Project Methodologies Report

Project Report

Filippo Cottone | Pietro Scandale | Francesco Vaiana
Luca Di Grazia | Julia Roca Garcia

Politecnico Di Torino
Master Degree in Embedded Systems

Contents

Contents 2

1 Introduction 3

1.1 A new firmware architecture 3

1.2 Our organization 3

1.3 Report structure 4

2 Project Management tools 5

2.1 Git 5

2.2 Overleaf 6

2.3 Redmine 6

2.4 Gantt Chart 6

2.5 Trello 7

3 Collaborations 8

3.1 SEkey 8

3.2 Functional tests 8

3.3 Debugging tools 9

4 Conclusion 10

1 Introduction

The project we developed is about optimizing the firmware structure of the SEcube™platform. This document aims to explain how we organized the team work and the management tool we used.

1.1 A new firmware architecture

The need of a new architecture comes from the fact that the already implemented firmware had a chaotic structure, without strong division of roles between inter-operating modules, making it much more inaccessible to further developments and improvements, and violating principles of holistic security, the philosophy behind this open-source project.

Figure 1.1: *SEcube™*.

The new architecture instead has a layered structure where macro-blocks communicate between them by sharing buffers and asking for services to other modules to perceive a goal, with a clear separation of duties among the cores.
Along the years, many developers have been working on this project, so a new organization was needed to allow better code accessibility and less memory usage.
The results we got at the end of our work are a clear underlying hardware abstraction layer, the interface to hardware and software modules that enables further developments, the creation of stand-alone and automatized debug tools, with high expressiveness and license-free, and optimization in terms of resource usage.

1.2 Our organization

We are a group of 5 people, so a great organization was done, in order to improve our productivity. The parallelization was not easy to reach, but we worked step by step, analyzing problems and finding solution together.

This project was a so much effort for us, but with hard and team working we solved all the problems that we met during this useful challenge and we are so satisfied about our results.

1.3 Report structure

This document is structured as follows: in **chapter 2** we describe in details the project management tools we used to cooperate, share data, schedule the work and assign responsibility during the whole duration of the project, in **chapter 3** we discuss about collaboration with people external to our team that helped us to reach our goal and meet all the requirements, and in the **chapter 4** we talk about the results of our work and the SEcube™heritage for us.

2 Project Management tools

2.1 Git

About the project management tools we used, git was the most useful one. We created a shared repository in order to try new ideas safely, and so each one of us was able to work on its own personal branch and then, in case of a successful progress, a merge with the master was executed.

Figure 2.1: *Git logo.*

This approach was efficient, because trying to find a new solution for a issue became a parallel operation that led us to multiple solutions, allowing us to choose the best one to implement a merge on the master. From our point of view Git was a necessary project management tool to organize and optimize the team productivity, even in a particular case in which the code development parallelization is limited, due to the nature of the project itself.

2.2 Overleaf

We used Overleaf, an online LaTeX editor that allowed us to write concurrently the documentation on how the new firmware architecture works.

Figure 2.2: *Overleaf logo.*

2.3 Redmine

In order to schedule the several steps of the project realization we used Redmine; then, the progress has been periodically updated and added in case of necessity.

Figure 2.3: *Redmine logo.*

2.4 Gantt Chart

Due to the coding nature and the intricate structure of this project, we were not able to parallelize ourselves; on the contrary, we have always been working together, evaluating each step carefully. When possible, we used pair programming technique to speed-up our progress, taking care of the quality of the final product.

The Gantt chart highlights how much time we needed to analyze the code and understand how it works, how its data structures were interconnected and how the offered functionalities were implemented.

After an initial phase of document study, we opened the git repository to share the project among us. Then we decided the guidelines for the development of the new architecture and sub-modules.

One by one, the macro-blocks were created, integrated and tested, and the existing code replaced. We followed the background of the V model development flow. So, at the end of the development phase, the test phase was performed by using a bottom-up approach: so, firstly the single unit was tested, then the integration test followed, and at the end the top level test was performed; in this way, if a bug rose, it was easier to understand where exactly it was located. Each test has been made possible thanks to the debugging trace logs that we obtained with the Debug tool that we developed.

2.5 Trello

During the last period, due to personal university commitments each one of us had, organizing some meeting to decide what we had do became difficult. So the solution was to use Trello, a web-based project management application to share the task to perform that each of us thinks necessary to complete the Project, and to keep us updated on the progresses of the remaining job.

Figure 2.4: *Trello logo.*

So, we created 5 different task lists:

- **To Do**: each day, during a stand up meeting of 15 minutes, the task to though to be performed were added here;
- **Executing**: The tasks in progress and in execution in a speficied moment were in this list;
- **Waiting For**: Here, the task that needs to wait for their completion due to other task dependencies are placed;
- **Done!**: The completed tasks are placed here.

Each task had some tag associated, in order to identify the nature of the task (Development, Documentation, Bug, Problem, Doubt,..), and who worked on it was connected to the task itself.

3 Collaborations

3.1 SEkey

The new Structure of SEcube™supports the interaction with the new SEkey device, a key management system able to create, erase, administer, update the communication keys of the user groups; every time a group has to be created, this new component has to provide to the administrator the necessary mechanisms to create it. Moreover, information about one group must be exclusively read or written by the users of the group itself. When creating a group, the administrator has to choose the algorithms to create and exchange the keys between the users (as DES, AES, DH, RSA, ...), the group policies, and the components of the group. All of these operations will be performed by using SEkey.

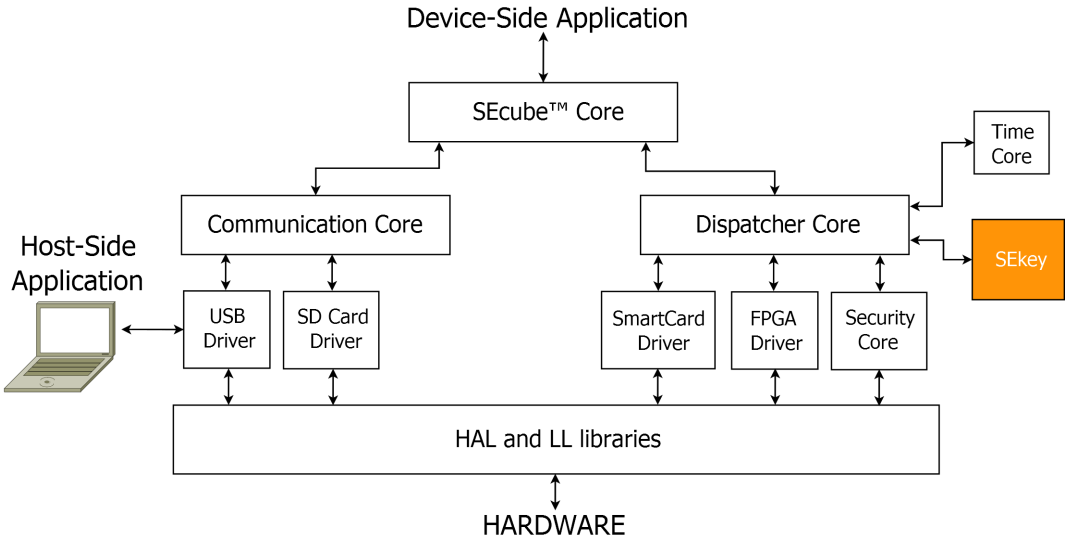

Figure 3.1: *SEkey interactions.*

A Master Thesis guy, *Mateus Françani*, is working on the development of this new device, so in order to develop the new structure a collaboration with people external to the team was needed to better understand the device specification and adapt our code development to SEkey, based on these knowledge.

3.2 Functional tests

In order to test the correct embedded firmware behavior, the provided host main.c files were used. In particular, the device_init.c file was used to test the first basic firmware functionalities.

After this, further tests needed to be proved and successfully passed. A PhD student, *Nicolò Maunero*, helped us to test our firmware using functionality tests provided in the SEcube™development kit, by executing some Host programs and finding and fixing some error introduced some years ago in the functional_test.c file, a program file that tests deeper functionalities w.r.t. the device_init.c one.

3.3 Debugging tools

We developed a debugging way, writing on SDcard a log trace file in order to see where our program stopped and step by step we fixed all the problems.

It was a very difficult challenge, because there are no systems call so using reverse engineering we implemented in the firmware some functions that emulates the creation of a file and writing on it in a system call style. At this moment, the mechanism of creating a file works only with a 16 GB SD card with a FAT32 file system.

4 Conclusion

In this singular course we learned how to improve our soft skills and how to better communicate in a working environment.

Even if ideas were sometimes different, we learned how to compare with each other and to respect our own imposed deadlines. Working in 5 people was not so easy, also due to the nature of this programming related project and due to different times of every group component, so team building was fundamental for us. We learned to not give up, discovered useful managing tools like git and had a practical view on the scheduling of duties in a working environment.

We learned to put our hand in an advanced code written by different programmers, understanding complex data structure and algorithms.

It is a firmware, no operating system runs on this board, so we wrote a lot of low level functions, because there are not system calls. We did a lot of reverse engineering in order to override all the difficulties met during this course.

We think this course, placed in this particular period of the master degree curricula, is fundamental for a next computer engineer who will work in dynamic and multicultural environment, with limited resources and with the need of smart solutions for the future.