**INTRODUCTION**

1.1. Intended Audience

1.2. Project Purpose

1.3. Key Project Objectives

1.4. Project Scope and Limitation

1.4.1. In Scope

1.4.2. Out of scope

1.4.3. Limitation

**DESIGN OVERVIEW**

2.1. High level diagram

2.1.1. Error Detection and Exceptional Handling

2.1.2. Memory Management

2.2 Low level Module Interaction

**INTRODUCTION**

Automatic Image Upgrade Application Software The project is to implement an automatic image upgrade application software using configuration file inputs to perform image upgrade of 2 or more device specific image files in a memory mapped file. It shall also perform validation before and after upgrade to ensure integrity of the image. Failure in image upgrade shall restore device back to original state.

**1.1 INTENDED AUDIENCE**

| **Employee ID** | **Name** | **Roles** |
| --- | --- | --- |
| **46192296** | **Karan Bajaj** | **Group Spoc** |
| **46192205** | **Priyam Jain** | **Team Member** |
| **46192200** | **Kshitij Pal** | **Team Member** |
| **46192273** | **Madhur Mangal** | **Team Member** |
| **46192180** | **Aditya Chauhan** | **Team Member** |

**1.2 PROJECT PURPOSE**

Automatic Image Upgrade Application Software The project is to implement an automatic image upgrade application software using configuration file inputs to perform image upgrade of 2 or more device specific image files in a memory mapped file. It shall also perform validation before and after upgrade to ensure integrity of the image. Failure in image upgrade shall restore device back to original state.

**1.3. KEY PROJECT OBJECTIVES**

* To be able to read 2 or more configuration file names as command line parameters.
* Source files should get picked from default path.
* Should verify the integrity of the source image and of copied image
* Should calculate the absolute address of the image to be loaded w.r.t to base address and start address. Handle errors if any in file content
* Use a memory mapped region to simulate Target device
* Should be able to query and get the current device information including details about all images

**1.4. PROJECT SCOPE and LIMITATION**

**1.4.1. In Scope**

* For this implementation, we are using Linux.
* We are going to test some test cases to check program is running fine.
* We have provided input of configuration files from both command line and by user input at run time.
* We have used base path address of image file.
* We have parsed the data of configuration file and validated it with the device data. If magic number of configuration file and device is similar then we will update the image file in device.

**1.4.2. Out of scope**

* + We are testing and executing the implementation on a local PC rather than on a cloud server in this case.
  + We have to change file path to run this project on different PC’s.
  + We have used text files instead of image files.

**DESIGN OVERVIEW**

Automatic Image Upgrade Application Software The project is to implement an automatic image upgrade application software using configuration file inputs to perform image upgrade of 2 or more device specific image files in a memory mapped file. It shall also perform validation before and after upgrade to ensure integrity of the image. Failure in image upgrade shall restore device back to original state easy to find a block of the appropriate size if one is available.

We have made the console for menu driven program

1. Option 1- Enter the no of devices and device details
2. Opton 2- Updation of image files in device
3. Option 3- Updation of image files in device from command line
4. Option 4- Display device details
5. Option 5- Quit

**2.1. HIGH LEVEL DIAGRAM**

### **2.1.1 Updating Configuration file**

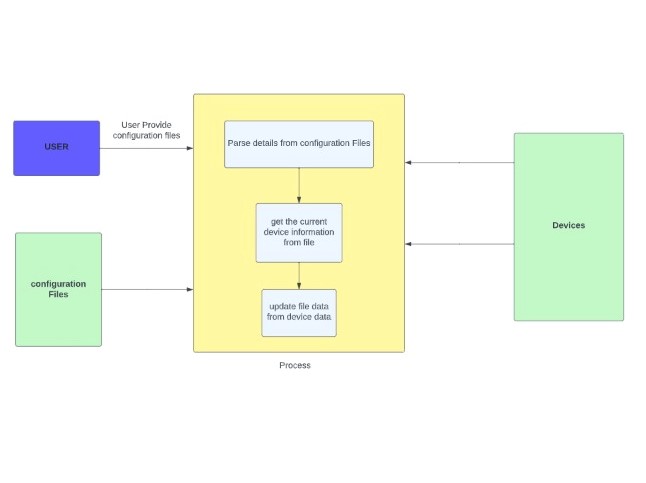
1. User passes the configuration file as command line arguments.

2. Search for the devices to be updated.

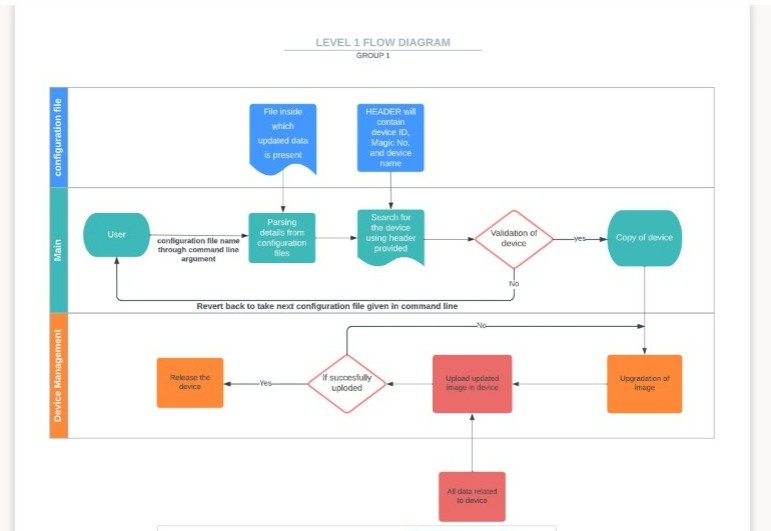
* If magic number and device number matches, file is updated.

It gets the current device information from the file and this uses a best-fit search to find the device with exact same device ID and magic number. When a device is found the configuration file is updated upon verifying the integrity of source image and copied image.

Here is an illustration of how the block is divided.



**2.2. LOW LEVEL MODULE INTERACTION**



**Device Class:**

**device(string deviceID, string magicNumber, string DeviceName, size\_t checkSum,string data,string updatedData, bool isDeviceUpdated)**

* Stores device information in the class.
* Contains get and set functions to call the values.

**Extract Details:**

**void extractdetails(vector<pp> &dataMagicNumberStore, vector<string> &fileContent)**

* Extract details of the device from the vector.
* Opens the base address file.

**Validate:**

**void validate(device \*p,vector<pp> & dataMagicNumberStore,int numberofDevice,string filename)**

* Checks the device with magic number and verifies if the file is updated or not.
* If the “getisDeviceUpdated() = false” the file is updated else error message is shown saying “ Device is already updated and this file is duplicate.”

I**nput Device Details:**

**device(string deviceID,string magicNumber,string DeviceName,size\_t checkSum,string data,string updated\_data, bool isDeviceUpdated)**

* Requests the number of device to be updated.
* We enter the device details including: device id, device name, magic number, check sum and the data string.
* Then we push both the blocks onto the level below it on our free list.

**Image File Upgrader:**

**void imageFileUpgrade(device \*p , int numberofDevice)**

* Requests the number of additional configuration file to be updated on the devices.
* We enter the name of the configuration files one at a time.
* Opens and reads the configuration file.
* Calls the extract Details function to validate the details with the device number and magic number.

**2.2.1. Merging all the functionalities :**

**Main() :**

**int main(int argc , char\* argv[])**

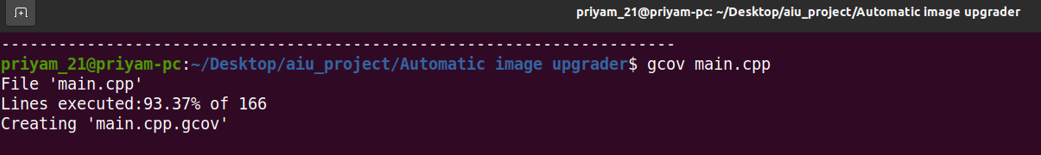
* Inputting device details like device address, Number of devices and unique representation of device.
* Checksum function for magic number, device configuration data.
* Image file upgrade and image upgrader command line.
* Check for configuration file data and upgrade if found True.

**Detection and Exceptional Handling**

* **Null Check :** For every dynamic memory allocated in heap we have checked whether memory is allocated or not.
* **Invalid Input Handling:** For every input provided by the user is checked for validation against boundaries as well as invalidation
* **File Handling:** we have checked whether file exist or not and if exist We have opened it otherwise we thrown error like file doesn’t exist and file doesn’t have complete permission.

**Code Coverage**

we have used **GCOV** tool for checking code coverage of our design and our code coverage is approx to 93%



**Memory Management**

we have allocated memory to device objects at compile time in heap and we have used delete [] device to deallocate/free memory and we have used valgrind tool to check memory leakage and command used for this : valgrind --leak-check=full

