|  |
| --- |
| **Software Description**  **(SD)** |
| ***for the***  ***[Linux Kernel Final Project]***  ***of***  ***[Presence Data Management System]***  SEQUENCE NO. *[1]*  Prepared for:  *[Real Time Group, Linux Kernel Course]*  Prepared by:  *[Tal Taieb, Shay Goldstein, Alex Shoyhit, Avi Strugo]* |

List of Changes from Previous Revision

|  |  |  |  |
| --- | --- | --- | --- |
| Par. | Change Description | Reason of Change | Date |
|  |  |  |  |
|  |  |  |  |

Table Of Contents

1. Scope 4

1.1 Identification 4

1.2 System Overview 4

1.3 Document Overview 5

2. Referenced Documents 5

2.1 Project Related Documents 5

3. Architectural Design 6

3.1 SW Structure 6

3.2 Concept of Execution 7

3.2.1 Execution and Data Flow 7

4. Detailed Design 9

4.1 Project Files 9

5. Abbreviations and Acronyms 10

List of Tables

List of Figures

[Figure 1 - *Presents Data Management System* View 4](#_Toc52737670)

[Figure 2 - *Linux Kernel Final Project* SW Structure 6](#_Toc52737671)

# Scope

The Software Description (SD) describes the software design of the *Linux Kernel Final Project* CSCI of the *Presence Data Management* System.

## Identification

System name: *Presence Data Management System*

System abbreviation: *PDMS*

CSCI number: *RTG100LK01*

CSCI Name: *Linux Kernel Final Project*

CSCI abbreviation: *LKFP*

## System Overview

The Linux Kernel Final Project implements a Presents Data Management System.

The system main function is to save the employee entrance and exit attendance data details in a remote server with a time stamp. The attendance events are measured via RFID reader.

In addition, the system provides end to end management functionality which includes:

* Read and write the employee RFID card.
* Secured channel communication with remote server.
* Handling employee's presence data base at the server size.
* User command and indications interfacing to control system operation.

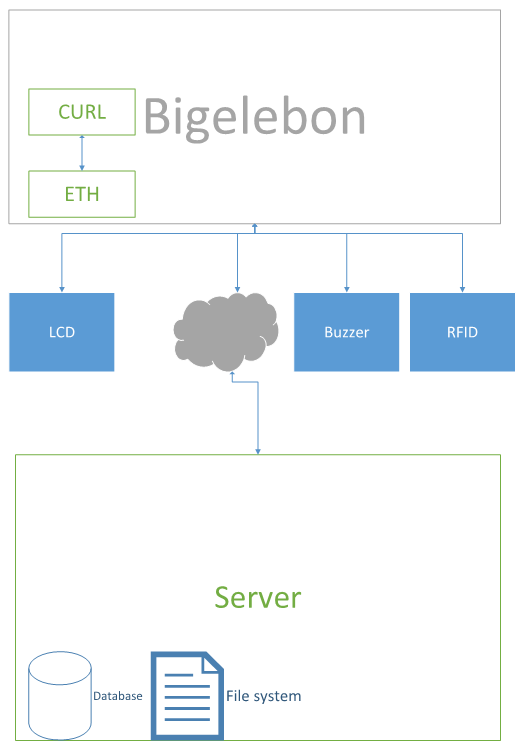


Figure 1 - *Presents Data Management System* View

## Document Overview

The document describes the software design of the *Linux Kernel Final Project*.

This document presents the system architectural description and the detailed design needed to implement the software.

The architectural description presents a feasible solution for the system/software requirements and for the end-user operational processes.

The *Linux Kernel Final Project* software architecture was derived from its system architecture. It was built in accordance with management and development concepts and according to software engineering principles that aid the team in producing a product based on the project definitions and components specifications.

This document is structured as follows:

| **Section** | **Contents** |
| --- | --- |
| Section 1 | Scope of this document |
| Section 2 | Documents referenced by this document |
| Section 3 | Architectural Design |
| Section 4 | Detail design |
| Section 5 | Notes and Acronyms |
| Appendix A | Architecture guidelines and/or Design concept |

# Referenced Documents

## Project Related Documents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Doc. ID** | **Title** | **Revision** | **Date** |
|  | Speed\_BBG\_SRM\_V3\_20150804.pdf | BeagleBone Green System Reference Manual |  |  |
|  | MFRC522.pdf | RFID product datasheet |  |  |
|  | HD44780.pdf | LCD datasheet and docs |  |  |
|  | Full time clock.jpg | Data Base example as created in the web server |  |  |
|  |  | Video clips for project description |  |  |
|  |  |  |  |  |

## 

# Architectural Design

## SW Structure

The following figure described the Linux Kernel Final Project SW structure.

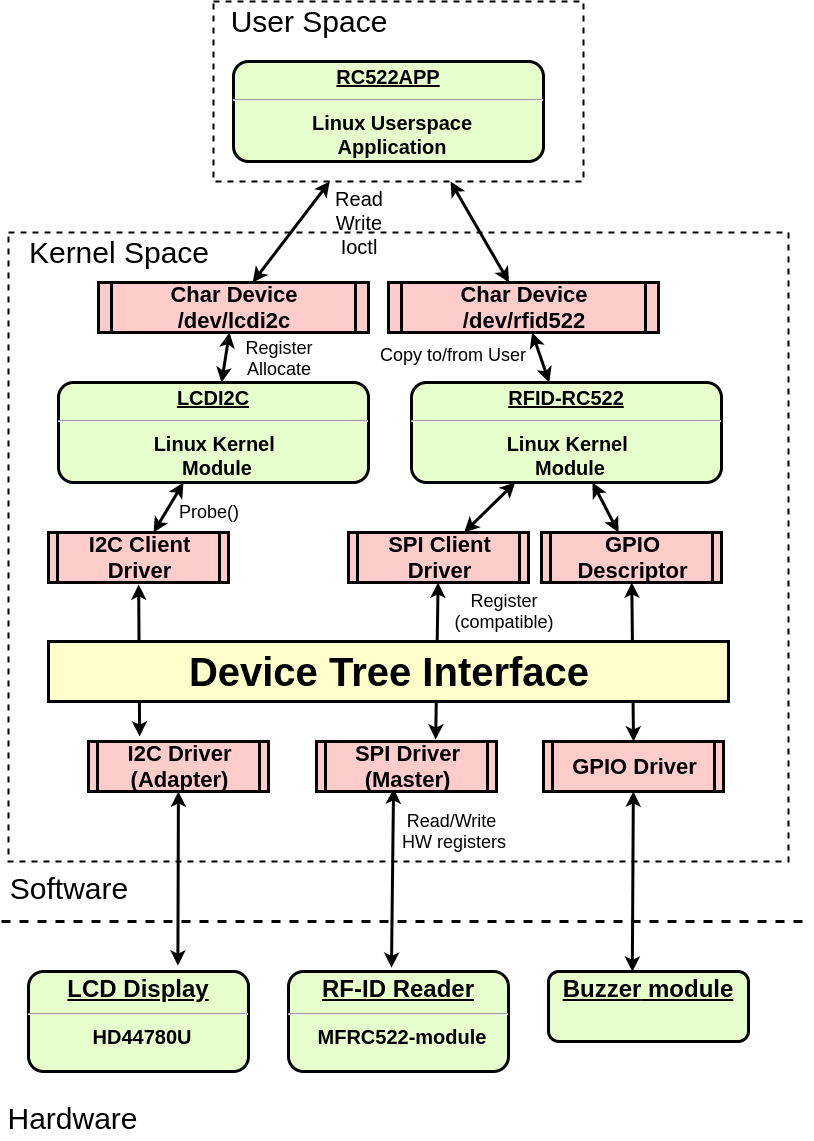


Figure 2 - *Linux Kernel Final Project* SW Structure

## Concept of Execution

## Execution and Data Flow

**System General Flow is depicted below:**

1. Initialization is performed by Call to init() which opens files for read and write:

- /dev/rfid\_rc522\_dev – A char dev from rfid card module.

- /dev/lcdi2c – A char dev from lcd module.

2. Application prints the main menu to screen.

3. The user selects which action to be performed:

**Read card serial number.**

* Invoke MFRC522\_ReadSerialNum().
* This sends ioctl command GET\_ID (with userspace buffer ‘a’) to the rfid kernel module through the char dev.
* The driver will fill the buffer with the serial number of the card.

**Write user name**

Invokes MFRC522\_WriteUserNameToCard() which triggers the following sequence:

* Send iotcl command CHANGE\_BLOCK to rfid driver in order to select card's memory block number(1) for the next write operation.
* Receive from user the user name.
* Write the user name to the card by a write command to char dev.
* Display the username on the Lcd by writing it to lcdi2c char dev.
* Send ioctl command BEEP to rfid card module to generate a sound from buzzer.

**Write user id**

Invoke MFRC522\_WriteUidToCard() which triggers the following sequence:

Send iotcl command CHANGE\_BLOCK to rfid driver in order to select card's memory block number(2) for the next write operation.

Receive from user the user id.

Write the user id to the card by a write command to char dev.

Display the user id on the Lcd by writing it to lcdi2c char dev.

Send ioctl command BEEP to rfid card module to generate a sound from buzzer.

**Read user name**

Invoke MFRC522\_ReadUserNameFromCard() which triggers the following sequence:

Call to readBlockFromCard().

Send ioctl CHANGE\_BLOCK to point the driver to read the user name from card's memory block number (1).

Read the rfid card's username data via char dev.

Write to the allocated memory region the username data.

Display the username on the Lcd by writing it to lcdi2c char dev.

Send ioctl command BEEP to rfid card module to generate a sound from buzzer.

**Read user id**

Invoke MFRC522\_ReadUserNameFromCard() which triggers the following sequence:

* Call to readBlockFromCard().
* Send ioctl CHANGE\_BLOCK to point the driver to read the user id from card's memory block number (2).
* Read the rfid card's user id data via char dev.
* Write to the allocated memory region the user id data.
* Display the username on the Lcd by writing it to lcdi2c char dev.
* Send ioctl command BEEP to rfid card module to generate a sound from buzzer.

**Read Username & Userid**

Invoke MFRC522\_ReadUnameAndUid() which triggers the following sequence:

* Call to readBlockFromCard() twice: in the first call send 1 as parameter to read user name; in second send 2 as parameter to read user id.
* Display the consequence username and user id received data on the Lcd by writing it to lcdi2c char dev.
* Send ioctl command BEEP to rfid card module to generate a sound from buzzer.
* Send the username data, user id data to a web server.

The web server will store the received data with time stamp.

# Detailed Design

## Project Files

*Linux Kernel Final Project*includes the following Files:

lcdi2c.c , lcdi2c.h

* LCD Linux Module Driver files.
* Register char dev to communicate with userspace.
* Register i2c driver with Device-Tree to communicate with hardware through i2c bus.
* Provide API’s to read/write/configure the hardware.

rc522.c , rc522\_api.c , rc522\_api.h

* RF-ID RFC522 Linux Module Driver files.
* Register char dev to communicate with userspace.
* Register spi driver with Device-Tree to communicate with hardware through spi bus.
* Get a gpio descriptor from Device-Tree to control a buzzer.
* Provide API’s to read/write/configure the hardware.

rc522App.c , rc522App.h

* Linux Userspace Application files.
* Print actions menu to the user.
* Write user name & user id to RFC522 cards.
* Read user name & user id from RFC522 cards.
* Display name & uid on the LCD module.
* Store the information from card in a database.

curl.c

* Handles data transfers to remote server.

# Abbreviations and Acronyms

|  |  |
| --- | --- |
| AI | Action Item |
| API | Application Program Interface |
| BMP | Bitmap format |
| BPM | Business Process Modeling |
| CDRL | Contractor Deliverable Resource List |
| CI | Configuration Item |
| COM | Common Object Model |
| CPU | Computer Processing Unit |
| CSCI | Computer Software Configuration Item |
| DB | Database |
| DBMS | Database management system |
| DCOM | Distributed Component Object Model |
| DID | Data Item Description |
| ERD | Entity Relationship Diagram |
| FG | Future Growth |
| FSM | Finite State Machine |
| FTP | File Transfer Protocol |
| GUI | Graphical User Interface |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| HWF | Human Work Flow |
| IDD | Interface Design Document |
| IDE | Interactive Development Environments |
| IPC | Inter-Process Communication |
| IT | Information Technology |
| LAN | Local Area Network |
| MSMQ | Microsoft Message Queue |
| MVC | Model View Controller |
| NA | Not Applicable |
| OLE | Object Linking and Embedding |
| OOPL | Object-oriented programming language |
| PUI | Project Unique Identifier |
| RAD | Rapid Application Development |
| RT | Real Time |
| SDD | Software Design Document |
| SDP | Software Development Plan |
| SRS | Software Requirements Specification |
| STD | Software Test Description |
| SW | Software |
| TBD | To be Defined |
| UML | Unified Modeling Language |
| WAN | Wide Area Network |
| XML | eXperimental Markup Language |
|  |  |