

DESIGN AND IMPLEMENTATION OF A MOBILE- TO-MACHINE WIRELESS COMMUNICATION SYSTEM FOR REMOTE DEVICE CONTROL

**By
Zahidul Islam**

A project submitted to the Department of Electrical and Electronic Engineering
Bangladesh University of Engineering & Technology
in partial fulfillment of the requirements
for the degree of

**MASTER OF ENGINEERING
IN
ELECTRICAL AND ELECTRONIC ENGINEERING**



**DEPARTMENT OF ELECTRICAL & ELECTRONIC ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA
2012**

CERTIFICATION

The project titled “**Design and implementation of a Mobile-to-Machine wireless communication system for remote device control**” submitted by Zahidul Islam, Roll No: 100706212P, Session: October, 2007 has been accepted as satisfactory in partial fulfillment of the requirements for the degree of MASTER OF ENGINEERING IN ELECTRICAL AND ELECTRONIC ENGINEERING on Jan 28, 2012.

BOARD OF EXAMINERS

1. _____
(Dr. Satya Prasad Majumder) **Chairman**
Professor and Dean
Faculty of EEE
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh

2. _____
(Dr. Md. Saifur Rahman) **Member (Ex-Officio)**
Professor and Head
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh

3. _____
(Dr. Abdul Hasib Chowdhury) **Member**
Associate Professor
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh

4. _____
(Dr. Shaikh Anowarul Fattah) **Member**
Associate Professor
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh

DECLARATION

It is hereby declare that this project has been done by me and no part of it has been submitted elsewhere partially or fully for the award of any degree or diploma.

Signature of the candidate

(Zahidul Islam)
Roll No: 100706212P

DEDICATION

To my parents, wife and younger brother...

ACKNOWLEDGEMENT

First of all, I am obliged to Almighty Allah for giving me the opportunity to complete my project work.

I show gratefulness to my supervisor, Dr. Satya Prasad Majumder , Professor and Dean, Faculty of EEE, BUET. His endless support, steadfast guidance, continuous motivation and valuable suggestions in each and every step of my project were just unparalleled. I deeply acknowledge his contribution and I am whole-heartedly indebted to him.

My heartiest acknowledgment goes to Dr. Kazi Mujibur Rahman , Professor, EEE, BUET, for his kind time and suggestions in developing the hardware part of this project. I would like to express my gratitude to him for sharing his valuable experience, knowledge and concept on hardware design.

I would also like to mention Dr. Abdul Hasib Chowdhury , Associate Professor EEE, BUET, who advised me to develop the remote monitoring part of this project.

And last but not the least; I would like to thank my beloved parents, wife and younger brother, who always inspired me and continuously kept on supporting me to complete my project work.

ABSTRACT

High technological advancement in the field of communication has introduced a new era in wireless device interconnectivity. The advent of wireless M2M (Machine-to-machine, mobile-to-machine and machine-to-mobile) facilitates networking between machines and devices. It combines- communication, electronics, software and power technologies to enable remote human and machine interaction with discrete systems and processes. Wireless M2M can be supported by almost all wireless technologies that include GSM/GPRS, UMTS/3G, LTE, WLAN, WiMAX etc. M2M over EDGE/GPRS is popular as its security features are already available, provides flexibility in mobility and being low-priced. Wireless M2M offers the opportunity of remote controlling a system as TCP/IP based protocols can run over the GSM network like a traditional local area network (LAN) or wide area network (WAN) environment. Already some costly tools are developed in the world market to remotely control a particular power system.

Parallel approach has been taken into this project to remotely control a power device by means of a cellular operator's EDGE/GPRS network. It involves selection of appropriate applications and ability to combine those to enable M2M interaction. A simple approach has been pursued to develop a client-server model which is the basic of the entire concept. Client and server will communicate to each other using wireless data network. Unique type of Access Point Name (APN) is required to enable server-client interaction in which traffic will not traverse to open Internet. Selection of appropriate application is also important as latency in the air interface appends extra delay and propagation constraints play a major role in packet loss which can distort control signal. To overcome these factors, data transfer rate between client and server may be kept as minimum as possible. Again, move from proprietary application to open source based application introduces security hole and leads the system open to vulnerable attacks. To avoid security breach, GSM layer protection in combination with application layer encryption can be used.

TABLE OF CONTENTS

	Page No.
TITLE PAGE.....	i
CERTIFICATION.....	ii
DECLARATION.....	iii
DEDICATION.....	iv
ACKNOWLEDGMENT.....	v
ABSTRACT.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xii
LIST OF IMPORTANT ABBREVIATIONS.....	xiii
COMPACT DISC	

CHAPTER 1- INTRODUCTION

1.1 M2M Communication.....	1
1.2 Historical Perspective of M2M Communication.....	2
1.3 Present State of the Problem.....	3
1.4 Objectives and Possible Outcome.....	4
1.5 Outline of Methodology or Experimental Design.....	4
1.6 Organization of This Project.....	5

CHAPTER 2- PROPOSED SYSTEM and SYSTEM SPECIFICATION

2.1 Introduction.....	6
2.2 Development of Web Server.....	6
2.2.1 Components and Types of XAMPP.....	7
2.2.2 Loading the First Test Page.....	7
2.2.2.1 First PHP Page.....	8

2.3 Interfacing with Parallel Port.....	10
2.3.1 Parallel Port Hardware Types.....	13
2.3.2 Configuring Parallel Port Hardware.....	15
2.3.3 Configuring Parallel Ports under Windows.....	15
2.3.4 Parallel Port Address.....	16
2.3.5 Sending Instruction to Parallel Port.....	19
2.3.6 Working Principle.....	20
2.3.7 Steps for Loading Parallel Port Web Page.....	21
2.4. Wireless Network.....	23
2.4.1 GPRS Network Architecture.....	25
2.4.2 EDGE Network Architecture.....	28
2.4.3 Data Rates of GPRS/EDGE Network.....	28
2.4.4 GSM Security Features.....	30
2.4.5 General Authentication Procedure and Data Confidentiality.....	30
2.4.6 Attach and PDP Context Activation Procedure.....	32
2.5 Integration with Wireless Network.....	33

CHAPTER 3- DESIGN and IMPLEMENTATION

3.1 Introduction.....	34
3.2 Requirements of Driving Circuit.....	34
3.3 External Driving Circuit's Working Principle.....	36
3.4 Logic Circuit's Working Principle.....	38
3.5 Scripting/ Coding.....	40
3.5.1 Discussion on Controlport.php.....	40
3.5.2 Discussion on Loadshedding.php.....	41

3.5.3 Supported Commands and Identifiers.....	42
3.6 File Organization.....	43
3.7 Networking between Server and Client.....	43
3.7.1 Data Flow Diagram.....	44
3.7.2 Loading the Desired Web Page.....	45
3.8 Conclusion.....	47

CHAPTER 4- RESULTS AND DISCUSSION

4.1 Introduction.....	48
4.2 Results and Discussion.....	48
4.3 Bandwidth Calculation Considering Practical Cases.....	52
4.4 Conclusion.....	54

CHAPTER 5- FUTURE SCOPE OF THIS PROJECT

5.1 Conclusion.....	55
5.2 Future Scope.....	55

APPENDIX A	DATA SHEET of MOSFET	59
APPENDIX B	DATA SHEET of OPTOCOUPLER.....	61
APPENDIX C	DATA SHEET of DIODE.....	63
APPENDIX D	PORTSTATUS.PHP CODE.....	64
APPENDIX E	SAJAX.PHP CODE.....	68
APPENDIX F	LOADSHEDDING.PHP CODE.....	77

LIST OF FIGURES

Name	Caption	Page No.
Figure 1.1	Generic Block Diagram of M2M Communication	2
Figure 2.1	PCIE Parallel port	6
Figure 2.2	Sample PHP Code	9
Figure 2.3	PC Parallel Port	10
Figure 2.4	Centronics Connector (a) Female (b) Male	11
Figure 2.5	IEEE 1284 Type C Hoover	11
Figure 2.6	Three Types of Connectors	11
Figure 2.7	Parallel Port Configuration under Windows 2000/XP	15
Figure 2.8	Pin outs of DB25 connector	18
Figure 2.9	Parallel Port Programming	20
Figure 2.10	C:\WINDOWS\system32\drivers\etc hosts file configuration	21
Figure 2.11	Basic GSM Network Architecture	23
Figure 2.12	Mobile Station Components	24
Figure 2.13	Simplified Architecture of GPRS Network	25
Figure 2.14	Static IP Addressing Mechanism by HLR	27
Figure 2.15	Dynamic IP Addressing Mechanism by GGSN/RADIUS	28
Figure 2.16	General Authentication Procedure	31
Figure 3.1	High Level Block Diagram for Design and Implementation	34
Figure 3.2	One Light and one Fan used as load	35
Figure 3.3	Block Diagram of External Driving Circuit	36
Figure 3.4	External Driving Circuit	37
Figure 3.5	Block Diagram of Logic Circuit	38
Figure 3.6	Logic Circuit	39
Figure 3.7	Configuring APN on Device	44
Figure 3.8	IP Connectivity between MS (Handset) and GGSN	44
Figure 3.9	Client-Server Communication	45
Figure 3.10	Editing httpd.conf file	46

Name	Caption	Page No.
Figure 3.11	Location of httpd.conf file	47
Figure 4.1	Contents of htdocs folder	48
Figure 4.2	Output of portstatus.php	49
Figure 4.3	Bit level control of data pins	49
Figure 4.4	Load Shedding Status	50
Figure 4.5	Controlport.php and loadshedding.php loaded at client	51
Figure 5.1	Recent M2M Devices	56

LIST OF TABLES

Name	Caption	Page No.
Table 2.1	Pin Assignments of the D-Type 25 pin Parallel Port Connector	13
Table 2.2	Parallel Port Base Addresses	16
Table 2.3	Data Port	17
Table 2.4	Status Port	17
Table 2.5	Control Port	17
Table 2.6	Summary of Register Addresses	18
Table 2.7	Signals and pin numbers for general purpose I/O	19

LIST of IMPORTANT ABBREVIATIONS

3GPP	Third Generation Partnership Project
APN	Access Point Name
AUC	Authentication Center
BIOS	Basic Input Output System
BSC	Base Station Controller
BSD	Berkeley Software Distribution
CDMA	Code Division Multiple Access
DLL	Dynamic Link Library
DMA	Direct Memory Access
ECP	Enhanced Capability Port
EDGE	Enhanced Data for Global Evolution
EPP	Enhanced Parallel Port
ETSI	European Telecommunications Standards Institute
GGSN	Gateway GPRS Support Node
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HLR	Home Location Register
HSPA	High Speed Packet Access
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ITU	International Telecommunication Union
LA	Location Area
LAN	Local Area Network
LED	Light-Emitting Diode

LTE	Long Term Evolution
MCS	Modulation and Coding Scheme
MSISDN	Mobile Station Integrated Services Digital Network Number
PCU	Packet Control Unit
PDP	Packet Data Protocols
PHP	Hypertext Preprocessor
PSK	Phase Shift Keying
RA	Routing Area
SCADA	Supervisory Control and Data Acquisition
SGSN	Serving GPRS Support Node
SIM	Subscriber Identification Module
SMS	Short Message service
SPP	Standard Parallel Port
TCP	Transmission Control Protocol
UMTS	Universal Mobile Telecommunications System
VPN	Virtual Private Network
WAN	Wide Area Network
WEB	World Wide Web
WiMAX	Worldwide Interoperability for Microwave Access)
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 M2M Communication

M2M stands for machine-to-machine, mobile-to machine and machine-to-mobile communication. M2M is about interactions that can take place between people and their products by networking between the machines and devices. It allows interconnectivity of machines and devices. M2M communications have appeared as a cutting edge technology for next-generation communications [1-3]. As people and devices with electronic systems have become more interconnected, machine-to-machine technology is on the edge to turn out to be a powerful force in the 21st century [4]. M2M will unleash new levels of “smart services”. There are millions of electronic devices in the world that are well suited to take the benefit of M2M technology. According to Bob Emmerson, European Editor of M2M Magazine - “Machine-to-Machine (M2M) communications is a healthy sector that is expanding rapidly and generating significant revenues for mobile network operators (MNOs). Devices outnumber subscribers by an order of magnitude, but the term doesn’t do justice to the concept and the market it represents.”

M2M communication can exist practically in any environment and market. Currently, successful implementations have been carried out for vehicles, security systems, electric metering, home appliances etc. Devices or machines can also be connected to other devices or machines for making a seamless and automated flow of data and service. Regardless of the type of device or data, information flow usually follows the same model- from a device over a network and then from the network to a device or machine and so on. M2M is more than just connected wireless devices sharing data remotely; it’s also about gathering and distributing the desired data mostly in real time. Basically, M2M technology permits one smart device talking to another smart device via a communication

network. With numerous advancements in wireless communication system, these devices can easily be placed in diverse locations many kilometers away from one to another.



Figure 1.1: Generic Block Diagram of M2M Communication

But in order to transmit and receive data over long distance for these devices a communication medium is also required such as IP, terrestrial or satellite-based network which in most cases is a GSM or GPRS/EDGE network or their later releases such as 3G or LTE [5]. One of the key factors of M2M, a primary distinction that makes this technology extremely useful for many applications, is its ability to give user access to a particular device via a Web Server, which allows it to be configured remotely. This certainly makes practical applications of M2M far more exciting and effective.

1.2 Historical Perspective of M2M Communication

The origin of M2M communication is not clear because of the many different possibilities of its inception. Year 2009 was important for the development of M2M technology, both in the U.S. and in the Europe. In the United States, ATandT and Jasper Wireless entered into an agreement to support the creation of M2M devices jointly. In Europe, the Norwegian incumbent Telenor concluded ten years of M2M research. In December 2009, Spanish Telefonica announced that they are also setting up an M2M entity in Madrid. In early 2010 in the U.S., ATandT, KPN, Rogers, America Movil, Jasper Wireless began to work together in the creation of a M2M site, which will serve as a hub for developers in the field of M2M communication electronics. In February 2010, Vodafone, Verizon Wireless and nPhase (a joint partnership of Qualcomm and Verizon) announced their strategic alliance to provide global M2M solutions that would offer their customers an easy way to roll out M2M solutions across Europe and the US. In March 2010, Sprint and Axeda Corporation announced their strategic alliance for global M2M solutions. In January 2011, Aeris Communications, Inc. announced that it is providing M2M

telematics services for Hyundai Motor Corporation. Partnerships like these have made it easier, faster and more cost-efficient for businesses to use M2M.

In Europe, the car to car (Car2Car) communication consortium has been established to develop a European-wide standard for inter-vehicle communications based on wireless local area network (LAN) technologies. Similar activities can be observed in the USA, where a family of new IEEE standards (IEEE 1609.3, 1609.4, 1609.11) has been released [6]. This new family of IEEE standards will ensure that cars and roadside structures can communicate with each other.

1.3 Present State of the Problem

Wireless M2M offers the opportunity of remote controlling a system as TCP/IP based protocols can run over the packet switched network compared to a traditional local area network (LAN) or wide area network (WAN). Already some costly tools like SCADA, BACnet etc. have been developed in the world market to remotely control a particular power system. Parallel approach has been taken into this project to remotely control a power device by means of a cellular operator's EDGE/GPRS network. It involves selection of suitable applications and ability to combine those to facilitate M2M interaction.

A parallel approach has been pursued to develop a client-server model which will interact via M2M connectivity. Client and server will communicate with each other using wireless data network. Unique type of Access Point Name (APN) is required to enable server-client interaction in which traffic will not flow using open Internet. Selection of proper application is also important as latency in the air interface appends extra delay and propagation constraints play a major role in packet loss which can distort the control signal. To overcome these factors, data transfer rate between client and server may be kept as minimum as possible. Again, move from proprietary application to open source based application introduces security hole and leads the system open to vulnerable

attacks. To avoid security breach, GSM layer protection in combination with application layer encryption can be used [7-9].

1.4 Objectives and Possible Outcome

The specific aim of this project is to develop a remote control arrangement for a power device using wireless network. It will facilitate the opportunity of remote controlling a power device managed by distant branch office from a centralized head office. Control signal will be sent to a branch office from a head office which will in turn switch on/off a device. Signal will be generated from mobile (client) to be delivered to web server and will be transmitted over available EDGE/GPRS network. Data will traverse in an intranet environment and will not be sent out to open Internet which will ensure secured communication. The proposed system will also facilitate to know the load shedding status at remote site. As a result, it will help to execute planned operation.

1.5 Outline of Methodology or Experimental Design

Apache based WEB server will be developed using free tool 'XAMPP for windows XP'. XAMPP is a complete package supporting different distributions like-APACHE, PHP and Pearl. It's free of charge and free to copy under the terms of the 'GNU General Public License'. It demands less resource compared to other WEB server development tools and can be deployed quickly maintaining given instructions. Client will be hosted at head office and server will be hosted at branch office. Interfacing with parallel port can be accomplished by writing source codes in C, VB or C# etc. In this project, free tool will be used to control parallel ports. It allows writing and reading particular value to/from parallel ports and bit-level operation of the port data. This application works with PHP installed on WEB server and JavaScript running on client's web browser. When a web page will be loaded by client mobile/PC, user will get a web page and user-end JavaScript code. This code will periodically check parallel port status from the server and will show updated status to the client mobile/PC. Clicking control button in the web-form instructs the server to perform a desired function. When the controlling is completed, WEB server

returns the current port status to client. As direct control of parallel port is not possible under WINXP by means of applications, we need device driver to access parallel ports.

Server side will be interfacing parallel ports to drive external circuits. When parallel port's data pin will receive signal from client mobile/PC, it will trigger a relay driven by a protection circuit. Receiving logical '1' i.e. ~5 volt, will close the relay and consecutively a device will begin to supply power to connected loads. A logical '0' i.e. ~0 volt will open the relay and consecutively the device will stop delivering power to loads.

As per standard practice, default APN property doesn't permit device to device communication to ensure user's security. To enable client-server communication, APN properties need to be changed. Static IP delivered by APN will ensure steady communication between client and server, however dynamic IP will introduce additional steps to be configured in web server. Introducing user name and password in APN properties will tighten the security.

1.6 Organization of This Project

This project consists of five chapters which are as below:

Chapter-1 is highlighting on the brief story of M2M communication and focuses on the main objective of this project.

In chapter-2, a brief overview has been given on proposed system and system specification.

In chapter-3, design and implementation part has been described.

In chapter-4, discussion has been made on the output of the system with analysis.

In chapter-5, conclusion and future scope of this project has been depicted.

CHAPTER 2

PROPOSED SYSTEM and SYSTEM SPECIFICATION

2.1 Introduction

In this project, the proposed system can be divided into three major segments which are as below:

- 1) Development of Web server using XAMPP package
- 2) Interfacing of parallel port to drive external circuits
- 3) Incorporation with wireless network for remote controlling and monitoring

2.2 Development of Web Server

Web server can be developed using any PC having parallel port. Pentium-III and higher is recommended for the server setup with 733 MHz processor and 1GB RAM will be enough. Many manufacturers of personal computers and laptops consider parallel port to be a legacy port and no longer include the parallel interface. As a result, PCIE (Peripheral Component Interconnect Express) parallel port card can be used. These cards must follow The IEEE 1284 standard.



Figure 2.1: PCIE Parallel Port

Apache based WEB has been developed using free tool ‘XAMPP for windows XP’ [10]. XAMPP is a complete package supporting different distributions like-APACHE, PHP and

Pearl. It's free of charge and free to copy under the terms of the 'GNU General Public License'. It demands less resource compared to other free Web server development tools and can be deployed quickly by following given instructions.

2.2.1 Components and Types of XAMPP

XAMPP is a small and light Apache distribution containing the most common web development tools in a single package. Its contents, size, and portability makes it the ideal tool for students who are developing and testing applications in PHP and MySQL. XAMPP is available as a free download in two specific packages: Full and Lite. While the full package download provides a wide array of development tools, XAMPP Lite contains the minimum tools to develop the web server. As the name implies, the Light version is a small package containing Apache HTTP Server, PHP, MySQL, phpMyAdmin, Openssl, and SQLite. For this project, we shall use the Lite version.

2.2.2 Loading the First Test Page

After successful installation, a folder under Xampp directory will be created named as htdocs. Whatever is kept in this folder can be accessed easily via http:// or https:// protocol. XAMPP package by default includes necessary tools to support PHP script. There is no need to compile anything nor do we need to install any extra tools to support PHP coding. PHP is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. PHP acts primarily as a filter, taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data; most commonly HTML will show the output. Following is an example of how PHP is embedded:

```
<html>
  <head>
    <title>My first PHP Page</title>
```

```
</head>
<body>
  This is normal HTML code
  <?php
    // php code goes here
  ?>
  Back into normal HTML
</body>
</html>
```

2.2.2.1 First PHP Page

PHP coding is simplified and doesn't need to put much effort to develop a page. Here, a sample code is shown which will show Hello World in a browser like Internet Explorer or Mozilla. For this, we have to open a text editor and write below lines of code:

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php echo '<p>Hello World</p>'; ?>
  </body>
</html>
```

This file will be parsed by PHP and the following output will be sent to the browser by HTML:

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
```

```
<body>
  <p>Hello World</p>
</body>
</html>
```

All it does is, display: *Hello World* using the PHP `echo()` statement. The file *does not need to be executable* or special in any way. The server finds out that this file needs to be interpreted by PHP because we used the ".php" extension, which the server is configured to pass on to PHP. In order to run and test this PHP code, we have to save it into our `xampplite\htdocs` directory with the file name of `helloWorld.php`. Then, to view it in our browser, we have to type simply: `http://localhost/helloWorld.php` into the address bar. If done correctly, we shall see `Hello World` line of text in the web browser as shown in below figure:

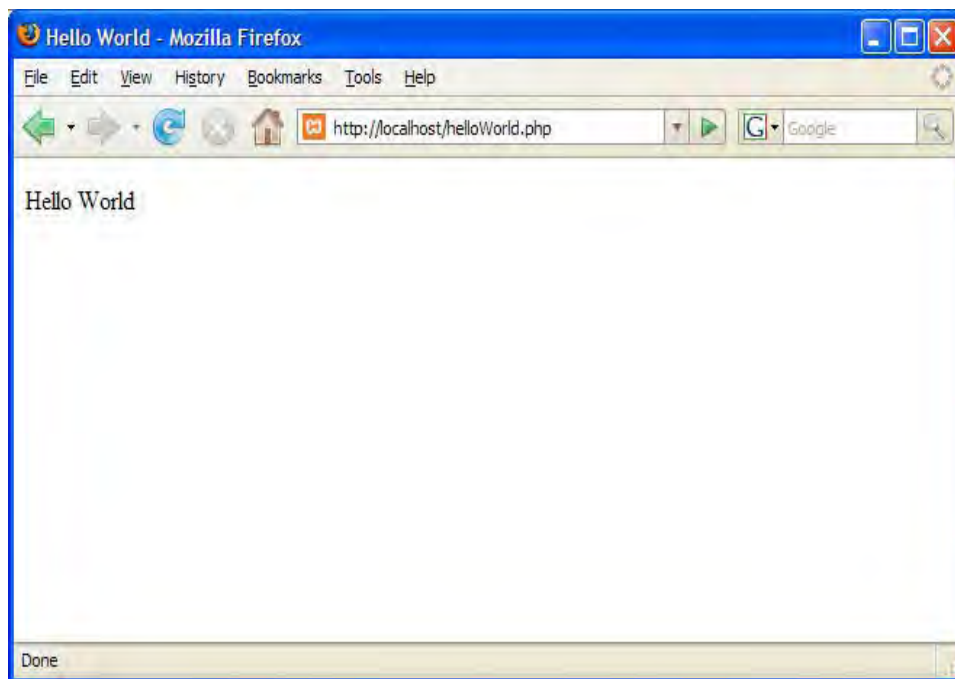


Figure 2.2: Sample PHP Code

2.3 Interfacing with Parallel Port

Parallel ports were originally developed by IBM as a way to connect a printer to PC. IBM engineers coupled a 25-pin connector, DB-25, with a 36-pin Centronics connector to create a special cable to connect the printer to the computer. When a PC sends data to a printer or other device using a parallel port, it sends 8 bits of data (1 byte) at a time. These 8 bits are transmitted parallel to each other, as opposed to the same eight bits being transmitted serially (all in a single row) through a serial port. The pins can be used as Input and Output, although there are some restrictions for the pins.

The output pins of the parallel port are TTL level output pins. This means that they put out ideally-

- 0V when they are in low logic level (0) and
- +5V when they are in high logic level (1).

In real world the voltages can be something different from ideal when the circuit is loaded. The output current capacity of the parallel port is limited to only few milli amperes (mA).

In ordinary parallel port implementations, the data outputs are 74LS374 IC totem-pole TTL outputs which can source 2.6 mA and sink 24 mA. The best think is to use a buffer, so that least current is drawn from the Parallel Port. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of our PC as a D-Type 25 Pin female connector.



Figure 2.3: PC Parallel Port

The D-Type 25 pin connector is the most common connector found on the Parallel Port of a computer [Figure 2.3], while the Centronics Connector is commonly found on printers [Figure 2.4] as mentioned before. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The first one, 1284 Type A is the D-Type 25 connector found on the back of most computers. The 2nd is the 1284 Type WEB which is the 36 pin Centronics Connector found on most printers.



(a)



(b)

Figure 2.4: Centronics Connector (a) Female (b) Male

IEEE 1284 Type C is a 36 conductor connector like the Centronics, but smaller and called as mini Centronics.



Figure 2.5: IEEE 1284 Type C Hoover



Figure 2.6: Three Types of Connectors

In this project, our concentration will be focused on D-Type 25 pin connector. Each pin of a D-Type 25 does the following when used with a printer [11]:

Pin 1 carries the strobe signal. It maintains a level of between 2.8 and 5 volts, but drops below 0.5 volts whenever the computer sends a byte of data. This drop in voltage tells the printer that data is being sent.

Pins 2 through 9 are used to carry data. To indicate that a bit has a value of 1, a charge of 5 volts is sent through the correct pin. No charge on a pin indicates a value of 0. This is a simple but highly effective way to transmit digital information over an analog cable in real-time.

Pin 10 sends the acknowledge signal from the printer to the computer. Like Pin 1, it maintains a charge and drops the voltage below 0.5 volts to let the computer know that the data was received. **Pin 11** will charge if the printer is busy. Then, it will drop the voltage below 0.5 volts to let the computer know it is ready to receive more data.

The printer lets the computer know if it is out of paper by sending a charge on **Pin 12**. As long as the computer is receiving a charge on **Pin 13**; it knows that the device is online. The computer sends an auto feed signal to the printer through **Pin 14** using a 5-volt charge.

If the printer has any problems, it drops the voltage to less than 0.5 volts on **Pin 15** to let the computer know that there is an error. Whenever a new print job is ready, the computer drops the charge on **Pin 16** to initialize the printer.

Pin 17 is used by the computer to remotely take the printer offline. This is accomplished by sending a charge to the printer and maintaining it as long as we want the printer offline. **Pins 18-25** are grounds and are used as a reference signal for the low (below 0.5 volts) charge.

Table 2.1: Pin Assignments of the D-Type 25 Pin Parallel Port Connector

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

The above table uses "n" in front of the signal name to denote that the signal is active low. e.g. nError. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware.

2.3.1 Parallel Port Hardware Types

Parallel port hardware may be of five types, described below in the order of their appearance in PCs. A computer may contain any of these port types, and may include ports of more than one type. Earlier ports are limited in functionality and performance. Later ports provide increased functionality and performance.

Unidirectional 4-bit parallel ports:

The unidirectional 4-bit parallel port, also called a standard parallel port (SPP), is based on the de facto Centronics standard, and was the type of parallel port supplied with the

original IBM PC and its clones. An SPP does 8-bit output and can accept 4-bit (nibble) input.

Bidirectional 8-bit parallel ports:

When IBM introduced the PS/2 line in 1987, all but the two lowest-cost models (the Models 25 and 30) included a bidirectional 8-bit parallel port. These ports support both 8-bit input and output.

Enhanced parallel ports (EPP):

EPP offered better performance and superior data speed while maintaining backward SPP compatibility. EPP supports 8-bit bidirectional communications at ISA (Industry Standard Architecture) bus speed, providing throughput similar to that of 8-bit ISA bus cards. Many 386 and 486 systems and most reasonably I/O expansion cards include EPP-capable parallel ports.

Extended capabilities ports (ECP):

ECP has been developed by Microsoft and Hewlett-Packard. It supports 8-bit bidirectional communications at ISA bus speed and uses DMA, provides a FIFO buffer of at least 16 bytes, and includes hardware data compression. These features allow ECP to provide better throughput than EPP. Some 486 systems, most Pentium and higher systems, and recent I/O expansion cards include ECP-capable parallel ports.

IEEE 1284 parallel ports:

The increasing diversity of parallel port hardware and the resulting potential for incompatibilities made it desirable to develop an umbrella standard that would combine and codify these earlier adhoc standards into a single formal standard. The resulting document, 1284-1994 IEEE Standard Signaling Method does so by defining five parallel transmission modes. IEEE 1284-compliant parallel port hardware, available on recent computers, motherboards, and expansion cards specifies five modes of operation, each

mode providing data transfer in either the forward direction (computer to peripheral), backward direction (peripheral to computer), or bi-directional (one direction at a time).

2.3.2 Configuring Parallel Port Hardware

The parallel port mode is often set to SPP by default. The first step is to determine the capabilities of the port hardware. On older motherboards and expansion cards, we may have to set the mode by using a jumper. On newer systems, we can usually set the mode using the BIOS Setup program. The parallel port modes available are determined first by the capabilities of the port hardware itself. Even if the hardware supports all modes, however, the BIOS may not, so we may be limited in the choices we can make.

2.3.3 Configuring Parallel Ports under Windows

Parallel port support by PC also differs widely between Windows distributions. Windows NT up-to V4.0 does not support EPP or ECP bidirectional communications ports. If a parallel port is configured as EPP or ECP; Windows NT 4.0 and prior will not detect the port. Windows 2000/XP fully supports most parallel modes, including Centronics mode, IEEE 1284 modes, ECP mode, and EPP mode. Windows 2000/XP partially supports IEEE 1284.3 modes.



Figure 2.7: Parallel Port Configuration under Windows 2000/XP

2.3.4 Parallel Port Address

The lines in DB25 connector are divided into three groups, they are:

- 1) Data lines
- 2) Control lines
- 3) Status lines

As the name refers, data is transferred over Data lines, Control lines are used to control the peripheral and of course, the peripheral returns status signals back to computer through Status lines. These lines are connected to Data, Control and Status registers internally. The registers are virtually connected to the corresponding lines. So whatever we write to these registers, will appear in corresponding lines as voltages. And whatever we give to Parallel port as voltages can be read from these registers (with some restrictions). For example, if we write '1' to Data register, the line Data0 will be driven to +5v. Just like this, we can programmatically turn on and off any of the data lines and Control lines. In an IBM PC, these registers are IO mapped and will have unique address. The data register resides at base address, status register at base address + 1 and the control register is at base address + 2. So once we have the base address, we can calculate the address of each registers in this manner. The table below shows the register addresses of LPT1 and LPT2. The 3BCh base address was originally introduced for Parallel Ports on early Video Cards. LPT1 is normally assigned base address 378h, while LPT2 is assigned 278h. 378h and 278h have always been commonly used for Parallel Ports. The lower case 'h' denotes that it is in hexadecimal. These addresses may change from machine to machine.

Table 2.2: Parallel Port Base Addresses

Address	Notes:
3BCh - 3BFh	Used for Parallel Ports which were incorporated in to Video Cards and now, commonly an option for Ports controlled by BIOS. - Doesn't support ECP addresses.
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

When the computer is first turned on, BIOS (Basic Input/output System) will determine the number of ports we have and assign device labels LPT1, LPT2 and LPT3 to them. The base address, usually called the Data Port or Data Register is simply used for outputting data on the Parallel Port's data lines (Pins 2-9).

Table 2.3: Data Port

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

This register is normally a write only port. If we read from the port, we should get the last byte sent. However, if our port is bi-directional, we can receive data on this address.

Table 2.4: Status Port

Base + 1	Status Port	Read Only	Bit No.	Properties
			Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

The Status Port (base address + 1) is a read only port. Any data written to this port will be ignored. The Status Port is made up of 5 input lines (Pins 10,11,12,13 and 15), an IRQ status register and two reserved bits. Bit 7 (Busy) is an active low input. If bit 7 happens to show a logic 0, this means that there is +5v at pin 11. If Bit 2 i.e. (nIRQ) shows a '1' then an interrupt has **not** occurred.

Table 2.5: Control Port

Base + 2	Control Port	Read/Write	Bit No.	Properties
			Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable bi-directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

The Control Port (base address + 2) was intended to be a write only port. When a printer is attached to the Parallel Port, four "controls" are used. These are Strobe, Auto Linefeed, Initialize and Select Printer, all of which are inverted except Initialize. However, these four outputs can also be used for inputs. Normally, data, control and status registers will have following addresses. We need these addresses in programming later.

Table 2.6: Summary of Register Addresses

Register	LPT1	LPT2
Data register (Base Address + 0)	0x378	0x278
Status register (Base Address + 1)	0x379	0x279
Control register (Base Address + 2)	0x37a	0x27a

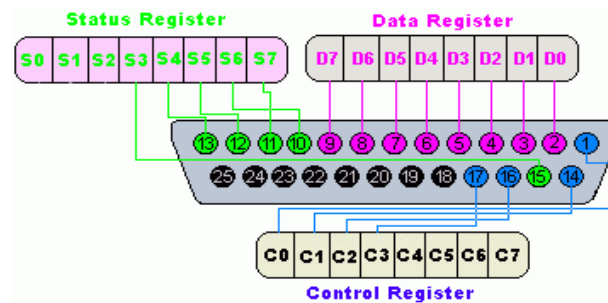


Figure 2.8: Pin outs of DB25 connector

Following shows a summary of all pins existing in a DB25 connector:

- 8 Output pins [D0 to D7]
- 5 Status pins [S4 to S7 and S3]
- 4 Control pins [C0 to C3]
- 8 ground pins [18 to 25]

Below table shows a summary of signals and associated pin numbers of a DB25 connector:

Table 2.7: Signals and Pin Numbers for General Purpose I/O

<i>Port</i>	<i>Signal Name</i>	<i>DB25 pin number</i>	<i>Comments</i>	
Data base	D ₀	2	All outputs latched	
	D ₁	3		
	D ₂	4		
	D ₃	5		
	D ₄	6		
	D ₅	7		
	D ₆	8		
	D ₇	9		
Status base + 1	bit 3	ERROR	15	input
	bit 4	SLCT	13	input
	bit 5	PE	12	input
	bit 6	ACK	10	input
	bit 7	BUSY	11	inverted input
Control base + 2	bit 0	STROBE	1	inverted output
	bit 1	AUTOFEED	14	inverted output
	bit 2	INIT PRN	16	output
	bit 3	SELECT	17	inverted output
	GND	18–25		

2.3.5 Sending Instruction to Parallel Port

As mentioned earlier, most recent computers that include a parallel port can operate the port in ECP or EPP mode, or both simultaneously. An IEEE-compliant cable must meet several standards of wiring and quality. Sending commands to the parallel port is very easy. We need to perform some “C” programming like below:

```
#include <stdio.h>

#include <dos.h>

void main(void)
{ outportb (0x378,0xFF);
}

```

By writing and executing the above script, we can set all data pins to 1. Now, if we take a LED and put one terminal at pin3 and the other to pin 18, it would glow. We have use a 2K resistor in series with the LED, otherwise it will end up ruining our LED, or source

too much current from the port pin which may damage the parallel port. Now, if we want to switch the LED off, we have to write below code:

```
outportb (0x378, 0x00);
```

instead of `outportb (0x378, 0xFF);`

2.3.6 Working Principle

As described before, 0x378 is the parallel port address. Usually this is the default address. Sometimes it is 0x278 as mentioned previously. 0x00 is the command appearing at the output pins. The Format is in Hexadecimal. So, if we want to make pin no 02 high, that will be the first data pin of the parallel port.

0x01 will mean 00000001 for the data port

0x04 will mean 00000100 for the data port

0x55 will mean 01010101 for the data port

0x0A will mean 00001010 for the data port

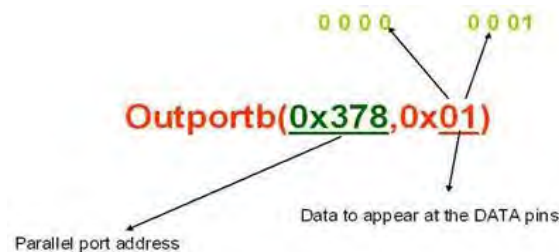


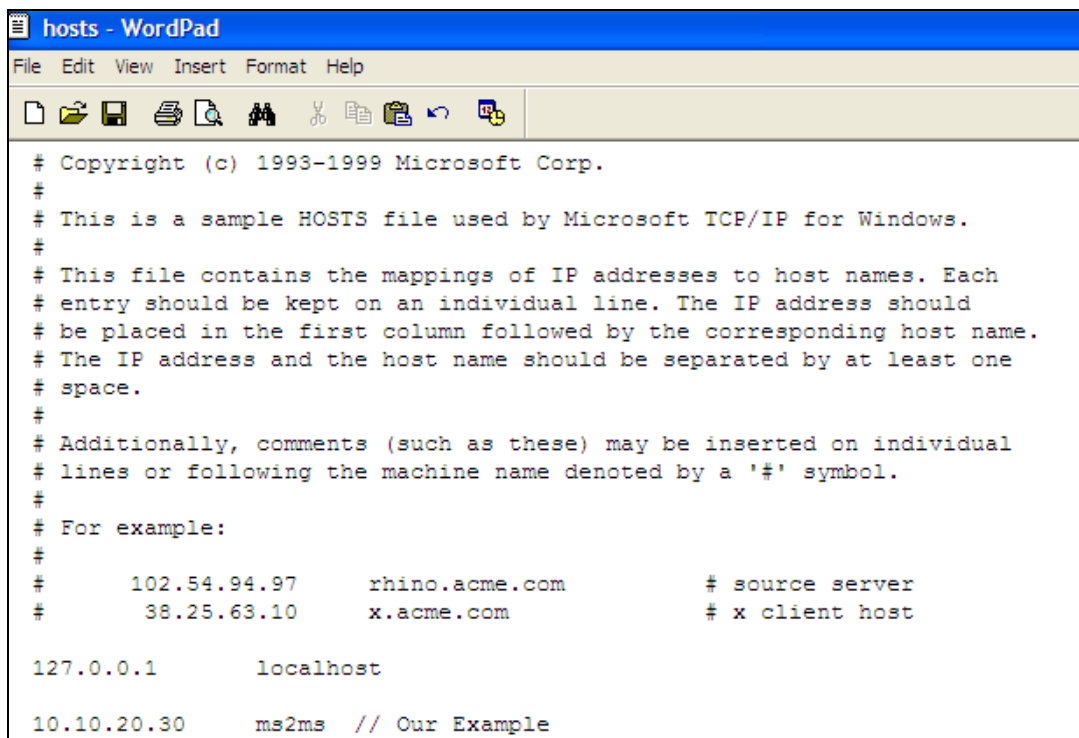
Figure 2.9: Parallel Port Programming

Using Windows 98, we can reach the ports with a function which is "outportb". But in Windows 2000 it will fail because of its kernel. We cannot reach the ports directly in NT, 2000 and XP because of their kernel and their drivers. In Windows 2000, we can see our by Settings > Control Panel > System > Hardware > Device Manager > Ports (COM and LPT) > Printer Port (LPT1) > Properties = in Resources > Resource Setting and we can see our address for our parallel port. This is hexadecimal like in math (mod 16). 0x378 belongs to 888 in decimal form. In this way, we can have a look for our port addresses.

To get access to parallel port under Windows NT, we need device driver like Inpout32.dll. The outstanding feature of Inpout32.dll is, it can work with all the NT windows versions without any modification in user code or the dll itself. Details of the dll are available at MSDN Online.

2.3.7 Steps for Loading Parallel Port Web Page

Before loading the parallel port web page we have to perform few activities in our Windows System32 folder and Apache configuration folder. We have to go to path: C:\WINDOWS\system32\drivers\etc of our PC and edit the 'hosts' file. Here, we have to insert the IP we received after dialing the APN name. This IP can also be found from command prompt typing syntax 'ipconfig'. While configuring the IP in host file, we can put a name against it. Next time, this name can be used to load the web page instead of using the IP.



```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10      x.acme.com             # x client host

127.0.0.1        localhost

10.10.20.30     ms2ms // Our Example
```

Figure 2.10: C:\WINDOWS\system32\drivers\etc\hosts File Configuration

Apache is configured by editing plain text configuration files. The main configuration file is usually called httpd.conf which is hosted at Apache's configuration folder. Here, we also have put the IP which we received after dialing the APN and the desired port for communication i.e. port 80 for http and port 443 for https.

After configuring those two files, we have to check below in sequential manner:

- ❖ Apache web server is up and running
- ❖ Parallel port controlling web page (php + html) containing port controlling script
- ❖ Parallel port control program which will ensure input and output data through the pins
- ❖ Insertion of html and port control file (port control page) at \xampp\htdocs
- ❖ Insertion of inpout32.dll files at \xampp\htdocs
- ❖ Loading the html file from web browser

The web interface part:

- ❖ It includes web server having web pages and needed scripts to take user controls
- ❖ The server end is based on small piece of PHP code and port control software
- ❖ Port control is a simple general purpose I/O port control program
- ❖ It allows us to write and read the supported I/O ports

The hardware controlling part:

- ❖ Port controlling part needs to be written using C language, because most scripting languages usually lack the features to do actual direct hardware controlling
- ❖ The idea is that the actual low level hardware controlling is done with simple C program, and then the script sends controls to this C program in some way

Control of parallel port will work in the following way:

- ❖ User loads a web page that has the required control features (forms, buttons, etc.)
- ❖ User presses control button to parallel port via port control program

2.4 Wireless Network

Wireless M2M can be supported by almost all wireless technologies that include GPRS/EDGE, UMTS/3G, LTE etc. The standard was expanded over time to time to include first circuit switched data transport, then packet data transport via GPRS [12]. Packet data transmission speeds later increased via EDGE [13]. The GSM standard is succeeded by the third generation (or "3G") UMTS standard developed by the 3GPP [14,15]. GSM networks will evolve further as they begin to incorporate fourth generation (or "4G")/ LTE Advanced standards [16]. The network is consisted of a number of elements which are briefly shown in below figure:

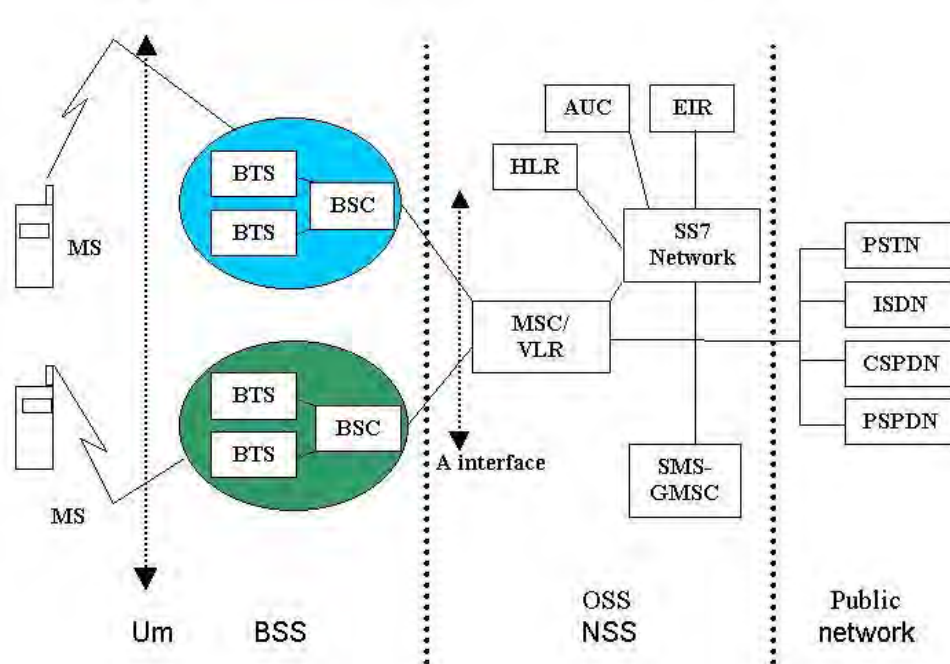


Figure 2.11: Basic GSM Network Architecture

In this project, mobile station (MS) plays a major role in client-server communication.

Brief description on mobile station and its relative components are given below.

Mobile Station:

- *Mobile Equipment (ME)* – this is the GSM terminal, excluding the SIM card
- *Subscriber Identification Module (SIM)* – this is the chip embedded in the SIM card that identifies a subscriber of a GSM network; the SIM is embedded in the SIM card.

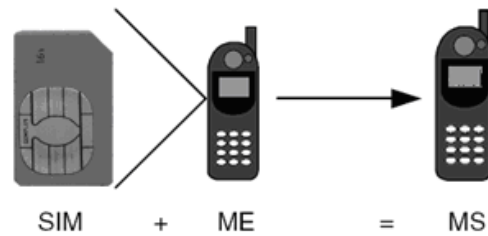


Figure 2.12: Mobile Station Components

When the SIM card is inserted in the ME, the subscriber may register with a GSM network. The ME is now effectively personalized for this GSM subscriber. The characteristics of the SIM are specified in GSM TS 11.11. For the UMTS network an enhanced SIM is specified, called- universal subscriber identity module (USIM) and specified in 3GPP TS 31.102.

The international mobile subscriber identity (IMSI) is embedded on the SIM card and is used to identify a subscriber. The IMSI is also contained in the subscription data in the HLR. The IMSI is used for identifying a subscriber for various processes in the GSM network.

Mobile Station Integrated Services Digital Network Number (MSISDN) is also used to identify the subscriber. The MSISDN is not stored on the subscriber's SIM card and is normally not available in the MS. The MSISDN is provisioned in the HLR, as part of the subscriber's profile, and is sent to MSC during registration.

2.4.1 GPRS Network Architecture

GPRS (General Packet Radio Service) is a packet based service for mobile devices that allows data to be sent and received across a mobile telephone network [17]. GPRS uses most of the GSM nodes mainly BSC, HLR, AUC and EIR to provide packet switched service. In addition, it has specific nodes which have made it distinct from other technologies. There are two new functional elements which play a major role in how GPRS works- The Serving GPRS Support Node (SGSN) and the Gateway GPRS support node (GGSN).

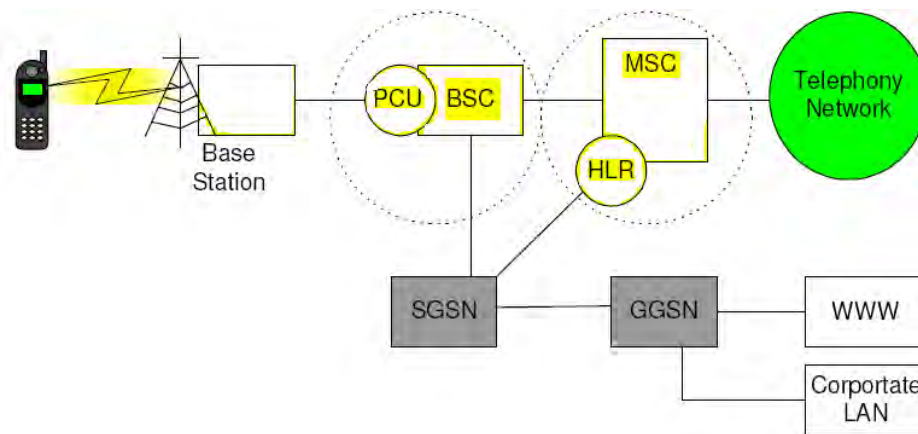


Figure 2.13: Simplified Architecture of GPRS Network

Also, packet control unit (PCU) will be required at base station controller (BSC) side to support packet switched data. Brief description of different elements is stated below:

Base Station Subsystem (BSS):

BSS is composed of one or more base station controllers (BSC) and one or more base transceiver stations (BTS). The BTS contains one or more transceivers (TRX). The TRX is responsible for radio signal transmission and reception.

Home Location Register (HLR):

HLR is the database that contains a subscription record for each subscriber of the network. A subscriber is normally associated with one particular HLR. The HLR is

responsible for sending subscription data to the VLR (during registration). When an individual buys a subscription from one of the operators, he or she is registered in the HLR of that operator.

Authentication Center (AUC):

The authentication center (AUC) provides authentication and encryption parameters that verify the user's identity and ensure the confidentiality of each user. The GSM has standard encryption and authentication algorithm which are used to dynamically compute challenge keys and encryptions keys for a call.

GPRS Terminals:

New terminals are required because existing GSM phones do not handle the enhanced air interface, nor do they have the ability to packetize traffic directly.

Packet Control Unit:

Each BSC will require the installation of one or more PCUs and a software upgrade. The PCU provides a physical and logical data interface out of the base station system (BSS) for packet data traffic. The BTS may also require a software upgrade, but typically will not require hardware enhancements. When either voice or data traffic is originated at the subscriber terminal, it is transported over the air interface to the BTS, and from the BTS to the BSC in the same way as a standard GSM call. However, at the output of the BSC the traffic is separated; voice is sent to the mobile switching center (MSC) and data is sent to SGSN, via the PCU over a Frame Relay or IP network.

Serving GPRS Support Node (SGSN):

The SGSN delivers packets to mobile stations (MSs) within its service area. SGSN sends queries to home location registers (HLRs) to obtain profile data of GPRS subscribers. SGSN detects new GPRS MSs in a given service area, process registration of new mobile subscribers, and keep a record of their location inside a given area. Therefore, the SGSN performs mobility management functions such as mobile subscriber attach/detach and

location management. The SGSN is connected to the base-station subsystem via a Frame Relay or IP connection to the PCU in the BSC.

Gateway GPRS Support Node (GGSN):

GGSN is used as interfaces to external IP networks such as the public Internet, other mobile service providers' GPRS services, or enterprise intranets. GGSN maintain routing information that is necessary to tunnel the protocol data units (PDUs) to the SGSN that serve particular MS.

IP Allocation Mechanism:

While using GPRS service, each device gets an IP. This IP is allocated by the GPRS core network [18]. There are 3 different ways in which a device can be assigned an IP address which are as below:

Fixed IP addressing:

Fixed IP addresses for mobile devices are provided by HLR.

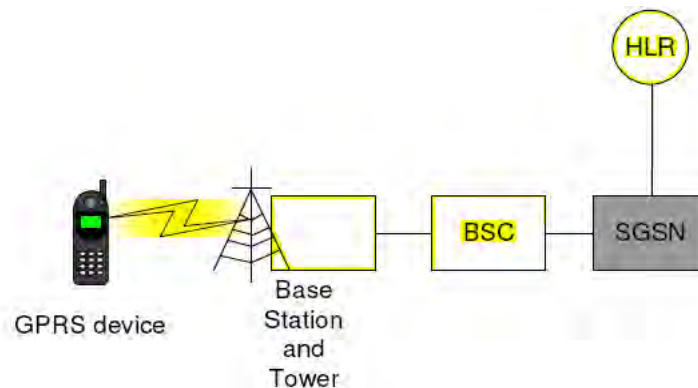


Figure 2.14: Static IP Addressing Mechanism by HLR

Dynamic IP addressing:

Here a mobile device does not have its own IP address stored in the HLR. Instead, the IP address is assigned by the GGSN from its DHCP (Dynamic Host Control Protocol) pool.

The third method is also a type of dynamic IP addressing in which the IP address is assigned by 'Remote Authentication Dial in User Service' (RADIUS) server GGSN.

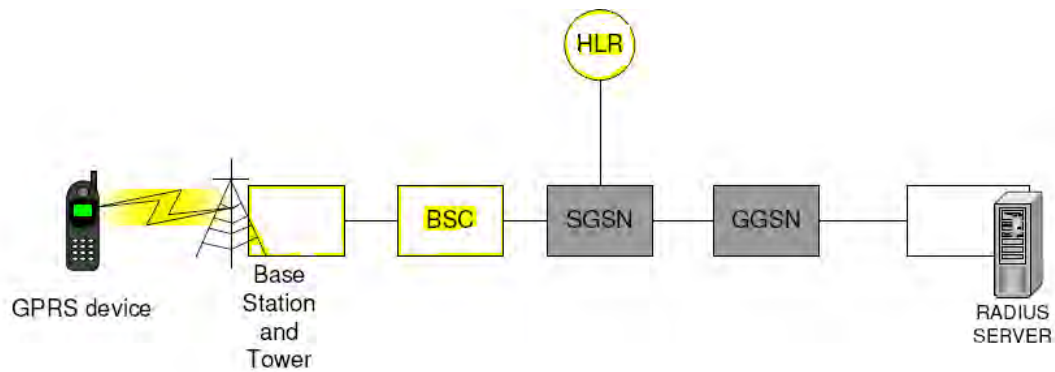


Figure 2.15: Dynamic IP Addressing Mechanism by GGSN/RADIUS

2.4.2 EDGE Network Architecture

Enhanced Data rates for GSM Evolution (EDGE) or Enhanced Data rates for Global Evolution is a digital mobile phone technology that allows improved data transmission rates as a backward-compatible extension of GPRS. EDGE is introduced within existing specifications and descriptions rather than by creating new ones [19].

2.4.3 Data Rates of GPRS/EDGE Network

GPRS allows data rates theoretically up to 170 kbps on the physical layer. EGPRS/EDGE is capable of offering data rates theoretically up to 473.6 kbps. A new modulation technique and error-tolerant transmission methods, combined with improved link adaptation mechanisms, make these EGPRS rates possible. This is the key to increased spectrum efficiency and enhanced applications, such as wireless Internet access, e-mail and file transfers.

To achieve higher bit rates per time slot than those available in GSM/GPRS, the modulation method requires a change. EDGE is specified to reuse the channel structure, channel width, channel coding and the existing mechanisms and functionality of GPRS

and HSCSD. The modulation standard selected for EDGE, 8-phase shift keying (8PSK), fulfills all of those requirements. 8PSK modulation has the same qualities in terms of generating interference on adjacent channels as GMSK. The 8PSK modulation method is a linear method in which three consecutive bits are mapped onto one symbol in the I/Q plane. The symbol rate, or the number of symbols sent within a certain period of time, remains the same as for GMSK, but each symbol now represents three bits instead of one. The total data rate is therefore increased by a factor of three. The distance between the different symbols is shorter using 8PSK modulation than when using GMSK. Shorter distances increase the risk for misinterpretation of the symbols because it is more difficult for the radio receiver to detect which symbol it has received. Under good radio conditions, this does not matter. Under poor radio conditions, however, it does. The “extra” bits will be used to add more error correcting coding, and the correct information can be recovered. Only under very poor radio environment GMSK is more efficient. Therefore the EDGE coding schemes are a mixture of both GMSK and 8PSK.

For GPRS, four different coding schemes, designated CS1 through CS4, are defined. Each has different amounts of error-correcting coding that is optimized for different radio environments. For EGPRS, nine modulation coding schemes, designated MCS1 through MCS9, are introduced. These fulfill the same task as the GPRS coding schemes. The lower four EGPRS coding schemes (MCS1 to MCS4) use GMSK, whereas the upper five (MCS5 to MCS9) use 8PSK modulation. GPRS user throughput reaches saturation at a maximum of 20 kbps with CS4, whereas the EGPRS bit rate continues to increase as the radio quality increases, until throughput reaches saturation at 59.2 kbps. Both GPRS CS1 to CS4 and EGPRS MCS1 to MCS4 use GMSK modulation with slightly different throughput performances. This is due to differences in the header size (and payload size) of the EGPRS packets. This makes it possible to re-segment EGPRS packets. This re-segmenting (retransmitting with another coding scheme) requires changes in the payload sizes of the radio blocks, which is why EGPRS and GPRS do not have the same performance for the GMSK modulated coding schemes.

2.4.4 GSM Security Features

System security of GSM networks is based primarily on the verification of equipment and subscriber identity [19]. Confidential data and keys are stored and generated in the AUC (Authentication Center). The EIR (Equipment Identity Register) stores the serial numbers (supplied by the manufacturer) of the terminals (IMEI). In GSM, subscriber is identified by International Mobile Subscriber Identity (IMSI), Mobile Subscriber ISDN Number (MSISDN), Temporary IMSI (TIMSI); Mobile station Roaming Number (MSRN) and Mobile Equipment is identified by International Mobile Station Equipment Identity (IMEI). When registering for service with a mobile network operator, each subscriber receives a unique identifier called IMSI. This IMSI is stored in the SIM. The real telephone number of MS is MSISDN. The VLR is responsible for the current location of a subscriber can assign a TIMSI which has only local significance in the area handled by the VLR. TIMSI is used in place of the IMSI for definite identification and addressing of the MS. This way nobody can determine the identity of the subscriber by listening to the radio channel, since this TMSI is only assigned during the mobile stations presence in the area of one VLR and can even be changed during this period. The mobile station stores the TMSI on the SIM card. The TMSI is stored on the network side only in the VLR and is not passed to the HLR. The association between IMSI and TMSI is stored in the VLR. The MSRN is a temporary location-dependent ISDN number which is assigned by the local VLR in its area. The IMEI uniquely identifies mobile equipment internationally. It is a kind of serial number. The IMEI is allocated by the equipment manufacture and registered by the network operator who stores it in EIR.

2.4.5 General Authentication Procedure and Data Confidentiality

When a subscriber is added to a network for the first time, a Subscriber Authentication Key (Ki) is assigned in addition to the IMSI to enable the authentication. At the network side, the key Ki is stored in the AUC of the home PLMN. At the subscriber side, the Ki must be stored in the SIM. The authentication procedure is based on the A3 algorithm. The A3 algorithm is implemented at both the network side and the MS side. This

algorithm calculates independently on both sides the Signature Response (SRES) from the K_i and a Random Number (RAND) offered by the network (i.e., $SRES=A3(K_i, RAND)$). The K_i and IMSI are allocated at subscription time. The MS transmits its SRES value to the network that compares it with its calculated value. If both values agree, the authentication is successful. Each execution of the A3 algorithm is performed with a new value of RAND which cannot be predetermined; in this way tapping the transmission channel cannot be used to copy an identity.

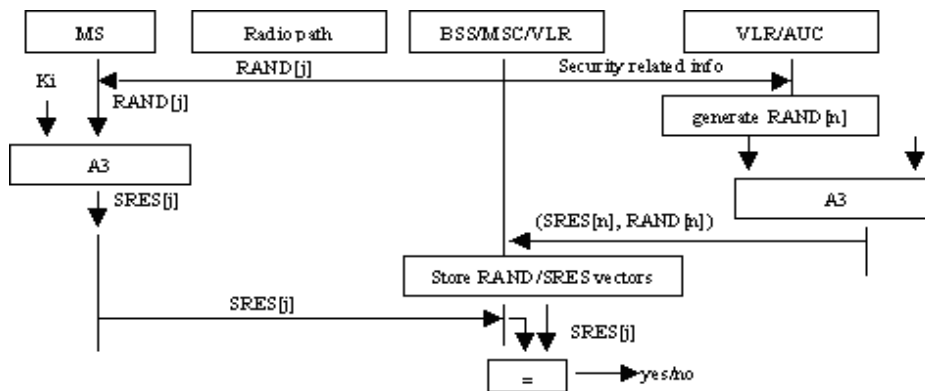


Figure 2.16: General Authentication Procedure

The signaling information elements related to the user, such as IMEI, IMSI, and Calling subscriber directory numbers need to be protected after connection establishment. In order to achieve confidentiality, a ciphering method, key setting, starting of the enciphering and deciphering processes, and a synchronization are needed. A ciphering method A5 is used to encrypt voice and signaling data. The layer 1 data flow (transmitted on Dedicated Control Channel (DCCH) or Traffic Channel (TCH)) is obtained by the bit per bit binary addition of the user data flow and a ciphering bit stream. The encryption of signaling and user data is performed at the MS as well as at the BSS. This is a case called symmetric encryption, i.e. ciphering and deciphering are performed with the same K_c and the A5 algorithm and start on DCCH and TCH. The scope of GSM is between BTS and MS. The scope of GPRS is from the SGSN to the MS. A new ciphering algorithm GPRS-

A5 is used because of the nature of GPRS traffic. The ciphering is done in the Logical Link Control (LLC) layer. The GPRS-Kc is handled by the SGSN.

2.4.6 Attach and PDP Context Activation Procedure

Before a mobile station (MS) can send data to a corresponding host, it must attach to a Serving GPRS Support Node (SGSN). An attachment procedure (GPRS attach) between the mobile station and the network is conducted and a Temporary Logical Link Identifier (TLLI) is assigned to the mobile station. Actually, the mobile uses the TLLI after the network assigns a Packet Temporary Mobile Subscriber Identity (P_TMSI). After attaching, one or more routing contexts for one or more Packet Data Protocols (PDPs) can be negotiated with the SGSN.

Three mobility management (MM) states are related to a GPRS subscriber and each state describes the level of functionality and information allocated. In idle state, the mobile station is not yet attached to the GPRS mobility management and a GPRS attach procedure must be performed. In ready state, the mobile station is attached to GPRS mobility management (GMM) and is known in the accuracy of the cell. Each cell in a GSM network has its own Cell Global Identity (CGI), which enables the network to identify the mobile station by the cell. Each cell is associated with a location area (LA) in GSM, but in GPRS, the association is with the routing area (RA). In this state, the mobile station may receive and send data for all relevant service types. If the ready timer (for the mobile station or SGSN) expires, the mobile station will move to the standby state.

In the standby state, the subscriber is attached to the GPRS mobility management and is known in the accuracy of the routing area (RA) and not in the accuracy of cell. At this point, if the subscriber wants to request e-mail or a web page, a PDP context must be activated in advance. If the standby timer (for the mobile station or SGSN) expires in this state, the mobility management contexts in both the mobile station and SGSN independently return to the idle state and may be deleted.

After the GPRS Attach procedure is over, the terminal is prepared for the data communication. When terminal wants to start the communication it initiates establishing of Packet Data Protocol Context (PDP context). Saying with other words: After GPRS attach, when the terminal is attached to an SGSN, it must activate a PDP address (in this case, an IP address) when it wishes to begin a packet data communication. This is usually done by application request from mobile station, but in some situations user could decide to be 'on-line' for the whole time and then the data connection is established during the phone boot sequence.

The SGSN identifies the corresponding GGSN for requested PDP and makes it aware of the mobile device. Then two-way point-to-point tunnel is established between the SGSN and the GGSN. In other words, activating a PDP address sets up an association between the mobile device's current SGSN and the GGSN that anchors the PDP address. A record is kept regarding the associations made between the GGSN and SGSN. This record is known as a PDP context. PDP context is describing the characteristic of the connection: network and address type, APN, QoS, priorities, billing, etc.

2.5 Integration with Wireless Network

Already discussed in the previous sections, we need to dial an APN to get a particular IP address. At server side, we can follow this process. However, at client side, where mobile set will be used, we have to use specific settings and have to push a particular button to activate packet data service. After activating the PDP context, the handset will receive an IP. This IP can be seen if smart phones are used at client side. Both of these IPs will fall under the same subnet. As a result, we shall be able to ping each other using the cellular data network. When ping is successful, it means, we have already got a transparent channel between client and server. It will help us for remote controlling and monitoring of a device. Considering the importance, this section will be discussed in more detail at next chapter.

CHAPTER 3

DESIGN and IMPLEMENTATION

Design and Implementation of this project involves hardware designing and working with scripting or coding. It also involves networking through mobile operator's data network. For simplicity; development and implementation of hardware part along with networking will be discussed in detail before going through the scripting or coding part.

3.1 Introduction

Hardware section can be divided into three major blocks. One of the blocks is the external driving circuit and the second block contains logic circuit which will determine the presence of electricity at remote site.

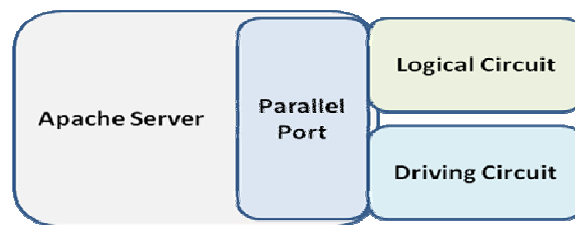


Figure 3.1: High Level Block Diagram for Design and Implementation

The third block is the Apache Server itself which has been discussed in detail in the previous chapter. As mentioned earlier, the server will contain integrated parallel ports. All of these blocks will interact with each other via the parallel port.

3.2 Requirements of Driving Circuit

Since parallel ports are very sensitive and can source or sink limited amount of current, an external driving circuit has been developed which will help the parallel ports to keep away from any accidental operation.

In this project, though we have the facility to control eight distinct loads, it is imperative that controlling two loads will be enough to rationalize the objective of this project. To implement the proposed system, we shall require below:

MOSFET	: IRF540	- Quantity: 02
Opto-coupler	: 4N35	- Quantity: 02
Resistor	: 1K Ω	- Quantity: 02
Resistor	: 320 Ω	- Quantity: 02
Relay	: 12V DC	- Quantity: 02
Relay	: 6VDC	- Quantity: 02
Diode	: IN4007	- Quantity: 02
Power Supply	: DC12V	- Quantity: 02
Power Supply	: DC06V	- Quantity: 01
Load	: Fan and Light	- Quantity: 02

Load:

One 12V rated small fan and one 12V rated small bulb have been used as load. These loads will be connected via the 12VDC relays which will be driven by the external circuit consisting of MOSFET, Opto-Coupler, Diode, Resistance etc.

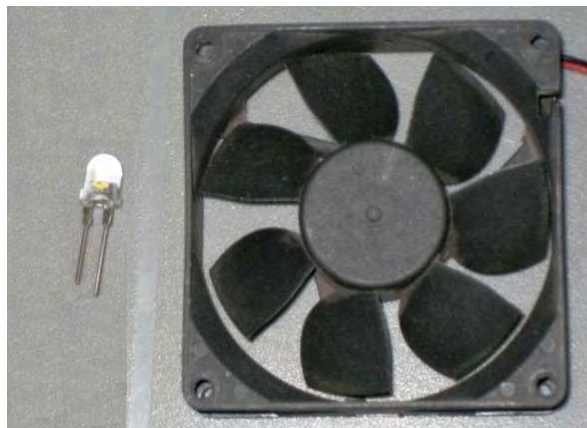


Figure 3.2: One Light and one Fan used as load

3.3 External Driving Circuit's Working Principle

Already mentioned, parallel port has eight data pins, which can generate logical 0V and 5V. Here, using Web interface, we can set/reset the data pins of the parallel port. Next to parallel port, opto-coupler is used, which will isolate input and output signal thus help parallel port from overloading. Pin-01 of opto-coupler is connected to parallel port's data pin D0 and pin-02 of opto-coupler is connected to parallel port's Ground (GND) pin-18. VCC (12V) is connected to opto-coupler's pin-05. Here, opto-coupler is driving MOSFET - IRF540. Gate of IRF540 is connected to opto-coupler's pin-04. Generating 5V to data pin D0 (position-02 of parallel port) will energize the primary side of the opto-coupler and it will be turned on. Consecutively MOSFET will be turned on having minimum gate voltage V_{gs} .

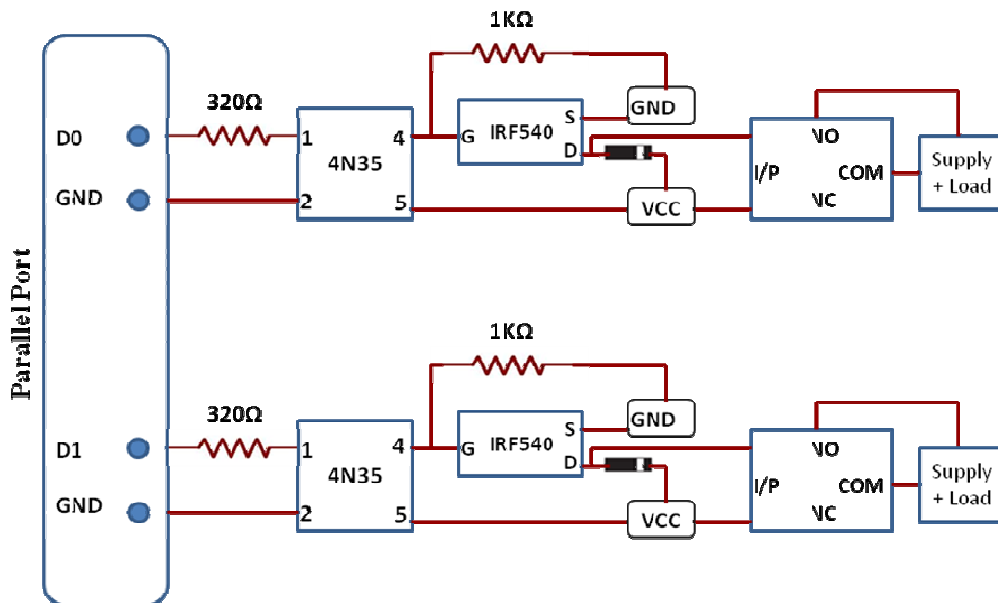


Figure 3.3: Block Diagram of External Driving Circuit

Relays input side is connected to MOSFET's output i.e. at Drain terminal. When the MOSFET is on, it will energize the coil of relay. As a result, normally open contact will become closed. Now if we use a load and supply between normally open (NO) contact and Common (COM), when NO will be closed, circuit at secondary side of the relay will

be completed and load will be able to draw necessary current from supply. It will act like a switch.

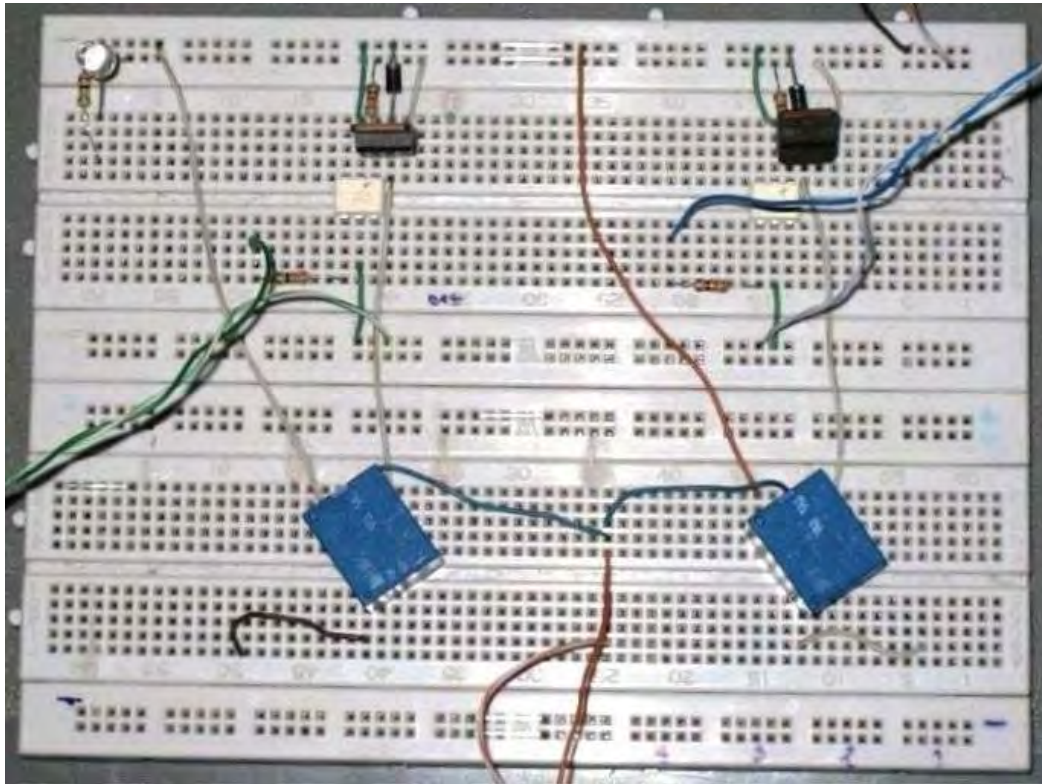


Figure 3.4: External Driving Circuit

The essence here is; parallel port is completely isolated from load and risk free from accidental operation. Generating 0V at pin D0 of parallel port will de-energize opto-coupler; MOSFET will stop supplying current to relay's primary (input) side. The relay will be turned off; means normally open contact will be at its usual open state and load will be disconnected from supply. In this way, we can switch on/off as many as eight loads.

3.4 Logic Circuit's Working Principle

This part of the circuit will be used to determine load shedding status at remote site. Taking input via the parallel port is independent of load controlling part which provides much flexibility. If there is load shedding, client side will be updated accordingly via the web browser. To develop this circuit two relays are used as indicated in the figure. Parallel port's data pin D8 is connected to the upper relay's normally open (NO) terminal. Parallel port's pin-13 will be used to take input signal.

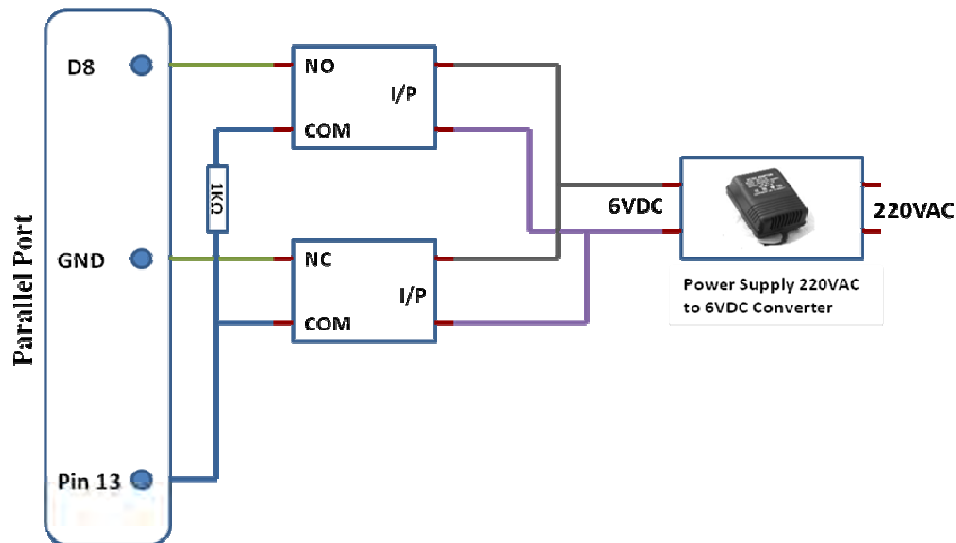


Figure 3.5: Block Diagram of Logic Circuit

Common (COM) terminal of the upper relay is connected to parallel port's pin-13 via a resistor $1k\Omega$ only for safety purpose. This will limit the flow of current between two pins of parallel port. Lower relay's normally closed (NC) terminal is connected to parallel ports ground (GND) pin-20. Common (COM) terminal of the lower relay is also connected to parallel port's pin-13. Primary sides of both of the relays are connected to a 220VAC to 6VDC converter and the input terminals are actually in parallel with the converter's 6V DC output. When there is electricity at remote side, both of the relay's input terminals will get necessary current from the converter and will be powered up. Now for the upper relay, normally open (NO) terminal will become closed and for the lower relay, normally closed (NC) terminal will become open.

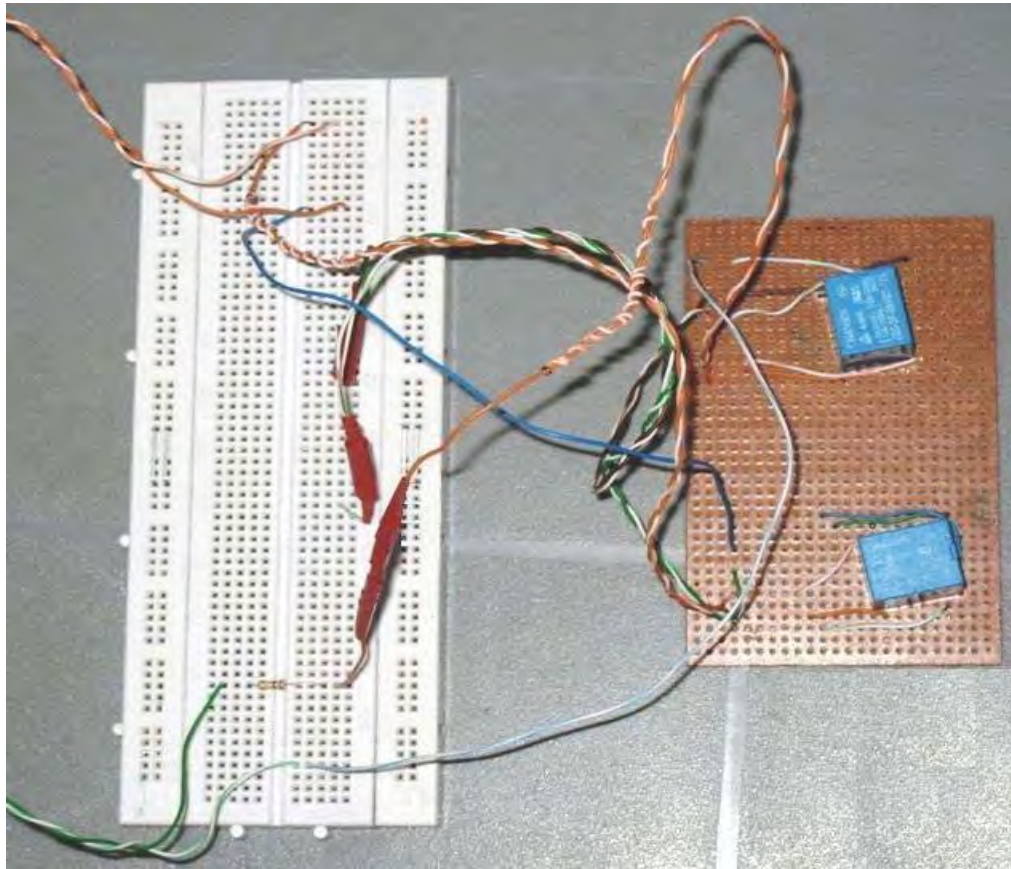


Figure 3.6: Logic Circuit

Now, if D8 is at high state, pin-13 will get high logic and accordingly the web browser will show no load shedding status. When there is no electricity at the secondary site, output of the converter is 0V. Both of the relays will be at de-energized state. Upper relay's normally open (NO) terminal will not be able to send high logic to parallel port's pin-13 via the COM port. On the contrary, lower relay's normally closed terminal will forward zero logic from ground pin-20 to pin-13. Whenever pin-13 will receive low logic, web browser will show load shedding status.

3.5 Scripting/ Coding

For simplicity, client side i.e. mobile station will load two separate web pages from Apache web server. One page i.e. Controlport.php has port control buttons and another page i.e. Loadshedding.php shows status of load shedding at server side. Both pages here contain Java Script and HTML codes which will be processed by PHP.

3.5.1 Discussion on Controlport.php

The code contained in Controlport.php is responsible for collecting port status calling the function portstatus (). It uses shell_exec - which executes command via shell and returns the complete output as a string. Here portcontrol.exe is a general purpose I/O port reading and writing application which is a free tool to be used. This application is planned to access LPT1 parallel port at I/O address 0x378. It allows us to write and read the supported I/O ports.

Command insertion:

portcontrol.exe **LPT1DATA read setbit 1 write** - Reads parallel port data, set bit D1 to 1 and writes value back. It effectively sets the LPT1 pin D1 in parallel port to logic 1.

portcontrol.exe **LPT1DATA read resetbit** - Reads parallel port data, set bit D1 to 0 and writes value back. It effectively sets the LPT1 pin D1 in parallel port to logic 0.

portcontrol.exe **LPT1DATA setvalue 0xff write** - Writes value 0xff (255 dec) to the parallel port. It effectively sets all LPT1 data pins to logic 1.

portcontrol.exe **LPT1DATA read print bin** - Reads the parallel port data pin states and prints the results as binary numbers to the screen (standard output).

portcontrol.exe **LPT1DATA read printbits 021** - Reads the parallel port data pin states and prints the states of bits D0, D2 and D1 in that order to the screen (standard output).

Within the code `<? php text ?>` part elaborates the functionality of portcontrol.exe library function and initializes the sajax.php java script file. Sajax is an asynchronous JavaScript and XML (Ajax) with PHP, which facilitates to refresh entire Web pages without each user click. There are several aspects of Sajax; one of them is sajax_init, which initializes the Sajax library. Next, there is a sajax_export function which is called to notify Sajax that we have a "panels" content section, for which corresponding JavaScript functions will be created later. Sajax_export can be called as many times as necessary for each of the dynamic content sections that our application may require. The next function to be used is: sajax_handle_client_request. This function initializes the Sajax data structures, preparing an application to handle client requests. \$sajax_remote_uri will be the URL where the client requests of an application will be sent. Finally, Sajax JavaScript functions have been included within JavaScript using the sajax_show_javascript function. The status bar should automatically update every nth second with the parallel port status and show current time on the server . Pressing the buttons will change the parallel port bit states.

`<html></html>` handles the html coding part. It calls the java scripts. This organizes the html codes to create the web browser forms; it also handles the relevant procedure while running the program from the form of the WebPages. Html body is coded to design the format of the web page form.

3.5.2 Discussion on Loadshedding.php

The code contained in loadshedding.php is also responsible for collecting port status by calling the function portstatus (). In addition to this, this code is responsible to show the load shedding status. It also uses shell_exec. Sajax.php works as before stated in previous section.

Command insertion:

portcontrol.exe **LPT1DATA read print dec** - Reads the parallel port data pin states and prints the results as decimal numbers to the screen (standard output).

Here the code reads external signal at pin 13 and shows output as 'Yes' or 'No' at screen. It compares the parallel port state value in decimal with reference value 120 and takes the decision.

3.5.3 Supported Commands and Identifiers

Following format has to be followed to instruct command to portcontrol.exe:

Supported commands:

PRINT DEC	Read data, gives output as decimal number
PRINT HEX	Read data, gives output as hexadecimal number
PRINT BIN	Read data, gives output as binary number
PRINTBITS bits	Reads the specified bits in the specified order (0...7)
WRITE	Writes register value to port
READ	Reads value from specified port to register
SETVALUE value	Sets the given value to register
AND value	Performs AND with given value and register
OR value	Performs OR with given value and register
XOR value	Performs XOR with given value and register
SETBITS bits	Sets given bits to 1 in register
RESETBITS bits	Sets given bits to 0 in register
SETBIT bits	Same as SETBITS
RESETBIT bits	Same as RESETBITS
TOGGLEBITS bits	Toggles specified bit values

Supported port identifiers:

LPT1DATA	LPT1 port data lines controlling (0x378)
LPT1STATUS	LPT1 printer status inputs register
LPT1HANDSHAKE	LPT1 handstake output controlling
JOYSTICK	Joystick port reading (only reading makes sense)

Value can be given as decimal number or hexa-decimal starting from 0x. 'Bits' is a list of bit position identifiers from 0 to 7. Port names have to be written at higher case. Commands have to be written at lower case.

3.6 File Organization

Already stated, all of these files will be within C:\xampp\htdocs folder to be accessed via web browser. For convenience, following is a summary of these files.

- Inpout32.dll
- Controlport.php
- Loadshedding.php
- Sajax.php
- Portcontrol.exe

3.7 Networking between Server and Client

To enable communication between client and server, one of the mobile operator's EDGE/GPRS network is used. Special type of APN will permit point to point reach ability. Already mentioned, at server side EDGE/GPRS enabled modem will be required to get connected with mobile operator's data network. Using the dialer software of the modem and dialing the specific APN name (which provides point to point connectivity), sever side will receive an IP either statically or dynamically. Static IP provides more stability and security than dynamic IP. At client side a mobile set will be using the same APN settings and will get connected to operator's EDGE/GPRS network. It will also receive the IP either statically or dynamically. Point to be noted, if server gets the IP statically, client will also get the IP statically. After having the IP, server and client will be able to ping each other. This will ensure first level reach ability between server and clients.



Figure 3.7: Configuring APN on Device

3.7.1 Data Flow Diagram

Already discussed in the previous chapter, GGSN performs IP routing and provides IP when dynamic IP allocation method is used. Here in this project, dynamic IP allocation arrangement will be selected for simplicity. Only disadvantage is, every time we dial, we shall get a new IP (in some cases old IP will be released by network) and we have to change configuration in Window's hosts file and httpd.conf file of APACHE application.

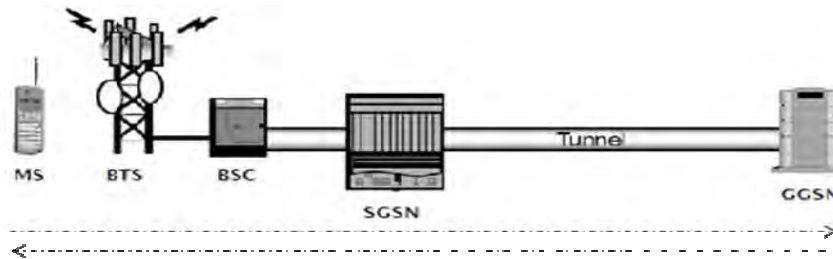


Figure 3.8: IP Connectivity between MS (Handset) and GGSN

In a point to point APN communication, client-server (end to end) communication will take place like below:

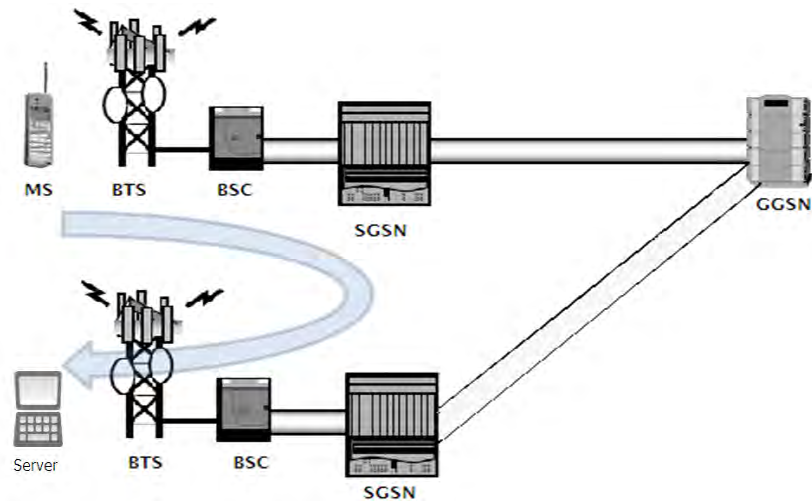


Figure 3.9: Client-Server Communication

Client will assume that there is a direct connectivity with server though in between client and server there is a vast packet data network of a mobile operator. For successful data flow between client and server, bandwidth required at air interface will be significant. Mobile operator's current data network can't commit guaranteed bandwidth at air interface due to technology limitation. Keeping this in mind, data transfer rate between client and server has been kept as minimum as possible.

3.7.2 Loading the Desired Web Page

To load the controlport.php and loadshedding.php pages, we have to type the subsequent URL at our web browser - <http://ms2device.com>. Point to be noted, we have to edit 'hosts' file located in C:\WINDOWS\system32\drivers\etc and httpd.conf which is located at configuration folder of Apache server (as stated at chapter 2) before we go for loading the URL. Any anomaly between these two files will prohibit loading the desired URL.

Editing hosts file:

127.0.0.1 localhost

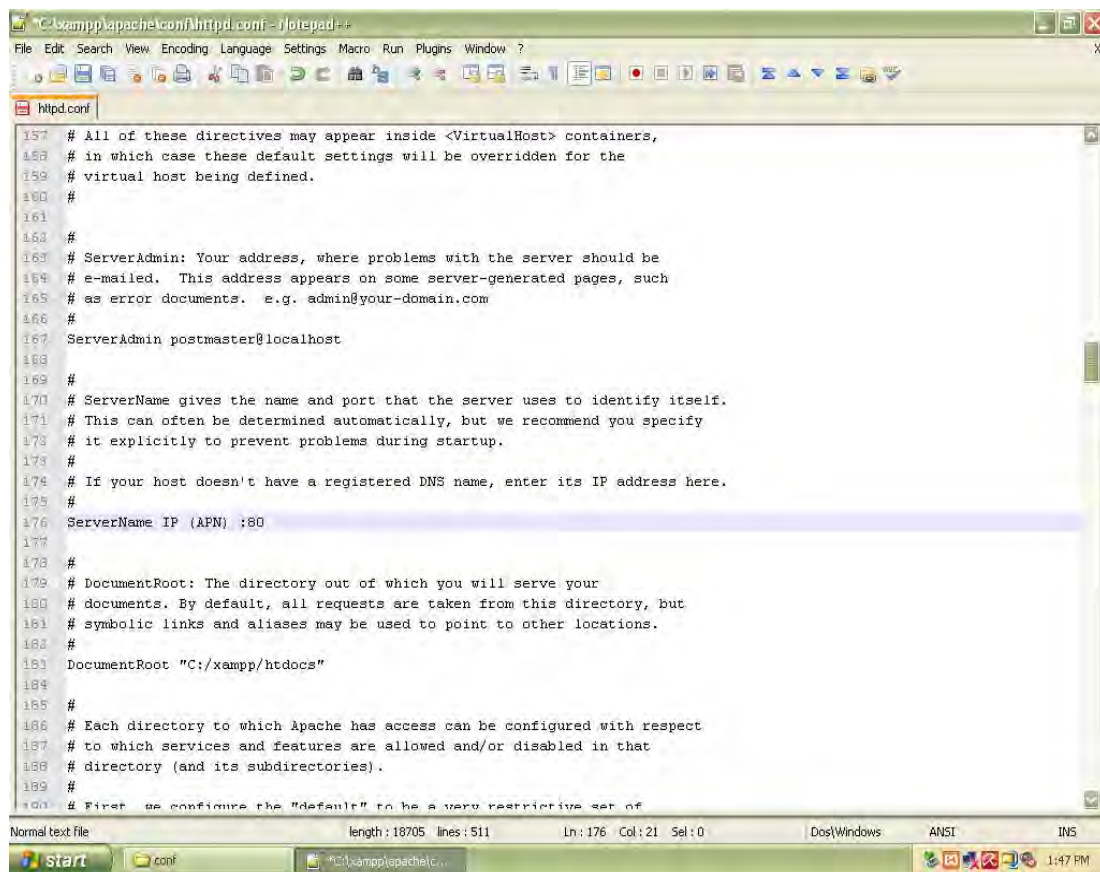
IP (APN) http://ms2device.com

Here, IP (APN) will be received after dialing the appropriate APN using the dialer.

Example: 10.10.20.30 http://ms2device.com

Editing httpd.conf file:

We have to edit line number 176 (shaded) under 'server name' section like below. In this case, the IP will be same as used in 'hosts' file i.e. 10.10.20.30. Port 80 is used to listen at http request.

A screenshot of a Notepad++ window titled "C:\xampp\apache\conf\httpd.conf - [Notepad++]". The window shows the contents of the httpd.conf file. Line 176 is highlighted in blue and contains the text "ServerName IP (APN) ;80". The status bar at the bottom indicates "Normal text file", "length: 18705 lines: 511", "Ln: 176 Col: 21 Sel: 0", "Dos|Windows", "ANSI", "INS", and the system tray shows the Start button, a folder icon, and the time "1:47 PM".

```
157 # All of these directives may appear inside <VirtualHost> containers,
158 # in which case these default settings will be overridden for the
159 # virtual host being defined.
160 #
161 #
162 #
163 #
164 # ServerAdmin: Your address, where problems with the server should be
165 # e-mailed. This address appears on some server-generated pages, such
166 # as error documents. e.g. admin@your-domain.com
167 #
168 ServerAdmin postmaster@localhost
169 #
170 # ServerName gives the name and port that the server uses to identify itself.
171 # This can often be determined automatically, but we recommend you specify
172 # it explicitly to prevent problems during startup.
173 #
174 # If your host doesn't have a registered DNS name, enter its IP address here.
175 #
176 ServerName IP (APN) ;80
177 #
178 #
179 # DocumentRoot: The directory out of which you will serve your
180 # documents. By default, all requests are taken from this directory, but
181 # symbolic links and aliases may be used to point to other locations.
182 #
183 DocumentRoot "C:/xampp/htdocs"
184 #
185 #
186 # Each directory to which Apache has access can be configured with respect
187 # to which services and features are allowed and/or disabled in that
188 # directory (and its subdirectories).
189 #
190 # First, we configure the "default" to be a very restrictive set of
```

Figure 3.10: Editing httpd.conf file

For convenience, below is a snapshot of ‘conf’ folder of Apache server. Here we see the httpd.conf file at left-bottom side.

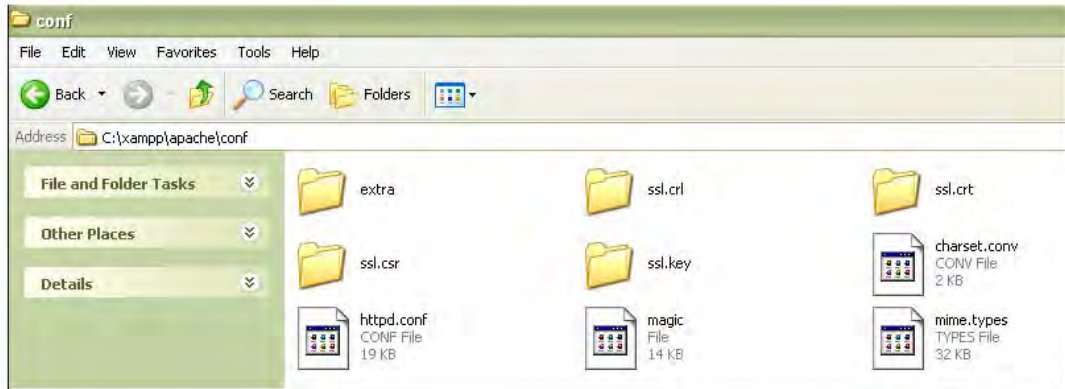


Figure 3.11: Location of httpd.conf file

3.8 Conclusion

In this chapter we have discussed on the hardware implementation part and scripting or coding part. One of the main objectives of this project is to keep the hardware part as cheap as possible. Taking input via pin 13 of parallel port was very exciting. Development of the logic circuit is very unique. Computer codes were very simple and readily available as free source codes. It was necessary to tune all codes and adapt for this project to get the desired output.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Main challenge of this project is to develop the web server and integrating with operator's packet switched network. Already there are hundreds of free codes available in 'C' language to control parallel port. We have to choose the right one and embed it with PHP code so that it can fit with the web server.

4.2 Results and Discussion

The final outcome of this project is very interesting. Typing `http://ms2device.com` in the of browser web server, we shall be able to view below:

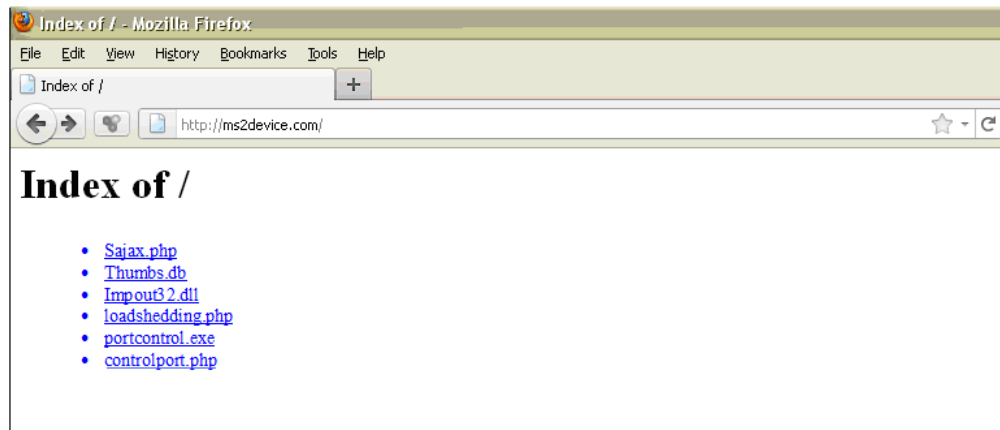


Figure 4.1: Contents of htdocs folder

These are the files we inserted into htdocs folder. Thumbs.db is thumbnail cache managed by Windows.

Now, if we click on Controlport.php link, we shall be able to see below:

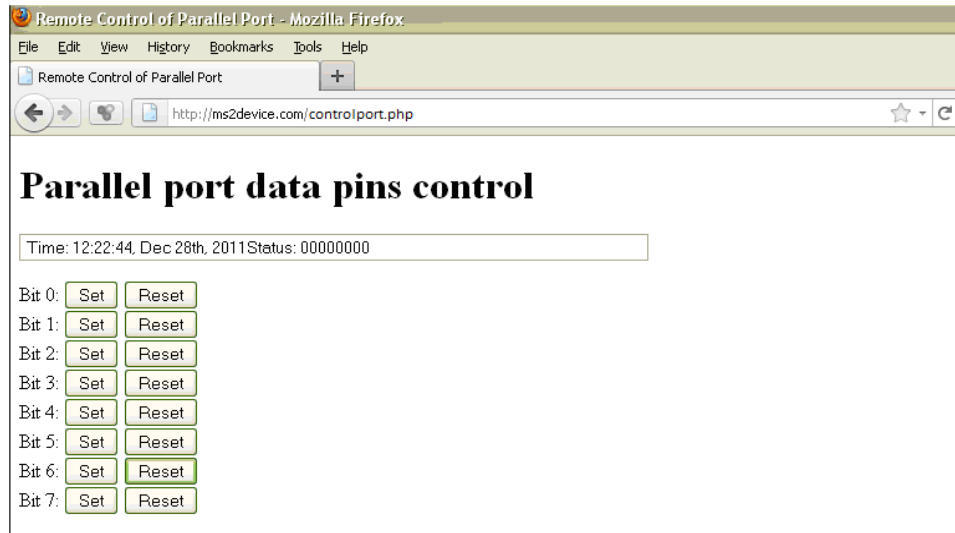


Figure 4.2: Output of Controlport.php

Clicking on any of the set/reset button will change the status of data pins and the output will be like below. Here, we have clicked on the set button of Bit 1: (Greenish shade).

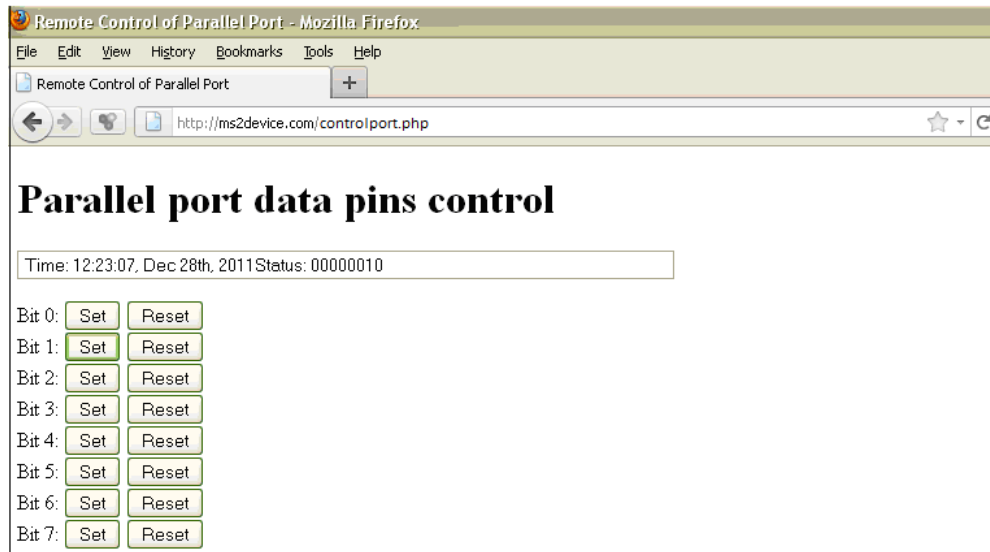


Figure 4.3: Bit level control of data pins

Consequently a change will be observed in the status bar just below “Parallel port data pins control” title. Previous digits after ‘status’: 00000000 will become 00000010. Again, data pin D1 will become high and will energize external driving circuit accordingly. Clicking on the same reset button will perform the opposite action. In this way we can control as many as eight parallel loads. As mentioned earlier in Chapter 2, this page will refresh itself after every few seconds which we can tune by ourselves.

In presence of electricity at remote site, the output of Loadshedding.php is like below:



(a)

In absence of electricity at remote site, the output of Loadshedding.php is like below:



(b)

Figure 4.4: (a, b) Load Shedding Status

The objective of this project has not yet been accomplished. Using java script enabled smart phone; one light and one fan were remotely controlled. For this, we need to ensure first peer to peer connectivity and type URL-http://ms2device.com in the web browser of the mobile phone.

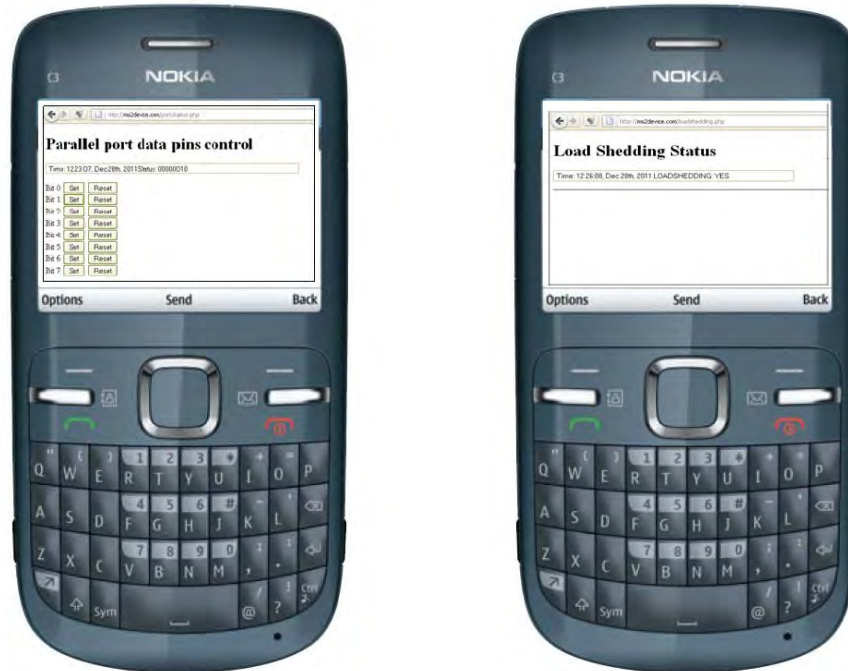


Figure 4.5: Controlport.php and loadshedding.php loaded at client

Load shedding status was also readily available on the screen of smart phone by clicking on loadshedding.php link. This page will also refresh itself after every few seconds by which we can update ourselves on load shedding state at remote site. However, page refresh time should be set to an optimum value for both cases. Otherwise, there will be unnecessary communication between client and server and data charges from mobile operator will increase. It will also drain out the battery quickly when handset will be used at client side.

When client and server are on the same cell, 32kbps bandwidth at air interface was required for smooth communication between client and server. If client and server are at different cells, 16kbps will be required per cell as seen from practical implementation.

4.3 Bandwidth Calculation Considering Practical Cases

It is very much important which technology is being used at air interface for smooth communication between client and server. Packet switched service is a shared service which means bandwidth at air interface will always be shared among the users under the same cell. If not dedicated from operator's side, available bandwidth at air interface can not be guaranteed for a particular user or subscriber. Detail discussion is given below considering different air interface technology i.e. GPRS, EDGE and 3G.

In GPRS, the most common coding scheme for packet data transfer is Coding Scheme-2. Other Coding Schemes (CS) are 1, 3 and 4. Theoretically, CS-2 enables a time slot to carry data at a rate of 13.2 kbps. This is the user rate over the air interface. However, the transmission of data involves number of layers above the air interface, with each layer adding certain amount of overhead. The result is that the rate of usable data is approximately 20 to 30 percent less than the air interface rate. Approximate usable data rate for CS-2 is 10.4 kbps. For CS-4, the rate will be around 16 kbps. CS-2 provides reasonably robust error correction over the air interface. In fact, CS-4 provides no error correction at all on the air interface. CS-3 and particularly CS-4 generate a great deal more retransmission over the air interface. With such retransmission, the net throughput may not be better than CS-2. Considering all these factors, if CS-2 is being used at air interface, we shall require four time slots for smooth communication between client and server when both are under the same cell. Otherwise, we shall require two time slots per cell between server and remote client.

Already discussed in chapter-2, there are nine coding schemes used in EDGE termed as Modulation and Coding Scheme (MCS) 1 to Modulation and Coding Scheme (MCS) 9. MCS-1 provides usable data rate 8.8 kbps and MCS-9 provides usable data rate 59.2 kbps per time slot. It should be noted that MCS-4 and MCS-9 offer no error protection for the user data.

Below table shows usable data rate per time slot from MCS-1 to MCS-9.

Coding and modulation scheme (MCS)	Air Interface Bit Rate (kbps/slot)
MCS-1	8.80
MCS-2	11.2
MCS-3	14.8
MCS-5	22.4
MCS-6	29.6
MCS-7	44.8
MCS-8	54.4
MCS-9	59.2

A packet sent with a higher coding scheme (less error correction) that is not properly received, can be retransmitted with a lower coding scheme (more error correction) if the new radio environment requires it. For a GSM channel, referred to as a TRX, there are a total of 8 time slots. As a result, it is possible to achieve 59.2×8 kbps = 473.6 kbps theoretical data rate per cell if all the time slots are configured as data i.e. EDGE channel. But in practical, the situation is totally different. As the same cell has to carry voice traffic, there will be less number of data i.e. EDGE channels. This will be true for GPRS also. Because data and voice traffic has different busy hours, normally a dedicated data channel and shared data slots are used. The dedicated data channel will carry only data and no voice traffic. The shared data channel can carry both voice and data, with voice having priority over data at all times . A typical approach used by operators is to have one or two dedicated time slots for GPRS/EDGE traffic and two or three additional shared channels for use with more data call. As a result, available data rate at air interface will depend upon these factors as well as number of users under a particular cell.

Third generation (3G) continues to receive much attention as the enabler for high-speed data for the wireless mobility market [20]. 3G is applied to mobile and stationary wireless applications involving high speed data rate. IMT-2000 mandates data rate of 144kbps at driving speed, 384 kbps for outside stationary use or at walking speed and 2 Mbps for indoors.

4.4 Conclusion

Looking at the different values at air interface, it seems 3G is better than EDGE and EDGE is better than GPRS to provide higher data rates. But number of subscribers under a particular cell is always the determinant factor for data rates rather the technology itself. If a cell is covered by 3G but congested in terms of users, it will perform worse compared to an EDGE/GPRS cell with few users. As 3G is an advanced technology, operators will often offer better service like dedicated bandwidth for a particular user under a specific cell. It will certainly help to execute real time operation and smooth communication between peers.

If point to point communication can't be arranged via cellular operator's data network, LAN/WAN connectivity can also be used to communicate with the web server. For short distance, LAN connectivity can easily be established. In that case, there will 100Mbps theoretical bandwidth between client and server.

If smart phone at client side is not available, a simple PC can also be used instead of mobile phone to control remote equipments at server side. In this case, client side will require an extra modem. Obviously, the SIM will now be used in the modem instead of mobile handset.

CHAPTER 5

FUTURE SCOPE OF THIS PROJECT

5.1 Conclusion

The inspiration of this project came from the demonstration of remote controlling and monitoring of power systems using SCADA based solutions. These are costly and complex solutions and generic from manufacturer's or developer's point of view. One of the aims of this project was to develop a simpler solution and to keep the architecture as simple as possible. Therefore, Apache based web server has been selected which is open and the easiest for understanding to all. But Apache has its known vulnerabilities which need special care and proper treatment to ensure industry grade security [21]. Software developers can put their effort and can build up commercially distributable web servers which will ensure proper safety of the entire system.

5.2 Future Scope

The web server has option to be accessed over Internet connectivity (instead of cellular network) if real IP is used. It needs to be connected via an ISP (Internet Service provider) to have this facility. In a country like us, where power crisis is enormous, individual can monitor and regulate their loads using this solution which will ultimately help to use scarce electricity more efficiently. Even small entrepreneurs and power generation companies can use this solution to control and monitor their switch gears. Alarm integration and fault handling will require additional coding and programming.

In Bangladesh, GSM operators are running 2G/2.5G GPRS/ EDGE network. It cannot ensure guaranteed bandwidth and expected QoS (Quality of Service). Due to high latency at air interface, real time control and monitoring is very difficult under 2G/2.5G network. CDMA network can also be used instead of GSM if it can ensure necessary bandwidth at air interface and nationwide network coverage. Next generation networks like 3G/LTE

will offer better opportunity for GSM operators by providing higher uplink and downlink bandwidth at air interface hence enhance the performance of peer to peer connectivity between client and server. Bandwidth hungry applications shall be able to take full advantage of the superior radio network.

Recent trend shows that at server side, special devices are already available having built-in Modem, USB, Serial and LAN ports etc.



Figure 5.1: Recent M2M Devices

These devices mainly run on customized operating systems (mostly based on Free BSD/ LINUX) and can be easily programmable. This will eliminate the need for PCs at server side.

In coming days, M2M can be used for below in our country:

1. Mobile Banking
2. Industrial automation
3. Vehicle tracking
4. Environmental monitoring
5. E-Governance etc.

APPENDIX B

DATA SHEET of OPTOCOUPLER 4N35

ABSOLUTE MAXIMUM RATINGS ⁽¹⁾				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
INPUT				
Reverse voltage		V_R	6	V
Forward current		I_F	60	mA
Surge current	$t \leq 10 \mu\text{s}$	I_{FSM}	2.5	A
Power dissipation		P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	$t \leq 1 \text{ ms}$	I_C	100	mA
Power dissipation		P_{diss}	150	mW
COUPLER				
Isolation test voltage		V_{ISO}	5300	V_{RMS}
Creepage			≥ 7	mm
Clearance			≥ 7	mm
Isolation thickness between emitter and detector			≥ 0.4	mm
Comparative tracking index	DIN IEC 112/VDE 0303, part 1		175	
Isolation resistance	$V_{IO} = 500 \text{ V}, T_{amb} = 25 \text{ }^\circ\text{C}$	R_{IO}	10^{12}	Ω
	$V_{IO} = 500 \text{ V}, T_{amb} = 100 \text{ }^\circ\text{C}$	R_{IO}	10^{11}	Ω
Storage temperature		T_{stg}	- 55 to + 150	$^\circ\text{C}$
Operating temperature		T_{amb}	- 55 to + 100	$^\circ\text{C}$
Junction temperature		T_J	100	$^\circ\text{C}$
Soldering temperature ⁽²⁾	max. 10 s dip soldering: distance to seating plane $\geq 1.5 \text{ mm}$	T_{sld}	260	$^\circ\text{C}$

CURRENT TRANSFER RATIO							
PARAMETER	TEST CONDITION	PART	SYMBOL	MIN.	TYP.	MAX.	UNIT
DC current transfer ratio ⁽¹⁾	$V_{CE} = 10 \text{ V}, I_F = 10 \text{ mA}$	4N35	CTR_{DC}	100			%
		4N36	CTR_{DC}	100			%
		4N37	CTR_{DC}	100			%
	$V_{CE} = 10 \text{ V}, I_F = 20 \text{ mA}$	4N38	CTR_{DC}	20			%
		$V_{CE} = 10 \text{ V}, I_F = 10 \text{ mA},$ $T_A = - 55 \text{ }^\circ\text{C to } + 100 \text{ }^\circ\text{C}$	4N35	CTR_{DC}	40	50	
	4N36		CTR_{DC}	40	50		%
	4N37		CTR_{DC}	40	50		%
	4N38		CTR_{DC}		30		%

Note

⁽¹⁾ Indicates JEDEC registered values.

SWITCHING CHARACTERISTICS							
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT	
Switching time ⁽¹⁾	$V_{CC} = 10 \text{ V}, I_C = 2 \text{ mA}, R_L = 100 \Omega$	t_{on}, t_{off}		10		μs	

Note

⁽¹⁾ Indicates JEDEC registered values.

ELECTRICAL CHARACTERISTICS (1)								
PARAMETER	TEST CONDITION	PART	SYMBOL	MIN.	TYP.	MAX.	UNIT	
OUTPUT								
Collector base breakdown voltage (2)	$I_C = 100 \mu A, I_B = 1 \mu A$	4N35	BV_{CBO}	70			V	
		4N36	BV_{CBO}	70			V	
		4N37	BV_{CBO}	70			V	
		4N38	BV_{CBO}	80			V	
Collector emitter leakage current (2)	$V_{CE} = 10 V, I_F = 0$	4N35	I_{CEO}		5	50	nA	
		4N36	I_{CEO}		5	50	nA	
	$V_{CE} = 10 V, I_F = 0$	4N37	I_{CEO}		5	50	nA	
		4N38	I_{CEO}			50	nA	
	$V_{CE} = 30 V, I_F = 0, T_{amb} = 100 ^\circ C$	4N35	I_{CEO}				500	μA
		4N36	I_{CEO}				500	μA
		4N37	I_{CEO}				500	μA
$V_{CE} = 60 V, I_F = 0, T_{amb} = 100 ^\circ C$	4N38	I_{CEO}			6		μA	
Collector emitter capacitance	$V_{CE} = 0$		C_{CE}		6		pF	
COUPLER								
Resistance, input output (2)	$V_{IO} = 500 V$		R_{IO}	10^{11}			Ω	
Capacitance, input output	$f = 1 MHz$		C_{IO}		0.5		pF	

Notes

(1) $T_{amb} = 25 ^\circ C$, unless otherwise specified.

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operational sections of this document. Exposure to absolute maximum ratings for extended periods of the time can adversely affect reliability.

(2) Refer to reflow profile for soldering conditions for surface mounted devices (SMD). Refer to wave profile for soldering conditions for through hole devices (DIP).

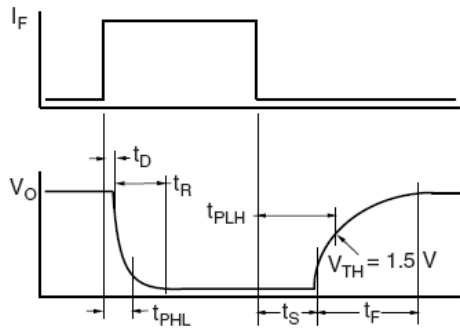


Fig. Switching Timing

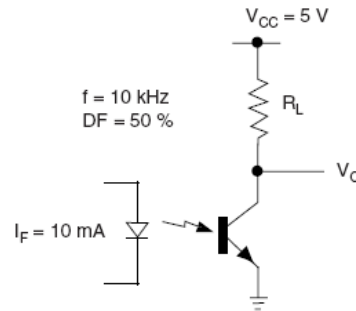


Fig. Switching Schematic

APPENDIX C

DATA SHEET of DIODE 1N4007

MAXIMUM RATINGS

Rating	Symbol	1N4001	1N4002	1N4003	1N4004	1N4005	1N4006	1N4007	Unit
*Peak Repetitive Reverse Voltage Working Peak Reverse Voltage DC Blocking Voltage	V_{RRM} V_{RWM} V_R	50	100	200	400	600	800	1000	Volts
*Non-Repetitive Peak Reverse Voltage (halfwave, single phase, 60 Hz)	V_{RSM}	60	120	240	480	720	1000	1200	Volts
*RMS Reverse Voltage	$V_{R(RMS)}$	35	70	140	280	420	560	700	Volts
*Average Rectified Forward Current (single phase, resistive load, 60 Hz, see Figure 8, $T_A = 75^\circ\text{C}$)	I_O	1.0							Amp
*Non-Repetitive Peak Surge Current (surge applied at rated load conditions, see Figure 2)	I_{FSM}	30 (for 1 cycle)							Amp
Operating and Storage Junction Temperature Range	T_J T_{stg}	- 65 to +175							$^\circ\text{C}$

ELECTRICAL CHARACTERISTICS*

Rating	Symbol	Typ	Max	Unit
Maximum Instantaneous Forward Voltage Drop ($i_F = 1.0$ Amp, $T_J = 25^\circ\text{C}$) Figure 1	v_F	0.93	1.1	Volts
Maximum Full-Cycle Average Forward Voltage Drop ($I_O = 1.0$ Amp, $T_L = 75^\circ\text{C}$, 1 inch leads)	$V_{F(AV)}$	—	0.8	Volts
Maximum Reverse Current (rated dc voltage) ($T_J = 25^\circ\text{C}$) ($T_J = 100^\circ\text{C}$)	I_R	0.05 1.0	10 50	μA
Maximum Full-Cycle Average Reverse Current ($I_O = 1.0$ Amp, $T_L = 75^\circ\text{C}$, 1 inch leads)	$I_{R(AV)}$	—	30	μA

*Indicates JEDEC Registered Data

APPENDIX D

CONTROLPORT.PHP CODE

```
<?php

    require("Sajax.php");

    function portstatus() {
        date_default_timezone_set('Asia/Dhaka'); //Built in fuction, Have changed
        // $loadsheddingstatus = shell_exec("portcontrol.exe LPT1STATUS read print dec") ==
120 ? "NO" : "YES"; //Execute command via shell and return the complete output as a string

        return " Time: " . date("H:i:s, M dS, Y") . "Status: " .
            shell_exec("portcontrol.exe LPT1DATA read print bin") ;
    }

    function portcontrol($x, $y) {
        $loadsheddingstatus = shell_exec("portcontrol.exe LPT1STATUS read print dec") ==
120 ? "NO" : "YES";
        //if ($loadsheddingstatus == 'YES'){return portstatus();}
        if (($x >= 0) && ($x < 8)) {
            if ($y == 1)
                shell_exec("portcontrol.exe LPT1DATA read setbit " . $x . " write");
            else
                shell_exec("portcontrol.exe LPT1DATA read resetbit " . $x . " write");
        }
        return portstatus();
    }

    sajax_init();
    // $sajax_debug_mode = 1; Asynchronous JavaScript and XML//JavaScript is executed
on the client end and PHP code is executed on the server end
    sajax_export("portstatus");
    sajax_export("portcontrol");
    sajax_handle_client_request();

?>
<html>
<head>
    <title>Remote Control of Parallel Port</title>
    <script>
```

```

<?php
sajax_show_javascript();

?>

function do_portstatus_cb(z) {
    // update status field in form
    document.getElementById("status").value = z;
}

function do_portstatus() {
    x_portstatus(do_portstatus_cb);
    setTimeout('do_portstatus();',30000); // executes the next data query in every n
milliseconds
}

function do_portcontrol_cb(z) {
    // update status field in form
    document.getElementById("status").value = z;
}

function do_portcontrol(bit,value) {
    x_portcontrol(bit,value,do_portcontrol_cb);
}

</script>

</head>
<body>

<SCRIPT LANGUAGE="JavaScript">
<!--
do_portstatus();
// -->
</SCRIPT>

<P>

<H1>Parallel port data pins control </H1>
<P>

<input type="text" name="status" id="status" value="No status yet" size="80">

<P>

```

Bit 0:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(0,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(0,0); return false;">
```


Bit 1:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(1,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(1,0); return false;">
```


Bit 2:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(2,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(2,0); return false;">
```


Bit 3:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(3,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(3,0); return false;">
```


Bit 4:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(4,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(4,0); return false;">
```


Bit 5:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(5,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(5,0); return false;">
```


Bit 6:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(6,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(6,0); return false;">
```


Bit 7:

```
<input type="button" name="check" value="Set"
      onclick="do_portcontrol(7,1); return false;">
```

```
<input type="button" name="check" value="Reset"
      onclick="do_portcontrol(7,0); return false;">
```

```
<BR>
```

```
<P>
```

```
<HR>
```

```
<P>
```

```
</body>
```

```
</html>
```

APPENDIX E

SAJAX.PHP CODE

```
<?php
if (!isset($SAJAX_INCLUDED)) {

    /*
     * GLOBALS AND DEFAULTS
     *
     */
    $GLOBALS['sajax_version'] = '0.12';
    $GLOBALS['sajax_debug_mode'] = 0;
    $GLOBALS['sajax_export_list'] = array();
    $GLOBALS['sajax_request_type'] = 'GET';
    $GLOBALS['sajax_remote_uri'] = "";
    $GLOBALS['sajax_failure_redirect'] = "";

    /*
     * CODE
     *
     */

    //
    // Initialize the Sajax library.
    //
    function sajax_init() {
    }

    //
    // Helper function to return the script's own URI.
    //
    function sajax_get_my_uri() {
        return $_SERVER["REQUEST_URI"];
    }
    $sajax_remote_uri = sajax_get_my_uri();

    //
    // Helper function to return an eval()-usable representation
    // of an object in JavaScript.
    //
    function sajax_get_js_repr($value) {
        $type = gettype($value);
```

```

if ($type == "boolean") {
    return ($value) ? "Boolean(true)" : "Boolean(false)";
}
elseif ($type == "integer") {
    return "parseInt($value)";
}
elseif ($type == "double") {
    return "parseFloat($value)";
}
elseif ($type == "array" || $type == "object" ) {
    //
    // XXX Arrays with non-numeric indices are not
    // permitted according to ECMAScript, yet everyone
    // uses them.. We'll use an object.
    //
    $s = "{ ";
    if ($type == "object") {
        $value = get_object_vars($value);
    }
    foreach ($value as $k=>$v) {
        $esc_key = sajax_esc($k);
        if (is_numeric($k))
            $s .= "$k: " . sajax_get_js_repr($v) . ", ";
        else
            $s .= "\"$esc_key\": " . sajax_get_js_repr($v) . ", ";
    }
    if (count($value))
        $s = substr($s, 0, -2);
    return $s . " }";
}
else {
    $esc_val = sajax_esc($value);
    $s = "$esc_val";
    return $s;
}
}

```

```

function sajax_handle_client_request() {
    global $sajax_export_list;

    $mode = "";

    if (! empty($_GET["rs"]))
        $mode = "get";

    if (!empty($_POST["rs"]))

```

```

        $mode = "post";

if (empty($mode))
    return;

$target = "";

if ($mode == "get") {
    // Bust cache in the head
    header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // Date in the past
    header ("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
    // always modified
    header ("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
    header ("Pragma: no-cache"); // HTTP/1.0
    $func_name = $_GET["rs"];
    if (! empty($_GET["rsargs"]))
        $args = $_GET["rsargs"];
    else
        $args = array();
}
else {
    $func_name = $_POST["rs"];
    if (! empty($_POST["rsargs"]))
        $args = $_POST["rsargs"];
    else
        $args = array();
}

if (! in_array($func_name, $sajax_export_list))
    echo "-:$func_name not callable";
else {
    echo "+:";
    $result = call_user_func_array($func_name, $args);
    echo "var res = " . trim(sajax_get_js_repr($result)) . "; res;";
}
exit;
}

function sajax_get_common_js() {
    global $sajax_debug_mode;
    global $sajax_request_type;
    global $sajax_remote_uri;
    global $sajax_failure_redirect;

    $t = strtoupper($sajax_request_type);
    if ($t != "" && $t != "GET" && $t != "POST")

```



```

        return "// Invalid type: $t.. \n\n";

    ob_start();
    ?>

    // remote scripting library
    // (c) copyright 2005 modernmethod, inc
    var sajax_debug_mode = <?php echo $sajax_debug_mode ? "true" : "false"; ?>;
    var sajax_request_type = "<?php echo $t; ?>";
    var sajax_target_id = "";
    var sajax_failure_redirect = "<?php echo $sajax_failure_redirect; ?>";

    function sajax_debug(text) {
        if (sajax_debug_mode)
            alert(text);
    }

    function sajax_init_object() {
        sajax_debug("sajax_init_object() called..")

        var A;

        var msxmlhttp = new Array(
            'Msxml2.XMLHTTP.5.0',
            'Msxml2.XMLHTTP.4.0',
            'Msxml2.XMLHTTP.3.0',
            'Msxml2.XMLHTTP',
            'Microsoft.XMLHTTP');
        for (var i = 0; i < msxmlhttp.length; i++) {
            try {
                A = new ActiveXObject(msxmlhttp[i]);
            } catch (e) {
                A = null;
            }
        }

        if(!A && typeof XMLHttpRequest != "undefined")
            A = new XMLHttpRequest();
        if (!A)
            sajax_debug("Could not create connection object.");
        return A;
    }

    var sajax_requests = new Array();

    function sajax_cancel() {

```

```

        for (var i = 0; i < sajax_requests.length; i++)
            sajax_requests[i].abort();
    }

function sajax_do_call(func_name, args) {
    var i, x, n;
    var uri;
    var post_data;
    var target_id;

    sajax_debug("in sajax_do_call(.." + sajax_request_type + "/" +
sajax_target_id);
    target_id = sajax_target_id;
    if (typeof(sajax_request_type) == "undefined" || sajax_request_type == "")
        sajax_request_type = "GET";

    uri = "<?php echo $sajax_remote_uri; ?>";
    if (sajax_request_type == "GET") {

        if (uri.indexOf("?") == -1)
            uri += "?rs=" + escape(func_name);
        else
            uri += "&rs=" + escape(func_name);
        uri += "&rst=" + escape(sajax_target_id);
        uri += "&rsrnd=" + new Date().getTime();

        for (i = 0; i < args.length-1; i++)
            uri += "&rsargs[]=" + escape(args[i]);

        post_data = null;
    }
    else if (sajax_request_type == "POST") {
        post_data = "rs=" + escape(func_name);
        post_data += "&rst=" + escape(sajax_target_id);
        post_data += "&rsrnd=" + new Date().getTime();

        for (i = 0; i < args.length-1; i++)
            post_data = post_data + "&rsargs[]=" + escape(args[i]);
    }
    else {
        alert("Illegal request type: " + sajax_request_type);
    }

    x = sajax_init_object();
    if (x == null) {
        if (sajax_failure_redirect != "") {

```

```

        location.href = sajax_failure_redirect;
        return false;
    } else {
        sajax_debug("NULL sajax object for user agent:\n" +
navigator.userAgent);
        return false;
    }
} else {
    x.open(sajax_request_type, uri, true);
    // window.open(uri);

    sajax_requests[sajax_requests.length] = x;

    if (sajax_request_type == "POST") {
        x.setRequestHeader("Method", "POST " + uri + "
HTTP/1.1");
        x.setRequestHeader("Content-Type", "application/x-www-
form-urlencoded");
    }

    x.onreadystatechange = function() {
        if (x.readyState != 4)
            return;

        sajax_debug("received " + x.responseText);

        var status;
        var data;
        var txt = x.responseText.replace(/^\s*\s*$/g, "");
        status = txt.charAt(0);
        data = txt.substring(2);

        if (status == "") {
            // let's just assume this is a pre-response bailout and
let it slide for now
        } else if (status == "-")
            alert("Error: " + data);
        else {
            if (target_id != "")

                document.getElementById(target_id).innerHTML = eval(data);
            else {
                try {
                    var callback;
                    var extra_data = false;

```

```

"object") {
    1].callback;
    1].extra_data;

    1];

    Could not eval " + data );

    if (typeof args[args.length-1] ==
        callback = args[args.length-
            extra_data = args[args.length-
        } else {
            callback = args[args.length-
        }
        callback(eval(data), extra_data);
    } catch (e) {
        sajax_debug("Caught error " + e + ":

    }

    }

    }

    sajax_debug(func_name + " uri = " + uri + "/post = " + post_data);
    x.send(post_data);
    sajax_debug(func_name + " waiting..");
    delete x;
    return true;
}

<?php
$html = ob_get_contents();
ob_end_clean();
return $html;
}

function sajax_show_common_js() {
    echo sajax_get_common_js();
}

// javascript escape a value
function sajax_esc($val)
{
    $val = str_replace("\\", "\\\\", $val);
    $val = str_replace("\r", "\\r", $val);
    $val = str_replace("\n", "\\n", $val);
    $val = str_replace("'", "\'", $val);
    return str_replace('"', "\\\"", $val);
}

```

```

function sajax_get_one_stub($func_name) {
    ob_start();
    ?>

    // wrapper for <?php echo $func_name; ?>

    function x_<?php echo $func_name; ?>() {
        sajax_do_call("<?php echo $func_name; ?>",
            x_<?php echo $func_name; ?>.arguments);
    }

    <?php
    $html = ob_get_contents();
    ob_end_clean();
    return $html;
}

function sajax_show_one_stub($func_name) {
    echo sajax_get_one_stub($func_name);
}

function sajax_export() {
    global $sajax_export_list;

    $n = func_num_args();
    for ($i = 0; $i < $n; $i++) {
        $sajax_export_list[] = func_get_arg($i);
    }
}

$sajax_js_has_been_shown = 0;
function sajax_get_javascript()
{
    global $sajax_js_has_been_shown;
    global $sajax_export_list;

    $html = "";
    if (! $sajax_js_has_been_shown) {
        $html .= sajax_get_common_js();
        $sajax_js_has_been_shown = 1;
    }
    foreach ($sajax_export_list as $func) {
        $html .= sajax_get_one_stub($func);
    }
    return $html;
}

```

```
}  
  
function sajax_show_javascript()  
{  
    echo sajax_get_javascript();  
}  
  
$SAJAX_INCLUDED = 1;  
}  
?>
```

APPENDIX F

LOADSHEDDING.PHP CODE

```
<?php
    require("Sajax.php");

    function portstatus() {
        date_default_timezone_set('Asia/Dhaka'); //Built in fuction,Have changed
        $loadsheddingstatus = shell_exec("portcontrol.exe LPT1STATUS read print dec") ==
120 ? "NO" : "YES"; //Execute command via shell and return the complete output as a string

        return " Time: " . date("H:i:s, M dS, Y") .
            " LOADSHEDDING: $loadsheddingstatus";
    }

    function portcontrol($x, $y) {
        $loadsheddingstatus = shell_exec("portcontrol.exe LPT1STATUS read print dec") ==
120 ? "NO" : "YES";
        //if ($loadsheddingstatus == 'YES'){return portstatus();}
        if (($x >= 0) && ($x < 8)) {
            if ($y == 1)
                shell_exec("portcontrol.exe LPT1DATA read setbit " . $x . " write");
            else
                shell_exec("portcontrol.exe LPT1DATA read resetbit " . $x . " write");
        }
        return portstatus();
    }

    sajax_init();
    // $sajax_debug_mode = 1; Asynchronous JavaScript and XML//JavaScript is executed
on the client end and PHP code is executed on the server end
    sajax_export("portstatus");
    sajax_export("portcontrol");
    sajax_handle_client_request();
?>
```

```

<html>
<head>
  <title>Remote Control of Parallel Port</title>
  <script>
  <?php
  sajax_show_javascript();
  ?>

  function do_portstatus_cb(z) {
    // update status field in form
    document.getElementById("status").value = z;
  }

  function do_portstatus() {
    x_portstatus(do_portstatus_cb);
    setTimeout('do_portstatus();',30000); // executes the next data query in every n
milliseconds
  }

  function do_portcontrol_cb(z) {
    // update status field in form
    document.getElementById("status").value = z;
  }

  function do_portcontrol(bit,value) {
    x_portcontrol(bit,value,do_portcontrol_cb);
  }

  </script>

</head>
<body>

<SCRIPT LANGUAGE="JavaScript">
<!--
do_portstatus();
// -->
</SCRIPT>

<P>

<H1>Load Shedding Status </H1>
<P>

<input type="text" name="status" id="status" value="No status yet" size="70">

```



```
<P>
```

```
<P>  
<HR>
```

```
<P>  
</body>
```

```
</html>
```

REFERENCES

- [1] M. Martsola, T. Kiravuo, J.K.O. Lindqvist, "Machine to machine communication in cellular networks", *Conference (MTAS)*, pp. 6, April 2005
- [2] G. Lawton, "Machine-to-machine technology gears up for growth", *IEEE Journal of Computer*, vol. 37, pp. 12-15, Sept. 2004
- [3] M. Sallinen, "Applications of wireless M2M communication", *Conference (ICTC) 2010*, pp. 384 - 385, Nov 2010
- [4] J.P. Conti, "The Internet of things", *IEEE Journals of Communication Engineer*, vol.-4, pp. 20-25, Jan 2006
- [5] W. Peng, Y. Lixiang, W. Jun, Z. Yuancheng, "The remote intelligent controlling and monitoring system of home appliances based on GPRS/GSM", *Conference (ICITIS)*, pp. 958 - 961, Dec 2010
- [6] http://standards.ieee.org/develop/wg/1609_WG.html
- [7] M. Ahmad, Izharuddin, "Security enhancements in GSM cellular standard", *Conference (WCSN) 2008*, pp. 116 - 117, Dec. 2008
- [8] A.B. Rekha, Y. Solanke, S.R. Kolli, B. Urnadevi, "End-to-end security for GSM users" *IEEE Conference (ICPWC)*, pp. 434 - 437, Jan. 2005
- [9] N. Boudriga, "Security of Mobile Communications", *Conference (ICSPC) 2007*, pp. 16 - 17, Nov. 2007
- [10] R.T. Fielding, G.Kaiser, "The Apache HTTP Server Project", *IEEE Internet Computing*, vol.1, no.4, Aug 1997
- [11] PC Hardware in a Nutshell, 2nd Edition by Barbara & Robert, O'Reilly
- [12] 3GPP TS 23.060 version 3.6.0 (2001-01), 2000, "3rd Generation Partnership Project; Technical Specification Group Services and Systems Aspects; General Packet Radio Service (GPRS); Service Description; Stage 2
- [13] 3GPP TS 43.051, "3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Overall description, Stage 2 (Release 5)," April 2002
- [14] 3GPP TS 36.300: E-UTRA and E-UTRAN Overall Description; Stage 2

- [15] 3GPP TR 25.913: "Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN)"
- [16] H.Y. Suk, Y.H. Kai, "Challenges in the Migration to 4G Mobile Systems", *IEEE Communications Magazine*, vol. 41, no. 54 – 59, Dec. 2003
- [17] C. Jain, D.J. Goodman, "General packet radio service in GSM", *IEEE Communications Magazine*, vol. 35, pp. 122 - 131, Oct 1997
- [18] 3GPP TS 23.003 V7.3.0 (2007-03), Technical Specification Group Core Network and Terminals; Numbering, Addressing and Identification
- [19] H. Olofsson, A. Furuskar, "Aspects of introducing EDGE in existing GSM Networks", *Conference (ICSPC) 2007*, pp. 421 - 426, vol. 1, Oct 1998
- [20] S. Martin, *Beyond 3G – Bringing Networks, Terminals and the Web Together*, John Wiley and Sons, Ltd, Publication
- [21] W.W.Sung, O.H. Alhazmi, Y.K. Malaiya, "Assessing Vulnerabilities in Apache and IIS HTTP Servers", pp. 103 - 110, Sep 2006