ANALYSIS, IMPROVEMENT, AND DEVELOPMENT OF NEW FIRMWARE FOR THE SMART CITIZEN KIT AMBIENT BOARD

Miguel Colom Barco

Universitat Oberta de Catalunya

January 18, 2015
**What is a smart citizen?**

*A smart citizen* is a person living in city who has some *influence* in

- management and organization,
- technology,
- governance,
- policy context,
- people and communities,
- economy,
- built infrastructure, and
- natural environment.

**How do they achieve it?**

- They *acquire* information (for example, environmental data using sensors),
- *share* it publicly, and
- use it to get *governments* take improvement *actions*. 
Current projects There exist several projects currently active for the development of Smart Cities, from both the public and private initiative. For example:

- Amsterdam Smart City (http://amsterdamsmartcity.com/project),
- Open Cities Project (UE project, http://opencities.net/content/project),
- Smart Santander (http://www.smartsantander.eu/),
- MiNT Platform by Ayuntamiento de Madrid,
- BCN Smart City (http://smartcity.bcn.cat/),
- and many other!
Introduction

Context of the UOC project

Objective of this UOC project: contribute to the development of the SmartCitizen Project.

→ What is it?

- A Smart City project ([http://www.smartcitizen.me/](http://www.smartcitizen.me/)).
- It was started by both the Fab Lab Barcelona ([fablabbcn.org/](http://fablabbcn.org/)) and the Institut de Arquitectura Avançada de Catalunya ([www.iaac.net](http://www.iaac.net)).

- They build autonomous Arduino-compatible boards which collect environmental data.
- The collected data is sent to a centralized platform to make it public.
The SmartCitizen initiative produces boards (SCK) with the following ambient sensors:

- noise
- temperature
- humidity
- CO and NO₂ gas concentration
- lighting
- number of WiFi networks around
Introduction

GLOBAL OBJECTIVES

Objectives of this UOC project. This UOC project has two parts:

PART I: improve current SCK firmware

- **Analyze** the current firmware and learn how it works.
- **Document** both the firmware and hardware.
- **Correct** any **bugs** found.
- **Improve** the overall **quality** of the current firmware.

PART II: create the firmware for the RTX4100

- The RTX4100 is a **low-power WiFi** module.
- It will be used in new versions of the SCK.
- A previous code does not exist. It had to be written **from scratch**.
Cross tasks

The are several cross tasks which span both parts of the project and were performed continuously:

- **Continuous refactoring.**
- **Communication** the SmartCitizen team (Guillem Camprodon and Alex Posada, mainly).
- Periodic and frequent **unit testing**; Follow an approach close to **Continuous Integration**.
- **Writing documentation.** In particular, automated documentation using Doxygen.
- Analyze the current firmware and learn how it works.
- **Document** both the firmware and hardware.
- **Correct** any bugs.
- Improve the overall **quality** of the current firmware.
Part I

Analysis, documentation, and improvement of the current SDK firmware
PART I: CURRENT SITUATION

What was the situation when I started the project?

- **Hardware** design totally finished. In production.
- Firmware overall working and in production stage.
- Firmware C++ project publicly available at GitHub.
- **Lack of unit testing** which would ensure quality of further improvements.
- **Lack** of any Built-In Self Test (BIST).
- **Lack of reference documentation**, with the exception of some descriptions of the hardware in the website.
- **Lack of automated documentation** generated from code markup (with Doxygen, for example).
- Some bugs found in the production firmware.
- Identified many parts of the code which might be improved.
Introduction

Part I: engineering tasks performed

Engineering tasks performed:

- Followed a development methodology close to Continuous Integration.
- Created proper and complete documentation for all the project (UOC project deliverable).
- Created a BIST mode which executes automatically a series of tests which determine if the hardware/firmware is working correctly.
- The BIST mode is designed as a unit test, and can be extended with more tests.
- Used the cppcheck tool to automatically find and correct problematic parts of the code.
- Created functions to prevent duplicated code, specially for I2C communication.
- Minimized the memory usage by using static objects instead of duplicated instances.
- Removed unused (dead) code.
PART I: BIST UML CLASS DIAGRAM

UML class diagram of the unit test design used in the BIST

Miguel Colom Barco (Universitat Oberta de Catalunya)
Conclusions of the first part of the project:

- The gas sensors in the SCK need to be recalibrated after some months. Since regular users can not do it by themselves, it is better to avoid these kind of sensors in next versions of the SCK.
- Complete documentation for the project (hardware and firmware) was written.
- Automated documentation was added in the source code (with Doxygen).
- A new BIST mode was added, which is designed as a unit test that can be extended with more test cases.
- All found bugs were corrected.
- Several improvements were performed: memory saving, move variables to private or lower scope, create functions to avoid duplicated code, etc.
- The RN131 WiFi module consumes too much energy. It must be changed by a more efficient module. This is discussed in Part II.
- Therefore, all the objectives of the first part of the project have been met.
Part II

Design and create from scratch the firmware of the new RTX4100 WiFi module
Motivation of the second part of the project: writing a complete firmware for the RTX4100.

- The current RN131 WiFi module in the current version of the SCK v1.1 is the component which consumes the most power. It must be replaced by a more efficient alternative.

- The proposed alternative is the RTX4100, which combines a EFM32G230 microcontroller with the Atheros AR4100 WiFi chip.

- RTX4100 works with its own RTOS operating system and software framework (CoLa applications).

- A firmware to manage the RTX4100 hardware is needed.

- Since the RTX4100 has its own microcontroller, it can be used to manage communication at the OSI transport level (TCP connections) and free the main SCK microcontroller from these duties.
Requirements of the firmware developed for the RTX4100:

- It should work in a way which ensures **minimum energy consumption**.
- It must provide an **API** to control all **energy parameters** and to adjust them when needed (i.e: signal strength, among others).
- The API must provide functions to send/receive data at the OSI transport level (**TCP connections**).
- The communication with the upper layer must use binary (**non-verbose**) commands through the **SPI protocol**.
The firmware was designed such a way it meets all the stated objectives, taking special care of the energy consumption constraints.

- The developed firmware architecture is event-driven. Therefore, the CPU is only active when it needs to process an event and idle otherwise. This ensures minimal energy consumption.
- The system is modular and most of the functions are implemented as RTOS protothreads. No polling loops.
- To help debugging, a debug mode through a serial terminal is available.
- SPI communication is used in production mode.
- A unit test function is available as a debug terminal command.
- Several energy control functions available to the upper layer.
- TCP communication functions available to the upper layer through the API.
Conclusions of the second part of the project:

- The RTX4100 hardware and its developed firmware have been documented properly at the UOC project deliverable.
- The developed firmware is finished and publicly available at https://github.com/mcolom/SmartCitizenRTX4100 under the GPL license.
- Automated Doxygen documentation available.
- All the objectives for the RTX4100 second part of the project have been met in time.
Thank you for your attention