can find our the serial number on driver CD

bag).

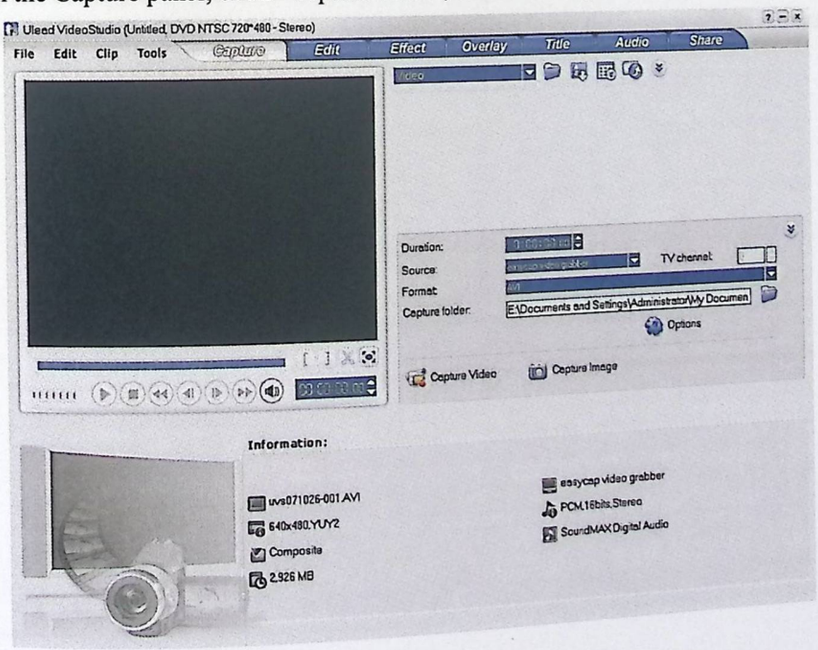


Click "finish" to complete setup, restart your computer if needed.
To start the Video Studio 10.0 application, and enjoy your digital life by use our products.

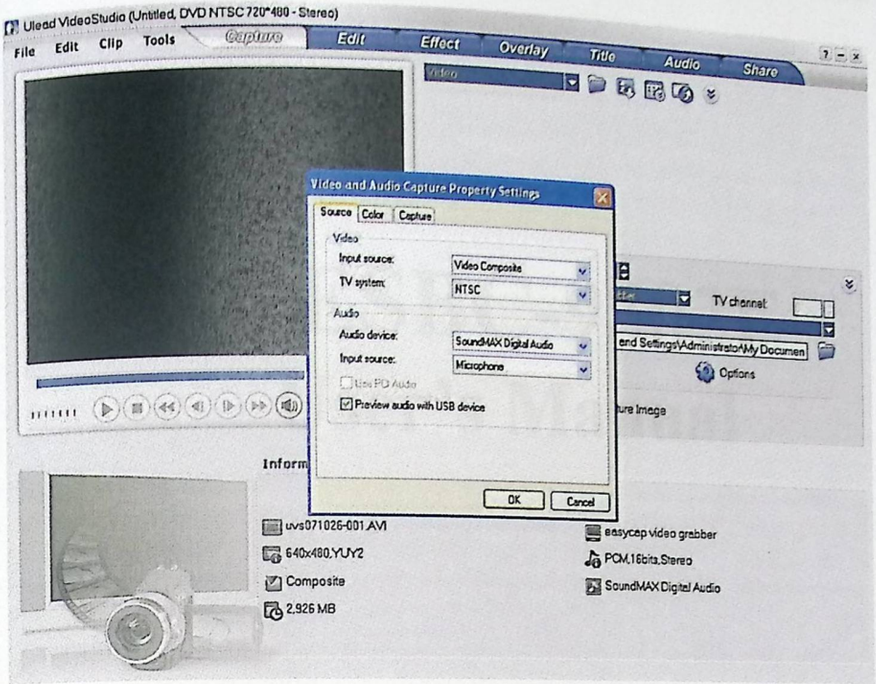
Important Information:

At you first time connect EasyCAP series USB2.0 Video Grabber, and run Ulead software, you may can not see the picture which you are inputting to the device, don't worry, do following setting, it will work well.

1, On the Capture panel, click " capture video", then click "Option" button.



2 Please click " Video and Audio Capture Property setting", and you will see one prompted window.



3, At the prompted window, click "Source" panel, choose the video source as " Video composite" or "S-Video".

4, At the TV System, choose "Pal" or "NTSC' which suitable for your camera.

5, Now you will see the picture, click " Preview audio with USB device" if you want to preview the video with sound.

Notice: For EasyCAP003 USB2.0 Video Grabber, please choose Audio Device is Sound Card.

6, Please view our Website: www.szforwardvideo.com to get the more products infomations.

GLOMATION

GESBC-9302E
User's Manual

Table of Contents

List of Tables

Chapter 1 – Introducing the GESBC-9302E

GESBC-9302E Overview

The GESBC-9302E is a low cost compact sized single board based on Cirrus Logic EP9302 processor. With a large peripheral set targeted to a variety of applications, the GESBC-9302E is well suited for industrial controls, digital media servers, audio jukeboxes, thin clients, set-top boxes, point-of-sale terminals, biometric security systems, and GPS devices will benefit from the EP9302's integrated architecture and advanced features.

Advanced Features

The heart of the GESBC-9302E is the EP9302 which is the one in a series of ARM920T-based processors. The ARM920T microprocessor core with separate 16 Kbyte 64-way set-associative instruction and data caches is augmented by the MaverickCrunch™ co-processor. This enables faster than real-time compression of audio CDs. The proprietary MaverickKey™ unique hardware programmed IDs provide an excellent solution to the growing concern over secure Web content and commerce. MaverickKey IDs can also be used by OEMs and design houses to protect against design piracy by presetting ranges for unique IDs.

The EP9302 is a high-performance, low-power RISC-based device built around a single ARM920T microprocessor core. The ARM920T on the EP9302 functions with a maximum operating clock rate of 200MHz and a power usage between 100mW and 750mW (dependent upon clock speed). The ARM core operates from a 1.8V supply while the I/O operates at 3.3V. The low power consumption makes it an idea platform for battery operated applications.

A high performance 1/10/100 Mbps Ethernet Media Access Controller (EMAC) is included along with external interfaces to SPI and I2S Audio. A two-port USB host and two UARTs are included as well. The list below summarizes the features of the GESBC-9302E.

- 200MHz Processor Core – ARM920T with MMU
- MaverickCrunch™ Math Engine
- MaverickKey™ Security
- 32 ~ 64 MB SDRAM
- 4~16MB NOR FLASH
- Optional 128MB ~ 2GB NAND FLASH
- Ethernet Media Access Controller (EMAC)
- 5 channel 12-bit Analog-to-Digital Converter (ADC
- Universal Asynchronous Receiver / Transmitters (UARTs) with RS-485 Support

- 2 USB Host Port
- Real-Time Clock with battery backup hookup
- Hardware Debug Interface
- Optional 8 channel 12 bit programmable gain A/D converter with 4 additional digital I/O port
- Optional 4 channel 12 bit D/A converter

Figure 1 below shows a picture of the GESBC-9302E .

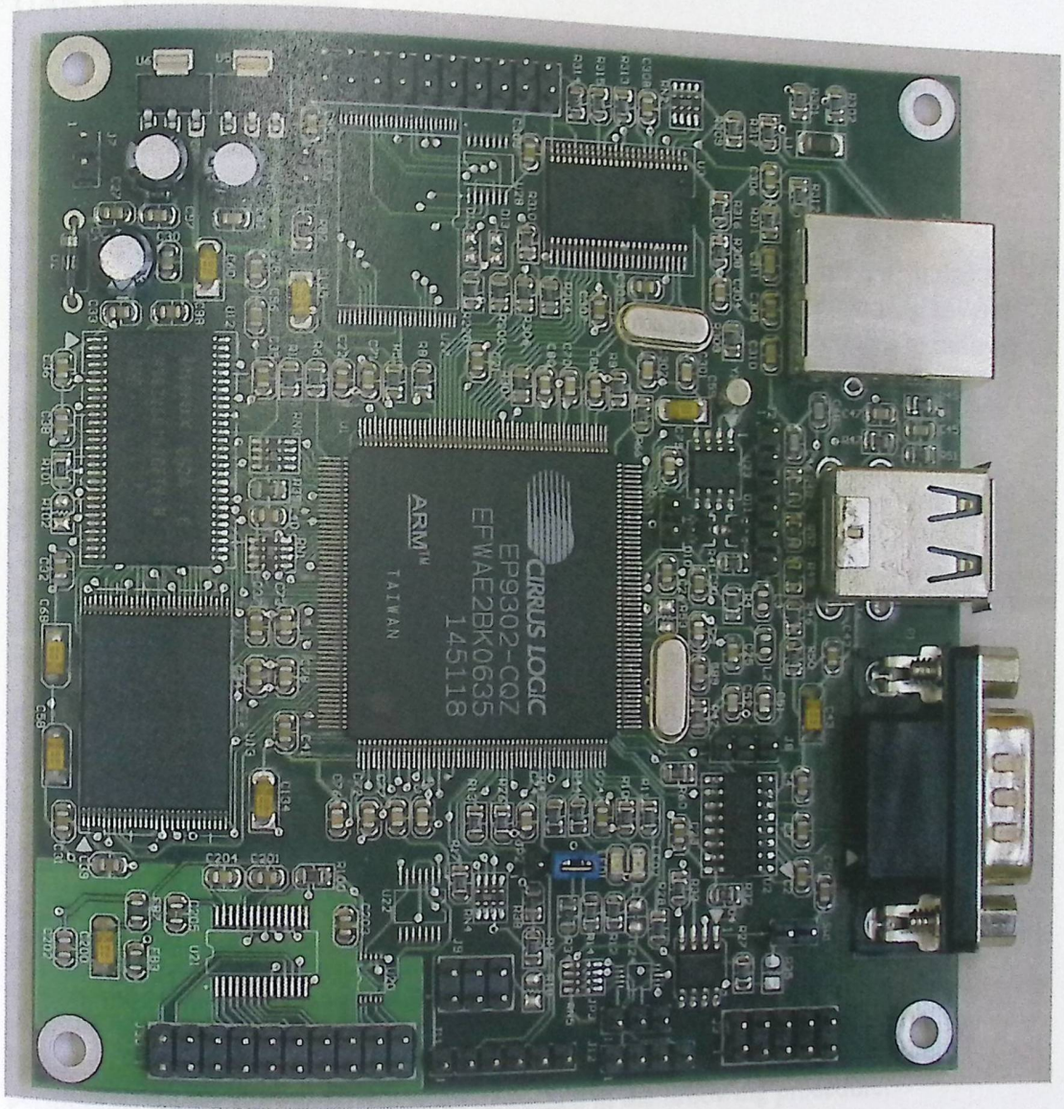


Figure 1. GESBC-9302E

EP9302

The GESBC-9302E is shipped with the Cirrus Logic EP9302 processor. For more information regarding the EP9302 processor please see the EP9302 datasheet.

SDRAM

The GESBC-9302E is shipped with 32MBytes of SDRAM.

FLASH

The GESBC-9302E is shipped with 4MBytes of asynchronous Intel Strata-Flash. 128MB ~ 2GB NAND FLASH is available as an option.

USB

The GESBC-9302E is shipped with two USB host ports.

UART 1

The GESBC-9302E is shipped with a 9-pin interface.

UART 2

The GESBC-9302E is shipped with the 3 wire UART 2 interface.

Ethernet

The GESBC-9302E is shipped with a complete physical and MAC subsystem that is compliant with the ISO/TEC 802.3 topology for a single shared medium with several stations. The EP9302 supports 1/10/100 Mbps transfer rates and interfaces to industry standard physical layer devices.

Chapter 2 – Getting Started

This chapter describes the GESBC-9302E working environment and familiarizes the user with its components and functionality. This chapter contains the following sections:

- Assembly and Connections
  - Describes how to assemble and connect components to the GESBC-9302E Single Board
- Operation
  - Describes how to operate the GESBC-9302E Single Board

Assembly and Connections

In order to use the GESBC-9302E the user must first assemble and connect the peripherals to the GESBC-9302E, as described in the following procedure.

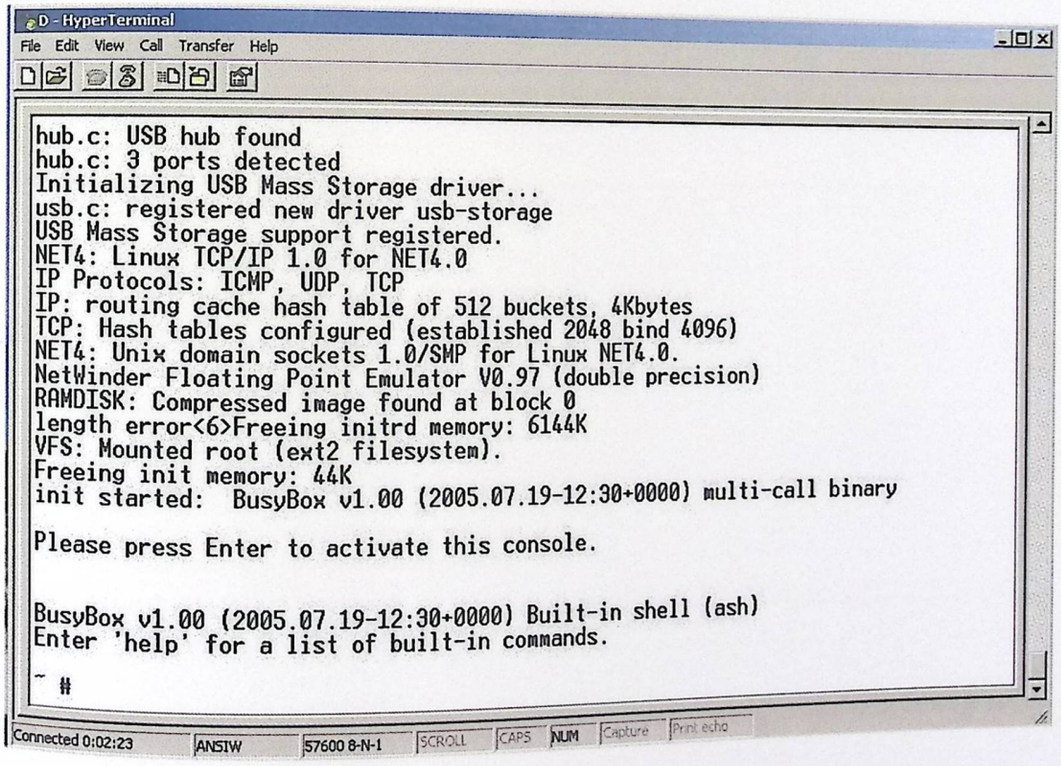
1. Place the GESBC-9302E on a static free surface.
2. Make sure all of the jumpers are in the factory default position. The unit is shipped in a factory default configuration. If the user is uncertain that the GESBC-9302E has the jumpers in the factory default configuration, please see the next section regarding board configuration.

3. Connect 5V regulated power supply to the board.
4. Connect null modem serial cable between GESBC-9302E UART 1 and PC/terminal serial port.
5. Launch a terminal emulator, such as HyperTerminal, or minicom, on the PC configured to connect to the serial port of the GESBC-9302E. Configure the serial port with the following parameters: 57600 bits per second, 8 data bits, no parity, 1 stop bit, no flow control.
6. Connect the board to a local area network (optional)

Operation

The startup procedure for the GESBC-9302E is straightforward. First, the connection of the power harness is required. Second, the null modem serial interface cable must be connected to the UART1 connector. Third, connect the GESBC-9302E to a network that has Internet access. It is recommended that all other cables be tested to determine they are properly seated.

A few seconds after applying power to the GESBC-9302E, debug information will be displayed on the terminal program. The following figure shows what this should look like.



Please see

Chapter 4 – Software Description for more details regarding the software functionality.

Configurations

Jumpers are used to configure the GESBC-9302E to operate in different mode. The following table lists all the settings for each jumper.

Table 1 System configuration

Jumper	Description
2	Boot mode: connect pin 1 and 2 - serial boot connect pin 2 and 3 - flash boot
SW1	Reset switch header
LED3	Power indicator LED header

Chapter 3 – GESBC-9302E Function Blocks

EP9302

The GESBC-9302E Single Board uses the Cirrus Logic EP9302 as the core processor on this development board. The top-level features of EP9302 processor are the following:

- ARM920T RISC Core Processor
- 200 MHz / 200 MIPS Performance
- 16 Kbyte Instruction Cache
- 16 Kbyte Data Cache
- Linux CE enabled MMU
    - Note: Cirrus Logic to supply either a Linux port , including the respective board support package (BSP).
- 100 MHz System Bus
- MaverickCrunch™ Math Engine
- MaverickKey™ Security Features
- 16 bit SDRAM Interface (Up To 4 Banks)
- 16 bit SRAM / FLASH / ROM Interface
- Serial EEPROM Interface
- 10 / 100 Mbps Ethernet MAC
- Two UARTs
- Two-port USB Host
- 5 channel 12 bit ADC
- SPI Port
- Serial Audio Interface
- JTAG Interface

More detailed information regarding the EP9302 processor can be found at www.cirrus.com and on the enclosed disk.

SDRAM

The EP9302 features a unified memory address model where all memory devices are accessed over a common address and data bus. The EP9302 can support a minimum of 1 to a maximum of 4 banks of 16-bit 66 or 100 MHz SDRAM. Additionally, the GESBC-9302E supports 32 SDRAM density. The K4S56132 SDRAM manufactured by Samsung is a part that matches this requirement. The features of the Samsung SDRAM include the following:

- JEDEC Standard 3.3V Power Supply
- LVTTL Compatible with Multiplexed Address
- 4 Bank Operation
- MRS Cycle with Address Key Programs
- CAS Latency of 2 Or 3
- Burst Length of 1, 2, 4, 8, and Full Page
- Burst Type of Sequential and Interleave
- All Inputs Are Sampled on Rising Edge of Clock
- DQM for Masking
- Auto and Self Refresh
- 64ms Refresh Period

FLASH

The GESBC-9302E is shipped with 4 Mbytes of flash memory. The GESBC-9302E can be also ordered with optional 128MB ~ 2GB NAND FLASH.

USB

The GESBC-9302E Single Board provides two USB host connections. The EP9302 USB host controller is configured for two root hub ports and features an integrated transceiver for each port. The EP9302 integrates two USB 2.0 Full Speed host ports. These ports are fully compliant to the OHCI USB 2.0 Full Speed specification (12 Mbps). The controller complies with the OHCI specification for USB Revision 1.1. The USB ports are brought out by a standard double deck USB type A connector.

UART 1 and 2

The GESBC-9302E Single Board is shipped with 2 3-wire UART interface. The UART 1 interface is provided via a standard DB-9 connector. The signal designation is listed in the following tables.

Table 2 P1 UART1 connector

Pin Number	Signal Name	Pin Number	Signal Name
1	NC	2	RX
3	TX	4	NC
5	GND	6	NC
7	NC	8	NC
9	NC	10	N/A

Table 3 J8 UART2 connector

Pin Number	Signal Name
1	RX
2	TX
3	GND

Ethernet

The GESBC-9302E Single Board is shipped with support for a complete Ethernet interface. The EP9302 contains a MAC subsystem that is compliant with the ISO/TEC 802.3 topology for a single shared medium with several stations. The Media Access Controller (MAC) within the EP9302 supports 1/10/100 Mbps transfer rates and interfaces to industry standard physical layer devices. The GESBC-9302E is shipped with the ICS1893AF 100Base-X / 10Base-T Transceiver device which, along with a RJ45 connector, provides the physical layer interface.

SPI bus

The GESBC-9302E Single Board is shipped with a SPI expansion bus header for peripheral expansion. The signal designation is listed in the following table.

Table 4 J11 SPI connector

Pin Number	Signal Name
1	SFRM
2	SSPRX
3	GND
4	SCLK
5	GND
6	SSPTX

On-chip A/D

The GESBC-9302E Single Board is shipped with a 5 channel 12 bit on-chip A/D converter. The signal designation is listed in the following table.

Table 5 J3 On-chip A/D converter

Pin Number	Signal Name
1	GND
2	Channel 1
3	Channel 2
4	GND
5	GND
6	Channel 3
7	Channel 4
8	GND
9	GND
10	Channel 5

GPIO

The GESBC-9302E Single Board provides 20 general purpose I/O signals for external use. The signal designation is listed in the following tables.

Table 6 J1 GPIO 1 connector

Pin Number	Signal Name	Pin Number	Signal Name
1	EGPIO8	2	EGPIO9
3	EGPIO10	4	EGPIO11
5	EGPIO12	6	EGPIO13
7	EGPIO14	8	EGPIO15
9	GPIO0 (CGPIO[0])	10	GPIO1 (HGPIO[5])

11	GPIO2 (HGPIO[4])	12	GPIO3 (HGPIO[3])
13	GPIO4 (HGPIO[2])	14	GPIO5 (FGPIO[1])
15	GPIO6 (FGPIO[2])	16	GPIO7 (FGPIO[3])
17	GND	18	GND
19	VDD3.3	20	VDD3.3

The GPIO3 and GPIO4 are used for optional A/D and D/A chip select when they are installed.

Table 7 J2 GPIO 2 connector

Pin Number	Signal Name	Pin Number	Signal Name
1	EGPIO2	2	EGPIO3
3	EGPIO4	4	EGPIO5
5	GND	6	VDD3.3

EGPIO 8 – 15 are connected directly to Port B of EP9302 CPU, EGPIO 2 – 5 are connected directly to Port A of EP9302 CPU. The GPIO 0 is mapped to Port C of EP9302 CPU. The GPIO 1 – 4 are mapped to Port H of EP9302 CPU which must be enabled in the DeviceCFG register at address 0x8093_0080 [1] . The GPIO 5 -7 are mapped to EP9302 port F.

RTC

In addition to the EP9302 on-chip RTC the GESBC-9302E contains an external real time clock with battery hook-up. The external RTC is DS1337/9 by Maxim-Dallas. The DS1337/9 interfaces the EP9302 CPU via the I2C bus. The DS1337 also is used to provide the 32KHz clock for the EP9302 CPU. When initializing/programming the DS1337 the 32KHz clock must be remain enabled to (default is enabled) allow the EP9302 CPU to proper functioning. The backup battery hook-up is listed in the following table,

Table 8 J5 RTC Battery Backup connector

Pin Number	Signal Name	Pin Number	Signal Name
1	3V	2	GND

[1] The EP9301/02 user guide does not provide clear documentation on the bit for port H. Please refer to page 145 of EP9315 user guide for Port H configuration bit.

JTAG

The GESBC-9302E Single Board is shipped with a 6 pin connector that provides JTAG debug signals for the CPU. The JTAG provides the user with the ability to debug system level programs. The signal designation is listed in the following table.

Table 9 J9 JTAG connector

Pin Number	Signal Name	Pin Number	Signal Name
1	TRSTN	2	TDO
3	TDI	4	TMS
5	TCK	6	GND

Optional A/D and D/A

The GESBC-9302E Single Board provides support for optional 12 bit 8 channel A/D and 4 channel 12 bit D/A. The A/D is provided by TI ADS7870 which is a 12 bit 8 channel analog to digital converter with programmable gain amplifier. It also provides 4 programmable digital I/O. The maximum sampling rate of ADS7870 is 100 ksps. The 8 single ended analog inputs can be also configured as 4 pairs of differential input channels. The optional D/A is provided via TI DAC7554 which is a voltage output 12 bit 4 channel digital to analog converter. The A/D and D/A interface is provided via a 2x10 2.54mm header. The signal designation is listed in the following table.

Table 10 J30 Optional A/D and D/A connector

Pin Number	Signal Name	Pin Number	Signal Name
1	DIO0	2	DIO1
3	DIO2	4	DIO3
5	VDD5	6	DGND
7	AIN0	8	AIN1
9	AIN2	10	AIN3
11	AIN4	12	AIN5
13	AIN6	14	AIN7
15	AGND	16	AGND
17	AOUT0	18	AOUT1
19	AOUT2	20	AOUT3

Optional RS-485

The GESBC-9302E Single Board provides support for optional full/half duplex RS-485 on UART2. The RS-485 signal is provided via J12. JP3 selects between full duplex and half duplex mode.

Table 11 J12 Optional RS-485

Pin Number	Signal Name	Pin Number	Signal Name

1	A1	2	
3	A2	4	B1
			B2

Table 12 JP3 RS-485 mode select

Pin Number	Signal Name
1	5V
2	Mode, high = half duplex mode, low = full duplex mode
3	GND

Reset Switch

The GESBC-9302E Single Board provides a connector to connect to external reset switch.

Table 13 SW1 Reset connector

Pin Number	Signal Name
1	Reset (active low)
2	GND

Power Requirement

The GESBC-9302E Single Board requires regulated 5v DC.

Table 14 J7 Power supply connector

Pin Number	Signal Name
1	5V DC
2	GND

Chapter 4 – Software Description

Overview

This chapter provides information regarding the software that is shipped with the GESBC-9302E Board. The software included with the board is Linux with a few test applications and network utilities. The Linux software provides the user with the ability to test some of the subsystems on the GESBC-9302E board. The download utility provides a means to program a binary image into the flash memory on the GESBC-9302E.

GESBC-9302E Linux Code

The pre-programmed software provides the user with the opportunity to test some of the subsystems on the GESBC-9302E via Linux. This software is programmed into the system FLASH located on the board prior to shipment. The binary image of the shipped code is included on the CD that ships with the board.

Serial Port Interface

The functionality of the serial interface can be demonstrated by looking at the debug messages while the system boot-ups and operates. The setting for the serial interface is described in chapter 2.

USB Interface

The functionality of the USB interface can be shown by hooking up a user supplied USB device.

Ethernet Interface

Ethernet is automatically detected by the Linux kernel and a DHCP client will try to lease network address from connected DHCP server.

Download Utility

The download utility provides the user with a tool for programming the flash memory on the GESBC-9302E with a binary image. The following procedure will allow in-circuit programming of the flash memory via the EP9302 processor:

1) Power the board off.
2) Connect null-modem serial cable to UART1.
3) Set jumper 2 to connect pin 1 and 2 (JP2 factory default is pin 2 and 3).
4) Stop any program that might use the serial port that is connected to GESBC-9302E.
5) Run download utility (assuming download.exe located in same directory as binary image)

   download binary_image_filename.bin
6) "Waiting for board to wake up…" message is displayed.

7) Power the board on.
8) Messages displayed regarding erasing, then programming the flash.
9) "Successfully programmed binary_image_filename.bin" message displayed upon programming completion.
10) Power the board off.
11) Install jumper on pin 2 and 3 of JP2.
12) Power the board on.

Redboot

RedBoot provides a simple interface for loading operating systems and applications onto the GESBC-9302E board. It can also serve as a debug platform for standalone programs using the GDB stub that is built into RedBoot. RedBoot uses a serial console for its input and output. The default serial port setting is 57600,8,N,1. It also supports the built-in Ethernet port and a flash file system and general flash programming.

The board is shipped with Redboot pre-installed. Please refer to Download Utility section for instructions to reload Redboot. Please refer to documents at ECOS website http://ecos.sourceware.org regarding how to rebuild Redboot.

Loading Linux Kernel and root File System

The Redboot boot-loader provides two ways to load Linux kernel and file system into FLASH memory, by Ethernet network and TFTP server or serial connection. The network connection method provides fast loading but if not available, serial port connection method can be used.

After power on the GESBC-9302E board, the following message should be shown on the terminal console on the host PC connected to the GESBC-9302E board[2].

```
+FLASH configuration checksum error or invalid key
EP93xx - no EEPROM, static ESA, or RedBoot config option.
No network interfaces found

RedBoot(tm) bootstrap and debug environment [ROMRAM]
Non-certified release, version v2_0 - built 07:39:29, Dec
18 2004

Platform: Cirrus Logic EDB9301 Board (ARM920T) Rev A
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.
```

[2] A slightly different message will be displayed if the FLASH memory has been initialized and programmed before.

```
RAM: 0x00000000-0x04000000 available
FLASH: 0x60000000 - 0x60400000, 16 blocks of 0x00040000
bytes each.
RedBoot>
```

It's possible to use bootp of Redboot to acquire network address automatically. For situation it is not available, the following procedure can be used to configure a static IP address for the SBC.

```
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis load ramdisk
>> fis load zImage
>> exec -r 0x800000 -s 0x300000
>>
Boot script timeout (1000ms resolution): 1
Use BOOTP for network configuration: false
Gateway IP address:
Local IP address: 192.168.0.112
Local IP address mask:
Default server IP address:
DNS server IP address:
Set eth0 network hardware address [MAC]: true
eth0 network hardware address [MAC]: 0x00:0x00:0x00:0x00:0x80:0x21
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Update RedBoot non-volatile configuration - continue (y/n)? y
... Erase from 0x60780000-0x60781000: .
... Program from 0x03fbe000-0x03fbf000 at 0x60780000: .
RedBoot>
```

The Redboot FLASH file system must be initialized in order to store data in the FLASH file system. The following procedure is used to initialize the Redboot FIS.

```
RedBoot> fis init
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
    Warning: device contents not erased, some blocks may not be
usable
... Erase from 0x607c0000-0x60800000: .
... Program from 0x03fbf000-0x03fff000 at 0x607c0000: .
RedBoot>
```

Load Root File System[3]

The default configuration of GESBC-9302E is using part of SDRAM as RAM disk for Linux root file system. The RAM disk image must be stored in the on-board FLASH memory and loaded by Redboot for the Linux kernel. The image must be loaded into dynamic memory before it can be stored in the on board FLASH memory. To load the ramdisk file to SDRAM, enter the following commands at the terminal console,

```
load -v -r -b 0x800000 -h tftp_host_ip ramdisk_file_name
```

where
-v : verbose
-r : binary format
-b : base address in memory

for serial port download, the command is,

```
load -v -r -b 0x800000 -m ymodem
```

Immediately after entered the above serial download command, start Y-Modem transfer on the terminal program, the download process should start.

The above commands will load ramdisk file into on board SDRAM. To store the image into non-volatile FLASH memory, use the following command,

```
fis create -b 0x800000 -l <ramdisk_file_length> ramdisk_file_name
```

where
-b : is the memory base address
-l : is the ramdisk size. It can be calculated by subtracting the end address from the base address from the Redboot response when loading the ramdisk file.
The ramdisk_file_name can be any arbitrary name.

To verify the image has been stored correctly in the FLASH memory, use the following command,

```
fis list
```

Load Linux Kernel

The kernel image must be loaded into dynamic memory before it can be stored in the on-board FLASH memory. To load Linux kernel, issue the following command at the terminal console connected to the GESBC-9302E board,

[3] The host computer should have *tfpt* server running and Linux kernel and file system file stored in the *tftp* root directory.

```
load -v -r -b 0x80000 -h host_ip_address kernel_image_name
```

where
-v : verbose
-r : binary format
-b : base address in memory

The above command will load Linux kernel image file into on board SDRAM. To store the image into non-volatile FLASH memory, use the following command,

```
fis create -b 0x80000 -l <kernel_image_length> kernel_image_name
```

where
-b : is the memory base address
-l : is the kernel image size. It can be calculated by subtracting the end address from the base address from the Redboot response when loading the kernel image file.
The kernel_image_name can be any arbitrary name.

To verify the image has been stored correctly in the FLASH memory, use the following command,

```
fis list
```

Multiple kernel images or root file systems can be stored in the on-board FLASH memory when memory space permits.

Chapter 5 – Development Tools

Overview

This chapter provides a brief introduction to development tools that are available for the EP9302 System-on-a-Chip processor. The central processing core on the EP9302 is a 200 MHz ARM920T processor. The ARM920T RISC processing core is supported through various toolsets available from third party suppliers. The typical toolset required for the code development is a compiler, assembler, linker and a source-level code debugger. Code debugging is supported via the on-chip JTAG interface.

Linux Development Tool Chain

The ARM Linux development tool chain is widely available on the internet. The support page on the Glomation website http://www.glomationinc.com/support.html contains links to some recent version of tool chain. A host PC running Linux operating system is required to run the development tools. This guide assumes user had basic Linux or Unix application development knowledge.

Host Computer Requirement

The host PC should run Redhead, SuSe, Debian, or other Linux distribution, a RS-232 serial port, at least 500MB free disk space, and a terminal program such as minicom.

Hardware Connection

A null modem cable is required to connect GESBC-9302E to the host computer.

Install Linux Development Tool Chain

The ARM Linux Development Tool chain can be installed in any directory on the host system. The following example uses /usr/local/arm as the installing directory for the ARM Linux Development Tool Chain.

1. Download the Generic-arm_gcc-4.2.3-glibc-2.3.3.tar.bz2 cross tool from Glomation website and login as root,

```
cd /
tar jxvf /[file-path]/Generic-arm_gcc-4.2.3-glibc-2.3.3.tar.bz2
```

2. Set up the directory path variable

```
export PATH=$PATH:/usr/local/arm/gcc-4.2.3-glibc-2.3.3/arm-
```

```
unknown-linux-gnu/
```

The above command can be included in the shell resource file so it is executed every time you login. For bash shell, a good place to put is in .bashrc in your home directory.

Compile Linux Kernel

The GESBC-9302E is shipped with Linux kernel version 2.6.23.12. The patch for the main line Linux kernel source can be downloaded from http://www.glomationinc.com/support.html/.

Prepare Linux Kernel source

Download the main line kernel source from http://www.kernel.org. Untar the kernel source. Download the kernel patch for GESBC-9302E from http://www.glomationinc.com/support.html. Patch the kernel source using the following command,

```
patch -p1 < patch_file_path_and_name
```

Configure Linux Kernel

In the Linux kernel directory, executing the following commands,

```
make ARCH=arm CROSS_COMPILE=generic-arm-linux-gnu- menuconfig
```

If problem occurs, make sure the default PATH variable is set to the correct tool chain directory

Compile Kernel

Once Linux kernel has been configured, it can be compiled using following commands

```
make     (or make zImage)
```

The Linux kernel should compile without error and the kernel image file will be created.

Native Application Development Over NFS

The network file system (NFS) can be used to do native kernel/application development. The include Debian based file system contains native compiler, debugger for ARM processor and many other utilities for fast application development.

The following steps outline the general procedure of mounting NSF on GESBC-9302E board.

1. Create a directory on server computer to house the file system for GESBC-9302E.

2. Download the Debian file system from Glomation website.
3. Unpack the included file system zip file in the newly created directory,
   `tar zxf <file path>/debian-etch-armbase.tar.gz`
4. Change the NFS server configuration to include the directory.
5. Power up GESBC-9302E and press CTRL-C at the terminal connected to the GESBC-9302E to stop the default redboot boot script. Load the on board kernel image from FLASH memory,
   `fis load zImage`
6. Start the Linux operating system by typing,
   `exec -c "root=/dev/nfs nfsroot=<server IP address>:/<NFS directory> ip=dhcp console=ttyAM0"`

The system is now ready to use for application development.

Chapter 6 – Troubleshooting

This chapter provides Troubleshooting information. Search the entries in the Problem column in order to find the item that best describes your situation. Then perform the corrective action in the same row. If the problem persists, contact Glomation.

Appendix B

Codes

Code for configurations of EasyCap device:

```
 THE FOLLOWING PARAMETERS ARE UNDEFINED:
* EASYCAP_DEBUG
* EASYCAP_IS_VIDEODEV_CLIENT
* EASYCAP_NEEDS_USBVIDEO_H
* EASYCAP_NEEDS_V4L2_DEVICE_H
* EASYCAP_NEEDS_V4L2_FOPS
* EASYCAP_NEEDS_UNLOCKED_IOCTL
* IF REQUIRED THEY MUST BE EXTERNALLY DEFINED, FOR EXAMPLE AS COMPILER
* OPTIONS.
#if (!defined(EASYCAP_H))
#define EASYCAP_H
/*---------------------------------------------------------------------------*/
     /*
 * THESE ARE NORMALLY DEFINED
 */
#define PATIENCE 500
#undef PREFER_NTSC
#define PERSEVERE
#undef EASYCAP_TESTCARD
#undef EASYCAP_TESTTONE
#undef NOREADBACK
#undef AUDIOTIME
     /*---------------------------------------------------------------------------
---*/
#include <linux/kernel.h>
#include <linux/errno.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/smp_lock.h>
#include <linux/module.h>
#include <linux/kref.h>
#include <linux/usb.h>
#include <linux/uaccess.h>
#include <linux/i2c.h>
#include <linux/version.h>
#include <linux/workqueue.h>
#include <linux/poll.h>
#include <linux/mm.h>
#include <linux/fs.h>
#include <linux/delay.h>
#include <linux/types.h>

#if defined(EASYCAP_IS_VIDEODEV_CLIENT)
#include <media/v4l2-dev.h>
#if defined(EASYCAP_NEEDS_V4L2_DEVICE_H)
#include <media/v4l2-device.h>
#endif /*EASYCAP_NEEDS_V4L2_DEVICE_H*/
#endif /*EASYCAP_IS_VIDEODEV_CLIENT*/
#define USB_EASYCAP_VENDOR_ID 0x05e1
#define USB_EASYCAP_PRODUCT_ID 0x0408

#define EASYCAP_DRIVER_VERSION "0.8.41"
#define EASYCAP_DRIVER_DESCRIPTION "easycapdc60"
#define USB_SKEL_MINOR_BASE 192
#define DONGLE_MANY 8
```

```
#define INPUT_MANY 6


---*/
#define SAA_0A_DEFAULT 0x7F
#define SAA_0B_DEFAULT 0x3F
#define SAA_0C_DEFAULT 0x2F
#define SAA_0D_DEFAULT 0x00
/*-----------------------------------------------------------------------------*/
/*
* VIDEO STREAMING PARAMETERS:
* USB 2.0 PROVIDES FOR HIGH-BANDWIDTH ENDPOINTS WITH AN UPPER LIMIT
* OF 3072 BYTES PER MICROFRAME for wMaxPacketSize.
*/
/*-----------------------------------------------------------------------------*/
#define VIDEO_ISOC_BUFFER_MANY 16
#define VIDEO_ISOC_ORDER 3
#define VIDEO_ISOC_FRAMESPERDESC ((unsigned int) 1 << VIDEO_ISOC_ORDER)
#define USB_2_0_MAXPACKETSIZE 3072
#if (USB_2_0_MAXPACKETSIZE > PAGE_SIZE)
#error video_isoc_buffer[.] will not be big enough
#endif
#define VIDEO_JUNK_TOLERATE VIDEO_ISOC_BUFFER_MANY
#define VIDEO_LOST_TOLERATE 50
/*-----------------------------------------------------------------------------*/
/*
 * VIDEO BUFFERS
*/
/*-----------------------------------------------------------------------------*/
#define FIELD_BUFFER_SIZE (203 * PAGE_SIZE)
#define FRAME_BUFFER_SIZE (405 * PAGE_SIZE)
#define FIELD_BUFFER_MANY 4
#define FRAME_BUFFER_MANY 6
/*-----------------------------------------------------------------------------*/
/*
* AUDIO STREAMING PARAMETERS
*/
/*-----------------------------------------------------------------------------*/
#define AUDIO_ISOC_BUFFER_MANY 16
#define AUDIO_ISOC_ORDER 3
#define AUDIO_ISOC_BUFFER_SIZE (PAGE_SIZE << AUDIO_ISOC_ORDER)
/*-----------------------------------------------------------------------------*/
/*
* AUDIO BUFFERS
*/
/*-----------------------------------------------------------------------------*/
#define AUDIO_FRAGMENT_MANY 32
/*-----------------------------------------------------------------------------*/
*
* IT IS ESSENTIAL THAT EVEN-NUMBERED STANDARDS ARE 25 FRAMES PER SECOND,
* ODD-NUMBERED STANDARDS ARE 30 FRAMES PER SECOND.
* THE NUMBERING OF STANDARDS MUST NOT BE CHANGED WITHOUT DUE CARE. NOT
* ONLY MUST THE PARAMETER
* STANDARD_MANY
* BE CHANGED TO CORRESPOND TO THE NEW NUMBER OF STANDARDS, BUT ALSO THE
* NUMBERING MUST REMAIN AN UNBROKEN ASCENDING SEQUENCE: DUMMY STANDARDS
* MAY NEED TO BE ADDED. APPROPRIATE CHANGES WILL ALWAYS BE REQUIRED IN
* ROUTINE fillin_formats() AND POSSIBLY ELSEWHERE. BEWARE.
*/
/*-----------------------------------------------------------------------------*/
#define PAL_BGHIN 0
#define PAL_Nc 2
#define SECAM 4
#define NTSC_N 6
#define NTSC_N_443 8
```

Codes

Code for configurations of EasyCap device:

```
 THE FOLLOWING PARAMETERS ARE UNDEFINED:
* EASYCAP_DEBUG
* EASYCAP_IS_VIDEODEV_CLIENT
* EASYCAP_NEEDS_USBVIDEO_H
* EASYCAP_NEEDS_V4L2_DEVICE_H
* EASYCAP_NEEDS_V4L2_FOPS
* EASYCAP_NEEDS_UNLOCKED_IOCTL
* IF REQUIRED THEY MUST BE EXTERNALLY DEFINED, FOR EXAMPLE AS COMPILER
* OPTIONS.
#if (!defined(EASYCAP_H))
#define EASYCAP_H
/*---------------------------------------------------------------------------*/
      /*
 * THESE ARE NORMALLY DEFINED
 */
#define PATIENCE 500
#undef PREFER_NTSC
#define PERSEVERE
#undef EASYCAP_TESTCARD
#undef EASYCAP_TESTTONE
#undef NOREADBACK
#undef AUDIOTIME
      /*---------------------------------------------------------------------------
---*/
#include <linux/kernel.h>
#include <linux/errno.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/smp_lock.h>
#include <linux/module.h>
#include <linux/kref.h>
#include <linux/usb.h>
#include <linux/uaccess.h>
#include <linux/i2c.h>
#include <linux/version.h>
#include <linux/workqueue.h>
#include <linux/poll.h>
#include <linux/mm.h>
#include <linux/fs.h>
#include <linux/delay.h>
#include <linux/types.h>

#if defined(EASYCAP_IS_VIDEODEV_CLIENT)
#include <media/v4l2-dev.h>
#if defined(EASYCAP_NEEDS_V4L2_DEVICE_H)
#include <media/v4l2-device.h>
#endif /*EASYCAP_NEEDS_V4L2_DEVICE_H*/
#endif /*EASYCAP_IS_VIDEODEV_CLIENT*/
#define USB_EASYCAP_VENDOR_ID 0x05e1
#define USB_EASYCAP_PRODUCT_ID 0x0408

#define EASYCAP_DRIVER_VERSION "0.8.41"
#define EASYCAP_DRIVER_DESCRIPTION "easycapdc60"
#define USB_SKEL_MINOR_BASE 192
#define DONGLE_MANY 8
```

```
#define NTSC_M 1
#define NTSC_443 3
#define NTSC_M_JP 5
#define PAL_60 7
#define PAL_M 9
#define PAL_BGHIN_SLOW 10
#define PAL_Nc_SLOW 12
#define SECAM_SLOW 14
#define NTSC_N_SLOW 16
#define NTSC_N_443_SLOW 18
#define NTSC_M_SLOW 11
#define NTSC_443_SLOW 13
#define NTSC_M_JP_SLOW 15
#define PAL_60_SLOW 17
#define PAL_M_SLOW 19
#define STANDARD_MANY 20
/*---------------------------------------------------------------------------*/
/*
* ENUMS
*/
/*---------------------------------------------------------------------------*/
enum {
AT_720x576,
AT_704x576,
AT_640x480,
AT_720x480,
AT_360x288,
AT_320x240,
AT_360x240,
RESOLUTION_MANY
};
enum {
FMT_UYVY,
FMT_YUY2,
FMT_RGB24,
FMT_RGB32,
FMT_BGR24,
FMT_BGR32,
PIXELFORMAT_MANY
};
enum {
FIELD_NONE,
FIELD_INTERLACED,
INTERLACE_MANY
};
#define SETTINGS_MANY (STANDARD_MANY * \
            RESOLUTION_MANY * \
            2 * \
            PIXELFORMAT_MANY * \
            INTERLACE_MANY)
/*---------------------------------------------------------------------------*/
/*
* STRUCTURE DEFINITIONS
*/
/*---------------------------------------------------------------------------*/
struct easycap_dongle {
struct easycap *peasycap;
struct mutex mutex_video;
struct mutex mutex_audio;
};
/*---------------------------------------------------------------------------*/
struct data_buffer {
struct list_head list_head;
void *pgo;
void *pto;
```

```
__u16 kount;
__u16 input;
};
/*---------------------------------------------------------------------------*/
struct easycap {
#define TELLTALE "expectedstring"
char telltale[16];
int isdongle;
/*vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv*/
#if defined(EASYCAP_IS_VIDEODEV_CLIENT)
struct video_device video_device;
#if defined(EASYCAP_NEEDS_V4L2_DEVICE_H)
struct v4l2_device v4l2_device;
#endif /*EASYCAP_NEEDS_V4L2_DEVICE_H*/
      #endif /*EASYCAP_IS_VIDEODEV_CLIENT*/
/*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*/
int status;
unsigned int audio_pages_per_fragment;
unsigned int audio_bytes_per_fragment;
unsigned int audio_buffer_page_many;
      #define UPSAMPLE
      #if defined(UPSAMPLE)
__s16 oldaudio;
#endif /*UPSAMPLE*/
int ilk;
bool microphone;
struct usb_device *pusb_device;
struct usb_interface *pusb_interface;
struct kref kref;
int queued[FRAME_BUFFER_MANY];
int done[FRAME_BUFFER_MANY];
wait_queue_head_t wq_video;
wait_queue_head_t wq_audio;
int input;
int polled;
int standard_offset;
int format_offset;
struct inputset inputset[INPUT_MANY];
bool ntsc;
int fps;
int usec;
int tolerate;
int skip;
int skipped;
int lost[INPUT_MANY];
int merit[180];
struct timeval timeval0;
struct timeval timeval1;
struct timeval timeval2;
struct timeval timeval3;
struct timeval timeval6;
struct timeval timeval7;
struct timeval timeval8;
long long int dnbydt;
int video_interface;
int video_altsetting_on;
int video_altsetting_off;
int video_endpointnumber;
int video_isoc_maxframesize;
int video_isoc_buffer_size;
int video_isoc_framesperdesc;
int video_isoc_streaming;
int video_isoc_sequence;
int video_idle;
int video_eof;
```

```
int video_junk;
struct data_buffer video_isoc_buffer[VIDEO_ISOC_BUFFER_MANY];
struct data_buffer \
        field_buffer[FIELD_BUFFER_MANY][(FIELD_BUFFER_SIZE/PAGE_SIZE)];
struct data_buffer \
        frame_buffer[FRAME_BUFFER_MANY][(FRAME_BUFFER_SIZE/PAGE_SIZE)];
struct list_head urb_video_head;
struct list_head *purb_video_head;
u8 cache[8];
u8 *pcache;
int video_mt;
int audio_mt;
long long audio_bytes;
u32 isequence;
int vma_many;
/*---------------------------------------------------------------------------*/
/*
* BUFFER INDICATORS
*/
/*---------------------------------------------------------------------------*/
int field_fill; /* Field buffer being filled by easycap_complete(). */
            /* Bumped only by easycap_complete(). */
int field_page; /* Page of field buffer page being filled by */
            /* easycap_complete(). */
int field_read; /* Field buffer to be read by field2frame(). */
            /* Bumped only by easycap_complete(). */
int frame_fill; /* Frame buffer being filled by field2frame(). */
            /* Bumped only by easycap_dqbuf() when */
            /* field2frame() has created a complete frame. */
int frame_read; /* Frame buffer offered to user by DQBUF. */
            /* Set only by easycap_dqbuf() to trail frame_fill.*/
int frame_lock; /* Flag set to 1 by DQBUF and cleared by QBUF */
/*---------------------------------------------------------------------------*/
/*
* IMAGE PROPERTIES
*/
/*---------------------------------------------------------------------------*/
__u32 pixelformat;
int width;
int height;
int bytesperpixel;
bool byteswaporder;
bool decimatepixel;
bool offerfields;
int frame_buffer_used;
int frame_buffer_many;
int videofieldamount;
int brightness;
int contrast;
int saturation;
int hue;
int allocation_video_urb;
int allocation_video_page;
int allocation_video_struct;
int registered_video;
/*---------------------------------------------------------------------------*/
/*

 * SOUND PROPERTIES                                                          */
/*---------------------------------------------------------------------------*/
int audio_interface;
int audio_altsetting_on;
int audio_altsetting_off;
```

```
int audio_endpointnumber;
int audio_isoc_maxframesize;
int audio_isoc_buffer_size;
int audio_isoc_framesperdesc;
int audio_isoc_streaming;
int audio_idle;
int audio_eof;
int volume;
int mute;
struct data_buffer audio_isoc_buffer[AUDIO_ISOC_BUFFER_MANY];
struct list_head urb_audio_head;
struct list_head *purb_audio_head;
/*---------------------------------------------------------------------------*/
/*
* BUFFER INDICATORS
*/
/*---------------------------------------------------------------------------*/
int audio_fill; /* Audio buffer being filled by easysnd_complete(). */
            /* Bumped only by easysnd_complete(). */
int audio_read; /* Audio buffer page being read by easysnd_read(). */
            /* Set by easysnd_read() to trail audio_fill by */
            /* one fragment. */
/*---------------------------------------------------------------------------*/
/*
* SOUND PROPERTIES
*/
/*---------------------------------------------------------------------------*/
int audio_buffer_many;
int allocation_audio_urb;
int allocation_audio_page;
int allocation_audio_struct;
int registered_audio;
long long int audio_sample;
long long int audio_niveau;
long long int audio_square;
struct data_buffer audio_buffer[];
};
/*---------------------------------------------------------------------------*/
/*
* VIDEO FUNCTION PROTOTYPES
*/
/*---------------------------------------------------------------------------*/
void easycap_complete(struct urb *);
int easycap_open(struct inode *, struct file *);
int easycap_release(struct inode *, struct file *);
long easycap_ioctl_noinode(struct file *, unsigned int, \
                           unsigned long);
int easycap_ioctl(struct inode *, struct file *, unsigned int, \
                           unsigned long);
/*vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv*/
#if defined(EASYCAP_IS_VIDEODEV_CLIENT)
int easycap_open_noinode(struct file *);
int easycap_release_noinode(struct file *);
int videodev_release(struct video_device *);
#endif /*EASYCAP_IS_VIDEODEV_CLIENT*/
/*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*/
unsigned int easycap_poll(struct file *, poll_table *);
int easycap_mmap(struct file *, struct vm_area_struct *);
int easycap_usb_probe(struct usb_interface *, \
                      const struct usb_device_id *);
void easycap_usb_disconnect(struct usb_interface *);
void easycap_delete(struct kref *);
void easycap_vma_open(struct vm_area_struct *);
void easycap_vma_close(struct vm_area_struct *, struct vm_fault *);
int easycap_vma_fault(struct vm_area_struct *, struct vm_fault *);
```

```
int easycap_dqbuf(struct easycap *, int);
int submit_video_urbs(struct easycap *);
int kill_video_urbs(struct easycap *);
int field2frame(struct easycap *);
int redaub(struct easycap *, void *, void *, \
                        int, int, __u8, __u8, bool);
void easycap_testcard(struct easycap *, int);
int fillin_formats(void);
int reset(struct easycap *);
int newinput(struct easycap *, int);
int adjust_standard(struct easycap *, v4l2_std_id);
int adjust_format(struct easycap *, __u32, __u32, __u32, \
                                int, bool);
int adjust_brightness(struct easycap *, int);
int adjust_contrast(struct easycap *, int);
int adjust_saturation(struct easycap *, int);
int adjust_hue(struct easycap *, int);
int adjust_volume(struct easycap *, int);
/*---------------------------------------------------------------------------*/
/*
 * AUDIO FUNCTION PROTOTYPES
*/
/*---------------------------------------------------------------------------*/
void easysnd_complete(struct urb *);
ssize_t easysnd_read(struct file *, char __user *, size_t, loff_t *);
int easysnd_open(struct inode *, struct file *);
int easysnd_release(struct inode *, struct file *);
long easysnd_ioctl_noinode(struct file *, unsigned int, \
                                unsigned long);
int easysnd_ioctl(struct inode *, struct file *, unsigned int, \
                                unsigned long);
unsigned int easysnd_poll(struct file *, poll_table *);
void easysnd_delete(struct kref *);
int submit_audio_urbs(struct easycap *);
int kill_audio_urbs(struct easycap *);
void easysnd_testtone(struct easycap *, int);
int audio_setup(struct easycap *);
/*---------------------------------------------------------------------------*/
/*
* LOW-LEVEL FUNCTION PROTOTYPES
*/
/*---------------------------------------------------------------------------*/
int audio_gainget(struct usb_device *);
int audio_gainset(struct usb_device *, __s8);
int set_interface(struct usb_device *, __u16);
int wakeup_device(struct usb_device *);
int confirm_resolution(struct usb_device *);
int confirm_stream(struct usb_device *);
int setup_stk(struct usb_device *, bool);
int setup_saa(struct usb_device *, bool);
int setup_vt(struct usb_device *);
int check_stk(struct usb_device *, bool);
int check_saa(struct usb_device *, bool);
int ready_saa(struct usb_device *);
int merit_saa(struct usb_device *);
int check_vt(struct usb_device *);
int select_input(struct usb_device *, int, int);
int set_resolution(struct usb_device *, \
                        __u16, __u16, __u16, __u16);
int read_saa(struct usb_device *, __u16);
int read_stk(struct usb_device *, __u32);
int write_saa(struct usb_device *, __u16, __u16);
int wait_i2c(struct usb_device *);
int write_000(struct usb_device *, __u16, __u16);
int start_100(struct usb_device *);
```

```
int stop_100(struct usb_device *);
int write_300(struct usb_device *);
int read_vt(struct usb_device *, __u16);
int write_vt(struct usb_device *, __u16, __u16);
int regset(struct usb_device *, __u16, __u16);
int regget(struct usb_device *, __u16, __u16, void *);
int isdongle(struct easycap *);
/*---------------------------------------------------------------------------*/
struct signed_div_result {
long long int quotient;
/*---------------------------------------------------------------------------*/
/*
* (unsigned char *)P pointer to next byte pair
* (long int *)X pointer to accumulating count
* (long int *)Y pointer to accumulating sum
* (long long int *)Z pointer to accumulating sum of squares
*/
/*---------------------------------------------------------------------------*/
#define SUMMER(P, X, Y, Z) do { \
    unsigned char *p; \
    unsigned int u0, u1, u2; \
    long int s; \
    p = (unsigned char *)(P); \
    u0 = (unsigned int) (*p); \
    u1 = (unsigned int) (*(p + 1)); \
    u2 = (unsigned int) ((u1 << 8) | u0); \
    if (0x8000 & u2) \
        s = -(long int)(0x7FFF & (~u2)); \
    else \
        s = (long int)(0x7FFF & u2); \
    *((X)) += (long int) 1; \
    *((Y)) += (long int) s; \
    *((Z)) += ((long long int)(s) * (long long int)(s)); \
} while (0)
/*---------------------------------------------------------------------------*/
#endif /*EASYCAP_H*/
```

References

Book	[1] Teik-Kheong Tan, Benny Bing and Stuart J. Herry, *WorldWide Wi-Fi, Technological Trends and Business Strategies*, (Aug 18, 2003)
	[5] Sao-Jie Chen and Yong-Hsiang Hsieh, *IQ Calibration Techniques for CMOS Radio Transceivers (Analog Circuits and Signal Processing)*, (Nov 24, 2010)
	[6] Shuangbao Paul Wang, Robert S. Ledley, *Computer Architecture and Security: Fundamentals of Designing Secure Computer Systems*, 2012
	[8] P. Gnanasivam , *Telecommunication Switching and Networks*, (Dec 1, 2008)
	[9] Marcel J.M. Pelgrom, *Analog-to-Digital Conversion*, 2010
	[12] Amit Kamra, Pankaj Bhambri, *Computer Peripherals And Interfaces*,2008
	[16] Tomas Marek, *Video Formats Guide*, TECHNICAL UNIVERSITY OF LIBEREC, 2010 http://gizmodo.com/5093670/giz-explains-every-video- format-you-need-to-know
Internet Document	[4] *Webopedia Site, DidYouKnow* available at : http://www.webopedia.com/DidYouKnow/Computer_Science/wifi_explained.asp
	[7] *EasyCap USB 2.0 Audio-Video Adapter Quick Installation Guide* available at: http://site.theimportsworld.com/support/manuals/easycap/hardwareguide.pdf
	[13] USB Implementers Forum Inc, available at : http://www.accesscomms.com.au/reference/usb.htm
	[15] Long Tail Video Forum available at : http://www.longtailvideo.com/blog/19578/what-is-video-streaming/
Chapter	[11] *Wilbert Dijkhof*, Doom9's Forum ,Video Formats,2011 , 55 http://www.doom9.org/index.html?/capture/analogue_video.html
Journal Article	[10] TV Without Borders, Tutorials, DTV,2009 ,1500
Thesis	[2] F. Swety,H. Mashrqa, *Wireless Technology Personality Television*, Bachelor Thesis, Palestine Polytechnic University 2011.
	[3] A. Amro, *Free Space Laser Video Transmission* Bachelor Thesis, Palestine Polytechnic University 2012.
Conference Workshop	[14] Mohammad Omar, *Information Security Event* at Beir Ziet University on February 14th, 2012