Open Sensor Networks

Alejandro Andreu Isábal
Internal Workshop
alejandro.andreu01@estudiant.upf.edu

***Abstract*—This brief paper explains how I have been doing with my degree final project. In particular, I show which sensor board I chose and what I have learned about them as well as some sensor integration.**

***Index Terms*—WSN, BuB, Arduino.**

I. Sensor board choice

INITIALLY we had lots of Crossbow TelosB rev. B nodes ready to use in the laboratory so this was quite a good option to consider. They run over an embedded UNIX-based operating system —TinyOS—, which has several advantages such as default multithreading, advanced customization options, etc. However this board lacks the community and good documentation that other platforms such Arduino and its derivatives do have. Also, complexity when using this platform is high.

Hence contemplating another options such as the one mentioned before is a must. Arduino, while maintaining simplicity is as well a very powerful platform used by many DIYers today. This, along with its open-source philosophy has created a large community that has been very active for the past 4 years or so. Moreover, many companies create modules or shields for this prototyping platform so we can have something working very fast. It does support Zigbee which was a prerequisite as I explained in the last workshop. Although Arduino is not specifically meant to from WSNs —wireless sensor networks— it can be adapted and tuned to consume less energy.

Therefore, being complexity my main criteria I chose Arduino as my sensor board to complete this project.

II. Data gathering

THE first step before designing the wireless sensor network itself is learning how to program an Arduino board and how to fetch data from different sensors, and this is what I spent my time on for the last two weeks. This board uses a programming language very similar to C++ with slight modifications. It is very easy and fast to develop a sensing application.

Furthermore, I also tested almost every module I wanted to use: temperature, humidity, light intensity, sound detector and GPS module. Air quality which is a key measurement is the only one that I haven't still tested.

III. Future work

NEXT steps involve, as I mentioned before, measuring air quality. This data could be gathered by just one sensor or several.

Also, testing point-to-point communication with XBee modules is the next thing in my schedule.

Key aspects and Main factors in NQN

Fernando Gros González
C4EU BuB Open Workshop
fernandogrgo@gmail.com

Abstract—**This is a short paper that tries to explain the most relevant elements that I have identified in the NQN Pilot. As the first meeting has not taken place yet, I have tried to do my best by analyzing the problem and extracting the key points.**

I. Introduction

TH e NQ is home to a wide range of SMEs from many sectors, including digital creatives, artist creatives, cafes and restaurants, galleries, fashion designers, bars and clubs, craft centres, architects and many more.

The relatively low rental and lease costs of many of the buildings in the NQ allow for starting businesses to have a physical trading presence in Manchester city centre that is very close to high footfall commercial areas

II. Public WiFi impact in the area

Providing public WiFi to the NQ will allow businesses to increase their revenues by increasing the number of customers and will give them a way to promote a big range of activities and/or events taking place in the zone. In addition, it is likely that this facts help to support the economy of the NQ area and of the whole city.

The NQ area is like a small village in the centre of Manchester where most of the small businesses know each other, and work together to strengthen the economy of the city. This is an important relationship that can be intensified by the implantation of that network, and so the economy will be boosted.

III. Pricing definition

One of the most important aspects of the project, apart from designing and deploying the network, is defining a good pricing model for commercial use. It is a basic point, because it is crucial that the network becomes self-funding and sustainable after the conclusion of the C4EU project.

This is going to be a very important thing, and I will have to find out the best solution in order to make the best approach.

IV. Regulatory issues

- Planning permission (mitigated by involvement of CityCo and Manchester City Council)
- Radio licensing (mitigated by use of equipment that does not require licensing)
- Data protection (mitigated by terms and conditions of use of the network)
- Constitution of the Cooperative (MDDA has local experts in that)
- Health and Safety (MDDa is well aware of the requirements for installing equipment, etc.)
- Public Liability Insurance
- Where relevant, adhere to English law in relation to internet service provision

V. Conclusions

The aspects described in this paper have been identified just by reading the pilot proposals document of the C4EU project. It is likely that I have to refocus the point of view I have given to the pilot once I have my first meeting with the MDDA colleagues.

Practicum, Mentor, Thesis, Advisor

Jaume Barcelo

***Abstract*—The training of the students in the Commons for Europe project has two differentiated sides: practicum and thesis. The practicum is about real-world work while the thesis is about academic work. The two aspects require a different guiding person: a mentor for the practicum and an advisor for the thesis. In this paper we provide details about this two complementary aspects of the training of the students in the project.**

***Index Terms*—Bottom-up broadband, training, mentor program, documentation**

I. Introduction

THE students selected to participate in the Commons for Europe (C4EU) project do so as part of their education at the university. Specifically, this training is divided in two different blocks: *practicum* and *degree thesis*. The practicum involves real-world work in which the students have the opportunity to use the skills they have learned in regular courses. It is also the opportunity to realize that real-world work is far away from the courses taught at the university, which means that the students have to make an extra effort to get acquainted with technologies and work-flows that they have not learned in class.

The *practicum* is not a controlled environment as the course lab assignments are. Things can go wrong, and it is important to understand and accept it. Furthermore, there is not a teacher that *knows the solution*. This means that the level of effort to achieve results is much higher in the practicum than in a course assignment, as it is possible to get stuck and it may take days or longer to find a solution or a workaround. The effor is measured in the European Credit Transfer System (ECTS). The *practicum* has a value of 20 ECTS credits, which is equivalent to 400 hours of work.

The students are not alone in this quest. A *mentor* is assigned to each student to indicate the tasks that the student has to do and provide the necessary help and guidance. As the practicum is tied to a real-world work, the *mentor* needs to be someone that has been working in this real-world for some time.

Besides the actual technical skills acquired in the execution of the *practicum*, the students are also expected to practice *soft* skills such as participation in meetings, effective communication, organization of work to meet schedules, generation of documentation, etc. For some people, the practicum can be the starting point of a professional career.

In the next months, we will assign a mentor to each of the student participating in the C4EU project. It is important that the mentor is someone from outside the university that is very familiar with bottom-up-broadband and with the pilot. We have already taken the decision for two students:

- Nacho Justel will be mentored by Giovanni Calcerano in the FreeEuropeWiFi pilot.
- Jorge Beltran will be mentored by Roger Baig in the fiber-from-the-home (FFTH) pilot.

In addition to the *practicum*, the students also have write their *degree thesis*. This thesis is an academic document that is necessary to obtain the degree. In the thesis, the students will comprehensively describe their pilot. As an academic document, it has to be carefully written, well structured and profusely documented. It is necessary to include introductory material, related work and references. It is also important to include a detailed work-plan with descriptions of the tasks. The work should be described in such a way that an external evaluator can understand what is the contribution and why it is important.

The *thesis* has also a value of 20 ECTS credits, which means 400 additional hours of work. This part of the work will be supervised by an academic advisor from the university. There is hard deadline for the *thesis* in June. Not meeting this deadline would represent a delay of one year in the obtention of the degree. For this reason, it may be a good idea to plan the work in such a way that the thesis is finished considerably earlier, to have some *safety margin* in case of unexpected events.

LaTeX is a popular document preparation system in the academia, that we will also use in the preparation of the thesis. It is convenient to structure a large document in chapters, sections and subsection. It also provides support for references and cross-references. And automatically generates tables of contents, tables of figures, bibliography, etc. Our idea is to use LaTeX also for the preparation of the documentation of the C4EU project, in such a way that it can be re-used in the preparation of the thesis of the students.

Another tool that can be helpful in the preparation of the documentation is github. Github is a web based extension of the git revision control system, and makes it possible that different people work in parallel on the same document, suggest changes, rollback modifications, etc. in a distributed fashion.

Acknowledgment

This work has been partially funded by the European Commission (grant CIP-ICT PSP-2011-5). The views expressed in this technical report are solely those of the author and do not represent the views of the common4europe project or the European Commission.

Jaume Barcelo is with Universitat Pompeu Fabra

FTTF/FTTP

Jorge A. Beltrán Calderón

October 2, 2012

El proyecto C4EU Rubí está sufriendo una serie de retrasos y dificultades entre el personal de Guifi.net y el ayuntamiento de Rubí, por lo tanto, enfocaré mi proyecto al estudio para el desarrollo e implementación de fibra óptica bajo el modelo BuB.

Para empezar, realizaré un estudio detallado sobre la fibra óptica desplegada en Gurb bajo el modelo BuB; hace poco tiempo ya se está dando cobertura de Banda Ancha con Fibra óptica (a 1 Gigabit) a unas cuantas casas y empresas del sector norte de Gurb. Este estudio incluye en detalle la situación donde se ha llevado a cabo el proyecto, los usuarios, la tecnología y material utilizados, la configuración e implementación, los permisos y contratos llevados a cabo (bajo el modelo BuB) y la legalidad y conformidad del proyecto.

La realización de este estudio proporcionará información para realizar la implementación en otros municipios, como por ejemplo, Rubí.

Wireless Data Transmission with Ettus USRP-E110

Luis Sanabria-Russo
2^{nd} NeTS Internal Workshop
luis.sanabria@upf.edu

***Abstract*—The unused channels at the UHF TV band (or TV White Spaces) are taking much attention nowadays, mainly due to its better capabilities on building penetration and propagation characteristics than those experienced at 2.4 GHz or 5 GHz bands used by the set of IEEE 802.11a/b/g protocols. In this technical report, data transmission over TV White Spaces is attempted using the open source Software Defined Radio (SDR) project GNURadio and the Ettus USRP-E110 as a radio front-end. Results show successful reception of binary files at small distances.**

***Index Terms*—GNURadio, TV White Spaces, USRP, Transmission.**

I. Introduction

As presented in the previous workshop [1] [2], now we have the means to sense the surrounding area in the search for TV White Spaces (TVWS). These *spectrum holes* are going to be used for data transmissions in the upcoming IEEE 802.11af and 802.22 standards, also known as WhiteFi and Super Wi-Fi respectively.

Now, it is time for transmission. This is not trivial, sending data over the air is a joint effort of multiple software layers (as in the OSI model), each one responsible of key tasks that ensure some levels of efficiency in the delivery. GNURadio [3] provides a set of examples to perform wireless data transmission with USRP-E110 [4].

Our current approach is described in Section II, preliminary results are displayed in Section III while the challenges ahead compose Section IV.

II. Our current approach

This is not our first report on data transmission. In [5] we successfully achieved two way communication between USRPs. Also, we were able to transmit 1 and 10 MBs binary files at 100 kbps with no errors over a distance of three meters, thus confirming the PHY platform provided by GNURadio.

This technical report provides an insight on our attempt to effectively transmit a 2 MB JPEG image over a distance of three meters using TVWS and USRP-E110. The hardware components involved are: two USRP-E110 [4] each one equipped with a WBX daughterboard [6] and two log-periodic antennas [7]. The network layout is shown in Figure 1, where a PC in the same subnet as the USRPs is used for configuration and management.

The GNURadio examples used to perform the transmission and reception are `benchmark_tx.py` and `benchmark_rx.py` respectively, which can be found under `/digital/narrowband/` of the GNURadio examples' directory.

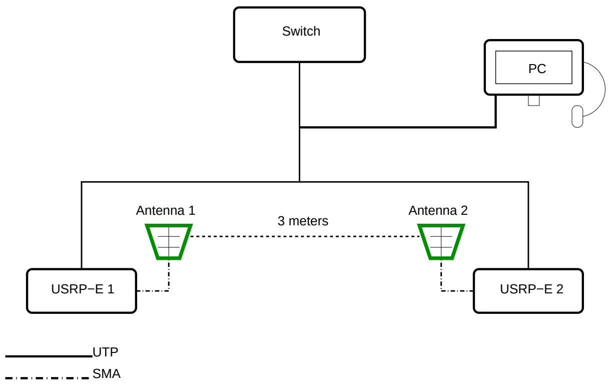

Fig. 1. USRP-USRP connection layout

III. Results

By default, the previous mentioned GNURadio examples are able to transmit and receive a sequence of zeros of a determined length; which can be defined by the user. Nevertheless, our attempt is to transmit a whole 2 MB JPEG image.

`benchmark_tx.py` has an option that allows transmission of data from a binary file. As JPEG images are binary by nature, the example code just had to be modified so it would construct data frames composed of actual bits from the image, instead of just packet sequence numbers as was set by default.

The modification of the transmitter code correctly formed and sent packets composed of bits from the image file. Nevertheless, at the receiver side the image appeared to be incomplete (see Figure 2).

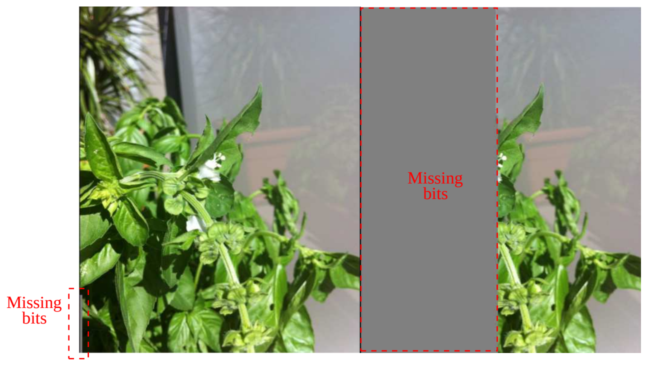

Fig. 2. Two examples of incomplete/corrupted received images

Because of the effect on the received image, the example code at the receiver needed to be adjusted to the task. This adjustment included the identification of the last packet to be received, which resulted in an effective reception of the sent image (see Figure 3).

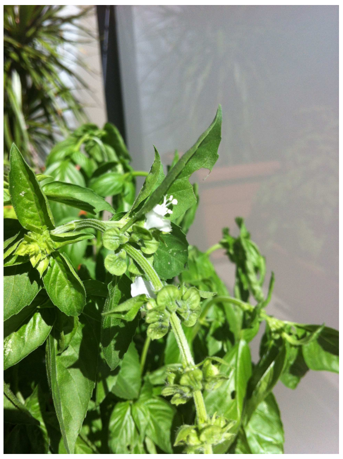

Fig. 3. Correctly received image

Successful transmission and reception was achieved using GMSK modulation, with four constellation points and a physical rate of 200 kbps.

The modified code can be found at [8], along with the instructions of how to implement it.

IV. Challenges

Now that we effectively sense the spectrum in the search for TVWS and are able to make data transmissions over these frequencies, the next challenge is to combine these activities into one continuous action.

In order to achieve this, our efforts are directed at building a single GNURadio example code able to perform the sensing and transmission/reception tasks.

Some of the involved challenges relate to signaling of the selected free TV channel for transmission and a carrier sense algorithm that would account for retransmissions and bit error rate (BER).

References

[1] *Spectrum Sensing with USRP-E110*, ser. MACOM 2012 LNCS proceedings, no. 7642. Springer, September 2012.

[2] Sanabria-Russo, L., "Spectrum Sensing with USRP-E110," http://bit.ly/PKtbSc, Tech. Rep., July 2012, 1st NeTS Internal Workshop.

[3] Blossom, E., "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, vol. 2004, pp. pp 4–, 2004.

[4] Ettus Research, "Universal Software Radio Peripheral (USRP) E110," https://www.ettus.com/product/details/UE110-KIT.

[5] Sanabria-Russo, L., "TCP/IP communication between two USRP-E110," http://arxiv.org/abs/1209.2635, Tech. Rep., July 2012, NeTS Technical Report.

[6] Ettus Rsearch, "WBX USRP RF Daughterboard," https://www.ettus.com/product/details/WBX.

[7] ——, "Log Periodic PCB Antenna," https://www.ettus.com/product/details/LP0410.

[8] L. Sanabria-Russo. USRP Transmissions: exchanging binary files with USRP and GNURadio. Code repository at Github. [Online]. Available: http://bit.ly/SXObn2

OpenWISP Modules

Ignacio A. Justel Pizarro

October, 2012

Abstract

The next paper will resume the module structure of **OpenWISP** (Open Wireless Internet Service Provider) in order to help to understand how does it works.

Contents

OpenWISP project is actually divided in five important modules or blocks, but there are also another items strictly related with this project. Let's review them, one by one:

$\boxed{1}$ **OpenWISP User Management System:**
It is a Ruby on Rails application, that interacts directly with the users. By using this module, the user can get access to the OpenWISP global system to get internet access, password recovery if needed and manage his account.
This module is directly related with third party modules as the FreeRadius server with a MySQL database to sign-up/sign-out and save user information. Once the user is registered in the service, and every time he tries to connect to the network, he will be asked to introduce his username and password to start surfing Internet. This application has been designed to be integrated with RADIUS authentication solutions.
At this moment, OpenWISP User Management System is being developed to be integrated into FreeRADIUS v2.1, so other implementations are not currently supported.

Some of its features are:

- User registration via mobile phone, ID card or credit card (Paypal IPN).
- User interface supporting most common browsers including mobile web browsers.
- Password recovery via mobile phone or email.
- Statistics of generated traffic per user.
- User administration via dedicated admin interface.
- Admin interface supports role based administration (operator, admin, superadmin).

- Via admin interface, new users can be added, modified, enabled/disabled or monitoring traffic per user/nationality and logins/registration.
- English and Italian language translations at the moment.

$\boxed{2}$ **OpenWISP Geographic Monitoring:**
It is a Ruby on Rails - HTML 5.0 based, web GUI that allows the management staff to get information about the geographic information of the deployed access points. It renders a geographic map of the status your networks: access point up/down/unknown (if an access point has an "unknown" status, then it would not be able to download the configuration to connect to the system from the OpenWISP Access Point Manager.

$\boxed{3}$ **OpenWISP Captive Portal Manager:**
It is a captive portal written from Scratch with Ruby on Rails. It allows the user surfing the Internet by enabling rules in the server firewall. This module is the operation center of all the system, so all the data that is generated or has as destination the system, will pass through this module.

Its main features are:

- Multiple captive portals (one per physical or virtual interface).
- RADIUS / Local authentication.
- Experimental traffic shaping per user.
- Multiple OS support.
- IPv6 support can be easily implemented.

$\boxed{4}$ **OpenWISP Access Point Manager:**
It is a Ruby on Rails based web GUI. This module allows the management staff to configure, monitorization and support of deployed access points. It also stores value information about the network, as the amount of traffic each VPN generate, MAC addresses, geographical addresses and network setting, among other data. The access points download the configuration and settings from this module to establish a connection to the gateway.

$\boxed{5}$ **OpenWISP Firmware:**
It is a set of scripts (shell and web cgi) that sits on top of OpenWrt[1]. OWF is a no visible module of the system. It is the firmware that every access point has installed to be able to connect to the network. Every time a new access point is connected to the network, it will download the settings from the Access Point Manager, as before explained. If an access point is rebooted without network connectivity, will get no configuration until it could establishes a connection with the OpenWISP Manager, so then, the configuration will be sent to it.
OpenWISP firmware provides a daemon for retrieving the configuration of some components at the manager:

- WiFi (currently only for madwifi-ng and ath9k drivers).

[1]OpenWrt is a linux distro for embedded devices that provides a fully writable file structure system with package management. This allows to customize devices keeping freedom from vendor configuration.

- Networking.
- Data layer traffic shaping.
- OpenVPN (data layer, TAP).
- Unix cron jobs

Far from only providing a daemon, it also has a web GUI where:

- Configuring basic net structure
- Network parameters.
- Configuring basic OpenWISP server settings.
- Performing test and technical fails menu.

Also there exists another application based on Ruby/Sinatra that communicates or interacts with the other modules using RESTful services. It is called **OpenWISP Middleware**.
This module has different tasks to do depending on the other module that is talking with.

- By communicating with the OpenWISP Captive Portal Manager, it personalizes the captive page for a group of different access points.
- On the other hand, by interacting with the OpenWISP User Management System, OpenWISP Geographic Monitoring and OpenWISP Access Point Manager, it provides very useful information in a full duplex way.

Just remember OpenWISP is an open minded, free source project, so "optional" modules can be replaced for your owns.