CENG 491

Senior Design Project and Seminar

DIGIPOST

I2Technology

Innovative & Intelligent

TECHNOLOGIES

Poster in Blue

Initial Design Report

1 Introduction

1 1. Definition and Scope of Our Project

„PosterInBlue" is a digital poster with interactive Bluetooth. Digital poster enables its users to take the advantage of digital advertising. Besides being more attractive, digital posters will be more useful than any other advertising media. The customers will be able to show their advertisements on LCDs instead of papers. What is more, they will be able to show animating posters, which will capture the attention of their target group. By using Bluetooth technology, all these advertisements will be more permanent, since the advertised information will also be sent to the audience"s Bluetooth devices. User friendly GUI will also enable Poster in Blue customers to design their posters easily and effectively.

As we searched the web for possible application areas, we have learned that , due to rapid developments in mobile technology and expanding use of wireless communication there is an increasing demand on Bluetooth based applications. And this technology is used for many purposes. Among them we are going to implement the followings:

- Information Sharing (Digital Posters, Presentations)
- Commercials (Promotions)
- Business Cards

As we have done market research due to hardware restrictions continuous data transfer is not in our scope (ie. Music broadcast, video broadcast) Also broadcasting for large areas like city or country is not in our scope since the range of Bluetooth is max 100mt. Lastly for ethical reasons non on demand protocols are not in our scope like forced advertisement.

1 2. Design Goals and Objectives

Our design goal is to achieve a well structured system by decomposing the whole system into subsystems. We will try to decompose the system such that subsystems will be highly cohesive and low coupled. This type of design will be helpful both for us to visualize, specify and construct our project. It will also be helpful for the people concerned of our project to understand it more easily. Working on such a design will also enable us to handle unavoidable changes more easily, therefore the software will be maintainable. Besides, we will try to simplify the design of user interface in order to make it more understandable for our potential customers.

1 3. Project Features

„PosterInBlue" project is initially given as a Digital Poster project. We have added new features in order improve flexibility of our design. Basic features of our project are:

→ Sending image files as advertisements

→ Sending vCards[1]

→ Sending vCalenders[2]

Which are shown in digital posters in LCD monitors at a public area via Bluetooth.

Extra features that we are planning to implement are:

→ Sending Presentations

→ Sending Animations

[1] vCard is a file format standard for personal data interchange, specifically electronic business cards

[2] vCalendar is an electronic calendaring and scheduling exchange format

2 Design Considerations

2 1. Reliability and Interoperability

As we mentioned in our requirement analysis report, we use XSA-3S1000 [1] Board with FPGA[3] chip. We program our chip by using our Bluetooth device. About this subject, we talked to Alper Kılıç. He proposed us basically two methods. The first one we are going to try to use for our system is getting the serial data from Bluetooth and storing it bit by bit in the registers of FPGA and creating 8-N-1[4] bit streams and then process these by XESS Board. The second way is using another board which has registers to achieve this serial to parallel conversion in order to communicate two boards. By choosing one of these ways, we will be able to integrate our devices to the system.

Apart from these devices we have LCD monitor where we are going to display our poster. Our XESS Board has great range of functionalities to do this. According to our research our board is one of the best choices for LCD display.

Furthermore, there are receiver hardware devices which poster data are going to be sent by the BlueRadios kit. Every receiver device having Bluetooth communication will obviously enable us to communicate with them via standard Bluetooth Protocols. Learning and applying these protocols we are going to ensure synchronization of our data between different receiver devices.

[3] field programmable gate array, a semiconductor device containing programmable logic components and programmable interconnects

[4] 8-N-1 is a common shorthand notation for a serial port parameter setting or configuration in asynchronous mode, in which there are eight (8) data bits, no (N) parity bit, and one (1) stop bit

2 2. Portability

We consider the portability term in two ways. One of them is the portability of operating systems where we are going to program FPGA. Since XILINX-WebPACK [2] has

versions for both Windows and UNIX, we guarantee this portability. The other aspect of portability is for the hardware receiver devices such as cell phones, palms, etc. We ensure this portability by the Bluetooth protocols.

2 3. Security

Our main concern about security mainly depends on admin actions. Since 3rd party users cannot make any changes in program or program data. We assure this property by setting one way communication between our Bluetooth server and 3rd party users. Our on demand protocol about data communication also makes one way data connection.

In admin connection our focus about security is connection time. Considering safety we avoided on the fly configuration. Instead of keeping the connection alive we first make connection in order to check the Administrator Password. During this connection our Broadcasting service goes offline for a short period of time for full security. This time is approximately 0.1 second which may be seen negligible for 3rd party users. By disabling broadcasting service we are quite sure about who connects our Configuration module. After confirming admin password we process the desired operation. The other advantage of this method is off-line working. Admin may fetch data in VGA[5] display. After fetching data admin can make necessary changes without VGA shutdown. As we said earlier this improves system in principles of minimum offline time & maximum security.

Below is the FSM[6] of our security system:

[5] Video Graphics Array (VGA) is an analog computer display standard first marketed in 1987 by IBM

[6] A finite state machine (FSM) or finite state automaton (plural: *automata*) is a model of behavior composed of a finite number of states, transitions between those states, and actions.

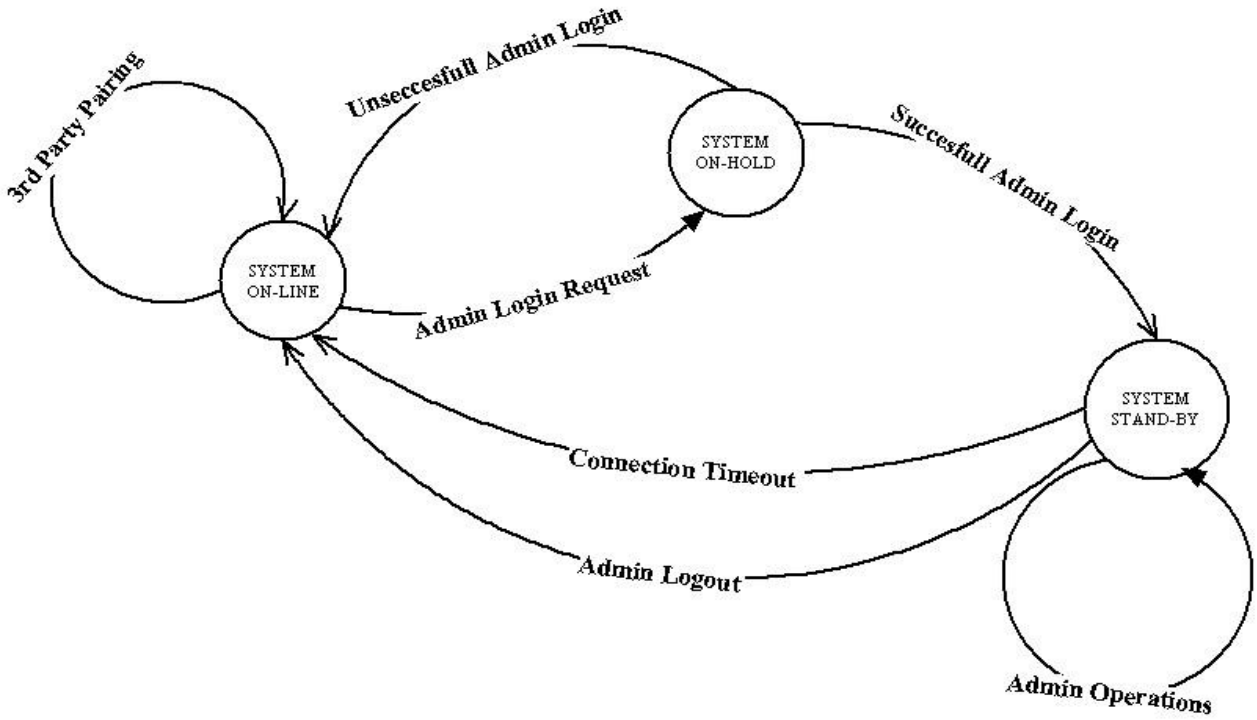


Figure 1: FSM of Security System

As expressed in diagram admin timeout is another property of our program. After certain period of time if no action was taken by admin system restores itself back to online position.

Authentication protocol between admin and Bluetooth Module is ensured by Bluetooth security protocols. We treat all devices as entrusted devices. Which means in all operations password (passkey) is required. Detailed diagram for pairing and authentication [3] can be seen below:

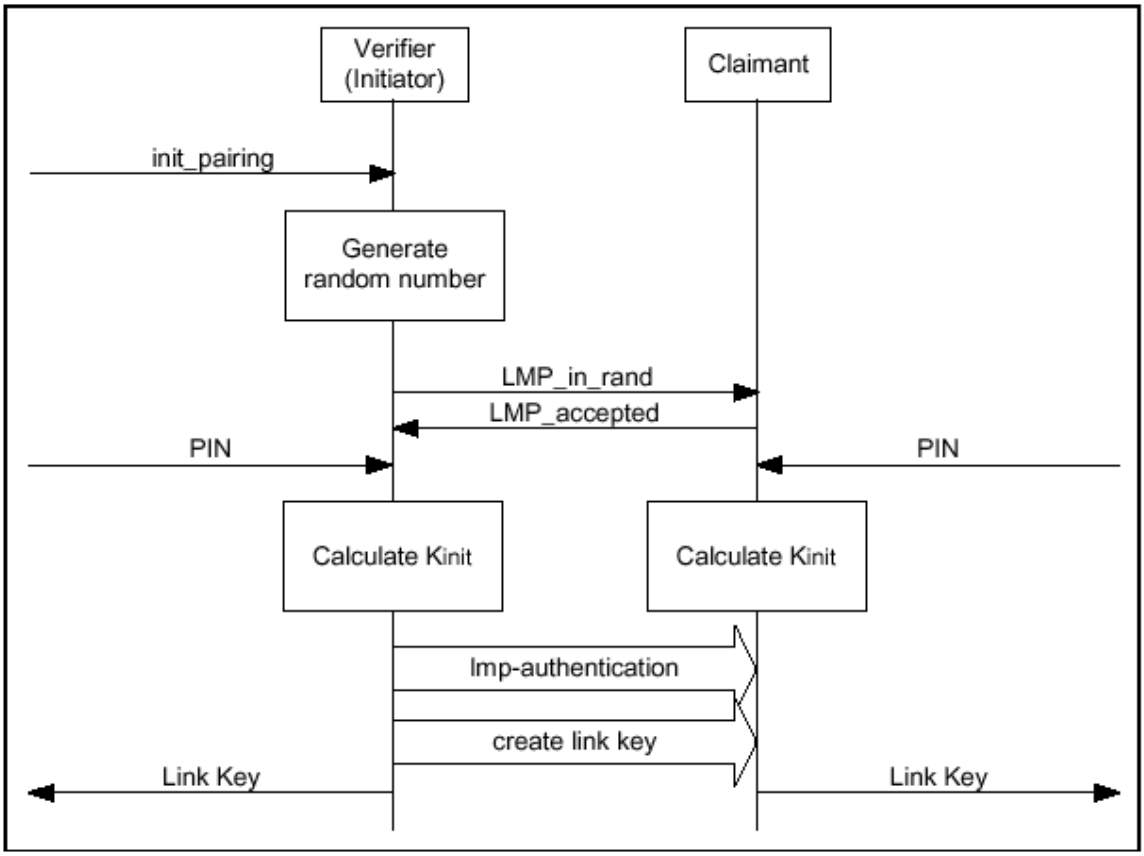


Figure 2: Pairing and Authentication

INFORMATION ABOUT LMP-Pairing: It is a procedure that authenticates two devices, based on a PIN, and subsequently creates a common link key that is used as the basis for a trusted relationship or a secure connection. This procedure consists of the steps:

- creation of an initialization key based on a random number and PIN [4]
- LMP-authentication based on the initialization key and creation of the common link key

3 System Architecture

Poster in Blue is a combined hardware/software Bluetooth solution the system of which must be designed to be extendible in the future. Besides, for this system conquering self manageable pieces will be a wise strategy to overcome the complexity of the problem. So, instead of designing the project as an "all of one piece" system which is supposed to be really massive, we decided to divide the system into modules. Designing it in this way will mainly provide us some degree of freedom that

- we will have a reduced and more specific problem space to overcome
- we can further enhance the system by extending each module
- we can even enlarge the system by adding new modules.

How we divided the system into modules? We firstly considered possible usage scenarios and in these scenarios we think about internally independent looked parts. For instance, displaying posters onto the VGA, the administration or sending-receiving data over Bluetooth. Then we further think about these parts and realized some other internal necessities of these parts. For example how to prepare the data to be sent-received over Bluetooth. As a result we decided to construct our system with these 6 modules:

1) Admin Module
2) Configuration Module
3) HCI Module
4) VGA Display
5) Service Supply Module
6) Power Supply for Boards

3 1. Admin Module

Admin module is basically the administration tool we will design for configuration of the posters, vCards and calendar events. It will have an authentication system which requires a password to discover PosterInBlue devices. After logging in Bluetooth communicator device starts a search for all BlueRadios devices in range more specifically our „PosterInBlue" devices.

We think that there may be more than one PosterInBlue in an area such as subways and our admin can discover all devices in this area. Then he may click on one of the discovered PosterInBlues and our GUI will request the license key of this device. This is done for two reasons. First one is security concerns; since only the owner of this PosterInBlue can see the contents and configure it. Second one is we decide that while configuring a device nothing will be shown on LCD, in other words LCD is locked during configuration. When configuration is finished and load new configuration command is issued by the admin new configuration is loaded into our FPGA and our LCD starts to display new posters with new time intervals as specified in our configuration.

We may not provide multiple device architecture in our project since we have only one (even a half) BlueRadios but we think security and handling of multiple devices is a very crucial issue if we can extend this project for commercial use in the future.

After selecting and getting an authorization from a specific PosterInBlue admin can explore the contents of this device. For instance he may see 4 posters are there in this device. Then select one of them to explore its details and attributes. Every poster object should include an image file and it"s attributes at least.

Apart from that some posters may contain vCards or calendar events in it. Clicking on a specific poster our admin will see two rollouts which are attributes of this poster. One of them is vCard and one of them is calendar event associated with this poster. These two may be configured before or not configured. Admin can reconfigure, add or delete them.

We will also provide the user the ability to delete existing poster or adding a new poster. Duration of the poster to be shown in LCD attributes of it and if it will have associated vCard or calendar event should be provided by admin when adding a new poster object.

After completing all configurations admin will press „load new configuration" button and admin module completes its work here.

3 2. Configuration Module

Configuration module basically configures the overall behavior of the system. To some extend it is a circuit logic which resides on the FPGA, keeps the internal structure stable and consistent. Configuration module mostly maintains the poster data, ensures its coherence. Since this module is going to be a circuit stored on the XESS board"s programmable array here we need to show how a circuit can be designed on a Spartan 3 FPGA.

Setting up a working circuit design

To achieve logic design for the Spartan FPGA of our XESS board, XILINX currently provides the WebPACK tools. To prepare design we use the XILINX ISE 8.2i [5] tool. Here we need to mention our VGA display example design. Firstly, we insert VHDL [6] modules to the project. "vga.vhd" is the vhdl file describes the VGA generator circuit. For the SDRAM control the state machine will be designed in "sdram cntl.vhd". Customizing the SDRAM controller is done via "xsasdram cntl.vhd". Some dummy functions and definitions in "common.vhd". And at last to create the overall testing application "test_vga.vhd" combines the SDRAM controller and the VGA generator. Apart from vhdl files, to handle VGA generator"s pin assignments we defined "test_vga.ucf" file. At last we have generated the bit stream file by using "Generate Programming File" process on the ISE. „test_vga.vhd" file is given in the Appendix B.

Beside from the bit stream file which is our actual logic we of course need the image data to display. In this example design, we convert a usual JPG file (I2Tech logo) to a special .xes format. We achieve this by means of a Perl [7] script which converts the graphic file into this xes format that works with the VGA generator circuit.

As a result we have the bit stream file that implements the logic on the board array and the xes file that keeps the image data. Loading these files on the board is handled by the XSTOOLS, *see figure 3*. GXSLOAD is the tool by which the bit stream and xes files are uploaded to the FPGA board.

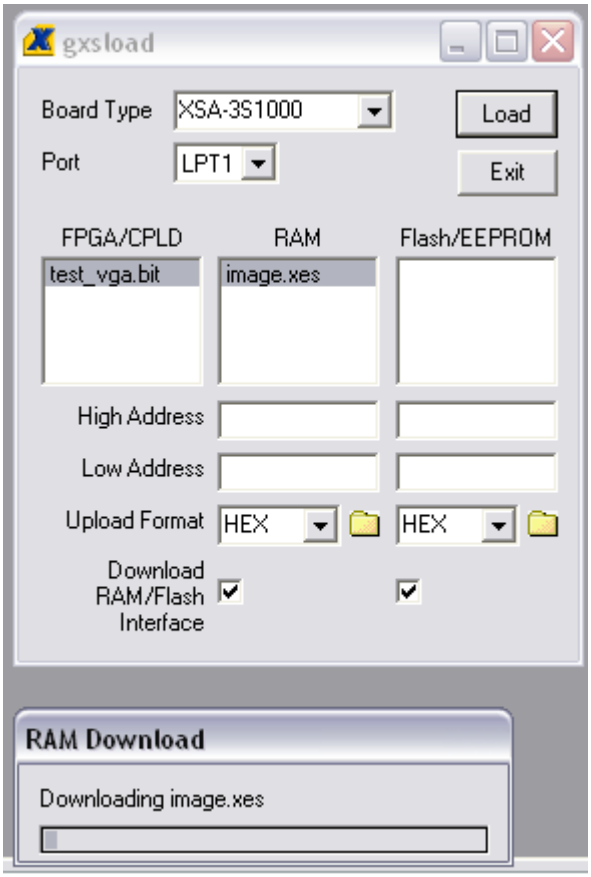


Figure 3: Using GXSLOAD

3 3. HCI Module

Bluetooth is not just a radio system but also a software specification to handle devices from different manufacturers. This specification defines header and packet formats, error checking and some other functions. Each layer in the stack is strictly defined and maintained by the Bluetooth Special Interest Group (SIG)[7]. The protocol stack consists of eight layers, *see figure 4*. For HCI[8] module we are not going to give detailed information of each layer and except from HCI layer we briefly mention some layers.

RFCOMM: Provides a RS232 interface

L2CAP: Multiplexes data from higher layers

Link Manager: Controls and manages links to other devices

Radio: The physical communication

HCI module will be the interface between the Bluetooth device and the host device running the software on which the rest of the stack will reside. In our case this host device will be our FPGA board.

Data and commands will be sent by using 4 types of packets: Command packets, event packets and ACL[9] packets. Despite not directly being a part of HCI module ACL is just a link type that HCI module will use to communicate with another device. HCI module will use command packets with which it can control the behavior of local and remote link managers. All results are returned in event packets which return replies and data. We will also use these packets to inform the host, FPGA board, when events occur in the Bluetooth module like termination of a connection. When a connection established between HCI module and another device, an ACL link automatically set up and L2CAP on the service supply module and the configuration module can use HCI for communication.

---

[7] Bluetooth Special Interest Group is a group of companies who promotes and defines Bluetooth specifications.The use of the Bluetooth is limited to the companies that joined the SIG

[8] Host Controller Interface

[9] Asynchronous Connection-Less and transport time insensitive data

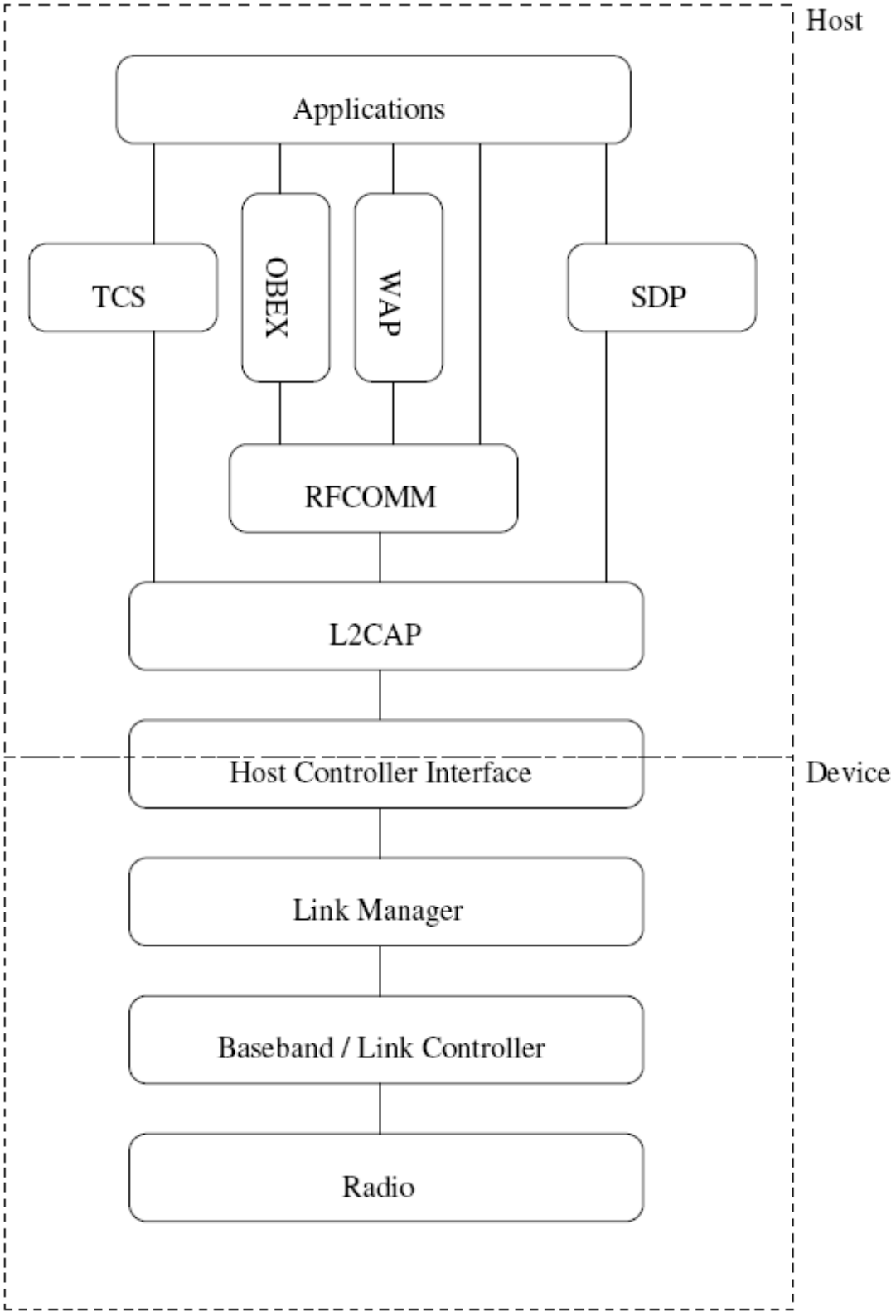


Figure 4: Bluetooth protocol stack

3 4. VGA Display Module

Video Display module deals with the video subsystem of the XESS Board. The main responsibility of this module is to display the poster on the screen. One of the concerns about this module is the amount of memory required by the video buffer. An 8 bits/pixels color depth for RGB[10] together with a 800x600 display size comes up to a 480 kilobyte video buffer which is really small amount of a 32 MB SRAM.

The other constraint is permanent storage of posters. Despite having a significant amount of (2Mbyte) flash memory, VGA display module may need to keep poster data in a somehow compressed way and decompress it for video buffer. Achieving this will need a further and detailed research on image compressing/decompressing algorithms and designing circuits on programmable array.

For an initial attempt on VGA module, we have tried a VGA display circuit designed for our XESS Board. This design example is a simple circuit that displays an image stored in the SDRAM of our FPGA board. Some features of this VGA generator are: flexible timing for the horizontal and vertical sync signals, adjustable width for the red/green/blue output signals, 255-word pixel buffer for storing pixels and selectable pixel width so that each word of memory can hold 1, 2, 4, 8 or 16 pixels. For this VGA display module experiment, we used a Perl script to convert our logo in JPG format into a format that works with the example circuit.

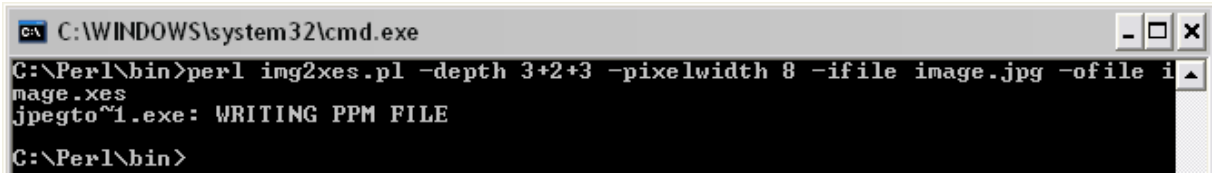


Figure 5: Using Perl to convert image files

We do not mention details of this circuit design at this part of the report. The vhdl file of the design is shown in Appendix B.

[10] The RGB color model is an additive model in which red, green and blue (often used in additive light models) are combined in various ways to reproduce other colors

We programmed the board's programmable array and displayed our logo in a 800x600 display size.

Figure 6: Sending our group logo

3 5. Service Supply Module

This module concerns with providing various services to end users. In other words, this is the module in which communication with customers Bluetooth devices are held and interactively transferring the data on customers' demand.

Service Supply Module should communicate with HCI module which in turn communicates with VGA Display Module. By this way we will maintain synchronization of posters in LCD and the posters we advertise.

To achieve our goal in this module we should use some of Bluetooth Profiles. Bluetooth Profiles are defined by Bluetooth specification as subsets. Profiles are a way to categorize different types of applications. This simplifies the development of Bluetooth products. For example in our project we are not dealing with audio and we will never have to know what „Headset profile" is. Profiles makes the communication between hardware and the Bluetooth software less complex since they are customized for different types of applications.

According to our previous research on this issue we decided that these are the profiles that we should use: File Transfer Profile (FTP), Generic Access Profile (GAP), Object Exchange (OBEX), Object Push Profile (OPP) and Synchronization Profile (SYNC). Since our BlueRadios kit supports "SDP, SPP, DUN, LAP, GAP, RFCOMM, L2CAP, Headset, Audio Gateway, FTP Client, OBEX, OPP - Object Push/Pull" we have no concerns on reliability of these profiles.
Below is a short description of each profile and how we will make use of them in our design:

OBEX is a transfer protocol that defines data objects and a communication protocol two devices can use to exchange those objects. OBEX enables applications to work over the *Bluetooth* protocol stack. For *Bluetooth* enabled devices, only connection-oriented OBEX is supported. Three application profiles have been developed using OBEX which include FTP, OPP and SYNC. We will use all these three application profiles.

FTP defines how folders and files on a server device can be browsed by a client device. FTP will be useful for us in sending presentations and selection of posters since it provides getting folder listings, changing to different folders and getting files.

OPP focuses on a narrow range of object formats to maximize interoperability. The most common acceptable format is the vCard and since our project scope includes sending vCards we should use this profile.

The SYNC profile is used in conjunction with GOEP to enable synchronization of PIM[11] items such as calendar and address information. A common application of this profile is the exchange of data between a PDA and computer. This will be useful for us to have synchronization in sending calendar events and personal information to different types of receiver devices.

GAP ensures the interoperability of any two Bluetooth devices, regardless of manufacturer and application. Bluetooth enabled devices not conforming to any other Bluetooth profile must conform to GAP to ensure basic interoperability and co-existence. It means we should use it in our project since all devices conform to at least GAP. We will use this since our aim is to provide a generic application, independent from brand, model and manufacturer.

3 6. Power Supply for Boards

One of the main problems of this design is that our XSA-3S1000, *see figure 6,* has only 2Mbyte Flash. We have a large SDRAM (32MB) and this is very suitable for getting high performance from our design. However, this RAM is not a stable storage device and in case of any power cut we will loose the data in our SDRAM. Due to our initial research about how we can solve this problem we have discovered that we may design a new energy supply board which supplies power to both our XSA and BlueRadios. We will do further research on this issue till our final design and try to find if there is any battery that we can use in order to solve this problem.

Supplying continuous energy by such a system to our two boards will not only solve our data loss problem but also it enriches the portability of whole design in a way that we can put our XES board, BlueRadios and this new battery or board into a one closed box that customer only needs to deal with a single plug and VGA Port.

[11] A personal information manager (PIM) is a type of application software that functions as a personal organizer.

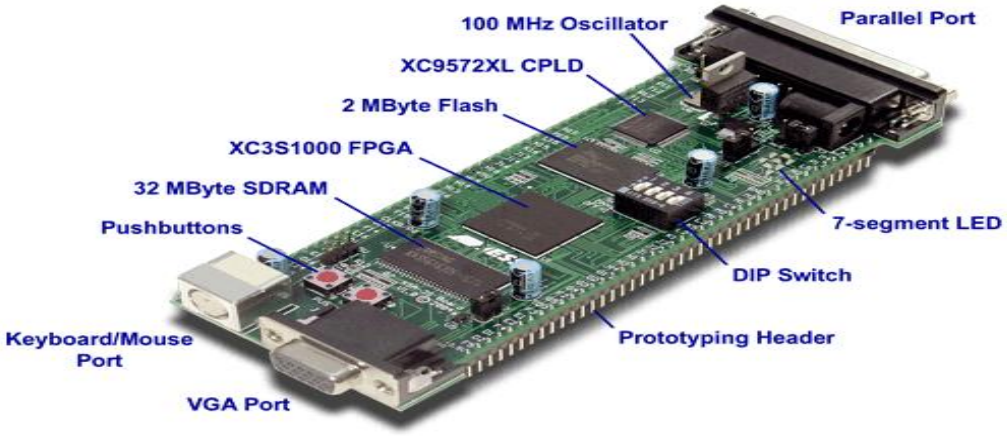


Figure 7: XSA-3S1000 Board

3 7. Module Interactions

Displaying posters:

Admin module is responsible to prepare the poster in a specific format. It also needs to prepare the required software stack to communicate with the HCI module. HCI module is the interface that Admin and Configuration modules communicate on. Configuration module also prepares the stack, forms the internal representation of the poster data coming from the HCI module. Now VGA module can handle the presentation.

Posting poster data to 3rd party users:

End users' Bluetooth devices communicate with HCI module. As usual HCI is the interface and Service Supply module use this interface to communicate with the end users. Service Supply module gives the poster, vCard and Calendar services on this virtual connection with the end users.

Making PIB configuration:

Configuration module maintains the current PIB settings and posts these settings to the Admin module via the virtual connection supplied by the HCI module. Admin module converts these settings into a representative format. With the help of the graphical user interface (GUI) a supposed administrator makes a new configuration, Bluetooth software on the admin module post the new configuration over the HCI module. Configuration module now gets the configuration over the HCI module and reconstructs the internal configuration.

4 Detailed System Design

4 1. GUI

We have a GUI design for admin user to do the necessary operations such as loading posters or configuring the poster settings. For this phase we have designed some of them.

Admin tool has an authentication system. He can authenticate the system by typing his password. After a valid authentication administrator tool searches, *see figure 8*, the near PIBs. It lists them on the "poster in blue configuration console". The list appears on the left side of the console. The user admin can refresh, select and remove PIB or clear the PIB list by using the buttons on the left below. Also he can load new configurations or reset the configurations of the selected PIB device.

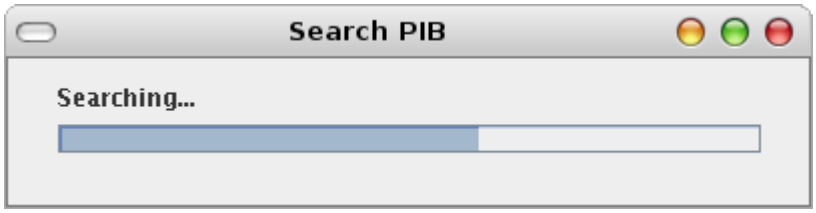


Figure 8: Searching PIBs in the range

From the GUI, *see figure 9*, admin can configure the VCard and calendar properties of the supposed poster. Admin can enter the information such as name, phone number, e-mail, web address, title, company name, address, birthday and general info for VCards. All of these fields do not need to be filled and of course it is optional to prepare a VCard for a poster. Corresponding textboxes and check box on the "VCard Properties" pane implements this capability. Also to configure the calendar properties there are name, subject, start time, end time, start date, end date, location areas.

As in the VCard field, it is optional to fill the text box areas and to prepare calendar properties. After the admin enters the information he can use the Apply and Reset buttons to save them.

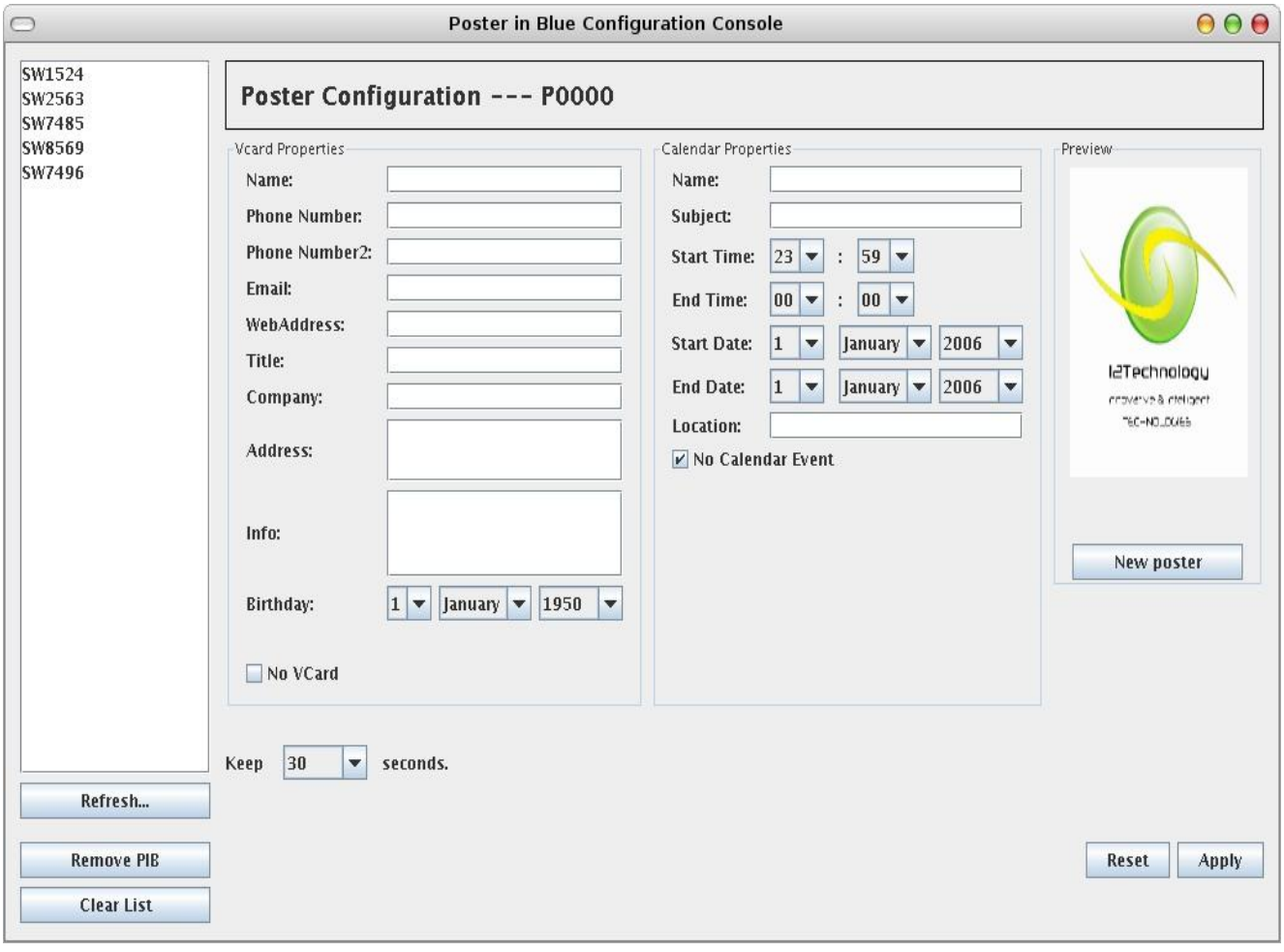


Figure 9: GUI for Poster Configuration

Also admin can preview the poster image on the right side of the GUI, *see figure 10*. He can add new poster by navigating administrator's local computer file system. From the dropdown menu on the below middle-left side on the GUI, he can decide on how many seconds the poster will stay on the VGA screen. After all the poster configurations, admin can load these configurations to the selected PIB by clicking the "Load New Configuration" button.

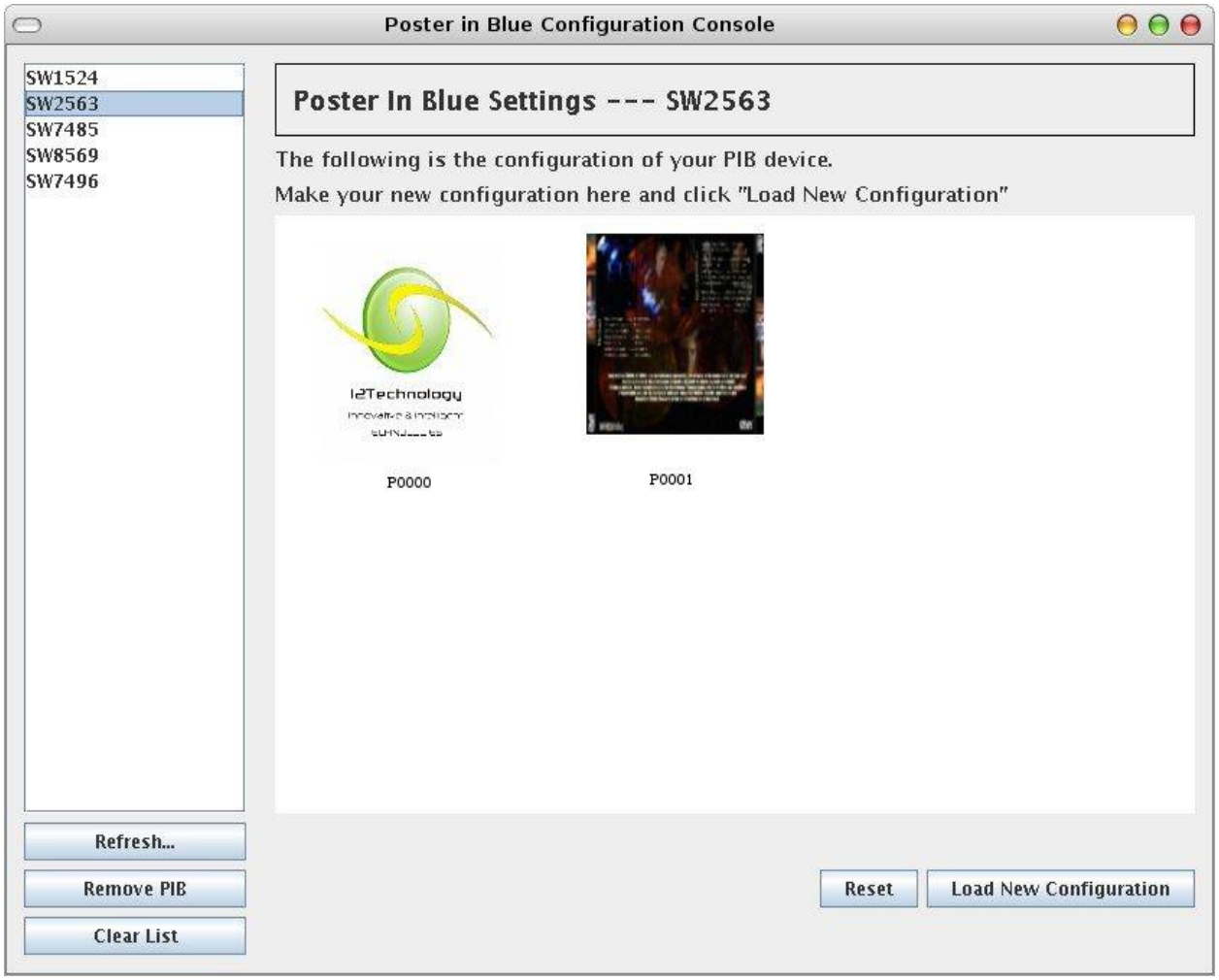


Figure 10: GUI for Poster Selection

4 2. Class Diagram

Although we will not use an object oriented language and we do not have actual classes, we decided that decomposing our design into classes will be much easier to understand. Our classes are explained below.

Classes:

Admin: This class holds the information about the administrator.

id:	Administrator's login id for system
password :	Administrator's login password for system
getCurrentConfiguration :	Administrator can learn the current configuration of system
sendNewConfiguration :	Administrator can change the configuration
startApplication :	Administrator can start displaying poster and sending data
stopApplication :	Administrator can stop displaying poster and sending data
login :	Administrator can login to system with his id and password

Configuration : This class holds the configuration of the displayed posters.

presentTime:	This array is used for how long each poster is displayed
presentQueue:	This array is used for when each poster is displayed
posterIds:	This array is used for which posters are being displayed

FPGABoard: This class is our XSA-3S1000 board's programmable part.

display:	FPGABoard can display posters using current configuration
preparePosterDataToBeSent:	When a new configuration is loaded new poster and its data is prepared
prepareConfigurationDataToBeSent:	When administrator requests current configuration, FPGA calls this function
processConfigurationData:	When new configuration is loaded FPGA processes this configuration to decide when to show which poster

setConfigurationData: When Admin changes the current configuration, new configuration is set

sendConfiguration: When administrator requests current configuration, FPGA sends it after preparing configuration data to be sent

Communication: This class supplies communication between FPGA Board, Bluetooth board , Administrator and end users. Bluetooth Receiver Controller and Bluetooth Sender Controller are parts of Communication class and they use Profiles. Bluetooth Sender Controller class is responsible for sending data to End Users and Bluetooth Receiver Controller is responsible for sending data from Administrator to the board when he wants to update the configuration.

End User: End users get the data transmitted by Poster in Blue. They can getPosterData or viewPosterData by their Bluetooth devices.

Poster: One or more posters forms a poster (since more than one poster can be shown). Poster uses poster data.

id: Size of the poster

name: Name of the poster

file: Name of the file the poster is using for display

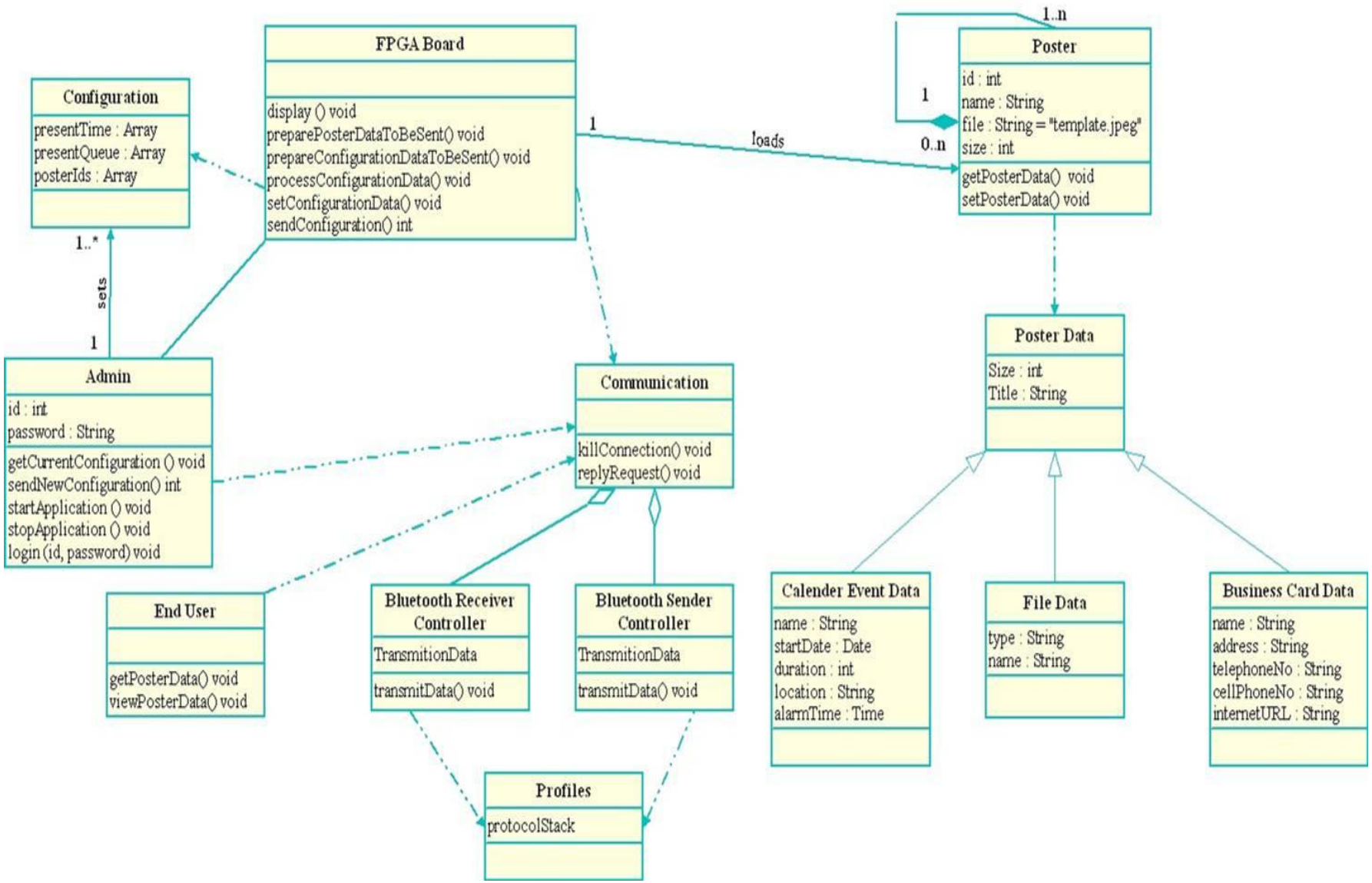
size: Size of the poster

getPosterData : Gets the data that will be sent while this poster is being displayed

setPosterData: Sets the data that will be sent while this poster is being displayed

Poster Data: Poster data can be in 3 forms. It can be a file, a calendar event or a business card.

Below is our Class Diagram:


FPGA Board
display () void
preparePosterDataToBeSent() void
prepareConfigurationDataToBeSent() void
processConfigurationData() void
setConfigurationData() void
sendConfiguration() int
1
loads
1..n
1
0..n
Poster
id : int
name : String
file : String = "template.jpeg"
size : int
getPosterData() void
setPosterData() void
Configuration
presentTime : Array
presentQueue : Array
posterIds : Array
1..*
sets
1
Admin
id : int
password : String
getCurrentConfiguration () void
sendNewConfiguration() int
startApplication () void
stopApplication () void
login (id, password) void
Communication
killConnection() void
replyRequest() void
Poster Data
Size : int
Title : String
End User
getPosterData() void
viewPosterData() void
Bluetooth Receiver Controller
TransmitionData
transmitData() void
Bluetooth Sender Controller
TransmitionData
transmitData() void
Calender Event Data
name : String
startDate : Date
duration : int
location : String
alarmTime : Time
File Data
type : String
name : String
Business Card Data
name : String
address : String
telephoneNo : String
cellPhoneNo : String
internetURL : String
Profiles
protocolStack

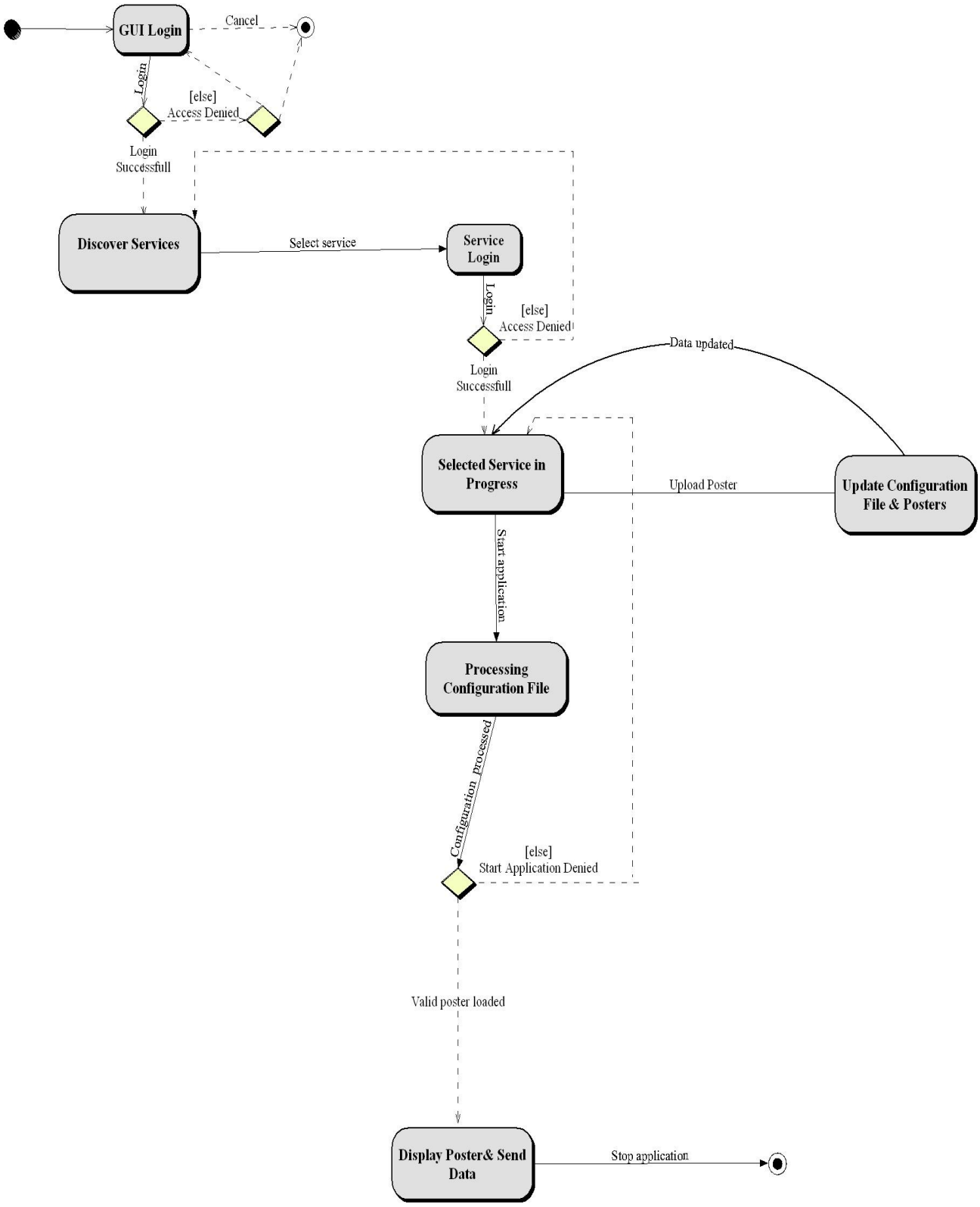
4 3. Revised State Transition Diagram

Since we have realized we should consider security again and provide a two level security system for handling multiple posters we revised our state diagram and reflected these changes to our state diagram accordingly.

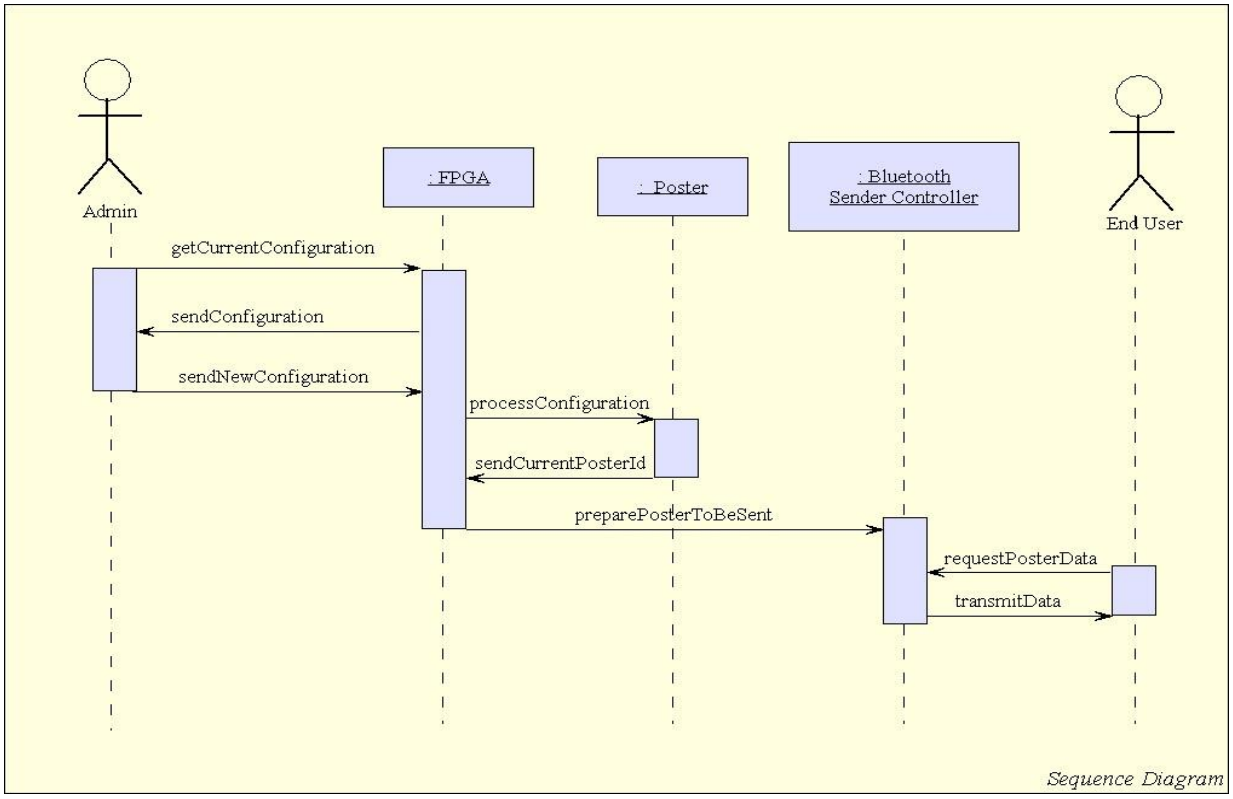
In our revised state transition diagram first state is *GUI Login* for administrator. If not authenticated system goes back to login state. If password is verified and login is successful we go to *Discover Services* state in which our system searches and finds the available „PosterInBlues" in the range. From these available PIBs our admin selects one of them which is owned by him. Since he owns this PIB he knows its password and he will be asked to enter this.

After getting authentication from this PIB he can fetch the configuration data, update it and upload to board. This new configuration is processed when the user starts the application, in other words when he clicks the „Load new Configuration" button. According to result of the process user either gets an error message indicating the possible error of configuration information or poster data is successfully sent and displayed on LCD.

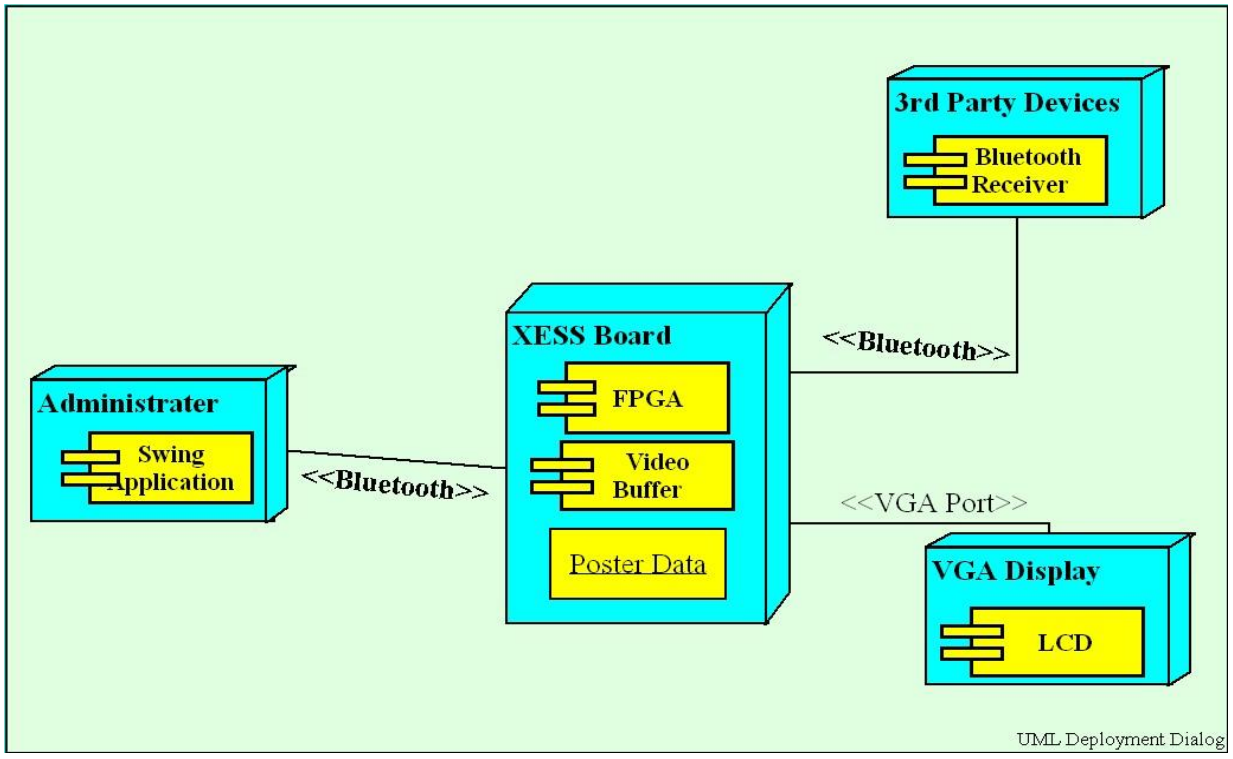
Below is our revised state transition diagram:


GUI Login
Cancel
Login
[else]
Access Denied
Login
Successfull
Discover Services
Select service
Service
Login
Login
[else]
Access Denied
Login
Successfull
Data updated
Selected Service in
Progress
Upload Poster
Update Configuration
File & Posters
Start application
Processing
Configuration File
Configuration processed
[else]
Start Application Denied
Valid poster loaded
Display Poster& Send
Data
Stop application

4 4. Sequence Diagram



4 5. Deployment Diagram

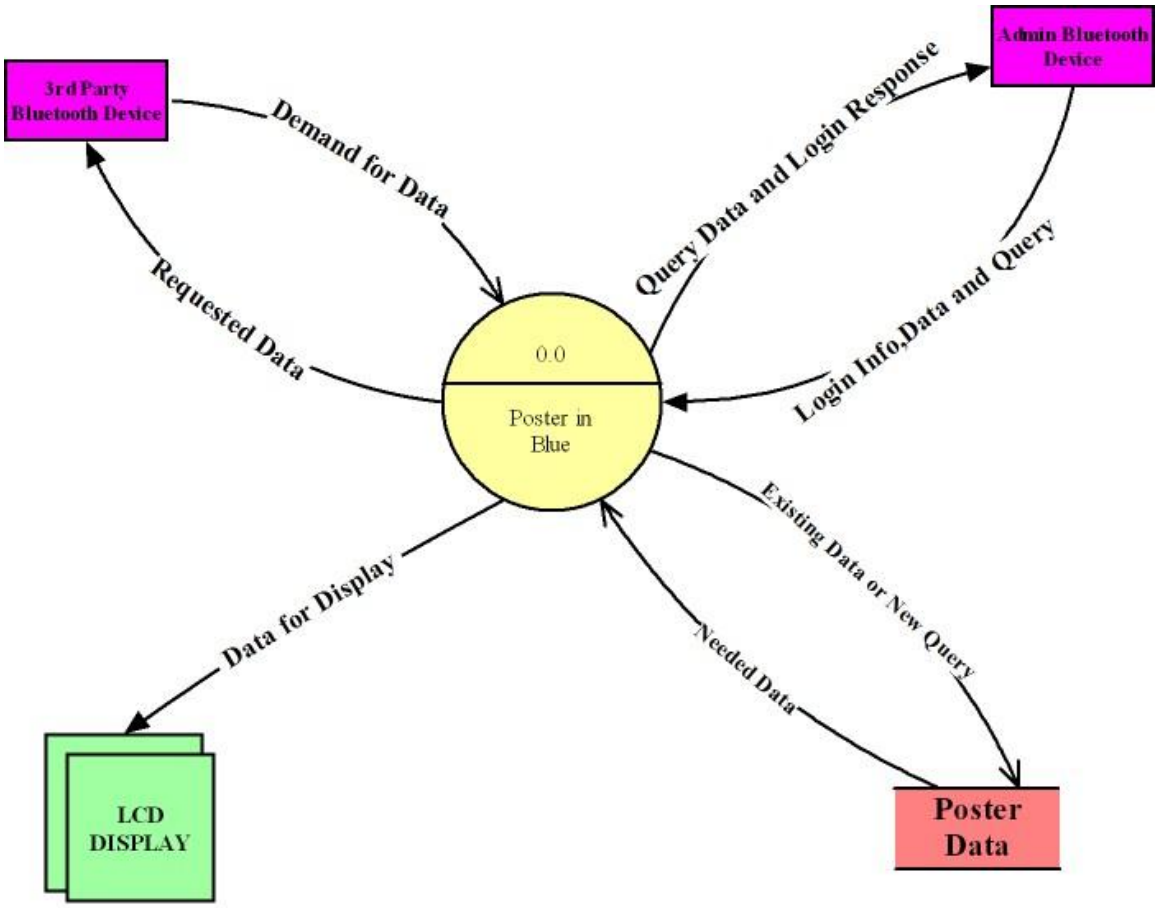


5 Data Modeling

For Data Dictionary of DFDs see APPENDIX C

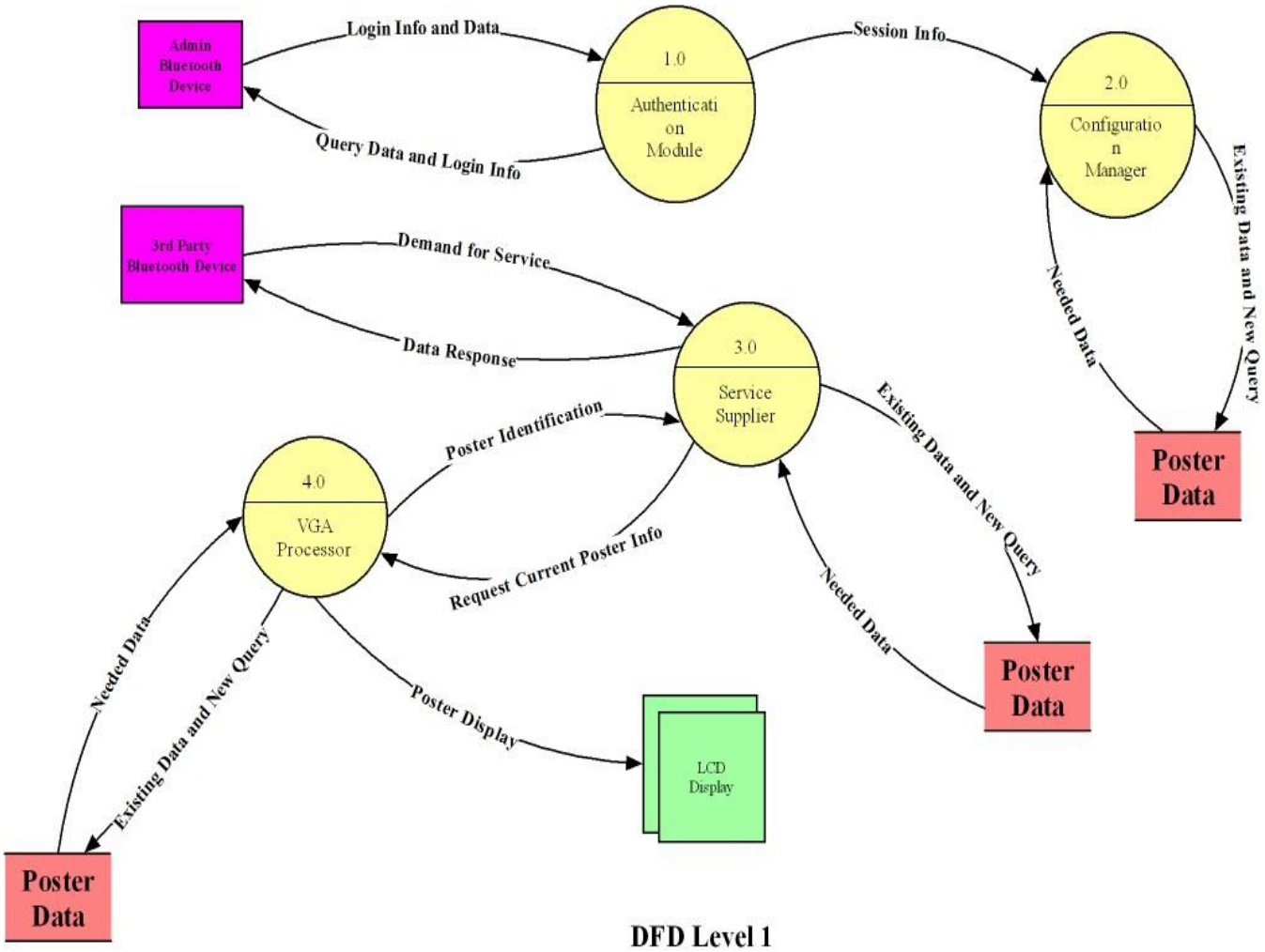
5 1. Revised Data Flow Diagrams

5.5.1 DFD Level 0



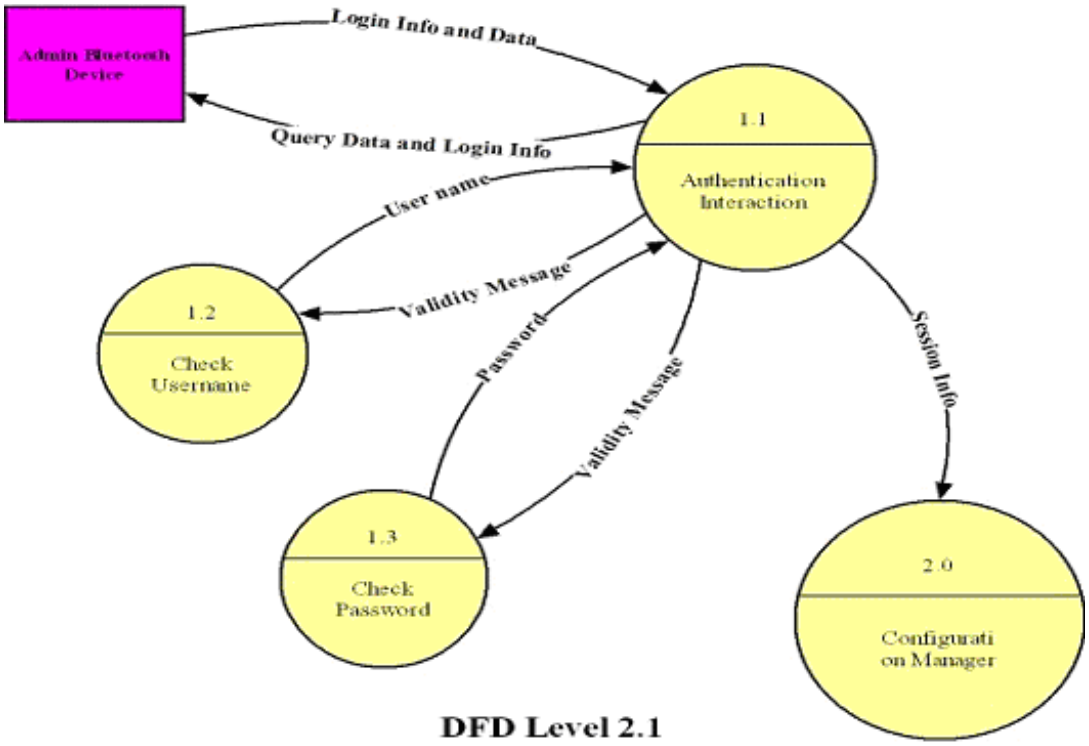
LEVEL 0 DFD

5.5.2 DFD Level 1

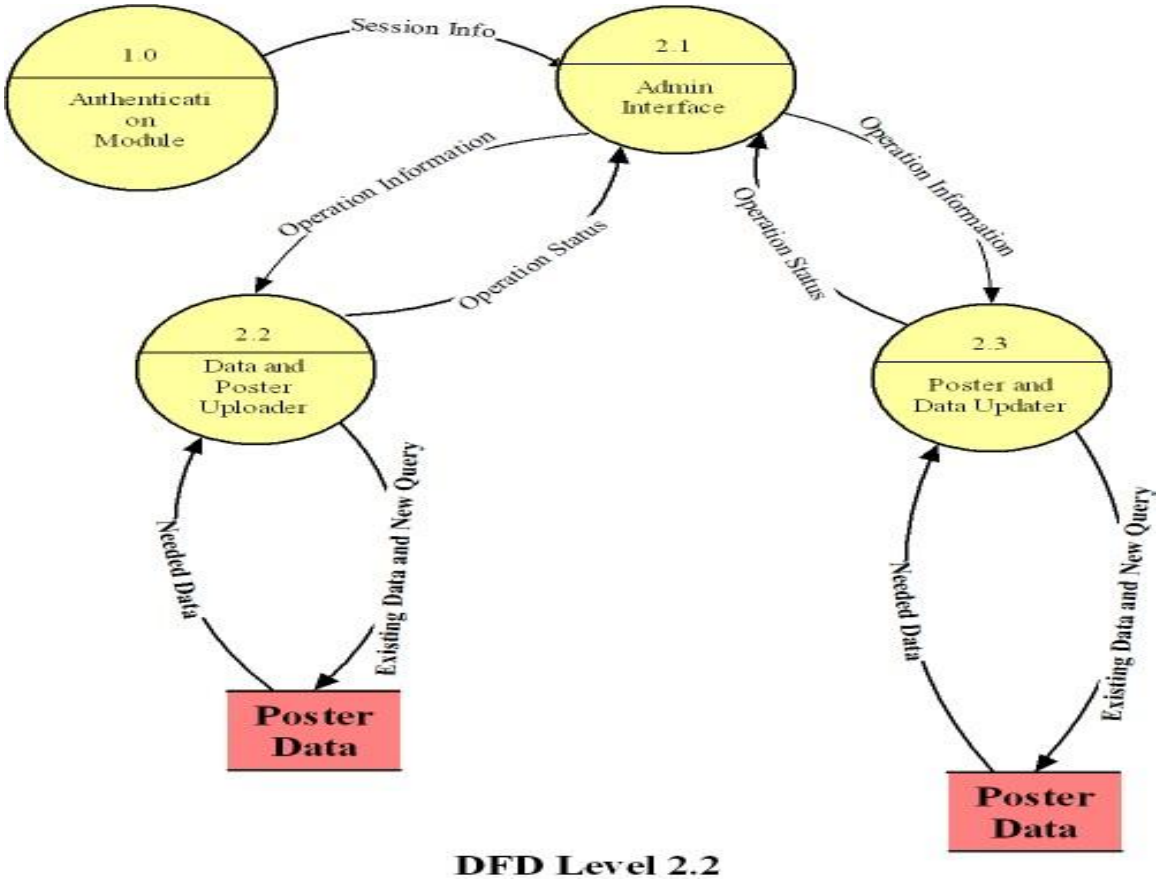


DFD Level 1

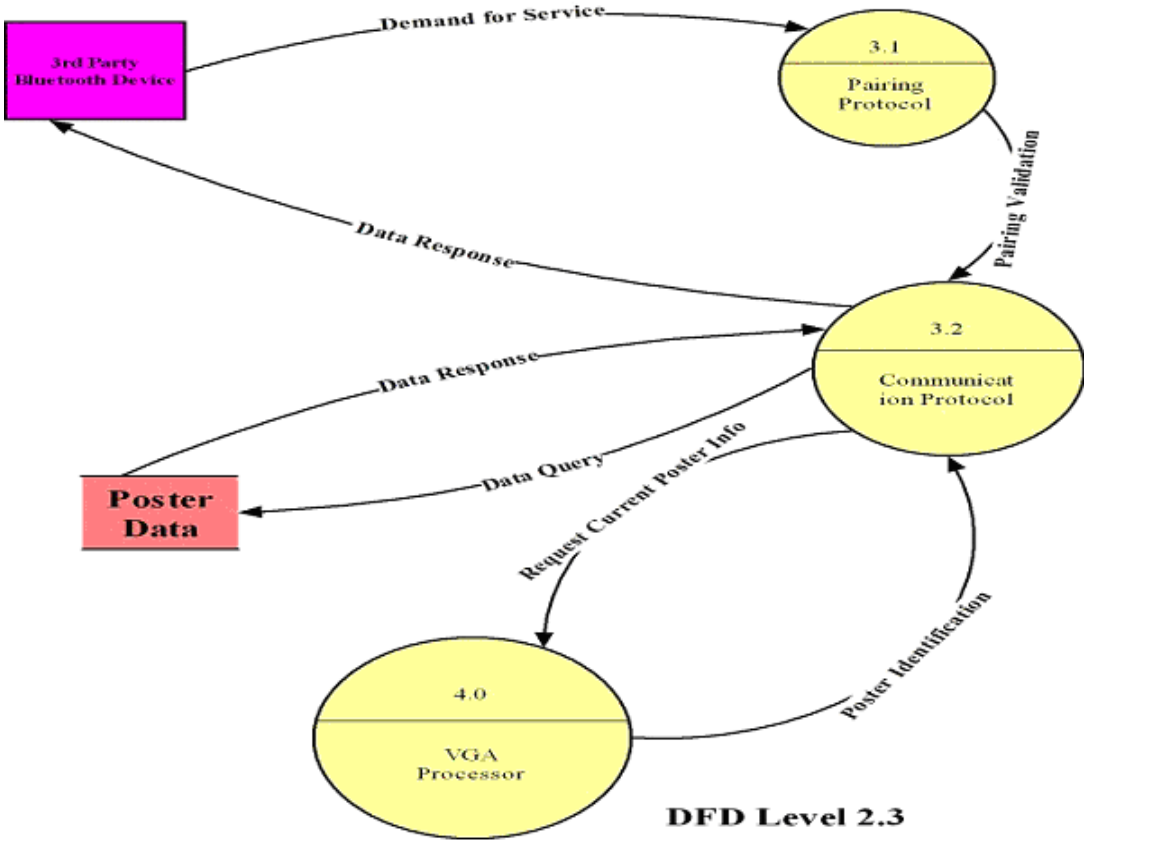
5.5.3 Level 2 DFDs



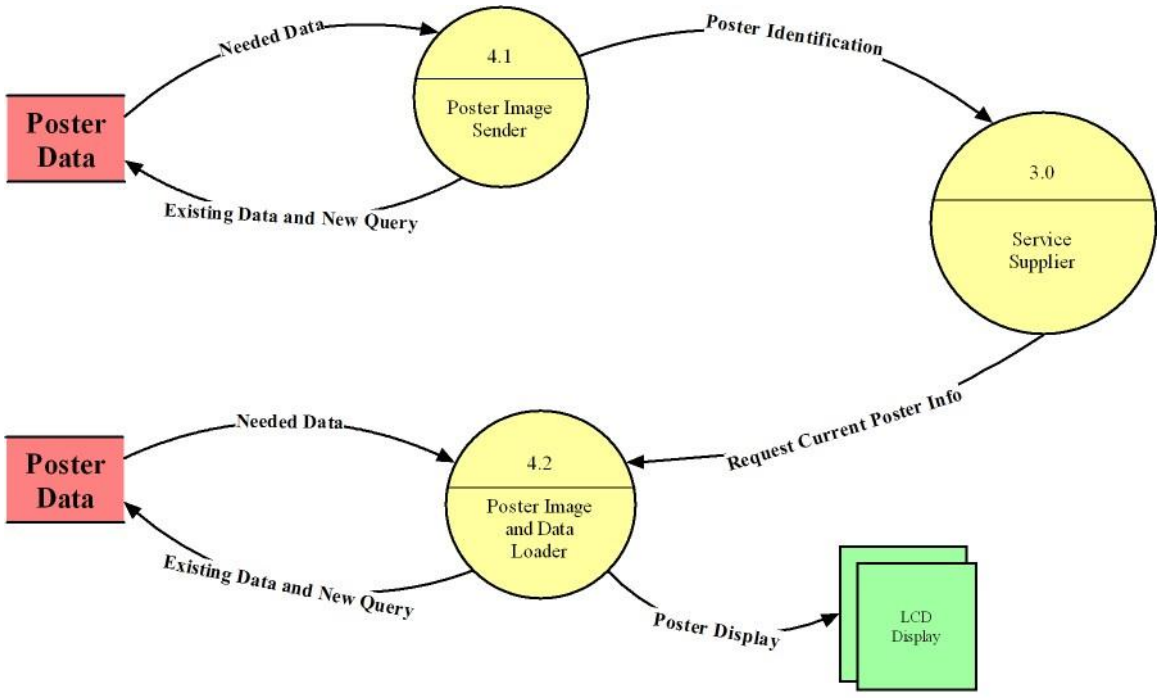
DFD Level 2.1



DFD Level 2.2



DFD Level 2.3



DFD Level 2.4

5 3. Revised Use Cases

Basically we have three major users: admins, server and clients.

Admins are the users who buy our product and display their posters and send their information to the clients via Poster in Blue. Clients are the end users; having a Bluetooth device such as cell phones, PDAs, Palms or laptops and taking the advantage of data sent via Poster in Blue by accepting the request. Finally server is the system that supplies LCD display and data transfer.

Admin Use Cases:

1- Firstly Admins must log in and get authentication to be able to do all Admin activities, *see figure 11.*
2- Admins can start Poster in Blue Application to enable data transmission and LCD Display or they can start Poster in Blue Application and only show the poster in LCD Display, *see figure 12.*
3- Admins can upload posters and upload the data they are sending via Bluetooth. For this purpose they must update the configuration file first. In the configuration file, information about all posters that will be displayed and the data that will be sent via Bluetooth and their display/send durations will be held (Since more than 1 poster can be shown in the LCD in turn, system must know when to show which poster and when to send which data).
   While uploading posters Admins can use their own posters or template posters. While uploading information Admins can only change the information that is being sent or may choose to send other kind of data such as Calendar Events, Files or Business Cards, *see figure 13.*
4- Admins can close Poster in Blue Application in two ways. They can either close the whole application or they can only stop the Bluetooth transmission. Because they may sometimes want not to send the information that is being displayed on LCD, *see figure 14*.
5- Admins can get the current configuration from the server, *see figure 15.*

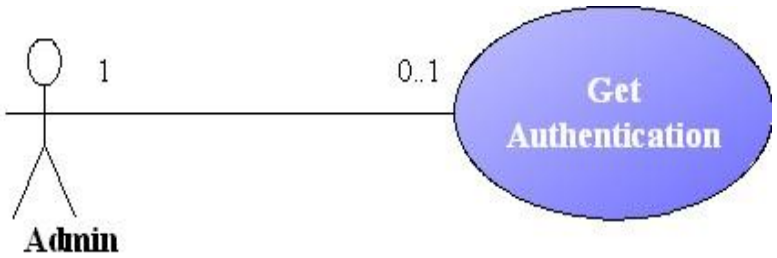


Figure 11: Admin Use Case1

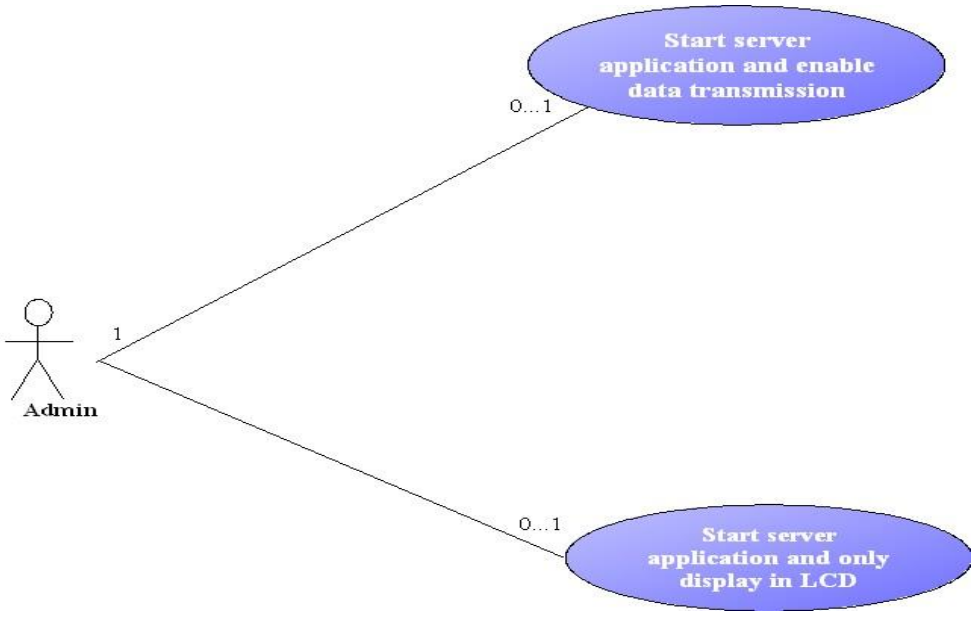


Figure 12 : Admin Use Case 2

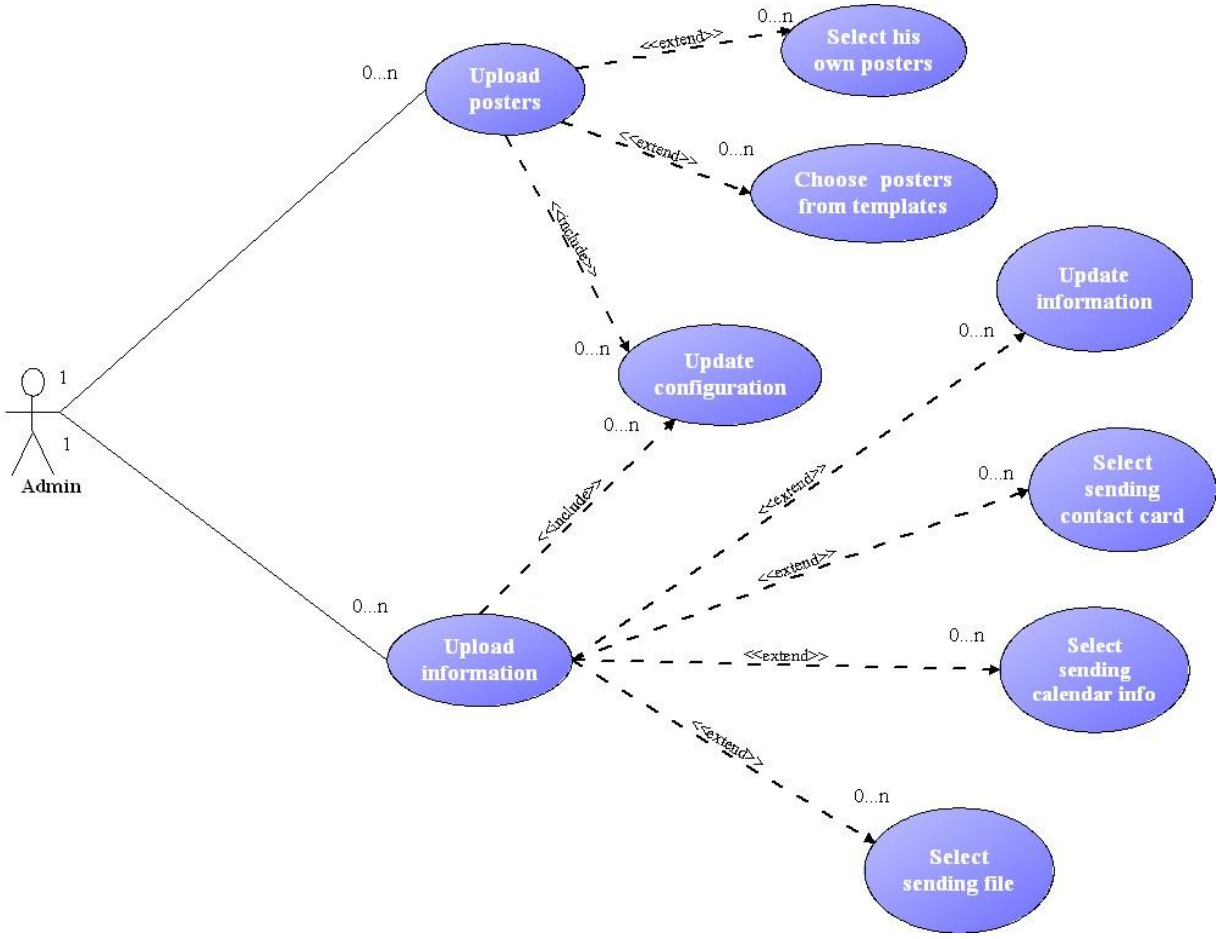


Figure13: Admin Use Case 3

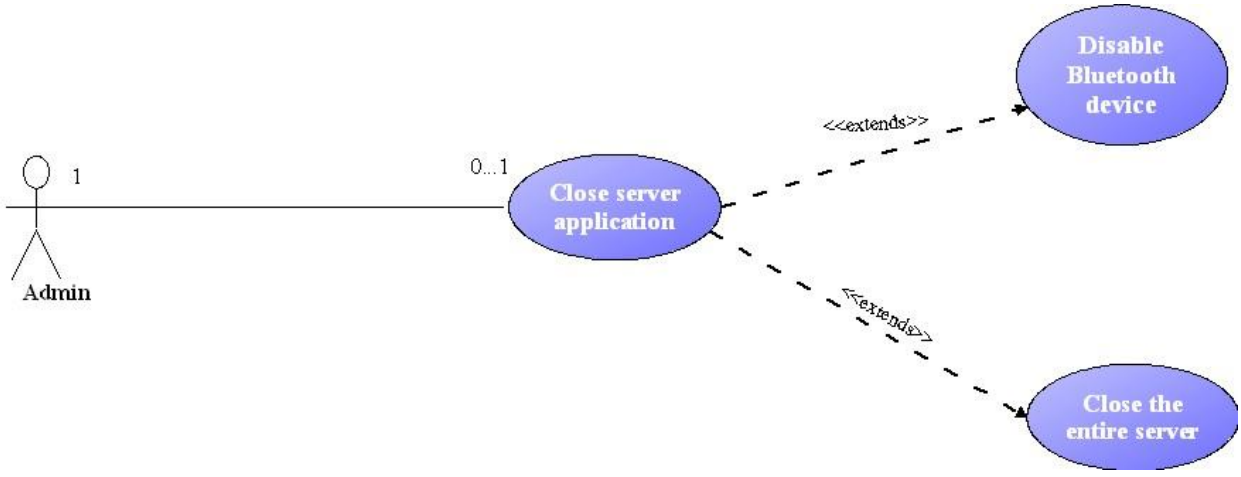


Figure14: Admin Use Case 4

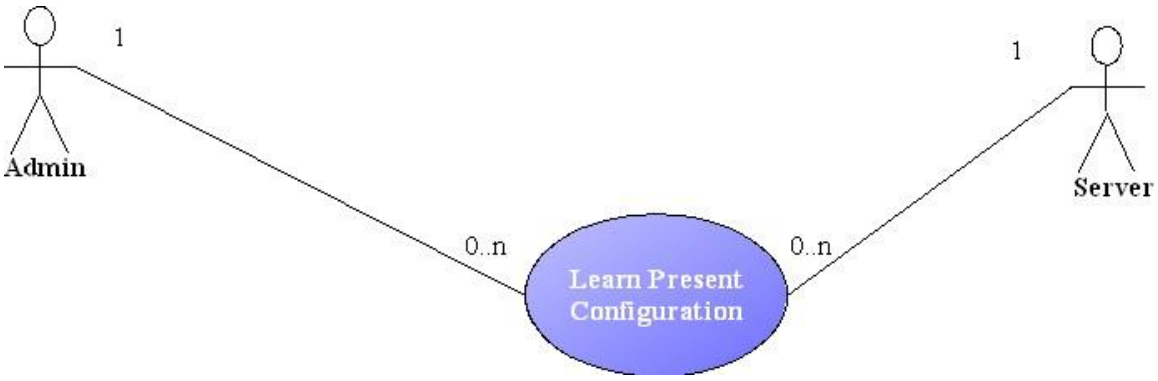


Figure 15: Admin Use Case 5

Client Use Case:

They accept or reject the data sent by Poster in Blue Application after discovering the servers in their range, *see figure 16.*

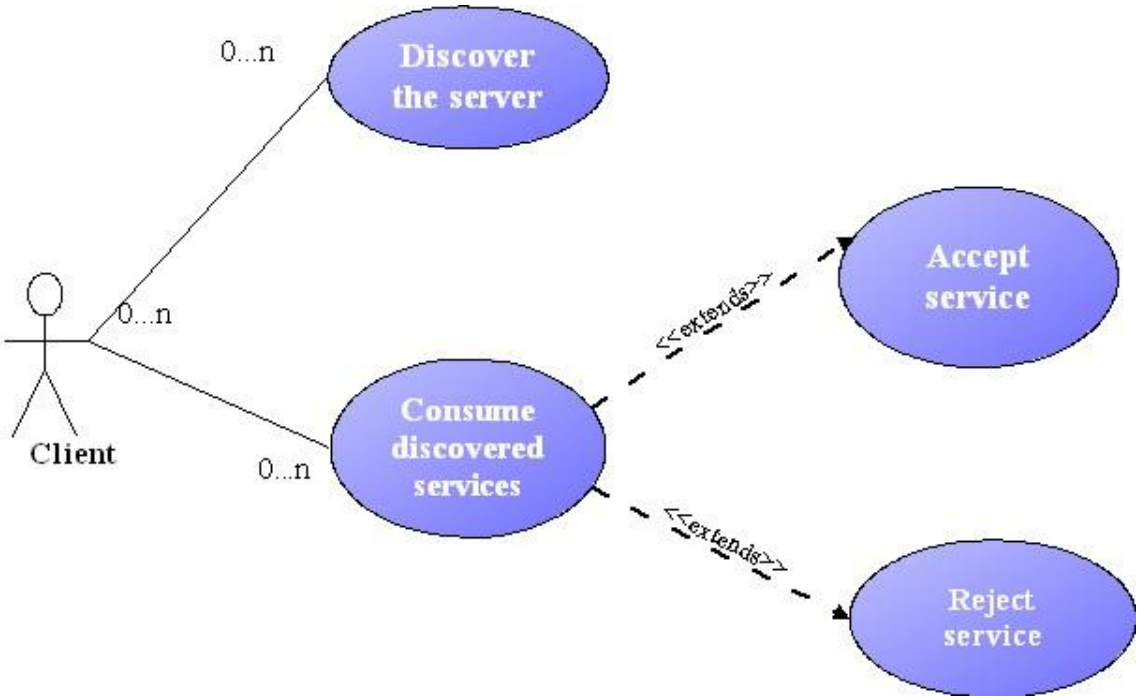


Figure16: Client Use Case

Server Use Case:

After server is started, it is going to advertise the services that it supplies. After processing the configuration files, it decides on which data to send and which poster to display. It waits for the clients and handles their requests by sending calendar event, business card or a file to the clients. It also displays the poster in the LCD. And finally it stops the advertising service. It can also send the current configuration of the application to the Admin, *see figure 17.*

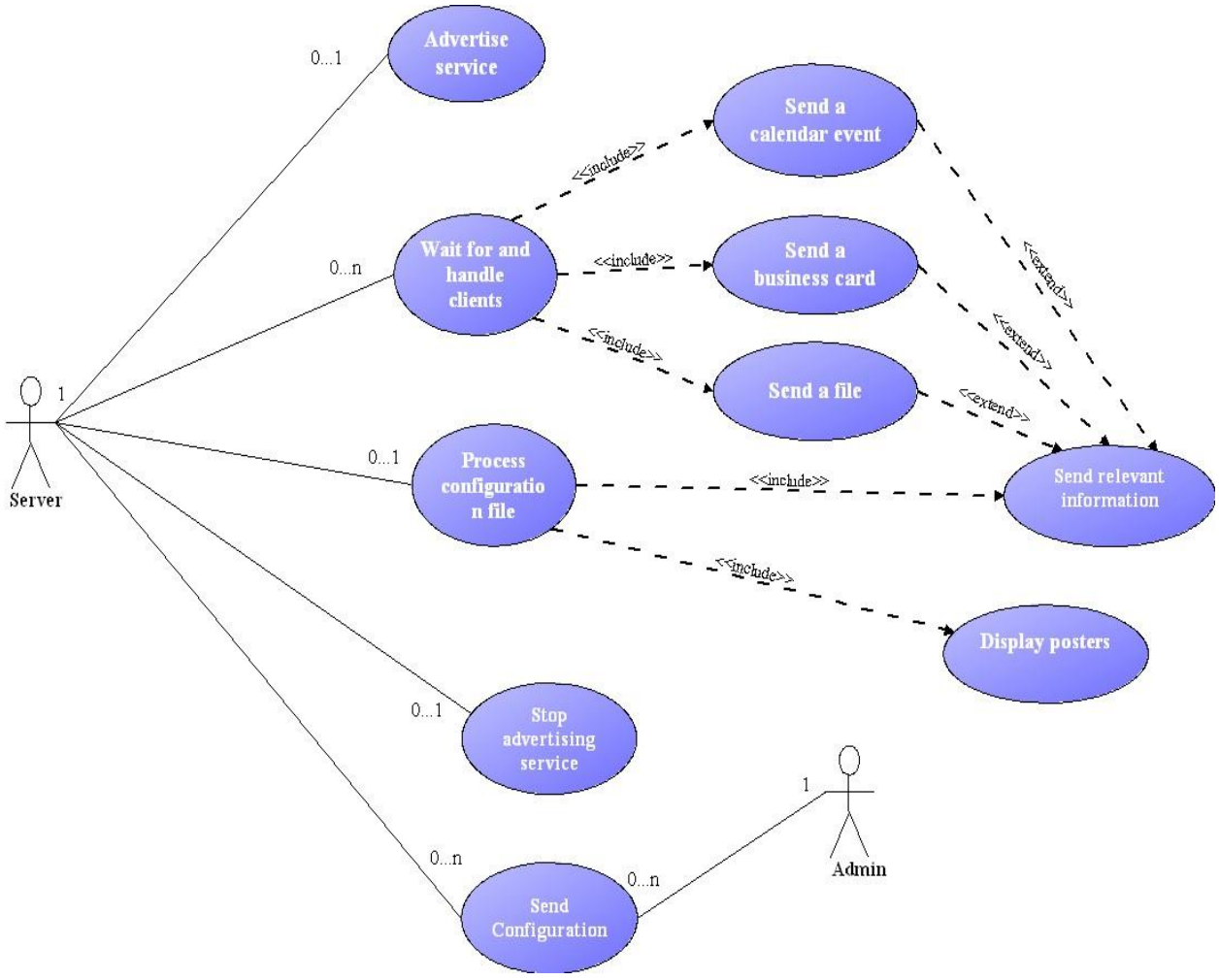


Figure17: Server Use Case

6 Bibliography

[1] XSA-3S1000 Board, http://www.xess.com/prod035.php3

[2] XILINX-WebPACK, XILINX Product and Services Web Page
http://www.xilinx.com/ise/logic_design_prod/webpack.htm

[3] Source, http://www.cs.utk.edu/~dasgupta/bluetooth

[4] Bluetooth passkey (PIN) is the key used to authenticate two Bluetooth devices (that have not previously exchanged link keys). The Bluetooth PIN has different representations at different levels. PINBB is a 128 bit (16 bytes) key used in base band level during pairing procedure while PINUI is the character representation of PINBB (coded using Unicode UTF-8) used at User Interface level.
Source, http://www.securityfocus.com

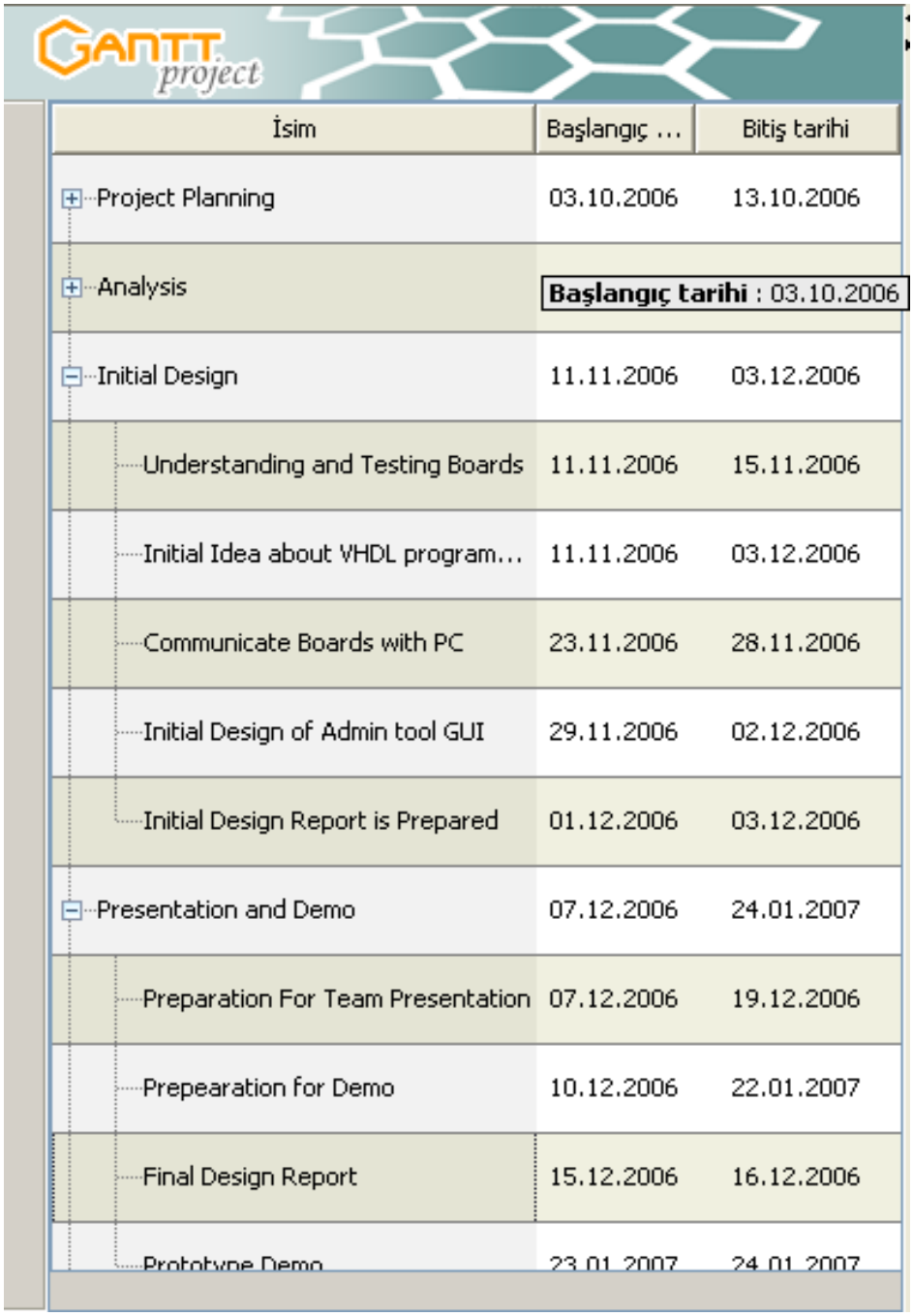
[5] XILINX ISE 8.2i, www.xilinx.com/support/sw_manuals/xilinx82/index.htm

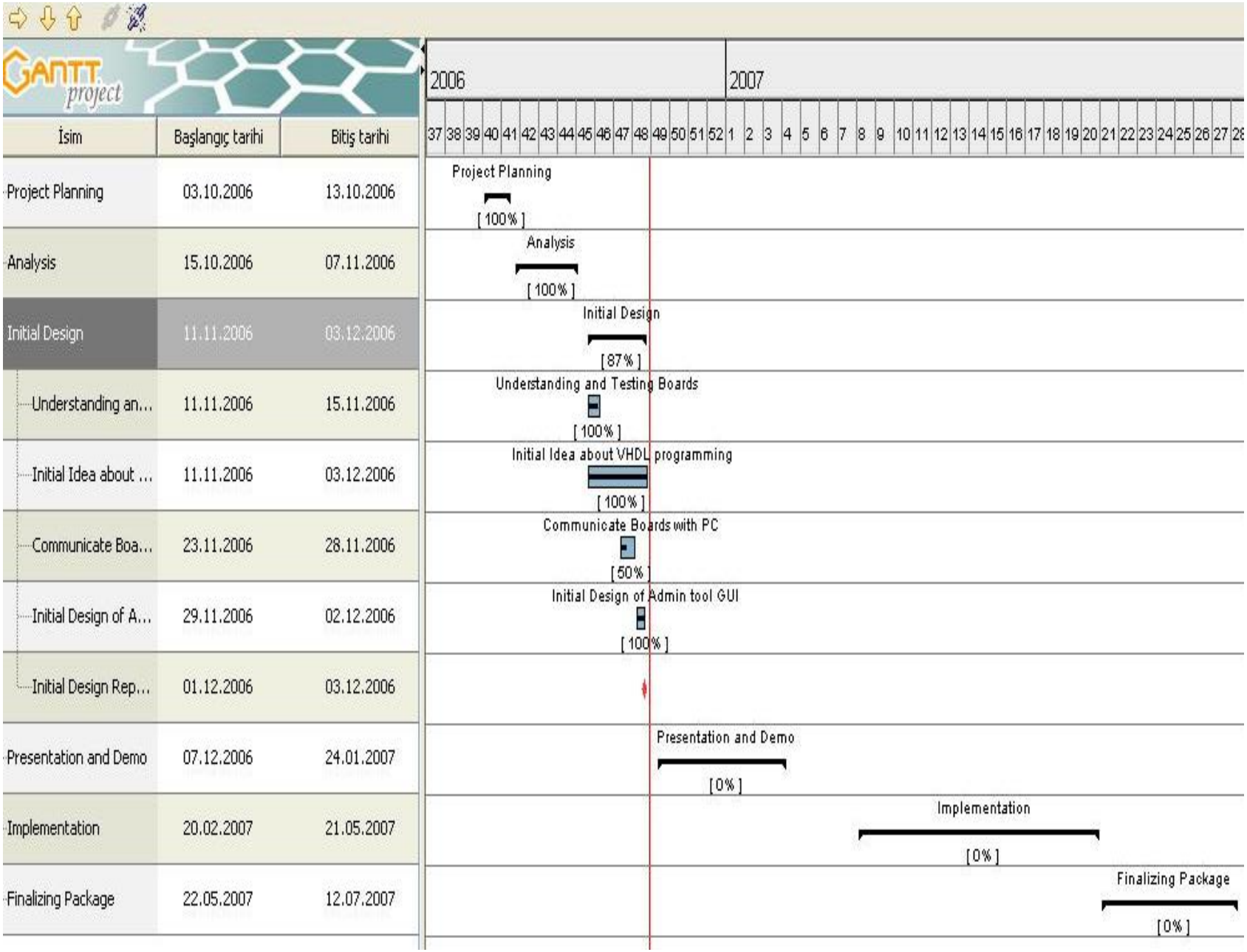
[6] VHDL (Very High Speed Integrated Circuits Hardware Definition Language),
http://www.vhdl-online.de/

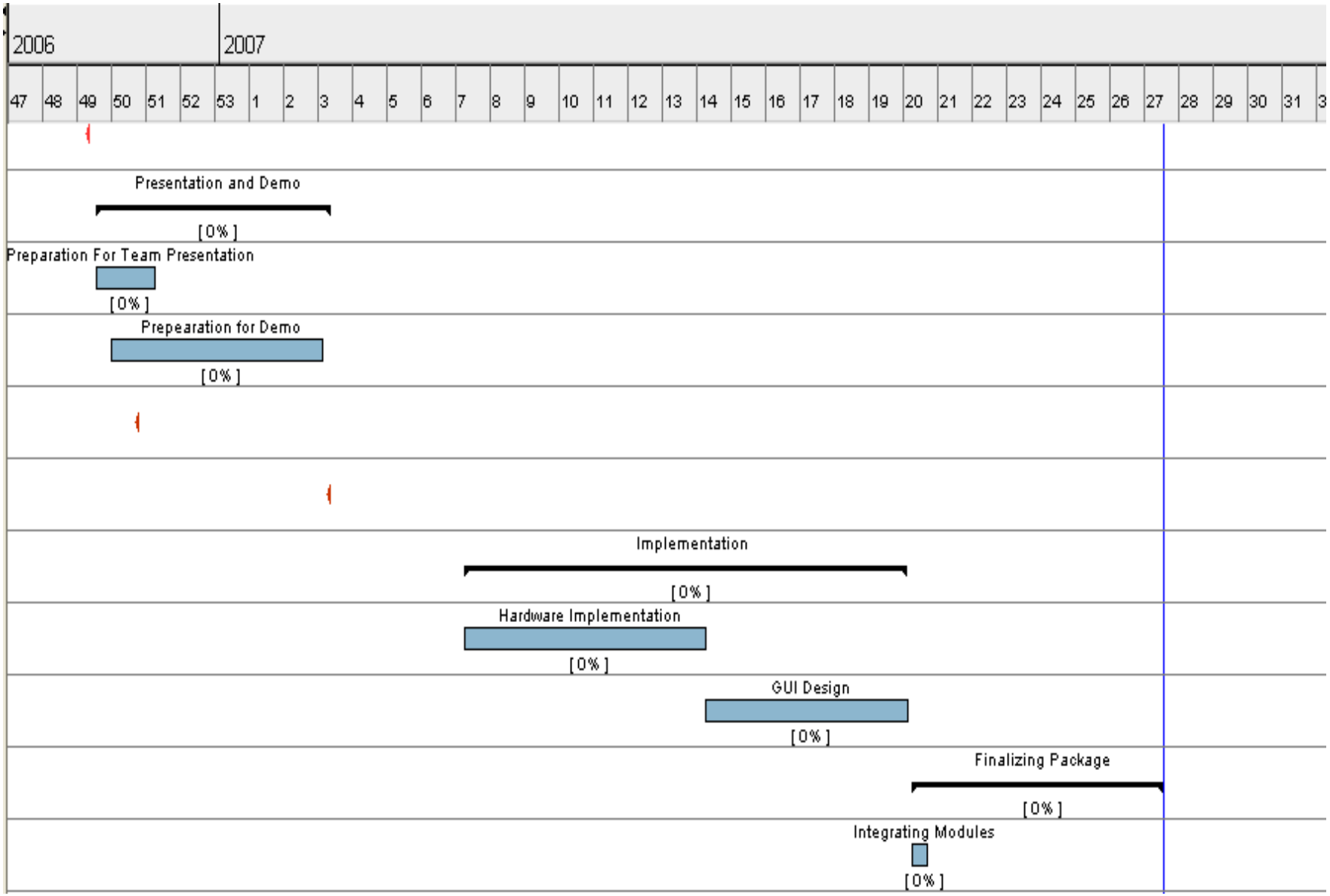
[7] Perl, http://www.perl.com/ - http://www.perl.org/

7 Appendix

Appendix A: Ganntchart




GANTT project
İsim
Başlangıç tarihi
Bitiş tarihi
2006
2007
Project Planning 03.10.2006 13.10.2006
Analysis 15.10.2006 07.11.2006
Initial Design 11.11.2006 03.12.2006
Understanding an... 11.11.2006 15.11.2006
Initial Idea about ... 11.11.2006 03.12.2006
Communicate Boa... 23.11.2006 28.11.2006
Initial Design of A... 29.11.2006 02.12.2006
Initial Design Rep... 01.12.2006 03.12.2006
Presentation and Demo 07.12.2006 24.01.2007
Implementation 20.02.2007 21.05.2007
Finalizing Package 22.05.2007 12.07.2007
[100%]
[87%]
[50%]
[0%]


2006
2007
47 48 49 50 51 52 53 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Presentation and Demo
[0%]
Preparation For Team Presentation
[0%]
Prepearation for Demo
[0%]
Implementation
[0%]
Hardware Implementation
[0%]
GUI Design
[0%]
Finalizing Package
[0%]
Integrating Modules
[0%]

```
--------------------------------------------------------------------------------
-- This design reads an image from SDRAM and displays it on a VGA monitor
--------------------------------------------------------------------------------
library IEEE, unisim, work;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use unisim.vcomponents.all;
use work.vga_pckg.all;
use work.xsasdram.all;
use work.common.all;

entity test_vga is
  generic(
    SDRAM_NROWS     :       natural := 8192;  -- 4096 for XSA-50, XSA-100; 8192 for XSA-200, XSA-3S1000
    SDRAM_NCOLS     :       natural := 512;  -- 256 for XSA-50; 512 for XSA-100, XSA-200, XSA-3S1000
    DATA_WIDTH      :       natural := 16;  -- SDRAM databus width
    SADDR_WIDTH     :       natural := 13;  -- # of SDRAM address bits
    HADDR_WIDTH     :       natural := 24;  -- host-side address width
    FREQ            :       natural := 100_000;  -- 50 MHz for XSA-50, XSA-100; 100 MHz for XSA-200, XSA-3S1000
    CLK_DIV         :       natural := 2;  -- pixel clock = FREQ / CLK_DIV
    PIXEL_WIDTH     :       natural := 8;  -- width of a pixel in memory
    NUM_RGB_BITS    :       natural := 3;  -- #bits in each R,G,B component of a pixel
    PIXELS_PER_LINE :       natural := 800;  -- width of image in pixels
    LINES_PER_FRAME :       natural := 600;  -- height of image in scanlines
    FIT_TO_SCREEN   :       boolean := true  -- adapt video timing to fit image width x height
    );
  port(
    rst_n           : in    std_logic;  -- reset
    clk             : in    std_logic;  -- master clock (frequency set by FREQ)
    vsync_n         : out   std_logic;  -- VGA vertical sync
    hsync_n         : out   std_logic;  -- VGA horizontal sync
    red             : out   std_logic_vector(NUM_RGB_BITS-1 downto 0);  -- VGA red signals
    green           : out   std_logic_vector(NUM_RGB_BITS-1 downto 0);  -- VGA green signals
    blue            : out   std_logic_vector(NUM_RGB_BITS-1 downto 0);  -- VGA blue signals
    -- SDRAM I/O
    sclkfb          : in    std_logic;  -- clock from SDRAM after PCB delays
    sclk            : out   std_logic;  -- SDRAM clock sync'ed to master clock
    cke             : out   std_logic;  -- clock-enable to SDRAM
    cs_n            : out   std_logic;  -- chip-select to SDRAM
    ras_n           : out   std_logic;  -- SDRAM row address strobe
    cas_n           : out   std_logic;  -- SDRAM column address strobe
    we_n            : out   std_logic;  -- SDRAM write enable
    ba              : out   unsigned(1 downto 0);  -- SDRAM bank address bits
    sAddr           : out   unsigned(SADDR_WIDTH-1 downto 0);  -- SDRAM row/column address
    sData           : inout unsigned(DATA_WIDTH-1 downto 0);  -- SDRAM in/out databus
    dqmh            : out   std_logic;  -- high databits I/O mask
    dqml            : out   std_logic   -- low databits I/O mask
    );
end entity;

architecture arch of test_vga is

  signal clklx        : std_logic;      -- sync'ed clock from SDRAM controller
  signal rst          : std_logic;      -- reset signal
  signal eof          : std_logic;      -- end-of-frame signal from VGA controller
  signal earlyOpBegun : std_logic;      -- indicates when an SDRAM read operation has begun
  signal rdDone       : std_logic;      -- indicates when data read from the SDRAM is available
  signal full, full_n : std_logic;      -- indicates when the VGA pixel buffer is full
  signal address      : unsigned(HADDR_WIDTH-1 downto 0);  -- SDRAM address counter
  signal pixel        : unsigned(DATA_WIDTH-1 downto 0);  -- pixel values from SDRAM

begin

  rst <= not rst_n;

  -- update the SDRAM address counter
  process(clklx)
  begin
    if rising_edge(clklx) then
      if eof = YES then
        address <= TO_UNSIGNED(0, address'length);  -- reset the address at the end of a video frame
      elsif earlyOpBegun = YES then
        address <= address + 1;         -- go to the next address once the read of the current address has begun
      end if;
```

```
    end if;
  end process;

  -- XSA SDRAM controller used to get pixel data from the external SDRAM
  u0 : XSASDRAMCntl
    generic map(
      FREQ         => FREQ,
      PIPE_EN      => true,             -- use pipelining for maximum speed
      MAX_NOP      => 1000000,          -- disable self-refresh since it takes too long to re-awaken the SDRAM with video timing
      DATA_WIDTH   => DATA_WIDTH,
      NROWS        => SDRAM_NROWS,
      NCOLS        => SDRAM_NCOLS,
      HADDR_WIDTH  => HADDR_WIDTH,
      SADDR_WIDTH  => SADDR_WIDTH
      )
    port map(
      -- host side
      clk          => clk,              -- master clock
      clklx        => clklx,            -- master clock resync'ed to account for delays to external SDRAM
      rst          => rst,
      rd           => full_n,           -- initiate a read when the VGA pixel buffer is not full
      hAddr        => address,          -- the address to read from is stored in the address counter
      earlyOpBegun => earlyOpBegun,     -- indicate when the read operation has actually begun
      rdDone       => rdDone,           -- indicate when the data from the read operation is available
      hDOut        => pixel,            -- this is the pixel data that was read from the SDRAM
      wr           => '0',              -- no SDRAM writing is needed in this application
      hDIn         => TO_UNSIGNED(0, DATA_WIDTH),  -- set the SDRAM write-data bus to zeroes
      -- SDRAM side
      sclkfb       => sclkfb,
      sclk         => sclk,
      cke          => cke,
      cs_n         => cs_n,
      ras_n        => ras_n,
      cas_n        => cas_n,
      we_n         => we_n,
      ba           => ba,
      sAddr        => sAddr,
      sData        => sData,
      dqmh         => dqmh,
      dqml         => dqml
      );

  -- VGA generator
  ul : vga
    generic map (
      FREQ            => FREQ,
      CLK_DIV         => CLK_DIV,
      PIXEL_WIDTH     => PIXEL_WIDTH,
      PIXELS_PER_LINE => PIXELS_PER_LINE,
      LINES_PER_FRAME => LINES_PER_FRAME,
      NUM_RGB_BITS    => NUM_RGB_BITS,
      FIT_TO_SCREEN   => FIT_TO_SCREEN
      )
    port map (
      rst             => rst,
      clk             => clklx,         -- use the resync'ed master clock so VGA generator is in sync with SDRAM
      wr              => rdDone,        -- write to pixel buffer when the data read from SDRAM is available
      pixel_data_in   => std_logic_vector(pixel),  -- pixel data from SDRAM
      full            => full,          -- indicates when the pixel buffer is full
      eof             => eof,           -- indicates when the VGA generator has finished a video frame
      r               => red,           -- RGB components
      g               => green,
      b               => blue,
      hsync_n         => hsync_n,       -- horizontal sync
      vsync_n         => vsync_n,       -- vertical sync
      blank           => open
      );
  full_n <= not full;                   -- negate the full signal for use in controlling the SDRAM read operation

end arch;
```

Appendix C: Data Dictionary

Name:	Login Info and Data
Aliases:	none
Where used/how used:	Admin Bluetooth Device (output) Authentication Interaction (input)
Description: Login Info and Data = Admin Login Info + Poster Data Admin Login Info = *a string of 15 characters* Poster Data = * file of 2.5 MB *	

Name:	Query Data and Login Info
Aliases:	none
Where used/how used:	Authentication Interaction (output) Admin Bluetooth Device (input)
Description: Query Data and Login Info = Poster Query Data + Admin Login Response Poster Query Data = * a string of 200 characters * Admin Login Response = * a string of 15 characters *	

Name:	User name
Aliases:	none
Where used/how used:	Check Username (output) Authentication Interaction (input)
Description: User name = * a string of 15 characters *	

Name:	Validity Message
Aliases:	none
Where used/how used:	Authentication Interaction (output) Check Username (input)
Description:	Validity Message = *a string of 25

Name:	Password
Aliases:	None
Where used/how used:	Check Password (output) Authentication Interaction (input)
Description: Password = * a string of 15 characters *	

Name:	Validity Message
Aliases:	none
Where used/how used:	Authentication Interaction (output) Check Password (input)
Description: Validity Message = *a string of 25 characters*	

Name:	Session Info
Aliases:	none
Where used/how used:	Authentication Interaction (output) Admin Interface (input)
Description: Session Info = * a string of 50 characters*	

Name:	Operation Information
Aliases:	none
Where used/how used:	Operation Status (output) Data and Poster Uploader (input)
Description: Operation Information = *string of 25 characters*	

Name:	Operation Status
Aliases:	none
Where used/how used:	Data and Poster Uploader (output) Operation Status (input)
Description: Operation Status = * string of 25 characters *	

Name:	Needed Data
Aliases:	none
Where used/how used:	Poster Data (output) Data and Poster Uploader (input)
Description: Needed Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters*	

Name:	Existing Data and New Query
Aliases:	none
Where used/how used:	Data and Poster Uploader (output) Poster Data (input)
Description: Existing Data and New Query = Poster Data + Data Query Poster Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters	

Name:	Operation Information
Aliases:	none
Where used/how used:	Operation Status (output) Poster and Data Updater (input)
Description:	Operation Information = *string of 25 characters*

Name:	Operation Status
Aliases:	none
Where used/how used:	Data and Poster Updater (output) Operation Status (input)
Description: Operation Status = * string of 25 characters *	

Name:	Needed Data
Aliases:	none
Where used/how used:	Poster Data (output) Data and Poster Updater (input)
Description: Needed Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters*	

Name:	Existing Data and New Query
Aliases:	none
Where used/how used:	Data and Poster Updater (output) Poster Data (input)
Description: Existing Data and New Query = Poster Data + Data Query Poster Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters	

Name:	Demand for Service
Aliases:	none
Where used/how used:	3rd Party Bluetooth Device (output) Pairing Protocol (input)
Description: Demand for Servis =Is_Service_Available Is_Service_Available = * boolean *	

Name:	Data Response
Aliases:	none
Where used/how used:	Poster Data (output) Communication Protocol (input)
Description: Data Response = Poster Data Poster Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters	

Name:	Pairing Validation
Aliases:	none
Where used/how used:	Pairing Protocol (output) Communication Protocol (input)
Description: Pairing Validation = Is_Pairing_Valid Is_Pairing_Valid = *bolean*	

Name:	Data Response
Aliases:	none
Where used/how used:	Poster Data (output) Communication Protocol (input)
Description: Data Response = Poster Data Poster Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters	

Name:	Data Query
Aliases:	none
Where used/how used:	Communication Protocol (output) Poster Data (input)
Description: Data Query = *string of 100 characters*	

Name:	Request Current Poster Info
Aliases:	none
Where used/how used:	Communication Protocol (output) Poster Image and Data Loader (input)
Description: Request Current Poster Info = Is_Poster_Info_Valid Is_Poster_Info_Valid = * boolean *	

Name:	Poster Identification
Aliases:	none
Where used/how used:	Poster Image Sender (output) Communication Protocol (input)
Description: Poster Identification = Identification code Identification code = * a string 0f 20 characters*	

Name:	Poster Display
Aliases:	none
Where used/how used:	Poster Image and Data Loader (output) LCD Display (input)
Description: Poster Display = * responding screen*	

Name:	Needed Data
Aliases:	none
Where used/how used:	Poster Data (output) Poster Image Sender (input)
Description: Needed Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters*	

Name:	Existing Data and New Query
Aliases:	none
Where used/how used:	Poster Image Sender (output) Poster Data (input)
Description: Existing Data and New Query = Poster Data + Data Query Poster Data = Image + Date + General Info Image = *image file of 2 MB* Date = *date* General Info = * string of 250 characters	

Name:	Needed Data
Aliases:	none
Where used/how used:	Poster Data (output) Poster Image and Data Loader (input)

Description:

Needed Data = Image + Date + General Info
Image = *image file of 2 MB*
Date = *date*
General Info = * string of 250 characters*

Name:	Existing Data and New Query
Aliases:	none
Where used/how used:	Poster Image and Data Loader (output) Poster Data (input)

Description:

Existing Data and New Query = Poster Data + Data Query
Poster Data = Image + Date + General Info
Image = *image file of 2 MB*
Date = *date*
General Info = * string of 250 characters