P-2178

REMOTE APPLIANCE CONTROL THROUGH PERSONAL COMPUTER

A PROJECT REPORT

Submitted By

Madhumitha.B	**71204205020**
Nethra.M	**71204205024**
Nithya.V	**71204205025**
Vidyalakshmi.G	**71204205058**

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"REMOTE APPLIANCE CONTROL THROUGH PERSONAL COMPUTER"** is a bonafide work of **"MADHUMITHA B, NETHRA M, NITHYA V, VIDYALAKSHMI G"** who carried out the project work under my supervision.

[illegible]	[illegible]
SIGNATURE	**SIGNATURE**
Dr.S.Thangaswamy	Mrs.N.Suganthi
DEAN	**SUPERVISOR**
	Senior Lecturer,
Computer Science and Engineering	Information Technology
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore-641006	Coimbatore-64106

The candidates with University Register Nos **71204205020, 71204205024, 71204205025, 71204205058** were examined by us in the project viva-voice examination held on .24..04..08

[illegible]	[illegible] 24/04/08
INTERNAL EXAMINER	EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr.JOSEPH V.THANIKAL, B.E, M.E, Ph.D, PDF. CEPIT,** Principal, Kumaraguru College of Technology for having given us a golden opportunity to embark on this project.

We are deeply obliged to **Dr.S.THANGASAMY**, Dean of Computer Science and Engineering for his concern and implication during the project course.

We extend our thanks to our project co-coordinator **Prof.K.R.BASKARAN**, Asst Professor, Department of Information Technology and our guide **Mrs.N.SUGANTHI**, Senior Lecturer, Department of Information Technology for their helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staff of our department for providing us the technical support during our project so far and friends for their timely help that culminated as good in the end we also thank our parents without whom we couldn't have come.

ABSTRACT

ABSTRACT

Remote appliance control through personal computer is a system that can monitor and control home appliances from a remote location. The design is based on a stand alone embedded system board integrated into a PC-based server at home. The home appliances are connected to the input/output ports of the embedded system board and their status are passed to the server. The monitoring and control is based on Virtual Network Computing and Interactive C. The home appliances can be monitored and controlled locally via the embedded system board, or from a remote desktop. The system is scalable and allows multi-vendor appliances to be added with no major changes to its core. Password protection is used to block unauthorized users from accessing the appliances at home. The embedded system board can control and operate the appliances locally.

TABLE OF CONTENTS

APTER No.	TITLE	P.No.
	ABSTRACT	iii
	LIST OF TABLES	iv
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1.	**INTRODUCTION**	
	1.1 HOME AUTOMATION	1
	1.2 PROBLEM DEFINITION	1
	1.3 OBJECTIVE	2
2.	**LITERATURE REVIEW**	
	2.1 CURRENT STATUS OF THE PROBLEM	3
	2.2 PROPOSED SYSTEM AND ITS ADVANTAGES	3
	2.2.1. Remote access	3
	2.2.2. E-Board access	3
	2.2.3. Check status	3
	2.2.4. Authentication	3
	2.3 BLOCK DIAGRAM	4
	2.4 CIRCUIT DIAGRAM	5
	2.5 HARDWARE REQUIREMENTS	7
	2.6 SOFTWARE REQUIREMENTS	7

	2.7 SOFTWARE OVERVIEW	8
	2.7.1. Cross compilers	
	2.7.2 Anyplace control	9
	2.7.2.1 Features of anyplace control	9
3.	**DETAILS OF METHODOLOGY EMPLOYED**	
	3.1 HARDWARE ASSEMBLY	11
	3.2 PORT COMMUNICATION	22
4.	**PERFORMANCE EVALUATION**	
	4.1 TASK TESTING	26
	4.2 BEHAVIORAL TESTING	26
	4.3 INTERTASK TESTING	26
	4.4 HARDWARE TESTING	26
5.	**CONCLUSION**	27
6.	**FUTURE ENHANCEMENT**	28
7.	**APPENDIX**	
	7.1. APPENDIX 1- SOURCE CODE	29
	7.2 APPENDIX2-SCREEN SHOTS	49
	7.3 APPENDIX 3- HARDWARE PHOTO	53
8.	**REFERENCES**	54

LIST OF TABLES

No.	TABLE NAME	Table No.	P.No
	Software requirements	2.6.1	7
	AT89 Series Microcontrollers	3.1	12
	Pin Description of AT89C51	3.2	16
	Function table of DM74157	3.3	20
	Base Address and Interrupt number of ports	3.4	23
	Parallel Port DB- 25 pin Description	3.5	24
	D-Type-9 Pin Description	3.6	25

LIST OF FIGURES

o.	FIGURE NAME	FIG.No.	P. No.
	Block Diagram	2.3.1	4
	Circuit diagram	2.4.1	5
	Anyplace Control	2.7.2.1	9
	Pin diagram of AT89C51	3.1	13
	Power supply (5V)	3.2	17
	Power supply (12V)	3.3	17
	Positive Voltage Regulator	3.4	18
	Relay as Switch	3.5	22

LIST OF ABBREVIATION

E-BOARD	Electronic Board
PC	Personal Computer
MUX	Multiplexer
LCD	Liquid Crystal Display
LAN	Local Area Network
EEPROM	Erasable Electrically Programmable Read Only Memory
CMOS	Complementary Metal Oxide Semi Conductor
CPU	Central Processing Unit
AC	Alternate Current
DC	Direct Current
IC	Integrated Circuit
IP	Internet Protocol

INTRODUCTION

1. INTRODUCTION

1.1 HOME AUTOMATION

Current Scenario demands that an individual's focus be mainly directed towards his/her time management at the most effective way. Mostly people look forward for facilities that can be remotely controlled thus eliminating the need for physical presence of the person and save time from waiting, in the most economical way.

1.2 PROBLEM DEFINITION

Remote appliance control through personal computer is a system that can monitor and control home appliances from a remote location. Appliances at home are connected to an embedded system board (E-board). The user can interact with the home automation system from anywhere at any time.

The home server is a high-end PC that hosts the management and control algorithm that enables the user to access the home appliances remotely. It also communicates with the E-board via the parallel port to download the control commands and via the serial port to upload the appliance's status. Using the E-board to control the appliances gives the user the ability to control the home locally even if the PC is not ON. The E-board has digital input and output ports, memory, an expansion slot, a two-line 16- digit LCD and extra hardware resources which make it suitable for the required task.

Home appliances are connected to the digital output of the E-board via relays to provide sufficiently high currents and voltage compatibility. A dual multiplexer, MUX, is used to enable the user to switch between local and remote mode using the multiplexer selection line. If the selection line is high, the local mode is enabled and the E-board processes the user's request based on the input commands from MUX's local inputs and the server is no longer communicating with the E-board. Otherwise, the remote mode is active and the commands are received by the E-board from the server's parallel port.

Sending a software command to turn a device ON/OFF may not grantee the successful operation of the device as the device may be defect. To overcome this problem, a feedback circuit has been designed and implemented to indicate the device's actual status after it received the software command (ON/OFF). Once a command is sent to turn a device ON, the feedback circuit senses the current and gives an output signal indicating that the device is ON. Otherwise, the device is not functioning and a message will pop up informing the user that the command was not executed successfully. The main sensing element is a current transformer and signal conditioning circuit that will output a digital signal indicating if the device responded to the command correctly.

1.3 OBJECTIVE

The objective of this project is to enable the user to ON/OFF the home appliances from a remote location anywhere at anytime. It must also enable the user to view the status of the appliances. The E-Board is used to control the system even when the PC is not ON. The status of the appliances is viewed on the LCD screen also.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 CURRENT STATUS OF THE PROBLEM

The existing system does provide access to home appliances from a remote location. No status checking facility is available to know what is happening on the other side. Authentication is not provided in the present scenario.

2.2 PROPOSED SYSTEM AND ITS ADVANTAGES

2.2.1 REMOTE ACCESS

The user can access the home appliances from a remote location anywhere and at any time.

2.2.2. E-BOARD ACCESS

The E board gives the user the ability to control the home appliances even if remote access is not available or the PC is not ON.

2.2.3. CHECK STATUS

The status of the appliances before and after its being accessed is returned to the user both at the local machine and remote desktop. A feed back loop senses the current to the appliances and displays the status.

2.2.4. AUTHENTICATION

Only the user who knows the password can control the devices.

2.3 BLOCK DIAGRAM

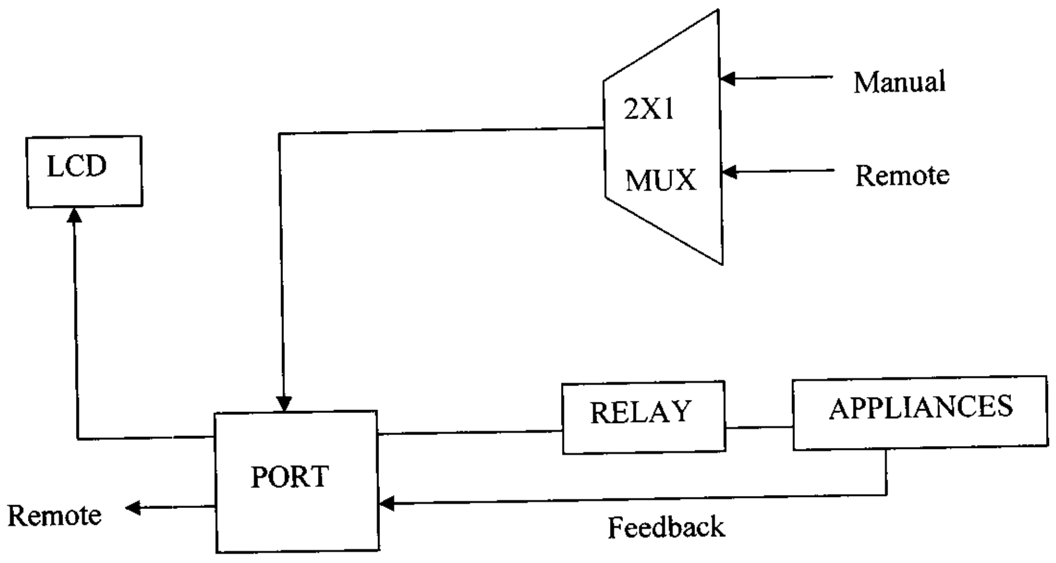


Figure 2.3.1 Block Diagram

Figure 2.3.1 Shows The Basic Functioning Of The Hardware Module

The 2x1 mux is used as a selection line (1 for manual, 0 for automatic). If the selection is through internet only the authenticated user can access. The valid user's command is identified by the computer and the corresponding value is send to the microcontroller through the parallel port. Relays act as an electromechanical switch. The feedback is sent to the microcontroller by means of the serial port. Control can also be done from the computer apart from the E-board. LCD displays the status of the appliances.

2.4 CIRCUIT DIAGRAM

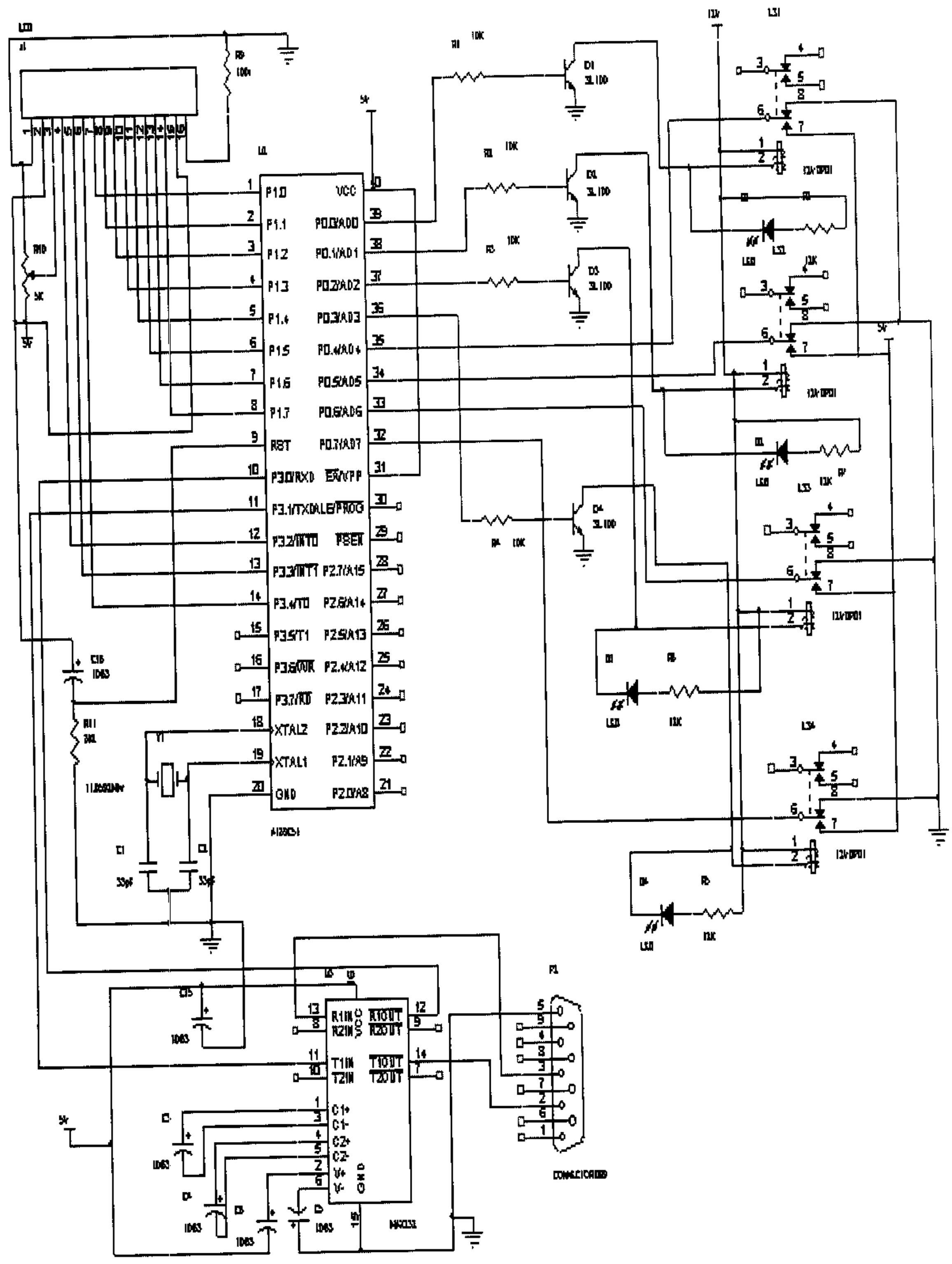


Figure 2.4.1 Communication Establishment

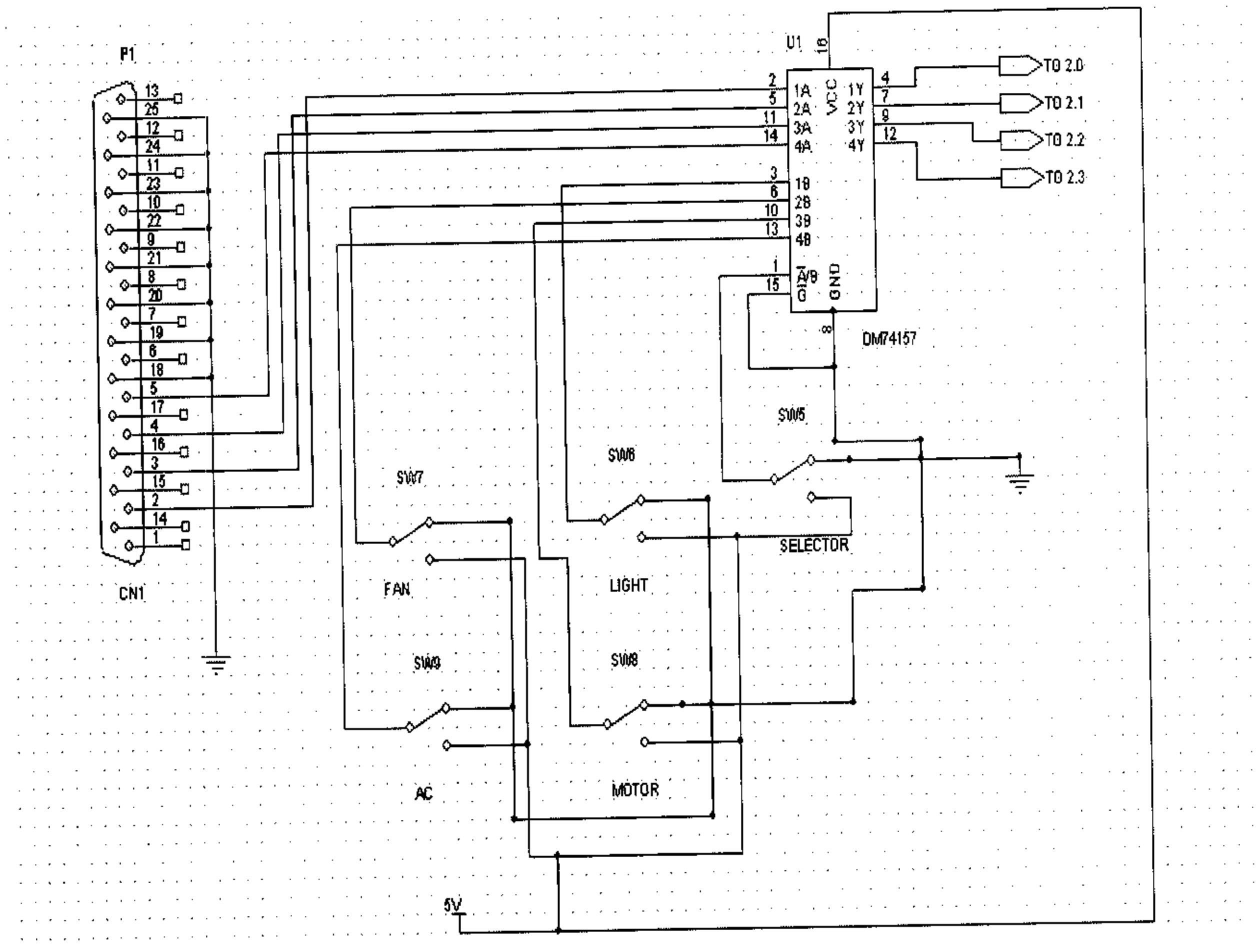


Figure 2.4.2 Display Circuit

2.5 HARDWARE REQUIREMENTS

- Microcontroller
- Multiplexer
- Relay
- LCD display
- Switch
- Ports
- Rectifiers
- Filters
- Transformers
- Power supply

2.6 SOFTWARE REQUIREMENTS

Operating System	Windows
Language	C
Application	Anyplace control
Tools	Cross Compiler

Table 2.6.1 Software Requirements

2.7 SOFTWARE OVERVIEW

2.7.1 CROSS COMPILERS

A cross compiler is a compiler capable of creating executable code for a platform other than the one in which the compiler is run. Cross compiler tools are generally found in use to generate compiles for embedded system or multiple platforms. It is a tool that one must use for a platform where it is inconvenient or impossible to compile on that platform, like microcontrollers that run with a minimal amount of memory for their own purpose.

The Cx51 optimizing C compiler is a complete implementation of the American National Standards Institute (ANSI) standard for the C language. Cx51 is a cross compiler, some aspects of the C programming language and standard libraries are altered or enhanced to address the peculiarities of an embedded target processor. Cx51 is not a universal C compiler adapted for the 8051 target. It is a ground up implementation dedicated to generating extremely fast and compact code for the 8051 microcontroller.

2.7.2. ANYPLACE CONTROL

Anyplace control is remote control software that allows controlling a remote PC over LAN or internet. It displays the remote computer's desktop on the local screen and lets us use the mouse and keyboard to control that PC remotely, in other words we could tell that one can operate a remote PC as if he/she is sitting in front of it no matter where the user is.

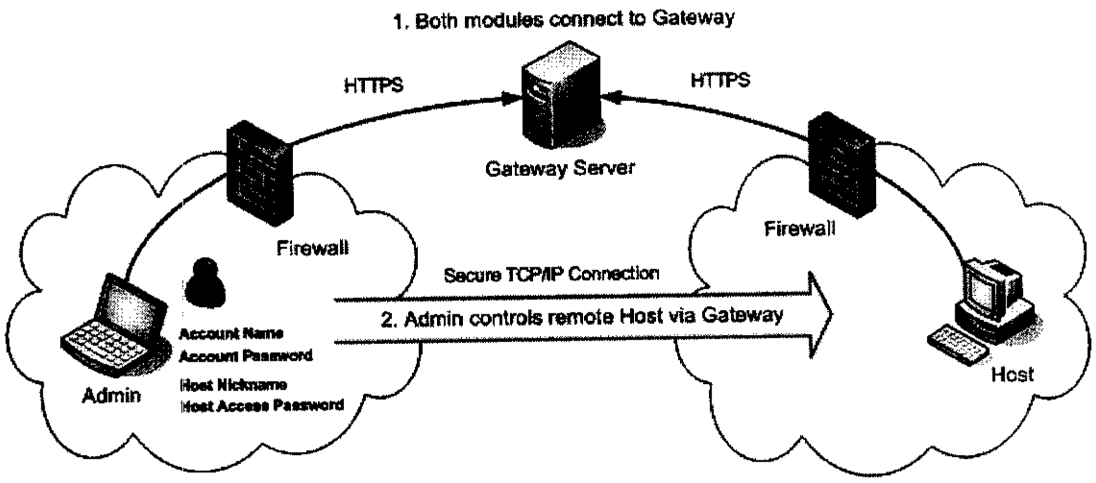


Figure 2.7.2 Anyplace Control

2.7.2.1 FEATURES OF ANYPLACE CONTROL

- Displays remote computer's desktop on the user's screen in
- User can use his keyboard and muse to operate other PC remotely.
- Transfer file to or from the remote computer.
- Turn ON, turn OFF or reboot remote PC.
- Access remote PC behind the router or firewall without any configuring.

DETAILS OF METHODOLOGY EMPLOYED

3. DETAILS OF THE METHODOLOGY EMPLOYED

The various modules involved in our project are

- Hardware assembly
- Embedding software
- Port communication
- Anyplace control
- Integration of manual and automatic functioning

3.1HARDWARE ASSEMBLY

MICROCONTROLLER

A microcontroller is a computer-on-a-chip. It is a type of microprocessor emphasizing high integration, low power consumption, self-sufficiency and cost-effectiveness, in contrast to a general-purpose microprocessor. In addition to the usual arithmetic and logic elements of a general purpose microprocessor, the microcontroller typically integrates additional elements such as read-write memory for data storage, read-only memory, such as flash for code storage, EEPROM for permanent data storage, peripheral devices, and input/output interfaces. They consume relatively little power (mill watts), and will generally have the ability to sleep while waiting for an interesting peripheral event such as a button press to wake them up again to do something. Power consumption while sleeping may be just nano watts, making them ideal for low power and long lasting battery applications. By reducing the size, cost, and power consumption compared to a design using a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to electronically control many more processes.

ATMEL SERIES

The Atmel AT89 series is an Intel 8051-compatible family of 8 bit microcontrollers manufactured by the Atmel Corporation. Based on the Intel 8051 core, the AT89 series remains very popular as general purpose microcontrollers, due to their industry standard instruction set, and low unit cost. This allows a great amount of legacy code to be reused without modification in new applications.

Device Name	Program Memory	Data Memory
AT89C1051	1K Flash	64 RAM
AT89C2051	2K Flash	128 RAM
AT89C51	4K Flash	128 RAM
AT89C52	8K Flash	256 RAM
AT89C55	20K Flash	256 RAM
AT89S8252	8K Flash	256 RAM
AT89S53	12K Flash	256 RAM

Table 3.1 AT89 Series Microcontrollers

AT89C51

In our project AT89C51 is used. It is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pin out.

The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to our control application.

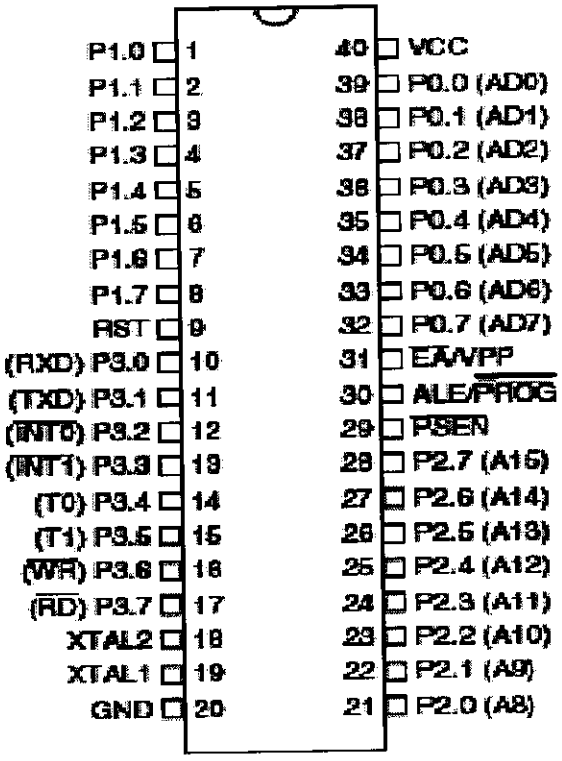


Figure 3.1 Pin diagram of AT89C51

Pin number	**Pin name**	**Function**
32-39	Port 0	8-bit open drain bidirectional I/O port. When 1s are written to port 0 pins, the pins can be used as high impedance inputs. Port 0 may also be configured to be the multiplexed low order address / data bus during accesses to external program and data memory. Port 0 receives the code bytes during Flash programming, and outputs the code bytes during program verification.
1-8	Port 1	8-bit bidirectional I/O port with internal pull-ups. Port 1 receives the low-order address bytes during Flash programming and verification.
21-28	Port 2	8-bit bidirectional I/O port with internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses. During accesses to external data memory that use 8-bit addresses, Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

10	P3.0 (RXD)	Serial Input Port
11	P3.1 (TXD)	Serial Output Port
12	P3.2 ($\overline{\text{INT 0}}$)	External interrupt 0
13	P3.3 ($\overline{\text{INT 1}}$)	External interrupt 1
14	P3.4 (T0)	Timer 0 external interrupt
15	P3.5 (T1)	Timer 1 external interrupt
16	P3.6 ($\overline{\text{WR}}$)	external data memory write strobe
17	P3.7 ($\overline{\text{RD}}$)	external data memory read strobe
9	RST	Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.
18	XTAL2	Output from the inverting oscillator amplifier.
19	XTAL1	Input to the inverting oscillator amplifier and input to the internal clock operating circuit.
20	GND	Ground

29	$\overline{\text{PSEN}}$	Program Store Enable is the read strobe to external program memory. When the AT89C51 is executing code from external program memory, PSEN is activated twice each machine cycle.
30	ALE/$\overline{\text{PROG}}$	Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming.
31	$\overline{\text{EA}}$/Vpp	External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH
40	Vcc	Supply Voltage

Table 3.2 Pin Description of AT89C51

POWER SUPPLY

The power supply circuit is built using filters, rectifiers and the voltage regulators. Starting with an AC voltage, a steady DC voltage is obtained by rectifying the AC voltage, then filtering to a DC level, and finally regulating to obtain a desired fixed DC voltage. The regulation is usually obtained from an IC voltage regulator unit, which takes a DC voltage and provides a somewhat lower DC voltage, which remains the same even if the input DC voltage varies, or the output load connected to the DC voltage changes.

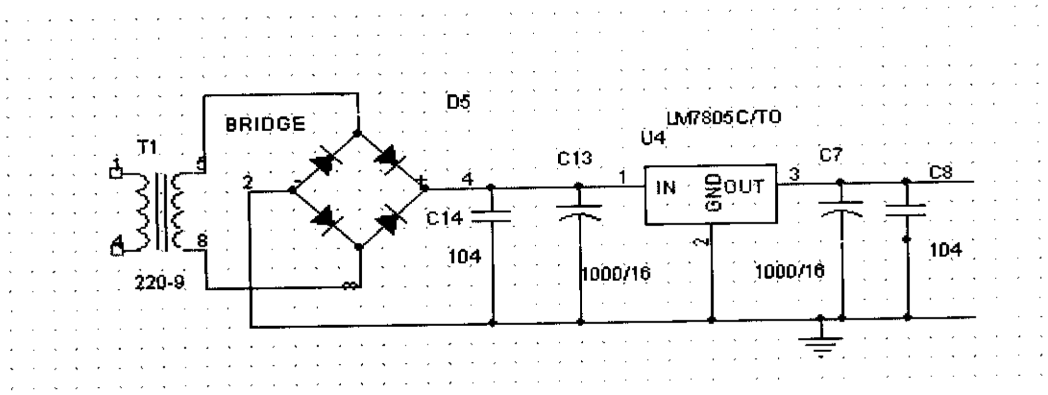


Fig 3.2 Circuit Diagram – Regulated Power Supply for 5V

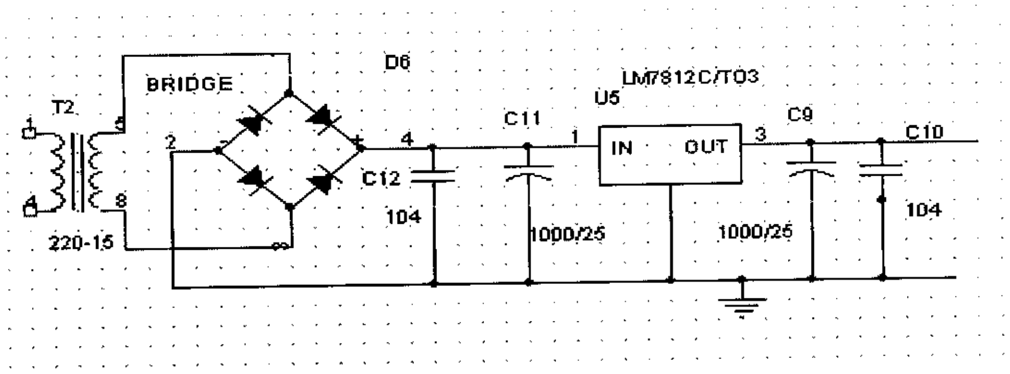


Fig 3.3 Circuit Diagram – Regulated Power Supply for 12 V

IC VOLTAGE REGULATORS

IC voltage regulators are three-terminal devices that provide a constant DC output voltage that is independent of the input voltage, output load current, and temperature. There are three types of IC voltage regulators: IC linear voltage regulators, IC switching voltage regulators, and DC/DC converter chips. IC linear voltage regulators use an active pass element to reduce the input voltage to a regulated output voltage. IC switching voltage regulators store energy in an inductor, transformer, or capacitor and then use this storage device to transfer energy from the input to the output in discrete packets over a low-resistance switch. DC/DC converter chips also provide a regulated DC voltage output from a different, unregulated input voltage. For each type of IC voltage regulator, the output voltage can be fixed or adjusted to a value within a specified range. The regulators can be selected for operation with load currents from hundreds of milli amperes to tens of amperes, corresponding to power ratings from milli watts to tens of watts.

FIXED POSITIVE VOLTAGE REGULATORS

The series 78 regulators provide fixed voltages from 5 to 24 V. Figure 3.4 shows how one such IC, a 7805 is connected to provide voltage regulation with output from this unit of +5V DC. An unregulated input voltage Vi is filtered by capacitor C1 and connected to the IC's IN terminal. The IC's OUT terminal provides a regulated +5V which is filtered by capacitor C2. The third IC terminal is connected to ground. While the input voltage may vary over some permissible voltage range, and the output load may vary over some acceptable range, the output voltage remains constant within specified voltage variation limits.

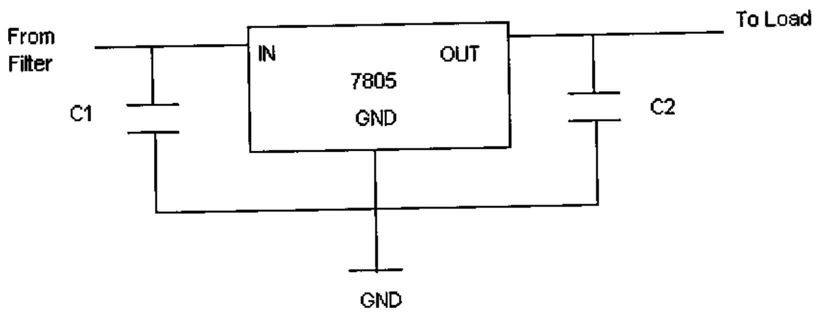


Fig 3.4 Positive Voltage Regulator

TRANSFORMER

A transformer is a static (or stationary) piece of which electric power in one circuit is transformed into electric power of the same frequency in another circuit. It can raise or lower the voltage in a circuit but with a corresponding decrease or increase in current. It works with the principle of mutual induction. In our project we are using step down transformer for providing a necessary supply for the electronic circuits.

RECTIFIER

The DC level obtained from a sinusoidal input can be improved 100% using a process called full-wave rectification. It uses four diodes in a bridge configuration. From the basic bridge configuration we see that two diodes (D2 and D3) are conducting while other two diodes (D1 and D4) are in "off" state during the period t=0 to T/2. for negative of the input the conducting diodes are D1 and D4.

FILTER

The filter circuit used here is capacitor filter circuit, where a capacitor is connected at the rectifier output, and a DC is obtained across it. The filtered waveform is a DC voltage with negligible ripples, which is ultimately fed to the load.

RESISTOR

A resistor is a two-terminal electrical or electronic component that opposes an electric current by producing a voltage drop between its terminals in accordance with Ohm's law: V=IR. Resistors are used as part of electrical networks and electronic circuits.

MULTIPLEXER

A multiplexer is a device that performs multiplexing; it selects one of many analog or digital input signals and outputs that into a single line. A 2n-to-1 multiplexer routes one of 2n input lines to a single output line. Multiplexers can implement arbitrary functions. The multiplexers contain inverters and drivers to supply full on-chip data selection to four output gates. In our project we use the 7400 series of Transistor-transistor Logic integrated circuit. DM74157 is used because its propagation time is 9ms and power dissipation is 150mW. In the multiplexer a 4-bit word is selected from one of the two sources and is routed to the four outputs.

Inputs				**Output Y**
Strobe	**Select**	**A**	**B**	
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

Table 3.3 Function Table Of DM74157

SWITCH

A simple semiconductor switch is a transistor. In the simplest case, a switch has two pieces of metal called contacts that touch to make a circuit, and separate to break the circuit. The moving part that applies the operating force to the contacts is called the actuator

LCD DISPLAY

An 8051 programmer must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common devices attached to an 8051 is an LCD display. Liquid crystal displays have materials which combine the properties of both liquids and crystals. A standard referred as HD44780U receives data from the external source (in this case, the 8051), and communicates directly with the LCD.

The 44780 requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).

The three control lines are referred to as EN, RS and RW. The EN line is called “Enable”. It is used to tell the LCD that data is being sent. Inorder to send data the program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. The RS line is called “Register Select” line. When RS is low (0), the data is to be treated as a command or special instruction. When RS is high (1), the data being sent is text data which should be displayed on the screen. The RW line is the “Read/Write” control line. When RW is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Finally the data bus consists of 4 or 8 lines depending on the mode of operation selected by the user. In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB 3, DB4, DB5, DB6 and DB7.

RELAY

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil currents can be on or off. So relays have two switch positions and they are double throw (change over) switches.

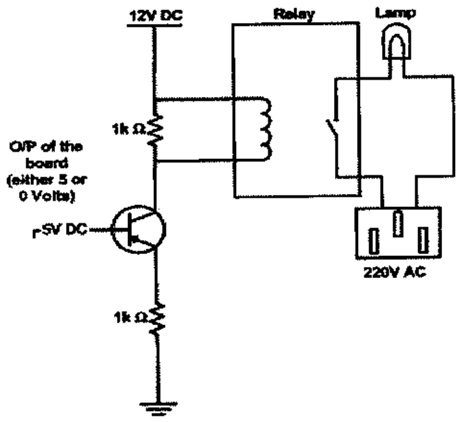


Fig 3.5 Relay as Switch

In our project, if the input is high, then the device will be ON. If input is low, then the device will be OFF.

3.2 PORT COMMUNICATION

SERIAL COMMUNICATION

To transfer data to a device located many meters away, the serial data transfer is used. In serial communication, the data is sent 1 bit at a time. For serial data communication to work, the byte of data must be converted to serial bits using a parallel-in-serial-out shift register because the microcontroller is sending parallel data. Then it can be transmitted over a single data line. This also means at the receiving end there must be a serial-in-parallel-out shift register.

RS232 STANDARDS

RS232 is the most widely used serial input/output interfacing standard. This standard is used in PC and other equipment. In RS232 a '1' is represented by -3 to -25 V while a '0' is +3 to +25V, making -3 to +3 undefined. For this reason to connect any RS232 to a microcontroller we must use voltage converters such as MAX232.

COM PORTS

COM ports are designed as COM1 and COM2. We connect the AT89C51 serial port to the COM1 or COM2 of a PC for serial communication data transfer.

PORT	BASE ADDRESS	INTERRUPT NUMBER
COM1	0x03F8	0x00C
COM2	0x02F8	0x00B
COM3	0x03E8	0x00C
COM4	0x02E8	0x00B

Table 3.4 Base Address and Interrupt number of ports

PARALLEL PORT

A parallel port is a type of interface found on computers for connecting various peripherals. It is also known as a printer port or Centronics port. The IEEE 1284 standard defines the bi-directional version of the port. The parallel port connector is in the rear panel of PC. It is a 25 pin female (DB25) connector.

Pin	Signal
1	Strobe
2-7	Data 0 -7
10	Acknowledge
11	Busy
12	Paper End
13	Select
14	Auto Feed
15	Error
16	Init
17	Select In
18-25	GND

Table 3.5 Parallel Port DB- 25 pin description

SERIAL PORT

A serial port is a asynchronous serial communication physical interface through which information transfers in or out one bit at a time. Voltage sent over the pins can be in one of two states, On or Off. On (binary value "1") means that the pin is transmitting a signal between -3 and -25 V, while Off (binary value "0") means that it is transmitting a signal between +3 and +25 V.

Serial Ports come in two pin configurations. They are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC, thus we will require a female connector on our device. We use the D-Type 9 pin connector.

PIN NO.	SIGNAL	DESCRIPTION
1	Carrier Detect (CD)	Carrier Detect When the modem detects a "Carrier" from the modem at the other end of the phone line, this Line becomes active
2	Receive Data (RD)	Receive Data Serial Data Input
3	Transmit Data (TD)	Transmit Data Serial Data Output
4	Data Terminal Ready (DTR)	This tells the Modem that the UART is ready to link
5	Signal Ground (SG)	This grounds the signal
6	Data Set Ready (DSR)	This tells the UART that the modem is ready to establish a link
7	Request To Send (RTS)	This line informs the Modem that the UART is ready to exchange data
8	Clear To Send (CTS)	This line indicates that the Modem is ready to exchange data.
9	Ring Indicator (RI)	Goes active when modem detects a ringing signal from the PSTN

Table 3.6 D-Type-9 Pin Description

PERFORMANCE EVALUATION

4. PERFORMANCE EVALUATION

4.1 TASK TESTING

The first step in the testing of software is to test each task independently. Each task is executed independently during these tests. In our project each task say hardware operation, remote control is done separately. Task testing uncovers errors in logic and function but not timing or behavior.

4.2 BEHAVIORAL TESTING

Each of the events is tested individually and the behavior of the executable system is examined to detect errors that occur as a consequence of processing associated with these events. The behavior of the software is examined to detect behavior errors.

4.3 INTERTASK TESTING

Once errors in individual tasks and in system behavior have been isolated, testing shifts to time-related errors. Asynchronous tasks that are known to communicate with one another are tested with different data rates and processing load to determine if intertask synchronization errors will occur.

4.4 HARDWARE TESTING

The E-board is tested with that of many PC's to determine the port communication and the working of Anyplace Control with various IP address.

CONCLUSION

5. CONCLUSION

Overall we are happy with the system we developed. It works quite well for its simplicity and lack of any more complicated techniques, though it is not super simple.

The advantage of remote appliance control is very cheap hardware and remote access provided with security. It is very easy to use.

Since many devices can be connected at the same time this project will be extremely useful in the industry as many parameters of a particular process can be controlled at the same time or various process can be done at the same time or various process can be done at the same time making multi tasking possible. Hence the project is social and commercial.

FUTURE ENHANCEMENT

6. FUTURE ENHANCEMENT

The control of appliances works well in a wired environment. This could be extended to that of a wireless environment with new concepts like zigbee, where the appliances could be controlled for over 10 meters. Even ordinary domestic works like watering of plants, could be added as new features.

Features like controlling the speed of the fan or the heat in an oven can be added without much change to the core.

Burglar alarm systems can be integrated with this system to enable support for physical support to the appliances.

APPENDIX

7. APPENDIX

7.1 APPENDIX 1- SOURCE CODE

7.1.1 LCD DISPLAY

```
_rom unsigned char n1[]=" Welcome to home ";
_rom unsigned char n2[]="   automation    ";
_rom unsigned char st[]=" getting status  ";
_rom unsigned char ato[]="   Auto mode    ";
_rom unsigned char man[]="  Manual mode   ";
_rom unsigned char sts[]="status:";
_rom unsigned char on[] = "1 ";
_rom unsigned char off[] = "0 ";
_data unsigned char i,j,dly_cnt;
```

7.1.2 USERINTERFACE

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <bios.h>

void interface(void);
void control(void);
extern void send(void);
extern int ch;
```

```
void interface()
{
            setbkcolor(BLACK);
            setfillstyle(1,RED);
            bar(10,10,20,460);
            bar(10,10,620,20);
            bar(610,20,620,460);
            bar(10,450,620,460);
            setcolor(LIGHTBLUE);
            settextstyle(1,0,0);
            setusercharsize(2,1,3,2);
            outtextxy(160,50,"REMOTE APPLIANCE CONTROL
      THROUGH PC");
            setfillstyle(1,LIGHTRED);
            bar(40,345,60,365);
            setfillstyle(1,LIGHTGREEN);
            bar(40,395,60,415);
            setcolor(YELLOW);
            settextstyle(1,0,0 );
            setusercharsize(1,2,1,2);
            outtextxy(70,350,"OFF STATE ");
            outtextxy(70,400,"ON STATE ");
            outtextxy(450,350,"TO OFF DEVICE ");
            outtextxy(450,380,"TO ON DEVICE ");
            outtextxy(450,410,"TO SELECT DEVICE ");
            outtextxy(225,300,"PRESS F1 TO EXIT ");
            setcolor(GREEN);
            outtextxy(180,320,"PRESS F2 TO KNOW THE
        STATUS");
            setcolor(CYAN);
            settextstyle(2,0,0 );
            setusercharsize(2,1,2,1);
            outtextxy(170,350,"UP ARROW KEY          : ");
            outtextxy(170,380,"DOWN ARROW KEY        : ");
            outtextxy(170,410,"RIGHT & LEFT ARROW KEY: ");
            setcolor(LIGHTGRAY);
            settextstyle(2,0,0 );
            setusercharsize(1,1,1,1);
            outtextxy(155,160,"DEV-1  DEV-2  DEV-3  DEV-4");
            outtextxy(125,260,"DEV-1   DEV-2   DEV-3  DEV-4
        A/M");
            }
void control()
{
        int x1=150,y1=100,x2=200,y2=150,key;
        int d0=0,d1=0,d2=0,d3=0,d4=0,d5=0,d6=0,d7=0;
        int value=0,i,chk;
        int x1b,y1b,x2b,y2b;
        outportb(0x378,value);
         for(i=1;i<=4;i++)
         {
             setfillstyle(1,LIGHTRED);
             bar(x1,y1,x2,y2);
             x1=x1+70;
            x2=x2+70;
        }
        x1=120;
        y1=200;
        x2=170;
        y2=250;
        for(i=1;i<=5;i++)
        {
            setfillstyle(1,LIGHTRED);
            bar(x1,y1,x2,y2);
            x1=x1+70;
            x2=x2+70;
        }
        x1=150,x2=200;
        y1=100;y2=150;
        setcolor(WHITE);
        rectangle(x1-2,y1-2,x2+2,y2+2);
x:
        key=bioskey(0);
        if(key==15104)
        {
        closegraph();
        exit(0);
        }
        if(key==15360)
        {
        send();
```

```
      x1b=x1;
      y1b=y1;
      x2b=x2;
      y2b=y2;
       x1=120;
       y1=200;
       x2=170;
       y2=250;
       chk = 1;
       for(i=1;i<=5;i++)
       {
              if(ch&chk)
              setfillstyle(1,LIGHTGREEN);
              else
              setfillstyle(1,LIGHTRED);
              bar(x1,y1,x2,y2);
              x1=x1+70;
              x2=x2+70;
              chk=chk << 1;
       }
       x1=x1b;
       y1=y1b;
       x2=x2b;
       y2=y2b;
goto x;


       }
```

```
        if(key==20480)
        {
                setfillstyle(1,LIGHTGREEN );     //ON
                bar(x1,y1,x2,y2);
                if(x1==150)
                 d0=1;
                if(x1==150+70)
                 d1=1;
                if(x1==150+2*70)
                 d2=1;
                if(x1==150+3*70)
                 d3=1;

        value=(d0*pow(2,0))+(d1*pow(2,1))+(d2*pow(2,2))+(d3*pow
        (2,3))+(d4*pow(2,4)));
        outportb(0x378,value);
goto x;
}
        if(key==18432)
        {
                setfillstyle(1,LIGHTRED );          //OFF
                bar(x1,y1,x2,y2);
                 if(x1==150)
                 d0=0;
                 if(x1==150+70)
                 d1=0;
                 if(x1==150+2*70)
```

```
          d2=0;
          if(x1==150+3*70)
          d3=0;

      value=(d0*pow(2,0))+(d1*pow(2,1))+(d2*pow(2,2))+(d3*pow
      (2,3))+(d4*pow(2,4)));
       outportb(0x378,value);
goto x;
}
      if (key==19712 )
      {
          setcolor(BLACK);
          rectangle(x1-2,y1-2,x2+2,y2+2);          //RIGHT MOVE
          x1=x1+70;
          x2=x2+70;
          if(x1>=150 && x2<=450)
          {
                setcolor(WHITE);
                rectangle(x1-2,y1-2,x2+2,y2+2);
          }
          else
          {
                x1=x1-70;
                x2=x2-70;
                setcolor(WHITE);
                rectangle(x1-2,y1-2,x2+2,y2+2);
          goto x;
```

```
                }
goto x;
}
        if (key==19200 )
        {
                setcolor(BLACK);
                rectangle(x1-2,y1-2,x2+2,y2+2);
                x1=x1-70;                    //LEFT MOVE
                x2=x2-70;
                if( x1>=150 && x2<=450)
                {
                        setcolor(WHITE);
                        rectangle(x1-2,y1-2,x2+2,y2+2);
                }
                else
                {
                        x1=x1+70;
                        x2=x2+70;
                        setcolor(WHITE);
                        rectangle(x1-2,y1-2,x2+2,y2+2);
                goto x;
                }
   goto x;
   }
        else goto x;
   }
```

```
int main(void)
{
  /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;
  /* initialize graphics mode */
  initgraph(&gdriver, &gmode, "");
  /* read result of initialization */
  errorcode = graphresult();
  if (errorcode != grOk)  /* an error occurred */
  {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);          /* return with error code */
  }
  interface();
  control();
  return 0;
}
```

7.1.2 SERIALPORT

```
#include <regat8x51.sfr>
#include "j_data.h"
extern void disp_init(void);
extern _regparm void row_init(unsigned char);
extern _regparm void clr_line(unsigned char);
extern _regparm void rom_disp(unsigned char _rom *, unsigned
        char);
extern _regparm void data_disp(unsigned char _data *,unsigned char);
void delay(void);

void main(void)
{
        // serial port initialization
        TL1=TH1=0xfd;
        TMOD =0x20;
        TR1 =1;
        SCON = 0x50;
        PCON=0x0;
        IE = 0x90;
        P2=0;
        P0=0;
        disp_init();
    rom_disp(n1,15);
        row_init(0xc0);
        rom_disp(n2,16);
        delay();
        delay();
        disp_init();
        rom_disp(st,16);
        // read from port 2 for control of relays
        while(1)
        {
```

```
delay();
delay();
clr_line(0x80);
if(P2_4)
rom_disp(man,16);
else
rom_disp(ato,16);
row_init(0xc0);
rom_disp(sts,0x07);
delay();
delay();
if(P2_0)
{
rom_disp(on,2);
P0_0=1;
}
else
{
rom_disp(off,2);
P0_0=0;
}
delay();
delay();
if(P2_1)
{
rom_disp(on,2);
P0_1 =1;
```

```
}
else
{
rom_disp(off,2);
P0_1=0;
}
delay();
delay();
if(P2_2)
{
rom_disp(on,2);
P0_2=1;
}
else
{
rom_disp(off,2);
P0_2=0;
}
delay();
delay();
if(P2_3)
{
rom_disp(on,2);
P0_3=1;
}
else
{
        rom_disp(off,2);
        P0_3=0;
        }
        delay();
        delay();
        }
        }

void delay(void)
{
        _data unsigned char ii,jj;
        for(ii=0;ii<200;ii++)
        for(jj=0;jj<200;jj++);
}
_interrupt(4) void serial(void)
{
_data unsigned char rx_val,tx_val;
        if(RI)
        {
        RI =0;
        rx_val = SBUF;
        if(rx_val == 0x31)
        {
        tx_val = P0&0xf0;
        tx_val = _ror(tx_val,4);
        rx_val = P2&0x10;
        tx_val = tx_val+rx_val;
        SBUF = tx_val;
        }
        }
        if(TI)
        TI = 0;
}
#include <dos.h>
#include <stdio.h>
#define PORT1 0x3F8
int c;
int ch;
void send(void)
{
outportb(PORT1 + 1 , 0);
outportb(PORT1 + 3 , 0x80);
outportb(PORT1 + 0 , 0x0c);
outportb(PORT1 + 1 , 0x00);
outportb(PORT1 + 3 , 0x03);
outportb(PORT1 + 2 , 0xC7);
outportb(PORT1 + 4 , 0x0B);
outportb(PORT1,0x31);
while (!(c & 1))
{
c = inportb(PORT1 + 5);
ch = inportb(PORT1);
}
}
```

7.1.4 MICROCONTROLLER

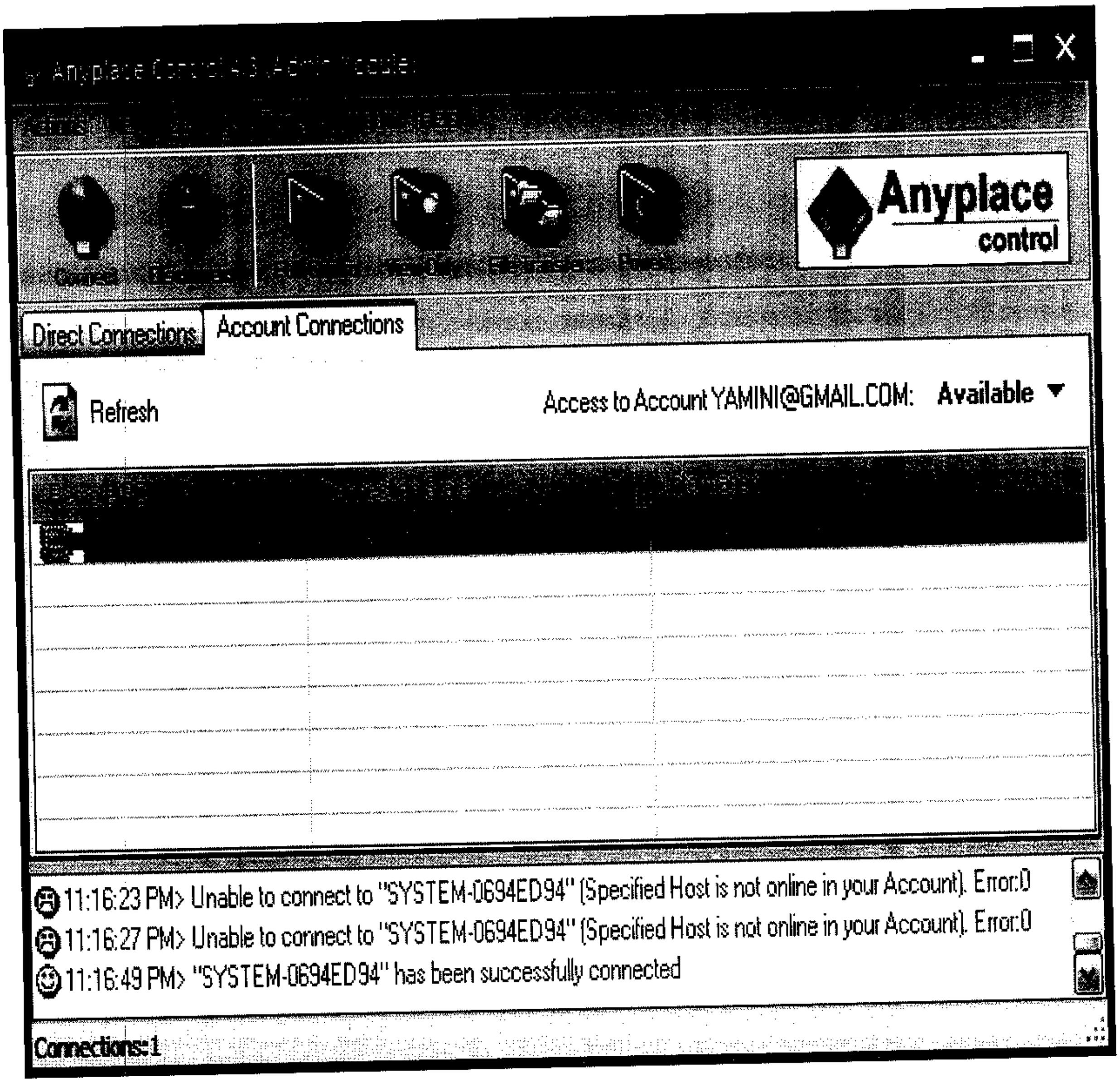
```
#ifndef __AT89X51_H__
#define __AT89X51_H__

/*              Byte Registers          */
sfr P0      = 0x80;
sfr SP      = 0x81;
sfr DPL     = 0x82;
sfr DPH     = 0x83;
sfr PCON    = 0x87;
sfr TCON    = 0x88;
sfr TMOD    = 0x89;
sfr TL0     = 0x8A;
sfr TL1     = 0x8B;
sfr TH0     = 0x8C;
sfr TH1     = 0x8D;
sfr P1      = 0x90;
sfr SCON    = 0x98;
sfr SBUF    = 0x99;
sfr P2      = 0xA0;
sfr IE      = 0xA8;
sfr P3      = 0xB0;
sfr IP      = 0xB8;
sfr PSW     = 0xD0;
sfr ACC     = 0xE0;
sfr B       = 0xF0;
```

```
/*              P0 Bit Registers          */
sbit P0_0 = 0x80;
sbit P0_1 = 0x81;
sbit P0_2 = 0x82;
sbit P0_3 = 0x83;
sbit P0_4 = 0x84;
sbit P0_5 = 0x85;
sbit P0_6 = 0x86;
sbit P0_7 = 0x87;

/*             PCON Bit Values               */
#define IDL_   0x01
#define STOP_  0x02
#define PD_    0x02
#define GF0_   0x04
#define GF1_   0x08
#define SMOD_  0x80

/*    TCON Bit Registers     */
sbit IT0  = 0x88;
sbit IE0  = 0x89;
sbit IT1  = 0x8A;
sbit IE1  = 0x8B;
sbit TR0  = 0x8C;
sbit TF0  = 0x8D;
sbit TR1  = 0x8E;
sbit TF1  = 0x8F;
```

```
/*      TMOD Bit Values          */
#define T0_M0_   0x01
#define T0_M1_   0x02
#define T0_CT_   0x04
#define T0_GATE_ 0x08
#define T1_M0_   0x10
#define T1_M1_   0x20
#define T1_CT_   0x40
#define T1_GATE_ 0x80
#define T1_MASK_ 0xF0
#define T0_MASK_ 0x0F

/*             P1 Bit Registers          */
sbit P1_0 = 0x90;
sbit P1_1 = 0x91;
sbit P1_2 = 0x92;
sbit P1_3 = 0x93;
sbit P1_4 = 0x94;
sbit P1_5 = 0x95;
sbit P1_6 = 0x96;
sbit P1_7 = 0x97;

/*       SCON Bit Registers          */
sbit RI   = 0x98;
sbit TI   = 0x99;
sbit RB8  = 0x9A;
sbit TB8  = 0x9B;
sbit REN  = 0x9C;
sbit SM2  = 0x9D;
sbit SM1  = 0x9E;
sbit SM0  = 0x9F;

/*              P2 Bit Registers                */
sbit P2_0 = 0xA0;
sbit P2_1 = 0xA1;
sbit P2_2 = 0xA2;
sbit P2_3 = 0xA3;
sbit P2_4 = 0xA4;
sbit P2_5 = 0xA5;
sbit P2_6 = 0xA6;
sbit P2_7 = 0xA7;

/*         IE Bit Registers          */
sbit EX0  = 0xA8;
sbit ET0  = 0xA9;
sbit EX1  = 0xAA;
sbit ET1  = 0xAB;
sbit ES   = 0xAC;
sbit ET2  = 0xAD;
sbit EA   = 0xAF;

/*          P3 Bit Registers (Mnemonics & Ports)          */
sbit P3_0 = 0xB0;
sbit P3_1 = 0xB1;
sbit P3_2 = 0xB2;
sbit P3_3 = 0xB3;
sbit P3_4 = 0xB4;
sbit P3_5 = 0xB5;
sbit P3_6 = 0xB6;
sbit P3_7 = 0xB7;

sbit RXD  = 0xB0;
sbit TXD  = 0xB1;
sbit INT0 = 0xB2;
sbit INT1 = 0xB3;
sbit T0   = 0xB4;
sbit T1   = 0xB5;
sbit WR   = 0xB6;
sbit RD   = 0xB7;

/*        IP Bit Registers        */
sbit PX0  = 0xB8;
sbit PT0  = 0xB9;
sbit PX1  = 0xBA;
sbit PT1  = 0xBB;
sbit PS   = 0xBC;
sbit PT2  = 0xBD;

/*        PSW Bit Registers        */
sbit P    = 0xD0;
sbit FL   = 0xD1;
sbit OV   = 0xD2;
sbit RS0  = 0xD3;
sbit RS1  = 0xD4;
sbit F0   = 0xD5;
sbit AC   = 0xD6;
sbit CY   = 0xD7;

/*      Interrupt Vectors:Interrupt Address = (Number * 8) + 3      */
#define IE0_VECTOR   0
#define TF0_VECTOR   1
#define IE1_VECTOR   2
#define TF1_VECTOR   3
#define SIO_VECTOR   4
#endif
```

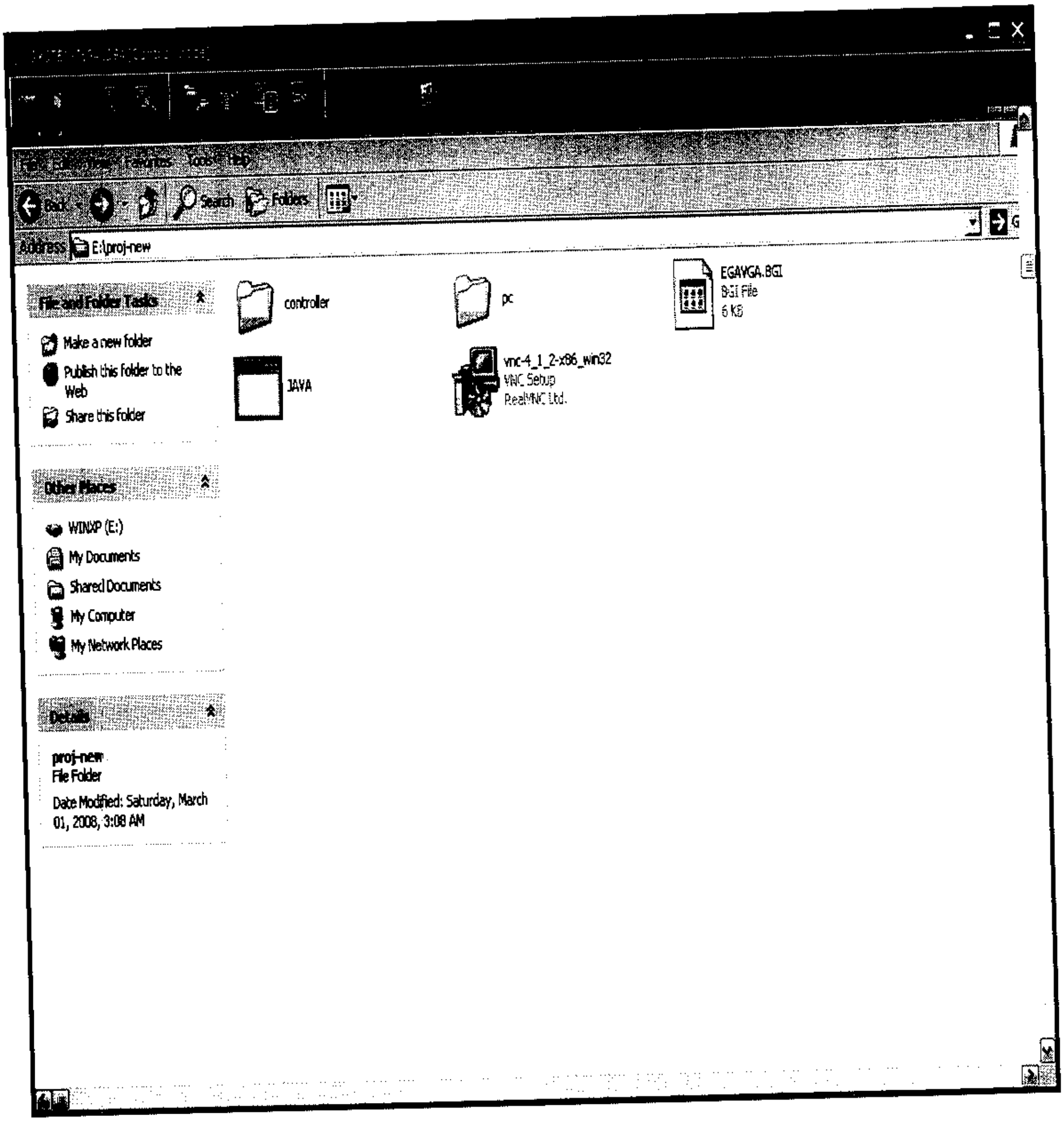
7.2 APPENDIX 2- SCREEN SHOTS

7.2.1 CONNECTION ESTABLISHMENT

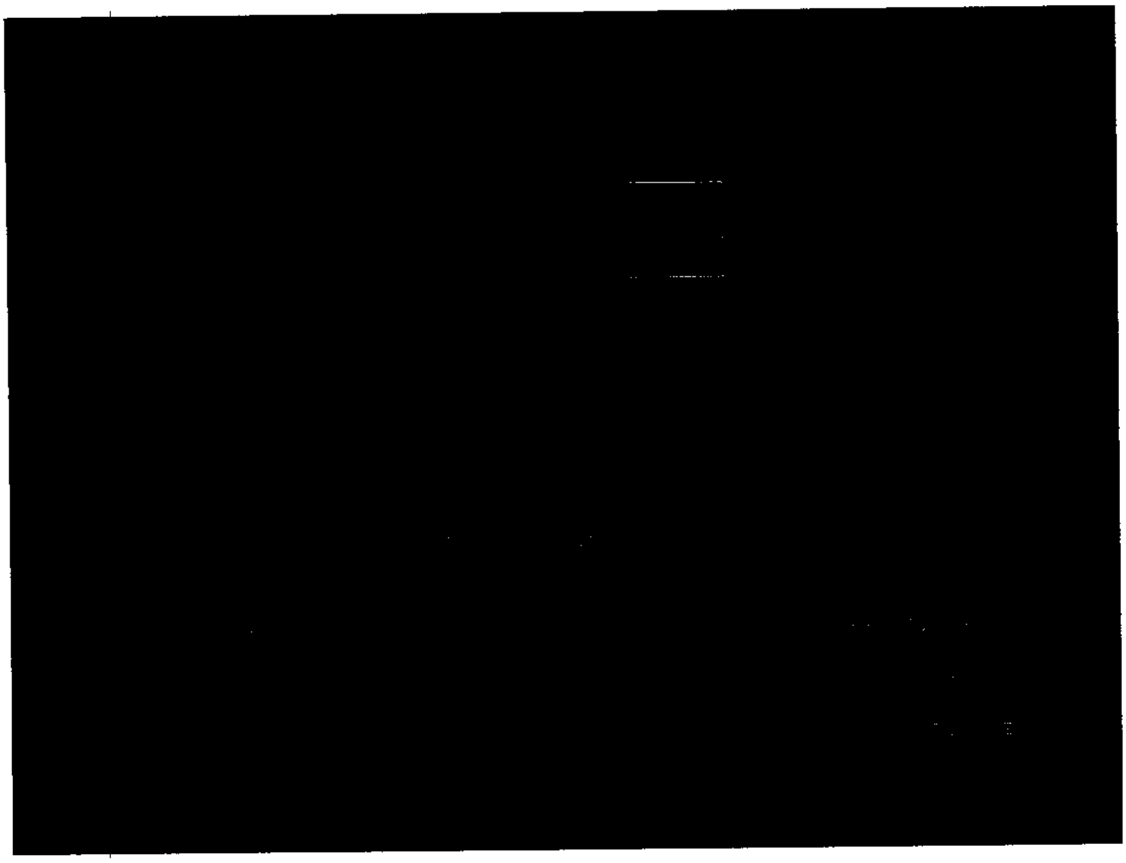


```
11:16:23 PM> Unable to connect to "SYSTEM-0694ED94" (Specified Host is not online in your Account). Error:0
11:16:27 PM> Unable to connect to "SYSTEM-0694ED94" (Specified Host is not online in your Account). Error:0
11:16:49 PM> "SYSTEM-0694ED94" has been successfully connected
```

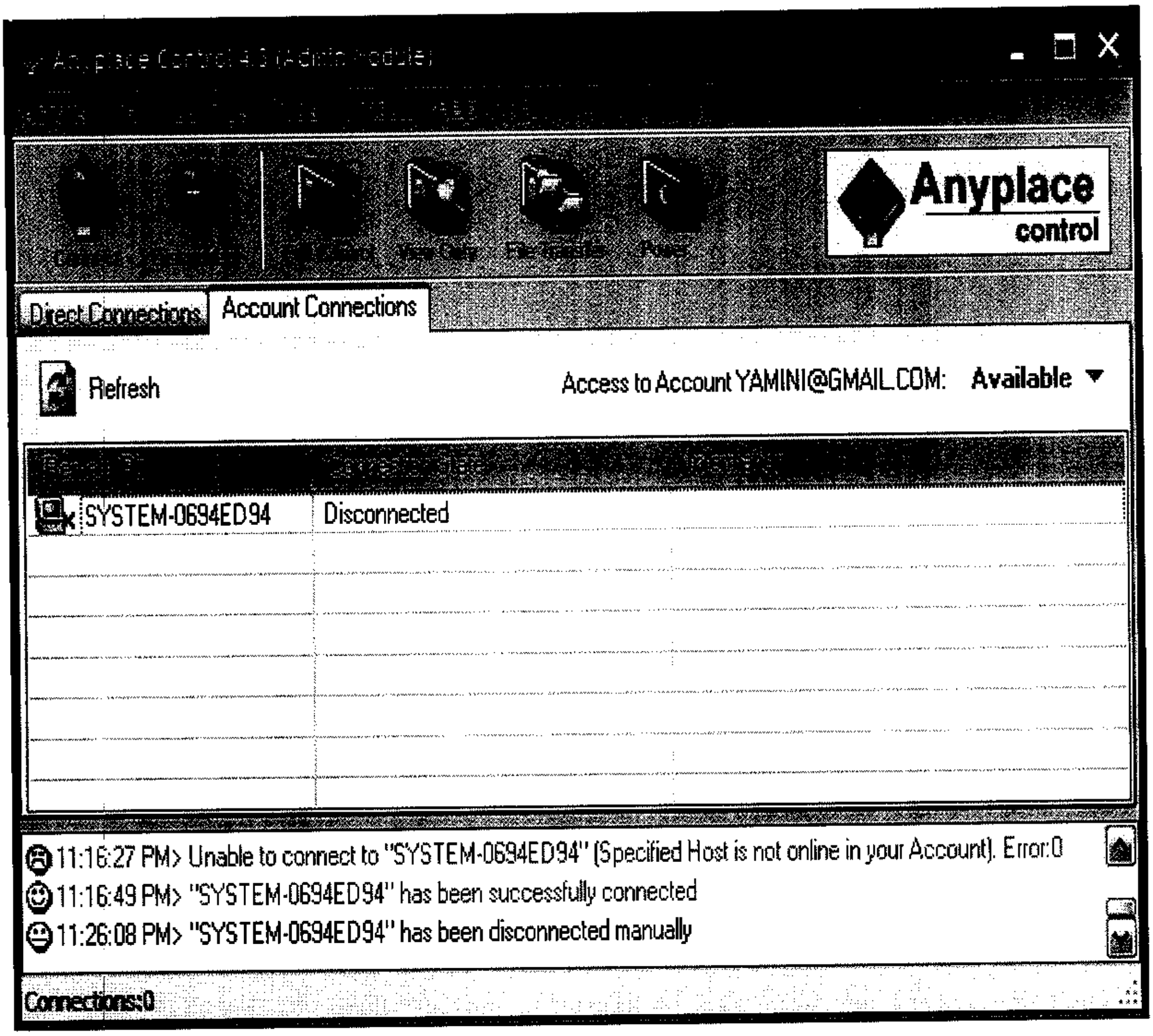
7.2.3 ACCESSING REMOTE DESKTOP



7.2.4 LOCAL ACCESS



7.2.5 DISCONNECTION



Access to Account YAMINI@GMAIL.COM: Available

SYSTEM-0694ED94 Disconnected

```
11:16:27 PM> Unable to connect to "SYSTEM-0694ED94" (Specified Host is not online in your Account). Error:0
11:16:49 PM> "SYSTEM-0694ED94" has been successfully connected
11:26:08 PM> "SYSTEM-0694ED94" has been disconnected manually
```

Connections:0

7.3 APPENDIX3-HARDWARE PHOTO

REFERENCES

8. REFERENCES

BOOKS

1. Kenneth J.Ayala (1996),'The Microcontroller architecture, programming and Application-2nd edition.
2. Muhammad Ali Mazidi and Janice Gillespie Mazidi (2000)' the 8051 Microcontroller and Embedded Systems'.
3. Jan Axelson (1999),"serial port complete", 4th Edition, Penram International Publication.

WEBSITES

1. http://www.datacatalog.com/
2. http://www.atmel.com/dyn/resources/prod_documents
3. http://www.codeproject.com/KB/aspnet/BasePageClass.aspx