

**USB SECURITY CAMERA**

By

**ZAIM B MOHD MAHDZIR**

Submitted to the Electrical & Electronics Engineering Programme  
in Partial Fulfillment of the Requirements  
for the Degree  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

© Copyright 2005

by

**ZAIM B MOHD MAHDZIR 2005**

# **CERTIFICATION OF APPROVAL**

## **USB SECURITY CAMERA**

by

Zaim bin Mohd Mahdzir

A project dissertation submitted to the  
Electrical & Electronics Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfilment of the requirement for the  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Approved:



---

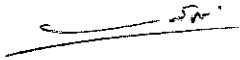
Dr Mohammad Awan  
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS  
TRONOH, PERAK

December 2005

## **CERTIFICATION OF ORIGINALITY**

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



---

Zaim B Mohd Mahdzir

## **ABSTRACT**

USB Security Camera is a new way for a cheaper and convenient security system. The objective of this project is to develop a software that can recognizes more than one USB cameras connected to a single computer and streams each camera's output simultaneously or alternately to a computer's monitor. The software was written in Visual Basic 6.0 so that it is compatible with Windows platform. The main features of the application developed are automatic picture snap, motion detector and can support any USB camera. Motion detection features is a useful features in a place where there should be no movement. Unfortunately, the main aim which is to develop software that can detect more than one camera has failed. The software is not working as planned.

## **ACKNOWLEDGEMENTS**

Highest appreciation and sincere gratitude regarded to the project supervisor, Dr Mohammad Awan, for the guidance and attention in helping me to come up with a progress during these two semesters. His comments had urged me to give my best in order to achieve the goal of the project.

Utmost thanks to the laboratory technician for the cooperation given in using all the equipments related to the project. The continuous help and information given have assist in ensuring the smoothness of the project carried out.

Not to forget, to all my peers, for their cooperation and willingness to share their ideas regarding this project. The clarifications given mainly on coding works that really had helped me a lot.

Thank you to all of you.

## TABLE OF CONTENTS

LIST OF TABLES .....	ix
LIST OF FIGURES.....	x
LIST OF ABBREVIATIONS .....	xi
CHAPTER 1 INTRODUCTION .....	1
1.1 Introduction to USB Security Camera.....	1
1.2 Background of Study .....	2
1.3 Problem Statement.....	3
1.3.1 Problem Identification .....	3
1.3.2 Significance of the Project .....	3
1.4 Objective and Scope of Study .....	4
CHAPTER 2 LITERATURE REVIEW/ THEORY .....	5
2.1 Existing security camera system in the market .....	5
2.1.1 Introduction.....	5
2.1.2 Home Security Camera .....	5
2.1.3 Wireless Security Camera.....	6
2.1.4 CCTV Security Camera .....	6
2.1.5 Internet Security Camera .....	6
2.1.6 Fake Security Camera .....	7
2.2 USB Camera Candidates .....	7
2.2.1 Pakatak U350 PC USB Camera [1] .....	7
2.2.2 Pakatak U703 PC USB Night Vision Camera [1].....	8
2.2.3 Mototech Webcam .....	8
2.2.4 Genius VideoCAM Web V4 [6] .....	9
2.3 Selected USB Camera .....	9
2.4 Existing Security Camera Software.....	10
2.4.1 Crime Catcher 3.0 .....	10
2.4.2 NET Video Spy 1.82.....	11
2.4.3 Honestech Video Security Software [2].....	12
2.5 Visual Basic 6.0.....	13
2.5.1 The Structure of Microsoft Visual Basic .....	13
CHAPTER 3 METHODOLOGY/ PROJECT WORK .....	17

3.1 Procedure Identification .....	17
3.1.1 Research and Analysis .....	18
3.1.2 Design the application.....	18
3.1.3 Coding Process.....	18
3.1.4 Debugging .....	18
3.2 Tools Required .....	18
3.2.1 Hardware : Genius VideoCAM Web V4 & Mototech Webcam .....	18
3.2.2 Hardware : PC with Windows XP Operating System .....	18
3.2.3 Software : Visual Basic 6.0 Enterprise Edition[5].....	19
CHAPTER 4 RESULT AND DISCUSSION .....	20
4.1 Program Description.....	20
4.2 Coding Work .....	20
4.2.1 Graphic User Interface design.....	20
4.2.2 Variables declaration.....	21
4.2.3 API Function.....	22
4.3 Motion Detection Code .....	23
4.4 Discussion and Problem Encountered .....	23
4.4.1 Windows XP cannot detect two cameras with identical WDM drivers .....	24
4.4.2 The software failed to detect more than one camera .....	24
CHAPTER 5 CONCLUSION AND RECOMMENDATION .....	25
5.1 Conclusion.....	25
5.2 Recommendation.....	26
REFERENCES.....	27
APPENDICES.....	28
Appendix A VISUAL basic CODES .....	29
Appendix B Screen shot from the USB Security Camera Software.....	41
Appendix C Gantt Chart .....	42

## **LIST OF TABLES**

Table 1 Visual Basic Naming Convention.....	16
---	----



## LIST OF FIGURES

Figure 1 Screenshot of Crime Catcher 3.0 .....	11
Figure 2 Screenshot of NET Video Spy 1.82 .....	11
Figure 3 Screenshot of Honestech Video Security Software .....	12
Figure 4 Visual Basic Editor .....	16
Figure 5 Methodology diagram.....	17
Figure 6 GUI for the application .....	20
Figure 7 API Viewer .....	22
Figure 8 Real movement detection diagram.....	23
Figure 9 <i>Visual Basic form showing empty scroll box</i> .....	24
Figure 10 Overview layout “USB Security Camera”.....	26
Figure 11 Screenshot of the USB Security Camera in action .....	41

## **LIST OF ABBREVIATIONS**

1. USB – Universal Serial Bus
2. API – Application Programming Interface
3. CCTV – Close Circuit TeleVision
4. GUI – Graphic User Interface
5. WDM – Windows Device Manager

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Introduction to USB Security Camera**

USB security camera systems are becoming increasingly popular in today's age because peoples want the utmost in protection. Many camera systems can be purchased anywhere and are available from as little as a hundred ringgit to sophisticated systems in the thousands or even tens of thousands of ringgit range.

Preventing crime is sometimes as important as solving a crime. Cameras act as deterrents in many ways and can sometimes prevent an act of crime from ever taking place. If one's home or property broken into, though, cameras can also aid authorities in catching the perpetrator.

Camera systems are also becoming very popular with families who are forced to be away from the home and rely on baby-sitters, pet-watchers or other attendants. Peace of mind can come from just knowing that there's always a set of eyes on the goings on inside one's home.

Camera systems also can be implemented in many different ways. Standard surveillance cameras can be placed in an out of reach, yet obvious position for the purposes of deterrence and monitoring. Home security camera systems can also be stealthy and hidden from sight so as to not alarm an individual or party that they are being monitored. One must always check on laws and regulations as to what is permitted, but assuming that your application is not illegal or immoral, hidden surveillance systems are often the popular choice.

Security camera equipment is manufactured by many companies. Some of the most popular companies include Sony, Samsung, Sanyo, Ademco (Honeywell Video Solutions), Bosch and more. These manufacturers typically package what they term CCTV systems which essentially include cameras, monitors and recording equipment.

## 1.2 Background of Study

The universal serial bus (USB) is an external bus architecture for connecting a USB-compatible peripheral device, such as a mouse, keyboard, or printer, to a host computer. USB is a communication protocol that supports serial data transfers between a host computer and a USB-compatible peripheral device.

USB provides an expandable, hot-pluggable Plug and Play serial interface that ensures a standard, low-cost connection for peripheral devices such as keyboards, mice, joysticks, printers, scanners, storage devices, modems, and video conferencing cameras. Migration to USB is recommended for all peripheral devices that use legacy ports such as the PS/2, serial, and parallel ports. USB camera usually a lot cheaper than any conventional camera that uses other interface making it a good candidate for a more cost efficient security system.

Currently, USB support focuses on the host side of the USB specification, which enables support for USB peripherals. Some support is provided for the device side of the USB specification through the USB function controller driver. The USB function controller driver, however, exists primarily to support USB connectivity to desktop computers.

USB Security Camera is a new trend comparable to the traditional yet effective security camera system using CCTV that it can deliver the same features and functions at a lower cost. It is a more favorable solution among small entrepreneurs and end users. In this project, the written program must be able to support multiple USB cameras and turning them into sophisticated security system. At the first stage of this project, this project aim is to develop security software that can support at least two USB cameras and have a motion detector.

There are currently many security camera software exist in the market, with many advance features. This project was not meant to bring any advancement in security technology, it is solely a learning tools to gain experience and more knowledge on this field.

### **1.3 Problem Statement**

There are currently a lot of Security Camera exist in the market based on USB technology. In this project, the aim is to develop an application that recognizes at least two USB cameras connected to a single computer and streams each camera's output simultaneously or alternately to a computer's monitor. It is crucial to limit the scope to the appropriate level where it should be not too difficult to accomplish.

#### ***1.3.1 Problem Identification***

The first problem encountered was to pick what kind of USB camera that will be used in this project. The camera should be handy, low cost and can deliver adequate picture quality. The best candidate for the camera is a WebCam which have USB interface, can deliver a decent level of picture quality for monitoring purpose and also the price is not expensive.

The best platform for the software should be the most common platform. In UTP, Windows XP is used as the default platform. So, it is decided that the application created must be able to run on any Windows XP system. The main working environment in this project is developing the interfacing software using the most appropriate programming language.

Visual Basic 6.0 seems the best candidate since it can provide enough function for the application and the level of difficulty is lower than other programming language such as Visual C++. Hence, a thorough study of Visual Basic programming is necessary to complete this project.

#### ***1.3.2 Significance of the Project***

Upon completion, the USB security camera software should be able to capture live video feed and process it for motion detection and streams it to the computer monitor.

## 1.4 Objective and Scope of Study

The foremost objective of the project is to design user-friendly USB camera security software to be used by a wide range of people. An application that recognizes multiple USB cameras connected to a single computer and streams each camera's output simultaneously or alternately to a computer's monitor.

For this project, learning the coding technique on how to implement USB technology into the software is crucial. The only programming language involved is high level programming language that is Visual Basic 6.0 .

Basically, the objectives of this project are

- To develop a software using Visual Basic 6.0 that recognizes multiple USB cameras connected to a single computer streams each camera's output simultaneously or alternately to a computer's monitor.
- To define the features of the application

The scopes of study are

- Study existing USB security camera solutions in the market
- Investigate the features needed by the cameras suitable for this project
- Learn Visual Basic 6.0 programming language
- Integrate the application within Windows Operating System
- Camera's implementation to the system

## **CHAPTER 2**

### **LITERATURE REVIEW/ THEORY**

#### **2.1 Existing security camera system in the market**

There are many types of security camera existing in the current market. They can be distinguished in five categories.

##### ***2.1.1 Introduction***

Security Cameras can be classified as both interior and exterior. Obviously exterior cameras provide some self protection against the damaging elements, and usually include a metal visor of sorts that mounts above the camera lens keeping inclement weather from obstructing the view. Interior security cameras are just what they sound like; units designed to only be used indoors within a certain temperature range and without coming into contact with water or the damaging elements.

Security camera can be divided into five types, there are:

- i. Home security camera
- ii. Wireless security camera
- iii. CCTV security camera
- iv. Internet security camera
- v. Fake security camera

Each type has different features and level of efficiency.

##### ***2.1.2 Home Security Camera***

As an addition to Home Alarm System, an observation system can be beneficial to protect things that are important to any properties. Many people today have observation systems set up in homes to protect family members and valuable possessions, and even in businesses to monitor and protect employees and equipment. There are a number of choices and features available for many situations.

### ***2.1.3 Wireless Security Camera***

Two other types of security cameras, wireless and hidden are becoming all the rage today. Because the manufacturing cost has come down with these cameras and demand has gone up, companies are now able to market these security cameras for surveillance to a large range of the consumer market. Now, every home that wants one can afford a security camera or two.

Wireless security cameras are popular because they allow the ultimate in flexibility of placement. Free from the limitations induced when hardwired models are used, wireless cameras can be used quite creatively in certain applications. The only real caveat of wireless security cameras is that they often utilize a power supply source that needs to be recharged every few hours. For this reason alone, many folks stay away from this type of application.

Hidden cameras are certainly the most popular. These types of security camera can be purchased for very low costs and implemented quite easily in the home or business. Many folks choose to install these hidden cameras to monitor suspect baby or pet sitters, cleaning ladies and maids or other risks to your private valuables, property or loved ones.

### ***2.1.4 CCTV Security Camera***

Closed Circuit Television (CCTV) is a visual surveillance technology with the capability to monitor a variety of environments and activities. A communication link that is generally used between the CCTV security camera and monitor and can include full pan, tilt and zoom functionality. A CCTV security camera can be remotely operated from a control room. Security can be greatly improved by using CCTV. A security camera will detect an intruder when the viewer senses movement, by verifying if an alarm signal is real, and by monitoring constant surveillance areas.

### ***2.1.5 Internet Security Camera***

With Internet Security camera, one can monitor his home from any location in the world. By using Internet Security Camera IP Web Server, one can log in to his IP address and take control of the camera. What is needed is a broadband connection, a pc with USB camera and the Internet Security Camera IP Web Server.



### ***2.1.6 Fake Security Camera***

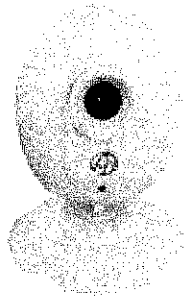
Dummy security cameras are an excellent way to deter thieves at a fraction of the cost of normal surveillance systems. A fake security camera however, should be chosen with care, as some of them are easily recognizable as fakes.

The placement and type of fake security camera should be considered carefully, because there are different types for indoor and outdoor use. The fake surveillance camera on the right is a perfect example of one that is suitable for outdoor use, but there are also other cameras available, such as dome types for indoor use.

## **2.2 USB Camera Candidates**

Before proceeding to pick the best camera for this project, several options are analyzed.

### ***2.2.1 Pakatak U350 PC USB Camera [1]***



Features :

Size : 90x70x70mm

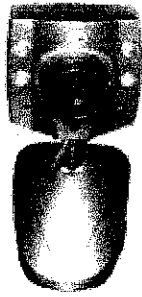
Picture Quality : 640x480 - 35K pixels

System : PAL/NTSC software regulated

Interface : USB plug-and-play support

Software : Video conferencing, digital picture editing, motion detection & alarm feature.

### **2.2.2 Pakatak U703 PC USB Night Vision Camera [1]**



Features :

Size : 90x40x40mm

Picture Quality : 640x480 - 35K pixels

System : PAL/NTSC software regulated

Interface : USB plug-and-play support

Software : Video conferencing, digital picture editing, motion detection & alarm feature.

### **2.2.3 Mototech Webcam**

Brand : Mototech

Sensor size : 1/3"

Sensor type : Color CMOS image sensor

Resolution : 640x480(300k pixels)

Frame rate : Up to 30 Frames/sec (CIF)

S/N Ratio : 42dB

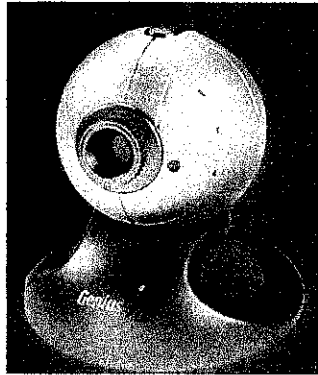
Video Information Format : 16, 24-bit RGB

Focal length : 6.00mm

Lens View Angle : +/-28

Focus Limit : 3cm-infinity

### 2.2.4 *Genius VideoCAM Web V4 [6]*



Features :

Size : 70x50x50mm

Picture Quality : 640x480 - 100K pixels

System : PAL/NTSC software regulated

Interface : USB plug-and-play support

Software : Video conferencing, digital picture editing, motion detection & alarm feature.

#### *Hardware Requirement*

- IBM PC compatible computer with Pentium II400 processor or higher
- Minimum 64MB DRAM
- Minimum 100MB hard disk drive space
- Available USB port
- Microsoft Windows XP, Me, 2000, 98SE
- CD-ROM driver for software installation
- VGA display graphics card
- Sound card with microphone and speakers for audio output

### 2.3 **Selected USB Camera**

After thorough comparison on those four candidates, finally the winning candidate for the first camera is the Genius VideoCAM Web V4. Priced at RM 75.00, it is a suitable candidate given the specification. The video resolution of the camera which is 640x480 pixels is adequate for this project purpose. The second camera selected to be used in this project is Mototech Webcam which have 640x480(300k pixels) resolution capability which is enough to capture a clear picture. The price is RM 65.00 .

## 2.4 Existing Security Camera Software

There are lots of security camera software's in the market now. Below are some of the interesting ones.

### 2.4.1 *Crime Catcher 3.0*

*Company:* Edward Torkington

*Price:* RM 150

*Size:* 4.16 Mb

*Release:* 2004-02-15

*Platform:* Win95, Win98, WinME, WinNT 4.x, WinXP, Windows2000

*Description:* (Quoted from the developer) [3]

Crime Catcher is software for webcam that allows any computer to act as a security system. The Professional Edition supports up to 4 webcams so security can be maximized. The webcam software will monitor for movement or motion. If motion is detected pictures will be taken of what the webcam can see. Other actions can take place such as sounds playing, email notification, or launch any external program. More features include: Advanced motion detection; Every single pixel being monitored for change. Hide Crime Catcher completely from view. Can appear in the system tray, taskbar, both, or neither - completely hidden from view. Email notifications with attachments Have emails of what the webcam has detected. FTP images to an FTP server Upload images to an FTP server for later retrieval. Webpage mode - image uploaded every x seconds. Put the webcam online. Have an image uploaded every x seconds. Remote surveillance Remotely login to the webcam and see live images in realtime. Mask out areas which could cause false alarms; Simply paint out areas which do not want to be monitored. Produce reports on captured images; Reports can be produced on captured images in HTML. Write images to an AVI movie file; Easily combine captured images into a movie! Play sounds when motion is detected; Have a dog bark or another sound play to warn/welcome an intruder/guest. Launch any program when motion is detected; Have any program run when motion is detected. This could be a cd player, image, word document. Password capture; make it difficult for an intruder by requiring a password be entered to stop the surveillance. Fully configurable and very easy to use. Crime Catcher truly is a simple but powerful security webcam system.

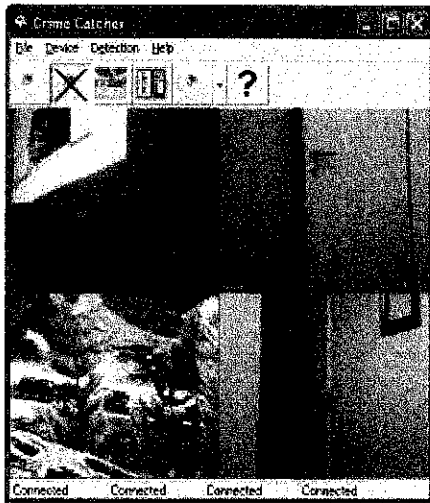


Figure 1 Screenshot of Crime Catcher 3.0

### 2.4.2 NET Video Spy 1.82

Company: SARBASH Lab.

Price: RM 180

Size: 5.65 Mb

Release: 2004-01-06

Platform: Win95, Win98, WinME, WinNT 4.x, WinXP, Windows2000

*Description: (Quoted from the developer) [4]*

NET Video Spy - is a video-surveillance system that allows users to monitor remote locations using LAN (local area network) or wireless network or Internet. NET Video Spy utilizes the power of the webcam or other video capture device and/or microphone to allow creation of video-surveillance system. The idea behind this software is about using hardware already available at disposal for creating cost-effective video-surveillance solutions.

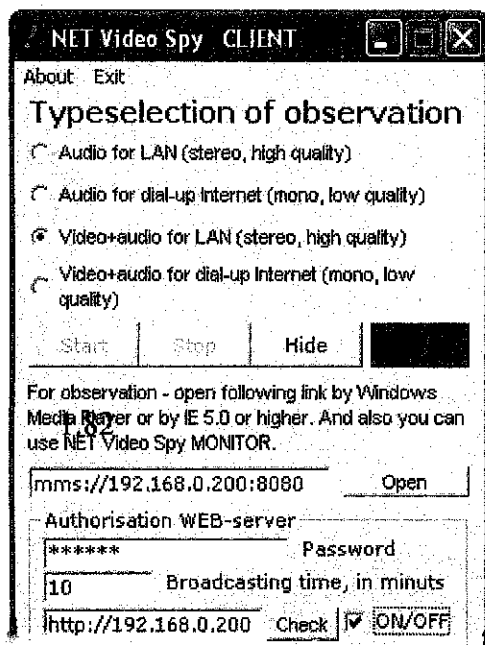


Figure 2 Screenshot of NET Video Spy

### 2.4.3 *Honestech Video Security Software [2]*

Company: Honest Technology International

Price: RM 75

Size: 8.26 Mb

Release: 2004-03-08

Platform: Win95, Win98, WinME, WinNT 4.x, WinXP, Windows2000

#### ***Product Features***

- Intelligent motion detection feature activates digital camcorder to capture full motion video.
- Creates high quality MPEG videos in real-time using powerful honestech MPEG Encoding Engine.
- Automatic E-mail routing of captured video.
- MPEG format reduces size of video files making them smaller when saved or transmitted across the internet
- Automatically files recorded videos for future review.
- Scheduling function allows security monitoring possible from any PC.
- "Motion Sense Test" function allows sensitivity calibration before monitoring is activated.
- Listing of recorded videos is flexible making it easier to manage video files.

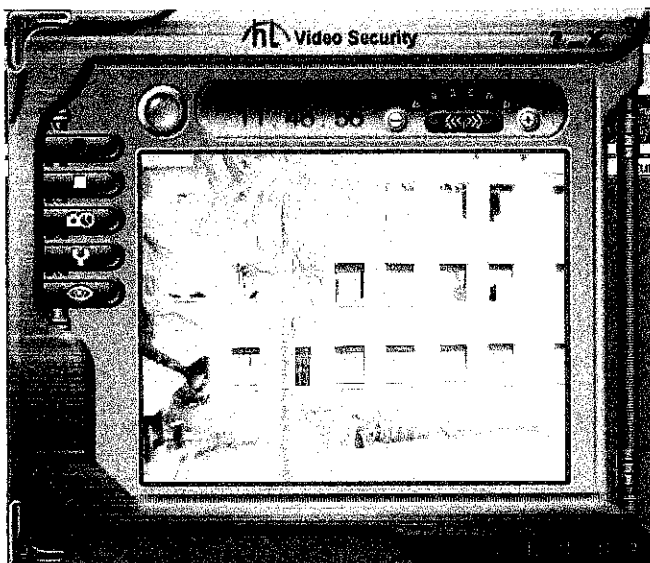


Figure 3 Screenshot of Honestech Video Security Software

## 2.5 Visual Basic 6.0

Microsoft Visual Basic is the most popular choice to create software's for Windows platform that evolves from the QBASIC programming language. In Visual Basic, new windows created are called form. Elements as text boxes and button that placed in the form are called control. Visual Basic allows event-driven programming where the user actions cause events, and such event in turn triggers a procedure that is associated with it.

Although version 6.0 is not the latest in its family as now the popular visual basic .net has taken over the visual basic 6.0 with more advancement and features. This version is chosen because it still can perform the function needed in this project and it is easier to learn.

Visual Basic has evolved from the simplest programming language for Microsoft Windows to an exceedingly complex development environment, capable of delivering virtually anything from tiny utilities to huge n-tier client/server applications.

Visual Basic 6 includes many new features, compared to the previous version, especially in the database and Internet areas. Among these are ADO, DHTML applications, and WebClasses, just to mention the outstanding ones.

### 2.5.1 *The Structure of Microsoft Visual Basic*

An application is really nothing more than a set of instructions directing the computer to perform a task or tasks. The structure of an application is the way in which the instructions are organized; that is, where the instructions are stored and the order in which instructions are executed. As applications become more complex, the need for organization or structure becomes obvious. In addition to controlling the execution of an application, the structure is important to the programmer.

As Visual Basic application is based on objects, the structure of its code closely models its physical representation on screen. By definition, objects contain data and code. The form that can be seen on screen is a representation of the properties that define its appearance and intrinsic behavior. For each form in an application, there is a related *form module* (with file name extension .frm) that contains its code.

Each form module contains *event procedures* — sections of code where the instruction is placed that will execute in response to specific events. Forms can contain controls. For each control on a form, there is a corresponding set of event procedures in the form module. In addition to event procedures, form modules can contain general procedures that are executed in response to a call from any event procedure.

Code that isn't related to a specific form or control can be placed in a different type of module, a *standard module* (.BAS). A procedure that might be used in response to events in several different objects should be placed in a standard module, rather than duplicating the code in the event procedures for each object.

A *class module* (.CLS) is used to create objects that can be called from procedures within your application. Whereas a standard module contains only code, a class module contains both code and data.

The following objects describe the different types of files and objects that user can include in a project.

- **Form Modules**

Form modules (.frm file name extension) can contain textual descriptions of the form and its controls, including their property settings. They can also contain form-level declarations of constants, variables, and external procedures; event procedures; and general procedures.

- **Class Modules**

Class modules (.cls file name extension) are similar to form modules, except that they have no visible user interface. You can use class modules to create your own objects, including code for methods and properties.

- **Standard Modules**

Standard modules (.bas file name extension) can contain public or module-level declarations of types, constants, variables, external procedures, and public procedures.

- **Resource Files**

Resource files (.res file name extension) contain bitmaps, text strings, and other data that user can change without having to re-edit your code. For example, if user plans to localize your application in a foreign language, he can keep all of the user-interface text strings and bitmaps in a resource file, which he can then localize instead of the entire application. A project can contain no more than one resource file.



- **ActiveX Documents**

ActiveX documents (.dob) are similar to forms, but are displayable in an Internet browser such as Internet Explorer. The Professional and Enterprise editions of Visual Basic are capable of creating ActiveX documents.

- **User Control and Property Page Modules**

User Control (.ctl) and Property Page (.pag) modules are also similar to forms, but are used to create ActiveX controls and their associated property pages for displaying design-time properties. The Professional and Enterprise editions of Visual Basic are capable of creating ActiveX controls.

In addition to files and modules, several other types of components can be added to the project.

- **ActiveX Controls**

ActiveX controls (.ocx file name extension) are optional controls which can be added to the toolbox and used on forms.

- **Insertable Objects**

*Insertable objects*, such as a Microsoft Excel Worksheet object, are components you can use as building blocks to build integrated solutions. An *integrated solution* can contain data in different formats, such as spreadsheets, bitmaps, and text, which were all created by different applications.

- **References**

User can also add references to external ActiveX components that may be used by application. User assigns references by using the References dialog, accessed from the References menu item on the Project menu.

- **ActiveX Designers**

ActiveX designers are tools for designing classes from which objects can be created. The design interface for forms is the default designer. Additional designers may be available from other sources.

▪ **Standard Controls**

Standard controls are supplied by Visual Basic. Standard controls, such as the command button or frame control, are always included in the toolbox, unlike ActiveX controls and insertable objects, which can be removed from or added to the toolbox.

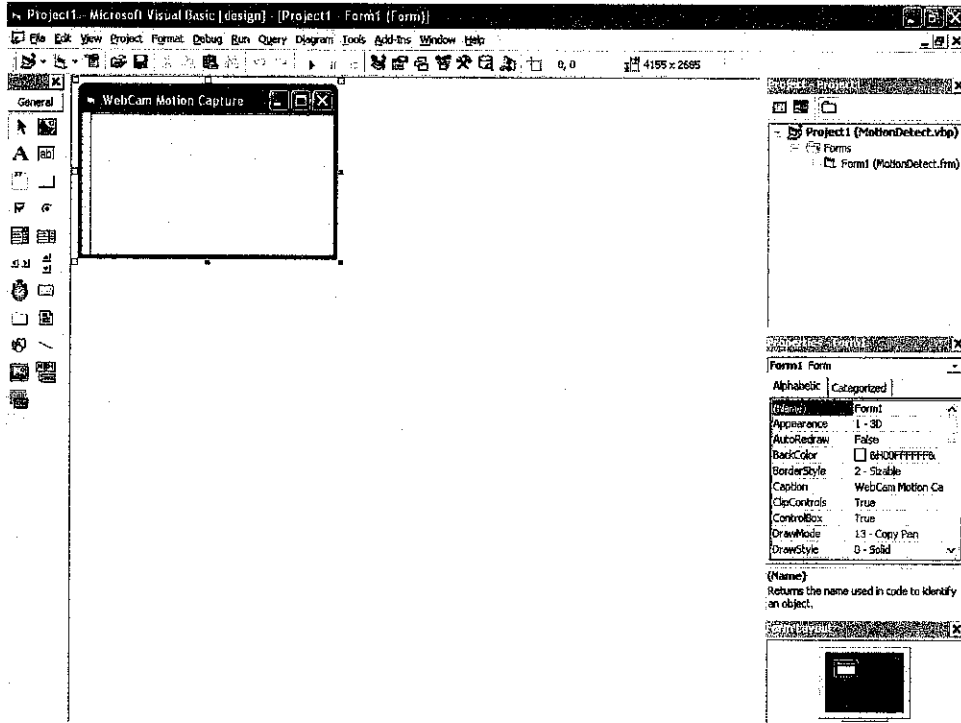


Figure 4 Visual Basic Editor

Form	frm
Command Button	cmd
Text Box	txt
Label	lbl
Option Button	opt
CheckBox	chk
Frame	fra
Horizontal Scrollbar	hsb
Vertical Scrollbar	vsb
Image	img
Picture Box	pic
Combo Box	cbo
List Box	lst
Shape	shp

Table 1 Visual Basic Naming Convention

## CHAPTER 3

### METHODOLOGY/ PROJECT WORK

#### 3.1 Procedure Identification

In ensuring the smoothness in the project execution, all the required procedures are identified. Figure 5 outlines the main approaches in accomplishing the project. All the procedures are carried out subsequently at all times. When it is necessary, the previous procedure is revised as to do any modification on the project.

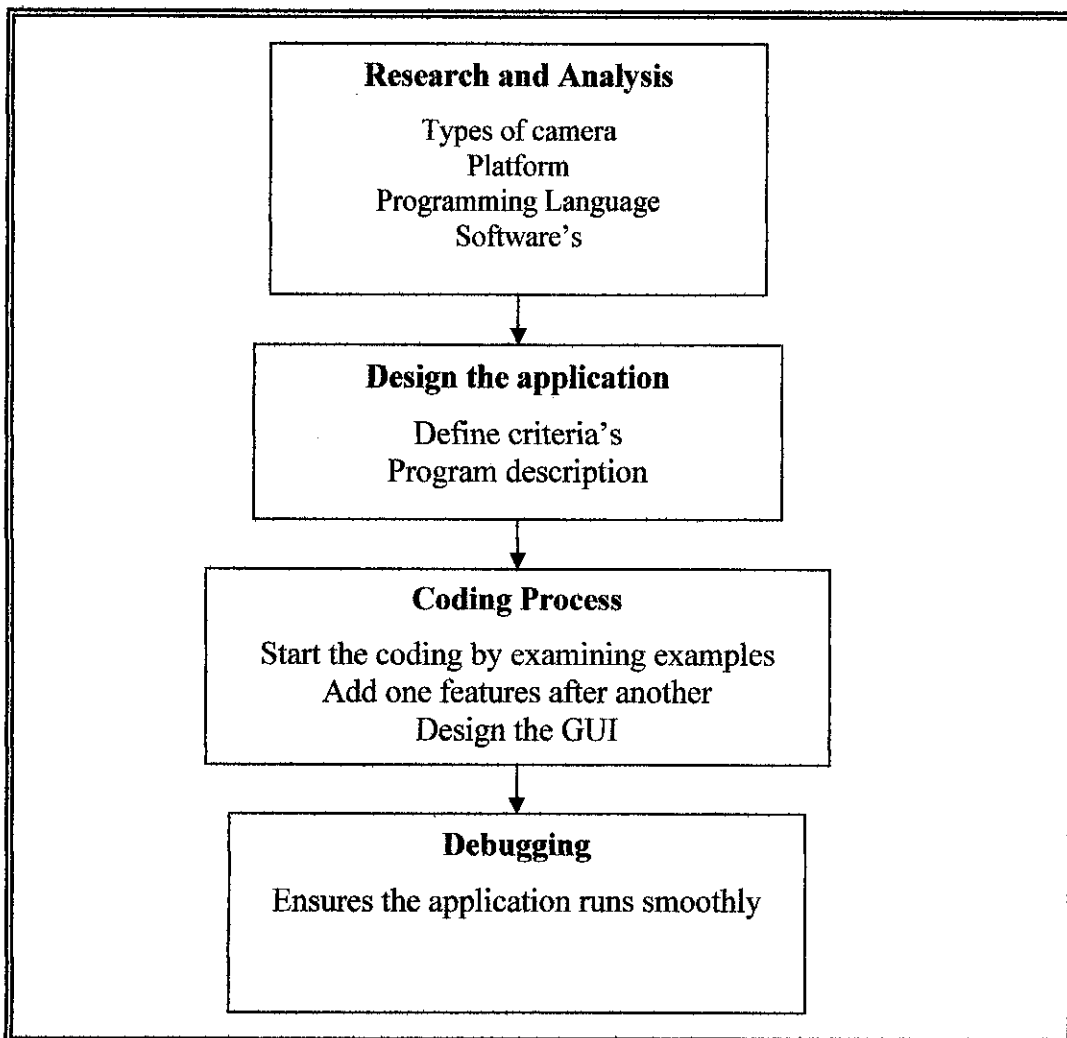


Figure 5 Methodology diagram

### ***3.1.1 Research and Analysis***

Research is the prerequisite in the early stage of the project as to pursue on the next three procedures of the project. Through the Internet and several references, the basics of USB camera and Visual Basic coding is investigated.

### ***3.1.2 Design the application***

Before starting the coding work, all the required criteria's must be defined first. How the application should work should be described first.

### ***3.1.3 Coding Process***

One good way to learn about coding is to learn by examples. A lot of examples were studied that is useful for learning VB coding. The coding work also should concentrate on one section at a time to avoid making mistakes.

### ***3.1.4 Debugging***

It is almost impossible to develop a successful large application at one try. A lot of debugging work must be involved in any case. Debugging means making the code run flawlessly.

## **3.2 Tools Required**

The tools required can be divided into two types, hardware and software.

### ***3.2.1 Hardware : Genius VideoCAM Web V4 & Mototech Webcam***

This camera was chosen because it is low cost and have adequate features needed for this project.

### ***3.2.2 Hardware : PC with Windows XP Operating System***

A PC with Windows XP was chosen as the main environment for the application because it is commonly used and easy to handle.

### ***3.2.3 Software : Visual Basic 6.0 Enterprise Edition[5]***

The Enterprise edition allows professionals to create robust distributed applications in a team setting. It includes all the features of the Professional edition, plus Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, and more. Printed documentation provided with the Enterprise edition includes the Visual Studio Enterprise Features book plus Microsoft Developer Network CDs containing full online documentation.

## CHAPTER 4

### RESULT AND DISCUSSION

#### 4.1 Program Description

Towards the end of two semesters, the application is finished. The main feature of the application is it can support any USB cameras connected to the PC and the user can select manually any camera that want to be activated. Other great features that the application has are, it can detect motion and snap pictures when there is a motion detected from the camera.

#### 4.2 Coding Work

The coding work includes defining the Graphic User Interface (GUI) design, variables declaration, Application Programming Interface (API) functions, and doing the actual coding. The Visual Basic codes for the software is showed in Appendix A.

##### 4.2.1 Graphic User Interface design

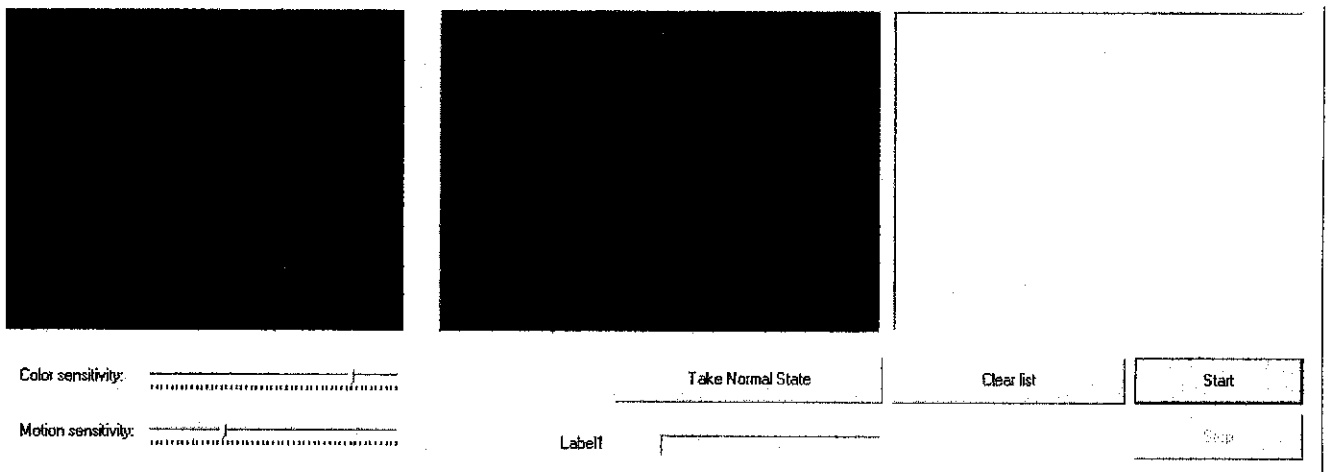


Figure 6 GUI for the application

The “PictureBox” will display the captured video from the default camera. Then the code will analyze the picture for any movement. If any movement detected then the movements will be marked as red in the video feed. Movement count then will be displayed in the label box at the bottom of the GUI. Color and Motion sensitivity can be adjusted using the slider at the left lower end of the GUI. Screenshot of the software running is showed in Appendix B.

#### 4.2.2 *Variables declaration*

There are mainly four types of variables declaration used. There are :

##### i) Global Variables

In Visual Basic jargon, global variable are those variable declared using the *Public* keyword in BAS modules. Conceptually, these variable are the simplest of the group because they survive for the life of the application and their scope is the entire application. (In other words, they can be read and modified from anywhere in the current program.)

The following code snippet shows the declaration of a global variable:

##### ii) Module Level Variables

These kinds of variables are visible only from within the module they belong to and can't be accessed from the outside. In general, these variables are useful for sharing data among procedures in the same module:

##### iii) Dynamic Local Variables

Dynamic local variables are defined within a procedure; their scope is the procedure itself, and their lifetime coincides with that of the procedure.

##### iv) Static Local Variables

Static local variables are a hybrid because they have the scope of local variables and the lifetime of module-level variables. Their value is preserved between calls to the procedure they belong to until their module is unloaded (or until the application ends, as is the case for procedures inside standard modules). These variables are declared inside a procedure using the *Static* keyword.

### 4.2.3 API Function

The Windows operating system is heavily based on messages. For example, when the user closes a window, the operating system sends the window a WM\_CLOSE message. When the user types a key, the window that has the focus receives a WM\_CHAR message, and so on. (In this context, the term *window* refers to both top-level windows and child controls.) Messages can also be sent to a window or a control to affect its appearance or behavior or to retrieve the information it contains. For example, WM\_SETTEXT can be send as message to most windows and controls to assign a string to their contents, also the WM\_GETTEXT message to read their current contents. By means of these messages you can set or read the caption of a top-level window or set or read the *Text* property of a TextBox control, just to name a few common uses for this technique.

Before API function can be used, Visual Basic must be told the name of the DLL (Dynamic Link Library) that contains it and the type of each argument. This can be done with a *Declare* statement, which must appear in the declaration section of a module. *Declare* statements must be declared as Private in all types of modules except BAS modules (which also accept Public *Declare* statements that are visible from the entire application).

The main API function that is used in this project is *SendMessage*, whose *Declare* statement is this:

```
"Private Declare Function SendMessage Lib "USER32" Alias "SendMessageA"  
(ByVal hwnd As Long, ByVal wParam As Long, ByVal wMsg As Long, lParam As Any) As Long"
```

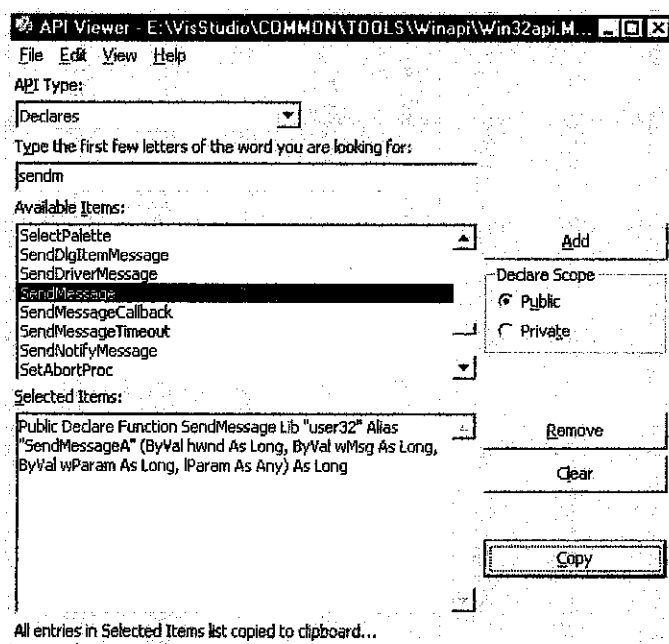


Figure 7 API Viewer



### 4.3 Motion Detection Code

The main part of the program is the motion detection code which is enclosed in Appendix A. The concept of the motion detection code is rather simple. Let say the screen resolution is 640x480 pixels, so there are total of 307200 pixels on the screen. To detect motion, every pixel on the screen is compared with itself. If any changes detected then the pixel will be marked red. To detect real movement, which is slightly a lot more motion, 1 pixel along with 4 other pixels around it is compared for changes. If there are changes in all 5 pixels then it will be marked green, indicating a real movement has occurred.

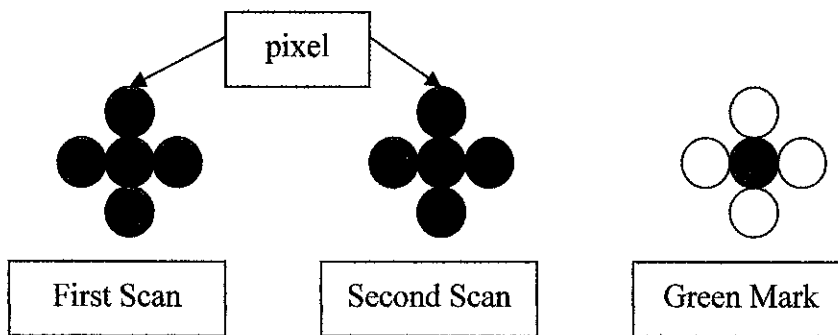


Figure 8 Real movement detection diagram

For example, in figure 8, first scan indicate that the pixel is red along with the 4 other pixel around it, while the second scan is blue, both scan is compared and it is not the same which means something is moving. To indicate real movement, the code simply marked the middle pixel as green.

### 4.4 Discussion and Problem Encountered

Naturally, it is almost inevitable to avoid problems and complications when it comes to developing a system from scratch. It can be said that the time spent on writing codes is almost equal to the time spent in the debugging phases. Most of the time spent is on finding the right syntax and choosing the right declaration type for the variables in coding work. It is hard to find the legal copy of Visual Basic as it is so expensive just to obtain the license, therefore the MSDN collection can't be used which contained all the help files needed to give more guidance during the coding process. The MSDN collection is quite important to guide any programmer to the right way.

#### 4.4.1 Windows XP cannot detect two cameras with identical WDM drivers

When two same camera using the same drivers installed in one machine, they tends to conflict with each other, making only one camera can be use at one time. This has become a big problem as two cameras that have identical drivers are used. Windows XP kept issuing 'code 38' error which indicate that the same instance of driver already exist in the memory so it cannot load the driver for the second camera.

After thorough investigation and experimentation, the problem is solved. The solution to this problem is to manually edit the setup information file of one of the webcam driver's by renaming all the system files and also the 'dynamic link library' files in the driver's package to something else. By doing so, both drivers can be avoided to have identical files thus the same two drivers can be uploaded into the memory with different names to drive two same cameras.

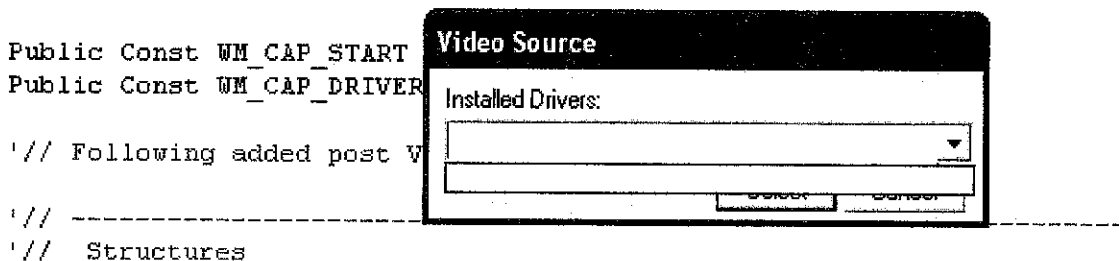


Figure 9 Visual Basic form showing empty scroll box

The empty scroll box meaning that the application failed to detect installed drivers in the machine.

#### 4.4.2 The software failed to detect more than one camera

The initial objective of this project was to develop an application that can simultaneously feed video streams from at least two cameras. After much trial and error, the software still failed to detect two cameras simultaneously due to coding difficulties. In spite of that, the software can be considered successful security software as it has adequate features to be considered as one. There are much to be learn in visual basic coding techniques.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

#### **5.1 Conclusion**

The four main procedures identified previously have been carried out during the two semesters. The research and analysis have been continuously conducted as to gain knowledge on the programming part. Although the motion detection and picture snap features of the software is not compulsory, it is really appropriate to be included in any USB Security Software. The main objective of the project was to develop software that can simultaneously feed video streams from at least two cameras onto one screen, but due coding difficulties, the project failed to achieve the main objective. But still the software can support any USB camera and the user can manually select any cameras connected to the PC, plus it have motion detector and automatic picture snap, so the software can be considered a successful USB security software. With a nice GUI, the software is easy to operate and understand but there a lot of other professional security software in the market that have a lot more features. With Visual Basic 6.0 used to create the software, there will be no compatibility issues if the software is used in Windows platform.

## 5.2 Recommendation

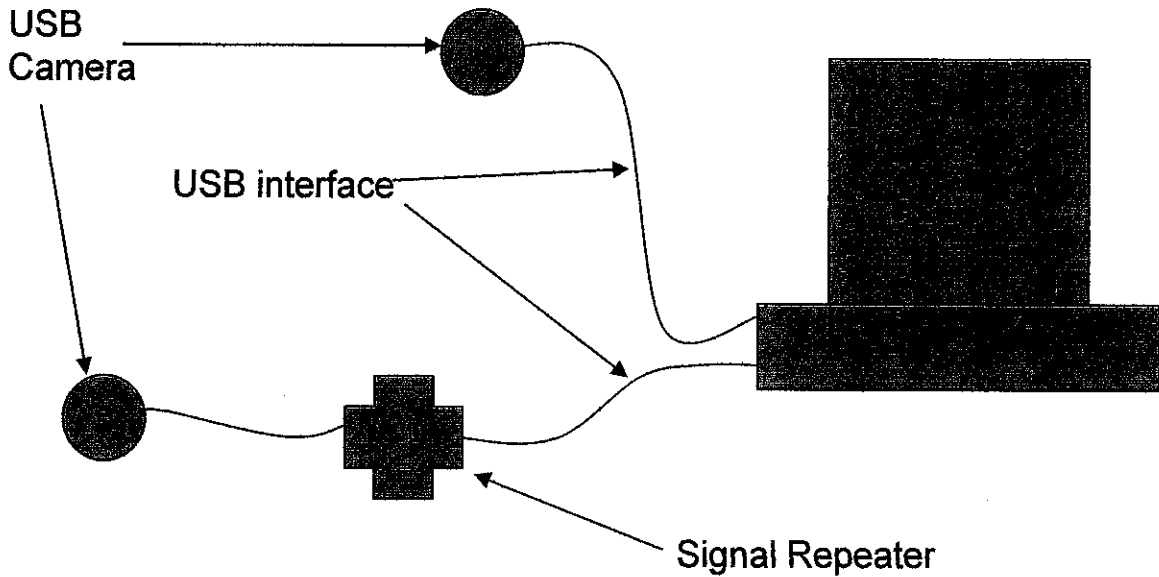


Figure 10 Overview layout “USB Security Camera”

USB have limited length of the connection wire. What use if too many cameras covering very limited angle from only limited spots. It would be very pleasing if the cameras can be situated at a farther distance, one at a corner and the other at the other corner of a room. To accomplish this, some kind of a signal repeater must be use to make the connection longer. Currently in the market, there is such product that performs the function called the ‘USB Xtended’. It can elongates the connection link to a greater length.

## REFERENCES

- [1] <<http://www.pakatak.co.uk/cactushop5/default.asp>>
- [2] <<http://www.honestech.com>>
- [3] <<http://www.123cctv.com/cctv/remote.html>>
- [4] <<http://gspy.sourceforge.net>>
- [5] <<http://www.vb2themax.com>>
- [6] <<http://www.genius.com>>

## **APPENDICES**

## APPENDIX A

### VISUAL BASIC CODES

#### *Main Form Codes*

#### *Start*

---

'FOR WEBCAM DECLARATIONS

```
Private Declare Function SendMessage Lib "USER32" Alias "SendMessageA" (ByVal  
hwnd As Long, ByVal wParam As Long, ByVal lParam As Any) As  
Long
```

```
Private Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias  
"capCreateCaptureWindowA" (ByVal lpszWindowName As String, ByVal dwStyle As  
Long, ByVal X As Long, ByVal Y As Long, ByVal nWidth As Long, ByVal nHeight As  
Long, ByVal hwndParent As Long, ByVal nID As Long) As Long
```

```
Private mCapHwnd As Long
```

```
Private Const CONNECT As Long = 1034
```

```
Private Const DISCONNECT As Long = 1035
```

```
Private Const GET_FRAME As Long = 1084
```

```
Private Const COPY As Long = 1054
```

```
'declarations
```

```
Dim P() As Long
```

```
Dim POn() As Boolean
```

```
Dim inten As Integer
```

```
Dim i As Integer, j As Integer
```

Dim Ri As Long, Wo As Long

Dim RealRi As Long

Dim c As Long, c2 As Long

Dim R As Integer, G As Integer, B As Integer

Dim R2 As Integer, G2 As Integer, B2 As Integer

Dim Tppx As Single, Tppy As Single

Dim Tolerance As Integer

Dim RealMov As Integer

Dim Counter As Integer

Private Declare Function GetTickCount Lib "kernel32" () As Long

Dim LastTime As Long

Option Explicit

Private Sub Form\_Load()

'set up the visual stuff

Picture1.Width = 640 \* Screen.TwipsPerPixelX

Picture1.Height = 480 \* Screen.TwipsPerPixelY

'Inten is the measure of how many pixels are going to be recognized.

inten = 15

'The tolerance of recognizing the pixel change

Tolerance = 20

Tppx = Screen.TwipsPerPixelX

Tppy = Screen.TwipsPerPixelY



ReDim POn(640 / inten, 480 / inten)

ReDim P(640 / inten, 480 / inten)

STARTCAM

End Sub

Private Sub Picture1\_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

If Button = 1 Then

STARTCAM

Elseif Button = 2 Then

STOPCAM

End If

End Sub

Private Sub Timer1\_Timer()

'Get the picture from camera.. the main part

SendMessage mCapHwnd, GET\_FRAME, 0, 0

SendMessage mCapHwnd, COPY, 0, 0

Picture1.Picture = Clipboard.GetData

Clipboard.Clear

Ri = 0 'right

Wo = 0 'wrong

LastTime = GetTickCount

For i = 0 To 640 / inten - 1

For j = 0 To 480 / inten - 1

'get a point

c = Picture1.Point(i \* inten \* Tppx, j \* inten \* Tppy)

'analyze it, Red, Green, Blue

R = c Mod 256

G = (c \ 256) Mod 256

B = (c \ 256 \ 256) Mod 256

'recall what the point was one step before this

c2 = P(i, j)

'analyze it

R2 = c2 Mod 256

G2 = (c2 \ 256) Mod 256

B2 = (c2 \ 256 \ 256) Mod 256

If Abs(R - R2) < Tolerance And Abs(G - G2) < Tolerance And Abs(B - B2) < Tolerance Then

'pixel remained same

Ri = Ri + 1

POn(i, j) = True

Else

'Pixel changed

Wo = Wo + 1

'make a red dor

P(i, j) = Picture1.Point(i \* inten \* Tppx, j \* inten \* Tppy)

Picture1.PSet (i \* inten \* Tppx, j \* inten \* Tppy), vbRed

```

POn(i, j) = False
End If
    Next j
Next i
RealRi = 0
For i = 1 To 640 / inten - 2
    For j = 1 To 480 / inten - 2
        If POn(i, j) = False Then
            'movement
            If POn(i, j + 1) = False Then
                If POn(i, j - 1) = False Then
                    If POn(i + 1, j) = False Then
                        If POn(i - 1, j) = False Then
                            RealRi = RealRi + 1
                            Picture1.PSet (i * inten * Tppx, j * inten * Tppy), vbGreen
                        End If
                    End If
                End If
            End If
        End If
    Next j
Next i
Label1.Caption = Int(Wo / (Ri + Wo) * 100) & " % movement" & vbCrLf & "Real
Movement: " & RealRi & vbCrLf _
& "Completed in: " & GetTickCount - LastTime

End Sub

```

```
Sub STOPCAM()
```

```
DoEvents: SendMessage mCapHwnd, DISCONNECT, 0, 0
```

```
Timer1.Enabled = False
```

```
End Sub
```

```
Sub STARTCAM()
```

```
mCapHwnd = capCreateCaptureWindow("WebcamCapture", 0, 0, 0, 640, 480, Me.hwnd,  
0)
```

```
DoEvents
```

```
SendMessage mCapHwnd, CONNECT, 0, 0
```

```
Timer1.Enabled = True
```

```
End Sub
```

```
'Private Sub Timer2_Timer()
```

```
'SavePicture Picture1.Image, "C:\pics\img" & Counter & ".bmp"
```

```
'Counter = Counter + 1
```

```
'End Sub
```

---

```
END
```

---

```
Modul.bas
```

```
// -----
```

```
// Windows API Constants / Types / Declarations
```

```
// -----
```

```
Public Const WS_BORDER = &H800000
```

---

```

Public Const WS_CAPTION = &HC00000
Public Const WS_SYSMENU = &H80000
Public Const WS_CHILD = &H40000000
Public Const WS_VISIBLE = &H10000000
Public Const WS_OVERLAPPED = &H0&
Public Const WS_MINIMIZEBOX = &H20000
Public Const WS_MAXIMIZEBOX = &H10000
Public Const WS_THICKFRAME = &H40000
Public Const WS_OVERLAPPEDWINDOW = (WS_OVERLAPPED Or WS_CAPTION
Or WS_SYSMENU Or WS_THICKFRAME Or WS_MINIMIZEBOX Or
WS_MAXIMIZEBOX)
Public Const SWP_NOMOVE = &H2
Public Const SWP_NOSIZE = 1
Public Const SWP_NOZORDER = &H4
Public Const HWND_BOTTOM = 1
Public Const HWND_TOPMOST = -1
Public Const HWND_NOTOPMOST = -2
Public Const SM_CYCAPTION = 4
Public Const SM_CXFRAME = 32
Public Const SM_CYFRAME = 33
Public Const WS_EX_TRANSPARENT = &H20&
Public Const GWL_STYLE = (-16)
Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd
As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

// Memory manipulation
Declare Function lStrCpy Lib "kernel32" Alias "lstrcpyA" (ByVal lpString1 As Long,
ByVal lpString2 As Long) As Long
Declare Function lStrCpyn Lib "kernel32" Alias "lstrcpynA" (ByVal lpString1 As Any,
ByVal lpString2 As Long, ByVal iMaxLength As Long) As Long
Declare Sub RtlMoveMemory Lib "kernel32" (ByVal hpvDest As Long, ByVal
hpvSource As Long, ByVal cbCopy As Long)
Declare Sub hmemcpy Lib "kernel32" (hpvDest As Any, hpvSource As Any, ByVal
cbCopy As Long)

```

---

```

// Window manipulation

Declare Function SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long,
ByVal cy As Long, ByVal wFlags As Long) As Long

Declare Function DestroyWindow Lib "user32" (ByVal hwnd As Long) As Boolean

Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long

Declare Function SetWindowText Lib "user32" Alias "SetWindowTextA" (ByVal hWnd
As Long, ByVal lpString As String) As Long

Public hWndC As Long    ' Handle to the Capture Windows

Function MyFrameCallback(ByVal hWnd As Long, ByVal lpVHdr As Long) As Long

    Debug.Print "FrameCallBack"

    Dim VideoHeader As VIDEOHDR
    Dim VideoData() As Byte

    //Fill VideoHeader with data at lpVHdr
    RtlMoveMemory VarPtr(VideoHeader), lpVHdr, Len(VideoHeader)

    // Make room for data
    ReDim VideoData(VideoHeader.dwBytesUsed)

    //Copy data into the array
    RtlMoveMemory    VarPtr(VideoData(0)),    VideoHeader.lpData,
VideoHeader.dwBytesUsed

    Debug.Print VideoHeader.dwBytesUsed
    Debug.Print VideoData

End Function

Function MyYieldCallback(hWnd As Long) As Long

```

---

```
Debug.Print "Yield"
```

```
End Function
```

```
Function MyErrorCallback(ByVal hwnd As Long, ByVal iid As Long, ByVal  
ipstrStatusText As Long) As Long
```

```
    If iid = 0 Then Exit Function
```

```
    Dim sStatusText As String
```

```
    Dim usStatusText As String
```

```
    'Convert the Pointer to a real VB String
```

```
    sStatusText = String$(255, 0) // Make room for message
```

```
    lStrCpy StrPtr(sStatusText), ipstrStatusText // Copy message into String
```

```
    sStatusText = Left$(sStatusText, InStr(sStatusText, Chr$(0)) - 1) // Only look at left of  
null
```

```
    usStatusText = StrConv(sStatusText, vbUnicode) // Convert Unicode
```

```
    LogError usStatusText, iid
```

```
End Function
```

```
Function MyStatusCallback(ByVal hwnd As Long, ByVal iid As Long, ByVal  
ipstrStatusText As Long) As Long
```

```
    If iid = 0 Then Exit Function
```

```
    Dim sStatusText As String
```

```
    Dim usStatusText As String
```

```
    // Convert the Pointer to a real VB String
```

```
    sStatusText = String$(255, 0) // Make room for message
```

```
    lStrCpy StrPtr(sStatusText), ipstrStatusText // Copy message into String
```

```

sStatusText = Left$(sStatusText, InStr(sStatusText, Chr$(0)) - 1) '// Only look at left of
null
usStatusText = StrConv(sStatusText, vbUnicode)           '// Convert Unicode

frmMain.StatusBar.SimpleText = usStatusText
Debug.Print "Status: ", usStatusText, IID

Select Case IID '

End Select

End Function

Sub ResizeCaptureWindow(ByVal hwnd As Long)

    Dim CAPSTATUS As CAPSTATUS
    Dim lCaptionHeight As Long
    Dim IX_Border As Long
    Dim IY_Border As Long

    lCaptionHeight = GetSystemMetrics(SM_CYCAPTION)
    IX_Border = GetSystemMetrics(SM_CXFRAME)
    IY_Border = GetSystemMetrics(SM_CYFRAME)

    '// Get the capture window attributes .. width and height
    If capGetStatus(hwnd, VarPtr(CAPSTATUS), Len(CAPSTATUS)) Then

        '// Resize the capture window to the capture sizes
        SetWindowPos hwnd, HWND_BOTTOM, 0, 0, _
            CAPSTATUS.uiImageWidth + (IX_Border * 2), _
            CAPSTATUS.uiImageHeight + lCaptionHeight + (IY_Border * 2), _
            SWP_NOMOVE Or SWP_NOZORDER

    End If

    Debug.Print "Resize Window."

End Sub

```



---

```
Function MyVideoStreamCallback(lwnd As Long, lpVHdr As Long) As Long
```

```
    Beep '// Replace this with your code!
```

```
End Function
```

```
Function MyWaveStreamCallback(lwnd As Long, lpVHdr As Long) As Long
```

```
    Debug.Print "WaveStream"
```

```
End Function
```

```
Sub LogError(txtError As String, IID As Long)
```

```
    frmMain.StatusBar.SimpleText = txtError
```

```
    Debug.Print "Error: ", txtError, IID
```

```
End Sub
```

---

### **Function Module for video source selection**

```
Public lwndC As Long    Public Const WM_USER = &H400
```

```
Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
```

```
    (ByVal hwnd As Long, ByVal wParam As Long, ByVal lParam As Integer, _
```

```
    ByVal lParam As Long) As Long    *
```

```
Public Const WM_CAP_START = WM_USER
```

```
Public Const WM_CAP_DRIVER_GET_CAPS = WM_CAP_START + 14
```

---

Type CAPDRIVERCAPS

wDeviceIndex As Long // Driver index in system.ini  
fHasOverlay As Long // Can device overlay?  
fHasDlgVideoSource As Long  
fHasDlgVideoFormat As Long  
fHasDlgVideoDisplay As Long  
fCaptureInitialized As Long  
fDriverSuppliesPalettes As Long  
hVideoIn As Long  
hVideoOut As Long  
hVideoExtIn As Long  
hVideoExtOut As Long

End Type

// The two functions exported by AVICap

Declare Function capGetDriverDescriptionA Lib "avicap32.dll" (  
ByVal wDriver As Integer, ByVal lpszName As String, ByVal cbName As Long,   
ByVal lpszVer As String, ByVal cbVer As Long) As Boolean

Function capDriverGetCaps(ByVal lwnd As Long, ByVal s As Long, ByVal wSize As  
Integer) As Boolean

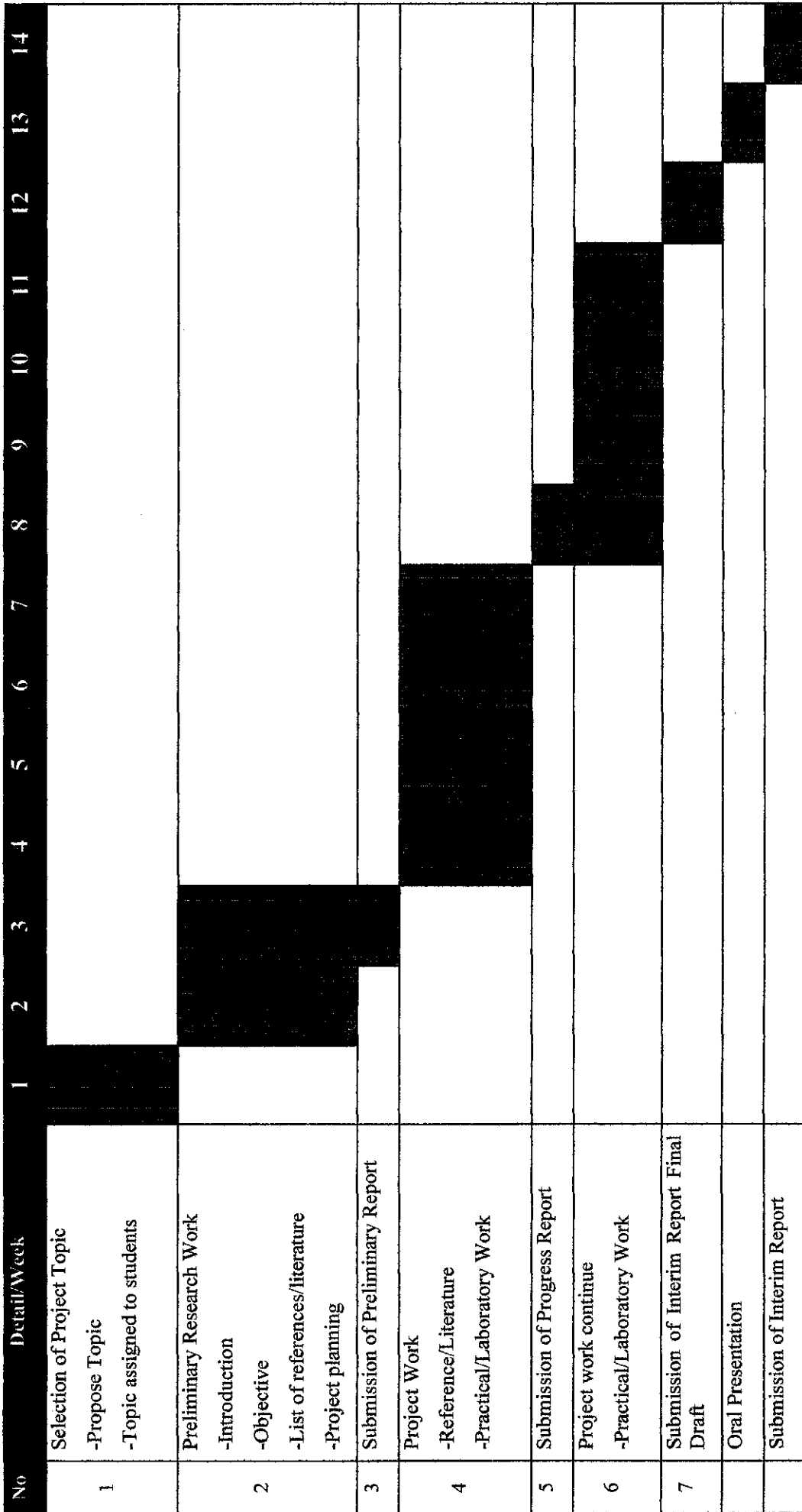
capDriverGetCaps = SendMessage(lwnd, WM\_CAP\_DRIVER\_GET\_CAPS, wSize, s)

End Function



# APPENDIX C

Gantt Chart for USB Camera Security Project in Semester 1



No	Detail/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Continuation of research -coding work -debugging														
2	Submission of Progress Report 1														
3	Submission of Preliminary Report														
4	Submission of Draft report														
5	Submission of Final Report and Technical report														

Grantt Chart for USB Camera Security Project in Semester 2