Software requirements specification for project “STM32-bootloader”

1. Authors

- Bortnikov Pavel
- Tarasov Artem
- Vasilev Pavel
- Patrushev Boris

2. Introduction

The main idea of this project is to develop a bootloader and a high-level bootloader utility program for the STM32, where the end user can choose a file with a new version of the program and then send it to the STM32. Then STM32 receives the program and stores it in the flash memory. For security reasons the transferred program is encrypted (it will be decrypted by the bootloader on the STM32 chip). Also the high-level program will be able to upload the bootloader to the device (for the updates to devices that don’t have it yet). In other words, the end user will have to just drag and drop the bootloader/program file to the utility program on their PC, choose the upload options.

3. Glossary

Bootloader - the program on the STM32 microcontroller that can download or upload data (programs or any other data)
Developer’s server - Developer server that interacts with the flasher
Device - an embedded system that has an STM32 chip to run it’s main program
Device’s firmware - the main program on the device that’s supposed to run on it.
Device’s ID - an identification number/code that developer has provided to the end users
Flasher - the utility program on the end user’s PC.

4. Actors

4.1. **Name:** End user

Goal: Having the up-to-date authentic device program
Responsibilities: Literally follow instructions and click the buttons on the PC

4.2. **Name:** Device developer

Goal: get the updates to the end user and protect their own intellectual property (namely the program)
Responsibilities: they have to provide the end user with the device’s firmware (and its updates) and the flasher.

4.3. **Name:** Developer’s server

Goal: Ensure the correct operation of the server that was created for distribution of firmware updates
Responsibilities: Transmit updates to the flasher and receive data from it

4.4. **Name:** End user without the bootloader (EUWOBL)

Goal: Install the bootloader and be up-to-date
Responsibilities: Register his device on the developer website, get the flasher and firmware, and install it. Bootloader will be installed automatically via flasher.

5. Functional requirements

5.1. Strategic Use-cases *(not really relevant for our project?)*

5.1.1. Use-case <UC-S-1>

5.2. Use-cases for End user

*5.2.1. Use-case "*Firmware update*"*

Actors: End user, device developer

Goal: To update the device's firmware

Precondition: End user must have a PC and flasher installed on it

Trigger condition: Outdated device's firmware

Extensions:

Main success scenario:

1. Device developer notifies end users that there is a new version of the device's firmware
2. End user must download the device's firmware from the internet
3. Then they open the flasher and choose the downloaded file.
4. Also they need to connect the device to the computer by USB-UART converter.
5. After all they need to click the button "upload" and wait for the device's program to be uploaded to the device.

Alternative scenario

1) <Action1>

2) <Action2>

3) …

Alternative scenario <scenario-name2>: *[optional]*

…

<Additional section>:

Notes: *[optional, any other useful information about the UC]*

5.2.2. Use-case <UC-1-2>

5.3. Use-cases for Device developer

5.3.1. Use-case "Upload firmware to developers"

Actors: End user, device developer
Goal: Get the program data
Precondition/Context: End user must have a PC and the flasher installed on it
Trigger condition: Request from developer to end user
Extensions:
Main success scenario:

1. Device developer asks user to send program
2. End user connects device to PC

3. Opens the flasher
4. Presses the button “upload program to developer”
5. Waits until program will be sent successfully

Alternative scenario Internet connection error**:**

End user must fix internet connection and try to send again

Alternative scenario Connection with device error**:**

Contact developer, report him error code and get additional information

5.4. Use-cases for EUWOBL

5.4.1. Use-case “Install the bootloader”

Actors: EUWOBL, developer’s server

Goal: Installing the bootloader (to become a End user)

Precondition: EUWOBL has to have a PC and the flasher installed on it

Trigger condition: none

Main success scenario:

1. EUWOBL presses the “Install bootloader” button
2. Flasher installs the bootloader and asks the EUWOBL for their device’s ID
3. EUWOBL puts in their device’s ID
4. Flasher contacts the developer’s server and gets the device’s firmware and installs it

6. System-wide functional requirements

[Optional. System-wide functional requirements that weave with multiple use-cases. Examples: authorization, audit]

7. Non-functional requirements

[All the subsections are optional.]

7.1. Environment

[Environment requirements are limitations for hardware and software usage including supported hardware platforms, networking infrastructure and protocols, programming languages, libraries and external services]

7.2. Performance

[The performance characteristics of the system should be outlined in this section. Examples are response time, throughput, capacity and startup or shutdown times.]

7.3. Reliability

[Reliability includes the product and/or system's ability to keep running under stress and adverse conditions. Specify requirements for reliability acceptance levels, and how they will be measured and evaluated. Suggested topics are availability, frequency of severity of failures and recoverability.]

7.4. Extensibility

[This section indicates requirements that will enhance the extensibility including extension points, compatibility, scalability, configurability]