Development of a Readout System for RadMON Sensors on Arduino YUN microcontroller

Author: José Cabrero Holgueras
Supervisors: Isidre Mateu, Georgi Gorine

EP-DT-DD
Experimental Physics - Detector Technologies - Detector Development

October 8, 2018
CERN

Abstract

Detector Development (EP-DT-DD)
Experimental Physics - Detector Technologies - Detector Development

Summer Student

Development of a Readout System for RadMON Sensors on Arduino YUN microcontroller

by José Cabrero Holgueras

CERN develops particle accelerators. As a side effect of running these accelerators, a radiation field is usually created on devices that are operated next to the locations where particles collide. Due to the negative effects that radiation has on the devices, it is crucial to measure it properly.

As a part of this radiation detection task, the EP-DT-DD group works on the development of new techniques to monitor residual radiation such detectors that are both accurate and cheap. Specifically, the group works on the detection of radiation through silicon-based devices.

The RadMON sensor, is one of these developments. It is a small and low-cost printed circuit board (PCB) mounting different combinations (tailored to specific measurement needs) of radFET and PiN diode dosimeters. Radiation is measured by monitoring the variations of conductivity of these devices. The readout of this sensor, is carried out by an Arduino YUN microcontroller plugged to a board and the system is called ReadMON. It provides the possibility to read the RadMON sensors by means of an Internet connectivity which permits the remote monitoring of the sensors. Up to now, the measured data were extracted by means of a LabView interface. However, this only allowed to obtain measurements but did not offer a dedicated display for the radiation data, and it and was not ready to be a production system.

The objective of this project was to create a WebServer hosted on the Arduino YUN micro-controller, that allowed to manage the sensors on the board with flexibility, to configure the parameters on the system, to visualize the data in a collective manner through graphs that allows a proper display and an interpretation of them, and to store the data on a database also hosted by the Arduino YUN micro-controller.

In this document, we first describe the set of features that are present on the system. Later, we expose the software installation process a user-manual way. Next, we describe the details of the development of the different components of the system, trying to facilitate the maintenance of the former. Finally, we conclude with the future work to be done on this system and the conclusions of the project.
## Contents

**Abstract** iii

1 Development of ReadMON Manager 1
   1.1 Server Election 1
   1.2 File Structure of the Server Side 2
      1.2.1 Modifying the Server Code 2
   1.3 Insights on the Client Side 3
      1.3.1 General aspects 3
      1.3.2 Specific aspects 3
      1.3.3 Understanding the Javascript Code 4
      1.3.4 Database Structure 4

2 Introduction To ReadMON Manager 7
   2.1 Login Page 7
   2.2 Home Page 7
   2.3 Settings Page 8
   2.4 Acquisition Settings Page 9
   2.5 Raw Data Charts Page 9
   2.6 Fluence and Dose Charts Page 10
   2.7 Download Page 11

3 Arduino Installation and Setup 13
   3.1 Pre-requisites 13
   3.2 Easy Installation 14
   3.3 Users Management 15
   3.4 Manual Installation 15
      3.4.1 Arduino Disk Expansion 15
      3.4.2 MySQL and Python Installation 17
      MySQL 17
      Python 17
      3.4.3 System Setup 17

4 Conclusions 19
   4.1 Recommended Future Work 19
   4.2 Final Thoughts 20
Chapter 1

Development of ReadMON Manager

This chapter provides an overview of what is the structure of the project. In Section 1.1 it is depicted the server election. Then, in Section 1.2, we show what is the file structure of the server. Finally, in Section 1.3 the division of the different files of the client application is described.

1.1 Server Election

In this section, we are going to outline the main reasons behind the election of Bottle.py as the framework for the development of the project. If we look at the most used frameworks in web servers, we find Django, Flask, PHP or Ruby (on Rails). As the objective was not to spend a lot of time on the implementation, Python was thought to be one of the best alternatives. In the following paragraphs, we discuss the advantages of each of these technologies.

With regard to PHP, we wanted to avoid it due to the fact that it is widely criticized because of lacking of real security support. If we want to implement a secure environment, we shall really focus on providing security to that environment.

In what respects Ruby (on Rails), we found, PHP also to be a complex framework. Due to the small time frame devoted for the development of the project, this framework appeared to be a huge time expenditure for this project mainly because of the steep learning curve.

Even though Python is not the best alternative in terms of performance at first glance, its ability to be compiled into C code makes it a great approach to solve the server implementation. However, OpenWRT OS released on Arduino YUN is dated on 2014. This means it lacks of 4 years of developments\(^1\), which is translated into the lack of support for virtual environments, PIP or Conda. All the installation attempts on Django and Flask ended up in failure since we did not manage to make a working installation for any of these two servers.

The solution to these problems was a framework called Bottle.py. Its main advantage in comparison to Django and Flask is that it lacks of dependencies, it just needs a .py file that can be used as module. The main reasons for the election of this server are:

\(^1\)We are aware of the existence of an update method for the Arduino YUN into a 2017 version, nevertheless, none of the Arduinos seemed to be able to upgrade the boot-loader to that version.
• Minimal resource consumption;
• Working on Arduino YUN;
• Python flexibility and multiprocessing capabilities (needed for the completion of the project).

1.2 File Structure of the Server Side

This file structure considers that the current working directory is /root/server/.

When we deal with the interaction with the server, the functionality was divided into different components so that related functionalities can be found together in a same file. The main files to be taken into account:

• **hello.py.** This file is the one needed to launch the server. Inside it we can find all the different URL paths to the services offered by the server. Apart from associating URLs to services, it also processes the arguments given to the URL and calls the corresponding handler for a task and returns with the corresponding answer.

• **sensors.py.** This file holds any interactions with the database. Its main functionality is dealing with the data from the database in the different formats needed by the application avoiding unnecessary computations.

• **settings.py.** This file is in charge of holding the in-memory structures that the server needs to make both the measurements and to provide the service to the user. It allows the user to establish certain settings that are not needed to be stored in the database and that are used to generate the information on the measurements.

• **parallel_process.py.** This file is in charge of performing all the measurements. When the user clicks on start measurements, the server creates a background process that holds a cyclic scheduler in charge of making the measurements and storing them into the database. Due to the lack of resources of the Arduino YUN platform, the server is limited to one process, and in case that there is a process active, the server does not allow to create a new one.

• **login.py.** This file implements the user authentication to the server as well as the session management system. This session management system was manually implemented due to the lack of one on the Bottle.py framework.

• **functions.py.** This file contains the set of functions that can be executed when a measurement is received in order to transform it. This file can be modified by the user to provide new functions to the calibrations screen. Be aware that introducing incorrect code in this file might make the server fail and not restarting itself properly.

1.2.1 Modifying the Server Code

This section aims to ease the modification of any of the components of the server. The server is strongly dependent on URLs, therefore it is an essential part to identify the URL that we are interested in modifying. Thus, we should look at hello.py and
find the associated URL.

Each URL has an associated function, where we can easily identify the processing carried out and the functions executed. If any external function is executed, we can easily identify the library where it is implemented taking a look at the first lines of hello.py, where the imports of that functions are found.

This same strategy can be applied recursively to find all the functions. It is interesting to take into account that URLs and function names try to have meaningful names. This can be used to understand what is the behaviour happening inside.

1.3 Insights on the Client Side

This file structure considers that the current working directory is /root/server/.

This section extracts the breakdown of the components of the frontend. This description is divided in general aspects for all web-pages and specific aspects of each one.

1.3.1 General aspects

This section details the main aspects to be taken into account when dealing with the frontend.

- The frontend has been developed mainly with HTML5, CSS3 (with Materialize.css framework) and Javascript (with jQuery and Materialize.js).
- The Materialize.css framework that can be found in /static/css/materialize.min.css has been modified to change the colours from the standard green provided to blue.
- Since the style from all the pages is shared, they all share the same css file which can be found in /static/css/style.css.
- The folder where all the HTML5 web-pages can be found is /html.
- The folder where all the CSS3 content can be found is /static/css.
- The folder where all the Javascript files can be found is /static/js.
- The folder where all the images for the help pages can be found is /static/img.

1.3.2 Specific aspects

This section breaks down the file dependencies for each of the URL’s made available for the ReadMON Manager. The HTML file serves to represent the static structure of the webpage, whereas the Javascript file is able to load all the content from the server as well as issuing different requests to it when any button is pressed on the webpage.

/login. The structure of this page is displayed from /html/login.html and its style is given by the Bootstrap framework.

/home. The html file of this page can be found in /html/homepage.html.
Chapter 1. Development of ReadMON Manager

/settings. This URL corresponds to the Settings web-page. The HTML code of this page can be found in /html/settings.html and the Javascript of this page can be found in /static/js/tablegen.js.

/tables. This URL corresponds to the Acquisition Settings web-page. The HTML code of this page can be found in /html/tables.html and the Javascript code of this page can be found in /static/js/tables.js.

/charts. This URL corresponds to the Raw Data Charts web-page. The HTML code of this page can be found in /html/charts.html and the Javascript code of this page can be found in /static/js/chartdata.js.

/charts2. This URL corresponds to the Dose and Fluence Charts web-page. The HTML code of this page can be found in /html/charts2.html and the Javascript code of this page can be found in /static/js/chartdata2.js.

/download. This URL corresponds to the Download Data web-page. The HTML code of this page can be found in /html/download.html and the Javascript code of this page can be found in /static/js/download.js.

1.3.3 Understanding the Javascript Code

This section aims to ease the task of someone having to read the Javascript code written in any of the mentioned Javascript files. Explaining the content of each of the files and its reasons would take too much time otherwise.

The first step is to identify the page where we have interacted, and using the explanation of Subsection 1.3.2, try to find the archives related to the corresponding file. If the issue is related to the aspect of the webpage, the modifications should be done to HTML and CSS files. However, if the dynamics of the database want to be changed, the Javascript file must be found.

Since the code might be a little unordered within those files, we always want to find the entry point of the code. The entry point in any of the Javascript files of this project is the function:

document.addEventListener( 'DOMContentLoaded' , function () {<content>})

When a web-page is loaded, the first code to be executed is the code found in the scope of that function. This means that the information of how the webpage is working can be found there and the order in which functions are executed.

1.3.4 Database Structure

This section outlines the main objects that can be found in the database and its dependencies. The database design was changed from the original one provided to focus on two points:

- Reducing the redundancy. When we deal with big amounts of data, reducing the stored data improves a lot queries and development.

- Reducing computations. If we have to perform any transformation of the data from the database, it ends up being a lot of cost in time.
The main table of the database is the *sensor* table. This table is in charge of storing the main settings of a sensor and associates it with a name. It uses the name as identifier that is related with other tables to avoid storing extra information.

The table used to store the information is called *readout*. This table stores the measurement that is done and the parameters with which it was done. Apart from that, it stores the reference to the sensor with which it was done.

The table used complementary to the *sensor* table to store the calibrations is called *calibration*. It stores a row for each of the ranges of a sensor calibration together with the function to.

Finally, we have the *user* table which stores the username and password combination for all the users of the system and it is used to authenticate them on login.
Chapter 2

Introduction To ReadMON Manager

This chapter gives a brief introduction about what can be done on each of the web-pages that the ReadMON Manager provides and their expected behaviour.

2.1 Login Page

This page is meant to authenticate the user through its username and password. It implements a system of sessions that, for security reasons, last only 1 hour. After that period of time, the user is requested to reintroduce the credentials. In any case, the changes on the server are not lost. Figure 2.1 shows a screenshot of the login page. If the user does not have a username see 3.3 to register it through the command line interface.

![Login Page](image)

**Figure 2.1: Login Page**

2.2 Home Page

After having logged in successfully, the user is redirected to the Homepage. In this page we can see the links to all the pages that we need to carry out measurements and see the different plots out of it. Figure 2.2 shows the homepage.
Chapter 2. Introduction To ReadMON Manager

2.3 Settings Page

In the settings page, the settings for the readout in the server can be defined. Each line represents a RadMON sensor. Since RadMON sensors vary a lot in its composition and channels, and the ReadMON system can address up to 8 sensors, the user has to determine define the connector to which each RadMON sensor is connected as well as the different dosimeters that compose it. By means of the red + sign button, we can add new lines in order to add new sensors to be read out by the ReadMON system. Figure 2.3 shows how the main page of the settings.
need. The system transforms the measurements to a radiation quantity measurement (Dose or Fluence) for its interpretation. If we click on **MODIFY CALIBRATIONS** another window pops up where the user is able to modify the calibration for a specific sensor. Figure 2.4 shows how the management of the sensors works.

Apart from the sensor configuration, there are four more buttons on the upper left part of the page. The first serves to store the changes performed into the ReadMON server. The second one allows to load the settings from a file, the third one serves to download the settings to a file and the final one is a help button to give information about the page.

### 2.4 Acquisition Settings Page

Once the user has established the data of RadMON sensors, the Acquisition Settings have to be established. The Acquisition Settings page serves for that purpose. It automatically generates a table with the Sensor Settings that the user has established, and then it allows the user to set a minimum readout period of each of the sensors. Once the user has the configuration ready, he can just click on the **Save Configuration** button. After that, we can click on the Start Measurements Button to start making measurements. The colour of the button changes in case that it was started correctly. Figure 2.5 shows how the frequency settings can be established. In addition to that, the server controls whether the minimum measurement time, makes the measurement feasible and provides the user with a correct answer. The user must specify a Tag, otherwise the measurements is not stored properly.

**Caution:** Be aware that a dosimeter takes a minimum time of 15 seconds to be read out, so introducing certain configurations might not be valid for the web-page. In addition to that it is mandatory to establish a tag for the set of readouts carried by the system.

### 2.5 Raw Data Charts Page

This page shows plots of the Voltage that is measured from the sensors, the current that is sent by the ReadMON board and the Temperature that was measured by the
ReadMON board. By default, the plots query the server in order to obtain new in-
formation not present on the plots each 10 seconds. However, if the user needs to 
visualize data from a previous measurement, the user can click on the LIVE button 
and a menu drops down with the options to filter. Figure 2.6 shows how this page 
is displayed.

Caution: By default, this page shows at most 20000 points. The reason behind 
this design decision is that querying more data to the Arduino causes it to kill the 
process running in it or to restart itself.

2.6 Fluence and Dose Charts Page

This page shows the plots of the RadMON Dose and RadMON Fluence depending 
on the type of the sensor determined in the settings page. As in the case of Raw Data 
Charts Page, the data is queried each 10 seconds in order to obtain new data from 
the server. It is possible to refine the query by clicking on the LIVE button. Figure
2.7 shows how this page is displayed.

Caution: By default, this page shows at most 20000 points. See 2.5 for more information.

![Figure 2.7: Fluence and Dose Charts Page](image)

2.7 Download Page

This page is a complementary page used in order to download the data as a .csv file. In case that the user wants to filter the data to be downloaded, this page provides a different mechanisms to refine the query. Since this page acts as a filter, if nothing is clicked, all the data is downloaded. Figure 2.8 depicts the different elements present on the download data page.

![Figure 2.8: Downloads Page](image)
Chapter 3

Arduino Installation and Setup

This chapter covers the different procedures that have to be carried out in order to connect the Arduino to a network and configure the server accordingly. In Section 3.1, it is covered what is needed to perform manually before the installation of the Arduino Server. Followingly, in Section 3.2, it is depicted how the system can be installed by means of the scripts provided. All the source code and scripts are available in gitlab.cern.ch/jcabrero/readmon_manager. Finally, in Section 3.4, we describe which are the steps to follow in case of a manual installation of the system.

3.1 Pre-requisites

Before starting with the installation, a manual procedure is required to introduce the Arduino to the network. If the Arduino Yun is not factory new and has already been included in the network, this step can be omitted. Also, if the Arduino Yun is connected by Ethernet interface, it is possible to omit this step.

However, it is highly advisable to connect the Arduino Yun through a WiFi connection too because it tends to make tunneling of the network interfaces through the wireless interface. In the case that the Arduino is connected through a Ethernet connection, there are cases where it is not able to tunnel the connection properly and it does not allow it to access to the internet, at least in the case of CERN Network.

The installation steps for the connection of the Arduino YUN to WiFi are the following:

1. A factory-new Arduino Yun is creates a local WiFi network that is meant to be used for its configuration. In order to do it, we must use a device that is able to connect through WiFi to this network that is usually named ArduinoYun-XXXXXXXXXXXX, where XXXXXXXXXXXX is the MAC address of the Arduino.

2. Once we are connected to the WiFi network, we need to access either http://arduino.local access point or http://192.168.240.1 access point.

3. The default password to access that site is "arduino".

4. We can click on Configure and then select our WiFi Network.

5. We shall also establish the REST API access to be Open Access instead of Password Protected.

6. Finally click on Save Changes to end the configuration.
3.2 Easy Installation

For a complete setup of the server, a set of scripts are provided. These scripts aim to ease the task of deploying the server and performing the initial configuration. In principle, these scripts should be enough to make the full installation, however, it always exists the possibility of having to do it manually because of any issue. In this section, we develop the easy installation process. This installation process requires to have a terminal emulator such as Putty, but it is highly recommended to use MobaXterm. However, this tutorial will work in both cases. The scripts and source code can be found in gitlab.cern.ch/jcabrero/readmon_manager.

1. First we need to place the configuration files in our Arduino. Since it has a remote connection, the first step is to send all the data necessary for the installation. For that, we will need to know the route of the arduino_conf folder that we just downloaded, since the installation of the database works only from that directory, and we will execute:

   $ scp -r <path_to_arduino_conf> root@<arduino_ip_address>:/root/

2. With everything already correctly placed, we execute the script install_part1.sh. This script carries out the Arduino YUN disk expansion. In order to execute it, we need to be in the path were it is found. Warning: This script, formats the internal storage of the SD card deleting everything present inside it. The commands to execute are:

   $ cd /root/arduino_conf
   $ chmod 755 install_part1.sh
   $ ./install_part1.sh

   With these commands, the first part of the installation is started in order to expand the Arduino YUN disk space. It will probably prompt several errors, however, they are due to some actions that the process performs preemptively. If the program finishes and reboots the Arduino, it should mean that the first part installation was successfully carried out.

3. Once the Arduino has rebooted, we already have the disk expanded and can proceed to the installation of the rest of the elements. Again, we are going to the directory where every of our scripts are placed. Then we will execute the second part of the installation.

   $ scp -r <path_to_server> root@<arduino_ip_address>:/root/
   $ cd /root/arduino_conf
   $ chmod 755 install_part2.sh
   $ ./install_part2.sh

4. Once the we the Arduino reboots, we have a working environment for the ReadMON Manager. We would just need to create a new user in order to authenticate. In order to prevent anyone from using the system in an unauthorized way, the creation of the users is done through the command line interface, however, it is possible to access the server in a default manner with the following combination:

   1In case we do not know the IP Address that the Arduino has taken, we can use the YunTerminal-Monitor sketch that is provided on the official webpage and the ifconfig command to obtain it.
   2It is strongly recommended to remove the default user and create a new user on the database with a different password.
3.3 Users Management

In this section, it is explained the steps need to be carried out in order to add and remove users from the database.

In order to add a new user we execute the following commands:

```
$ cd arduino_conf
$ chmod 755 user.sh
$ ./user.sh add <username> <password>
```

In order to remove a user from the database:

```
$ cd arduino_conf
$ chmod 755 user.sh
$ ./user.sh remove <username> <password>
```

3.4 Manual Installation

This manual installation guide is only recommended in case that the other installation guide is not properly working. The installation guide is divided in three main parts Arduino Yun Disk Expansion covered in subsection 3.4.1, MySQL and Python Installation covered in 3.4.2 and System Setup covered in 3.4.3.

3.4.1 Arduino Disk Expansion

In case that the easy installation procedure fails, it is recommended to use the official supported Arduino Yun Disk Expander. It can be downloaded from this link. Now the steps to carry out a manual configuration will be listed in script form. It is not advisable to do it manually since it requires certain expertise and deals directly with the partition tables.

```
# If any of this commands fails, there is an error before starting.
mount | grep ^/dev/sda | grep 'on /overlay'
df / | awk '/rootfs/ {print \
$4}'
ls /mnt/sda1

# Installing software
opkg update
opkg install e2fsprogs mkosfs fdisk rsync

# Partitioning and Formatting
umount /dev/sda?
rm -rf /mnt/sda?
dd if=/dev/zero of=/dev/sda bs=4096 count=10
```
### First Partition

#### 8192M means the first partition will have 8GB of space

```
(\texttt{echo n; echo p; echo 1; echo; echo +8192M; echo w}) | \texttt{fdisk /dev/sda}
```

```
\texttt{umount /dev/sda?}
\texttt{rm -rf /mnt/sda?}
```

### Second Partition

```
(\texttt{echo n; echo p; echo 2; echo; echo; echo w}) | \texttt{fdisk /dev/sda}
```

```
\texttt{umount /dev/sda?}
\texttt{rm -rf /mnt/sda?}
```

### First Partition to FAT32

```
(\texttt{echo t; echo 1; echo c; echo w}) | \texttt{fdisk /dev/sda}
```

```
\texttt{umount /dev/sda?}
\texttt{rm -rf /mnt/sda?}
\texttt{sleep 5}
\texttt{umount /dev/sda?}
\texttt{rm -rf /mnt/sda?}
\texttt{mkfs.vfat /dev/sda1}
\texttt{sleep 1}
```

### Second Partition to ext4

```
\texttt{mkfs.ext4 /dev/sda2}
```

# Create Arduino Folder

```
\texttt{mkdir -p /mnt/sda1}
\texttt{mount /dev/sda1 /mnt/sda1}
\texttt{mkdir -p /mnt/sda1/arduino/www}
\texttt{umount /dev/sda?}
\texttt{rm -rf /mnt/sda?}
```

# Copy Files From Yun to SD

```
\texttt{mkdir -p /mnt/sda2}
\texttt{mount /dev/sda2 /mnt/sda2}
\texttt{rsync -a --exclude=/mnt/ --exclude=/www/sd /overlay/ /mnt/sda2/}
\texttt{umount /dev/sda?}
\texttt{rm -rf /mnt/sda?}
```

# Enabling Ext Root

```
\texttt{uci add fstab mount}
\texttt{uci set fstab.@mount[0].target=/overlay}
\texttt{uci set fstab.@mount[0].device=/dev/sda2}
\texttt{uci set fstab.@mount[0].fstype=ext4}
\texttt{uci set fstab.@mount[0].enabled=1}
\texttt{uci set fstab.@mount[0].enabled_fsck=0}
\texttt{uci set fstab.@mount[0].options=\textit{rw, sync, noatime, nodiratime}}
\texttt{uci commit}
```

# Reboot the system

```
\texttt{reboot}
```
3.4. Manual Installation

3.4.2 MySQL and Python Installation

This section describes how to install and setup the database for the Arduino Yun readout system. It includes all the tables in the database in order to be able to deal with it. The first step is to carry out the installation of the MySQL Database.

**MySQL**

```bash
opkg update
opkg install mysql-server

# Creating directories
mkdir -p /mnt/data/mysql
mkdir -p /mnt/data/tmp

# Installing MySQL
mysql_install_db --force
/usr/bin/mysqld --skip-grant-tables --skip-networking &
# First configuration of database...
mysql -uroot -e 'source /root/arduino_conf/mysql_install/setup_db.sql'

# Creating database service
/etc/init.d/mysqld enable
/etc/init.d/mysqld start

# Generating RADMON database
cd /root/arduino_conf/
chmod 755 /root/arduino_conf/mysql_install/db.sh
/root/arduino_conf/mysql_install/db.sh all
```

**Python**

For the installation of the Python, we need to execute the following script:

```bash
# Installing python necessary libraries
opkg update
opkg install python python-json python-mysql
```

3.4.3 System Setup

In this section it is described the set of modifications that have to be carried out on the OpenWRT system for the server to work properly. The steps needed are the following:

```bash
# Moving uhttpd server to port 8000
mv /etc/config/uhttpd /etc/config/.uhttpd.old.conf
cp /root/arduino_conf/config/uhttpd /etc/config/

# Setting up server initialization
cp /root/arduino_conf/config/readmon_manager /etc/init.d/
chmod 755 /etc/init.d/readmon_manager
/etc/init.d/readmon_manager enable
```
# Rebooting
reboot
Chapter 4

Conclusions

This chapter serves as a conclusion to the work. Section 4.1 indicates the guidelines to follow in case that the project want to be perfectioned. Finally, section 4.2 draws the final thoughts of this project.

4.1 Recommended Future Work

This sections aims to indicate the main work lines that this software might have to follow, in case that someone wants to continue it.

- The project is developed under HTTP because of a physical limitation of the Arduino YUN. This is something to be taken into account, and in case, that it is considered, implementing a secure system with end-to-end encryption is suggested.

- Bottle.py does not implement a session system, thus, to create the login page, a new system was created. However, revisiting the sessions system is something that is highly recommended to be done.

- The server was protected, both on the server and client side, as much as it was possible, however, a second visit to protect and secure the inputs shall be done. Anyways, all the inputs from the user are escaped in order to prevent SQL Injection.

- In case of using Arduino YUN, I highly recommend migrating the database from MySQL to SQLite3, which is a much more appropriate system to be hosted on an Arduino YUN because is better Database Management System for smaller and limited devices.

- Perfecting the interaction between the two cores of the Arduino, doing through the URL might not be the most effective option, given the fact that it is present the

This following lines aim to present, an alternative which could improve the quality of the ReadMON Manager. This project was developed in an Arduino YUN by explicit indication in the project guidelines. This highly influenced several of the decisions during the elaboration of the project and was a limiting factor in many cases.

After having dealt with Arduino YUN, I do not consider it the best platform for this project. It is a platform meant for Internet of Things(IoT) projects, this implies that it is not meant to host a server with such intensive calculations, but more to be used in a more basic way. This could easily be improved by combining an Arduino
UNO and a Raspberry Pi 3 B, or even, a Raspberry Pi 3 B alone. The following lines present the advantages of having used this alternative:

- Ability to use Django.
  - This would have speed up noticeably the implementation and would have reduced a lot of the effort made.
  - It already implements a secure session system and HTTPS securing the communications.
  - It provides Jinja, a system that would have prevented to load manually all the data of the frontend manually from the database.

- Faster processor would make faster queries.

- More memory that would permit making bigger queries without having to sub-sample data manually.

- Cheaper.$^1$

### 4.2 Final Thoughts

Despite of the fact, that this project was bigger and more complex than it appeared at the beginning, it was a tremendously enriching experience. It allowed me to deal with a low-level device such as the Arduino YUN and find a solution to a really complex problem.

Apart from that, it allowed me to work really closely with a group of people which came from different disciplines and opened my mind to think in different ways and to solve problems in the most efficient way. But the most important thing is that I was able to know and live CERN from inside, a unique experience.

---

\(^1\)Considering the price of Arduino YUN to be 49€, the price of a Raspberry Pi 3 B 30€ and the price of Arduino UNO 20€