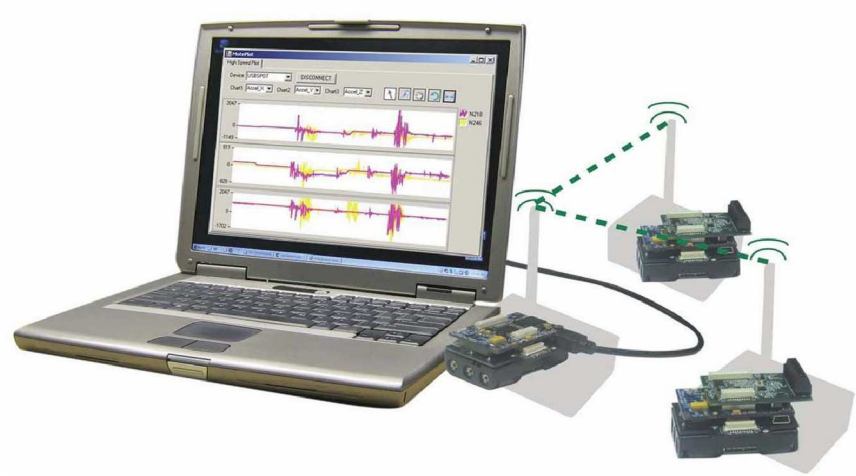
Internship report
From 06/09/2008 to 08/29/2008

ROUAISSIA Rida
Second year INPG Telecom



Development of a software infrastructure for experiments on Wireless Sensor Networks

INRIA, Montbonnot
Project leaders : Christophe Braillon - Roger Pissard-Gibollet

Contents

List of Figures

Word of gratitude

I would like to thank all the people who helped me during that internship, especially my project leaders Christophe Braillon and Roger Pissard-Gibollet, Jean-Francois Cuniberto who were very friendly and always ready to help me.

I also would like to thank the others students in internship, Abdel Griche, Rémi Tassan and Mickael Pontal, who integrated me in the team and were always very nice with me.

Introduction

This internship was done from 06/09/2008 to 08/29/2004 at:

INRIA Rhône-Alpes
655 Avenue de l'Europe
Montbonnot
38334 Saint Ismier cedex
France

I've done my internship in the SED service of the INRIA (the French national institute for research in computer science and control). One of the main role of the SED service is to develop hardware and software tools for experiments on Wireless Sensor Networks for research project-teams. My project leaders were Christophe Braillon (research engineer) and Roger Pissard-Gibollet (research engineer).

Wireless Sensor Networks (WSN) are regarded as special ad-hoc networks. The nodes of this networks are composed of many sensors capable of receiving and transmitting data in an autonomous way. The position of these nodes is not necessarily predetermined. They are randomly scattered through a geographical zone, called *field of catchment*.

In advanced Wireless Sensor Networks data are forwarded, using a multi-hop routing, to a special node called *"the sink"*. This node can be connected to the network via serial link, satellite or Internet. So, the user can send requests to the others network nodes, clarifying the required data type, and he can receive the environmental data by using the sink.

Wireless sensor networks applications are very numerous and are a set of ery active research subjects. Sensors size getting reduced, cheaper, WSN invade several application domains. Sensor networks can be very useful in various applications dealing with collecting and handling environmental

data. Among the domains where these networks can be very useful, we can quote: military, environmental, domestic, health, security, etc. It's used, for example, to do monitoring and motion capture.

Wireless sensors are microcontroller-based electronic boards that embedded a radio link (IEEE 802.15.4, ZigBee, CC1100). To these basic boards are added daughter boards constituted by a set of sensors (microphone, accelerometers, temperature, light...).

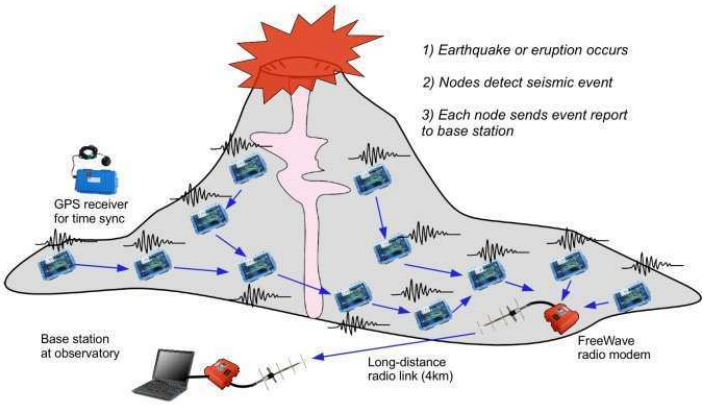


Figure 2.1: Wireless Sensor Network example : Volcano

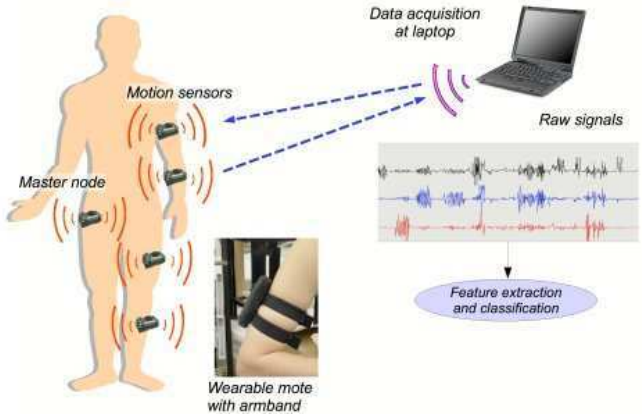


Figure 2.2: Wireless Sensor Network example : Motion capture

The objective of this internship was to develop a software infrastructure allowing to unify tools for experiments on wireless sensor networks for the SensLab project.

The purpose of the SensLAB project is to deploy a very large scale open wireless sensor network platform. SensLAB's main and most important goal is to offer an accurate and efficient scientific tool to help in the design, development, tuning, and experimentation of real large-scale sensor network applications. The SensLAB platforms will be distributed among 4 sites and will be composed of 1024 nodes. Each location will host 256 sensor nodes with specific characteristics in order to offer a wide spectrum of possibilities and heterogeneity. The four testbeds will however be part of a common global testbed as several nodes will have global connectivity such that it will be possible to experiment a given application on all 1024 sensors at the same time. If deployed, SensLAB would be a unique scientific tool for the research on wireless sensor networks.

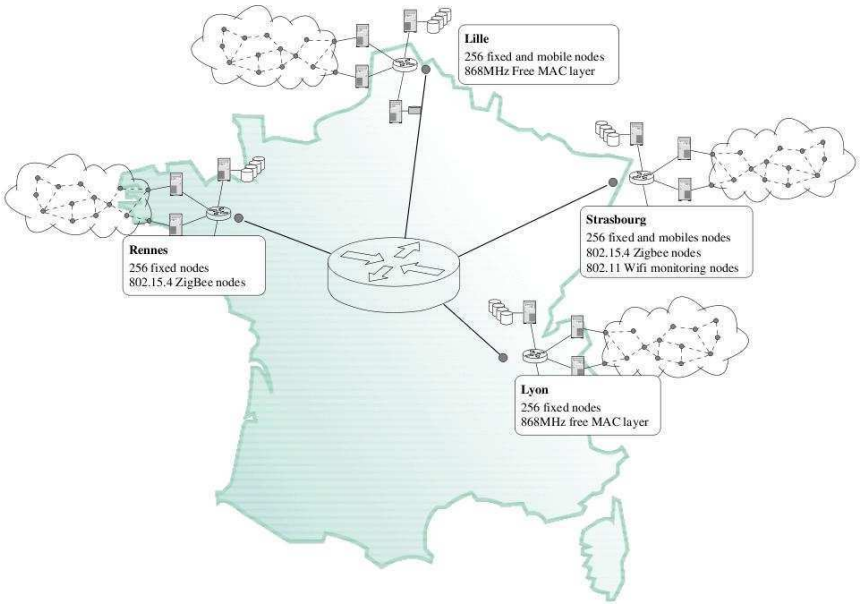


Figure 2.3: SensLAB national wireless grid testbed

It's necessary to set up a sending and routing strategy to use the radio link in an optimal way (avoiding the 250 Kb/s connection saturation). Another problem of wireless sensor networks is the energy consumption. Boards are powered by small batteries (e.g. 2xAA size batteries) so it's necessary to optimize the microcontroller use to save energy. Indeed, for a WSN, we wish an autonomy from 3 to 4 years with this battery type.

This report is composed of 3 main parts:

- Internship localization :
  In this part I present the INRIA and the SED Team.
- Working environment :
  Here I explain the software and hardware tools I've used.
- Project realization :
  Finally I detail here all I've concretely designed during my internship.

Internship localization

3.1 Presentation of the INRIA

INRIA, the French national institute for research in computer science and control, operating under the dual authority of the Ministry of Research and the Ministry of Industry, is dedicated to fundamental and applied research in information and communication science and technology (ICST). The Institute also plays a major role in technology transfer by fostering training through research, diffusion of scientific and technical information, development, as well as providing expert advice and participating in international programs.

Figure 3.1: INRIA - Rhone-Alpes

By playing a leading role in the scientific community in the field and being in close contact with industry, INRIA is a major participant in the development of ICST in France. Throughout its eight research centers in Rocquencourt, Rennes, Sophia Antipolis, Grenoble, Nancy, Bordeaux, Lille and Saclay, INRIA has a workforce of 3 800, 2 800 of whom are scientists from INRIA and INRIA's partner organizations such as CNRS (the French National Center for Scientific Research), universities and leading engineering schools. They work in 150 joint research project-teams. Many INRIA researchers are also professors and approximately 1 000 doctoral students work on theses as part of INRIA research project-teams.

INRIA develops many partnerships with industry and fosters technology transfer and company foundation in the field of ICST - some ninety companies have been founded with the support of INRIA-Transfer, a subsidiary of INRIA, specialized in guiding, evaluating, qualifying, and financing innovative high-tech IT start-up companies.

INRIA maintains important international relations and exchanges. In Europe, INRIA is a member of ERCIM which brings together research institutes from 19 European countries. INRIA is a partner in about 120 FP6 actions and 40 FP7 actions, mainly in the ICST field. INRIA also collaborates with numerous scientific and academic institutions abroad (joint laboratories such as LIAMA, associated research teams, training and internship programs).

INRIA has an annual budget of 186 million Euros, 20% of which comes from its own research contracts and development products.

The Institute's strategy closely combines scientific excellence with technology transfer. INRIA's major goal for 2008-2012 is to achieve scientific and technological breakthroughs in several priority domains:

- Modeling, simulation and optimization of complex dynamic systems.
- Programming: security and reliability of computing systems.
- Communication, information, and ubiquitous computing.
- Interaction with real and virtual world.

3.2 The SED Team

The SED service (software Developments and Experimentations Support service) is the team of about 10 engineers and technicians which provides support for researches leaded at INRIA Rhone-Alpes. This service is both an experiment and a development support.

Concerning the experiment support, this service is in charge of:

- Maintenance of experiment platforms (robotics, virtual reality...).
- The support for the development by the implementation of experiments and software.
- Communication, information, and ubiquitous computing.
- The support for the research by the participation in experiments.

Concerning the software development, the service is mainly in charge of:

- The implementation and the animation of an exchange structure in the aim to facilitate the sharing of knowledges.
- The help for software conception by bringing expertise in software architecture.
- Communication, information, and ubiquitous computing.
- The training of students and beginners.

Figure 3.2: SED Logo

Working environment

Before programming, a long time was necessary to find and understand documentations. I also had to use other people's code, which I needed to understand and sometime correct. First I studied the WSN430 and its components. Then I discovered the software tools to flash a program and to use the WSN430 board with or without a real-time operating system kernel.

4.1 WSN430 boards

The WSN430 (Wireless Sensor Network - MSP430) is an electronic board designed to be integrated in Wireless Sensor Networks. Each WSN430 will be a node in the wireless sensor network so they are composed by all components needed like transmitter/receiver component, CPU, basic sensors(temperature, light, sound), daughter board with specifics sensors (accelerometer, magnetometer...)

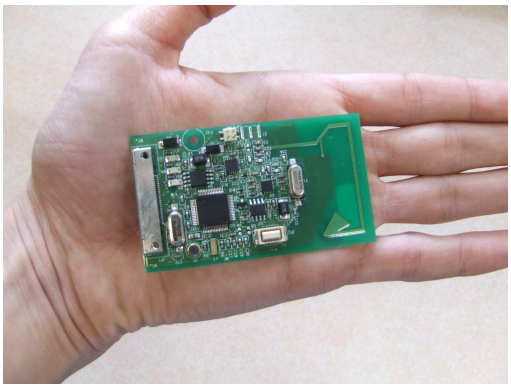


Figure 4.1: Picture WSN430

4.1.1 Features

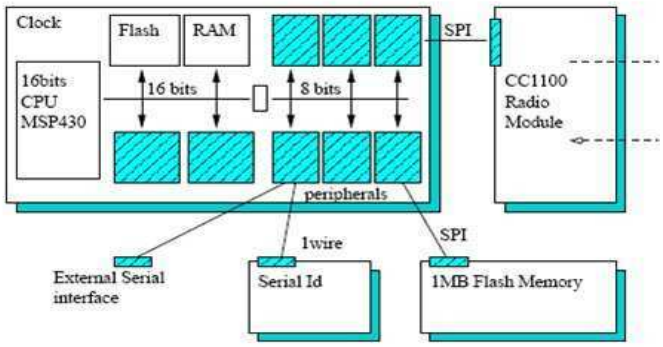


Figure 4.2: WSN430 Block Diagram

LEDs:

3 LEDs of various colors (red, green, blue) are integrated in the WSN430. They are essentially used for debugging or for small tests.

MSP430:

The MSP430 is a Texas Instrument microcontroller running with an external 8MHz clock. This microcontroller is programmable via a JTAG connection. It integrates a 48 KBytes flash memory, a 10 Koctets RAM memory, 48 configurable Inputs/Outputs, 12-bit analog-to-digital conversion pins, a watchdog, 2 serial communication ports and 2 configurable timers. This microcontroller is compatible with most of real-time operating systems such as FreeRTOS.

CC1101:

The CC1101 is a RF transmitter/receiver designed for wireless applications which need a low power consumption. It is entirely configurable to work at several frequency (315, 433, 868 and 915 MHz). This chip is commonly used in the wireless sensor networks development. It can use several modulation types (2-FSK, GFSK, MSK, OOK and ASK) and has a -111 dbm reception sensibility.

DS2411:

The DS2411 is a chip that gives an unique ID. This ID is commonly used to differentiate each node (WSN430 in our case) in a wireless sensor network.

Communication Bus:

The WSN430 has a SPI communication bus. The microcontroller communicates with the cc1101 and an external 1MB flash memory via this bus: The MSP430 is the master and the others components are slaves.

Connectivity:

The WSN430 has 2 serial communication ports with configurable baudrates. Thanks to those serial communication ports we can connect a PC to a WSN430 to debug or to collect data from the sink node.

4.2 Software tools

During my internship I've used several software tools already familiar. I've worked with Fedora which is a Linux-based open source software operating system. I've already worked with Fedora or similars OS so I've handled quickly the environment.

4.2.1 The GCC toolchain for MSP430 family

I've written all programs in C language. I've used the GCC toolchain for MSP430 family to flash programs into the microcontroller. This includes the GNU C compiler (GCC), the assembler and linker (binutils), the debugger (GDB), and some other tools needed to make a complete development environment for the MSP430. These tools can be used on Windows, Linux, BSD and most other flavours of Unix.

4.2.2 Minicom

To debug programs and to acquire data collected by the sink node I' ve used Minicom which is a software tool capable of receiving data from serial port. Minicom is a free communication program.

4.2.3 FreeRTOS

FreeRTOS means Free Real-Time Operating System. FreeRTOS is a real-time operating system kernel for microcontrollers. It provides tasks (process) and co-routines (threads) management. So it facilitates the programmer work because he doesn't have to program management functions, scheduling, context switches, lists, semaphores... FreeRTOS was designed for several architectures and compilers.

I've used FreeRTOS for my applications so I spent a long time reading documentations for discovering the features. FreeRTOS was never used before for WSN430 boards so I've also customized it to be compitable with the hardware.

Figure 4.3: Logo FreeRTOS

During this first part of my internship I've learnt many technical aspects. After having studied all the documentations and handled the different software and hardware tools I began to enter in the subject of my internship.

Project realization

In this part I detail all I've made concretely during my internship other than reading technical documentations, collecting information... First of all I spent some time to handle the WSN430 board and all components. After having designed some basic programs for the WSN430 I worked on FreeRTOS, I designed demonstrations for this latter. Then I did a user documentation for future programmers who will use FreeRTOS on WSN430 boards. Finally I designed a basic wireless sensor network.

5.1 Handling WSN430

5.1.1 LEDs

Firstly I tried to flash simple programs into the microcontroller for handling LEDs. This one allows me to manipulate the necessary material for programming and to manipulate the new toolchain. The program just toggle 3 LEDs of the WSN430 by using states:

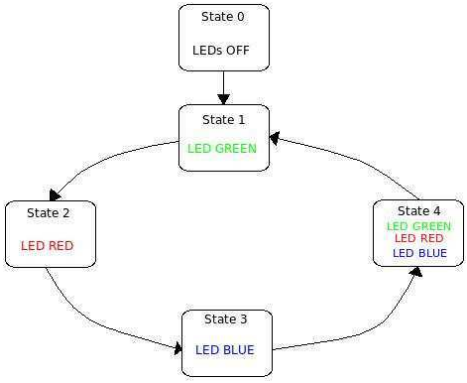


Figure 5.1: LEDs states

5.1.2 Serial communication

I spent some time in this part beacause the drivers which I had, wasn't adapted for my WSN430 board version. There is 2 serial ports in the WSN430: UART0 and UART1. In the given drivers, the UART1 is used for communication while in my version it's the UART0 that I had to use. It took me some time to localize the problem and after that I had to fix it by modifying the driver.

To manipulate and to test the serial communication I've made a little program that send continuously the same character string. I connected the WSN430 to my Linux machine and with minicom I checked that the message was well received continuously.

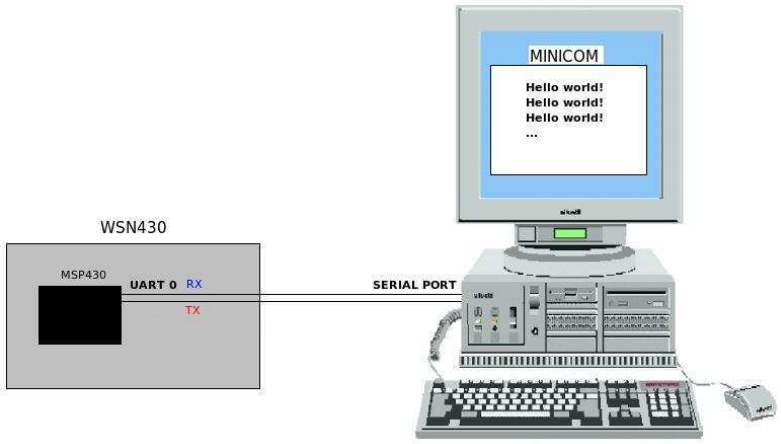


Figure 5.2: Serial communication

5.1.3 Radio communication

In this phase, I've needed 2 WSN430 boards. The principle of the application I designed was that one WSN430 is the receiver (RX) and the other one is the transmitter (TX). The TX transmits continuously the same packet and the receiver toggle the green LED when it receives the packet and after having checked that the packet is correct (the same message that the one sended).

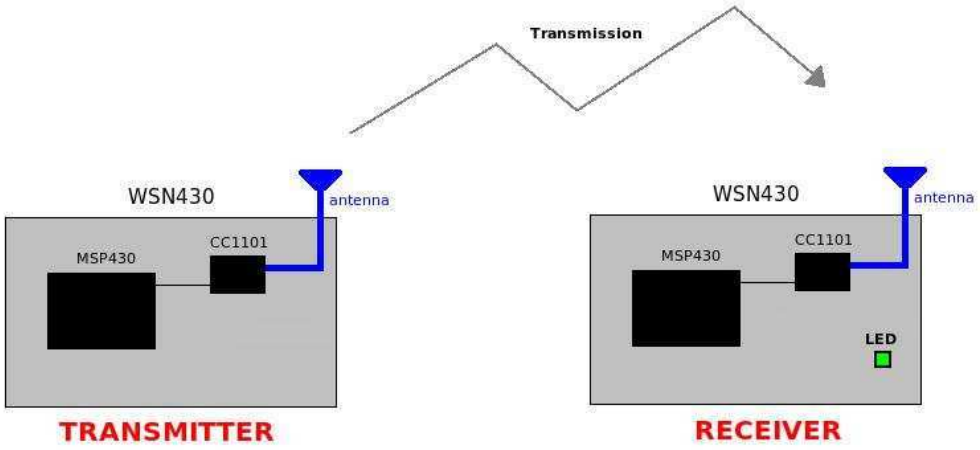


Figure 5.3: RF transmission/reception

I've encountered a lot of problems in this part of my internship. At first I had to familiarize myself with the given cc1100 drivers. There is a lot of configurations that I had to understand and to apply in my application. Indeed, for each program using the radio communication, we have to choose the modulation type, the crc policy, the whitening data policy, the tx power... It took me some time to master all these aspects.

Then I encountered another problem : the communication wasn't reliable. Indeed, we had a 25-50% transmission loss. It was an important problem because we can't do complex aplication with a none reliable communication. I've tried to change the modulation type,txs power, antennas orientation, packets size, but the problem didn't come from these aspects. I've localized the problem after some days by asking my project leader, consulting forums, searching in the drivers, and it came from the given drivers. In fact, there was a quality threshold too high. The quality threshold determines the threshold at which a received packet is treated or rejected. This quality is measured by counting the number of preamble state changes. The preamble is a 010101... succession. If there isn't enough state changes the packet is rejected. After having reduced this threshold the transmission became good.

5.1.4 Sampling data from a microphone

A microphone is integrated in the WSN430 board. This one is connected to an analogical to digital converter microcontroller pin (ADC). In the aim to validate the board for production I've tested this microphone by doing an application. This program samples the microphone ADC and it sends, with a configurable frequency, digital data on the serial port. To define the sampling frequency I've used one of the two microcontroller timers. For each timer interruption the microcontroller samples the microphone ADC and it transmits data on the serial port. Thanks to Minicom we can collect the sound data into a file and play it.

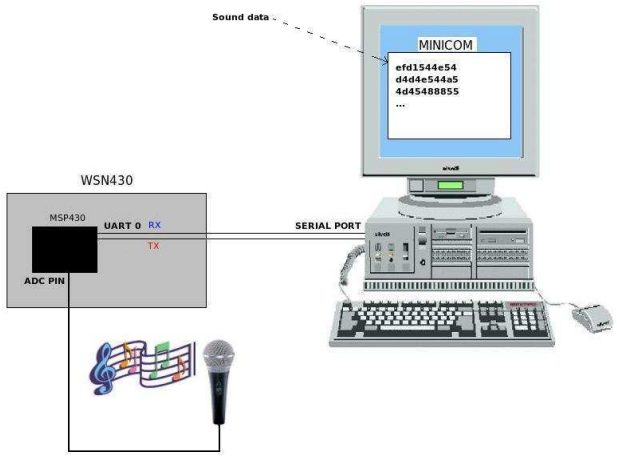


Figure 5.4: Sound

The sound level was very weak even with a software amplification or with a very powerful sound source, but it's enough for the future microphone use (ambient noise detection).

5.2 Installation of FreeRTOS

FreeRTOS is an open source program so I've downloaded it from the official site. I familiarised myself with the source code organisation and directory structure. FreeRTOS has been ported to many different architectures and compilers. Each port is accompanied by a pre-configured demonstration application. I studied these demonstrations to know how to use the different features of FreeRTOS.

After that, I've organized the structure and I've created a *makefile* model to facilitate the development of application using FreeRTOS.

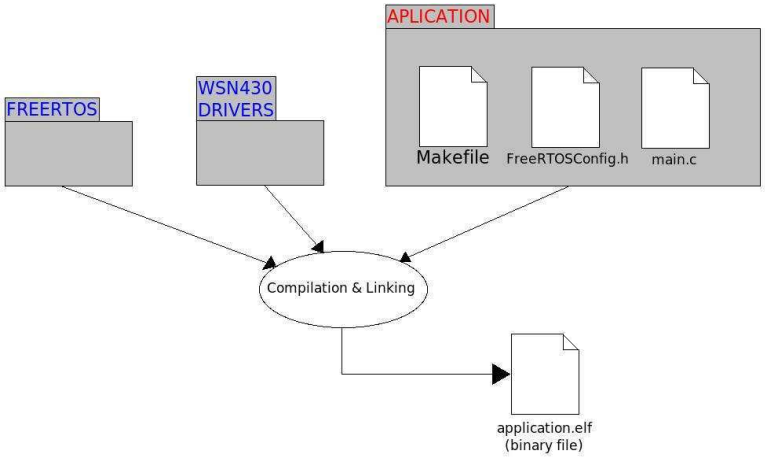


Figure 5.5: Organization

5.3 FreeRTOS Demonstrations

I've programmed many demonstrations using FreeRTOS kernel for future programmers who will use FreeRTOS on WSN430 board. These demonstrations are ready to run and pre-configured.

5.3.1 LEDs

This application uses only one WSN430 board and creates 3 tasks. Each task is in charge of one LED. The function associated to each task toggle the LED in charge and sleep during a defined time depending of the task. The result is that all the 3 LEDs toggle with different frequencies:

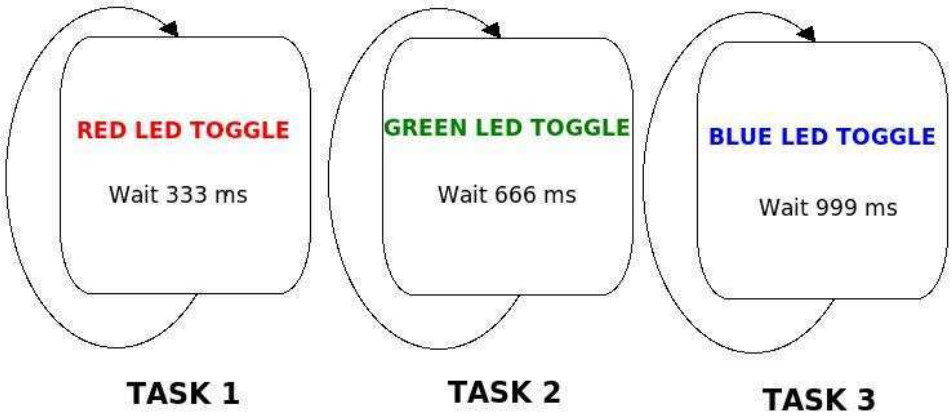


Figure 5.6: LED tasks

5.3.2 Transmitter/Receiver HF

This application has the same result that the one already presented without FreeRTOS. But, instead of using a transmission loop in the main function, we create a task in charge of sending periodically the same message.

5.3.3 DS2411

This application uses only one WSN430 board and creates 1 task. The task created sends periodically the node ID on the serial port (UART 0).

5.3.4 Timer synchronisation

This application uses only one WSN430 board, creates one task and configures the *B timer* to run with a 200 Hz frequency. The task changes LEDs state and then wait on a semaphore. For each *B timer* interruption a counter is incremented and when this counter reaches 200, we check the task state. If the task is suspended we free it by making a *'semaphore give'* instruction, else if the task is still running there is *'over-run'* and we stop the application. It's a typical real time application example.

5.3.5 Ping-Pong

This application uses two WSN430 boards. Each one is configured to be both receiver and transmitter. First of all, one node sends a known message. The other node receives the message, checks if it's correct and then sends the received message to the other node which repeats the same process...

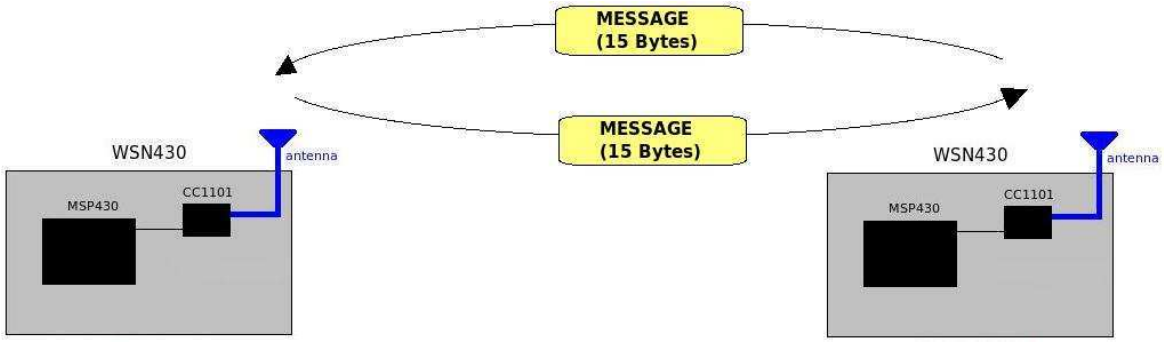


Figure 5.7: Ping Pong

5.4 User Documentation

After having discovered most of FreeRTOS aspects, I've written a detailed user documentation in french (cf. appendix). In this report I've approached the following points :

- Necessary material:
  In this part it's detailed the necessary material to program the WSN430 and how we have to connect material.
- Programming on WSN430:
  I've explained quickly the toolchain to flash the microcontroller and then I've detailed the drivers and I've explained step by step 2 examples: LEDs and Transmitter/Receiver RF examples. It shows to the user how to create a *makefile* and to include the drivers to projects.
- Programming with FreeRTOS:
  In this part it's explained in a very detailed way how to make an application with FreeRTOS. First of all it describes the different FreeRTOS features and then it explains how to create tasks, use features, create makefile...

5.5 Wireless Sensor Network

5.5.1 Principle

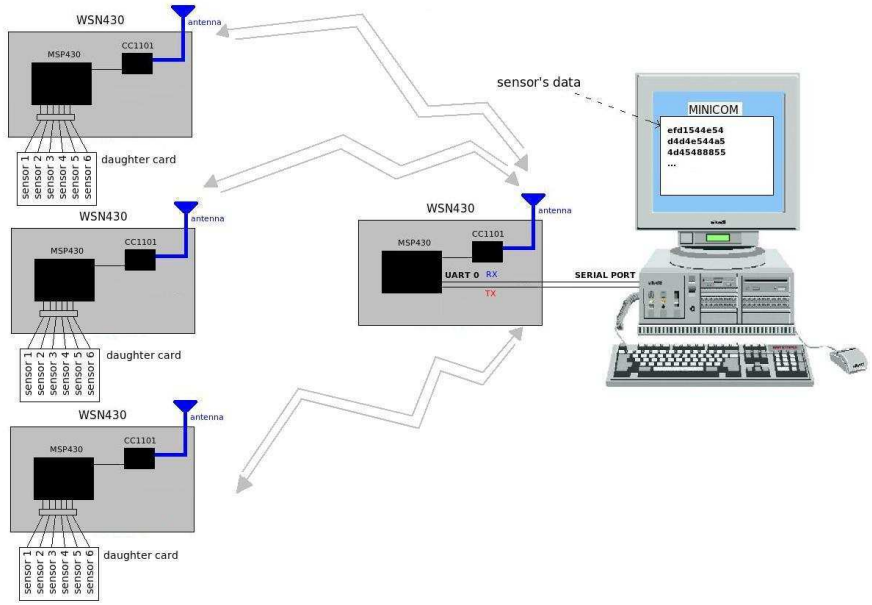


Figure 5.8: Principle

This application includes one RX (the sink) and several TXs. After a discovery phase, the sink will collect data from TXs in the catchment field.

- The RX:
  It's connected to a Linux machine by the serial port and the user can communicate with it by using *Minicom*. The RX behaves like a master with the application TXs.

- The TXs:
  For each TX node, are added a daughter board with 6 sensors. Each sensor is connected to an ADC microcontroller pin. TXs have to sample these ADC pins and to send data to the sink.

5.5.2 Discover protocol

Before the data transmission beginning, a 10 secondes discovery phase is needed. The role of this phase is for the sink to discover and memorize all TXs presents in the catchment field and to select for each one, the sensors we want data from. In this aim I designed a protocol.

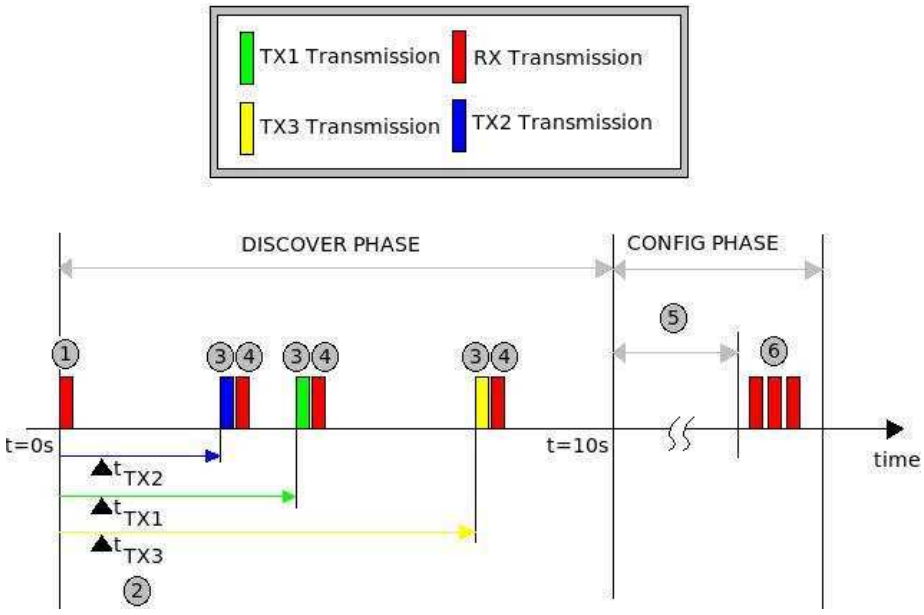


Figure 5.9: Discover

1. The RX sends a *'discover request packet'*.
2. Each TX waits during a periode, calculated in function of the TX ID (to avoid collisions)
3. Each TX sends a *'discover reply packet'*
4. The RX acknowledges by sending a *'discover ack reply packets'*.
5. The user chooses the sensors he wants to collect data for each TX discovered.
6. The RX sends *'configuration packets'* to all TXs.

5.5.3 Data transmission

The choice of the protocol

I've made some researches about protocol avoiding collisions. We finally choose a kind of TDMA protocol for the application (Time Division Multiple Access). Among all the protocols existing, the TDMA is the most adapted.

Indeed, the FDMA protocol (Frequency Division Multiple Access), isn't adapted because it would mean that the RX could receive data at many different frequencies. The RX can't do this because we configure only once the frequency, in addition only 4 frequencies are avaibles to configure the WSN430 (315, 433, 868 and 915 MHz).

The CSMA protocol (Carrier Sense Multiple Access), isn't adapted because it would mean that the nodes are capable of collision detection. In fact, nodes are not able to do this because they can't distinguish a corruption from a collision.

Principle of the protocol

For each TX are dedicated a 20 ms periode called a *'slot'*. During one slot the RX sends a *start packet* to the concerned TX. After having received the *start packet*, the concerned TX sends his data to the RX. The RX receives the data and store it. After having collected data from all TXs, a slot is dedicated for the RX to send data on the serial link and an other one if there is dead nodes detected. A node is considerated dead when it hasn't sent data 5 times in succession. The synchronisation is made with a timer which generates an interruption each 20 ms.

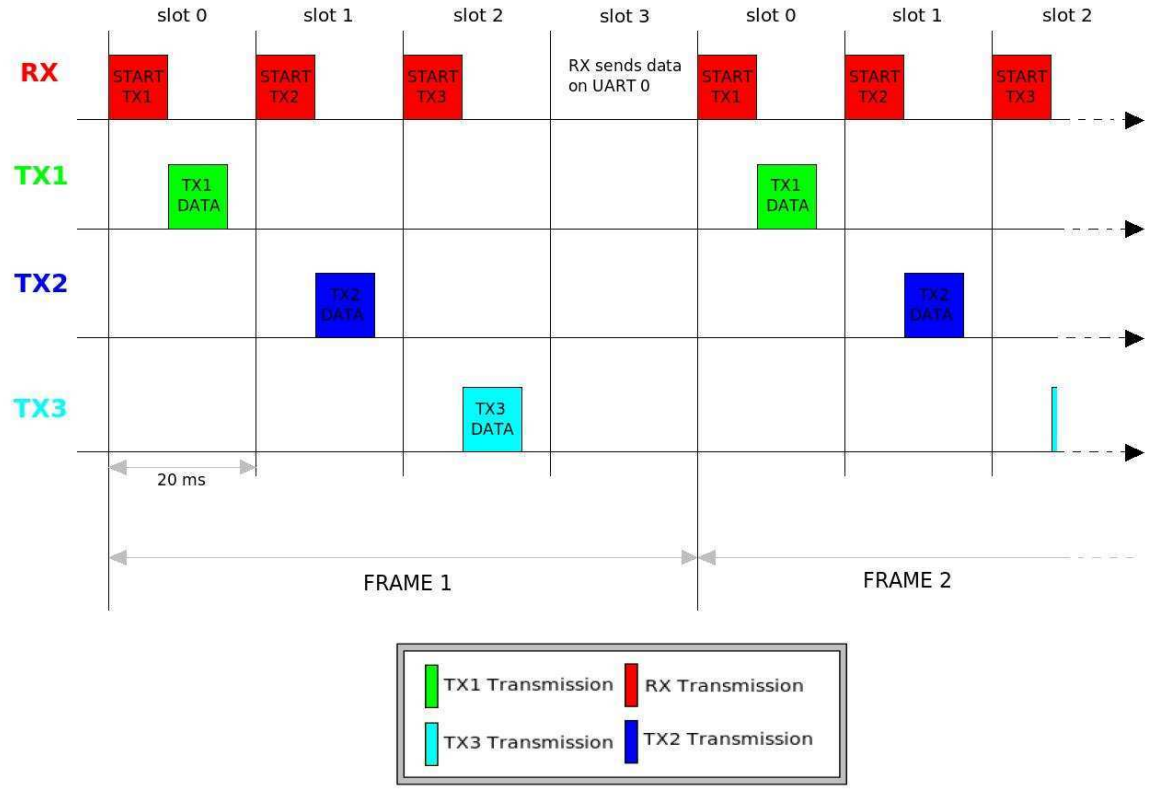


Figure 5.10: Data transmission

Conclusion

The objective of this internship was to develop a software infrastructure allowing to unify tools for experiments on wireless sensor networks for the SensLab project.

In this context I've studied the functionning of such a network and the different possibilities to design it. I've designed software demonstrations and a user documentation for using FreeRTOS on a WSN430 board. In spite of the encountered problems I finally designed a little WSN that works correctly with many TXs.

The main problem encountered while I was designing the WSN was that I had two WSN430 board versions. The drivers didn't work with the new WSN430 board version so I had to adapt one more time the drivers. In fact the RF component of the new WSN430 board doesn't work with the drivers I had. It was painful to adapt the drivers and to switch between the 2 versions for flashing boards.

This internship was very fruitfull to me because it has brought me an experience which I can turn to good account in the future and it has covered many different fields : architecture, computing, algorithm, protocol, telecommunications, signal. Furthermore, during my 2 years at INPG Telecom I've acquired many technical skills that I've put into practice in my project.

I also learnt new concepts and new working ways different than the way we work in school. Indeed I worked in a small team and I think that internship was very benefic to me as I learnt a lot, and it made me discover the research and work's world.

Bibliography

[1] INRIALPES
http://www.inrialpes.fr

[2] SED
http://sed.inrialpes.fr

[3] WORLDSENS
http://worldsens.citi.insa-lyon.fr

[4] FREERTOS
http://www.freertos.org

[4] MSPGCC
http://mspgcc.sourceforge.net

[5] Texas Instrument : MSP430 Family User's Guide

[6] Texas Instrument : CC1100 User's Guide

Appendix : FreeRTOS on WSN430 (French)

INRIA Montbonnot
SED

FreeRTOS sur wsn430v1.3

Rida ROUAISSIA

Montbonnot
, le 21 août 2008

Table des matières

Table des figures

Partie 1

Introduction

1.1 WSN430

1.1.1 Déscriptif

Contexte

La plateforme WSN430 est concu pour être intégré dans les réseaux de capteurs sans fils.

Les réseaux de capteurs sans fil - Wireless Sensor Networks (WSN) - sont considérés comme un type spécial de réseaux ad hoc. Les nœuds de ce type de réseaux consistent en un grand nombre de capteurs capables de récolter et de transmettre des données environnementales de manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils sont dispersés aléatoirement à travers une zone géographique, appelée champ de captage, qui définit le terrain d'intérêt pour le phénomène capté. Les données captées sont acheminées grâce à un routage multi-saut à un nœud considéré comme un "point de collecte", appelé nœud puits (ou *sink*). Ce dernier peut être connecté à l'utilisateur du réseau via Internet ou un satellite. Ainsi, l'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud puits.

La taille de plus en plus réduite des capteurs, le coût de plus en plus faible, la large gamme des types de capteurs disponibles (thermique, optique, vibrations,...) ainsi que le support de communication sans fil utilisé, permettent aux réseaux de capteurs d'envahir plusieurs domaines d'applications. Ils permettent aussi d'étendre les applications existantes et de faciliter la conception d'autres systèmes tels que le contrôle et l'automatisation des

chaînes de montage. Les réseaux de capteurs ont le potentiel de révolutionner la manière même de comprendre et de construire les systèmes physiques complexes. Les réseaux de capteurs peuvent se révéler très utiles dans de nombreuses applications lorsqu'il s'agit de collecter et de traiter des informations provenant de l'environnement. Parmi les domaines où ces réseaux peuvent offrir les meilleures contributions, nous citons les domaines : militaire, environnemental, domestique, santé, sécurité, etc.

Fonctionnalités :

Le WSN430 est une plateforme faible consommation et permet donc d'avoir une longue autonomie. La plateforme est alimentée par des piles/batteries. Le WSN430 est constitué de plusieurs composants fournissant ainsi des fonctionnalités diverses.

LEDs : Le WSN430 comporte 3 LEDs de différentes couleurs (rouge,verte,bleue). Elles servent éssentiellement à debugger, à faire des petits tests ou à faire de la signalisation.

MSP430F1611 : Le MSP430F1611 est un microcontrôleur cadencé à 8MHz. Il est fabriqué par Texas Instrument. Ce microcontrôleur est programmable via un connecteur JTAG. Il intègre une mémoire Flash de 48 Koctets, une mémoire RAM de 10 Koctets et 48 GPIO. Ce microcontrôleur est compatible avec la plupart des OS temps réel tels que FreeRTOS, TinyOS, Contiki.

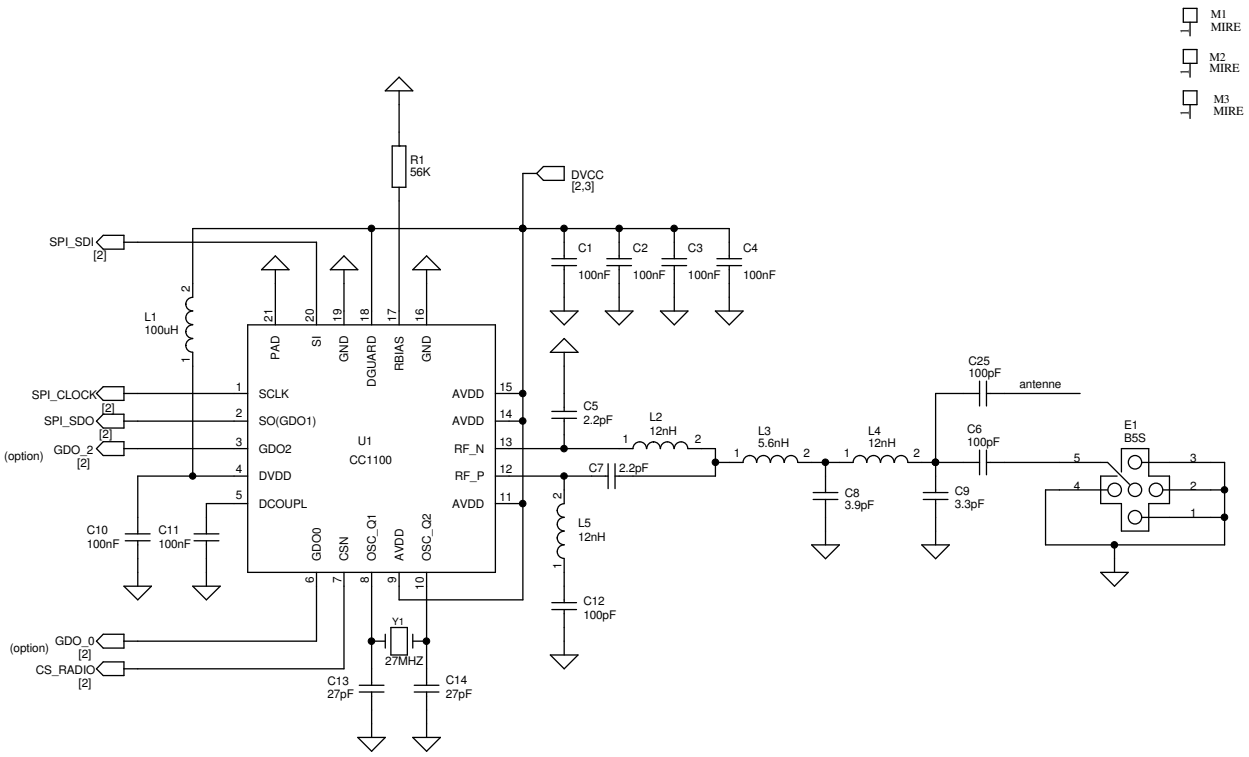
CC1101 : Le CC1101 est un émetteur/récepteur RF concu pour des applications sans fil très faible consommation. Il est entièrement configurable pour pouvoir travailler à 315, 433, 868, et 915 MHz. Ce circuit est très couramment utilisé dans le dévelopement des réseaux de capteurs sans fil. Il supporte plusieurs types de modulations (2-FSK, GFSK, MSK, OOK et ASK) et a une sensibilité de réception de -111 dBm.

DS2411 : Le DS2411 est un circuit integré qui fournit un identificateur unique pour chaque plateforme WSN430. Cet identificateur va pouvoir servir à différencier chaque noeud dans un réseau de capteur sans fil par exemple.

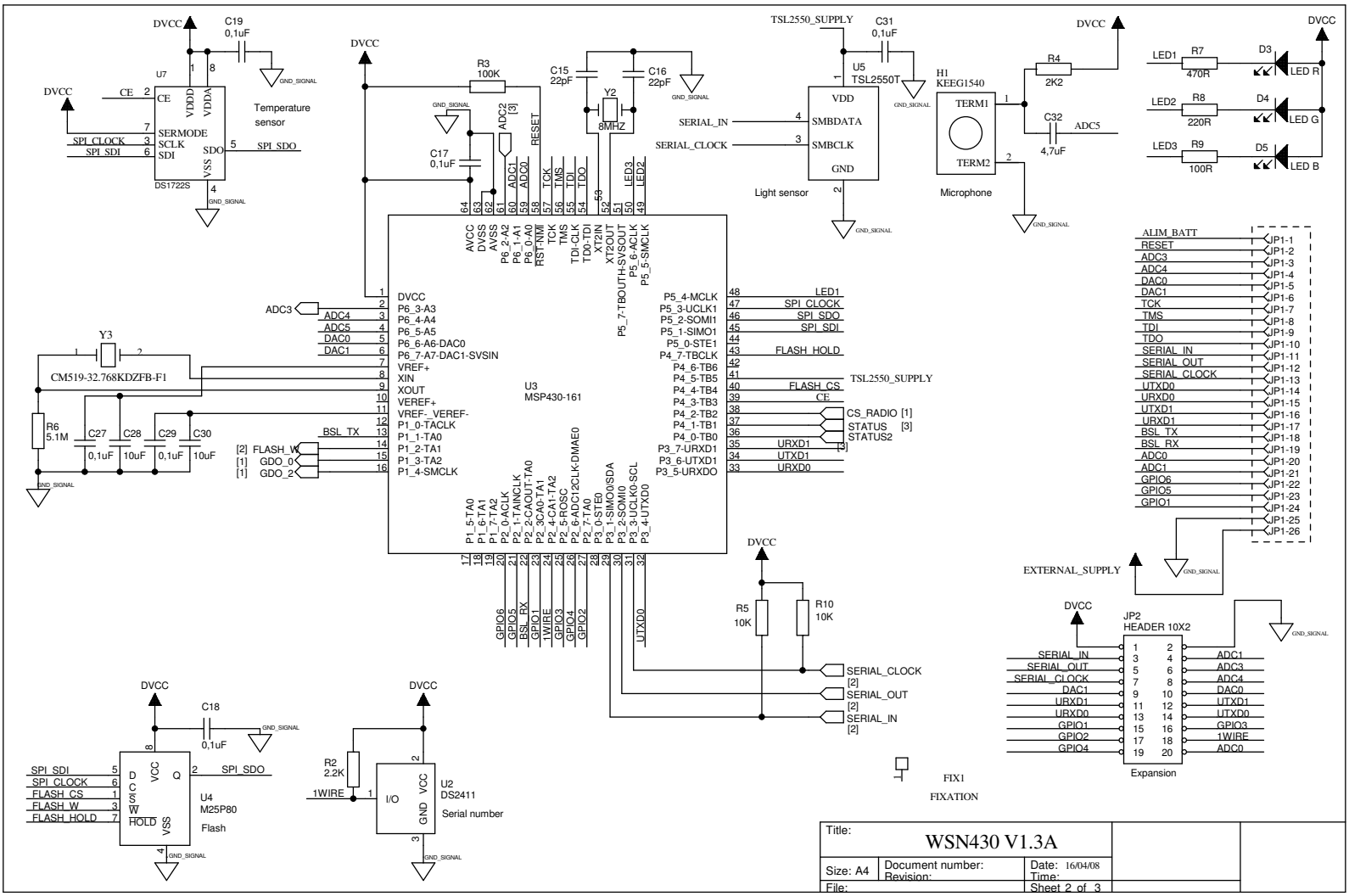
'Bus' de communication : Le WSN430 dispose d'un 'Bus' de communication qui utilise le protocole de communication SPI. C'est notamment par le biais de ce bus de communication que le microcontrôleur communique avec le cc1101 : Le cc1101 étant un esclave et le MSP430 le maître.

Connectivité : Le WSN430 dispose de 2 ports de communication série dont la vitesse de transmission est configurable. On peut ainsi connecter le WSN430 sur un PC par exemple pour pouvoir debugger lors du dévelopement d'applications.

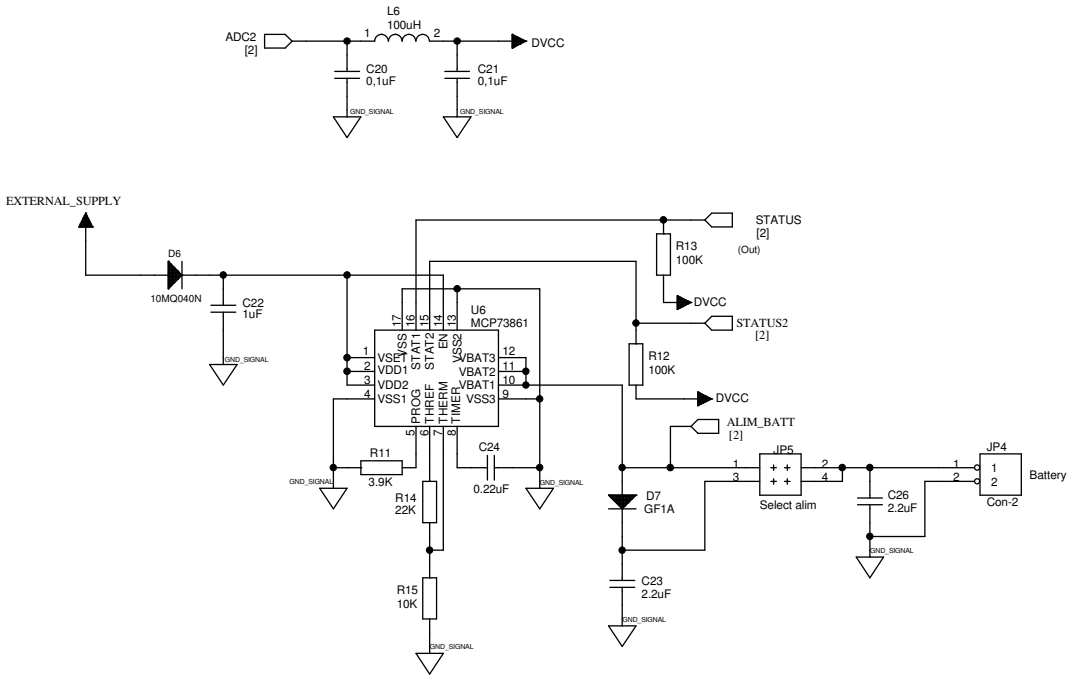
WSN430v1.3a.sch-1 - Thu Apr 17 10:15:41 2008



Title: WSN430 V1.3A			
Size: A4	Document number:	Date: 16/04/08	
	Revision:	Time:	
File:		Sheet 1 of 3	



Title: WSN430 V1.3A			
Size: A4	Document number:	Date: 16/04/08	
	Revision:	Time:	
File:		Sheet 2 of 3	



Title: WSN430 V1.3A			
Size: A4	Document number:	Date: 16/04/08	
	Revision:	Time:	
File:		Sheet 3 of 3	

1.1.2 Matériel nécessaire pour la programmation du WSN430

Pour flasher un programme sur le WSN430 il faut disposer :
– D'une plateforme wsn430. (Fig. 1.1 et 1.2)

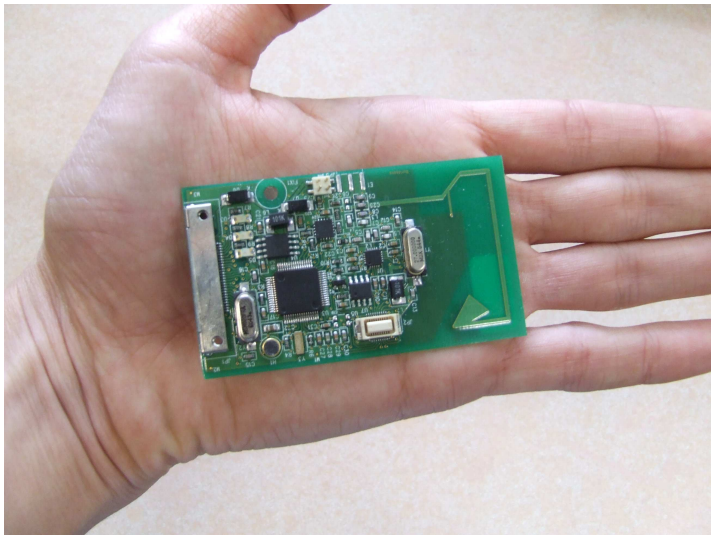


FIG. 1.1 – Photo wsn430v1.3 (main)

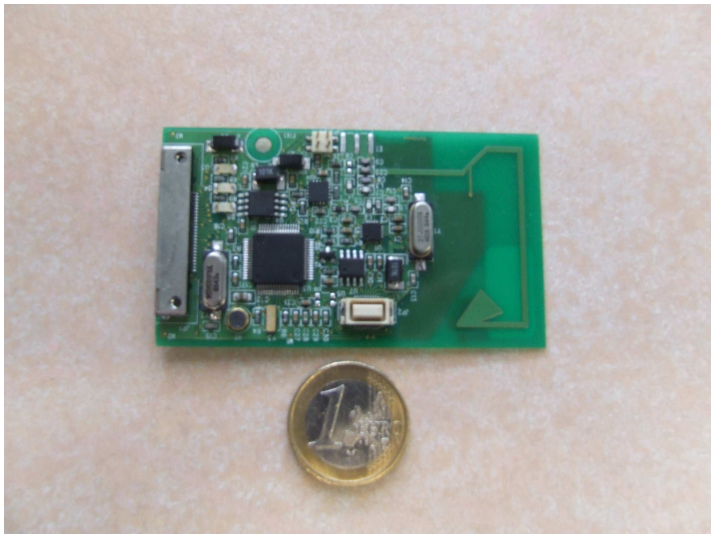


FIG. 1.2 – Photo wsn430v1.3 (pièce)

– D'une alimentation continue qu'on règle à 5-6 Volts avec les fils adéquats. (Fig. 1.3)

FIG. 1.3 – Photo Alimentation + Cables

– D'un adaptateur "WSN430 JTAG". (Fig. 1.4)

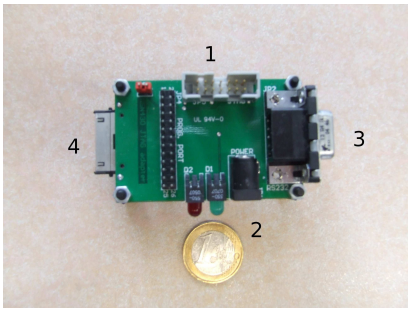


FIG. 1.4 – Photo wsn430 JTAG adapter

L'adaptateur "WSN430 JTAG" fournit :

1. une connexion JTAG pour pouvoir programmer le MSP430 de la plateforme via un PC.
2. une entrée d'alimentation pour alimenter le wsn430.
3. une connection serie reliée à l'UART0 du MSP430.
4. une connexion JTAG pour le wsn430.

– D'un cordon d'adaptation JTAG - PORT PARALLELE avec une rallonge port parallèle si besoin. (Fig. 1.5)

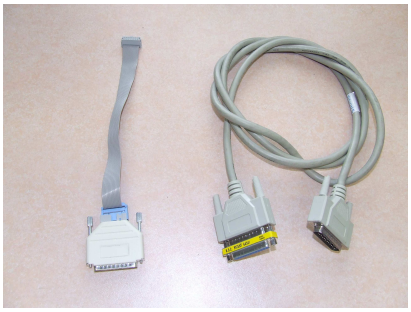


FIG. 1.5 – Photo cordon adaptation JTAG PARALLELE

Il faut à présent brancher le matériel comme le montre la photo Fig. 1.6.

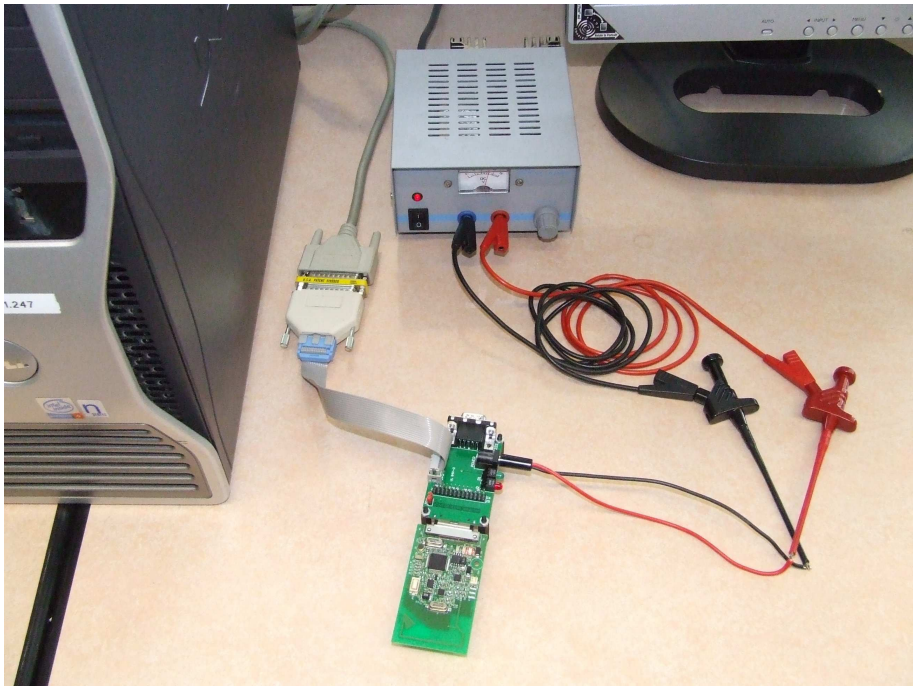


FIG. 1.6 – Photo Branchement

1.2 Introduction à FreeRTOS

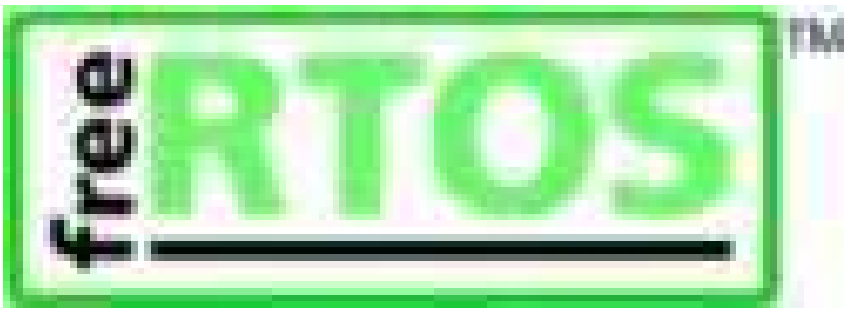


FIG. 1.7 – Logo FreeRTOS

FreeRTOS signifie *Free Real Time Operating System*. FreeRTOS est un noyau de système d'exploitation temps réel pour microcontrôleur.

Il permet, entre autres, de faire tourner plusieurs tâches (processus) et de s'affranchir de la programmation pennible des fonctions de gestion du cadenceur, des changements de contexte, des listes et files ...

FreeRTOS a été concu pour plusieurs architectures et compilateurs différents. Pour chaque architecture compatible il éxiste une démo pré-configurée prête à l'emploi. Il éxiste une configuration pour le microcontrôleur MSP430F449 dont nous nous inspirons pour créer un projet pour le wsn430.