Department of Electrical Engineering, IIT Kanpur

B.Tech Project Report

Networked Service Request System (NSRS)

Submitted by	:	Biplab Deka (Y5147)
		Piyush Keshri (Y5303)
Supervisor	:	Prof. S. Qureshi
Date of Submission	:	13th April, 2009

Certificate

This is to certify that the B.Tech Project Report titled

"Networked Service Request System (NSRS)"

By

Piyush Keshri *&* **Biplab Deka**

is approved by me for submission. It is also Certified that the report represents the original work has been carried out by the students in VLSI Lab under my guidance and support during, July'08 – April'09 and original material has been presented in the report and has not been submitted elsewhere for any other degree or diploma.

Date:

Prof. S. Qureshi
Department of Electrical Engineering
IIT Kanpur

Acknowledgements

We are extremely thankful to Prof. S. Qureshi, Department of Electrical Engineering, IIT Kanpur, for giving us the opportunity to work on this project and for his constant guidance and motivation.

We would also like to thank our batch mates Praneeth Bodduluri for helping us configure the network gateway kit and programming with Python; Kshijit Deo for helping us with the embedded design part and Rahul Gupta for his help in learning PHP script.

We would also like to thank the employees of the company “Simmortel Voice Technologies Pvt. Ltd”, located at SIDBI Innovation and Incubation Centre, IIT Kanpur for their support in developing the Interactive Voice Response System for our project.

Regards,

Biplab Deka
&
Piyush Keshri

List of Figures

Acronyms

NSRS	–	Network Service Request System
RSM	–	Remote Service Machine
RFID	–	Radio Frequency Identification
PC	–	Personal computer
LAN	–	Local Area Network
GSM	–	Global System for Mobile Communication
SMS	–	Short Message Service
UART	–	Universal Asynchronous Receiver/ Transmitter
USB	–	Universal Serial Bus
SPI	–	Serial Peripheral Interface
PCB	–	Printed Circuit Board
SIM	–	Subscriber Identity Module
PBX	–	Private Branch Exchange
PSTN	–	Public Switched Telephone Network
IVR	–	Interactive Voice Response
AGI	–	Asterisk Gateway Interface
DTMF	–	Dual Tone Multi Frequency
DBMS	–	Data Base Management System
LCD	–	Liquid Crystal Display

List of Tables

Table of Contents

1. Introduction

1.1 Motivation

Currently, an individual in the society has to request for various services either on paper or by making a trip to the service provider, which inherently adds sluggishness and unaccountability to the system. This led to the wastage of consumers' time and may lead to unnecessary delays. Moreover, there do not exist any proper centralised system in most of the common usage services via which the user can be informed, whether his request has been serviced or not. An example of such services is the domestic gas booking system.

The motivation of the project is to develop a system that can be used to automate various processes involved in addressing the service requests of different users, making it more accountable and easily accessible. It should enable the end users to request for services easily and without wasting much of his time. It should also enable the service provider to view all such requests and to notify the users when their requests are met. Additionally, the system should be usable even in remote areas apart from urban and rural areas making it easy accessible and reliable. The system should also be able to serve illiterate masses across the country. The system should be affordable & flexible enough so that it can be easily scaled and deployed at various places.

1.2 Related Work

The project on "Remote Service Machine" by Mayank Raj and Gopesh Mittal completed under the supervision of Prof. S. Qureshi was an attempt towards developing such a system. It used RFID technology to enable the users to send requests to the service provider. A schematic of the system is shown Fig. 1.

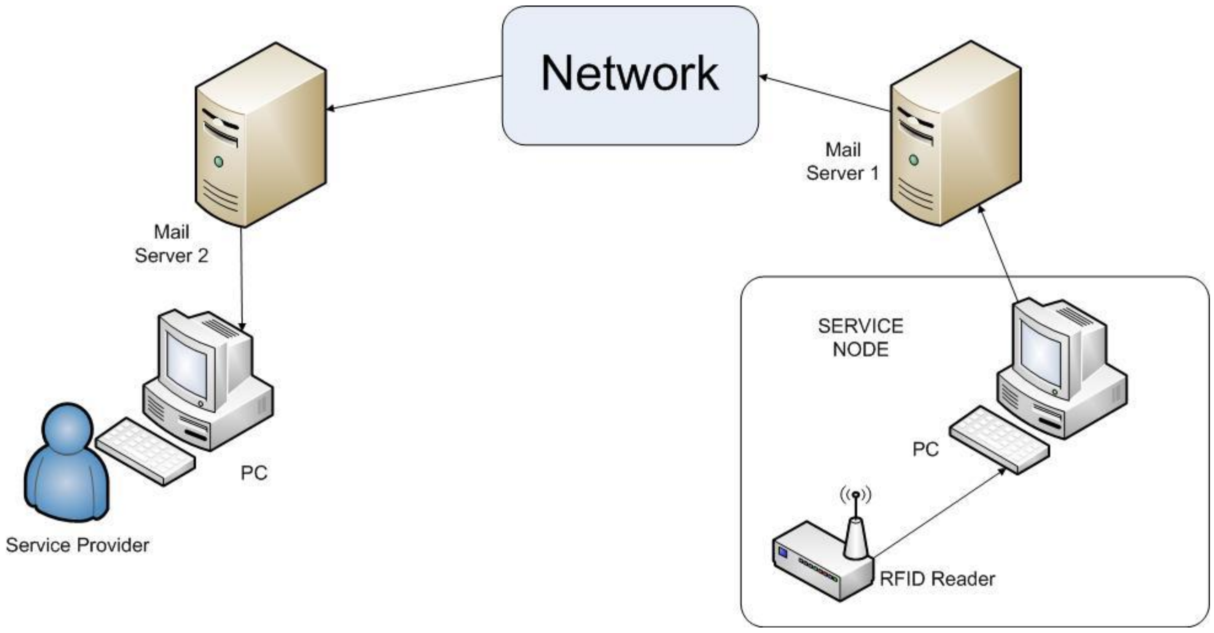
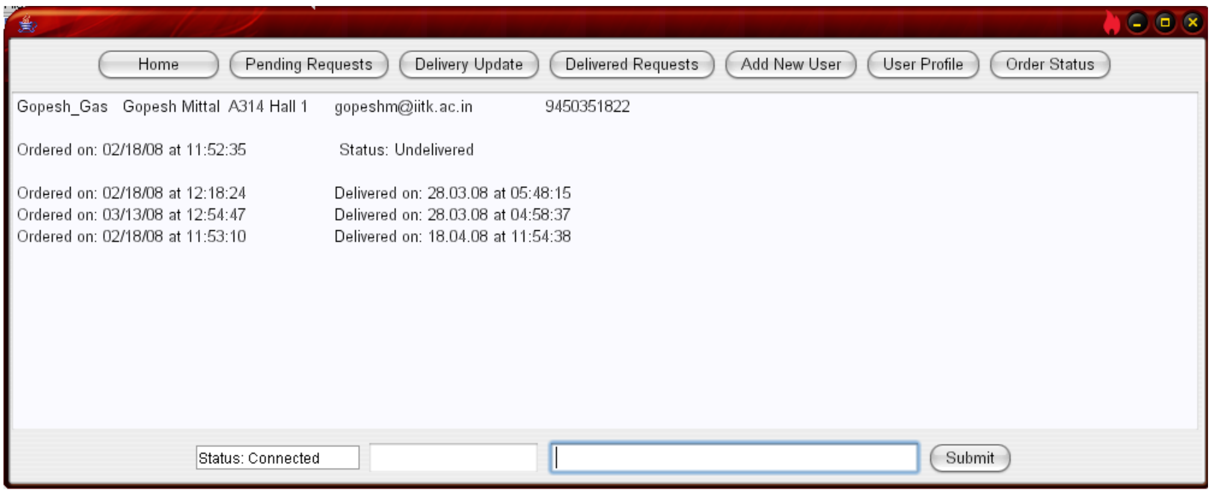


Figure1. A schematic of the Remote Services Machine (RSM)

The system uses an RFID reader to interact with the user. It reads the information on the RFID tags of the users. The reader is connected to a PC that receives the information related to these requests and connects to a mail server and sends out emails to the service provider. The service provider runs software developed in Java on his personal Computer. This software downloads the emails, extracts the necessary information from them, stores them in a few text files and displays this information to the service provider. On completion of the requests, the service provider updates this information in the software and it sends out emails to the users to inform them that their requests have been met. A screenshot of the software is shown in Fig. 2.



```
Gopesh_Gas   Gopesh Mittal  A314 Hall 1     gopeshm@iitk.ac.in          9450351822

Ordered on: 02/18/08 at 11:52:35             Status: Undelivered

Ordered on: 02/18/08 at 12:18:24            Delivered on: 28.03.08 at 05:48:15
Ordered on: 03/13/08 at 12:54:47            Delivered on: 28.03.08 at 04:58:37
Ordered on: 02/18/08 at 11:53:10            Delivered on: 18.04.08 at 11:54:38

Status: Connected
```

Figure2. Screenshot of the software used by the service provider in the RSM system

1.3 Shortcomings of the RSM System

Although the RSM system fulfilled its basic purpose, there are some short comings associated with it such as:

- **Underutilized PC at the Service Node:**
  The only function of the PC at the service node is to receive messages from the reader and to connect to the mail server. This doesn't justify the use of an expensive PC. This could have been achieved by using a device with much lesser processing capabilities and peripherals.

- **Unreliability of emails:**
  The system depends on emails for sending information. The delivery of emails on time is not guaranteed. Also, mail servers are not in our control and can go down anytime. Hence, the reliability of the system suffers.

- **Dependence on custom software at the Service Provider's end:**
  The service provider depends on the custom software that is provided. This will create a problem in case his PC crashes and he has to use another PC which doesn't have this software installed.
- **Usage of text file for data storage :**
  Text files are not considered to be a good way of storing data because it is very cumbersome to search for relevant data when a text file grows huge. Also, there is no data security.
- **User cannot view the data :**

1.4 Introduction to NSRS

NSRS is an attempt to develop a system free from the above disadvantages that can be used by various sections of the society to request for different services like delivery services or emergency services. It uses RFID for identification, LAN and GSM for communication.

All users are issued a RFID tag that they can swipe at a service node to request a particular service. This method is ideal for the illiterate section of the society as they do not need to provide any additional information other than that already contained in the tag. Additionally, NSRS also has the capability to receive requests via SMS, phone calls and the internet. This is ideal for the affluent classes of the society as it would save them the hassle of physically travelling to the service node to request the service.

The rest of this report is organized as follows. In Section 2, the architecture of NSRS is described. Detailed description of various components of NSRS is presented in Section 3. Section 4 presents an example implementation of NSRS for a Gas Booking Service. Section 5 highlights the salient features of NSRS. We present our conclusions in section 6 and suggestions for future work are presented in section 7.

2. Architecture of the Networked Service Request System (NSRS)

The architecture of our Networked Service Request System (NSRS) is shown in Fig. 3.

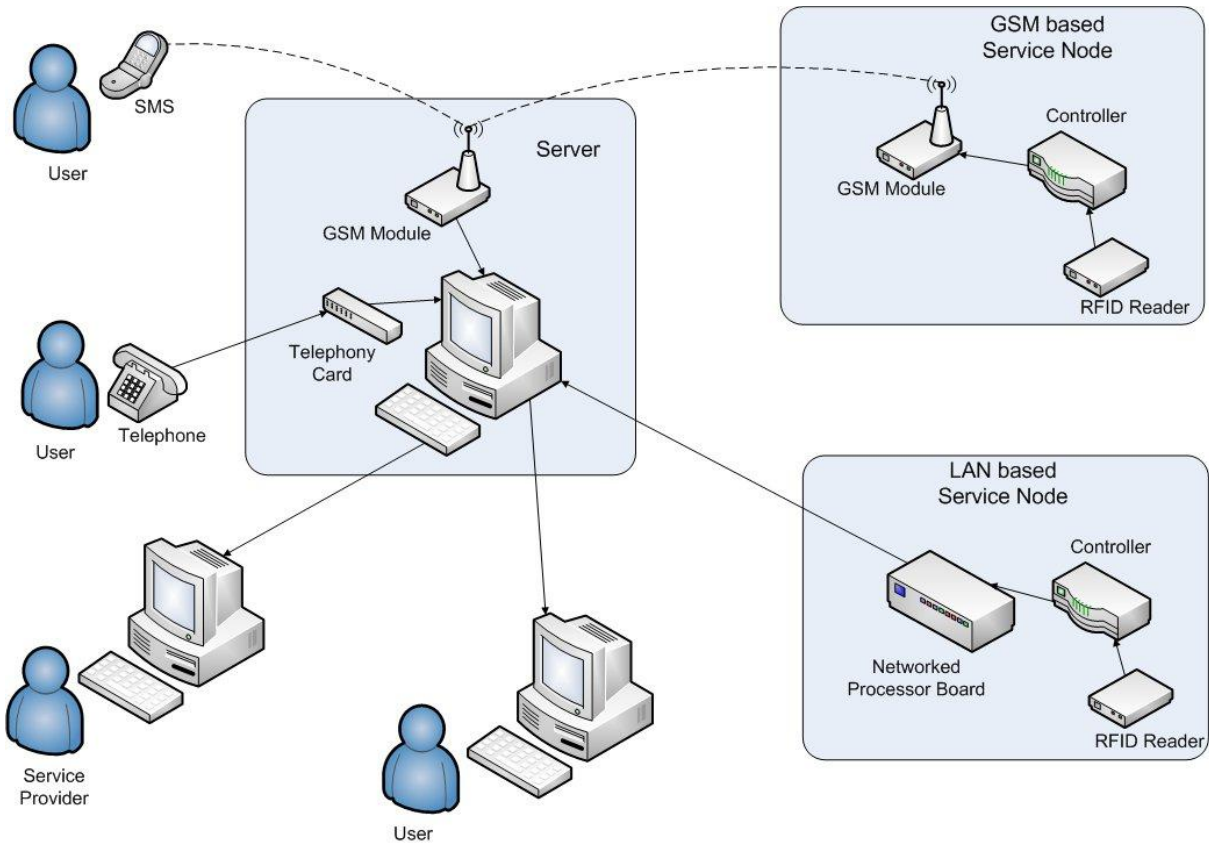


Figure3. Schematic of the Networked Service Request System (NSRS)

The users request for a service by visiting the service nodes. Our system has two types of service nodes:

- LAN based Service Node
- GSM based Service Node

Both types of service nodes use a RFID Reader connected to a controller. When a user carrying a compatible tag comes in range of the reader, the reader signals the controller. The controller then asks the reader to read the desired data from the card. This data may vary depending on the application.

The LAN based Service Node is used when LAN is available at the place of deployment. The controller in this case sends the information to a Networked Processor Board that sends it to the server using the LAN. When there is no LAN available, the GSM based Service node is used. The information in this case is sent using a GSM module in the form of Short Message Service (SMS). The SMS is received by another GSM module connected to the server and is passed on to the server.

The server processes this information and stores it in a database. The server also hosts a website that enables the service provider to view these requests from any computer that is connected to the internet.

Our service nodes are easy to use and can be used even by illiterate masses. Users don’t need to enter any information manually as all the required information is stored on their tag.

Additionally, for literate users our system offers three other methods of requesting services. Firstly, they can use the website hosted on the server to book their requests. They can use this from any computer connected to the internet. They can also use this to check the status of the requests he made earlier. Secondly, they can register requests by sending SMS to the GSM module connected to the server. Lastly, they can make a request by making a telephone call. The call is handled by the server using an Interactive Voice Response System.

The last three methods have the added advantage that the user doesn’t need to physically make a trip to the service node.

3. Components of NSRS

3.1 RFID System

RFID stands for Radio Frequency Identification. This technology can be used to identify, track, sort or detect a wide variety of objects based on communication takes place between a reader (interrogator) and a transponder (chip connected to an antenna) often called a tag. Tags can either be active (powered by battery) or passive (powered by the reader field), and come in various forms including cards, tags and labels.

Each tag has a certain amount of memory (EEPROM) in which it can store information about the object, such as its unique ID (serial) number, or in some cases more details including batch number, manufacture date and product composition. When these tags pass through a field generated by a reader of a RFID system, they transmit this information back to the reader, thereby identifying the object.

RFID systems are designed to operate over a variety of frequency ranges.

- Low-frequency (LF): 125 to 134.2 kHz and 140 to 148.5 kHz have shorter read range (< 0.5m or 1.5 ft) and slower read speed.
- High-frequency (HF): 13.56MHz has greater read range and higher read speed than LF systems.
- Ultra-high-frequency (UHF): 860 to 960MHz is up to 3m (9.5 ft) and the data transfer rate is faster than HF systems.
- Microwave band: 2.45GHz, 5.8GHz and above. This offers the highest data read rates but are the most expensive systems and have a limited read range of up to 1 m (3 ft) (8).

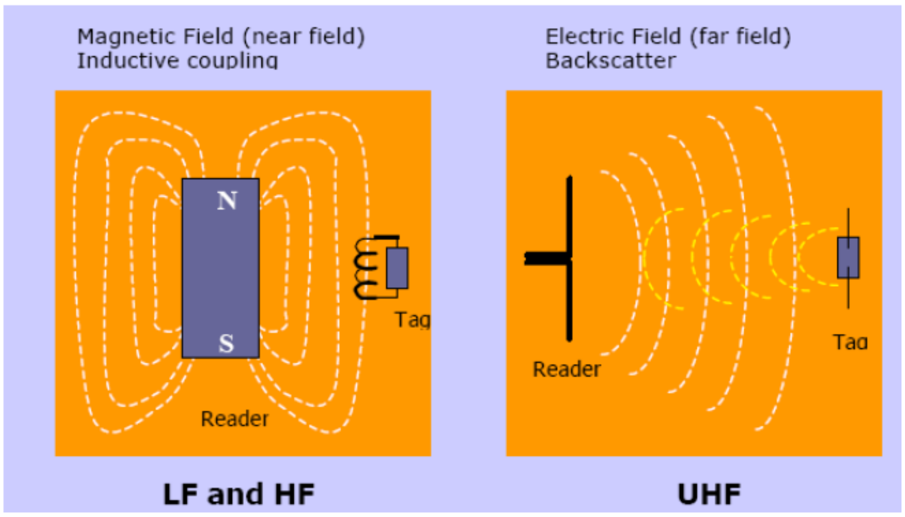


Figure 4. Two different ways of energy and information transfer in tags.

In order to receive energy and communicate with a reader, passive tags use one of the two following methods shown in Fig 4. These are near field which employs inductive coupling of the tag to the magnetic field circulating around the reader antenna (like a transformer), and far field which uses similar techniques to radar (backscatter reflection) by coupling with the electric field. The near field is generally used by RFID systems operating in the LF and HF frequency bands, and the far field for longer read range UHF and microwave RFID systems.

The communication process between the reader and tag is managed and controlled by standard protocols, such as the ISO 15693 and ISO 14443 A for HF.

3.2 Controller

A RFID reader needs a separate controller that tells it when and what to read from the tag or write to the tag. The controller does this my sending specific commands as specified by the RFID manufacturer.

The controller is also responsible for communicating with the networked processor and the GSM module. The communication protocol with the networked processor can be user defined as the user has the freedom to execute his code on the processor. The GSM modules however follow a fixed protocol (details in section 3.4).

Most popular microcontrollers can serve these two purposes provided they have the right communication interfaces to interact with the above mentioned devices. We evaluated some of the above devices available in the market and found that the following were the popular interfaces:

- RFID Reader: USB, RS232
- Networked Processor: RS232
- GSM Module: RS232

Almost all microcontrollers have UART interfaces that can communicate with devices having RS232 interface whereas microcontrollers with USB interfaces are rare and expensive.

3.3 Networked Processor Board

The networked processor board replaces the PC at the service node in the RSM design. Its main function is to communicate with the server over the LAN. It also has to communicate with the controller mentioned above. Hence, it should have Ethernet ports and RS232 interface.

It is also desirable to have an operating system running on the board. This would make it easier to develop applications and would make the process less time consuming.

3.4 GSM Modules

GSM stands for Global System for Mobile communications and is the most popular standard for mobile communications in the world today. Short Message Service (SMS) is a communication service standardized in the GSM mobile communication system, allowing the interchange of short text messages between mobile devices.

We have decided to use SMS for communication between the service node and the server in the absence of LAN as GSM coverage is easily available throughout India. We require two GSM modules, one at the server end and the other at the service node. Each module has a RS232 interface and accepts a Subscriber Identity Module (SIM) card which stores the subscription information and the text messages.

Various commands are given to the GSM modules using AT commands (also known as Hayes AT commands). AT commands can be used for operations like calling a number, sending, reading, or deleting an SMS, looking, reading and deleting phonebook data, reading the signal strength, and so on.

At the service node end, when the controller receives some information from the RFID reader, it composes an SMS and asks (using necessary AT commands) the GSM module to send it to the GSM module connected to server. When the SMS arrives at that GSM module it is stored in the SIM card memory and the server is informed of its arrival. A program running on the server accesses this message and stores the necessary data in the database.

Additionally, users can also make service requests by directly sending SMS to the GSM module connected to the server.

3.5 Interactive Voice Response System

Interactive Voice Response (IVR) is an interactive telephony technology that allows a computer (integrated with telephone line) to detect voice and keypad inputs entered by the user. It is also capable of responding with pre-recorded or dynamically generated audio to further direct users on how to proceed depending upon the inputs from the user.

IVR systems generally use Dual Tone Multi Frequency (DTMF) signals (entered from the telephone keypad) and natural language speech recognition to interpret the caller's response to voice prompts.

In the NSRS system an IVR system is run on the server that is capable of taking requests from the users and storing them in the database.

3.6 Database

The NSRS needs to store data related to various user requests and to display them to the service provider and the user. This data is stored on a database on the server and is managed by software commonly known as a database management system (DBMS). DBMS controls the organization, storage, management, and retrieval of data in the database.

This system of storing data offers various advantageous over data storage in files like data independence, efficient data access and increased data stability and security.

3.7 Web Interface

A main purpose of providing a web interface to the NSRS is to enable the service provider to access the data from any PC connected to the internet. Additionally, the users can also access certain data on the website. The service provider and the users might also be allowed to modify certain data online depending on the application.

4. Implementation for Gas Delivery Service

We implemented a NSRS for a domestic gas delivery service. The service provider in this case is the gas agency and the users are the domestic and commercial gas users.

According to the NSRS plan, each user would be issued a RFID tag. Each RFID tag has a unique serial number associated with it which is located in the chips memory and cannot be erased. For our application we do not need to store any additional information on the tag. At the time of issuing the tag, the tag number is stored in the database along with all the information about the user. Hence, we can easily retrieve the information associated with a serial number at any time from the database.

4.1 GSM Based Service Node

We designed a portable GSM based service node that is shown in Figure 5.

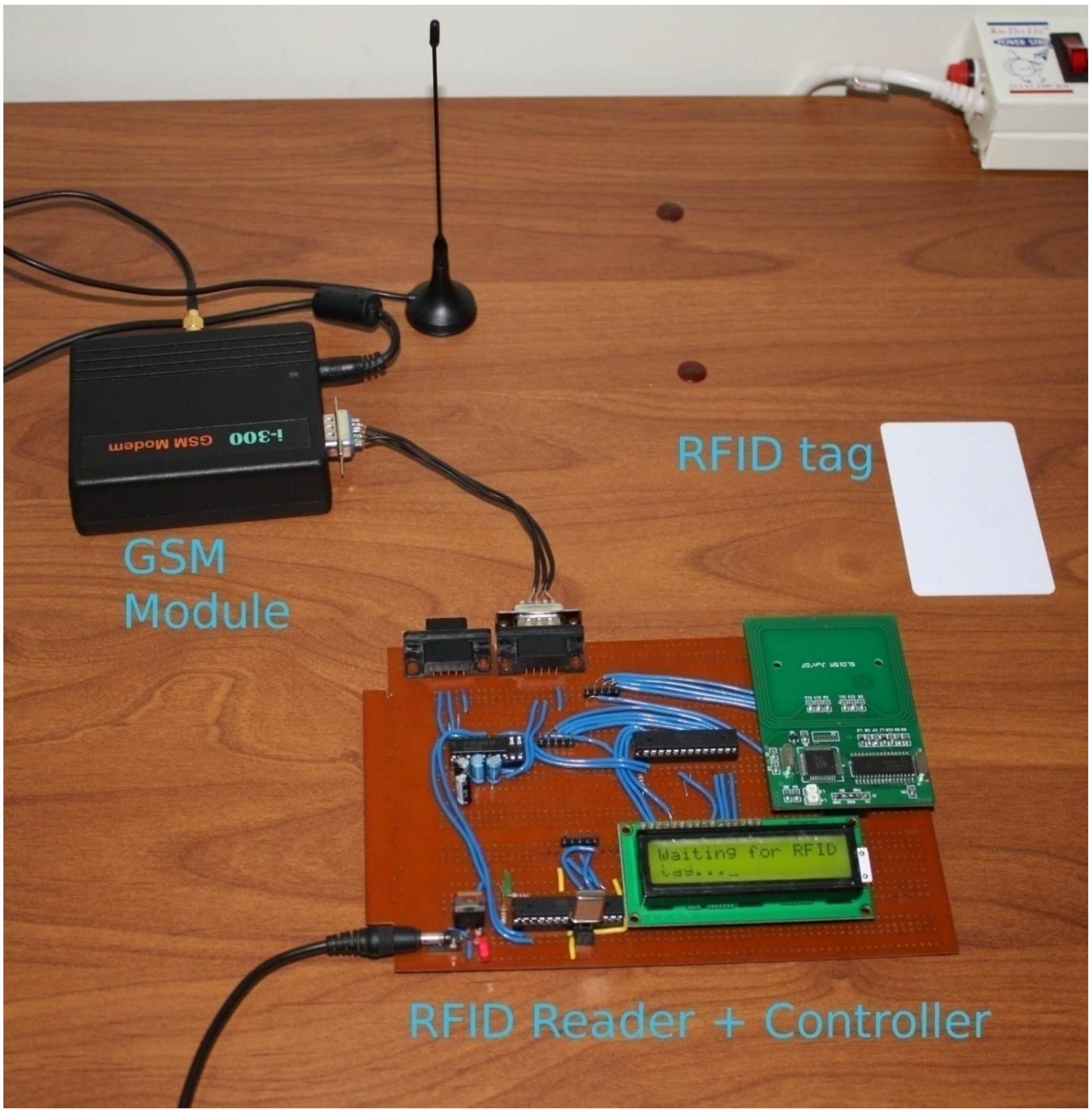


Figure5. The GSM based Service Node

Various components of the LAN based service node are:

• RFID Reader:

For this application, we decided to use a RFID system with operating frequency of 13.56 MHz as the range offered by such system is of the order of a few centimetres which is ideal for our application. We used a serial reader as interfacing with a microcontroller is easier.

We use the SL015M-1 reader from Stronglink. The diagram of the reader is given in Figure 6. Pin 1 is drawn low when a tag is in detection range. Pin 2 and 3 serve as the Tx (transmit) and Rx (receive) pins for UART communication respectively. Pin 4 and 5 are for V_{cc} and ground respectively.

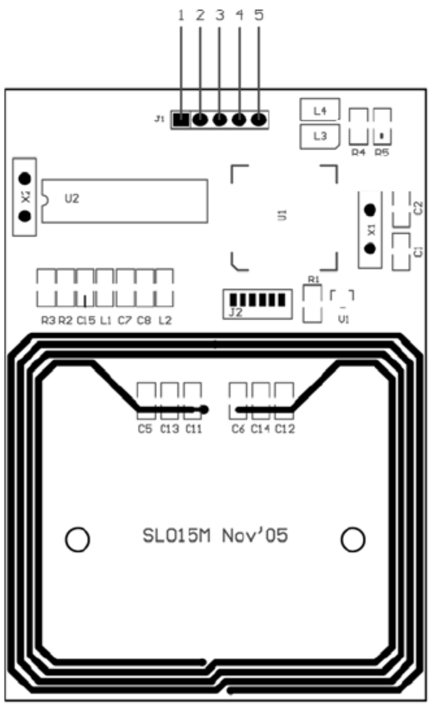
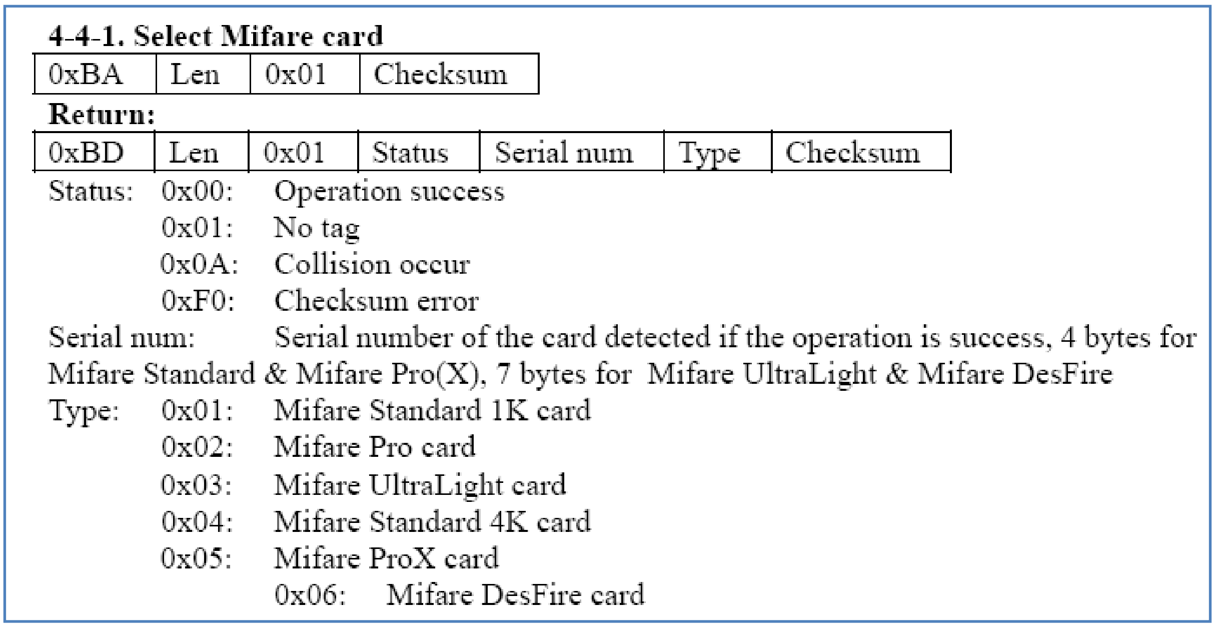


Figure6. Diagram of the RFID reader

The serial communication parameters for this reader are:

Baud rate: 9600
Data : 8 bits
Stop : 1 bit
Parity : None
Flow control: None

The reader accepts 13 different commands details of which can be found in [6]. Out of these we predominantly use the "Select Mifare Card" command which returns the serial number of the tag. The details of this command can be found in Figure7.



4-4-1. Select Mifare card

0xBA	Len	0x01	Checksum

Return:

0xBD	Len	0x01	Status	Serial num	Type	Checksum

Status:	0x00:	Operation success
	0x01:	No tag
	0x0A:	Collision occur
	0xF0:	Checksum error

Serial num: Serial number of the card detected if the operation is success, 4 bytes for Mifare Standard & Mifare Pro(X), 7 bytes for Mifare UltraLight & Mifare DesFire

Type:	0x01:	Mifare Standard 1K card
	0x02:	Mifare Pro card
	0x03:	Mifare UltraLight card
	0x04:	Mifare Standard 4K card
	0x05:	Mifare ProX card
	0x06:	Mifare DesFire card

Figure7. Details of the Select Mifare Card command

For this application we use the MIFARE Classic 1k tags as they are compatible with this reader.

• GSM Module:

Our system uses two i-300 GSM modules obtained from Innovative Embedded Systems. These modules provide a RS-232 interface to interact with the controller board using AT commands.

<u>Following are the steps and corresponding AT commands to send an SMS</u>

- Put the modem in SMS text mode :
  AT+CMGF=1 <ENTER>
- Replace the number below with the recipients phone number :
  AT+CMGS="+31638740161" <ENTER>
  The module will respond with:
  >
- Send the body of the message followed by <CTRL>-<Z>
  Hello World ! <CTRL-Z>
  The modem will respond with the message ID of the message, indicating that the message was sent correctly
  +CMGS: 62

<u>Following are the steps and corresponding AT commands to read an SMS</u>

- To read the message on memory location '1' use:
  AT+CMGR=1 <ENTER>
  The modem will list the single message:

```
+CMGR: "REC READ","+31625044454",,"07/07/05,09:56:03+08"
Hello World!

OK
```

- Delete the message to free memory on the SIM card:
  AT+CMGD=1<ENTER>

• Controller Board

We designed a controller board capable of communicating with both the RFID reader and the GSM module via their serial interfaces. Since, microcontrollers with two UART interfaces are relatively expensive we decided to use two separate microcontrollers. The system is shown in Figure 8.

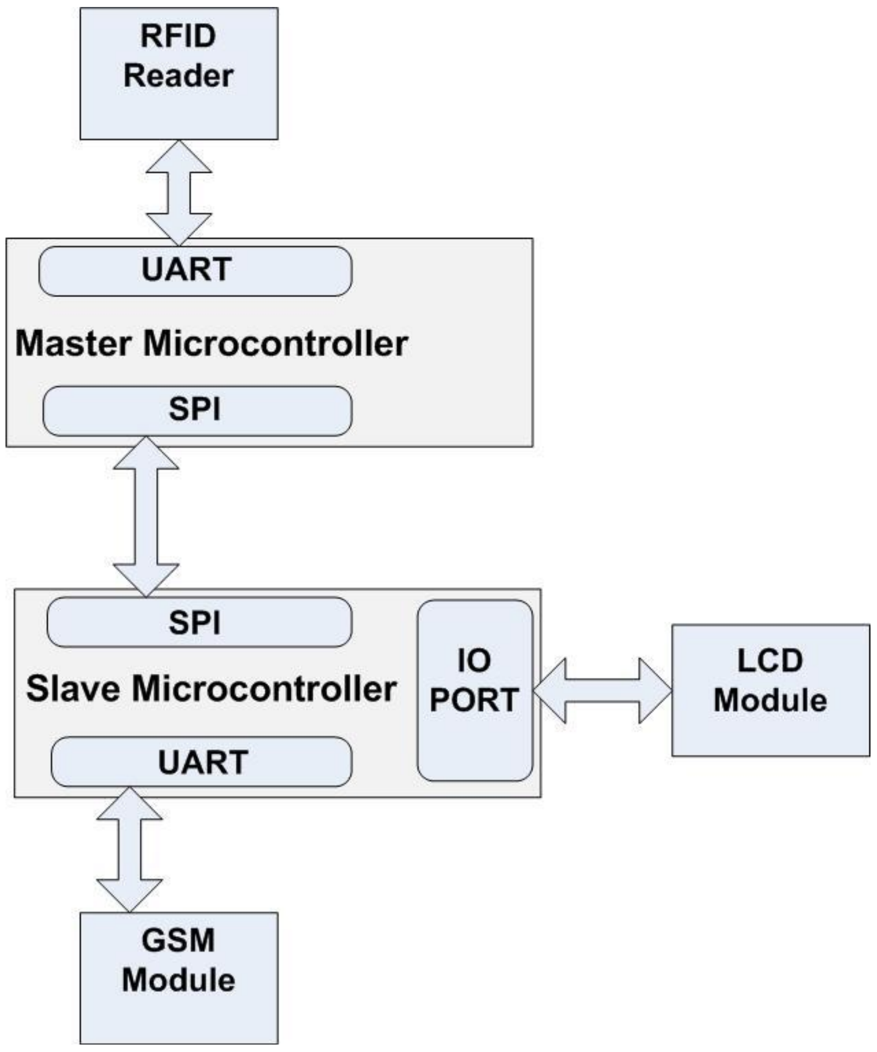


Figure8. Design of the Controller Board

Both the microcontrollers communicate via the Serial Peripheral Interface (SPI). The first microcontroller is configured as the master and uses its UART to communicate with the RFID Reader. It waits for the RFID reader to signal the presence of a tag. It then issues the

"Select Mifare Card" command and as soon as it gets the tag number it transmits this information on the SPI towards the second microcontroller.

The second microcontroller which is configured as the slave listens on its SPI for messages from the master. Its UART is used to communicate with the GSM module and one of its IO ports is used to communicate with a Liquid Crystal Display (LCD). As soon as this microcontroller gets a message from the master, it sends an SMS containing the information by sending appropriate AT commands to the GSM module. It then displays a confirmatory message on the LCD and goes back to listening on the SPI.

We implemented this system using Atmega8 microcontrollers from Atmel on a general purpose Printed Circuit Board (PCB) (shown in Figure 9.)

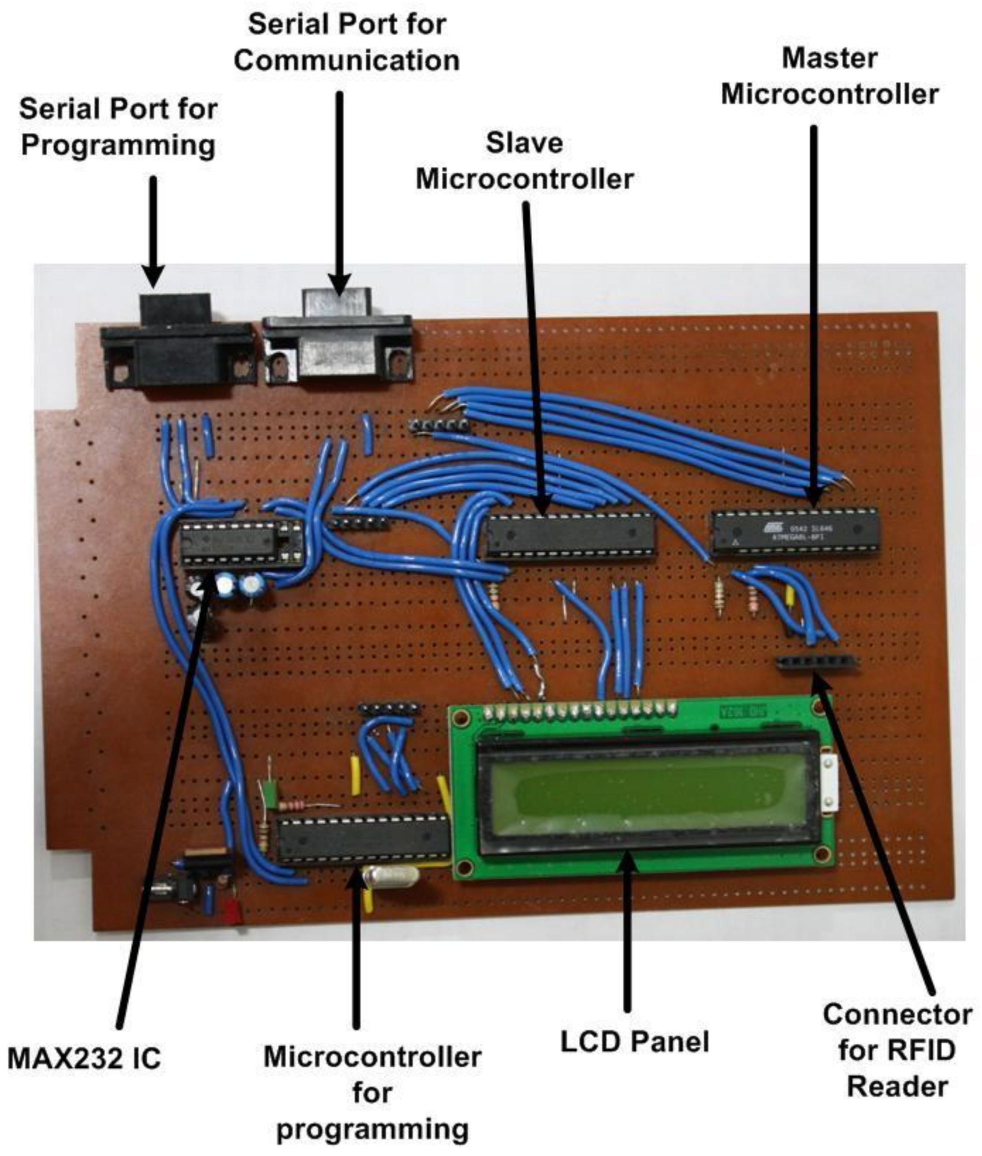

Serial Port for Communication
Master Microcontroller
Serial Port for Programming
Slave Microcontroller
MAX232 IC
Microcontroller for programming
LCD Panel
Connector for RFID Reader

Figure 9. The Controller Board

We also included another microcontroller on the board that is used to program the other two microcontrollers. The flowchart for the Master and the Slave microcontrollers is shown in Figure 10. The actual code is given in Appendix C.1

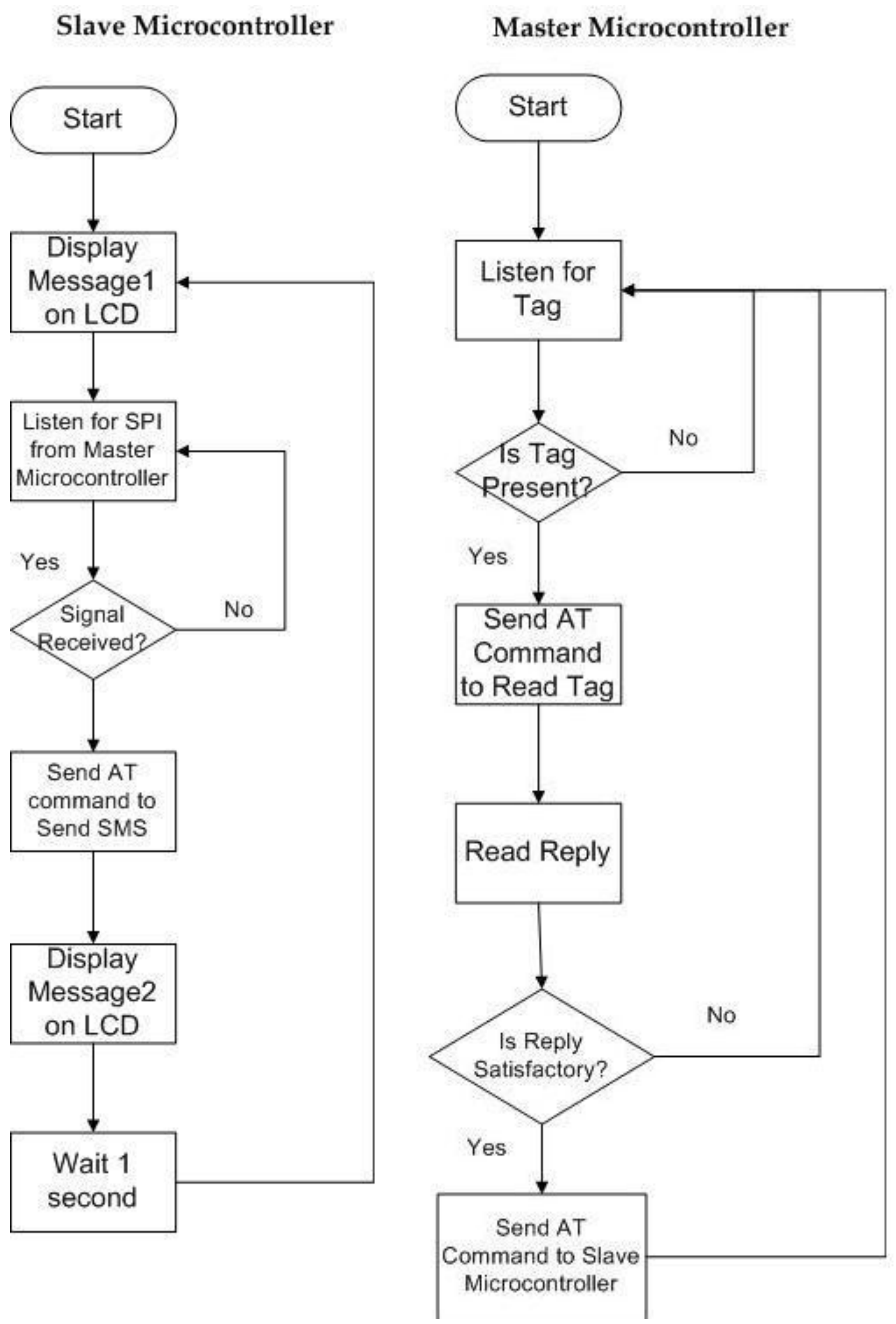


Figure10. Flowchart for Master and Slave Microcontroller

4.2 LAN Based Service Node

We designed a portable LAN based service node that is shown in Figure 11.

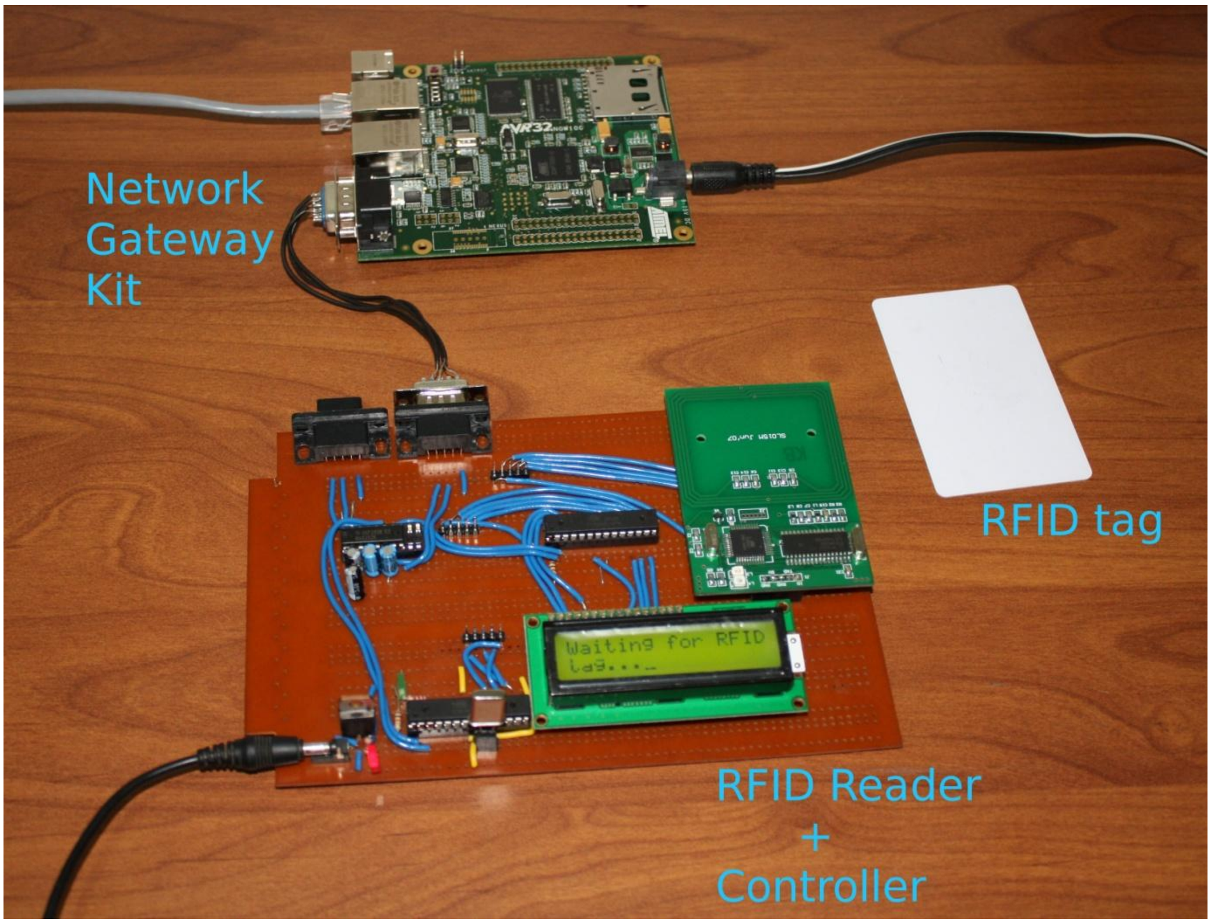


Figure 11. LAN based Service Node

Various components of the LAN based service node are:

- **RFID Reader:**

  Same as in 4.1

- **Network Gateway Kit:**

  For our system we required a processor board that has a minimum of one Ethernet port to access the web server. We decided to use the ATNGW100 Network Gateway Kit manufactured by Atmel. Its main features are:

  - 140MHz AT32AP7000 processor with Atmel's AVR32 RISC architecture
  - 32MB of SDRAM and 16MB of flash
  - Two Ethernet ports
  - Serial interfaces including RS232

- High-speed USB 2.0 B client port
- Preinstalled Linux Operating System
- Can be configured remotely via the Ethernet interface

Since this board kit has a serial port, we need not change the structure of the controller board mentioned in 4.1. We just plug in this kit in place of the GSM module and program this kit so that it understands the previous AT commands, extracts the message from them and send this information to the sever.

- **Controller Board:**

  Same as in 4.1

4.3 Database

We have used MySQL to maintain our database. MySQL is an example of Relational Database Management System (RDBMS) that stores data in the form of tables and the relationship among the data is also stored within the tables. MySQL program runs as a server providing multi-user access to a number of databases. It is the most popular open source DBMS and is known for its ease of use and low operation costs.

The structure of our database is shown in Figure 12. Table 1 gives the description of various tables in the database.

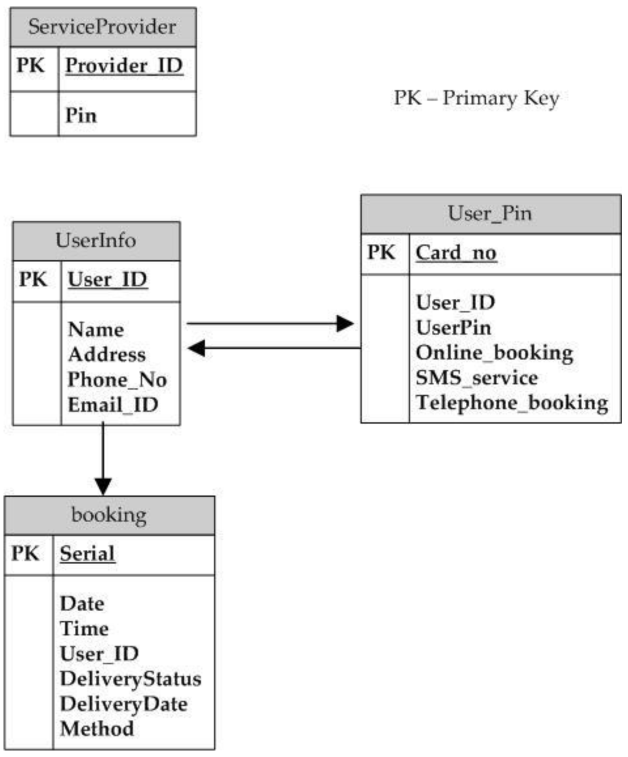


Figure 12. Structure of our database

Database Table	Column Name	Description
ServiceProvider	Provider_ID	6-Digit Service Provider ID required for Login
	Pin	6-Digit PIN provided to the Service Provider for Login
UserInfo	User_ID	Unique 4-Digit User ID required for Authentication
	Name	Name of the User
	Address	Address of the User
	Phone_No	Phone Number of the User
	Email_ID	Email ID of the User

User_Pin	User_ID	Unique 4-Digit User ID required for Authentication
	UserPin	4-Digit User PIN required for Authentication
	Online_booking	0 – Access Denied 1 – Access Allowed
	SMS_service	0 – Access Denied 1 – Access Allowed
	Telephone_booking	0 – Access Denied 1 – Access Allowed
Booking	Serial	Serial Number of the booking
	Date	Date when booking was done
	Time	Time when booking was done
	User_ID	Unique 4-Digit User ID whose booking was done
	DeliveryStatus	0 – Delivery Pending 1 – Delivered
	DeliveryDate	Date on which gas was Delivered
	Method	0 – Online Booking 1 – Booking by LAN RFID Reader 2 – Booking by GSM RFID Reader 3 – Booking by Telephone 4 – Booking by SMS

Table 1. Description of various tables

4.4 Web Interface

The web interface for this application is divided into two sections:

- Service Provider Interface
- User Interface

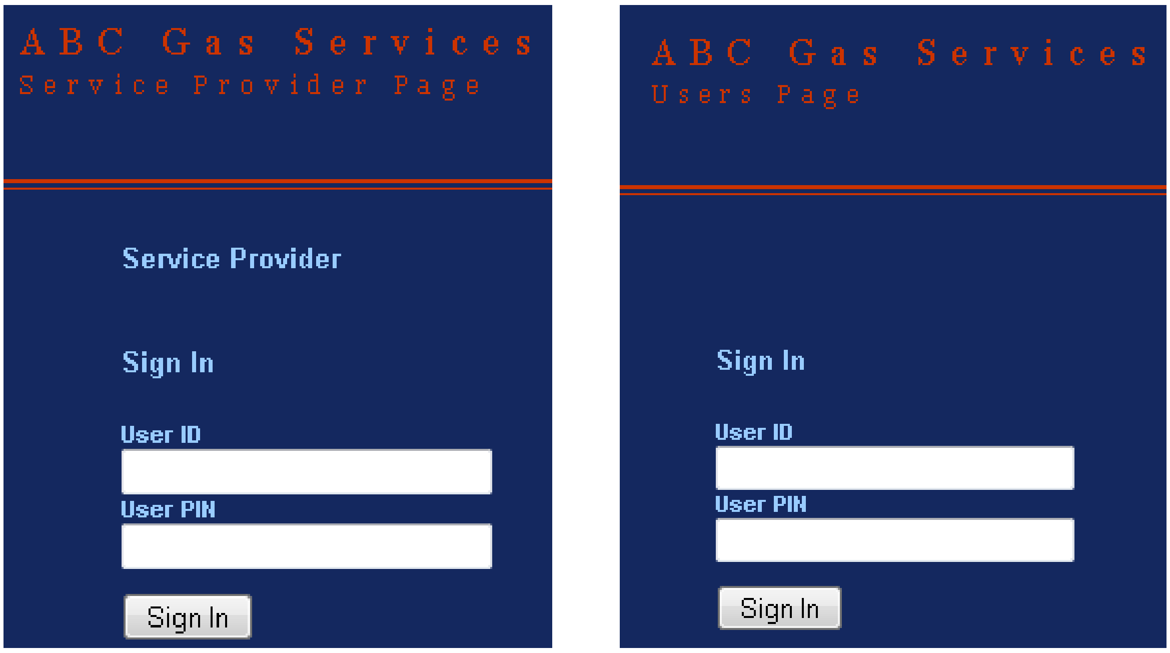
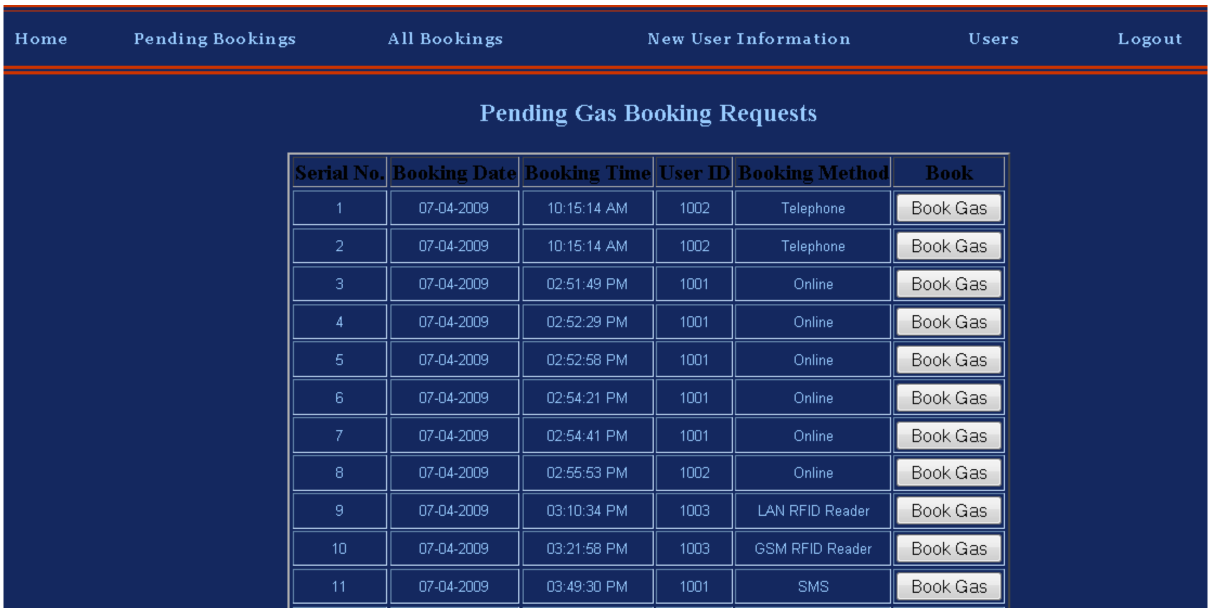


Figure 13. Screenshots of Service Provider Login Page and User Login Page.

Service Provider Interface

The following features and services are given to the Service Provider:

- Information about all the Pending Requests as well as the Method used for booking Gas (Online/ LAN RFID Reader/ GSM RFID Reader/ Telephone/ SMS).
- Update the status of a request from "Pending" to "Delivered".
- View a list of all requests i.e. pending and delivered.
- View a list of Information about all the users along with their User ID, Card number, Name, Address, Phone No. and Email-id.
- Add a New User to the System along with all the information and RFID card number.



Pending Gas Booking Requests

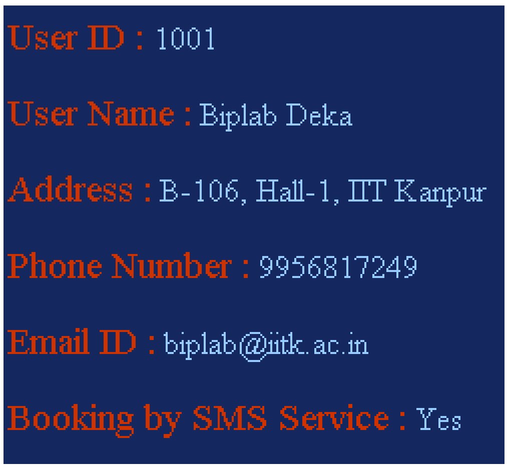
Serial No.	Booking Date	Booking Time	User ID	Booking Method	Book
1	07-04-2009	10:15:14 AM	1002	Telephone	Book Gas
2	07-04-2009	10:15:14 AM	1002	Telephone	Book Gas
3	07-04-2009	02:51:49 PM	1001	Online	Book Gas
4	07-04-2009	02:52:29 PM	1001	Online	Book Gas
5	07-04-2009	02:52:58 PM	1001	Online	Book Gas
6	07-04-2009	02:54:21 PM	1001	Online	Book Gas
7	07-04-2009	02:54:41 PM	1001	Online	Book Gas
8	07-04-2009	02:55:53 PM	1002	Online	Book Gas
9	07-04-2009	03:10:34 PM	1003	LAN RFID Reader	Book Gas
10	07-04-2009	03:21:58 PM	1003	GSM RFID Reader	Book Gas
11	07-04-2009	03:49:30 PM	1001	SMS	Book Gas

Figure 14. Screenshot of Pending Gas Booking Request page of the service Provider web interface.

User Interface

The service users can log on this section and carry out the following actions:

- Choose the number of cylinders to book and make a request
- Can view a list of all the request made by him/her and see their status
- Update his information like phone number or address.


User ID : 1001

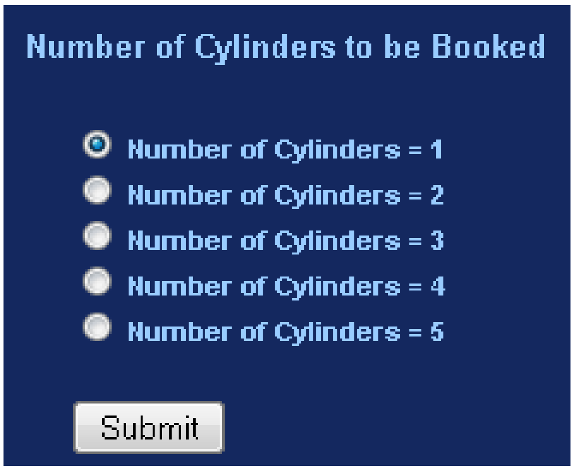
User Name : Biplab Deka

Address : B-106, Hall-1, IIT Kanpur

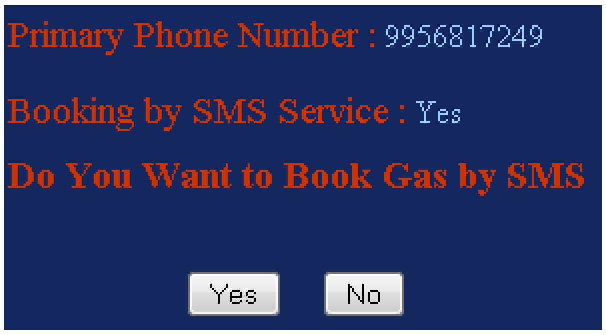
Phone Number : 9956817249

Email ID : biplab@iitk.ac.in

Booking by SMS Service : Yes


Number of Cylinders to be Booked

[✓ Number of Cylinders = 1]
Number of Cylinders = 2
Number of Cylinders = 3
Number of Cylinders = 4
Number of Cylinders = 5



Primary Phone Number : 9956817249

Booking by SMS Service : Yes

Do You Want to Book Gas by SMS

Figure15. Screenshot of the User's section of the gas booking website providing User Information, Cylinder Booking Service and Booking by SMS activation information.

New User Registration

To add new user to the database, service provider needs to enter the required information of the user. Card number of the RFID tag being provided to the user can be obtained by reading the card number through an RFID reader. 4-Digit User ID and 6-Digit User PIN will be generated by the system itself. User PIN will be randomly generated and will be mailed to the new user being added.

Service Provider will be having the authority to provide specific services to the new user. Optional services to be activated by the service provider for the new user are:

- Online booking
- Booking by SMS
- Booking by Telephone

Booking by SMS can be activated by the online access as well. However, booking by SMS will be restricted to the only cell phone number which has been registered in the database (can be changed by online accessing the user profile) to prevent any hoax booking and developing more secure system.

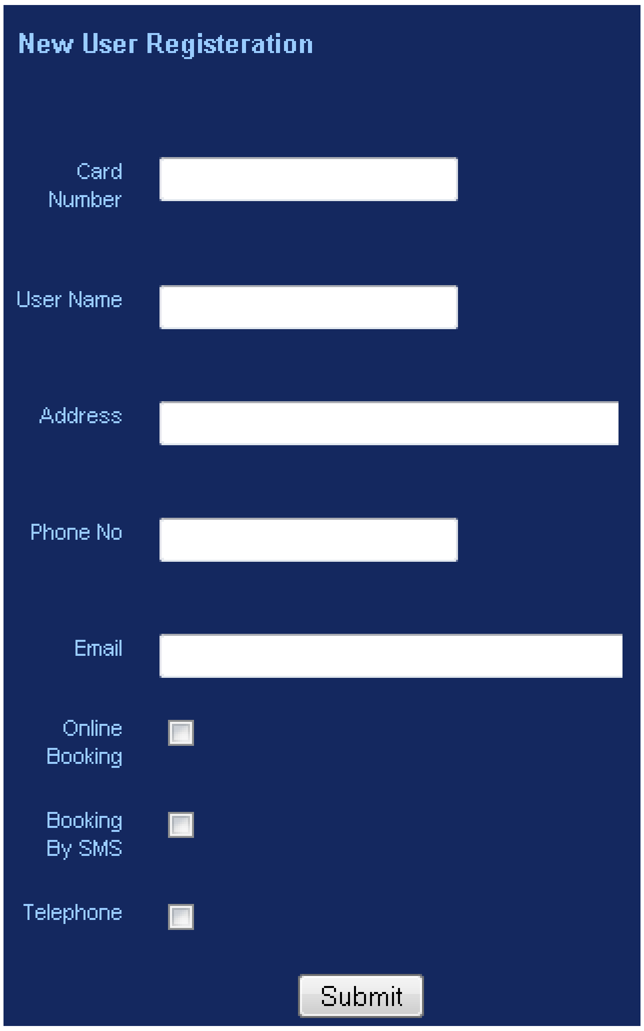


Figure 16. The new user Registration Form

4.5 Interactive Voice Response (IVR) System

To handle service requests made through telephone calls we have implemented an Interactive Voice Response (IVR) System. This alternate way of access application has been developed to provide users with 24-hour accessibility and affordability.

The set up of the IVR system consists of:

- **Voxzone X100P Telephony Card:** The Voxzone X100P is a single FXO interface for connecting the Asterisk PBX server (developed dial plan for the handling of the call) to the Analog Telephone line.
- **Asterisk software:** Asterisk is an open source free software implementation of a telephone private branch exchange (PBX). It allows attached telephones to make calls to one another, and to connect to other telephone services including the public switched telephone network (PSTN). Asterisk can also be programmed via external applications using Asterisk Gateway Interface (AGI). This has been utilised to write flexible code for the dial plan (sequence of steps to be followed when user makes a call to the IVR system).
- **Telephone Line:** Telephone Line is required to connect it to the telephony card installed within the computer.

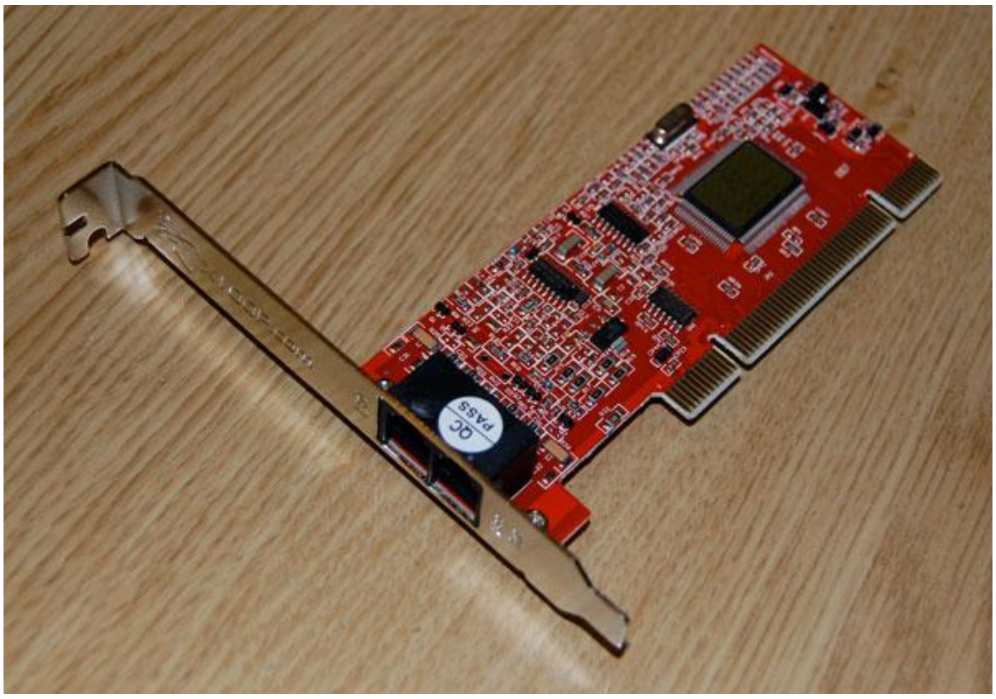


Figure 17. Voxzone X100P Telephony Card installed in the computer

System Operation

The user has to call the system at (0512-259-7847) to book gas cylinders. The system is up for 24-hours, providing flexibility to the user and extending business hours of operation. The user will be guided by the interactive response system developed. The user will be asked to enter 4-Digit User ID and 6-Digit User Pin being provided by the service provider (User PIN can be changed at a later stage by online accessing the user profile). The user will be able to book gas cylinders (either 1/2/3) only after being authenticated. The user will be given maximum of 3-trials and the call will get disconnected automatically if user does not enter correct information in 3-trials. The call cannot extend beyond nearly 3 minutes. This feature has been added to ensure that system do not become busy if call gets disconnected abruptly or incidentally.

Figure 18 depicts the flowchart of the dial plan. Table 2 describes the Asterisk Commands used in Designing IVR System.

Salient features of the IVR System:

- Increase operational efficiency
- Scales well to handle large call volumes
- 24 hours Service - Extend the business hours of operation
- Improved customer experience

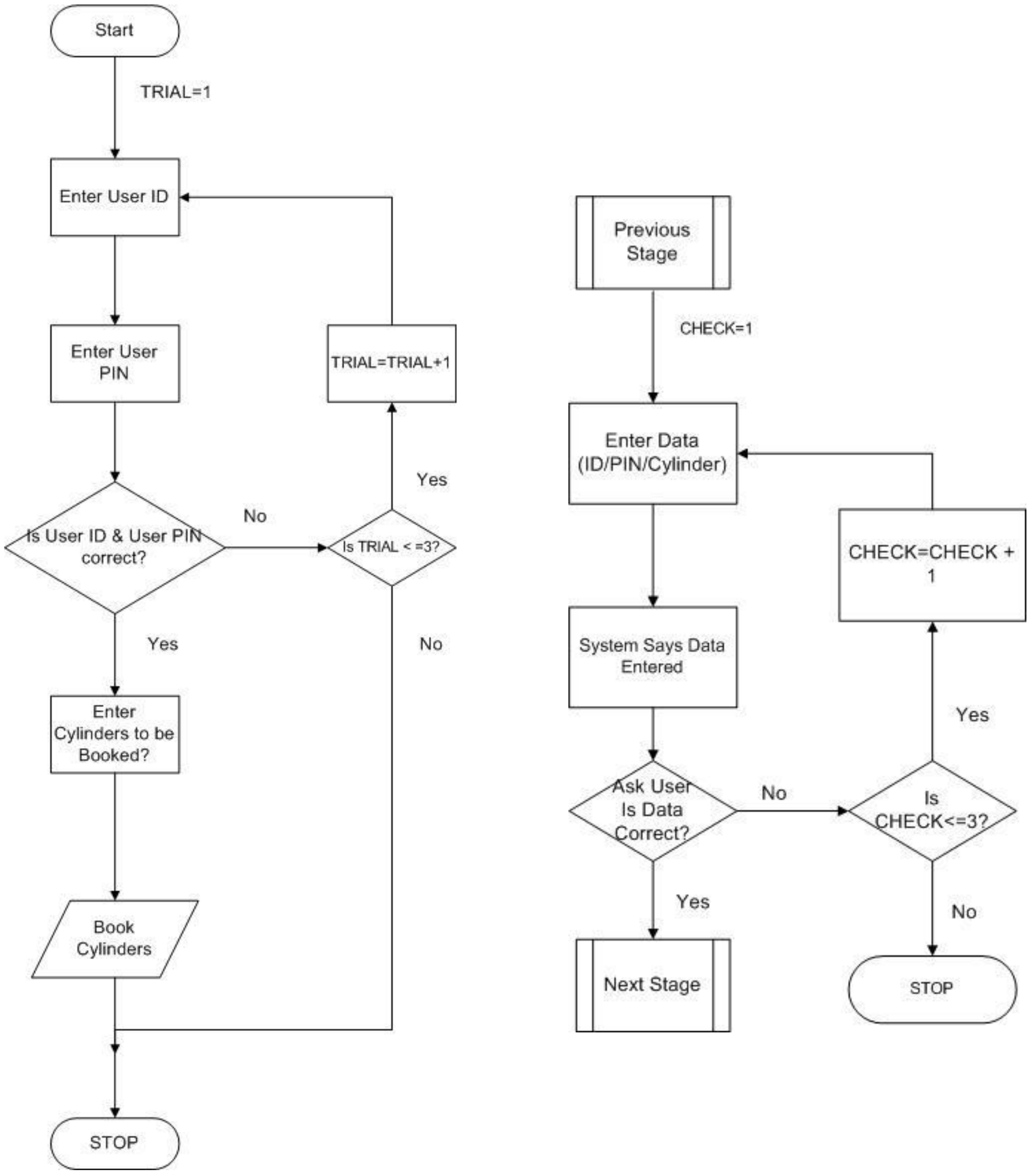


Figure 18. Flowchart depicting the sequence of steps followed by the call to handle the input data (User ID/ User PIN/ Cylinders) entered by the user and to confirm it. The maximum number of invalid trials allowed at each stage is three.

Command	Function
ANSWER	Asserts answer
SET(TIMEOUT(absolute))	Sets channel absolute timeout: The absolute maximum amount of time permitted for a call.
WAIT	Wait for given time before proceeding to next step of Dialplan
EXEC	Executes a given Application. (Applications are the functions you use to create a dial plan in extensions.conf( ).
SET(GLOBAL(VAR))	Sets a Global channel variable (VAR)
PLAYBACK	Playback the saved audio file in /var/lib/asterisk/sounds/
GOTO	Goto a particular priority, extension, or context
GOTOIF	Conditional Goto
READ	Read a variable in the form for DTMF tones as pressed by the caller
SAYDIGITS	Says a given digit string
AGI	To call another program written in different language(PHP) through Asterisk Gateway Interface (AGI)
HANGUP	Hangup the current channel
STREAM FILE	Sends audio file on channel
VERBOSE	Logs a message to the asterisk verbose log

Table 2. Asterisk Commands used in Designing IVR System

4.6 SMS Reception at Server

A script runs on the server in order to process the SMS received at the Server end from the GSM based Service Node. This script is written in the Python programming language and can be found in Appendix C.2

We have also extended our system to cater to requests made by users via SMS as cell phones find widespread usage in India. The same script also has the capability to process messages received directly from users who have used the SMS booking service.

Steps to be followed to book gas cylinder by SMS are:

- Enter 4-Digit User ID
- Followed by a blank space ('_')
- Enter 6-Digit User PIN
- Again followed by a blank space('_')
- Finally enter a full stop ('.').
- Send SMS to '9651923022'

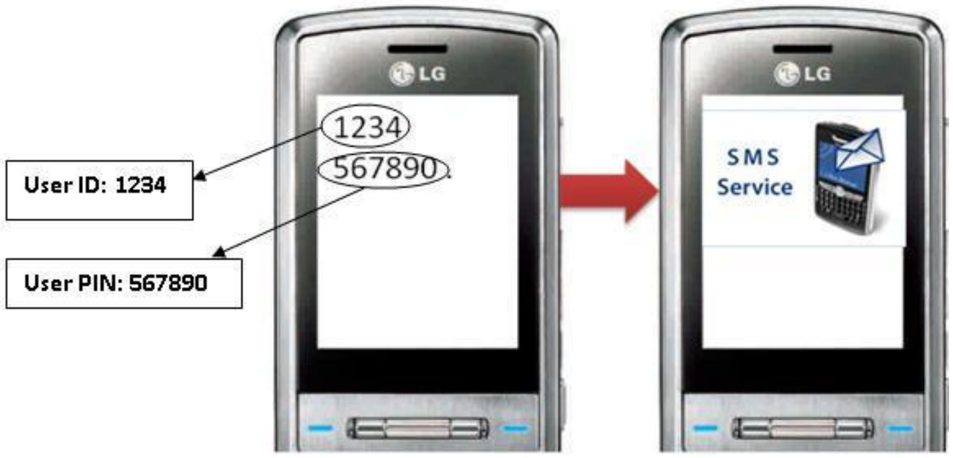


Figure19. Requests by SMS

Salient features of booking by SMS Service

- Less time-consuming
- Reduced call volume
- Easy to use while roaming
- Can be used to issue important alerts (broadcasting) to customers by Service Provider
- Improved customer experience

5. Salient Features

5.1 Low System Cost

By replacing the PC at the service node by a low cost processor board we have been able to reduce the cost of the system significantly. As the cost of the node comes down, it makes it more attractive to deploy a large number of such nodes all of which connect to one server and adds practicality to the system.

5.2 Flexibility for the User

The system has provided provision for making requests to the user in four ways. Firstly, the user can use the web interface. This is intended for users who have easy access to internet. For users who have no such facility, we have the provision of making requests using the RFID based service nodes present nearby their locality. To cater to the large base of mobile phone users in India, we have also implemented SMS based request system to add mobility to the system. And lastly, our system also caters to telephone based service requests for household users. The web interface is available for all the users to monitor the status of their requests via the website and other features provided in web interface.

5.3 Ease of use for the Service Provider

Since the service provider need not install any software, he becomes truly platform independent. He can work from any environment on any computer. All he needs is a web browser and access to the centralised database. He has the flexibility to access database easily by utilising different features provided within service provider web interface.

5.4 Easy Data Management

By using the MySQL Database Management System (DBMS) we have centralized the data storage and have made it easier to access and display the data. It also makes it easier to take backups. It is easy to use, fast and reliable. It even adds feature of multi-platform availability.

5.5 Open Source Software

Our system uses open source software only and hence enjoys all its advantages including zero investment in software, flexibility to change the source code and to develop customized software for our platforms. The software we used are:

- Apache : Web Server
- PHP : Scripting language for web development
- MySQL : Database Management System
- Linux : Operating System for the networked processor
- Python : Programming language
- Asterisk : Telephone Private Branch Exchange (PBX)

5.6 Remote Configuration of the Service Node

The Network Gateway Kit that we use as the networked processor in the service node enables it to be accessed remotely via Ethernet. Hence, we can configure it and change it settings and software remotely. We can add different utilities to the network processor kit by accessing it remotely.

6. Conclusions

A Networked Service Request System has been proposed using various technologies like GSM, RFID and IVR System and has been successfully implemented in testing phase in VLSI Lab.

It has been designed for a real life application and is functioning satisfactorily and can be extended for other important applications like Centralised Emergency Request System.

7. Scope for Future Work

Future work that can be incorporated into the NSRS design are:

7.1 Designing a PCB for the Controller Board

Since the design of the controller board has been finalised, it can now be implemented on a custom made Printed Circuit Board.

7.2 Improving the System Security

The communication between the networked processor kit and the web server requires no authentication at present. This could lead to security problems and can be made more secure. SQL injection can be looked into consideration to prevent illegal online access to the database.

7.3 Further Cost Reduction

At present we have build the system using modules made by different manufacturers. Further cost reduction can be achieved by designing these modules ourselves on a single PCB.

7.4 Extending the IVR System

The interactive voice response system has been developed in 'English' language which can be extended to other regional languages to add versatility to the system and enhance consumer experience.

References

1. H. Bhatt, B. Glover, "*RFID Essentials*" Publisher: O'Reilly Pub. Date January 2006.

2. Mayank Raj, Gopesh Mittal, "Remote Service Machine (RSM)", B. Tech Project, IIT Kanpur, April 2008.

3. A. Harris, "*PHP/MySQL Programming for the Absolute Beginner*", Premier Press, 2003.

4. M. Spencer, M. Allison, C. Rhodes, *"The Asterisk Handbook version 2",* The Asterisk Documentation Team, March 2003.

5. Atmega8 datasheet (http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf)

6. Stronglink SL015M-1 User Manual (http://www.stronglink.cn/english/sl015m1.htm)

7. Python Quick Reference (http://rgruet.free.fr/PQR24/PQR2.4.html)

8. ATNGW100 Documentation (http://www.avrfreaks.net/wiki/index.php/Documentation:NGW)

Appendix A: Work Breakdown

Biplab Deka

- Design and Soldering of the Controller Board
- Microcontroller Programming
- Programming the server to receive SMS using Python
- Programming the Network Gateway Kit using shell script

Piyush Keshri

- Implementation of the IVR System using Asterisk
- Design and implementation of the website in PHP and HTML
- Implementation of database in MySQL

Appendix B: Cost Analysis

Product Used	**Cost**
Network Gateway Kit	$100
Voxzone X100P Telephony Card	$ 30
GSM Modules	2 x $ 60
RFID Reader	$ 20
SIM Cards	2 x $ 2
Microcontroller	3 x $ 2
LCD	$ 2
Softwares	Free
Miscellaneous	$ 20
TOTAL	**$302**

Appendix C: Codes

C.1 Code for Atmega8 Microcontrollers in C

- **Master Microcontroller**

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

#define F_CPU 8000000UL

#define DDR_SPI DDRB;
#define DD_MISO 4;
#define DD_MOSI 3;
#define DD_SS 2;
#define DD_SCK 5;

void USART_Init();
char USART_Receive( void );
void USART_Transmit( char );
void led_off(void);
void led_on(void);
void blink(char, char);
void SPI_MasterInit(void);
void SPI_MasterTransmit(unsigned char);

void main(){

	unsigned char answer[10];
	int i;
	sei();

	DDRD &= ~(0x04);
	MCUCR &= 0xFC; //INT0 is low level triggered


	//set MCUCR to 1010xxxx  : enables sleep modes and
	//selects power down mode
	MCUCR &= 0xAF;    //reset the bits  x0x0xxxx
	MCUCR |= 0xA0;


	USART_Init();
	SPI_MasterInit();

	PORTB |= (1<<2);

	//wait for 2 seconds
	for(i=0; i<20; i++){

		_delay_ms(100);
	}

	while(1){

		GICR |= 0x40;  //enable INT0
		sleep_cpu();
		USART_Transmit(0xBA);
		USART_Transmit(0x02);
		USART_Transmit(0x01);
		USART_Transmit(0xB9);
		for(i=0; i<2; i++){
			answer[i]= USART_Receive();
		}
		for(i=2; i<2+answer[1]; i++){
			answer[i]= USART_Receive();
		}

		blink(answer[1]/2, 1);

		if(answer[1] == 0x08){

			PORTB &= ~(1<<2);
			_delay_ms(5);
			SPI_MasterTransmit(answer[4]);
			SPI_MasterTransmit(answer[5]);
			SPI_MasterTransmit(answer[6]);
			SPI_MasterTransmit(answer[7]);
			//SPI_MasterTransmit('i');
			_delay_ms(5);
			PORTB |= (1<<2);
			_delay_ms(5);

		}

		while(1){
			if((PIND & 0x04) == 0x04)
				break;
		}

	}

}


ISR(INT0_vect){
	GICR &= 0xBF; //disable INT0
}

void SPI_MasterInit(void)
{
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = ((1<<SPE)|(1<<MSTR)|(1<<SPR0));
    // Set MOSI, SCK output, all others input
  DDRB |=((1<<3)|(1<<5)|(1<<2));
}


void SPI_MasterTransmit(unsigned char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF))){
  }
}

void USART_Init(){
	// USART initialization
	// Communication Parameters: 8 Data, 1 Stop, No Parity
	// USART Receiver: On
	// USART Transmitter: On
	// USART Mode: Asynchronous
	// USART Baud Rate: 9600
	UCSRA=0x00;
	UCSRB=0x18;
	UCSRC=0x86;
	UBRRH=0x00;
	UBRRL=0x33;
```

```
}

void USART_Transmit(char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Put data into buffer, sends the data */
  UDR = data;
}



char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get and return received data from buffer */
  return UDR;
}




void blink(char times, char delay){

       int i,j;
       for(i=0; i<times; i++){
              led_on();

              for(j=0;j<(3*delay); j++){
                     _delay_ms(100);
              }

              led_off();
              for(j=0;j<(3*delay); j++){
                     _delay_ms(100);
              }
       }
}

void led_on(){
       char c[5];
       int i;

       USART_Transmit(0xBA);
       USART_Transmit(0x03);
       USART_Transmit(0x40);
       USART_Transmit(0x01);
       USART_Transmit(0xF8);

       for(i=0; i<5; i++){
              c[i]= USART_Receive();
       }
}

void led_off(){
       char c[5];
       int i;

       USART_Transmit(0xBA);
       USART_Transmit(0x03);
       USART_Transmit(0x40);
       USART_Transmit(0x00);
       USART_Transmit(0xF9);
```

```
        for( i=0; i<5; i++){
                c[i]= USART_Receive();
        }
}
```

- **Slave Microcontroller**

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <stdlib.h>

#define F_CPU 8000000

#define DDR_SPI DDRB;
#define DD_MISO 4;
#define DD_MOSI 3;
#define DD_SS 2;
#define DD_SCK 5;
void SPI_SlaveInit(void);
char SPI_SlaveReceive(void);
char htc( char);
void send_msg(void);
void send_no(void);
void send_msg(void);
void USART_Init(void);
void USART_Transmit(char);
char USART_Receive(void);

void main(){
        char c1, c2, c3, c4, d;
        char a[3];
        char s[16];
        LCDinit();
        LCDclr();
        int i;

        USART_Init();
        _delay_ms(20);
        sms_mode();

        for(i=0;i<16;i++){
                s[i]='0';
        }
        _delay_ms(20);

        SPI_SlaveInit();

        while(1){

                LCDclr();

                LCDsendChar('W');
                _delay_ms(20);
                LCDsendChar('a');
                _delay_ms(20);
                LCDsendChar('i');
                _delay_ms(20);
                LCDsendChar('t');
                _delay_ms(20);
                LCDsendChar('i');
                _delay_ms(20);
                LCDsendChar('n');
                _delay_ms(20);
                LCDsendChar('g');
```

```
_delay_ms(20);
LCDsendChar(' ');
_delay_ms(20);
LCDsendChar('f');
_delay_ms(20);
LCDsendChar('o');
_delay_ms(20);
LCDsendChar('r');
_delay_ms(20);
LCDsendChar(' ');
_delay_ms(20);
LCDsendChar('R');
_delay_ms(20);
LCDsendChar('F');
_delay_ms(20);
LCDsendChar('I');
_delay_ms(20);
LCDsendChar('D');

LCDGotoXY(0,1);

_delay_ms(20);
LCDsendChar('t');
_delay_ms(20);
LCDsendChar('a');
_delay_ms(20);
LCDsendChar('g');
_delay_ms(20);
LCDsendChar('.');
_delay_ms(20);
LCDsendChar('.');
_delay_ms(20);
LCDsendChar('.');
_delay_ms(20);

c1 = SPI_SlaveReceive();
c2 = SPI_SlaveReceive();
c3 = SPI_SlaveReceive();
c4 = SPI_SlaveReceive();


LCDclr();

send_no();
_delay_ms(100);




d = htc(c1/16);
USART_Transmit(d);

d = htc(c1%16);
USART_Transmit(d);

d=htc(c2/16);
USART_Transmit(d);

d=htc(c2%16);
USART_Transmit(d);

d=htc(c3/16);
USART_Transmit(d);

d=htc(c3%16);
USART_Transmit(d);

d=htc(c4/16);
```

```
USART_Transmit(d);

d=htc(c4%16);
USART_Transmit(d);

USART_Transmit(0x1A);
_delay_ms(100);

USART_Transmit(0x0D);
_delay_ms(100);

USART_Transmit(0x0A);
_delay_ms(100);

USART_Transmit(0x04);
_delay_ms(100);


LCDclr();

LCDsendChar('Y');
_delay_ms(20);
LCDsendChar('o');
_delay_ms(20);
LCDsendChar('u');
_delay_ms(20);
LCDsendChar('r');
_delay_ms(20);
LCDsendChar(' ');
_delay_ms(20);
LCDsendChar('r');
_delay_ms(20);
LCDsendChar('e');
_delay_ms(20);
LCDsendChar('q');
_delay_ms(20);
LCDsendChar('u');
_delay_ms(20);
LCDsendChar('e');
_delay_ms(20);
LCDsendChar('s');
_delay_ms(20);
LCDsendChar('t');
_delay_ms(20);
LCDsendChar(' ');
_delay_ms(20);
LCDsendChar('h');
_delay_ms(20);
LCDsendChar('a');
_delay_ms(20);
LCDsendChar('s');

LCDGotoXY(0,1);

_delay_ms(20);
LCDsendChar('b');
_delay_ms(20);
LCDsendChar('e');
_delay_ms(20);
LCDsendChar('e');
_delay_ms(20);
LCDsendChar('n');
_delay_ms(20);
LCDsendChar(' ');
_delay_ms(20);
LCDsendChar('r');
_delay_ms(20);
LCDsendChar('e');
```

```
            _delay_ms(20);
            LCDsendChar('c');
            _delay_ms(20);
            LCDsendChar('e');
            _delay_ms(20);
            LCDsendChar('i');
            _delay_ms(20);
            LCDsendChar('v');
            _delay_ms(20);
            LCDsendChar('e');
            _delay_ms(20);
            LCDsendChar('d');
            _delay_ms(20);
            LCDsendChar('.');

            for (int i=0; i<50; i++){
                  _delay_ms(100);
            }
      }

}

void send_msg(){
      USART_Transmit('h');
      USART_Transmit('i');
      USART_Transmit(0x1A);
}

void send_no(){
      USART_Transmit('A');
      USART_Transmit('T');
      USART_Transmit('+');
      USART_Transmit('C');
      USART_Transmit('M');
      USART_Transmit('G');
      USART_Transmit('S');
      USART_Transmit('=');
      USART_Transmit('"');
      USART_Transmit('+');
      USART_Transmit('9');
      USART_Transmit('1');
      USART_Transmit('9');
      USART_Transmit('6');
      USART_Transmit('5');
      USART_Transmit('1');
      USART_Transmit('9');
      USART_Transmit('2');
      USART_Transmit('3');
      USART_Transmit('0');
      USART_Transmit('2');
      USART_Transmit('2');
      USART_Transmit('"');
      USART_Transmit(0x0D);
      USART_Transmit(0x0A);

}

void sms_mode(){
      USART_Transmit(0x41);
      USART_Transmit(0x54);
      USART_Transmit(0x2B);
      USART_Transmit(0x43);
      USART_Transmit(0x4D);
      USART_Transmit(0x47);
      USART_Transmit(0x46);
      USART_Transmit(0x3D);
      USART_Transmit(0x31);
      USART_Transmit(0x0D);
```

```
	USART_Transmit(0x0A);
}


char htc(char c){
	if(c< 10)
		return c+48;
	else
		return c+55;
}

void SPI_SlaveInit(void)
{
  // SPI initialization
// SPI Type: Slave
// SPI Clock Rate: 62.500 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
//	SPCR=0x43;
//	SPSR=0x00;

/* Set MISO output, all others input */
  DDRB |= (1<<4);
  /* Enable SPI */
  SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
  /* Wait for reception complete */
  while(!(SPSR & (1<<SPIF)))
    ;
  /* Return data register */
  return SPDR;
}


void USART_Init(){
	// USART initialization
	// Communication Parameters: 8 Data, 1 Stop, No Parity
	// USART Receiver: On
	// USART Transmitter: On
	// USART Mode: Asynchronous
	// USART Baud Rate: 9600
	UCSRA=0x00;
	UCSRB=0x18;
	UCSRC=0x86;
	UBRRH=0x00;
	UBRRL=0x33;
}

void USART_Transmit(char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
	;
  /* Put data into buffer, sends the data */
  UDR = data;
}



char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
	;
```

```
  /* Get and return received data from buffer */
  return UDR;
}
```

C.2 Python Code (code used on server to receive SMS)

```
#!/usr/bin/python
import serial
import time
import os
ser=serial.Serial('/dev/ttyUSB0',9600,timeout=0)
i=1
while i<26:
	print i
	print "AT+CMGD="+str(i)
	time.sleep(0.1)
	ser.write("AT+CMGD="+str(i)+'\x0d\x0a')
	i=i+1

print "All messages deleted"

#configure device in text mode
ser.write("AT+CMGF=1"+'\x0d\x0a')

time.sleep(5)
ser.flushInput()

print "Input Flushed"


ser.close()

ser=serial.Serial('/dev/ttyUSB0',9600,timeout=1)

while i<200:
	x=ser.readline()

	if len(x) > 2:
		#print "printing x: "+x
		#print x[0:5]

		if x[0:5] == "+CMTI":
			print "cmti received"
			j=0
			msg_number=""
			k=0
			while j < len(x):
				if x[j] == ',':
					k=1
				elif x[j] == '\x0d':
					k==0

				if k==1:
					msg_number = msg_number + str(x[j])
				j = j+1
			msg_number = msg_number[1:len(msg_number)]

			#read the message
			ser.write("AT+CMGR="+msg_number+'\x0d\x0a')

			l=0

			j=1
			msg=""
			line_num=1
			number=""
```

```
                    while line_num<6:
                          line = ser.readline()

                          if l==1:
                                #print "l==1 is true"
                                #print "msg is "+line
                                msg=line
                                break

                          if line[0]=='+':
                                #print "+CMGR received"
                                #print "line is "+line
                                m=0
                                n=0
                                while j < len(line):
                                      if line[j] == "+":
                                            m=1
                                      if line[j] == '\"':
                                            m=0
                                      if m==1:
                                            number=number+ str(line[j])



                                      j=j+1
                                l=1

                          line_num=line_num+1

                    number = number[3:13]

                    print "Number: "+number
                    print "Msg: "+msg

                    if number == "9651923011":
                          msg = msg[0:8]
                          os.system("wget --no-proxy
http://172.26.92.202/gas/sp/gsm_booking.php?card="+str(msg)+"\&sim="+str(number))
                          #print("wget --no-proxy
http://172.26.92.202/gas/sp/gsm_booking.php?card="+str(msg)+"\&sim="+str(number))
                          os.system("rm gsm_booking*")

                    else:
                          pointer1 = 0
                          pointer2 = 0
                          pointer3 = 0

                          user = ""
                          pin = ""
                          counter = 0
                          print "Message is: "+msg

                          while pointer3 == 0 and counter < len(msg):
                                print counter
                                if msg[counter] == ' ':
                                      if pointer2 != 0:
                                            pointer3 = counter
                                      else:
                                            pointer2 = counter
                                counter = counter+1

                          user = msg[pointer1:pointer2]
                          pin = msg[pointer2+1:pointer3]

                          os.system("wget --no-proxy
http://172.26.92.202/gas/sp/sms_booking.php?user="+str(user)+"\&pin="+str(pin)+"\&s
im="+str(number))
```

```
                            print("wget --no-proxy
http://172.26.92.202/gas/sp/sms_booking.php?user="+str(user)+"\&pin="+str(pin)+"\&s
im="+str(number))

                            os.system("rm sms_booking*")




                    #delete the message
                    ser.write("AT+CMGD="+msg_number+'\x0d\x0a')


             else:
                    print "no cmti"

ser.close()
```

C.3 IVR System Code in Asterisk and PHP

- **Dial Plan Code**

```
exten => s,1,Answer
exten => s,n,Set(TIMEOUT(absolute)=250)
exten => s,n,Set(GLOBAL(ID=0))
exten => s,n,Set(GLOBAL(PIN=0))
exten => s,n,Set(GLOBAL(IDINVALID)=0)
exten => s,n,Set(GLOBAL(PININVALID)=0)
exten => s,n,Set(GLOBAL(GASINVALID)=0)

exten => s,n,Playback(/var/lib/asterisk/sounds/welcome)
exten => s,n,Goto(100)

exten => s,100,Set(IDINVALID=$[${IDINVALID}+1])
exten => s,n,Wait(1)
exten => s,n,GotoIf($["${IDINVALID}" < "4"]?:600)
exten => s,n,Read(ID,enter_userid,4,,1,20)
exten => s,n,Playback(/var/lib/asterisk/sounds/enteredid)
exten => s,n,SayDigits(${ID})
exten => s,n,Read(ch1,toconfirm,1,,1,10)
exten => s,n,GotoIf($["${ch1}" = "0"]?600)
exten => s,n,GotoIf($["${ch1}" = "1"]?200:100)

exten => s,200,Set(PININVALID=$[${PININVALID}+1])
exten => s,n,Wait(1)
exten => s,n,GotoIf($["${PININVALID}" < "4"]?:600)
exten => s,n,Read(PIN,enter_pin,6,,1,30)
exten => s,n,Playback(/var/lib/asterisk/sounds/enteredid)
exten => s,n,SayDigits(${PIN})
exten => s,n,Read(ch2,toconfirm,1,,1,10)
exten => s,n,GotoIf($["${ch2}" = "0"]?600)
exten => s,n,GotoIf($["${ch2}" = "1"]?300:200)
exten => s,n,Goto(300)

exten => s,300,AGI(authenticate.php,${ID},${PIN})
exten => s,n,Goto(400)
exten => s,350,Playback(/var/lib/asterisk/sounds/wronginput)
exten => s,n,Goto(600)

exten => s,400,Set(GASINVALID=$[${GASINVALID}+1])
exten => s,n,GotoIf($["${GASINVALID}" < "4"]?:600)
exten => s,n,Wait(1)
exten => s,n,Playback(/var/lib/asterisk/sounds/cylinder)
exten => s,n,Read(GAS,enter_cylinder,1,,1,10)
exten => s,n,Playback(/var/lib/asterisk/sounds/enteredcylinder)
exten => s,n,SayDigits(${GAS})
exten => s,n,Read(ch3,toconfirm,1,,1,10)
exten => s,n,GotoIf($["${ch3}" = "0"]?600)
exten => s,n,GotoIf($["${ch3}" = "1"]?:400)
exten => s,n,Goto(500)

exten => s,500,AGI(book.php,${ID},${GAS})
exten => s,n,Goto(600)

exten => s,600,Playback(/var/lib/asterisk/sounds/goodbye)
exten => s,n,Hangup() ;bye bye
```

- **Authenticate code**

```
#!/usr/bin/php -q
<?php
set_time_limit(6);
require 'phpagi.php';
ob_implicit_flush(true);
error_reporting(0);

if (!defined('STDIN'))
{
        define('STDIN', fopen('php://stdin','r'));
}
if (!defined('STDOUT'))
{
        define('STDOUT', fopen('php://stdout','w'));
}
if (!defined('STDERR'))
{
        define('STDERR', fopen('php://stderr','w'));
}

$username=$argv[1];
$pass=$argv[2];

$db=mysql_connect("172.26.92.202", "vlsi", "vlsi");
if(!$db)
{
        verbose("Could not connect to DB!");
        exit(1);
}

if(!mysql_select_db("gasbooking", $db))
{
        verbose("Could not use DB!");
        exit(1);
}

$chk=0;
$qry="SELECT* FROM User_Pin WHERE User_ID='$username' AND UserPin='$pass'";
$result=mysql_query($qry);

       if($result)
         {
                if(mysql_num_rows($result) == 1)
                {
                        $chk=1;

                }
         }

if ($chk==0)
{
write("EXEC GoTO s,350");
}
mysql_close($db);
?>
```

- **Book Code**

```
#!/usr/bin/php -q
<?php
set_time_limit(6);

require 'phpagi.php';
ob_implicit_flush(true);
error_reporting(0);


if (!defined('STDIN'))
{
        define('STDIN', fopen('php://stdin','r'));
}
if (!defined('STDOUT'))
{
        define('STDOUT', fopen('php://stdout','w'));
}
if (!defined('STDERR'))
{
        define('STDERR', fopen('php://stderr','w'));
}

$username=$argv[1];
$no_of_cylinder=$argv[2];

if ($no_of_cylinder>=4)
{
        $no_of_cylinder=1;
}

$db=mysql_connect("172.26.92.202", "vlsi", "vlsi");

if(!$db)
{
        verbose("Could not connect to DB!");
        exit(1);
}

if(!mysql_select_db("gasbooking", $db))
{
        verbose("Could not use DB!");
        exit(1);
}

        fwrite(STDOUT,"STREAM FILE book \"\"\n");
        fflush(STDOUT);
        $result = trim(fgets(STDIN,4096));
        checkresult($result);

        fwrite(STDOUT,"SAY NUMBER $no_of_cylinder \"\"\n");
        fflush(STDOUT);
        $result = trim(fgets(STDIN,4096));
        checkresult($result);

        fwrite(STDOUT,"STREAM FILE book1 \"\"\n");
        fflush(STDOUT);
        $result = trim(fgets(STDIN,4096));
        checkresult($result);

$user=$username;

$conn=mysql_connect("172.26.92.202", "vlsi", "vlsi") or die(mysql_error());
mysql_select_db("gasbooking") or die(mysql_error());

date_default_timezone_set('Asia/Calcutta');

$c1=date("d-m-Y");
$b1=date("h:i:s A");
$val=$no_of_cylinder;

if($val==1)
{
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                 mysql_query($sql,$conn);
}
else if($val==2)
{
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                mysql_query($sql,$conn);
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                mysql_query($sql,$conn);
}
else if($val==3)

         {
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                mysql_query($sql,$conn);
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                mysql_query($sql,$conn);
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                mysql_query($sql,$conn);
        }

else if($val>=4)
        {
                $sql="INSERT INTO booking(Date,Time,User_ID,Method)
VALUES('$c1','$b1','$user',3)";
                mysql_query($sql,$conn);

        }

return $chk;

function checkresult($res)
{
        trim($res);
        if (preg_match('/^200/',$res))
        {
                if (! preg_match('/result=(-?\d+)/',$res,$matches))
                {
                        fwrite(STDERR,"FAIL ($RES)\n");
                        fflush(STDERR);
                        return 0;
                }
                else
                {
                        fwrite(STDERR,"PASS (".$matches[1].")\n");
                        fflush(STDERR);
                        return $matches[1];
                }
        }
        else
        {
                fwrite(STDERR,"FAIL (unexpected result '$res')\n");
                fflush(STDERR);
                return -1;
        }
}

mysql_close($db);
?>
```