



# LILLIAD: Connected learning center

Mageshwaran SEKAR  
Department of Electronics and Computer Science  
Polytech Lille

February 2016



# Contents

<b>Acknowledgement</b>	<b>2</b>
<b>1 Introduction</b>	<b>3</b>
<b>2 Scope statement</b>	<b>4</b>
2.1 Preamble . . . . .	4
2.2 Problem statement . . . . .	4
2.3 Objective . . . . .	4
2.4 Materials . . . . .	4
2.5 Project phases . . . . .	4
<b>3 Implementation</b>	<b>6</b>
3.1 Database . . . . .	6
3.2 Deploying configuration via network . . . . .	9
3.3 Collecting sensor data . . . . .	10
3.4 Creating a website to display sensor data . . . . .	12
<b>4 Problems encountered</b>	<b>14</b>
<b>Conclusion</b>	<b>15</b>
<b>References</b>	<b>16</b>

## Acknowledgement

This final year project (FYP) will not be a success without the willingness and commitment from a number of individuals. Thereby, I would like to thank everyone who assisted me during this project and a special thanks to the following person.

First of all, I would like to thank my tutors, Mr. BOE Alexander and Mr. VANTROYS Thomas who supervised me throughout the project. Throughout the project, they came out to help when we were stuck with problems. Secondly, I would like to thank the professor from my university, Mr. HOOGSTOEL Frédéric for helping me out during the conception of a database for the FYP.

I would like to extend a special thanks to my colleague ROCHE François with whom I worked in parallel to carry out important tasks especially the electronics parts.

# 1 Introduction

As a 5<sup>th</sup> year student at Lille Graduate School of Engineering (Polytech'Lille), it is compulsory to carry out a final year project (FYP). Thus, I chose one of the Internet of Things (IoT) related project which is entitled *Connected Learning Center* since I find myself interested in this field.

This report is a description of the work done from September 2015 to February 2016. In the following chapter, details of the scope statement are given. Later on, I am going to explain the important tasks carried out during the FYP and give specific technical details. Finally, a conclusion is drawn from the experience.

## **2 Scope statement**

### **2.1 Preamble**

The library of University of Lille 1 is in the transformation phase to becoming LILLIAD Learning Center Innovation by 2016. To enrich the user experience, the new library will be equipped with multiple sensors (for temperature, air pollution, etc.) in large scale. However, it requires a lot of effort to implement and configure these sensors.

### **2.2 Problem statement**

Since the sensors are implemented in large quantity, it is difficult and tiresome to reconfigure them one by one. And each sensor is configured in a different manner. Hence, we had to come with a solution that could automate the reconfiguration process.

### **2.3 Objective**

The main goal of this project is to develop a user interface which allows the personnel of LILLIAD to use and maintain the deployed network of sensors. This interface should allow them to configure the sensors by uploading the required files to the main server. On the other hand, we will also be developing a reliable communication between the sensor boards and the server through the network.

### **2.4 Materials**

These following materials are required for this project:

- Apache Web and MySQL database server
- Raspberry Pi 2
- Several ATxmega256A3BU cards
- Sensors for temperature, air pollution, luminosity and noise

### **2.5 Project phases**

This FYP is divided into few important phases.

#### **2.5.1 Reservation of time slot to roll-out configuration**

During this phase of the project, we will be developing a Web site, which serves as a user interface where the admin of LILLIAD would be able to upload files that will be used later on for configuring the equipment (main board, daughterboard, sensors, etc.) automatically through scripts.

### 2.5.2 Deploying configuration via network

In this phase, we are going to implement few scripts in the server so that the configuration of daughter boards (DB) and daughter<sup>2</sup>board (D<sup>2</sup>B) could be carried out automatically. The result of the configuration (success or failure) will be stored in a log file in the server so that the admin could verify it and take action in case of failure.

### 2.5.3 Collecting sensor data

The data from sensors will be sent in from daughter board (through USB or serial in case of failure) to Raspberry Pi which in turn will redirect those data to the server using UDP datagram. These data will be stored in a file in the form of TSV (Tab Separated Values).

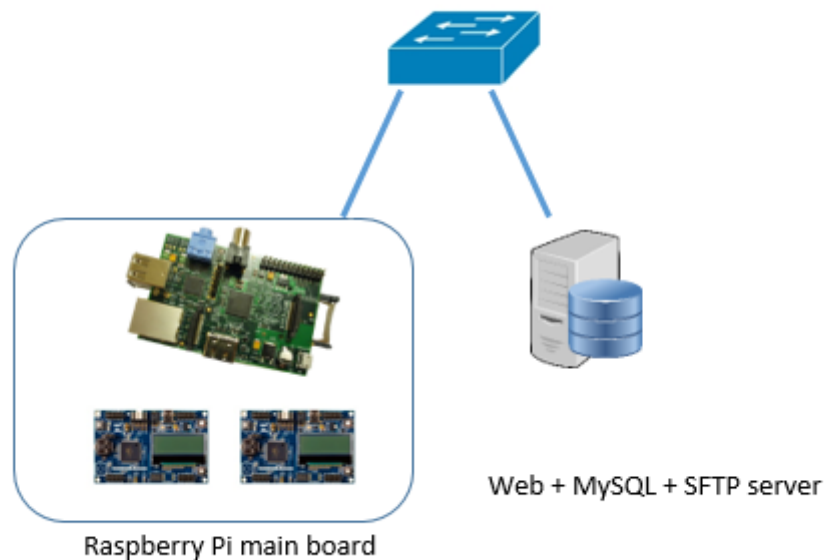


Figure 1: Partial architecture of the project

### 2.5.4 Creating a website to display sensor data

To display the collected data in an understandable way, a page with graph display will be implemented. There will be also a possibility to view instantaneous values (including minimum and maximum) through the web page.

## 3 Implementation

During the period of September 2015 to February 2016, I worked mostly on the automatic configuration of the boards and log and data collection. At the same time, I worked in parallel with François ROCHE to configure few important parts of the board and carried out some basic tests.

### 3.1 Database

#### 3.1.1 Analyse

First of all, I had to create a database which will be used to store all the important information regarding the files and the boards that will be used for the autoconfiguration. After having a few discussions with one of my professors, Mr. HOOGSTOEL Frédéric, we came up with a relational model for the database.

#### 3.1.2 Relational model

The relational model of the database is based on the following logic:

- A user should have logged in before modifying the database
- A user can play more than one role (but only the admin can modify the database)
- A user can plan an autoconfiguration
- The autoconfiguration can be implemented in one or more boards but only with a single file
- The autoconfiguration is carried out during the reserved timeslot
- A board can be configured multiples times but during a different timeslot

This database model is represented in the figure below :

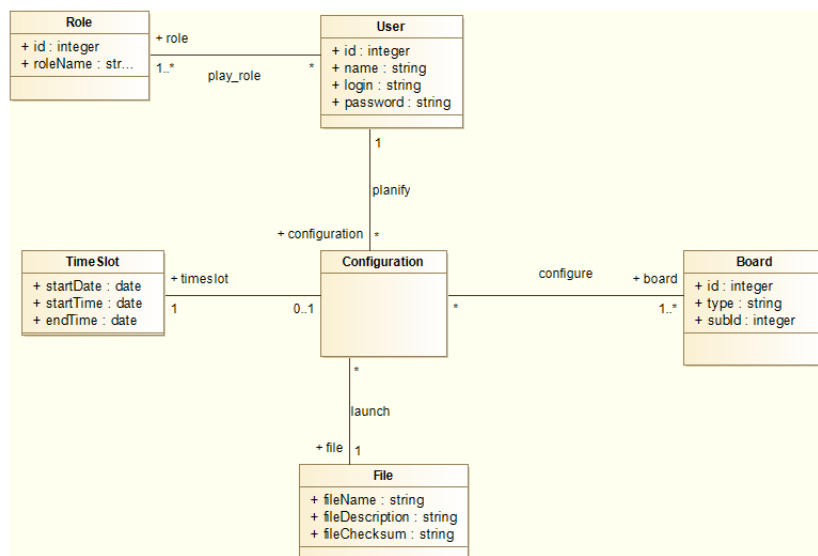
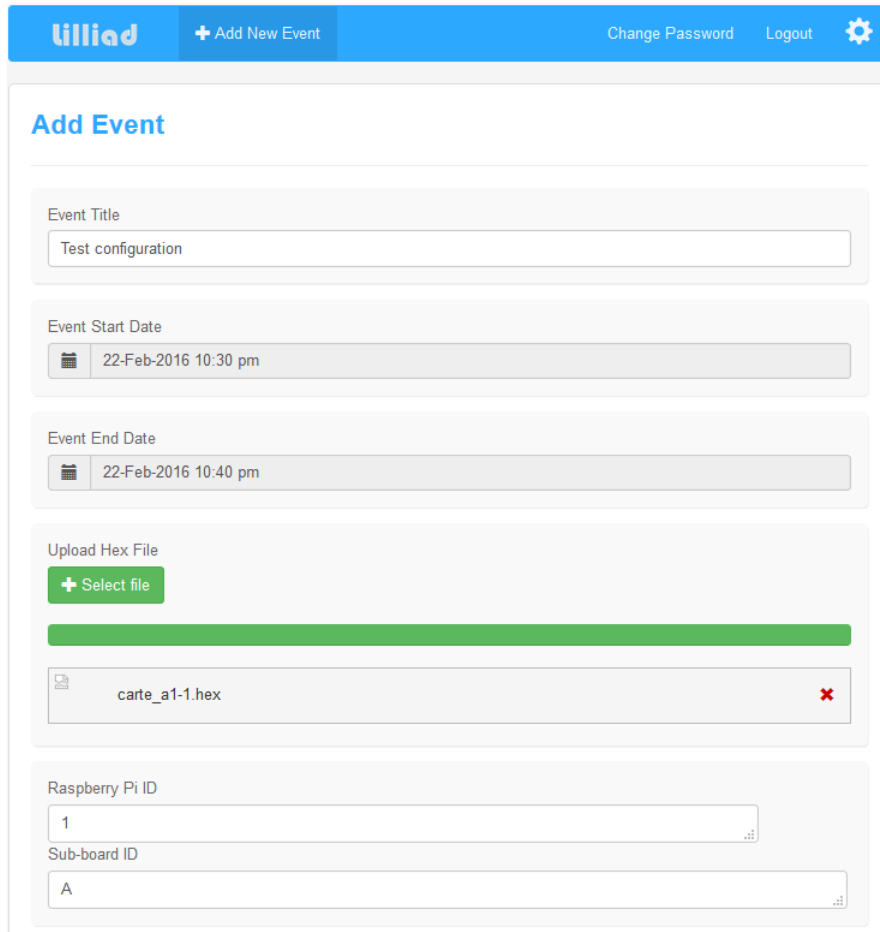


Figure 2: Database for autoconfiguration of boards

### 3.1.3 Integration of the database in a website

A website was created (cf. [link](#)) where an administrator would be able to upload files and determine timeslot to configure the boards.



The screenshot shows the 'Add Event' form in the liliad website. The form is titled 'Add Event' and is located in the main content area. The form has a blue header with the liliad logo and navigation links: '+ Add New Event', 'Change Password', 'Logout', and a settings gear icon. The form fields are as follows:

- Event Title:** A text input field containing 'Test configuration'.
- Event Start Date:** A date and time picker showing '22-Feb-2016 10:30 pm'.
- Event End Date:** A date and time picker showing '22-Feb-2016 10:40 pm'.
- Upload Hex File:** A section with a '+ Select file' button, a green progress bar, and a file upload area showing 'carte\_a1-1.hex' with a red 'x' icon.
- Raspberry Pi ID:** A text input field containing '1'.
- Sub-board ID:** A text input field containing 'A'.

Figure 3: Autoconfiguration of boards through website

In the figure above, the uploaded file is *carte\_a1-1.hex* and the chosen time for configuration is between 10.30 pm to 10.40pm on the 22nd February. These information is added to the following tables in the database: *timeslot*, *board*, *configure* and *configuration*. The schedule will be updated and the user can view it through the website (as shown in the figure below).



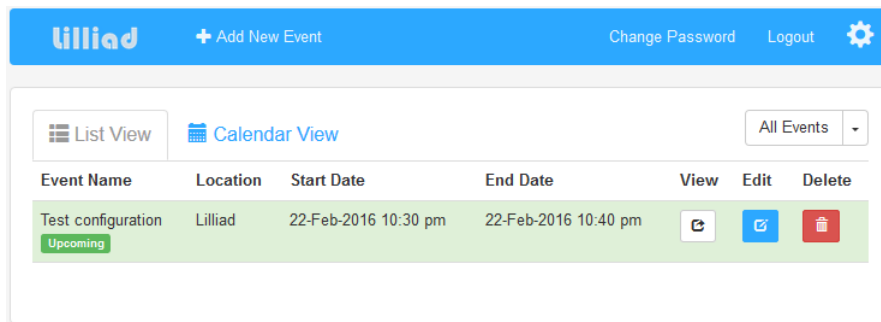


Figure 4: Schedule view

On the other hand, the server will execute a bash script everyday starting from 10pm so that all boards can be tested without disrupting user experience during the day time. Only the files configured to be deployed after 10pm and before 6am (the next day) will be taken into consideration. If the chosen schedule is out of this period, the boards will not be configured.

## 3.2 Deploying configuration via network

To simplify the task of an administrator of deploying the configuration for hundreds of boards, I came up with a solution so that the boards could be configured automatically in a given period of time (eg: from 10pm to 6am the following day). To configure the boards, we had to study about the Atmel's proprietary PDI interface.

### 3.2.1 Configuration through PDI interface

The Program and Debug Interface (PDI) is an Atmel proprietary interface for external programming and on-chip debugging of XMEGA devices. It uses pin interface using the reset pin for clock input PDI\_CLK and a dedicated data pin (PDI\_DATA) for input and output.

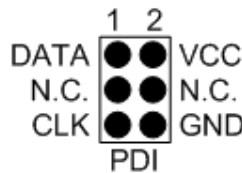


Figure 5: PDI pin headers

To program the ATXMEGA256A3BU cards, we connected the GPIO and ground pins of Raspberry to the PDI\_CLK, PDI\_DATA and ground pins of ATXMEGA respectively.

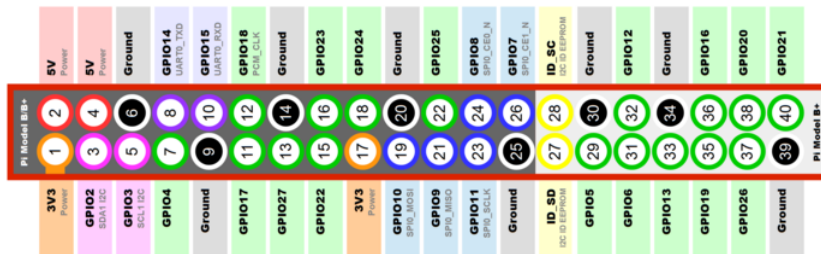


Figure 6: Raspberry Pi IO pins

We used PDI uploading program found at [GitHub](#) to configure the XMEGA card using the *hex* file uploaded through the website.

### 3.2.2 Automatic configuration of the boards

We were able to reconfigure the daughterboard. For this, we execute the script to copy the configuration file to the raspberry Pi via SCP (secure copy) and execute the command to transfer the hex file to the daughter board. And, the server will collect the information regarding the state of the configuration (success or failure) and save them in a log file. To automatize the configuration, we added the script to be launched to the */etc/crontab* so that it will be executed everyday at the given period.

### 3.3 Collecting sensor data

#### 3.3.1 From daughter board

First of all, the raspberry Pi will collect the data from the daughter boards using the High-Speed USB link (and if there's any problem with the USB link, the data will be collected through serial interface). The table below shows the data that Raspberry Pi normally receives. The possible mode of communications are: S(serial), U(USB), R(Radio communication).

Communication mode	RPi ID	Daughter Board ID	D <sup>2</sup> B ID	Data
--------------------	--------	-------------------	---------------------	------

The data will then be sent to the server through UDP communication.

#### 3.3.2 UDP communication

The data from the DB will be sent to the server through a UDP connection where the raspberry Pi acts as UDP client and the server listens to a port for UDP datagrams. So, the UDP server will be able to receive UDP datagrams and extract data sent by raspberry Pi.

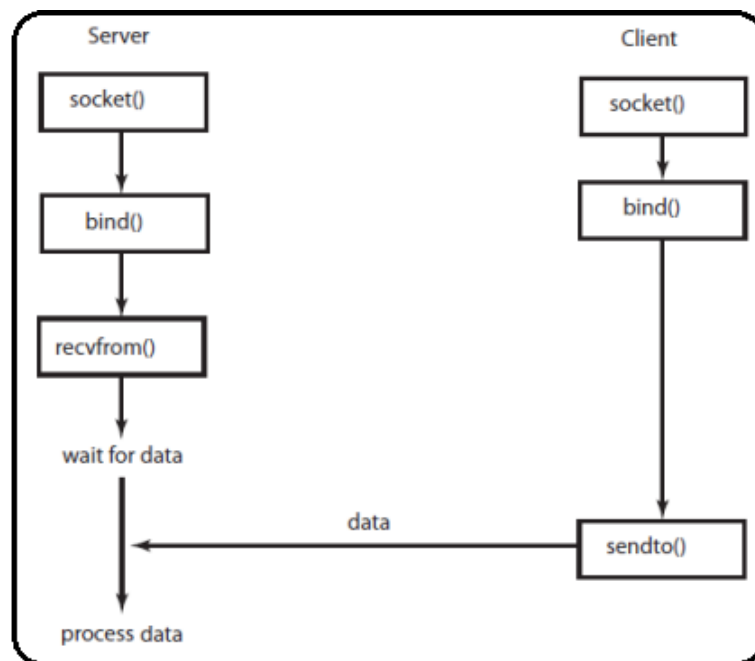


Figure 7: UDP client-server communication

### 3.3.3 Storing the data

The data extracted from the UDP datagram are stored in the form of TSV (Tab Separated Values) where each row constitutes of two columns (date/time and values). The example below is a TSV file which contains temperature as values.

date	values
2016-02-05T21:02:24	30
2016-02-05T21:02:25	30
2016-02-05T21:02:26	30
2016-02-05T21:02:27	30
2016-02-05T21:02:28	30
2016-02-05T21:02:29	29
2016-02-05T21:02:29	30
2016-02-05T21:02:30	29
2016-02-05T21:02:31	29
2016-02-05T21:02:32	30
2016-02-05T21:02:33	30

To identify the data from different boards, the values are stored in a file which corresponds to the board ID. And to identify the type of data, the file is stored in a specific folder. For example, the file `./temperature/A/board001.tsv` corresponds to the temperature data received from daughter board A of raspberry Pi (ID : 1).

### 3.4 Creating a website to display sensor data

The data stored in the TSV files are in raw form and difficult for a human-being to interpret them. Thus, I decided to display them in the form of graph.

#### 3.4.1 Graph data

We used an example from [D3 JS](#) to plot the data read from the TSV file. The graph as in below, shows the data plotted from the TSV file using javascript functions. The x-axis contains the different date/time when the data was retrieved and on y-axis contains the value corresponding to the x-axis (in this example, it's temperature).

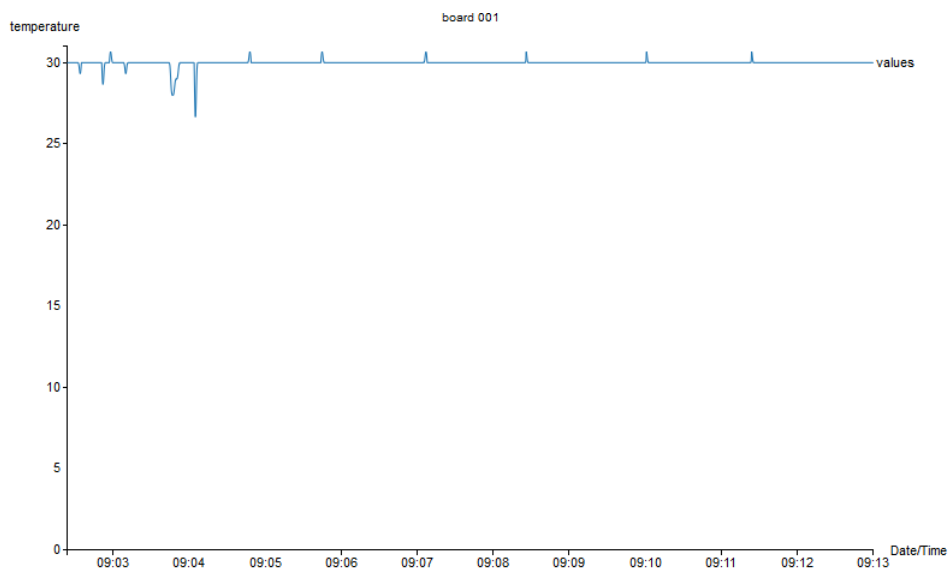
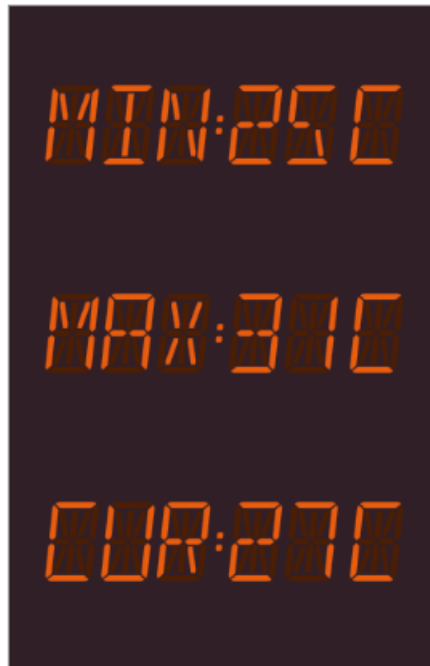


Figure 8: Graph display

### 3.4.2 Instantaneous display

We also decided to display instantaneous values (along minimum and maximum values) read from the TSV file. To do this we used the javascript from D3 JS website and for each value we will be comparing with the previous minimum or maximum values.



*Instantaneous values from board 001*

Figure 9: Instantaneous temperature value

## 4 Problems encountered

One of the important part for the automatic configuration of the boards through the network, is the database. Although, we had a database course during our 3rd year in Polytech Lille, it was not sufficient to conceptualise the idea of this project. Thus, with help from Mr Frédéric HOOGSTOEL, I was able to come up with a relational model for the database.

On the other hand, I had to collaborate with François ROCHE who worked on the electronic aspect of the project. During this collaboration, we worked mainly on a proprietary programming interface, PDI. Despite facing a lot of problems configuring PDI (due to the lack of documentation), we came out with a solution that enabled us to continue using PDI.

During the early phase of this project, we wanted to use radio communication as the main method for data exchange. However, due to the unavailability of the equipments, we could not test the radio communication.

## Conclusion

During this FYP, I was able to apply knowledges acquired throughout 3 years of studies in Polytech Lille. In addition, I also acquired non-negligeable amount of knowledge in programming especially Javascript, MySQL and PHP. I had also learned a lot about microcontrollers (eg: XMEGA) thanks to my colleague.

Even though it took sometime to learn about web programming, it was worth it since it is widely used in many situations. I am glad to be able to do this project since IoT is one of the fields that I am interested in.



## References

- [1] Unknown author. Bash scripting tutorial. <http://linuxconfig.org/bash-scripting-tutorial>.
- [2] Unknown author. Basic Makefile for AVR-GCC. <http://arduino.stackexchange.com/questions/12114/basic-makefile-for-avr-gcc>.
- [3] Unknown author. GPIO: Models A+, B+ and Raspberry Pi 2. <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/>.
- [4] Unknown author. PDI programming. [http://www.atmel.com/webdoc/avrdragon/avrdragon.pdi\\_description.html](http://www.atmel.com/webdoc/avrdragon/avrdragon.pdi_description.html).
- [5] Sandra HENRY-STOCKER. Unix: Flexibly moving files with lftp. <http://www.itworld.com/article/2833203/operating-systems/unix--flexibly-moving-files-with-lftp.html>.