DEVELOPMENT OF MOBILE PHONE BASED SURVEILLANCE SYSTEM

by

Badrun Nahar

POST GRADUATE DIPLOMA IN INFORMATION AND COMMUNICATION TECHNOLOGY

Institute of Information and Communication Technology

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

May 2010

The project report titled "Development of Mobile Phone Based Surveillance System" submitted by Badrun Nahar, Roll No: 1008311028, Session 2008-2009, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Post Graduate Diploma (ICT) held on 23rd May 2010.

BOARD OF EXAMINERS

1. ______________________________

Dr. Md. Liakot Ali — Chairman
Associate Professor
Institute of Information and Communication Technology
BUET, Dhaka-1000

2. ______________________________

Dr. Md. Abul Kashem Mia — Member
Professor and Associate Director
Institute of Information and Communication Technology
BUET, Dhaka-1000

3. ______________________________

Dr. Md. Saiful Islam — Member
Associate Professor
Institute of Information and Communication Technology
BUET, Dhaka-1000

CANDIDATE'S DECLARATION

It is hereby declared that this project report or any part of it has not been submitted elsewhere for the award of any degree or diploma.

B. Nahar

Badrun Nahar

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
EDGE	Enhanced Data Rates for GSM Evolution
ECP	Extended Capabilities Port
EPP	Enhanced Parallel Port
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
J2ME	Java 2 Micro Edition
JMF	Java Media Framework
MMAPI	Mobile Media Application Programming Interface
PIRQ	Parallel Port Interrupt Request
RTSP	Real Time Streaming Protocol
RTP	Real-time Transport Protocol
URL	Uniform Resource Locator

ACKNOWLEDGEMENTS

First of all I would like to express my deepest gratitude to my project supervisor Dr. Md. Liakot Ali, Associate Professor, for his continuous guidance, advice, encouragement and help throughout the work. His dedication, collaboration and interaction were key factors in the success of my project. I am thankful to him for his inspiration, help and courage. Without his support and supervision it would have been very difficult for me to complete the project work.

I would like to convey my thanks to Professor Dr. S. M. Lutful Kabir, Director, IICT, BUET. His motivation and inspiration gave me the courage to do this work.

I want to convey my thanks to my uncle Engg. M. M. Jahangir Alam for giving his valuable time during buying the apparatus for the project. I also gratefully acknowledge the restless support and advice of my very good friends Muhitur Rahman Shuvo and Georgis Alam during the implementation phase of this project. My thanks also go to Mir Abu Sayem and Khaled Hossain Ansary and all the staffs of IICT, BUET. My special thanks to one of my teacher Sarwar Jahan of East West University (EWU) for supporting and giving me courage.

I would like to thank all my friends for their continuous support and inspiration throughout the period of this undertaking.

ABSTRACT

In today's world, ensuring security for important locations is a burning issue. Different surveillance methodologies such as alarm system, CCTV, PC based video system are used to ensure this security. But using all these systems, it is not possible for a person to monitor the security of his or her desired location when they are outside. Now-a-days anybody can communicate with anyone at anytime around the globe with the help of mobile phone technology. By keeping the technological facility of mobile phone in mind, a mobile phone based surveillance system has been developed in this project. This project will give a solution for the security of corporate houses as well as corporate personnel. In this project there are server and client end. A webcam is connected to the server and it will capture images of the desired location continuously. The images will be saved in a specific location of the server. Then client will start fetching images one by one from that specific location of the server by key-in the URL (Uniform Resource Locator) from their mobile phone. Client has also the option to send control instruction to the server for the movement of the webcam. Server receives the control instructions from the mobile phone and control the movement of the webcam based on the instructions. The developed system has been tested first using the GUI (emulator) designed by NetBeans IDE. It has also been tested using different mobile phones to see the images in real time. The results in both cases are as desired and prove that the proposed system functions correctly in real time. This project can also be implemented for other handheld devices like PDA.

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Security is a prime concern in our day-today life. Everyone wants to be as much secure as possible. Ensuring security for a specific place like home, offices in both personal and corporate life is a burning issue in recent time. In corporate life, the need is more. A security guard can be a physical solution but if the specific location, office or parking area can be seen and monitored from a remote place, it will be more secured. That is why different surveillance systems such as alarm system, CCTV, PC based video system etc. have been proposed [1-4]. But it's not always feasible to be physically near to the system. So, to be in touch with this sort of important systems by not being physically close, we need some sort of remote solution. Today's communication world ensures that anybody can communicate to anyone anywhere anytime across the globe with the help of mobile phone technology [5]. Various type of research works have been conducted in different time by different authors such as a video surveillance system based on Multimedia Messaging Service (MMS) [6]. This system is desired to monitor an area continuously and to capture the hazardous momentary event and to send this video image to the user as an MMS or as an e-mail immediately. But the problem with this system is that as the mobile operators do not encourage sending multimedia message to mobile phone from e-mail in fear of spam, the concept could not be verified with the mobile phones. Another research work is controlling remote system using mobile telephony [7] which introduces the mechanism of the mobile phone so that the ordinary services of the mobile phones can be leveraged to communicate with and control the remote systems. After reviewing the past researches and exploiting the communication facility of the mobile phone, a mobile phone based surveillance system has been developed in this project for giving a solution about the insecurity of corporate houses as well as corporate personnel. Although different types of camera with built-in hardware for moving it in any direction are available in the market, these are very expensive. In the proposed system a hardware circuit using stepper motor and other accessories are developed for the movement of the webcam to provide a low cost solution. The proposed system will provide the true sense of real mobility and security by accessing the desired location from the mobile phone anytime anywhere whenever wishes.

1.2 OBJECTIVES

The aim of this project is to develop a mobile phone based surveillance system through which a person can get snaps of an important location of his/her desired place at any time. To achieve the above goal, this project has following objectives:

- To develop a Java Program for capturing image sequences of the important location.
- To develop an interface through which a person can communicate with the server and access the captured images to monitor the desired location at any time.
- To develop a hardware circuit for the movement of the webcam.

1.3 PROJECT ORGANIZATION

Chapter 1 of this report describes importance of security system and their drawback. Then a new approach for ensuring security of any desired location of a person is proposed. Chapter 2 provides the detail description about the elements used in interfacing purpose for developing the proposed security system. Chapter 3 is all about project initiation and planning. Chapter 4 describes step by step system design and development of the proposed system. Chapter 5 describes the implementation, testing and results of the proposed system. In final chapter (Chapter 6) conclusion and recommendation for future scopes have been stated. The project report ends with an appendix that contains the program code for the proposed system.

CHAPTER 2
INTERFACING ELEMENTS

2.1 STEPPER MOTOR

2.1.1 Definition

A stepper motor is basically an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical pulses are applied to it in the proper sequence. The rotation of the motor has direct relationships to the applied input pulses. The sequence of applied pulses is directly related to the direction of motor shaft rotation. The speed of the motor shafts is directly related to the frequency of the input pulses and the length of rotation is directly related to number of input pulses applied.

2.1.2 Characteristics

Characteristics of the stepper motor are as follows:

Holding torque: Stepper motors have very good low speed and holding torque. They are usually hold a position (to a lesser degree) using magnetic torque without application of power.

Open loop positioning: Perhaps the most valuable and interesting feature of the stepper motor is its ability to position the shaft in fine predictable increments. Steppers can run "open-loop" without the need for any kind of encoder to determine the shaft position. Closed loop system (system that uses feedback position information) is known as servo system. Compared to servos, steppers are very easy to control; the position of the shaft is guaranteed as long as the torque of the motor is sufficient for the load under all its operating condition.

Load independent: The rotation speed of a stepper motor is independent of load provided; it has sufficient torque to overcome slipping (moving as on a slippery surface, move smoothly and easily). The higher rpm a stepper motor is driven, the more torque it needs. So all steppers eventually poop out at same rpm and start slipping. Slipping is usually a disaster for steppers, because the position of the shaft becomes unknown. For this reason, software usually keeps the stepping rate within a maximum top rate. In applications where a known rpm is needed under a varying load, steppers can be very handy.

2.1.3 Working principles

The stepper motor uses the theory magnets to make the motor shaft turn a precise distance when a pulse of electricity is applied. The theory of magnet says that same poles of the magnet repel and unlike poles attract.

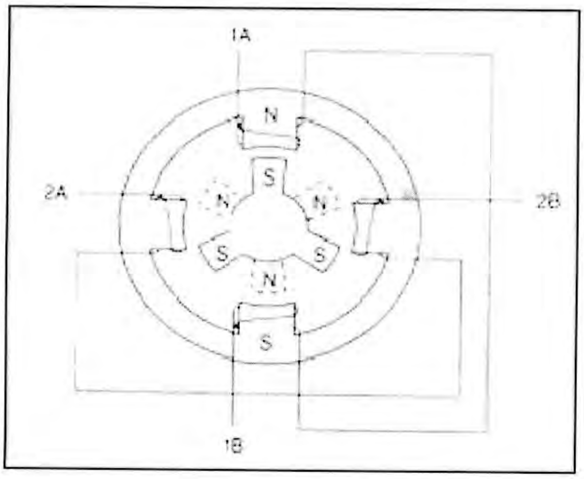


Figure 2.1 Position of the six-pole rotor and four-pole stator of a typical stepper motor [8]

Figure 2.1 shows that the stator (stationary winding) has four poles, and the rotor has six poles (three complete magnets). The rotor will require 12 electric pulses of electricity to move the 12 steps to make one complete rotation. Another way to say this is that the rotor will move precisely 30^0 for each pulse that the motor receives. The number of degrees the rotor will turn when a pulse is delivered to the motor can be calculated by dividing the number of degrees in one complete rotation of the shaft (360^0) by the number of poles (north and south) in the rotor. In the stepper motor 360^0 are divided by 12 to get $30^{0.}$

When no power is applied to the motor, the residual magnetism in the rotor magnets will cause the rotor to detent or align one set of its magnetic poles with the magnetic poles of one of the stator magnets. This means that the rotor will have 12 possible detent positions. When the rotor is in a detent position, it will have enough magnetic force to keep the shaft from moving to the next position. This is what makes the rotor clicking from one position to the next as it is rotated by hand with no power applied.

When power is applied, it is directed to only one of the stator pairs of windings, which will cause that winding to become a magnet, one of the coils for the pair will become the North Pole and the other will become the South Pole. When the opposite polarity is applied the stator coil that is the South Pole will attract the closest rotor tooth that has the opposite polarity. When current flows through these poles, the rotor gets much stronger attraction to the stator windings and the increased torque is called *holding torque.*

By changing the current flow to the next stator winding, the magnetic field will change to 90^0. The rotor will only move 30^0 before its magnetic field will change to 90^0. The rotor will only move 30^0 before its magnetic field in the stator is continually changed as the rotor moves through the 12 steps to move a total of 360^0.

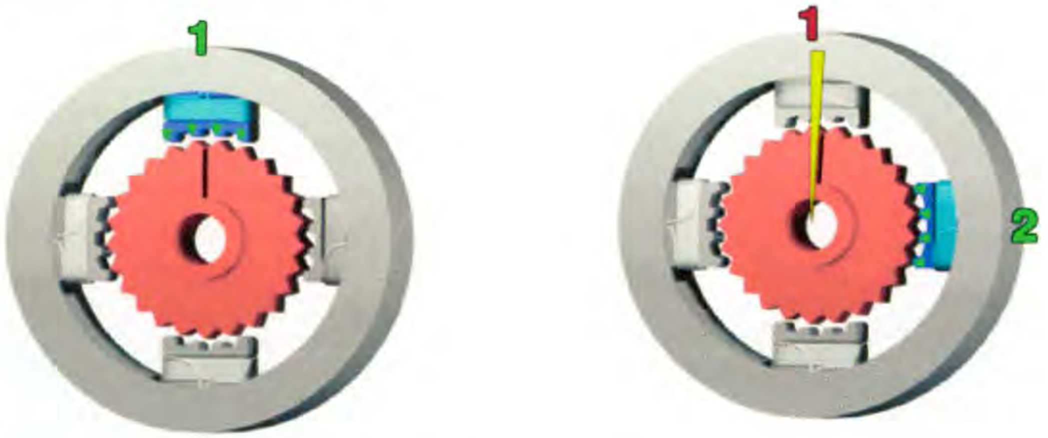


Figure 2.2 Position of the rotor changing as the current supplied to the stator changes [9]

2.1.4 Stepping mode

The following are the most common drive modes:

- Wave drive
- Full step drive
- Half step drive
- Micro stepping

In wave drive only one winding is energized at any given time. The stator is energized and the rotor steps from position 8→2→4→6.

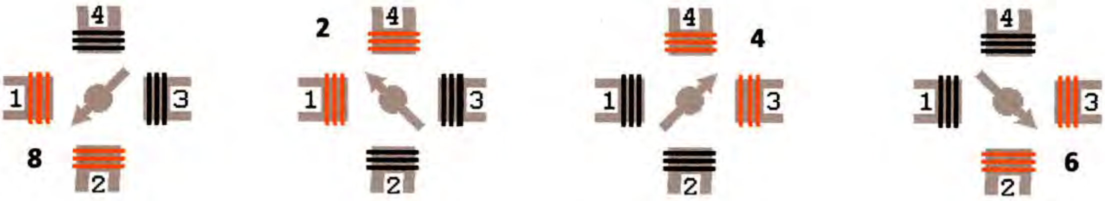


Figure 2.3 Steps for wave drive [9]

For unipolar and bipolar wound motors this excitation mode would result in the same mechanical position with the same winding parameters. The disadvantage of this drive mode is that in the unipolar wound motor only 25% and in bipolar motor only 50% of the total motor winding is used at any given time. This means that it is not getting the maximum torque output from the motor.

In full step drive the stator is energized and the rotor steps from position 1→3→5→7.

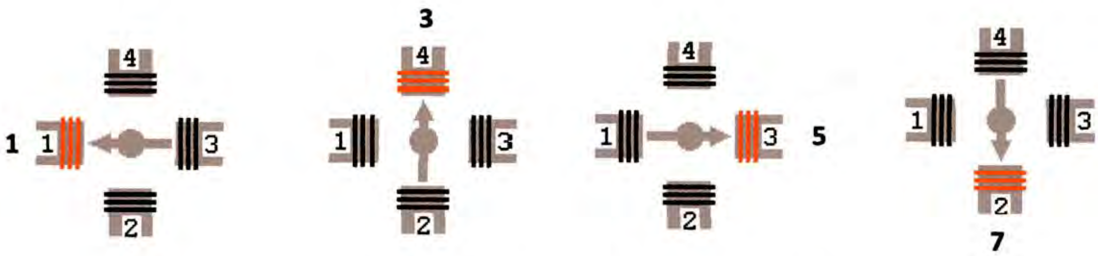


Figure 2.4 Steps for full step drive [9]

Full step mode results in the same angular movements as 1 phase on drive but the mechanical position is offset by one half of a full step. The torque output of the unipolar wound motor is lower that the bipolar motor (for motors with the same winding parameters) since the unipolar motors uses only 50% of the available winding while the bipolar motor uses the entire winding.

Half step drive combines both wave and full step drive modes. Every second step only one phase is energized. The stator is energized and rotor steps position 1→ 2→3→ 4→ 5→6→7→8. These results in angular movements that are half of those in 1- or 2- phases drive modes. Half stepping can reduce a phenomena referred to as resonance which can be experienced in 1- 2- phases on drive modes.

The excitation sequences for the above drive mode are summarized below.

Single stepping or single-coil excitation: Each successive coil is energized in turn as shown in Figure 2.5.

The energized coil is shown in red (coil 4, coil 3, coil 2, coil1 consecutively) and others in black.

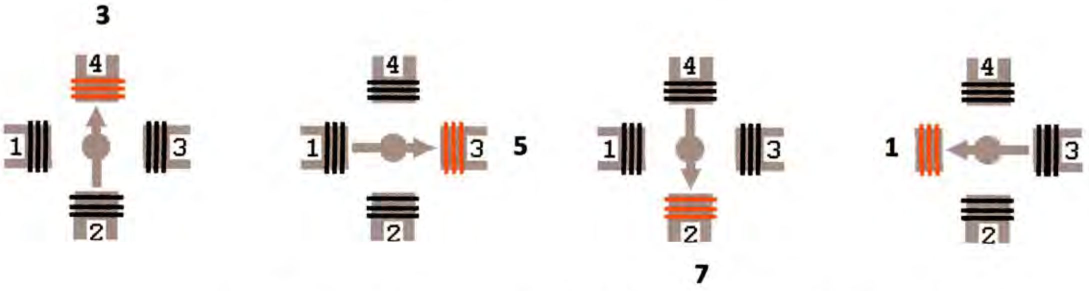


Figure 2.5 Single stepping or single-coil excitation [9]

If coil 1 is connected to D0, coil2 to D1, coil 3 to D2 and coil 4 to D3, to energize coils in single stepping it this pattern must be applied to the data pins (D0,D1,D2,D3) of the parallel port.

Table 2.1 Pattern for single stepping or single-coil excitation

Single Step				
Step No.	D3	D2	D1	D0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

High torque stepping or two-coil excitation: Each successive pair of adjacent coils is energized in turn.

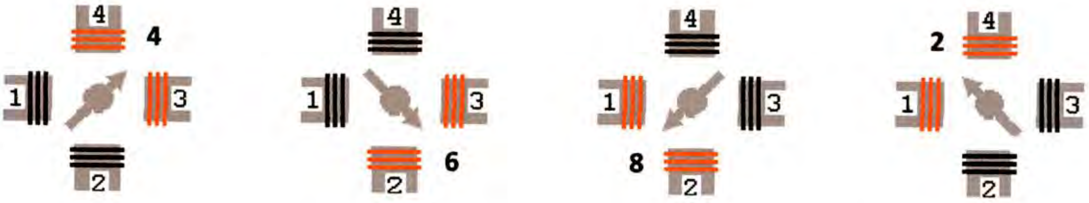


Figure 2.6 High torque stepping or two-coil excitation [9]

Table 2.2 Pattern for high torque stepping or two-coil excitation

High Torque Stepping				
Step No.	D3	D2	D1	D0
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Half stepping: It is the mix of single stepping mode and high torque mode.

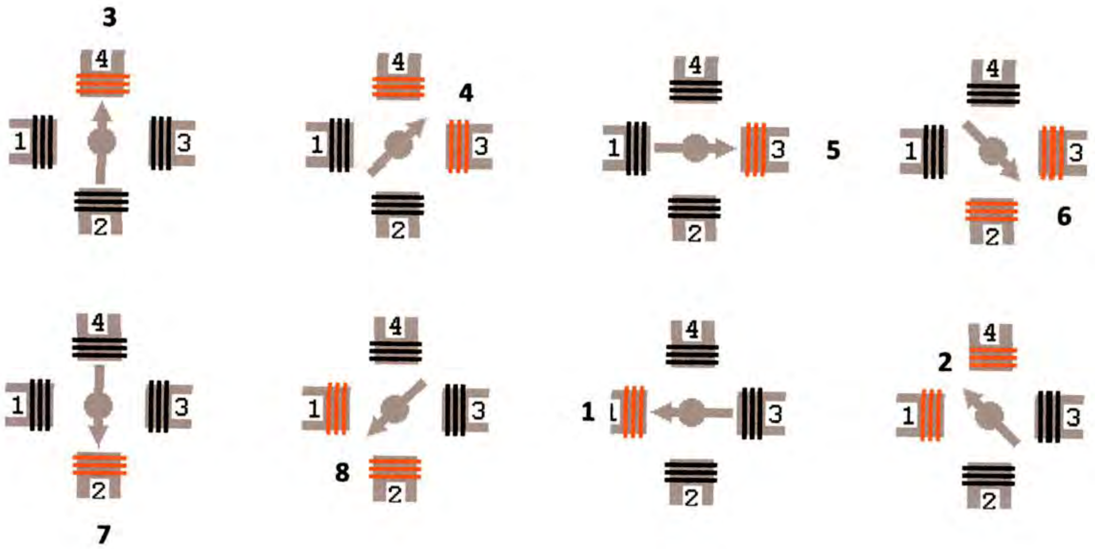


Figure 2.7 Half stepping [9]

Table 2.3 Pattern for half stepping

Half Stepping				
Step No.	D3	D2	D1	D0
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

All the above patterns cause clockwise rotation of the motor. For anti-clockwise rotation the pattern has to be applied in reverse order.

2.2 PARALLEL PORT

2.2.1 Definition

Parallel port is a simple and inexpensive medium for building computer controlled system. The simplicity and ease of programming makes parallel port popular in electronic hobbyist world. The parallel port is often used in computer controlled robots, Atmel/PIC programmers; home automations etc. The primary use of parallel port is to connect printers to computers and is specifically designed for this purpose. Thus it is often called as printer port or Centronics port (this name came from a popular printer manufacturing company "Centronics" who devised some standards for parallel port). In the proposed project parallel port is used for communication between PC and our hardware.

2.2.2 Parallel port modes

The IEEE 1284 standard which has been published in 1994 defines five modes of data transfer for parallel port. They are:

1) Compatibility Mode
2) Nibble Mode
3) Byte Mode
4) EPP (Enhanced Parallel Port)
5) ECP (Enhanced Control Port)

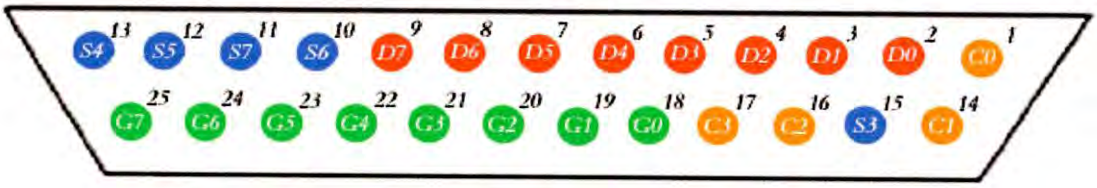


Figure 2.8 Parallel port [10]

The pins in DB25 connector are divided into three groups, they are

1) Data pins (data bus D0-D7)
2) Control pins (C0-C7)
3) Status pins (S0-S7)

As the name refers, data is transferred through data pins, control pins are used to control the peripheral and of course, the peripheral returns status signals back to computer through status pins. These pins are connected to data, control and status internally.

2.2.3 Parallel port- Data pins

For sending the sequences (the data ports which can be seen in the picture of Figure 2.8) from D0 to D7 are needed.

2.2.4 Parallel port –Status pins

These ports are made for reading signals. The range is S0-S7. But S0, S1, S2 are invisible in the connector. And S0 is different; this bit is for timeout flag in EPP compatible ports. The DATA pins are located at the base address of the port e.g. DATA address for LPT1 is 0x378. The STATUS is located at "portBase + 1", e.g. the STATUS address for LPT1 is 0x379. It can send 5 numeric data from the 10 – 11 – 12 – 13 – 15th pins. Every parallel port has an address. In windows 2000, it is found by setting>control panel > System >Hardware > Device Manager> Ports (COM & LPT) > Printer Port (LPT1)> Properties = in resources> Resource setting and it can be seen the address for the parallel port. For example: generally it is 0378-037F. This is hexadecimal like in math (mod 16). 0x378 belongs to 888 in decimal form. In this way we can look for com port or game port addresses. Description of the status pins are as follows:

- S0: This bit becomes higher (1) if a timeout operation occurs in EPP mode.
- S1: Not used (May be for decoration).
- S2: Mostly not used but sometimes this bit shows the cut condition PIRQ (Parallel Port Interrupt Request) of the port.

- S3: if the printer determines an error it becomes lower (0). This is called nError or nFault.
- S4: it is high (1) when the data inputs are active. This is called select.
- S5: it is high (1) when there is no paper in printer. This is called PaperEnd, PaperEmpty or Perror.
- S6: It sends low impact signaling when the printer gets a one byte data. This is called nAck or nAcknowledge.
- S7: This is the only revered pin in the connector. If the printer is busy and it cannot get any additional data this pin becomes lower. This is called Busy.

2.2.5 Parallel port – Control pins

These signals are usually used as output but these can also be used for input. The range is like in data port C0- C7 but C4, C5, C6, and C7 are invisible in connector. And the address for this is 0x37A.

- C0: This pin is reversed. It sends a command to read D0-D7 on the port. When the computer starts it is high in the connector. This is called nStrobe.
- C1: This pin is reversed. It sends a command to the printer to feed the next line. It is high in the connector after the machine starts. This is called auto LF.
- C2: This pin is to reset the printer and clear the buffer. This is called nInit, nInitialize.
- C3: This pin is reversed. Sends a high (1) for opening data inputs. It is low after the machine starts. This is called nSelection.
- C4: opens the cut operation for the printer. Not visible in the connector.
- C5: sets the direction control in multidirectional ports. Not visible in the connector.
- C6: not used and also not visible in the connector.
- C7: mostly not used and invisible in the connector.

2.2.6 Parallel port – Ground pins

These are (G0- G7) pins from 18 to 25. These are mostly used for the completing the circuit. Different pins are required when using all the pins including the inputs.

After these data ports can be used in experimental purpose because there are reversed pins in control and status ports. Here is an explanation for reversed pins: while not sending any signals to the data port it is in closed position like "00000000" so the 8 pins have no voltage on it (0 Volt). If send decimal "255" (binary "11111111") then every pin (D0-D7) will have 5 volt. On the other hand, if use control ports, there are reversed pins which are C0, C1, and C3 so while send nothing to the control port its behavior is "0100" in binary (Decimal "11").

2.3 WEBCAM

2.3.1 Basics of webcam

A simple webcam consists of a digital camera attached to computer, usually through the USB port. The camera part of the webcam is just a digital camera. The "Webcam" nature of the camera derives from the software. Webcam software takes a frame from the digital camera at a preset interval (for example, the software might grab a still image from the camera once every 30 seconds) and transfers it to another location for viewing.

2.3.2 Capturing images through webcam

In the developed system (server end) webcam captures images of desired locations and save the images in specific location of local host.

Figure 2.9 Sample webcam

CHAPTER 3
PROJECT INITIATION AND PLANNING

3.1 PROJECT INITIATION

As a project it has been decided to develop software that will fetch images to the hand held devices (mobile phone, PDA) over the internet using HTTP protocol continuously and also control the camera with the help of the server. This is important because at times it feels the necessity to check the condition of someone's or organizations valuable possessions.

3.2 PROJECT PLANNING

After choosing this project, at client end, the first job is to study the J2ME language and various communication protocols to find out the suitable protocol to serve the purpose. Initially it has been planned for RTSP protocol. It is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The protocol is used to establish and control media sessions between end points. The transmission of streaming data itself is not a task of the RTSP protocol. Most RTSP servers use the Real-time Transport Protocol (RTP) for media stream delivery. But eventually it is found out that RTSP is supported in J2ME but MMAPI doesn't support real time video streaming. Hence alternatives are started to find out and eventually HTTP protocol is found suitable to transfer image for the job.

At server end, socket programming is needed for the communication over the internet between both ends, Java Media Framework (JMF) to capture images from webcam, NetBeans IDE, parallel port accessing and stepper motor control.

Java Media Framework (JMF) is a Java library that enables audio, video and other time-based media to be added to Java applications and applets. This optional package, which can capture, play, stream, and transcode multiple media formats, extends the Java Platform, Standard Edition (Java SE) and allows development of cross-platform multimedia applications.

Java Platform, Standard Edition or Java SE is a widely used platform for programming in the Java language. It is the Java Platform used to deploy portable applications for general use. In practical terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or "packages") needed to allow the use of file systems, networks, graphical interfaces, and so on, from within those programs.

NetBeans IDE is built using an open source (describes practices in production and development that promote access to the end product's source materials) API called NetBeans platform. The API is the base on which the NetBeans IDE is developed and is available for others to avoid creating biolerplate (any text that is or can be reused in new contexts or applications without being changed much from the original. Many computer programmers often use the term boilerplate code) code. The NetBeans platform is a generic framework for Swing applications. It provides the "plumbing" that, before, every developer had to write themselves -the plumbing-saving state, connecting actions to menu items, toolbar items and keyboard shortcuts; window management, etc. Some advantages of using the NetBeans platform are as follows:

- Extensible and easy modular design
- Vast API of commonly used tasks
- Easy to design Swing GUI using NetBeans IDE
- Pre-defined update center and automatic update notification

The NetBeans IDE is an award-winning integrated development environment available for Windows, Mac, Linux, and Solaris. The NetBeans application platform enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as JavaFX, PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy and Grails, and C/C++.

CHAPTER 4

SYSTEM DESIGN AND DEVELOPMENT

4.1 OPERATING SYSTEM REQUIREMENTS

In this project mobile phones that support MMAPI on top of JAVA operating system as well as Symbian Operating System (OS) is used. Symbian OS is a 32-bit multi-tasking operating system that is specifically designed for portable, battery-powered mobile phones (smart phones). Symbian OS programming is event-based. The CPU is switched off when applications are not directly dealing with an event. It is built for handheld devices, with limited resources. Most of the vendors like NOKIA, SIEMENS, SAMSUNG, ERICSSON, Sony Ericsson, and Panasonic use Symbian OS for their mobile phone.

At server end, Windows XP is used, but printer port in XP is in lock mode. So in order to use Windows XP firstly, it will be needed to unlock the printer port. With the help of third party program "parport" the printer port can be unlocked. The purpose of parport is to allow multiple device drivers to use the same parallel port.

4.2 COMPONENTS USED IN THE PROJECT

Following are some components that are used to build the hardware for this project:

Mobile Phone: A mobile phone that must be capable of image handling and EDGE (preferably but GPRS will do) supportive. I have used NOKIA 5300 for experimental purpose in this project.

Webcam and Stepper motor: One webcam is used to capture images and a stepper motor is used to control the webcam movement.

Parallel port: Parallel port is needed to connect the stepper motor with my system.

ULN 2003: It is a 7-bit 50V 500mA TTL-input NPN transistor. In this project it is used to drive the stepper motor. A datasheet for ULN 2003 is given in Appendix B.

Others: Breadboard, 12 V AC adaptor and some connect wires.

4.3 SYSTEM BLOCK DIAGRAM

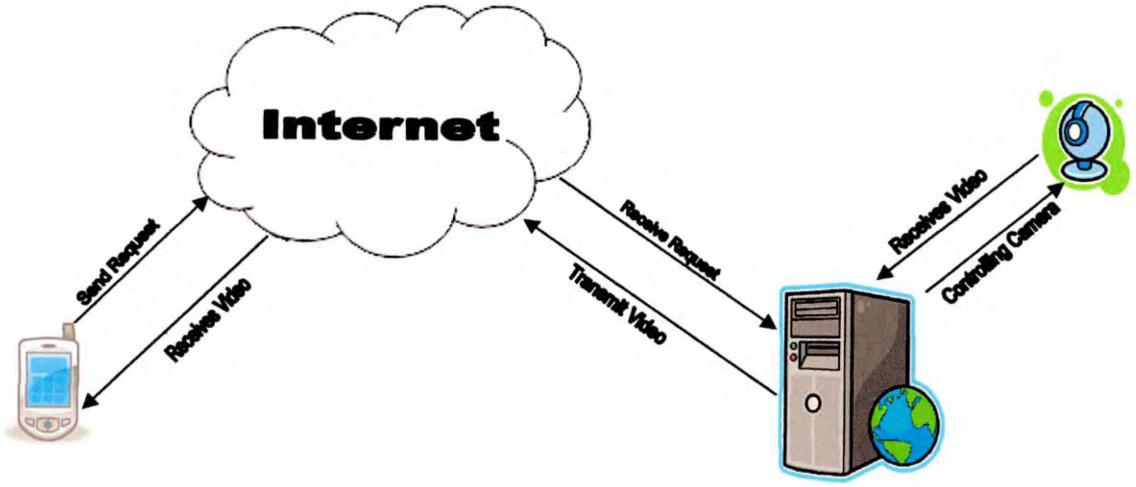


Figure 4.1 System block diagram

Figure 4.1 shows how client will send request to establish connection with the server. For this purpose client will send request for establishing connection with the server by key-in the URL from their mobile phone. The server will send acknowledgement to the client and when the client confirms the address, the connection is established. Then client will start fetching images from the specific location of the local host where images are being saved that is captured by the webcam using HTTP protocol.

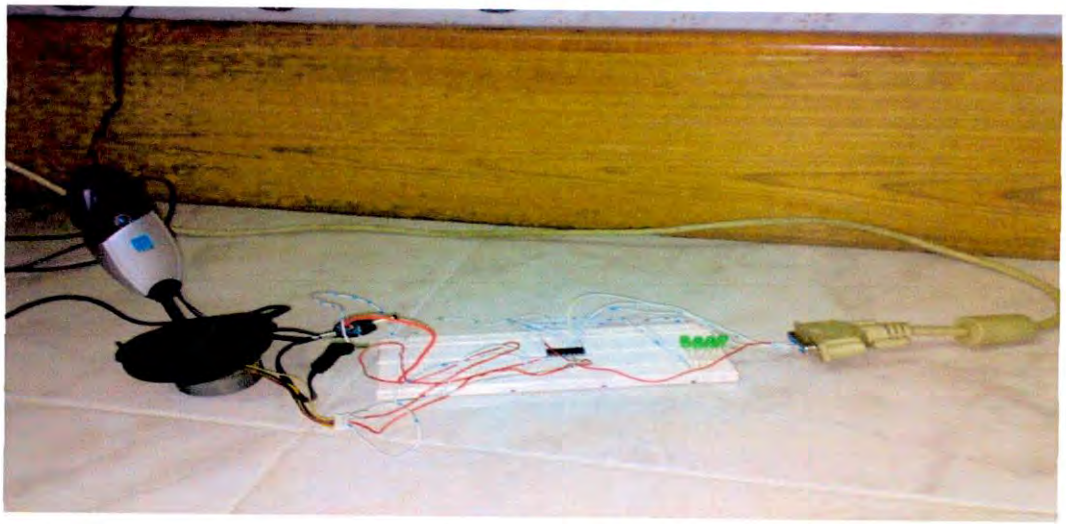


Figure 4.2 Hardware connection at server side

Client has also option to send control instruction from the mobile phone for horizontal movement of the webcam. Figure 4.2 shows how control instruction will be transmitted from the parallel port to the ULN 2003 on client's demand and then it drives the stepper motor which in turn will move the webcam. For better understanding a schematic diagram of hardware circuit is given in Figure 4.3:

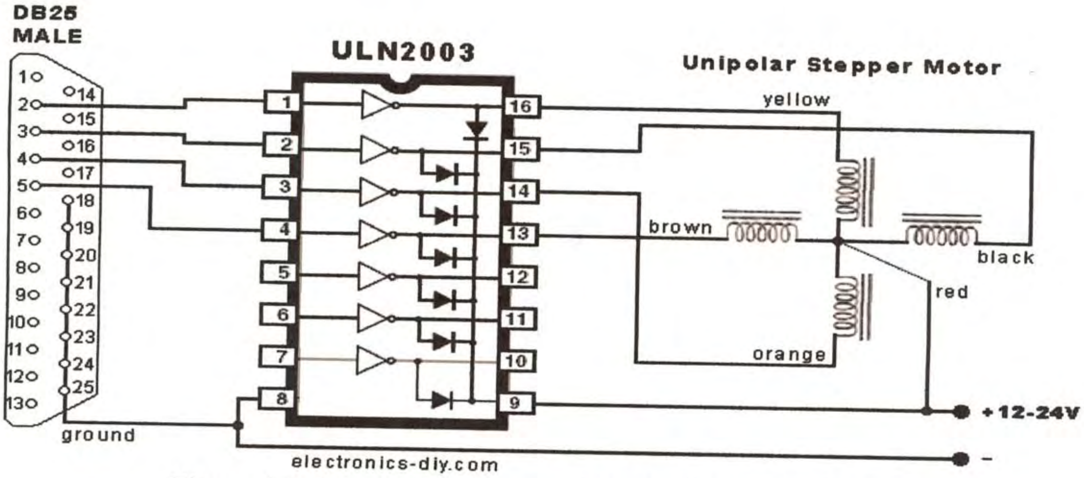
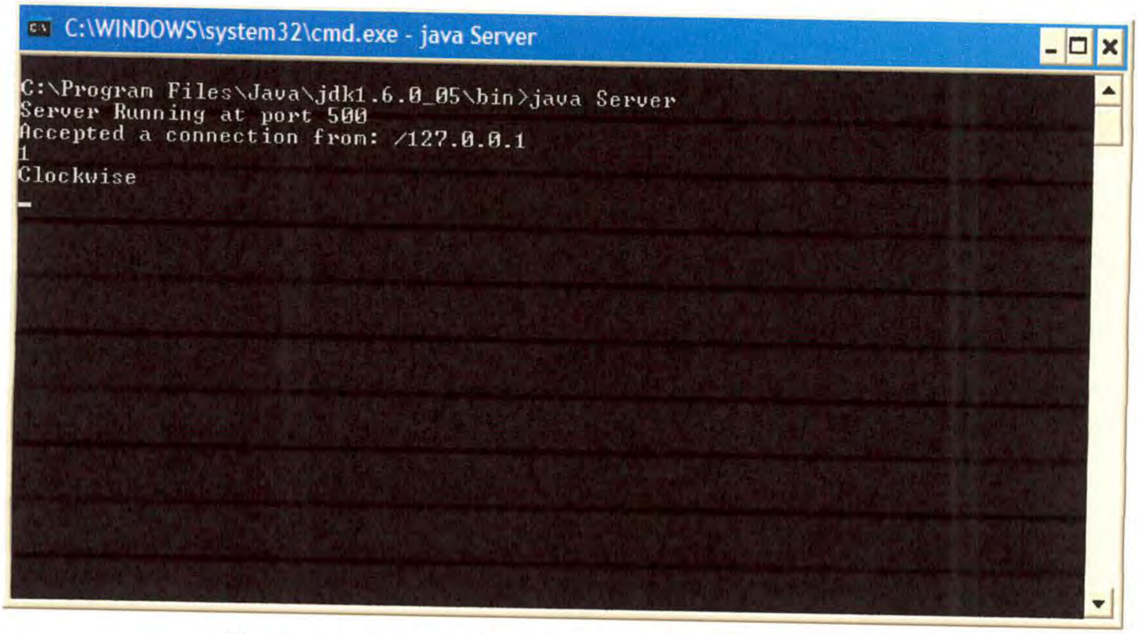


Figure 4.3 Schematic diagram of the hardware circuit [10]



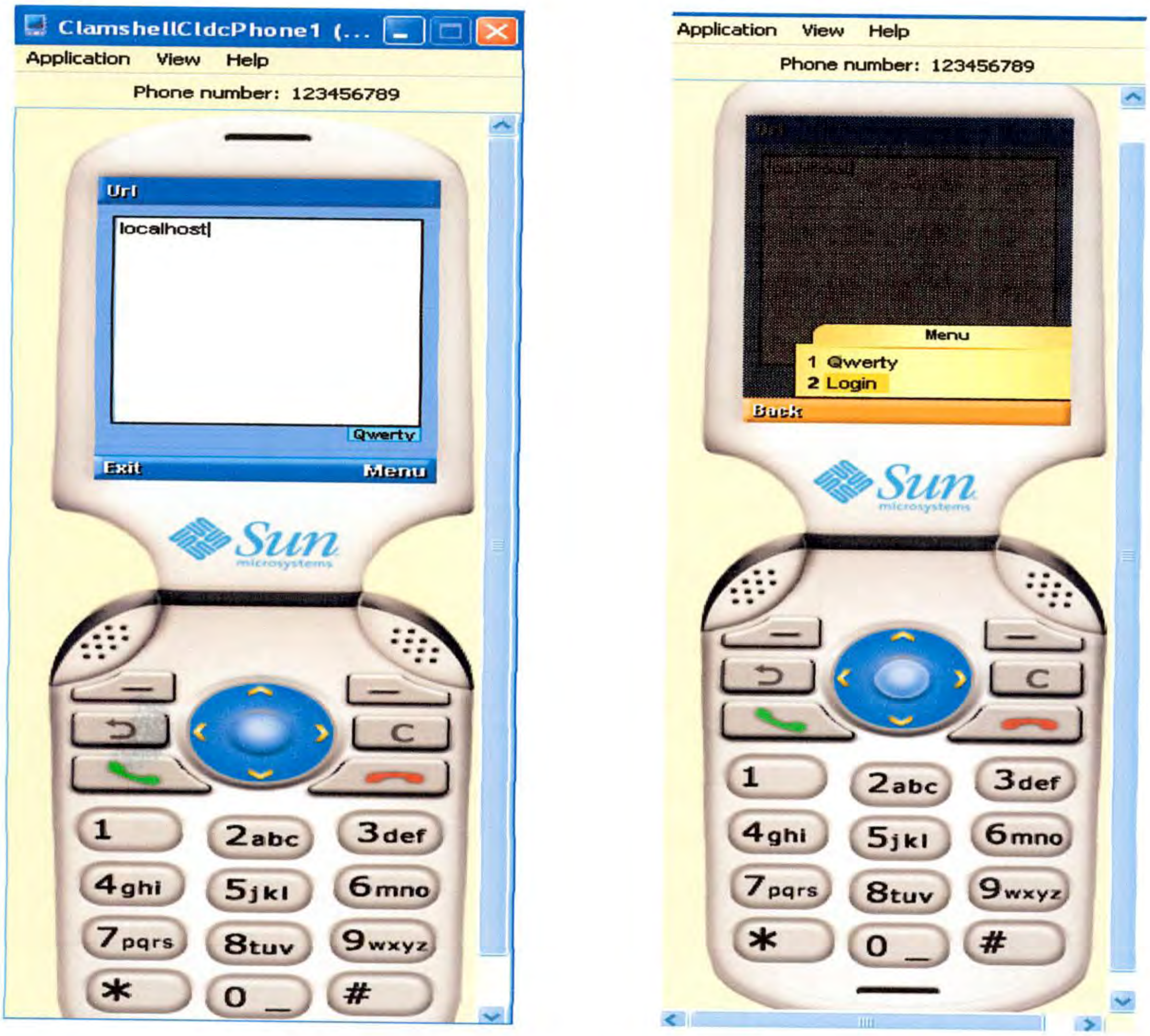
```
C:\Program Files\Java\jdk1.6.0_05\bin>java Server
Server Running at port 500
Accepted a connection from: /127.0.0.1
1
Clockwise
```

Figure 4.4 Server is listening for controlling instruction

Figure 4.4 shows the server's response after receiving control instruction from client. It shows 1 and clockwise when it will get instruction for left movement from the client and vice versa.

4.4 GRAPHICAL USER INTERFACE (GUI) DESIGN

The user interfaces help the user to enter the server address by pressing Login from the menu option. When the user confirms the address, the GUI starts displaying the images it fetched from the server's specific location. From the menu option the user can also select the control instruction to be transmitted to the server for controlling the horizontal movement of the webcam. Figure 4.5 shows the snaps taken from the emulator (GUI) while testing the program using local host. It is the initial screen while sending request to establish connection with the server.


ClamshellCldcPhone1 (...

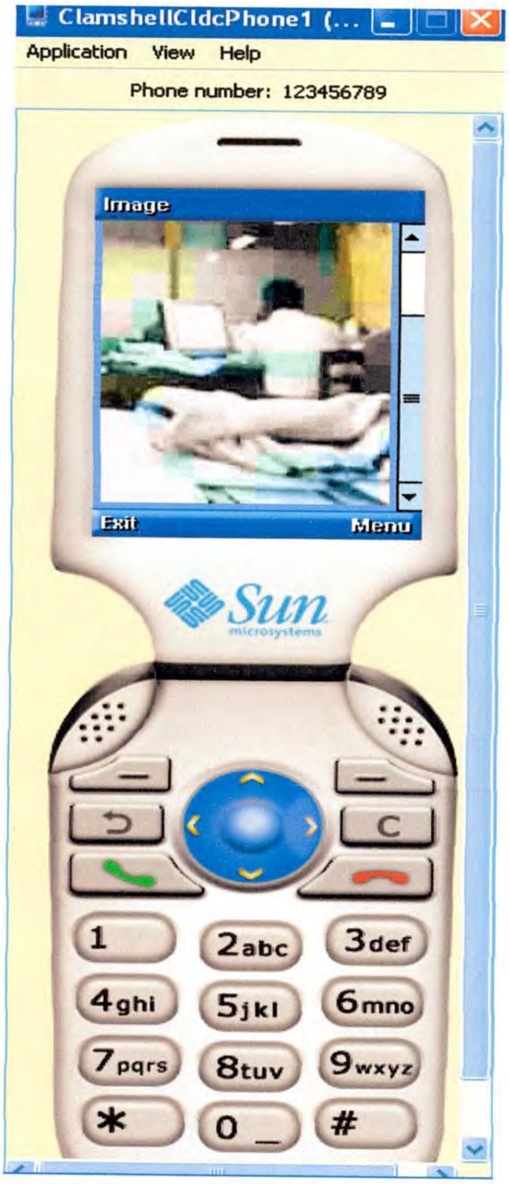
Phone number: 123456789

Url

localhost

Phone number: 123456789

Figure 4.5 (a) Initial Screen (b) Sending request for establishing connection



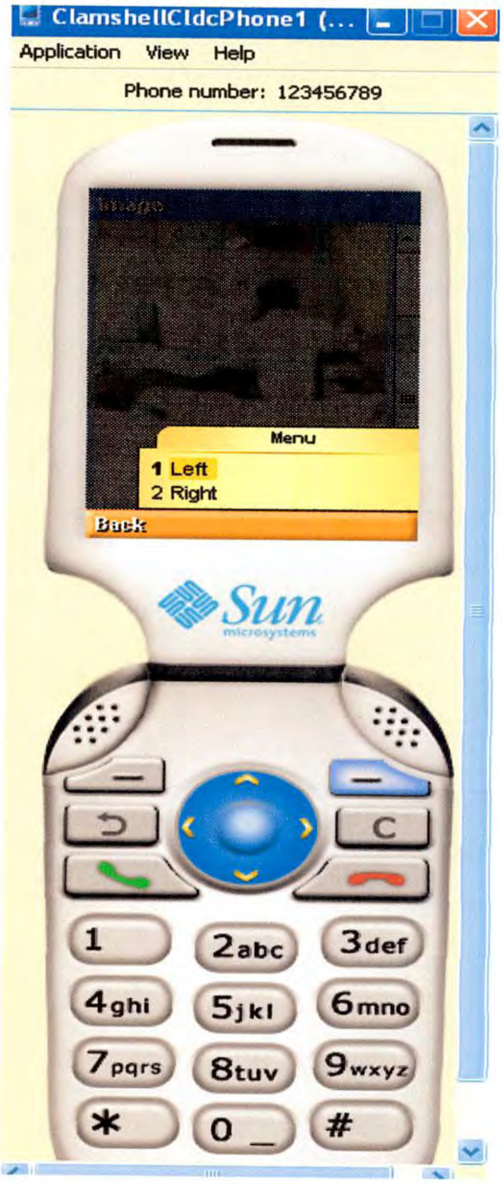


Figure 4.6 Viewing image of a desired location

Figure 4.7 Sending control instruction to server

Figure 4.6 shows the images after establishing connection. This image is fetched from the specific location of the local host. Figure 4.7 shows how control instruction can be sent to server for the movement of the webcam by using menu option. If left option is clicked, then webcam will move its face to left and then take snaps of that position and vice versa.

4.5 WORKING PROCEDURE

4.5.1 Overall description

In this project, there are Client and Server end. Client end has three jobs to perform. First job is to send request for establishing connection with the server end. After establishing connection, client end starts fetching images from the webcam through the server which is the second job. Client end also has the option to send control instruction for horizontal movement of the webcam.

4.5.2 Client commands

Client has two commands: Left Move and Right Move. Commands are sent to the server from the client end. The server moves the webcam based on the commands it receives.

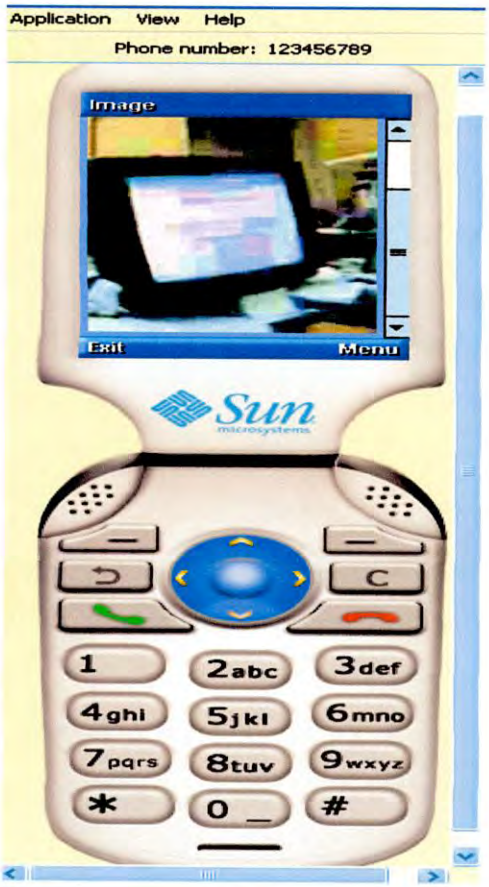


Figure 4.8 Left image of a desired location

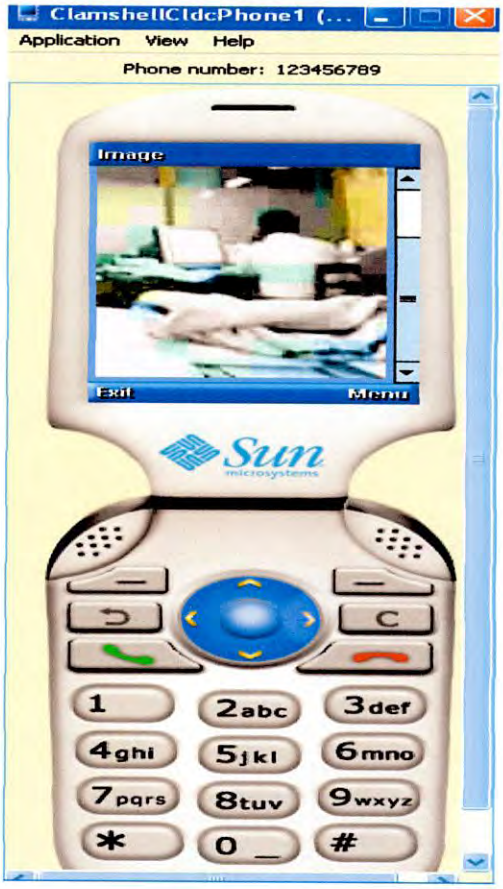


Figure 4.9 Right image of a desired location

CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 PROTOTYPE

In real field, it is not yet possible for MMAPI to support real time video streaming. According to Java forum, they are working on this for real time video streaming, and they are hopeful that this feature will be supported in their next release [11].

On the server side Windows XP operating system is used. As Windows XP with Service Pack2 and its higher versions keep the parallel port locked for security purpose, it is required to open the port using third party tools and access it for communicating via the parallel port. In other operating systems like Windows 98 the same software will work fine but for running in LINUX some minor changes in the code are required.

5.2 TESTING AND RESULT

Test experiments for the project have been conducted using various mobile phones such as NOKIA N-73, Siemens M75, NOKIA 6630, Sony Ericsson W810i, NOKIA 5300 and Philips 960. In all the devices result is successful. But time for fetching image varies as GPRS class is not same in all the devices and the data service provider supporting EDGE performs better. Figure 5.1 shows the images fetched by NOKIA 5300 and Sony Ericsson W810i from the server being captured by webcam.

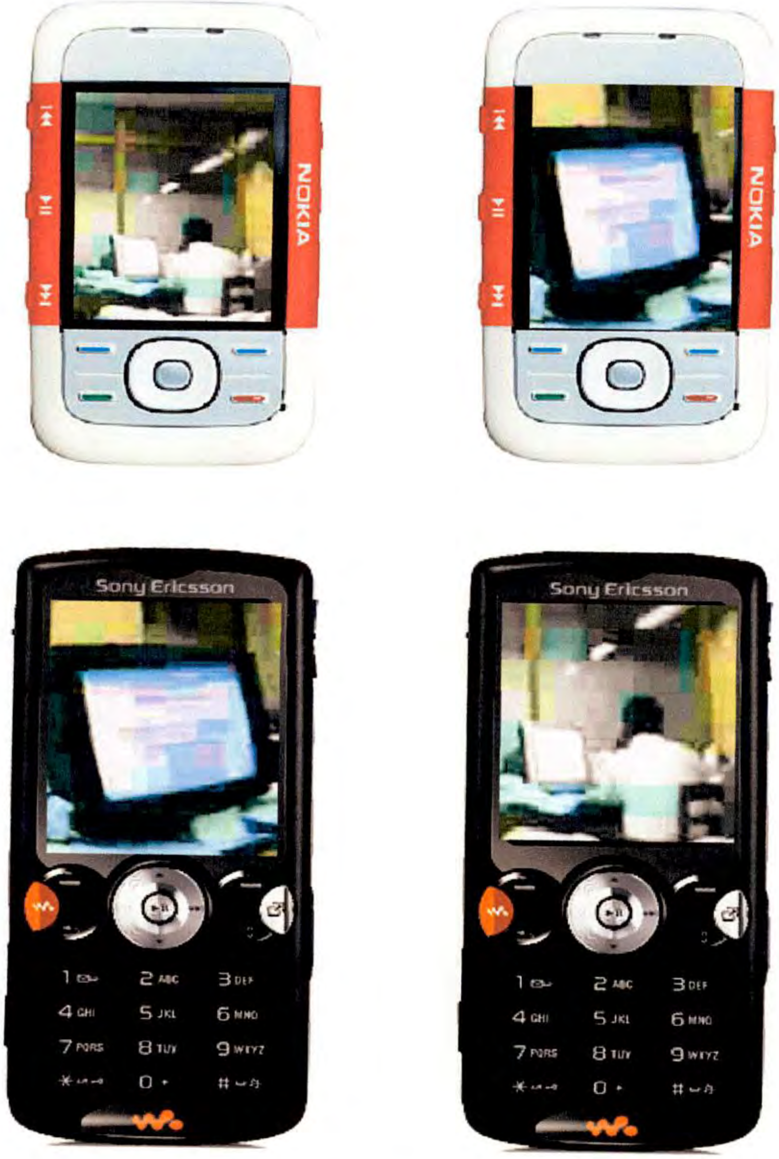


Figure 5.1 Viewing images by NOKIA 5300 and Sony Ericsson W810i

Pictures as shown in Figure 4.8, 4.9 and 5.1 are same in both emulator and real mobile phones. So it proves that the proposed system functions successfully in real time.

CHAPTER 6

CONCLUSION AND FUTURE SCOPES

6.1 CONCLUSION

Now-a-days security is an important issue in our daily life. This project is related to the mobile phone environment as it is becoming a most used technology and it is growing too fast. Exploiting the benefits of the mobile phone technology, a new approach of surveillance system that can monitor any desired places from a remote location has been proposed in this project. A webcam integrated with stepper motor and other hardware accessories are connected to a server placed in the desired location. User can access the server from any remote location anytime and view the images of his/her desired location captured by the webcam. The project has been tested first by using an emulator (GUI) which is designed using the NetBeans IDE and then using different types of mobile phones. The result is same in both platforms and it proves the proper functionality of the proposed system. The proposed approach can also be implemented on other hand held platforms like PDA by some minor modifications.

6.2 FUTURE SCOPES

To enhance the work presented in this report, we may dagger the following recommendation for future extension:

a) In next version of this project USB port can be used instead of parallel port for controlling the webcam movement. Because a USB port is a standard cable connection interfaces on personal computers and consumer electronics. It allows stand-alone electronic devices to be connected via cables to a computer (or to each other).

b) One new feature such as recording of video of a particular location for a time and sending it to the user through email can be incorporated in the next version of the project.

c) In order to increase the security, movement of the webcam can be controlled not only horizontally but also vertically too. This feature can also be incorporated in the next version of the project.

References:

1) http://www.securitypark.co.uk/security_news_9_CCTV%20System.html, Last accessed on 05.01.2010.
2) http://www.deskshare.com/Resources/Articles/wcm_setup_PCvideosurveillancesystem.aspx; Last accessed on 26.01.2010.
3) http://www.securitypark.co.uk/security_article264278.html , Last accessed on 26.01.2010.
4) Kanade, T., Collins, R. T. And Lipton, A. J, "Advances in cooperative multi-sensor video surveillance", Proceedings of DARPA Image Understanding Workshop, pp. 3-24, November 1998.
5) http://www.buzzle.com/articles/benefits-of-using-a-cell-phone.html; Last accessed on 26.01.2010.
6) Chandramathi S., Chandra Sekhar N.,Adarsh M.G.,"A Novel Video Surveillance System based on Multimedia Messaging Service", Journal of Computer Science, Science Publications 2005.
7) Shahriyar Rifat, Hoque Enamul, Naim Iftekhar and Sohan S.M., "Controlling Remote System using Mobile Telephony", Undergraduate Thesis, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
8) Kissell E. Thomas," Stepper Motor Theory of Operation", September 6, 2006.
9) http://en.wikipedia.org/wiki/Stepper_motor ; Last accessed on 17.02.2010.
10) http://en.wikipedia.org/wiki/Parallel_port; Last accessed on 17.02.2010.
11) Miller P. Frederic, Vandome F. Agnes and McBrewster John,"Electric Motor: Servomechanism, Synchronous motor, Induction motor, Brushed DC electric motor, Brushless DC electric motor, AC motor, Stepper motor, Linear motor, Doubly-fed electric machine " ,Paperback - Oct. 11, 2009.
12) http://www.java-forums.org/; Last accessed on 16.05.2010.

Appendix A

```
/* Project Coding */

/* Server.Java*/

import parport.ParallelPort;
import java.net.*;
import java.io.*;
import java.util.*;

public class Server extends Thread
{
      private ServerSocket ser;

      public static void main(String args[]) throws Exception
      {
            new Server();
      }

      public Server() throws Exception
      {
            ser = new ServerSocket(500);
            System.out.printf("Server Running at port 500");
            this.start();
      }

      public void run()
      {
            while(true)
            {
                  try
                  {
                        Socket client = ser.accept();
                        System.out.println("\nAccepted a connection from:
"+client.getInetAddress());
                        Connect c =new Connect(client);
                  }
                  catch(Exception e)
                  {

                  }
            }
      }
}

class Connect extends Thread
{
      private Socket client=null;
      private InputStreamReader dis=null;
      //private ObjectOutputStream oos=null;
```

```
 public Connect()
 {
 }

 public Connect(Socket cs)
 {
        client=cs;
        try
        {
                dis = new InputStreamReader(client.getInputStream());
               //oos = new ObjectOutputStream(client.getOutputStream());
        }
        catch(Exception e)
        {
               try
               {
                      client.close();
               }
               catch(Exception e1)
               {
               //     System.out.println(e1.getMessage());
               }
               return;
        }
        this.start();
}
public static void  delay()
   {
              double i,l;
              for( i=0;i<100000;i++)
              for(l=0;l<50;l++);
       }


public void run()
{
       try
       {
              int x=dis.read();
              System.out.println(x);
              if(x==1)
              {
                     ParallelPort lpt1 = new ParallelPort(0x378);
                     System.out.println("Clockwise");
                     for(int j=0;j<1;j++)
                     {
                            lpt1.write(8);delay();
                            lpt1.write(4);delay();
                            lpt1.write(2);delay();
                            lpt1.write(1);delay();
                     }
              }
              else if(x==2)
              {
                     ParallelPort lpt1 = new ParallelPort(0x378);
                     System.out.println("AntiClockwise");
                     System.out.println(x);
```

```
                                for(int j=0;j<1;j++)
                                {
                                        lpt1.write(1);delay();
                                        lpt1.write(2);delay();
                                        lpt1.write(4);delay();
                                        lpt1.write(8);delay();
                                }

                        }
                        else
                        {
                                System.out.println("Invalid");
                                System.out.println(x);
                        }
                        dis.close();
                        client.close();
                }
                catch(Exception e)
                {
                }
        }
}


/*Client.Java */

import java.io.*;
import java.net.*;


public class Client2
{
        public static void main(String args[])
        {
                OutputStreamWriter oos = null;
                InputStream ois = null;
                Socket soc = null;
                int x=1;
        //      System.out.print(x);
                try
                {
                        soc = new Socket("localhost", 500);
                        oos = new OutputStreamWriter(soc.getOutputStream());
                //  oos.writeObject(x);
                        //System.out.println("Opened Socket");
                        oos.write(x);
                        oos.flush();
                        oos.close();
                }
                catch(Exception e)
                {
                        System.out.println(e.getMessage());
                }
        }
}
/* ParallelPort.Java */
```

```
package parport;

public class ParallelPort {

   /** The port base address (e.g. 0x378 is base address for LPT1) */
   private int portBase;

   /** To cunstruct a ParallelPort object,
     * you need the port base address
   */
   public ParallelPort (int portBase)
   {
      this.portBase = portBase;
   }
   /** Reads one byte from the STATUS pins of the parallel port.
     *
     * The byte read contains 5 valid bits, corresponing to 5 pins of input
     * from the STATUS pins of the parallel port (the STATUS is located
     * at "portBase + 1", e.g. the STATUS address for LPT1 is 0x379).
     *
     * This diagram shows the content of the byte:
     *
     *  Bit | Pin # | Printer Status  | Inverted
     * -----+-------+-----------------+-----------
     *   7  |  ~11  | Busy            |   Yes
     *   6  |   10  | Acknowledge     |
     *   5  |   12  | Out of paper    |
     *   4  |   13  | Selected        |
     *   3  |   15  | I/O error       |
     *
     * Note that Pin 11 is inverted, this means that "Hi" input on pin
     * means 0 on bit 7, "Low" input on pin means 1 on bit 7.
   */
   public int read ()
   {
      return ParallelPort.readOneByte (this.portBase+1);
   }
   /** Writes one byte to the DATA pins of parallel port.
     * The byte is written to the DATA pins of the port. The DATA pins are
     * located at the base address of the port (e.g. DATA address for LPT1
     * is 0x378).
     *
     * This diagram shows how the byte is written:
     *
     *  Bit | Pin # | Printer DATA
     * -----+-------+--------------
     *   7  |   9   |   DATA 7
     *   6  |   8   |   DATA 6
     *   5  |   7   |   DATA 5
     *   4  |   6   |   DATA 4
     *   3  |   5   |   DATA 3
     *   2  |   4   |   DATA 2
     *   1  |   3   |   DATA 1
     *   0  |   2   |   DATA 0
   */
   public void write (int oneByte)
   {
        ParallelPort.writeOneByte (this.portBase, oneByte);
    }

    /** Reads one byte from the specified address.
      * (normally the address is the STATUS pins of the port)
      */
    public static native int readOneByte (int address);

    /** Writes one byte to the specified address
      * (normally the address is the DATA pins of the port)
      */
    public static native void writeOneByte (int address, int oneByte);

    static
    {
       System.loadLibrary("parport");
    }
}
```

/*ViewImage.Java*/

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
/**
 * @author Administrator
 */
public class ViewImage extends MIDlet implements CommandListener,Runnable
{
    private Display display;
    private TextBox tbmain;
    private Alert alStatus;
    private Form frm;
    private Command exit;
    private Command ok,left,right;
    private static final int Alerttime=3000;
    private String url;
    Image im=null;

    int controlVal;

    public ViewImage()
    {
        display=Display.getDisplay(this);
        tbmain=new TextBox("Url","localhost",100,0);

        exit=new Command("Exit", Command.EXIT,1);
        ok =new Command("Login", Command.SCREEN,1);
        left =new Command("Left", Command.SCREEN,2);
        right=new Command("Right", Command.SCREEN,3);
        tbmain.addCommand(exit);
        tbmain.addCommand(ok);
        tbmain.setCommandListener(this);
        frm=new Form("Image");
        frm.addCommand(exit);
        frm.addCommand(left);
        frm.addCommand(right);
        frm.setCommandListener(this);
    }


    public void startApp()
    {
        display.setCurrent(tbmain);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d)
    {
        if(c==exit)
        {
            destroyApp(false);
            notifyDestroyed();
        }
        else if(c==ok)
        {

            int i=0;
            while(i<90)
            {
url="http://"+tbmain.getString()+"/test/vid2jpg/images/"+i+".jpg";
                Download dl=new Download(url,this);
                dl.start();
                i++;
             }
        }
        else if(c==left)
        {
            controlVal = 1;
            Thread t = new Thread(this);
            t.start();

            //System.out.println("Left");
            //new Control(1);
            //System.out.println("Called");
        }
        else if(c==right)
        {
            controlVal = 2;
            Thread t = new Thread(this);
             t.start();
            //System.out.println("Right");
            //new Control(2);
            //System.out.println("Called");
        }
    }
    public void com1(int x)
    {
        OutputStreamWriter oos = null;
        StreamConnection soc=null;
        try
        {
            System.out.println(tbmain.getString());
soc=(StreamConnection)Connector.open("socket://"+tbmain.getString()+":500");
            oos= new OutputStreamWriter(soc.openOutputStream());
            oos.write(x);
            oos.flush();
            oos.close();
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
    public void run()
    {
        com1(controlVal);
        System.out.println(controlVal);
    }

    public void ShowImage(boolean flag)
    {
        if(flag==false)
        {
            showAlert("Download Failed", true, tbmain);
        }

        else if(flag==true)
        {
            ImageItem ii=new ImageItem(null,im,ImageItem.LAYOUT_DEFAULT,
null);
            if(frm.size()!=0)
                frm.set(0,ii);
            else
                frm.append(ii);
          display.setCurrent(frm);
        }
    }

    public void showAlert(String msg, boolean bool, Displayable displayable)
    {
        alStatus = new Alert("Status", msg, null, AlertType.INFO);
        if(bool)
            alStatus.setTimeout(Alerttime);
        else
            alStatus.setTimeout(Alerttime);

    }
}
```

```
class Download implements Runnable
{
        private String url;
        private ViewImage MIDlet;
        private boolean downloadSuccess = false;
        public Download(String url, ViewImage MIDlet)
        {
                this.url = url;
                this.MIDlet = MIDlet;
        }
        /*--------------------------------------------------
        * Download the image
        *-------------------------------------------------*/
        public void run()
        {
                try
                {
                        getImage(url);
                }
                catch (Exception e)
                {
                        System.err.println("Msg: " + e.toString());
                }
        }

        public void start()
        {
                Thread thread = new Thread(this);
                try
                {
                        thread.start();
                }
                catch (Exception e)
                {
                }
        }

          private void getImage(String url) throws IOException
        {
                ContentConnection connection = (ContentConnection)
Connector.open(url);
                DataInputStream iStrm = connection.openDataInputStream();
                ByteArrayOutputStream bStrm = null;
                Image im = null;
                try
                {

                        byte imageData[];
                        int length = (int) connection.getLength();
                        if (length != -1)
                        {
                                imageData = new byte[length];
                                iStrm.readFully(imageData);
                        }

                                else
                                {
```

```
                         bStrm = new ByteArrayOutputStream();
                         int ch;
                         while ((ch = iStrm.read()) != -1)
                         bStrm.write(ch);
                         imageData = bStrm.toByteArray();
                   }

                   im = Image.createImage(imageData, 0, imageData.length);
             }
             finally
             {

                   if (connection != null)
                   connection.close();
                   if (iStrm != null)
                   iStrm.close();
                   if (bStrm != null)
                   bStrm.close();
             }
                 if (im == null)
                   MIDlet.ShowImage(false);
            else
             {
                   MIDlet.im = im;
                   MIDlet.ShowImage(true);

             }

      }

}
/*
class Control implements Runnable
{
    int val;
    public Control(int val)
    {
        this.val=val;
        System.out.println("In Constructor :"+val);
        Thread t=new Thread();
        t.start();
    }


    public void com1(int x)
    {
        OutputStreamWriter oos = null;
        StreamConnection soc=null;
        //int x=100;
        try
        {

soc=(StreamConnection)Connector.open("socket://114.31.18.218:500");
            oos= new OutputStreamWriter(soc.openOutputStream());
            oos.write(x);
            oos.flush();
            oos.close();
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
    public void run()
    {
        com1(val);
        System.out.println(val);
    }
}*/
```

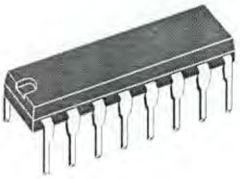
Appendix B

Datasheet for ULN 2003

ULN2001A-ULN2002A
ULN2003A-ULN2004A

SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A

SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

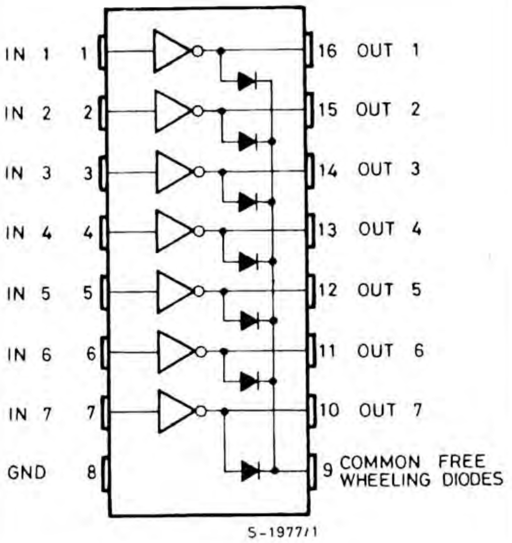
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

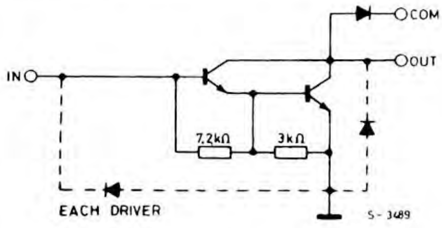
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

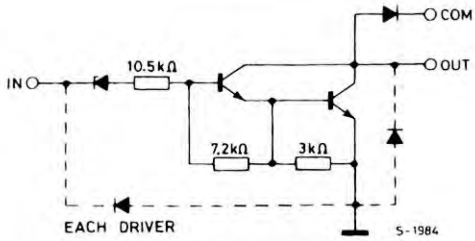
PIN CONNECTION



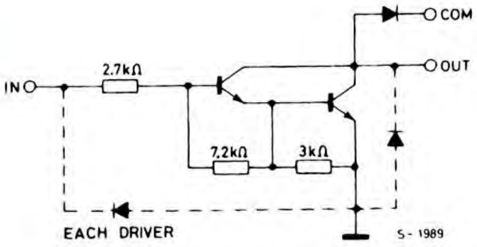
SCHEMATIC DIAGRAM



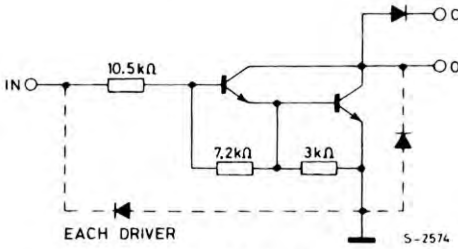
Series ULN-2001A
(each driver)



Series ULN-2002A
(each driver)



Series ULN-2003A
(each driver)



Series ULN-2004A
(each driver)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_{in}	Input Voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_c	Continuous Collector Current	500	mA
I_b	Continuous Base Current	25	mA
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
T_j	Junction Temperature	150	°C

THERMAL DATA

Symbol	Parameter		DIP16	SO16	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	70	120	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V T_{amb} = 70°C, V_{CE} = 50V T_{amb} = 70°C for ULN2002A V_{CE} = 50V, V_i = 6V for ULN2004A V_{CE} = 50V, V_i = 1V			50 100 500 500	µA µA µA µA	1a 1a 1b 1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA I_C = 200 mA, I_B = 350µA I_C = 350mA, I_B = 500µA		0.9 1.1 1.3	1.1 1.3 1.6	V V V	2 2 2

$I_{i(on)}$	Input Current	for ULN2002A, V_i = 17V
		for ULN2003A, V_i = 3.85V
		for ULN2004A, V_i = 5V
		V_i = 12V
				0.82
				0.93
				0.35
				1
					1.25
					1.35
					0.5
					1.45
						mA
						mA
						mA
						mA
							3
							3
							3
							3

$I_{i(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{i(on)}$	Input Voltage	V_{CE} = 2V for ULN2002A I_C = 300mA for ULN2003A I_C = 200mA I_C = 250mA I_C = 300mA for ULN2004A I_C = 125mA I_C = 200mA I_C = 275mA I_C = 350mA			13 2.4 2.7 3 5 6 7 8	V	5
h_{FE}	DC Forward Current Gain	for ULN2001A V_{CE} = 2V, I_C = 350mA	1000				2
C_i	Input Capacitance			15	25	pF	
t_{PLH}	Turn-on Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
t_{PHL}	Turn-off Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
I_R	Clamp Diode Leakage Current	V_R = 50V T_{amb} = 70°C, V_R = 50V			50 100	µA µA	6 6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7

TEST CIRCUITS

Figure 1a.

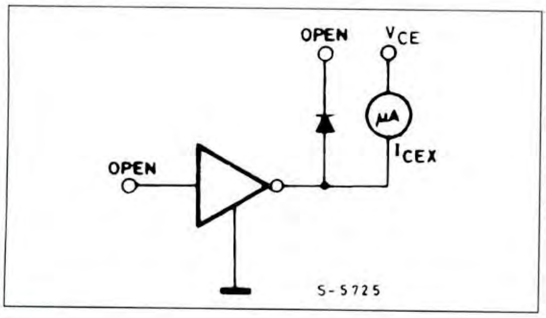


Figure 1b.

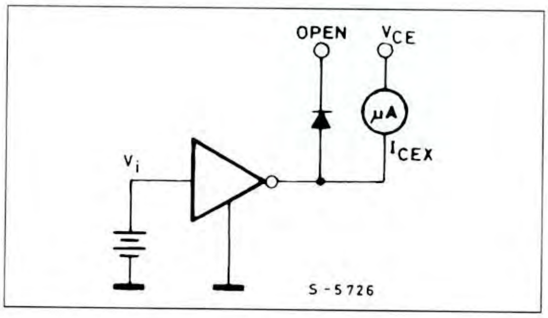


Figure 2.

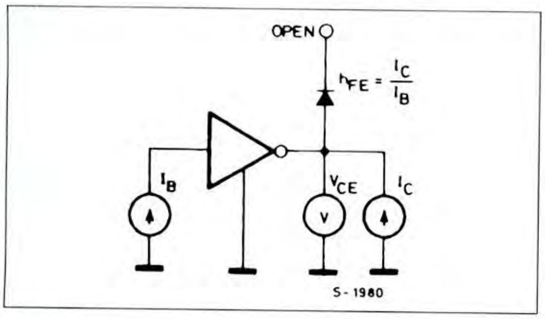


Figure 3.

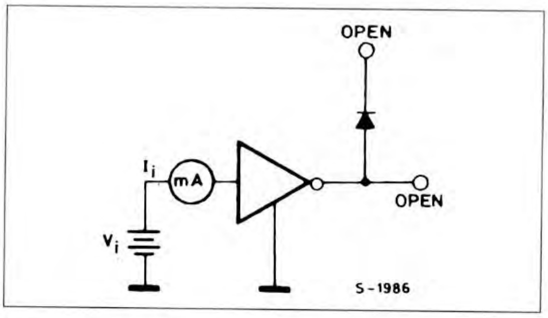


Figure 4.

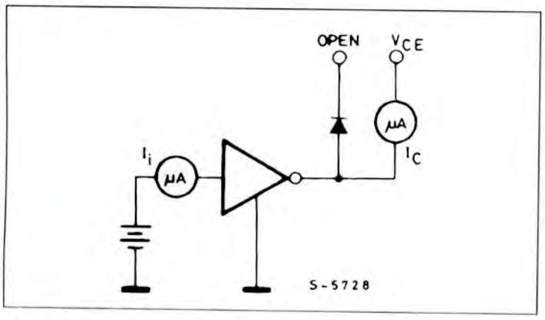


Figure 5.

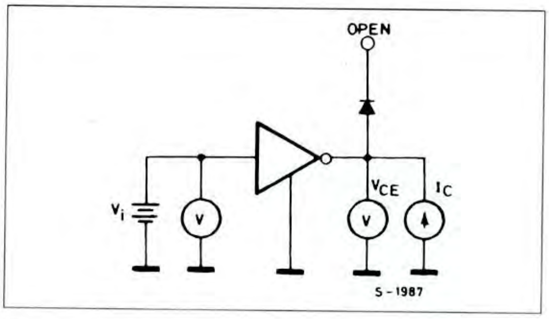


Figure 6.

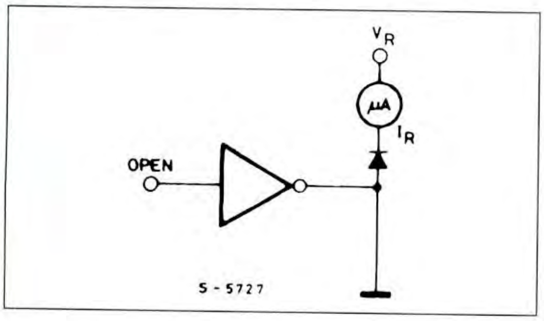


Figure 7.

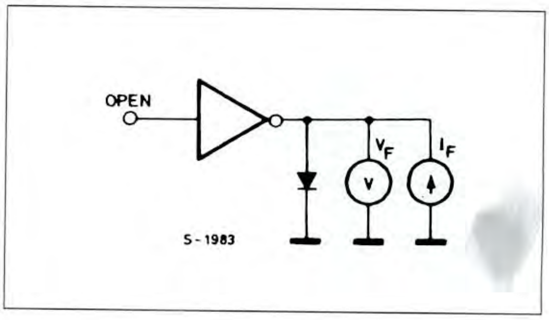


Figure 8: Collector Current versus Input Current

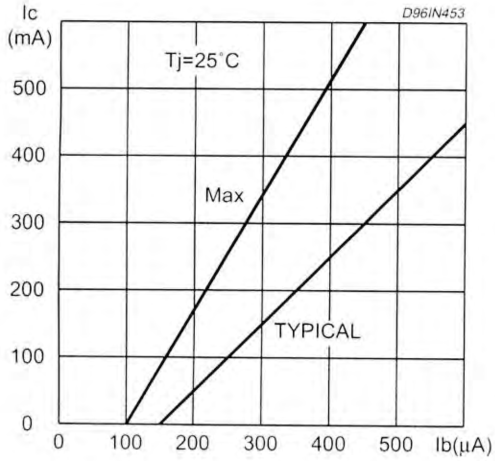


Figure 9: Collector Current versus Saturation Voltage

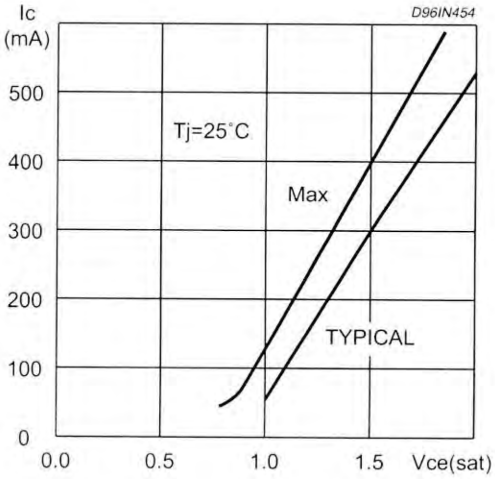


Figure 10: Peak Collector Current versus Duty Cycle

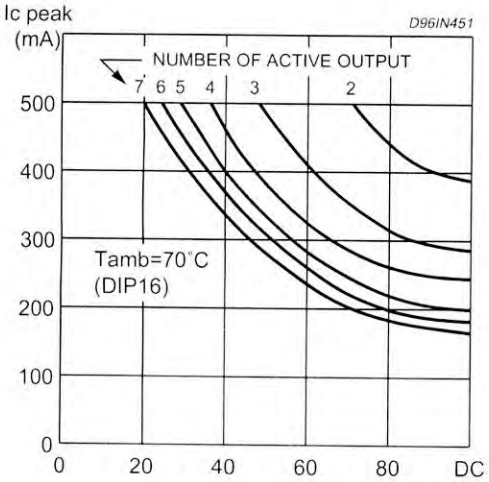
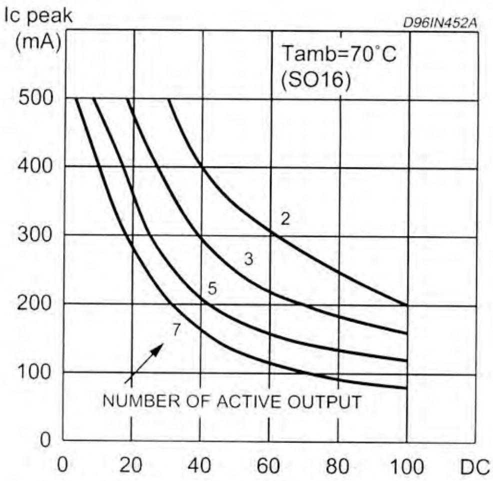


Figure 11: Peak Collector Current versus Duty Cycle


D96IN452A
Ic peak
(mA)
Tamb=70°C
(SO16)
NUMBER OF ACTIVE OUTPUT
DC