



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# MASTER THESIS

**TITLE: THE MAGCLOUD WIRELESS SENSOR NETWORK**

**MASTER DEGREE: Master in Science in Telecommunication Engineering  
& Management**

**AUTHORS: Narciso Cuartero Moya  
Sergio Quintana Alcaraz**

**DIRECTOR: José Polo Cantero  
2nd DIRECTOR: Jorge Eduardo Higuera Portilla**

**DATE: July, 4th 2011**

**Title:** THE MAGCLOUD WIRELESS SENSOR NETWORK

**Authors:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

**Director:** José Polo Cantero  
**2nd Director:** Jorge Eduardo Higuera Portilla

**Date:** July, 4th 2011

## Overview

Initially, the aim of this project consisted in manufacturing some nodes for a wireless sensor network by hand. If this document concludes that they can be properly produced in the EETAC lab, the cost of a future large deployment using raw components would be much lower than in the case of acquiring the genuine factory assembled hardware. Also, the future students involved in the process could learn many useful advanced techniques along the way.

The project ended sowing a future WSN concept, so powerful that even could end competing on the market. We designed an almost unlimited scalable platform in terms of range, number of nodes, connectivity and measuring capabilities that is 100% free, open and environment sustainable. We called this unique wireless magnitude acquisition cloud: *THE MAGCLOUD*. The whole system cannot be fully finished within the time and budget restrictions of a single PFC but slicing it into diverse future upgrades is a completely realistic approach.

In this document, sticking to the original idea, we explain how to produce the functional hardware and software skeleton but also guide the reader on the future upgrades required to complete the MAGCLOUD system.

During the realization of the project we found countless problems that luckily end up solved. Those are carefully treated so can be avoided in the future.

**Título:** THE MAGCLOUD WIRELESS SENSOR NETWORK

**Autores:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

**Director:** José Polo Cantero

**2nd Director:** Jorge Eduardo Higuera Portilla

**Fecha:** 4 de Julio de 2011

## Resumen

Inicialmente, el objetivo del proyecto era conseguir fabricar a mano al menos un nodo apto para la integración en una red conocida de sensores inalámbricos. Si se podía demostrar que era factible ensamblar éstos nodos usando los recursos del laboratorio de la EETAC, otros estudiantes podrían en un futuro seguir produciendo unidades hasta conseguir mallar poco a poco el *Parc Mediterrani de la Tecnologia*.

Sin embargo, el proyecto no se quedó ahí. Terminamos por engendrar un nuevo concepto de WSN, tan potente que incluso podría llegar a competir en el mercado en un futuro. Una plataforma casi infinitamente escalable en términos de alcance, número máximo de nodos, conectividad y rango de magnitudes medibles que puede presumir de ser 100% abierta y sostenible. Decidimos llamarla MAGCLOUD. Como el sistema no podía completarse en un único PFC, decidimos dividirlo en varias porciones en las que otros estudiantes interesados pudieran basarse.

Manteniéndonos fieles a la idea original, en este documento se detallan los pasos a seguir para fabricar rápidamente nodos con los que desplegar una nube de sensores autónomos. También guiamos al lector en las ampliaciones aplicables sobre ésta base que permitirían completar el proyecto MAGCLOUD.

## **ACKNOWLEDGEMENTS**

To José Polo and Jorge Eduardo Higuera for their guidance and support.

To Francis López for having been always at our side whenever we needed.

To Óscar López for its help on the FTDI IC understanding.

To Juan López for its lessons on programming.

To Ramón Pallars for listening and helping when all was mess and trouble.

To our families for their patience and support along the thesis.

To Mar Blázquez for her dedication and objective point of view.

Because MAGCLOUD would have not been possible without your help:

# **THANK YOU!**

# INDEX

|   |           |
|---|-----------|
| <b>INTRODUCTION .....</b>                           | <b>1</b>  |
| <b>REQUERIMENTS .....</b>                           | <b>2</b>  |
| <b>CHAPTER 1. WSN TECHNOLOGIES .....</b>            | <b>3</b>  |
| 1.1 Wireless Sensor Network .....                   | 3         |
| 1.2 IEEE 802.15.4 technology .....                  | 6         |
| 1.2.1 Components of WPAN .....                      | 7         |
| 1.2.2 Physical Layer .....                          | 7         |
| 1.2.3 MAC Layer .....                               | 8         |
| 1.2.4 Associated protocols .....                    | 9         |
| 1.3 ZigBee .....                                    | 10        |
| 1.3.1 Key Features .....                            | 11        |
| 1.4 6LOWPAN .....                                   | 12        |
| 1.4.1 Key Features .....                            | 12        |
| <b>CHAPTER 2. CHOSING THE PLATFORM .....</b>        | <b>13</b> |
| 2.1 Software Platform .....                         | 13        |
| 2.1.1 TinyOS .....                                  | 13        |
| 2.2 Hardware Platform .....                         | 14        |
| 2.2.1 Comparative Platforms .....                   | 15        |
| 2.3 TMOTE SKY .....                                 | 18        |
| 2.3.1 Key Features .....                            | 18        |
| 2.3.2 Module Description .....                      | 19        |
| 2.3.3 Microprocessor .....                          | 21        |
| 2.3.4 Radio .....                                   | 22        |
| 2.3.4.1 <i>Measured Output Power</i> .....          | 23        |
| 2.3.5 Antenna .....                                 | 24        |
| 2.3.5.1 <i>Radiation Pattern</i> .....              | 25        |
| 2.3.6 Expansion Connector .....                     | 25        |
| <b>CHAPTER 3. THE HARDWARE IMPLEMENTATION .....</b> | <b>27</b> |
| 3.1 Step 1: Finding the Right Components .....      | 27        |
| 3.2 Step 2: The Board Assembly .....                | 31        |
| 3.3 STEP-BY-STEP MAGCLOUD NODE CONSTRUCTION .....   | 50        |

|   |               |
|---|---------------|
| <b>CHAPTER 4. THE SOFTWARE IMPLEMENTATION .....</b> | <b>52</b>     |
| 4.1 Ubuntu .....                                    | 52            |
| 4.1.1 Installing TinyOS .....                       | 53            |
| 4.1.2 Using TinyOS .....                            | 57            |
| 4.1.2.1 <i>Main commands</i> .....                  | 57            |
| 4.1.2.2 <i>Examples applications</i> .....          | 58            |
| 4.1.2.3 <i>CC2420.h</i> .....                       | 59            |
| 4.2 Our Software .....                              | 60            |
| <br><b>CHAPTER 5. RESULTS .....</b>                 | <br><b>65</b> |
| 5.1 Accuracy Measuring .....                        | 65            |
| 5.2 Consumption .....                               | 68            |
| 5.3 Time and Costs .....                            | 71            |
| <br><b>CHAPTER 6. UPGRADES .....</b>                | <br><b>74</b> |
| 6.1 Hardware Upgrades .....                         | 74            |
| 6.2 Software Upgrades .....                         | 76            |
| <br><b>CHAPTER 7. ENVIRONMENTAL IMPACT .....</b>    | <br><b>77</b> |
| <br><b>CHAPTER 8. CONCLUSIONS .....</b>             | <br><b>78</b> |
| <br><b>CHAPTER 9. REFERENCES .....</b>              | <br><b>79</b> |
| <br><b>ANNEXES .....</b>                            | <br><b>82</b> |
| ANNEX A: PCB Suppliers Conversation                 |               |
| ANNEX B: 2Cisa Budget                               |               |
| ANNEX C: Solar submodules                           |               |
| ANNEX D: Installation of blip under Ubuntu 10.04    |               |
| ANNEX E: Tmote Sky datasheet                        |               |

## FIGURES INDEX

|   |    |
|---|----|
| Fig.0.1. The PMT: Location for deploying the WSN .....                      | 1  |
| Fig.1.1. WSN Application Areas .....  | 3  |
| Fig.1.2. Hardware Block diagram of a WSN mote .....                         | 4  |
| Fig.1.3. Basic functionality of a sensor node .....                         | 5  |
| Fig.1.4. WSN Network Topologies .....                                       | 6  |
| Fig.1.5. Operating frequency bands .....                                    | 8  |
| Fig.1.6. MAC frame format .....   | 9  |
| Fig.1.7. 802.15.4/ZigBee Stack .....  | 10 |
| Fig.1.8. ZigBee application areas .....                                     | 11 |
| Fig.1.9. 802.15.4/6LoWPAN Stack .....                                       | 12 |
| Fig.2.1. TinyOS platforms .....   | 14 |
| Fig.2.2. Front and Back of the Tmote Sky module .....                       | 20 |
| Fig.2.3. Functional Block Diagram of Tmote .....                            | 20 |
| Fig.2.4. Functional Block Diagram of MSP430 .....                           | 21 |
| Fig.2.5. Received Signal Strength Indicator mapping to RF Power [dBm] ..... | 22 |
| Fig.2.6. Measured RF output power over the modulated spectrum .....         | 23 |
| Fig.2.7. S11 measurements internal inverted-F antenna no battery .....      | 24 |
| Fig.2.8. S11 measurements internal inverted-F antenna with battery .....    | 24 |
| Fig.2.9. Radiated pattern with horizontal mounting .....                    | 25 |
| Fig.2.10. Radiated pattern with vertical mounting .....                     | 25 |
| Fig.2.11. Functionality of the 10-pin expansion connector .....             | 25 |
| Fig.2.12. Functionality of the 6-pin expansion connector .....              | 26 |
| Fig.3.1. Providers .....  | 28 |
| Fig.3.2. EAGLE software .....   | 29 |
| Fig.3.3. Final schematic .....  | 20 |
| Fig.3.4. The PCB four layers .....  | 30 |
| Fig.3.5. Components arrived .....   | 30 |
| Fig.3.6. CC2420 .....   | 31 |
| Fig.3.7. Microscope .....   | 32 |
| Fig.3.8. Tin gel syringe .....  | 33 |
| Fig.3.9. Tips and needles for the syringe .....                             | 33 |
| Fig.3.10. Roll VS gel tin .....   | 34 |
| Fig.3.11. Picking the IC with the reversed tweezers .....                   | 35 |
| Fig.3.12. ERROR 1 .....   | 36 |
| Fig.3.13. Pedal timer .....   | 36 |
| Fig.3.14. Air compressor .....  | 37 |
| Fig.3.15. Time and pressure .....   | 37 |
| Fig.3.16. Pointed tweezers .....  | 38 |
| Fig.3.17. Prototyping oven .....  | 39 |
| Fig.3.18. Thermal glue syringe .....  | 39 |
| Fig.3.19. JTAG interface .....  | 41 |
| Fig.3.20. The unsoldering process .....                                     | 41 |
| Fig.3.21. FTDI Software .....   | 43 |
| Fig.3.22. Original and blank TelosB USB Controller content .....            | 43 |
| Fig.3.23. Commercial PCB with the removed components .....                  | 44 |
| Fig.3.24. CMR200T32 .....   | 46 |
| Fig.3.25. MSP430 in the programing socket .....                             | 46 |

|  |    |
|--|----|
| Fig.3.26. IAR WORKBENCH .....                                      | 47 |
| Fig.3.27. Firmware reading using ELPROTRONIC FET430 Software ..... | 47 |
| Fig.3.28. Programed MSP430 .....                                   | 48 |
| Fig.3.29. Our finished MAGmotes .....                              | 48 |
| Fig.3.30. Lead-free gel tin heating curve .....                    | 49 |
| Fig.3.31. First time plugged final mote .....                      | 49 |
| Fig.3.32. Detail of our oven-soldered board .....                  | 51 |
| Fig.4.1. Ubuntu installation .....                                 | 52 |
| Fig.4.2. TinyOS apt sources .....                                  | 53 |
| Fig.4.3. TinyOS packages .....                                     | 54 |
| Fig.4.4. Binutils from gnu .....                                   | 54 |
| Fig.4.5. TinyOS.sh .....   | 56 |
| Fig.4.6. TinyOS installed .....                                    | 56 |
| Fig.4.7. TinyOS examples applications .....                        | 57 |
| Fig.4.8. Motelist command .....                                    | 57 |
| Fig.4.9. CC2420.h .....  | 59 |
| Fig.4.10. MAGCLOUD complete system .....                           | 60 |
| Fig.4.11. Easy-Sen server module default output .....              | 62 |
| Fig.4.12. Customized output.....                                   | 64 |
| Fig.5.1. Metered inputs sensitivity .....                          | 66 |
| Fig.5.2. Energeizer alkaline AA capacity .....                     | 68 |
| Fig.5.3. Measured <i>Null</i> software consumption .....           | 69 |
| Fig.5.4. Measured full power consumption.....                      | 70 |
| Fig.5.5. Maximum consumption.....                                  | 70 |
| Fig.5.6. Finished node .....                                       | 71 |
| Fig.6.1. Solarex MSX-005 .....                                     | 74 |
| Fig.6.2. Power submodule.....                                      | 74 |
| Fig.6.4. Gumstix .....   | 75 |
| Fig.6.5. Monitoring tools .....                                    | 76 |



## TABLES INDEX

|   |    |
|---|----|
| Table 1.1. Wireless standards comparative .....                       | 7  |
| Table 2.1. Physical Characteristics .....                             | 15 |
| Table 2.2. Cost per Node .....  | 16 |
| Table 2.3. Microprocessor Specifications .....                        | 16 |
| Table 2.4. Memory Specifications .....                                | 16 |
| Table 2.5. Radio Chip Power Consumption .....                         | 17 |
| Table 2.6. Radio Chip Specification .....                             | 17 |
| Table 2.7. Typical Operating Condition .....                          | 19 |
| Table 2.8. Typical Operating Condition .....                          | 21 |
| Table 2.9. Output power configuration for the CC2420 .....            | 22 |
| Table 2.10. Typical Operating Conditions .....                        | 23 |
| Table 5.1. Inputs theoretical range .....                             | 65 |
| Table 5.2. Average and Desviation metered values for each input ..... | 67 |
| Table 5.3. Costs .....  | 67 |

## INTRODUCTION

The aim of this project is to create the skeleton of the ultimate Wireless Sensor Network for the Mediterranean Park of Technology (PMT).

The PMT is a multi-disciplinary nerve center for the innovation formed of hi-tech content enterprises, research entities and universities. It is located in Castelldefels (Spain) at <1000m of the Mediterranean coast. It covers a total surface of 28 hectares, containing numerous lakes and vegetation reserve. This is also where our campus (UPC-EETAC) is found.



**Fig.0.1.** The PMT: Location for deploying the WSN

The chosen WSN system will cover the PMT, allowing a broad spectrum of physical magnitudes to become monitored. Those include some environmental, surveillance and research related. We called this system the MAGCLOUD. Our duty is making sure the nodes will be powerful and low cost enough by building some functional hardware evidence. In this document, the rest of the MAGCLOUD parts are also designed, including the future upgrades.

To understand the power of MAGCLOUD, the reader can take a look at the requirements we initially fixed. They acted as a guide during the realization of the project and shaped the final product. They are specified in the following page.

=

On **Chapter 1** the theory behind the WSN concept is exposed. On **Chapter 2** the proper technological choices are taken. **Chapter 3** relates in great detail the nodes construction process. **Chapter 4** tells how the software was implemented. In **Chapter 5** nodes behavior is metered and discussed. **Chapter 6** introduces some future upgrades. **Chapter 7** tells about the possible environmental impact.

**CHEAP WHILE POWERFUL:** The nodes have to be enough reliable to become a real option for the majority of uses. Even outdoor, (inside a protective case) they have to resist for very extended periods without physical failure.



MAGCLOUD has to support extensive ready to use third party code. Every piece of software has to be open, cost free and provide the sources. This way, the only expenses will be the ones related to the hardware. Also because of this, upgrades in any direction will be possible whenever required. A robust and intuitive support platform including massive published documents is required. Someone with no previous training has to be able to understand how to operate the MAGCLOUD skeleton or any of the upgrades in a short time.

**SCALABLE:** Every node has to reach a long range by itself and has to provide a socket to customize the radiation pattern and gain by using an external antenna. Adding or removing any physical nodes of the wireless sensor cloud has to make it evolve to an optimal addressing and routing new formula without user intervention. In an easy upgrade, MAGCLOUD will be able to grow to an almost unlimited number of nodes or even combinations of subsectors located in physically separated locations.



A computer using a common OS is enough to interface the cloud, serve the sensor data to a network and store or render the magnitudes. In an easy upgrade, this machine should be able to classify, process, store in a powerful database, graph and serve the data to the Internet in a robust way. Then, any connected HTML capable device will be able to realize tasks ranging from visualizing information to securely administrating MAGCLOUD.

On publication date, there has to exist an extensive number of magnitude acquisition modules offering free schematic ready to plug and play. Even more important, MAGCLOUD has to accept any measurable magnitude fitting within the cloud bandwidth by just the upgrader designing a very simple adaptation stage.

**ENVIRONMENTAL FRIENDLY:** The deployment of MAGCLOUD is the perfect option for taking care of the environment but it has also to set example trough its life cycle: The introduction of any dangerous substances in the final hardware construction formula will be avoided.



They have to require a very little amount of power to work autonomously for long periods of time when using tiny batteries. In an easy upgrade, all the used energy will come from a solar panel. Using it along with a capacitor will make the nodes of the cloud useable without physical maintenance until hardware failure.

## CHAPTER 1. WSN TECHNOLOGIES

The aim of this chapter is to introduce the WSN concept, the WSN applications and some of the technologies that have been recently developed to improve its efficiency.

### 1.1 Wireless Sensor Network

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices that use sensors to cooperatively monitor physical and/or environmental conditions at the different locations.

The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. Today, those networks are used in many industrial and consumer applications. Industrial process control, environment monitoring or home automation are examples of common WSN applications.



**Fig.1.1.** WSN Application Areas

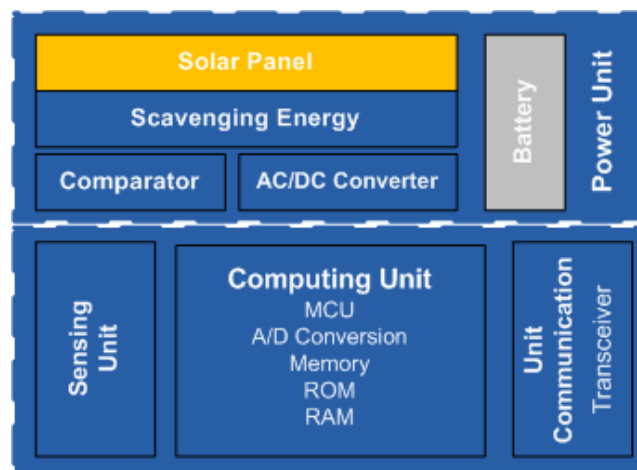
Networked sensing offers unique advantages over traditional centralized approaches. Dense networks of distributed communicating sensors can increase energy efficiency by the multi-hop topology of the network. The greatest advantages of networked sensing are robustness and scalability. A decentralized sensing system is inherently more robust against individual sensor node or link failures, because of redundancy in the network.

The main characteristics of a WSN are:

- Low power consumption
- Ability to cope with node failures
- Mobility of nodes
- Dynamic network topology
- Communication failures
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions
- Ease of use
- Low cost devices

As said before, a WSN is essentially a computer network consisting of many small, intercommunicated computers equipped with one or several sensors. Each small computer represents a node of the network. These nodes are called sensor nodes or motes. Normally, each basic sensor node is composed by:

- A processing unit: Microcontroller
- A communication unit: Transceiver
- A power unit
- A sensing unit



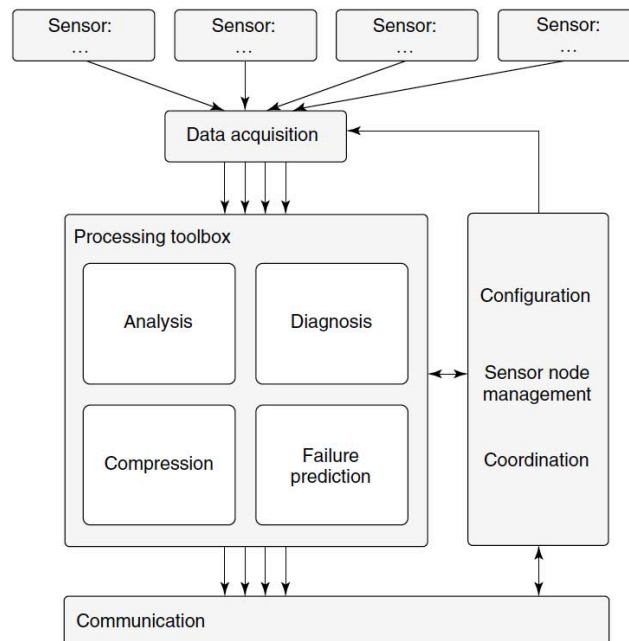
**Fig.1.2.** Hardware Block diagram of a WSN mote

The communication within the network is established using radio frequency transmission techniques. The sensor nodes typically form a multihop mesh network by establishing communication links with neighbour nodes.

Multihop networks provide different advantages when magnitudes data has to be transmitted over long distances. Mainly, the network robustness over sensor node failure and the high power efficiency make multihop networks so attractive for monitoring applications.

The sensor nodes are the main components of a WSN. These sensor nodes have to provide the following basic functionality:

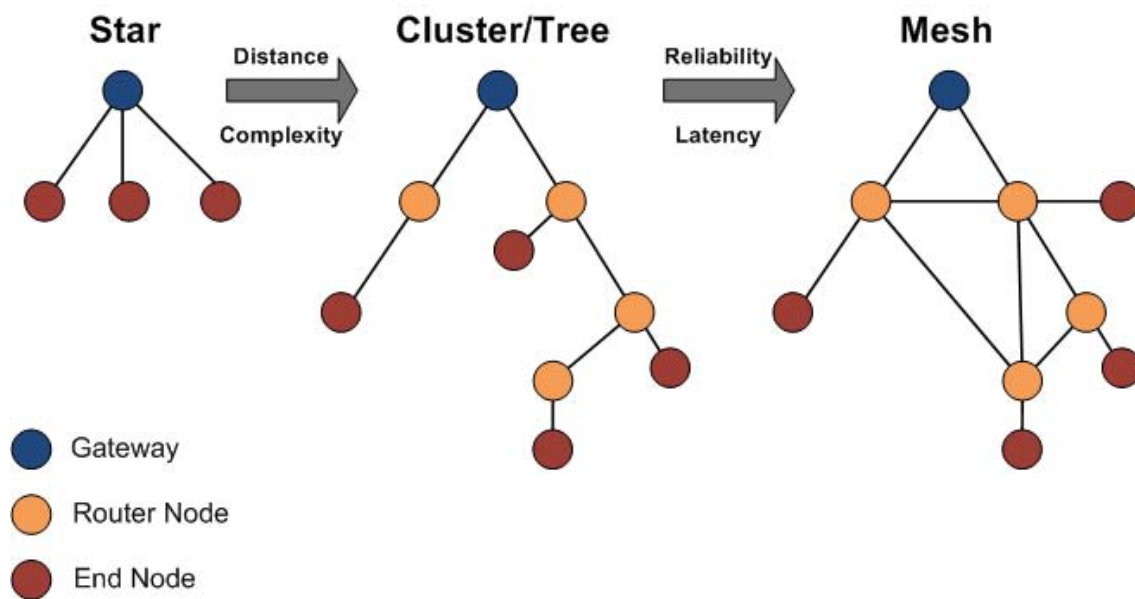
- Signal conditioning and data acquisition for different sensors
- Temporary storage of the acquired data
- Processing of the data
- Analysis of the processed data for diagnosis and potentially, alert generation
- Self monitoring (e.g., supply voltage)
- Scheduling and execution of the measurement tasks
- Management of the sensor node configuration (e.g., changing the sampling rate and reprogramming of data processing algorithms)
- Reception, transmission, and forwarding of data packets
- Coordination and management of communication and networking



**Fig.1.3.** Basic functionality of a sensor node

The cost of a sensor node ranges from some few euros to hundreds, depending on the complexity of the particular node. Size and cost constraints result on compromised resources such as energy, memory, computational speed and communications bandwidth.

The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network (*Fig.1.4*). The propagation technique between the hops of the network can be routing or flooding.



**Fig.1.4.** WSN Network Topologies

## 1.2 IEEE 802.15.4 technology

IEEE 802.15.4 wireless technology is a standard that specifies the physical layer and media access control (MAC) for low-rate wireless personal area networks (LR-WPANs). The key features of 802.15.4 wireless technology are low complexity, low cost, low power consumption, low data rate transmissions, to be supported by cheap either fixed or moving devices. The main field of application of this technology is the implementation of WSNs.

In order to see the main differences between the most popular existing radio standards and 802.15.4, the following table shows the comparison between them.



**Table 1.1.** Wireless standards comparative

|                           | IEEE 802.15.4                             | Bluetooth                            | 802.11b             |
|---------------------------|---|--------------------------------------|---------------------|
| <b>Frequency</b>          | 868 MHz, 902-982 MHz, 2.4 GHz             | 2.4 GHz                              | 2.4 GHz             |
| <b>Bandwidth</b>          | 250 kbps                                  | 1-3 Mbps                             | 11 Mbps             |
| <b>Operational Range</b>  | 10-75 m                                   | 10-50 m                              | 100 m               |
| <b>Network Topologies</b> | Star, mesh, and cluster tree              | Ad-hoc, piconet                      | Point-to-multipoint |
| <b>Battery life</b>       | years                                     | weeks                                | 1 week              |
| <b>Advantages</b>         | Low power, cost, flexibility, scalability | Convenience                          | Speed, ubiquity     |
| <b>Focus Applications</b> | Monitoring, control and location          | Wire substitution in small distances | Web, mail, video    |

As seen some of the main differences between these standards relapse on the bandwidth and the battery life. While 802.15.4 has a small bandwidth with a low energetic consumption, at the far end, 802.11b (like other technologies belonging to the IEE 802.11 family) offers a high bandwidth without taking care of the energy requirements, which is essential for our application.

### 1.2.1 Components of WPAN

IEEE 802.15.4 defines two types of nodes:

- Full Function Devices (FFD)
- Reduced Function Devices (RFD)

A network shall include at least one FFD, operating as the PAN coordinator. RFDs are used on more simple applications where no large amounts of data have to be sent. An FFD can talk to both devices while an RFD can only talk to an FFD.

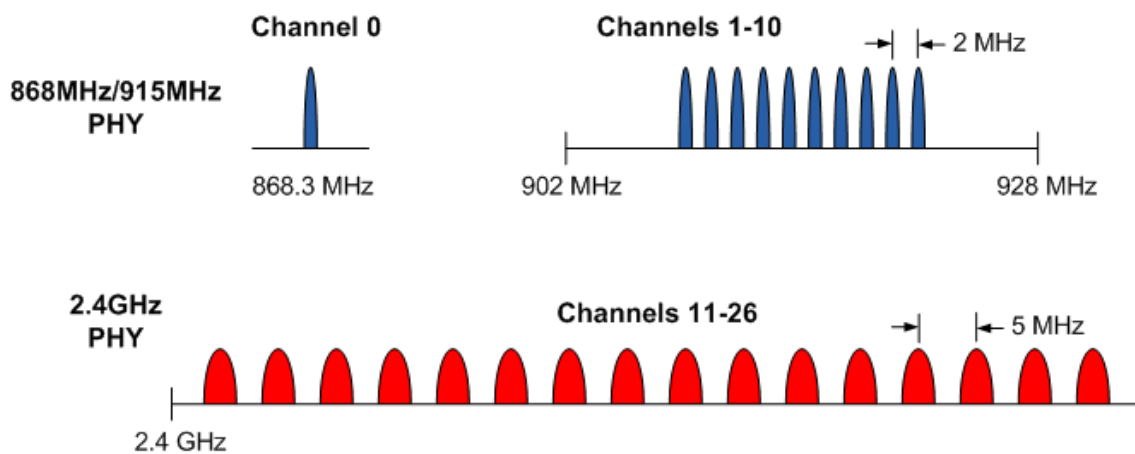
### 1.2.2 Physical Layer

The 802.15.4 physical layer operates in three different unlicensed bands, (and with different modalities) according to the geographical area where the system is deployed. However, spread spectrum techniques are wherever mandatory to reduce the interference level in shared unlicensed bands.

IEEE 802.15.4 specifies a total of 27 half-duplex channels across the three frequency bands and it is organized as follows:



- 868 MHz band: only a single channel with data rate 20 kbps is available;  $-92$  dBm RF sensitivity required and ideal transmission range approximately equal to 1 km.
- 915 MHz band: ten channels with rate 40 kbps are available; the receiver sensitivity and the ideal transmission range are 1 km.
- 2.4 GHz ISM band: sixteen channels with data rate 250 kbps available; minimum  $-85$  dBm RF sensitivity required and ideal transmission range equal to 220 m.

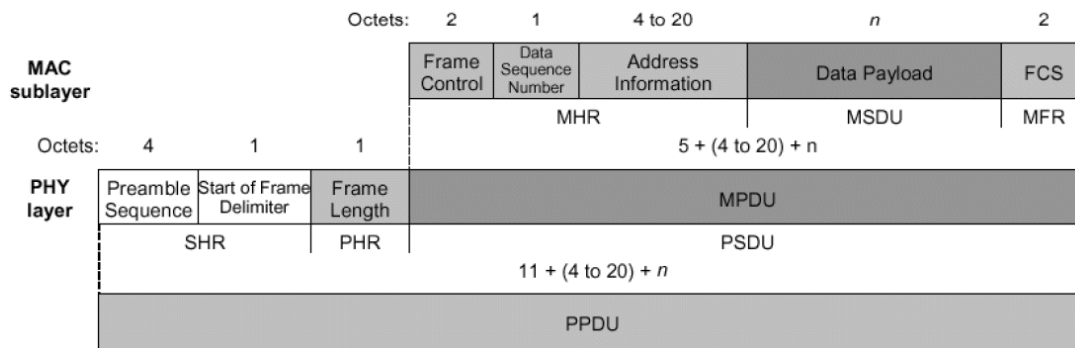


**Fig.1.5.** Operating frequency bands

According to the energy efficiency issue, low rate and low duty cycle are provided. IEEE 802.15.4 is ready to work with low-duty cycles. It means the transceiver can be sleeping most of the time (up to 99% on average) while the receiving and sending tasks can be set to take just a small part of the devices' energy. This percentage depends on the kind of communication model used. If beacon mode is used (star or PAN networks) the minimum amount of time used to transmit/receive this frames will increase the total time the transceiver is used. They can be sleeping for minutes or hours and wake up all at the same time to perform an Adhoc communication creating a mesh network just when really needed.

### 1.2.3 MAC Layer

The medium access control sublayer (MAC) allows the transmission of MAC frames through the use of the physical channel and its fundamental goal is to reduce or avoid packet collisions in the medium.



**Fig.1.6.** MAC frame format

MAC and PHY layers packet structure is shown in the figure above (*Fig.1.6*).

802.15.4 uses two techniques to avoid all the nodes start emitting at the same time: CSMA-CA and GTS.

- The most common is the Carrier Sense Multiple Access-Collision Avoidance (CSMA-CA). This method is described as follows: each node listen the medium prior to transmit. If the energy found higher of a specific level the node the transceiver waits during a random time (including in an interval) and tries again. There is a parameter defined in the standard: macMinBE which sets the back-off exponent to be used when calculating this time slot.
- The second one is Guaranteed Time Slots (GTS). This systems uses a centralized node (PAN coordinator) which gives slots of time to each node so that any knows when they have to transmit. There are 16 possible slots of time. As a first step a node must to send to the PAN coordinator a GTS request message, as response the coordinator will send a beacon message containing the slot allocated and the number of slots assigned.

### 1.2.4 Associated protocols

There are several protocols using 802.15.4 as its MAC layer. The most known are:

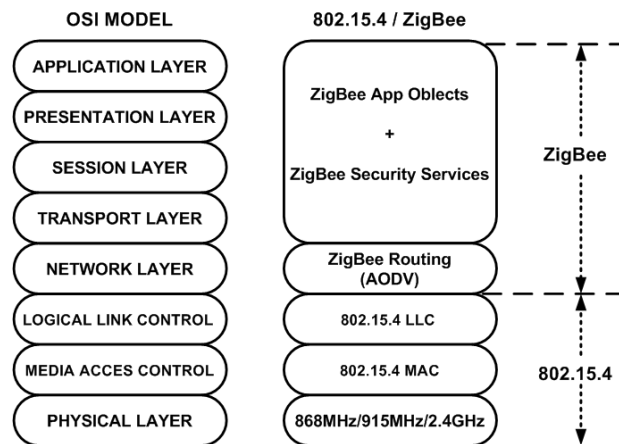
- ZigBee: is a mesh-networking standard intended for uses such as medical data collection, consumer devices like television remote controls, and home automation.
- Wireless HART: It is the wireless version of the HART protocol, which is the most used in the automation and industrial applications requiring real time. It uses Time Synchronized Mesh Protocol (TSMP). A "time coordinator" node is required in order to assign the time slot to all the motes.
- ISA-SP100: It also focuses in the process and factory automation. It is being developed by the Systems and Automation Society (ISA) and tries to be a standard for this kind of uses.

- IETF IPv6 – 6LoWPAN: As the same name points it is the implementation of the IPv6 stack on top of 802.15.4 to bring devices Internet capability.
- New Mesh protocols: Many companies are implementing new mesh protocols over the 802.15.4 MAC layer.

Nowadays, the most extended protocol in WSN networks and the one that has the minor energetic consumption is Zigbee. Multiple commercial systems and equipment are actually supporting it. This, in combination with its features, has made it the chosen protocol for MAGCLOUD. 6LowPan would be also a great candidate, permitting an almost unlimited number of simultaneous nodes but we rejected as it is not enough polished at the moment.

### 1.3 ZigBee

The ZigBee standard defines the communication layer at level 3 and uppers in the OSI model. Its main purpose is to create a network topology (hierarchy) to let a number of devices communicate among them and to set extra communication features such as authentication, encryption, association and the upper layer application services. ZigBee was created by a set of companies that form the ZigBee Alliance.



**Fig.1.7.** 802.15.4/ZigBee Stack

ZigBee is a specification for a suite of high level communication protocols using small, low-cost, low-power digital radios based on the IEEE 802.15.4 standard. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other WPANs, such as Bluetooth. ZigBee is targeted at radio-frequency (RF) applications that require a low data rate, long battery life, and secure networking.

### 1.3.1 Key Features

The main features of the ZigBee standard are:

- **Low Power:** The benefits of simple, cost-effective, low-power wireless connectivity that ZigBee technology provides is addressed to a variety of markets, including industrial and home monitoring, control and automation, or health care diagnostics. Freescale provides all the building blocks used in a complete ZigBee-compliant platform solution: the RF transceiver, MAC and ZigBee software, microcontrollers and sensors. The development hardware and reference designs provide developers with the tools they need to easily and quickly implement these building blocks.
- **Robust:** 802.15.4 provides a robust foundation for ZigBee, ensuring a reliable solution in noisy environments. Features such as energy detection, clear channel assessment and channel selection help the device pick the best possible channel, avoiding other wireless networks such as Wi-Fi. Message acknowledgement helps to ensure that the data was delivered to its destination. Finally, multiple levels of security ensure that the network and data remain intact and safe.
- **Mesh Networking:** The ability to cover large areas with routers is one of the key features of ZigBee that helps differentiate itself from other technologies. Mesh networking can extend the range of the network through routing, while self healing increases the reliability of the network by re-routing a message in case of a node failure.
- **Interoperability:** The ZigBee Alliance helps ensure interoperability between vendors by creating testing and certification programs for ZigBee devices. Users can be assured the devices that go through certification testing and use the ZigBee logo will work with other devices based on the same applications. This provides end customers peace of mind by creating brand awareness of products with the ZigBee logo.

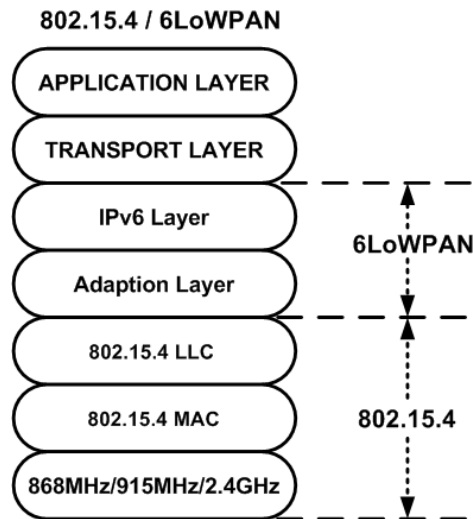


Fig.1.8. ZigBee application areas

## 1.4 6LoWPAN

6LoWPAN is an acronym of IPv6 over Low power Wireless Personal Area Networks. 6LoWPAN is the name of a working group in the internet area of the IETF. The 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices" and that low-power devices with limited processing capabilities should be able to participate in the *Internet of Things*.

The 6LoWPAN group has defined encapsulation and header compression mechanisms that allow IPv6 packets to be sent to and received from over IEEE 802.15.4 based networks. IPv4 and IPv6 are the workhorses of data delivery for local-area networks, metropolitan area networks, and wide-area networks such as the Internet. The specification to allow the use of IPv6 over IEEE 802.15.4 networks is defined in RFC4919.



**Fig.1.9.** 802.15.4/6LoWPAN Stack

### 1.4.1 Key Features

The main features of 6LoWPAN specification are:

- Support for both 16-bit short (unique within a PAN) or IEEE 64-bit extended MAC addresses
- Low bandwidth. Data rates of 250 kbps, 40 kbps, and 20 kbps for each previously defined physical layers
- Topologies include star and mesh operation
- Low power and low cost devices
- Large number of devices supported
- 6LoWPAN devices may sleep for long periods of time in order to save energy

## CHAPTER 2. CHOSING THE PLATFORM

In this chapter, the most interesting choices are evaluated in order to decide which one best accomplishes our requirements.

### 2.1 Software Platform

Unlike general-purpose operating systems for standard PCs such as Windows or Linux, the WSN software platforms are highly tailored to the limited node hardware. These WSN software frameworks are not full-blown operating systems, since they lack a powerful scheduler, memory management, and file system support. However, these frameworks are widely referred to as WSN operating systems. The most used WSN operating systems are Contiki, Mantis, SOS and TinyOS.

TinyOS is the most widespread WSN operating system and the one offering more free software and support. TinyOS is the choice for MAGCLOUD.

#### 2.1.1 TinyOs

TinyOS is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal area networks, smart buildings, and smart meters. A worldwide community from academic and industry use, develop, and support the operating system as well as its associated tools, averaging 35,000 downloads a year.



TinyOS is written in nesC, an extension to the C language, which supports event-driven component-based programming. The basic concept of component-based programming is to decompose the program into functionally self-contained components. These components interact by exchanging messages through interfaces. The components are event-driven. Events can originate from the environment (a certain sensor reading exceeds a threshold) or from other components, triggering a specific action. The main advantage of this component-based approach is the reusability of components.

The nesC language extension introduces several additional keywords to describe a TinyOS component and its interfaces. NesC and TinyOS are both Open Source projects supported by a fast growing community.

TinyOS has been ported to over a dozen WSN platforms and provides a concurrency model and mechanisms for structuring, naming, and linking software components into a robust network embedded system. Today, TinyOS is a sort of de facto standard in WSN programming and widely used in the WSN community.

As a result, a huge amount of software components for various sensors, network protocols, algorithms, and other WSN related topics are freely available on the network.

TinyOS is the OS MAGCLOUD nodes will be running. Of the many versions available, we are referring to the TinyOS 2.1.1 for the rest of the document. TinyOS 2.1.1 is the most recent version at publication date. It includes:

- Support for the epic, mulle, and shimmer2 platforms,
- Support for 6LoWPAN, an IPv6 networking layer within the TinyOS network,
- Support for simple, uniform low-power networking across many protocols,
- Support for security on the CC2420 radio
- Improvements to many existing services and protocols, including the inclusion of a new dissemination protocol (DHV), improvements to CTP, improved TOSThreads documentation, and numerous bug fixes.

## 2.2 Hardware Platform

As it was mentioned in the first chapter, a sensor node is composed of one or more sensors, a signal conditioning unit, an analog-to-digital conversion module (ADC), a processing unit with memory, a radio transceiver, and a power supply (*Fig.1.2.*).

If the sensor nodes are actually deployed in the field, especially in harsh environments, they have to be protected against chemical and mechanical impacts. Therefore, an adequate packaging for the hardware is required. There are a lot of different types of motes which supports TinyOs, in the next tables are the comparison of recently used platforms. These table only shows a selection.



**Fig.2.1.** TinyOs platforms

- **TelosB/TMote Sky:** Wireless sensor modules developed from research carried out at UC Berkeley and currently available in similar form factors from both Sentilla and CrossBow Technology.
- **Mica2/MicaZ:** Another wireless sensor networking mote family from CrossBow Technology.
- **SHIMMER:** SHIMMER (Sensing Health with Intelligence, Modularity, Mobility, and Experimental Reusability) is a wireless sensor platform designed to support wearable applications.
- **IRIS:** latest wireless sensor network module from Crossbow Technologies. Includes several improvements over the Mica2/MicaZ family of products as increased transmission range.
- **Sun SPOT:** Sun “Small Programmable Object Technology” (SPOT) is a wireless sensor network mote from Sun Microsystems. Both the hardware and software are open-source.
- **EZ430-RF2480/2500:** EZ430-RF2480 and EZ430-RF2500 wireless networking solutions from Texas Instruments incorporate the MSP430 microprocessor and CC2480/2500 radio transceiver on each board.

### 2.2.1 Comparative Platforms

To select the mote that fits better our requirements, we have developed a comparative of the main platforms that support TinyOS, divided into 5 separate categories: general parameters, processor and memory, communication capabilities, sensor support and power consumption.

Our mote must have a small size to be able to be integrated into any possible scenario, either directly (indoor scenario) or inside a sealed box (outdoor scenario). As shown in the following table (*Table 2.1.*) all these platforms have a small size.

**Table 2.1.** Physical Characteristics

| Mote Platform              | WxLxH<br>[inches]  | Weight<br>No batt [g] | Weight<br>Batt [g] |
|----------------------------|--------------------|-----------------------|--------------------|
| <b>TelosB/TMote</b>        | 1.26 x 2.58 x 0.26 | 14.93                 | 63.05              |
| <b>MicaZ/Mica2</b>         | 1.25 x 2.25 x 0.25 | 15.70                 | 63.82              |
| <b>IRIS</b>                | 1.25 x 2.25 x 0.25 | 21.29                 | 69.40              |
| <b>Sun SPOT</b>            | 2.5 x 1.5 x 1      | 33.49                 | 58.08              |
| <b>EZ430-RF2500 (USB)</b>  | 1.16 x 3.17 x 0.43 | 1.80                  | 30.89              |
| <b>EZ430-RF2480 (Batt)</b> | 1.02 x 3.72 x 0.55 | 1.80                  | 30.89              |

*NOTE: All sizes are for mote assemblies i.e. PCB, batteries, enclosures, etc.*



The purpose of MAGCLOUD is to achieve a complete system with the best quality-price ratio. So we have to achieve all our initial expectatives with the minimal cost. The usual prices for each platform are shown in the following table (*Table 2.2.*).

**Table 2.2.** Cost per Node

| Mote Platform              | Price            | Comments                           |
|----------------------------|------------------|------------------------------------|
| <b>TelosB/TMote</b>        | US\$99 / US\$139 | no sensors / with sensors          |
| <b>MicaZ/Mica2</b>         | US\$99           | Mica2 no longer available          |
| <b>SHIMMER</b>             | EUR199           | base SHIMMER SDK with 2 boards     |
| <b>IRIS</b>                | US\$115          |                                    |
| <b>Sun SPOT</b>            | US\$750          | 3 base boards and 2 sensors boards |
| <b>EZ430-RF2500 (USB)</b>  | US\$99           | 3 nodes, one is a USB interface    |
| <b>EZ430-RF2480 (Batt)</b> | US\$49           | 2 nodes                            |

Processor and memories (RAM, Flash and EEPROM) are essential to chose for one or other platform. The tables below show the specifications for each one.

**Table 2.3.** Microprocessor Specifications

| Mote Platform             | uProcessor        | Bus    | Clock  |
|---------------------------|-------------------|--------|--------|
| <b>TelosB/TMote</b>       | TI MSP430F1611    | 16-bit | 4-8MHz |
| <b>MicaZ/Mica2</b>        | Atmel Atmega 128L | 8-bit  | 8MHz   |
| <b>SHIMMER</b>            | TI MSP430F1611    | 16-bit | 4-8MHz |
| <b>IRIS</b>               | Atmel ATmega 1281 | 8-bit  | 8MHz   |
| <b>Sun SPOT</b>           | Atmel AT91RM9200  | 32-bit | 180MHz |
| <b>EZ430-RF2500 (USB)</b> | TI MSP430F2274    | 16-bit | 16MHz  |

**Table 2.4.** Memory Specifications

| Mote Platform       | RAM  | Flash | EEPROM |
|---------------------|------|-------|--------|
| <b>TelosB/TMote</b> | 10K  | 48K   | 1M     |
| <b>MicaZ/Mica2</b>  | 4K   | 128K  | 512K   |
| <b>SHIMMER</b>      | 10K  | 48K   | none   |
| <b>IRIS</b>         | 8K   | 640K  | 4K     |
| <b>Sun SPOT</b>     | 512K | 4M    | none   |
| <b>EZ430-RF2500</b> | 1K   | 32K   | none   |

There is no doubt that an essential requirement of MAGCLOUD is low consumption. Therefore the radio role is fundamental in order to choose the platform. In sleep mode our node must have an almost null consumption and as small as possible during the transmission or reception. According to the table, the best provisions are offered by CC2420 and CC2500.

**Table 2.5.** Radio Chip Power Consumption

| Radio Module           | Sleep                | Idle/Rx              | Tx (mA)      |
|------------------------|----------------------|----------------------|--------------|
| <b>TI CC1000</b>       | 0.2 $\mu$ A          | 74 $\mu$ A to 7.4 mA | 10.4 (0 dBm) |
| <b>TI CC2420</b>       | 0.02 - 426 $\mu$ A   | 18.8 mA              | 17.4 (0 dBm) |
| <b>TI CC2500</b>       | 400 nA - 160 $\mu$ A | 13.3 - 19.6 mA       | 21.2 (0 dBm) |
| <b>TI CC2480</b>       | 0.3 - 190 $\mu$ A    | 26.7 mA              | 26.9 (0 dBm) |
| <b>Mitsumi WML-C46</b> | 50 $\mu$ A - 1.4 mA  | 40 mA                | 60.0 (0 dBm) |

**Table 2.6.** Radio Chip Specification

| Radio Module     | Frequency (MHz) | Modulation     | Data Rate  | Tx Power (dBm) | Rx Sensitivity (dBm) |
|------------------|-----------------|----------------|------------|----------------|----------------------|
| <b>TI CC1000</b> | 300 - 1000      | FSK            | 76.8 kBaud | -20 - 10       | -110                 |
| <b>TI CC2420</b> | 2400 - 2483.5   | OQPSK          | 250 kbps   | -24 - 0        | -95                  |
| <b>TI CC2500</b> | 2400 - 2483.5   | OOK, GFSK, MSK | 500 kBaud  | -30 - 1        | -108                 |
| <b>TI CC2480</b> | 2400 - 2483.5   | OQPSK          | 250 kbps   | -55.8 - 0      | -92                  |
| <b>WML-C46</b>   | 2400 - 2483.5   | GFSK           | 721 kbps   | -6 - 14        | -82                  |

At this point we see that the nodes that fit better our requirements are TMote and EZ430-RF2500.

The maximum versatility will allow us to go further in the future. The factors to consider are:

- The quantity of usable ports
- The native capacity of modify the gain or radiation diagram using an external antenna.
- The ability of connecting to another machine by other channels (for ex. USB+JTAG)

Only the TMote Sky nodes allow every one of these possibilities.

It should be added that nowadays, TMote are the most extended nodes and the ones with a greater support from the community both hardware and software level.

Luckily, our school possesses a complete mounted GRID of TMote boards and also individual nodes are available to study.

It's clear that the platform that fits better our expectations is TMOTE SKY.

## 2.3 TMOTE SKY

As said before, TMote Sky is an ultra low power wireless module for use in sensor networks, monitoring applications, and rapid application prototyping. TMote Sky leverages industry standards like USB and IEEE 802.15.4 (see 1.2) to interoperate seamlessly with other devices. By using industry standards, integrating humidity, temperature, and light sensors, and providing flexible interconnection with peripherals, TMote Sky enables a wide range of mesh network applications. TMote Sky includes increased performance, functionality, and expansion. With TinyOS (see 2.1.1) support out-of-the-box, TMote Sky leverages emerging wireless protocols and the open source software movement.

TMote Sky is part of a line of modules featuring on-board sensors to increase robustness while decreasing cost and package size.

### 2.3.1 Key Features

The main features of this mote are:

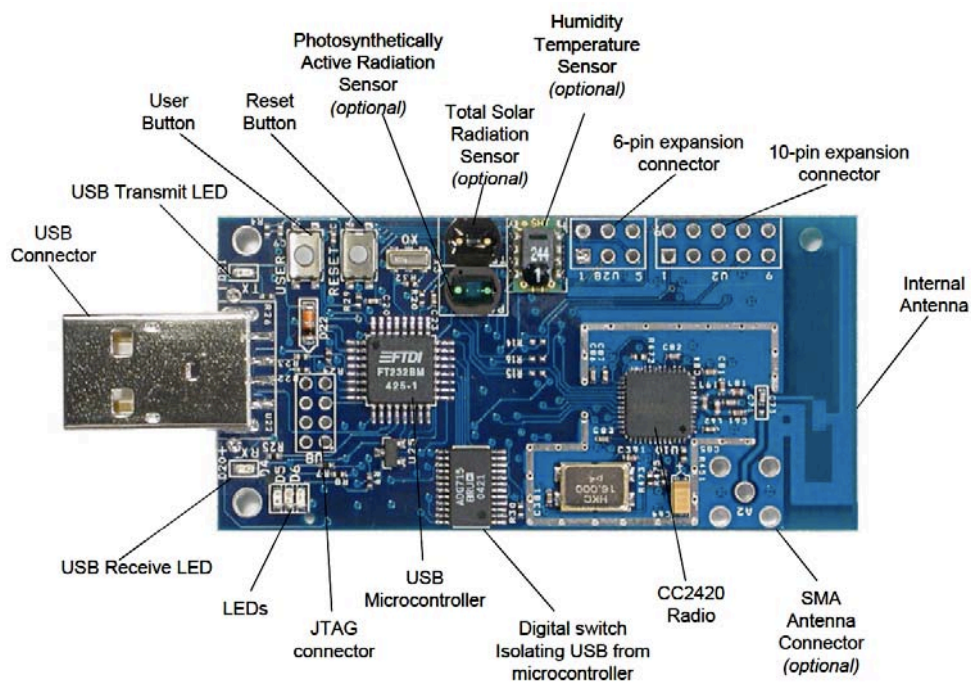
- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver
- Interoperability with other IEEE 802.15.4 devices
- 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
- Integrated ADC, DAC, Supply Voltage Supervisor, and DMA Controller
- Integrated onboard antenna with 50m range indoors / 125m range outdoors
- Integrated Humidity, Temperature, and Light sensors
- Ultra low current consumption
- Fast wakeup from sleep ( $<6\mu\text{s}$ )
- Hardware link-layer encryption and authentication
- Programming and data collection via USB
- 16-pin expansion support and optional SMA antenna connector
- TinyOS support : mesh networking and communication implementation
- Complies with FCC Part 15 and Industry Canada regulations
- Environmentally friendly – complies with RoHS regulations

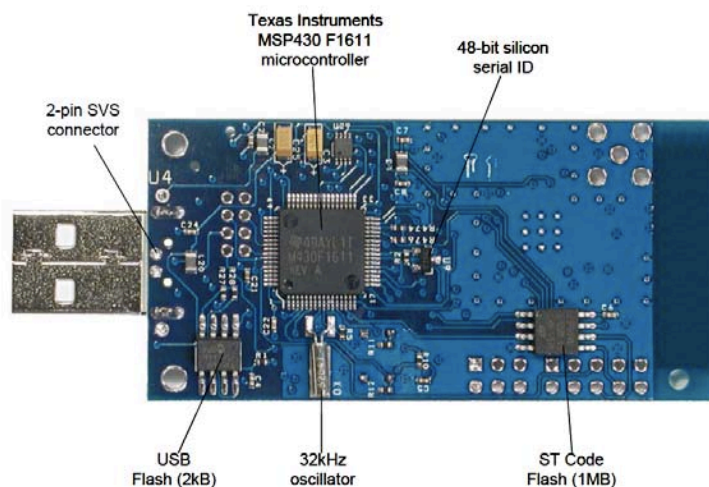
**Table 2.7.** Typical Operating Condition

|   |                             |              |              |              |
|---|-----------------------------|--------------|--------------|--------------|
| <b>Supply voltage</b>                                 | 2.1 to 3.6 V                |              |              |              |
| <b>Supply voltage during flash memory programming</b> | 2.7 to 3.6 V                |              |              |              |
| <b>Operating free air temperature</b>                 | -40 to 85 °C                |              |              |              |
| <b>Antenna</b>  | Integrated onboard          |              |              |              |
| <b>Range</b>  | 50 m indoor, 125 m outdoor  |              |              |              |
| <b>I/O</b>  | 16-pin expansion connector  |              |              |              |
| <b>External oscillator</b>                            | 32 KHz                      |              |              |              |
| <b>Power</b>  | Two AA batteries / USB port |              |              |              |
| <b>Current consumption</b>                            | <b>MCU</b>                  | <b>Radio</b> | <b>NOM</b>   | <b>MAX</b>   |
|   | ON                          | RX           | 21.8 mA      | 23 mA        |
|   | ON                          | TX           | 19.5 mA      | 21 mA        |
|   | ON                          | OFF          | 1800 $\mu$ A | 2400 $\mu$ A |
|   | idle                        | OFF          | 54.5 $\mu$ A | 1200 $\mu$ A |
|   | standby                     | OFF          | 5.1 $\mu$ A  | 21 $\mu$ A   |

### 2.3.2 Module Description

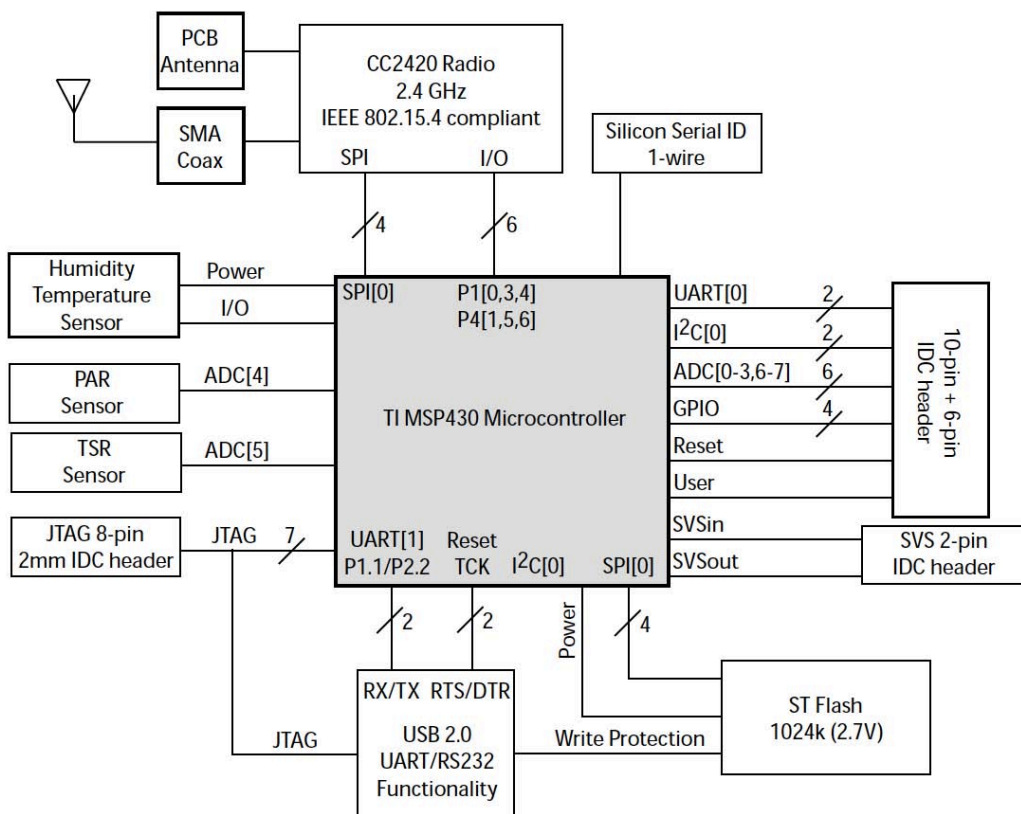
Like a normal WSN mote, the TMote Sky module is composed by a sensors, radio, antenna, microcontroller, and programming capabilities.





**Fig.2.2.** Front and Back of the TMote Sky module

In the next figure (*Fig.2.3.*) it is shown the functional block diagram of the TMote Sky module, its components, and buses.



**Fig.2.3.** Functional Block Diagram of TMote

### 2.3.3 Microprocessor

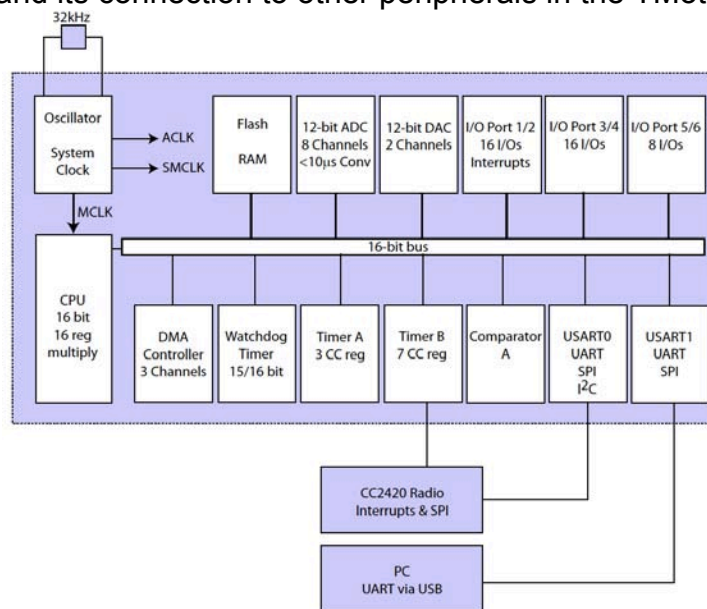
The low power operation of the TMote Sky module is due to the ultra low power Texas Instruments MSP430 F1611 microcontroller featuring 10kB of RAM, 48kB of flash, and 128B of information storage. This 16-bit RISC processor features extremely low active and sleep current consumption that permits TMote to run for years on a single pair of AA batteries.

The MSP430 has 8 external ADC ports and 8 internal ADC ports. The ADC internal ports may be used to read the internal thermistor or monitor the battery voltage. A variety of peripherals are available including SPI, UART, digital I/O ports, Watchdog timer, and Timers with capture and compare functionality. The F1611 also includes a 2-port 12-bit DAC module, Supply Voltage Supervisor, and 3-port DMA controller.

**Table 2.7.** Typical Operating Condition

|   | MIN | NOM    | MAX | UNIT |
|---|-----|--------|-----|------|
| <b>Supply voltage during program execution</b>  | 1.8 |        | 3.6 | V    |
| <b>Supply voltage during flash memory prog.</b> | 2.7 |        | 3.6 | V    |
| <b>Operating free air temperature</b>           | -40 |        | 85  | °C   |
| <b>Low frequency crystal frequency</b>          |     | 32.768 |     | KHz  |
| <b>Active current at Vcc = 3V, 1MHz</b>         |     | 500    | 600 | μA   |
| <b>Sleep current Vcc=3V, 32.7kHz active</b>     |     | 2.6    | 3.0 | μA   |
| <b>Wake up from LPM3 (low power mode)</b>       |     |        | 6   | μs   |

In the next figure it is shown the functional block diagram of the TI MSP430 microcontroller and its connection to other peripherals in the TMote Sky module.



**Fig.2.4.** Functional Block Diagram of MSP430

### 2.3.4 Radio

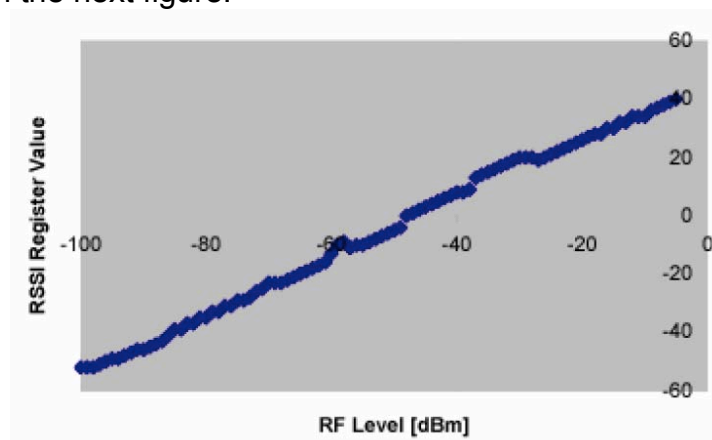
TMote Sky features the Chipcon CC2420 radio for wireless communications. The CC2420 is an IEEE 802.15.4 compliant radio providing the PHY and some MAC functions. With sensitivity exceeding the IEEE 802.15.4 specification and low power operation, the CC2420 provides reliable wireless communication. The CC2420 is highly configurable for many applications with the default radio settings providing IEEE 802.15.4 compliance. The CC2420 is controlled by the TI MSP430 microcontroller through the SPI port and a series of digital I/O lines and interrupts. The radio may be shut off by the microcontroller for low power duty cycled operation.

The CC2420 has programmable output power (*Table 2.8.*). Common CC2420 register values and their corresponding current consumption and output power are shown in the next table:

**Table 2.8.** Output power configuration for the CC2420

| PA_LEVEL | TXCTRL register | Output Power [dBm] | Current Consumption [mA] |
|----------|-----------------|--------------------|--------------------------|
| 31       | 0xA0FF          | 0                  | 17.4                     |
| 27       | 0xA0FB          | -1                 | 16.5                     |
| 23       | 0xA0F7          | -3                 | 15.2                     |
| 19       | 0xA0F3          | -5                 | 13.9                     |
| 15       | 0xA0EF          | -7                 | 12.5                     |
| 11       | 0xA0EB          | -10                | 11.2                     |
| 7        | 0xA0E7          | -15                | 9.9                      |
| 3        | 0xA0E3          | -25                | 8.5                      |

The CC2420 provides a digital received signal strength indicator (RSSI) that may be read any time. Additionally, on each packet reception, the CC2420 samples the first eight chips, calculates the error rate, and produces a link quality indication (LQI) value with each received packet. A mapping from RSSI to the RF level in dBm is shown in the next figure.



**Fig.2.5.** Received Signal Strength Indicator mapping to RF Power [dBm]

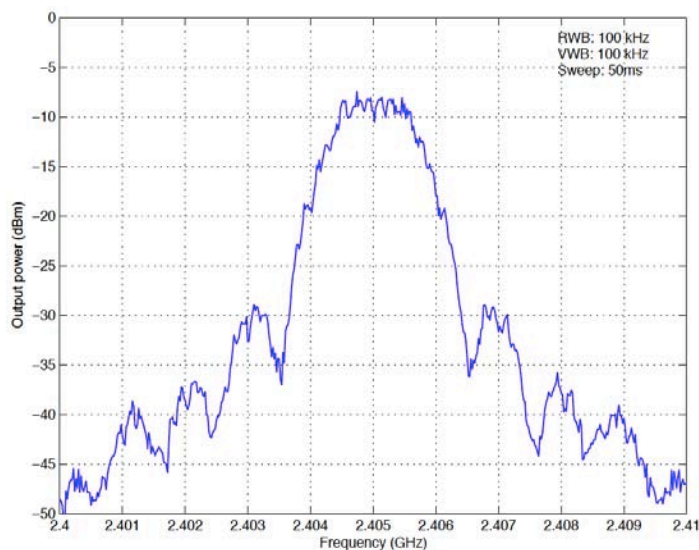


**Table 2.9.** Typical Operating Conditions

|  | MIN  | NOM  | MAX    | UNIT |
|--|------|------|--------|------|
| Supply vol. during radio operation (Vreg on)   | 2.1  |      | 3.6    | V    |
| Operating free air temperature                 | -40  |      | 85     | °C   |
| RF frequency range                             | 2400 |      | 2483.5 | MHz  |
| Transmit bit rate                              | 250  |      | 250    | Kbps |
| Nominal output power                           | -3   | 0    |        | dBm  |
| Programmable output power range                |      | 40   |        | dBm  |
| Receiver sensitivity                           | -90  | -94  |        | dBm  |
| Current consump.: Radio transm. at 0 dBm       |      | 17.4 |        | mA   |
| Current consumption: Radio receiving           |      | 19.7 |        | mA   |
| Current consump.: Radio on, Oscillator on      |      | 365  |        | μA   |
| Current consump.: Idle mode, Oscillator off    |      | 20   |        | μA   |
| Current consumption: Power Down mode, Vreg off |      | 1    |        | μA   |
| Voltage regulator current draw                 | 13   | 20   | 29     | μA   |
| Radio oscillator startup time                  |      | 580  | 860    | μs   |

#### 2.3.4.1 Measured Output Power

The RF output power of the TMote Sky module from the CC2420 radio is shown in Fig.2.6. For this test, the TMote Sky module is transmitting at 2.405GHz (IEEE 802.15.4 channel 11) using the O-QPSK modulation with DSSS. The CC2420 programmed output power is set to 0 dBm. The measured output power of the entire modulated spectrum is 2.4 dBm.



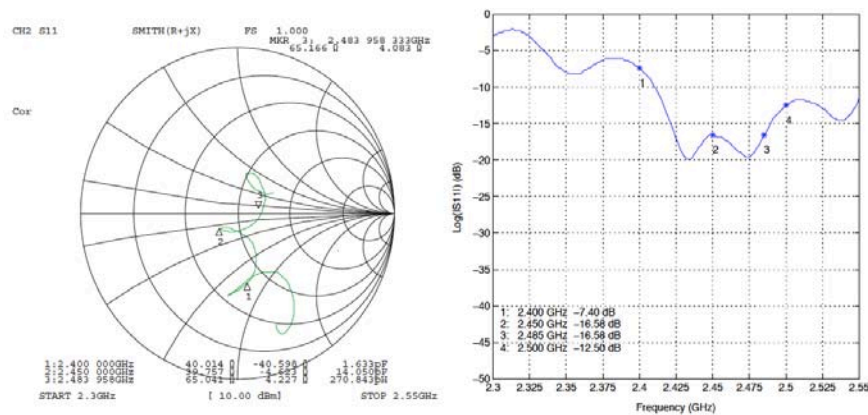
**Fig.2.6.** Measured RF output power over the modulated spectrum from the TMote Sky module



### 2.3.5 Antenna

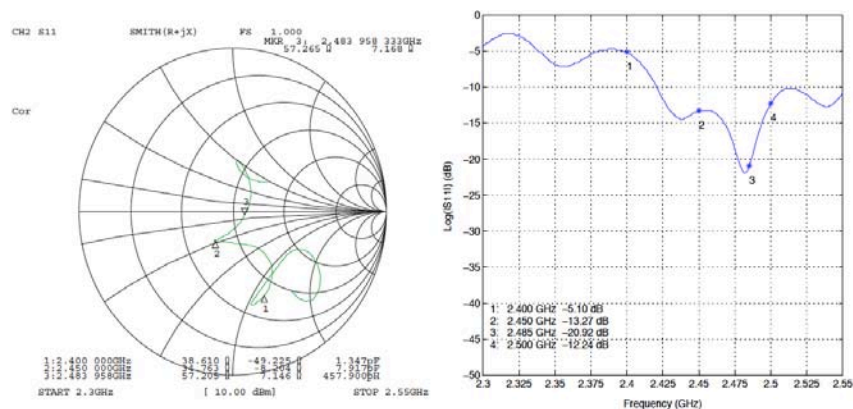
TMote Sky's internal antenna is an Inverted-F microstrip design protruding from the end of the board away from the battery pack. The Inverted-F antenna is a wire monopole where the top section is folded down to be parallel with the ground plane. Although not a perfect omnidirectional pattern, the antenna may attain 50-meter range indoors and upwards of 125-meter range outdoors. Measurements of the internal antenna's performance with and without a battery pack are shown in the next figures. Approximate radiation patterns for the Inverted-F antenna as provided by Chipcon AS are shown in Fig.2.9 and Fig.2.10.

#### Internal Antenna without Battery Pack



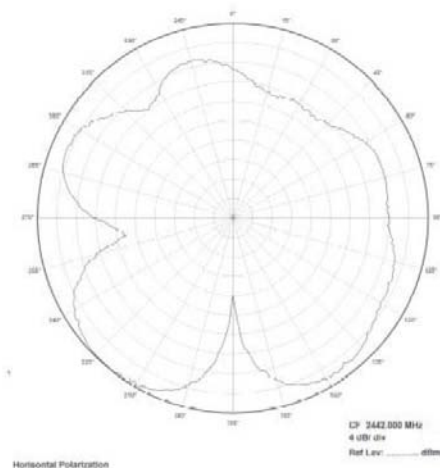
**Fig.2.7.** S11 measurements for the internal inverted-F antenna when no battery pack is present

#### Internal Antenna with Battery Pack

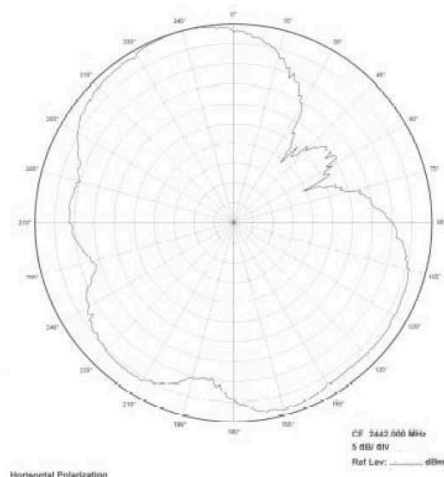


**Fig.2.8.** S11 measurements for the internal inverted-F antenna with battery pack underneath

### 2.3.5.1 Radiation Pattern



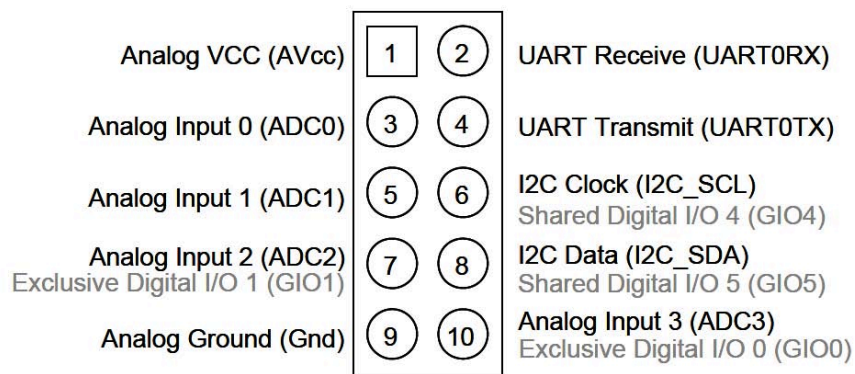
**Fig.2.9.** Radiated pattern of the Inverted-F antenna with horizontal mounting (from Chipcon AS)



**Fig.2.10.** Radiated pattern of the Inverted-F antenna with vertical mounting (from Chipcon AS)

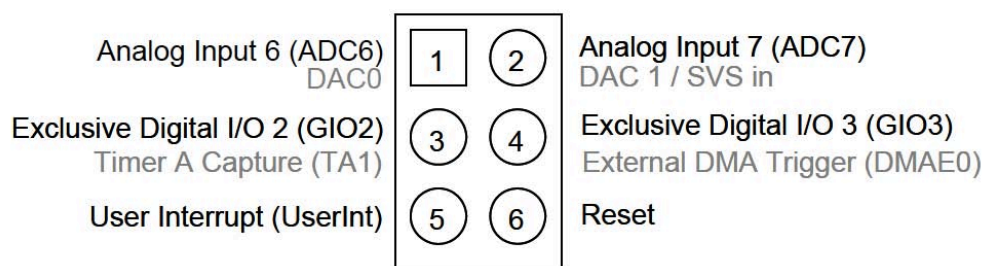
### 2.3.6 Expansion Connector

An important feature of TMote is that provides two expansion connectors and a pair of onboard jumpers that may configured so that additional devices (analog sensors, LCD displays, and digital peripherals) may be controlled by the TMote Sky module. On the far side of the board from the USB connector is a 10-pin IDC header and a 6-pin IDC header.



**Fig.2.11.** Functionality of the 10-pin expansion connector

The 10-pin connector (*Fig.2.11.*) has the same connections as TMote Sky and is the primary connector. It provides digital input and output signals as well as and analog inputs. Peripherals may be connected to the 10-pin connector using an IDC header, an IDC ribbon cable, or by designing a printed circuit board that solders directly on to the IDC header providing a robust connection to the module.



**Fig.2.12.** Functionality of the 6-pin expansion connector

An additional 6-pin header provides access to the exclusive features of TMote Sky. Two additional ADC inputs are provided that may be reconfigured by software to be two 12-bit DAC outputs. ADC7 may also act as the input to the supply voltage supervisor. The user interface elements (the reset and user buttons) are exported by the 6-pin header for use in external interfaces and packaging.

For more detail information about TMote Sky see *Annex E*.

## CHAPTER 3. THE HARDWARE IMPLEMENTATION

MAGCLOUD is designed for real world use. The idea of assembling a node in a protoboard or over a PCB not suitable for hard use was early discarded.

The TMote Sky nodes were originally designed for a final thesis in the Berkeley University and they are 100% open and free to use. At publication date, those nodes were commercially exploited by two enterprises but no project publicly available has tried to reproduce them. The MAGCLOUD project requires a large amount of ready-to-use nodes but buying them assembled is not an option considering its price is about four times the cost for the components alone.

This chapter covers the whole node hardware construction process. It will be treated as diary so, following the timeline, the reader can easily understand the order of the facts. In this particular stage, we found countless troubles but luckily all of them end up solved.

For further reference, days on the lab usually were about 5 hours long.

Before starting, we tried to identify which ones were the most critical components:

**MSP430F1611 Microcontroller:** When a commercial board is bought, it contains the factory firmware. Could a board assembled using a blank microcontroller be used by the TinyOS provided tools?

**FT232BL USB controller:** Similar to the MSP430, the FT232BL when in a node coming from the original assembler, contains a firmware inside but ours will not. Is this pre-programmed data necessary for proper working?

**DS2411 1-Wire Node ID:** Identifies the node in the network with a serial number. It is bought pre-programmed and cannot be user altered. Will the firmware allow working only the nodes that correspond to a range of identifiers?

### 3.1 Step 1: Finding the Right Components

To build a node, first step is checking if every component is easily available at a reasonable price. We started making a list containing all the parts to later check availability and price.

All components but one were available in the Digi-Key USA website. The MCP1700T power regulator was discontinued and we could not find it on the market at normal prices. After comparing datasheets, we found that MCP1701AT must be fully compatible with our discontinued model and also widely available.

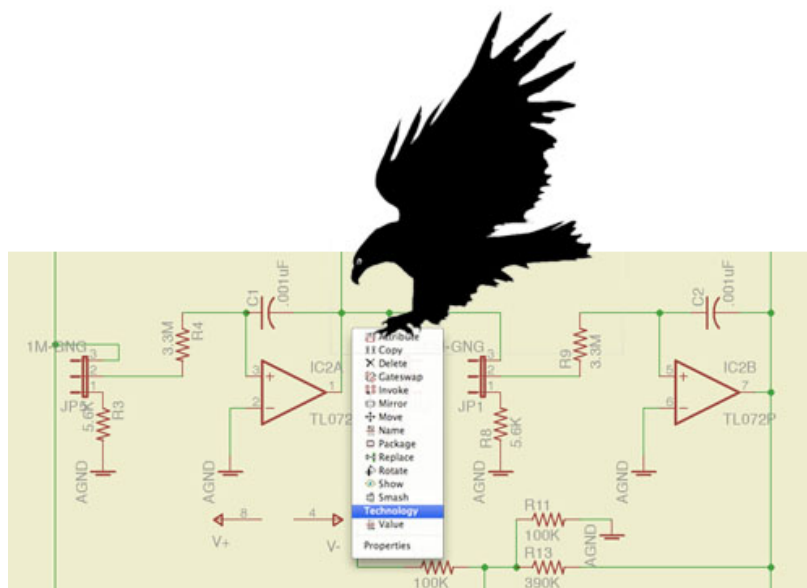
After some research, we found compatible models for all components but one within Farnell and Mouser. Those two providers have been widely used by our lab. Capacitors, resistors and crystals were easily found within tolerances. Inductors can be critical if they don't perfectly match so we decided to get the exact models. That latter forced us to get an inductor from Digi-Key USA, as that model was not found on any closer distributor.



**Fig.3.1.** Providers

At this point, we just possess the logical schematic of a node provided in the TMote Sky datasheet. In order to physically implement it, we need this logical circuit to be translated into a drawing that can be physically produced. We were going to draw this logical schematic into some software that let us physically shape it and then export it to a Gerber file. We knew that when working at 2,4GHZ, the shape of the tracks has to be carefully planned to avoid high frequency RF related problems but we had no training on RF circuit planning or drawing.

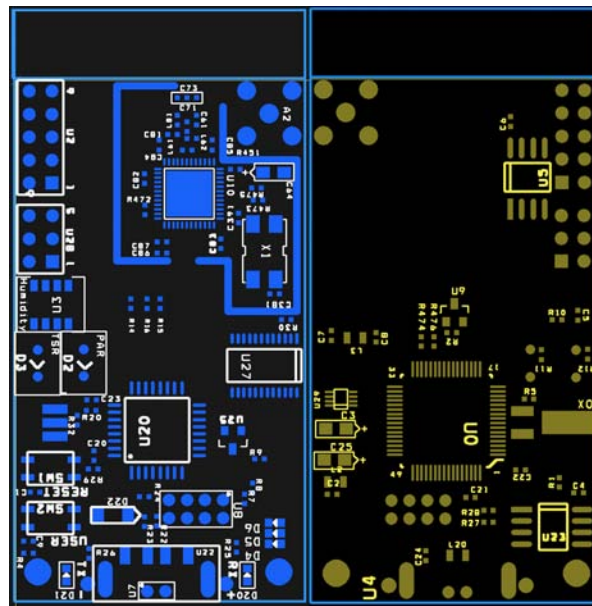
The software to draw the schematic was the free version of EAGLE. It is fully supported in the TMOTE SKY Berkeley website. They in theory provided all needed libraries and plugins to transport the original schematic to EAGLE.



**Fig.3.2.** EAGLE software

We installed the free version of EAGLE on the computer, downloaded the corresponding TMOTE SKY libraries and put them into the required place of the disk tree. They were not including all the required components but just some. We spent some days finding the remaining parts and finally placed them on the library folder. The workspace was ready and we could start drawing the board. After some time dealing with EAGLE, we found something very encouraging on the Berkeley website: The original Gerber of the TMOTE SKY board, completely free to use.

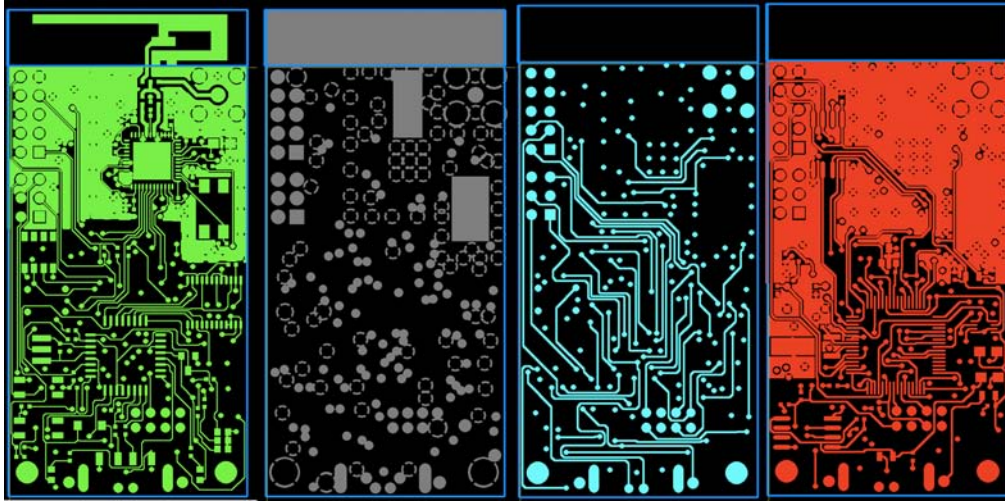
The latest version of the real thing; exactly what we needed.



**Fig.3.3.** Final schematic

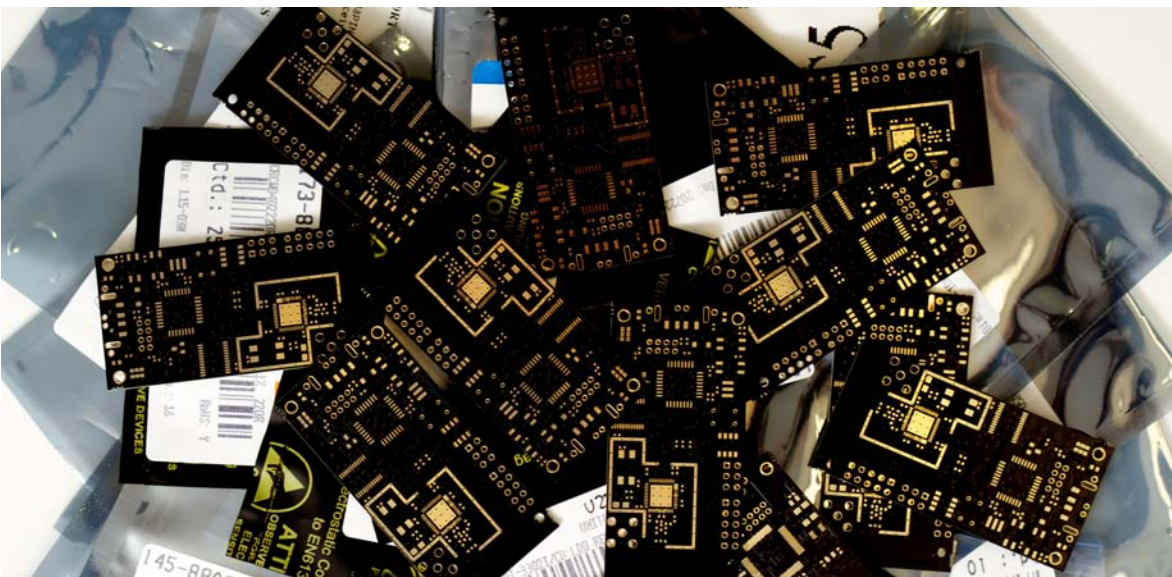
With the Gerber in our power, it was time to get the boards produced. We wanted fully rugged and environmental friendly boards at the lowest possible price. After deep research, we started contacting the enterprises that promised more interesting. Some of them were chosen because their closeness to the university, others for price or good references on the Internet. We have not understood yet what happened at that point: The lack of seriousness of all of them was incredible unexpected. The *Annex A* contains some of the conversations with the enterprises and its lecture is absolute recommended. Luckily, after a long way, we found a really professional enterprise that combined a very good price (almost the same as manufacturing in china), exquisite quality, closeness and good manner. This enterprise name is CISA2CISA and we absolutely recommend it whenever this kind of service is required in the future.





**Fig.3.4.** The PCB four layers

As it was so near, we decided to go there in person with the Gerber inside a pen drive. They helped adjusting and tuning the size of the tracks, holes and the rest of parameters to make sure the boards would perfectly match our needs. After some time waiting for the budget, the PCB's and the rest of components arrived to our lab. The costs for all pieces are in 5.3.

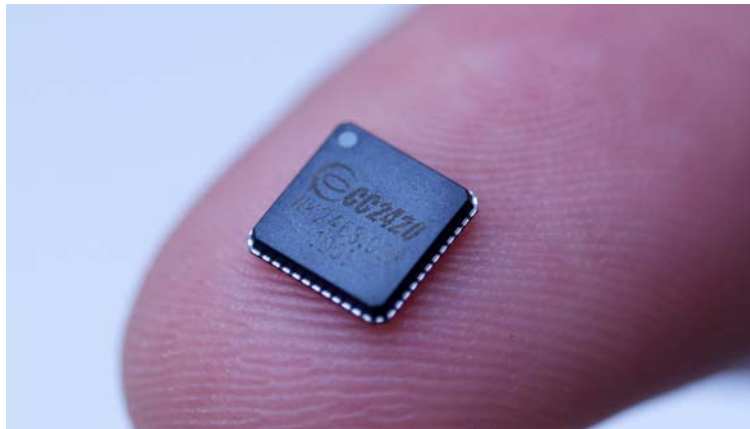


**Fig.3.5.** Components arrived

### 3.2 Step 2: The Board Assembly

This step took most of the time invested on the project. It started as something frustrating but ended the way we desired. It is important to notice that before this project we had no previous experience in SMD soldering.

Most of the components are extremely tiny. For example, the CC2420 IC is only manufactured in QLP48 package. This means 48 pin in a 5,3x5,3 mm IC. Each pin measures just 0,28mm wide and pins are incredible close between. Passive components as resistors, capacitors and inductors are 402 package (0,5mm wide). Until we tried, we could not imagine how something so small could be soldered by hand.



**Fig.3.6.** CC2420

Luckily, Francis López shown us some soldering tips and helped on the soldering of the first board. This first board was made using a large backlighted loupe and the soldering station with the tiniest tip available on the lab (about 1mm wide). The tin came in the usual roll format with a diameter of 0,5 mm.

It was almost impossible to distinguish the pins of some components using the loupe. The best method consisted on putting the board very close to the eyes, which produced incredible headache in a short time. The soldering iron tip was so large in comparison to the pins to solder that the tin gone into neighbour pins every time. The 402 components were so small that when touching them with the melt tin, they were attracted and stick into the tip of the soldering iron. When the hot tip repeatedly get in contact with the same area of the board, the pads and tracks unstick and break. That board was clearly not intended for soldering by hand. Anyway, as we already had the PCB and the components for 10 boards, we decided to keep trying.

Every time the tin gone to neighbour pin, we applied unsoldering track and tried again. To avoid the sticking to the tip of the soldering iron, we tried to make pressure with over it. It worked on resistors and capacitors, but when tried on led



and inductors they flattened completely because their heat sensitive plastic composition. To avoid those components end up thinned, we had to carefully fix them from the side using a spinning needle so didn't stick to the tin. This resulted on fingertips burning but the soft components ended on place. We had to discard two PCBS and start from scratch due to chipping and pad break caused by repeated hot tip pressure. After 3 weeks of headaches the board was finished. It looked not so bad from the loupe, so we connected it to the USB port. The result: Nothing detected by the computer.

We tried to revise that every pin was properly soldered but even using that big loupe we really could not be sure of it. We then tried with a kind of electronic optical amplifying tool. It used a loupe connected to a digital camera that had to be plugged to the computer via USB port. That worked even worse than the initial backlighted loupe. The metallic parts badly flared and the optical amplification of the lens and resolution of the camera were not enough.

Those pins were just too small. At this point we understood we needed the most extreme optical amplifying tool we can get: A double eye microscope.



**Fig.3.7.** Microscope

With a two eyes microscope, we stopped grasping at straws. It was like night and day. We started seeing what was really happening there. From the microscope, the board was a total mess of misaligned components, dirt from resin, scratches and chips. We found and repaired 18 crossed points. After that, we connected the node to the computer but as the first time, it did anything.

We supposed our problems were derived from the soldering and decided to make a new board, now using the microscope. Because the feedback was much

improved we were solving the problems at the precise moment they were showing up. Anyway, the precision of using a tin that was enormous in comparison to the pins was close to zero. With the help of a needle, unsoldering track and lots of patience, we finished our second board in about two weeks. When connected to the PC, the led blinked once, but sadly no device was detected connected to the USB port.

It was clear that at that point, the limiting factor was the diameter of the tin and the tip of the soldering iron. Even replacing both for smaller ones, they would still be too big. We needed to use a different approach. We were told about a new kind of tin that comes in a gel format, inside a syringe. This way, depending on the size of the attached needle, we could modify the diameter. When heat is provided, this kind of gel responds the same way the usual tin does. We get one of those to try.



**Fig.3.8.** Tin gel syringe

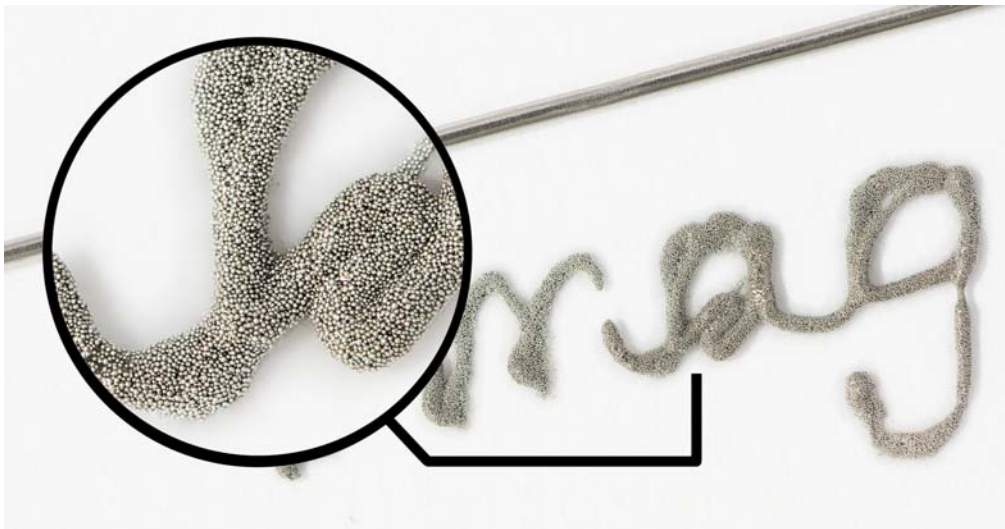
We also get a new set of tips with ranging diameters from 0,16 mm to 1,8 mm. We had two kinds of tips for the syringe: Plastic conic tips and metallic needles.



**Fig.3.9.** Tips and needles for the syringe

As we needed the narrowest possible diameter, we mounted the smallest needle into the syringe and vigorously pressed it. The more force we were applying, nothing was flowing out of the needle. Even increasing its size, it was clear that those needles must be used in another way: the only needle that allowed gel to come out was the largest one that, by the way, was even larger than the roll tin. When using the conic plastic tips, the minimum diameter that allowed the gel to come out was smaller so we chose it instead of a needle. Anyway, the required pressure to apply for a not so small as we desired jet was large.

To test the gel, we made some trials on a blank piece of circuit board. What we first noticed was that the required high pressure on the syringe was hurting our thumbs and causing an imprecise tin placement. The gel, when seen through the microscope revealed its nature: It is composed of very small metallic spheres surrounded by some transparent fluid. Because of this the texture was not perfectly fluid but kind of sandy. We placed a small gel line, touched it with the tip of the soldering iron and it instantly melt, getting the same texture the tin we were used to. This looked very promising.



**Fig.3.10.** Roll VS gel tin

A new PCB was placed under the microscope and we started painting with the gel. One great thing about the paste is that we could easily place the component before melting, and once perfectly aligned, touch the gel with the tip. It was far more foreseeable and clean than using the roll of tin. Gel placement on the largest pins was relatively easy but on smaller ones, it spilled everywhere. When making so high force with the thumb it was very hard not to shake and the diameter was still large. It was also very annoying that the gel did not come out from the tip as a perfect line but was twisting from one side to another all the time. The gel also did not stick well on the board but continued stuck to the syringe.

For applying paste on the tiniest IC's, we tried a new trick. As we had various units of every piece, we applied a line of gel from under the pins and "combed" using

another equal IC. The idea sounded nice but in the practice was not so good. We finally get all IC's on place but two MSP430F1611 ended on the bin because pin damage.

The placing of some IC's on its corresponding pre-tinned zone of the board is a critical step. After picking the component with a reversed tweezers, the IC has to be carefully approached to the PCB and perfectly placed it over its definitive home. For example, for the CC2420 radio, if the IC just touches the tin 0,3mm over or under the optimal position in any of its four sides, all the gel has to be cleared and the placing and shaping operation of new gel has to start again. Specifically for this IC, this painting and shaping operation may take over 1 hour and bad misalignment happens easily. It is the single operation one has to take most care of the whole board construction.



**Fig.3.11.** Picking the IC with the reversed tweezers

The most disgusting problem related to this method was that thumb fingerprint adopted the top syringe squared texture and ended with pain. One nice thing is more than one components can be placed before applying heat and then one can melt them one after the other. This speeded up the process even more.

It is very important that every journey, before leaving the lab, every tip or needle that has been in contact with the gel gets properly cleaned and the syringe itself gets sealed using a watertight tip. The clean is done by removing the most of the residue of the base using some pointed tool and then filling an empty syringe with alcohol, mounting the tip or needle on it and forcing that alcohol to pass through at high pressure.

The third node was ready in 7 days and looked much nicer under the microscope. When we plugged to the USB, the led blinked and computer detected it as a USB to Serial converter. When the "*motelist*" command (See 4.1.2.1) was executed, the node also appeared in the list with the name "USB<---> Serial Converter". Then, to see if it was working properly, we tried to install the "*Blink*" test. This test makes some leds blink following a certain sequence. It started the programming but after a while, we get stuck at the following error.

```

root@ubuntu: /opt/tinyos-2.1.1/apps/Blink
File Edit View Terminal Help
root@ubuntu:/opt/tinyos-2.1.1/apps/Blink# make tmote install
mkdir -p build/telosb
compiling BlinkAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -O -mdisable-hwmul -fnesc-separator=_ -Wall -
Wshadow -Wnesc-all -target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFI
NED_TOS_AM_GROUP=0x22 -DIDENT_APPNAME=\"BlinkAppC\" -DIDENT_USERNAME=\"root\" -D
IDENT_HOSTNAME=\"ubuntu\" -DIDENT_USERHASH=0xa3473ba6L -DIDENT_TIMESTAMP=0x4dc2d
e72L -DIDENT_UIDHASH=0xaa430b61L BlinkAppC.nc -lm
compiled BlinkAppC to build/telosb/main.exe
2648 bytes in ROM
54 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
writing TOS image
cp build/telosb/main.ihex build/telosb/main.ihex.out
found mote on /dev/ttyUSB0 (using bsl,auto)
installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out
MSP430 Bootstrap Loader Version: 1.39-telos-8
Mass Erase...

An error occurred:
Timeout
make: *** [program] Error 1
root@ubuntu:/opt/tinyos-2.1.1/apps/Blink#

```

**Fig.3.12. ERROR 1**

We revised the board again but no crossed lines were detected. We tried on three different computers but got the same result. The original commercial board was working as expected. We supposed the error was caused once more by a defective assembly and tried to improve our technique.

Francis López told us about product that could become our ally. A low price machine that precisely times the flow coming from an air compressor, giving an air pressure timed shot every time the user pushes a pedal. We were provided with one of those. The manufacturer sent it directly from China.



**Fig.3.13. Pedal timer**



We connected a compressor to the pedal timer and an airflow adaptor for our tin syringe. The adaptor was included for free with the timing machine. The used compressor was the model Herkules WALKAIR.



**Fig.3.14.** The air compressor

Our pedal timer was not specific for gel tin pushing. It can be used on different industries and no configuration data for our application was available. Because of that, we started studying how the optimized parameters would look for our tin shot.

After setting the pressure to the maximum possible level and the shot timer on 0,5 Seconds, the tin was flowing out of the syringe on every pedal push. We tried the smallest diameter needle but the compressor was not enough powerful to push the tin out. The minimum needle for the gel to flow out at maximum pressure was the intermediate blue one. The smallest plastic cone also worked. The plastic cone gel output was faster and thinner but much more unstable. It was spinning and moving around all the time and that already caused trouble on the construction of the last node. The shot provided by the minimum usable diameter needle was not thin in comparison to the pins but was placed in a much more easy and convenient way. After long trial we determined the best combination for our delivery needs was consisting on give a short shot at the maximum pressure. When more delivery is needed, the operator pushes the pedal more times.



**Fig.3.15.** Time and pressure

We also started using a new tool, a very pointed tweezers. This proved to be very useful in slicing and shaping the gel shots. Using this technique and the new machine, we get a finished board on five days. The look of the node was phenomenal, noticeably improved from the last one. Thumbs were nice and the whole experience was smooth.



**Fig.3.16.** Pointed tweezers

On the computer, we got the same result as with the previous node: Leds blinking once and USB to Serial detected but *ERROR 1* when trying to program. This *ERROR 1* is a generic error. No found guide gave us trustable information about where the problem was coming. Again soldering problems? At this point we started suspecting there was something else causing node malfunction.

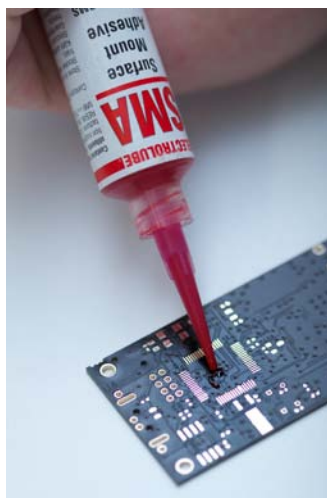
From the construction point of view, the tin delivery and placing was fine enough. Also was the cold tin slicing and shaping when we get used. It required patience to learn the proper technique and enhance the ability but at the end it feels like kind of painting. We were not happy with the IC placing method as often the filling and placing tin operation had to be done twice or even three times. Also, melting with the hot tip one by one all the components was very time consuming. If a massive implementation was going to cover the campus, the boards needed to be constructed in much less time.

To speed up the node building process, we eliminated completely one of the stages using a special oven. This kind of instrumentation is intended for PCB prototyping and promised perfect automatic soldering of our tin in gel. This kind of oven is highly specialized lab material and thus expensive. It works by melting the tin of the whole board at one time using a special heat process that gives perfect appearance and long lasting professional results. Just what we needed: much cleaner and resistant nodes that are in line of our idea of real world usable material. After some time, our lab installed one of those ovens for us: the model ProtoFlow E.



**Fig.3.17.** Prototyping oven

To prepare a board using the oven, we had to proceed slightly different. Our board has components on both sides. Because of this, one side has to be prepared and soldered first, and then the mote turned and done the same on the other side. All the tin on the board melts at the same time, at a certain point of the heating process. There is no problem with the first side but at the moment the second side is melting, the first one also melts again. The components of the first side that were already soldered on place will fall inside the oven when the tin is melt because they are now facing downwards. To avoid that, the components of the first side must be glued one by one using a thermal glue that turns into a solid rock when heat applied, sticking what it has around.



**Fig.3.18.** Thermal glue syringe



We assembled a new PCB to see how the oven was doing. Also, this could clear our doubts about bad soldering. Three boards built using three different procedures that fail at the same point because of bad soldering does not sound probable. The oven takes about 8 minutes to complete the process. In order to avoid overheat of the components, we wanted to cook the board only two times, one for side. That was meaning that every side had to be finished on a single working day.

This time, we noticed the gel tin was not as fluid as days ago. Probably it was caused because not following the advice of the manufacturer to leave the paste into a cold environment when not in use. Since then, to avoid further hardening, we stored in the refrigerator when leaving the lab. It is important to point that only the largest components of the first side were glued. The tin itself retained the smallest ones on the board. We finally completed the first side in 8 stressful continue hours. The second side took about 7. We used the default-heating program for small boards included on the oven and worked perfect.

The oven soldered board had some crossed points and some misalignments. Some of the components seemed to move inside the oven and needed manual realigning. Other than that, the board looked absolute wonderful. Really, it was hard to believe we did ourselves that so nice board. When connected to the computer, it happened again: *ERROR 1* when trying to program.

Seeing that the error was not happening on commercial boards, we get some commercial nodes from an eBay auction. We bough them using money from our pockets but as price was good and we were provided with just two nodes we though it was a good idea. The nodes the UPC provided were already used to test and build the software so we were not in position to disassemble them to play.

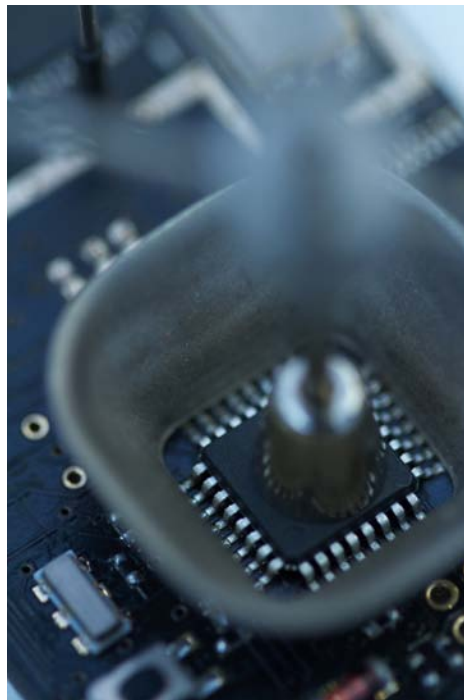
With some non-working boards and some fully functional commercial nodes on our power, diagnose could start. The first problematic candidates we discarded were the MCP1701AT Voltage regulator and the DS2411 1-Wire ID. It was an easy operation as they just have three pins. Using two soldering irons at a time, we switched them on the boards. The result was that our node and the commercial node behaviour did not change a single bit. Our board stopped at *ERROR 1* and the commercial successfully programmed. Both components were discarded.

The next step was to see if the error was caused by our CC2420 microcontroller, lacking the original firmware. Using the original programming software provided from Texas Instruments and a JTAG interface, we tried to extract the firmware image. The after losing some time trying to understand how to proceed, we finally understood that this operation couldn't be done by using that software. One can program using its own firmware image or live debug it with what it contains, but can't extract it and save it on a file.



**Fig.3.19.** JTAG interface

There was an option that for sure would dissipate out our doubts: As we did with the three pin components, directly replace our factory blank IC for the commercial one. It could not be done by using the oven because the boards we had did not had glue on the opposite side. Soldering this kind of component by hand was problematic in the past, so we preferred to make a new board directly using the programmed commercial IC instead of the blank one. To extract the IC from the commercial board, we used a station designed for that application.



**Fig.3.20.** The unsoldering process

To operate the unsoldering station, some steps have to be followed:

- 1: Placing a metallic protector over the IC. This metallic piece minimizes heat spill over the rest of the board. The chosen size has to be the minimum it fits but without touching the pins.
- 2: Attach the tube of the correct size suction component to the suction hole of the station.
- 3: Clean the surface of the IC and mount the tripod in a way it stays completely flat over the board.
- 4: Start pumping pushing the corresponding button. When it is sucking, carefully attach the suction pipe to the IC. If the tripod is not completely flat mounted (It may be hard depending on the zone of the board the IC resides) the suction pipe will not attach well.
- 5: On the pedal hot air blower controls, configure the adequate temperature, (the hottest possible while not damaging the IC) press the blower pedal and vigorously start heating every side of the microchip in a continuous rotating operation: Side 1,2,3,4,1,2,3,4,1,2,3,4... On every rotation one has to fast avoid the tripod legs. If the soft attached legs or pump pipe are touched with the blower tip, the suction pipe falls and one has to start back from step.
- 6: At the moment the tin is simultaneously melted on all pins, the spring on the suction pipe makes the IC jump from the board. Let the machine blow some cold air and turn off.
- 7: With a tin sucking stripe and a soldering iron, carefully remove the remaining tin from the IC pins.

It takes some time to get used to the rotating movement. If excessive time or temperature is applied, the IC is damaged. After lots of time applying heat on that circular way without any result, we stopped the blower to avoid breaking the component. After some thinking, we noticed: Maybe the IC was glued to the board? Could we unstick it while circular heat blowing?

We started blowing again. While one of us was circular hot blowing, the other started making pressure up with a screwdriver on the corner of the IC. After some a while, every time applying more force, we heard a 'clack' noise and the IC jumped out of the board. Yes, it was hardly glued with a big rock of red material. Because of this operation the commercial board got damaged. The glue caused chipping the PCB and breaking some tracks under the IC.

Once clean, we carefully aligned all somehow twisted pins (they are extremely delicate) and proceed making a new board with the oven. It took about 7 hours for

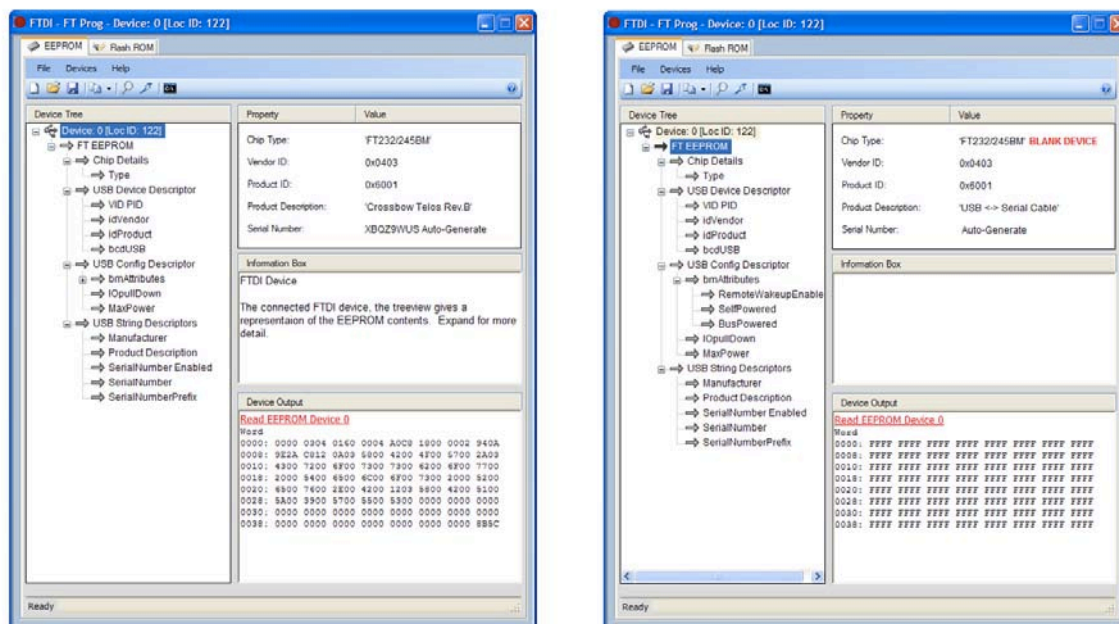
the first side and 6 for the second. When it was finished, we connected it to the USB and tried programming: *ERROR 1* again. The microcontroller was discarded.

The next component to test was the FTDI USB controller and its associated memory. To read its content, we used the FT-PROG software from the IC manufacturer.



**Fig.3.21.** FTDI Software

We connected a commercial board and read the contents of the controller and external memory:



**Fig.3.22.** Original and blank TelosB USB Controller content

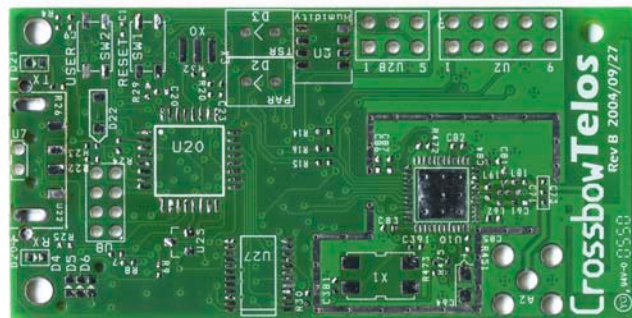
We saved the commercial IC content to a file on the disk and then read the content of one of our blank boards. They had clearly different data inside, including parameters as the USB device name the computer detects for the genuine node.

We programmed our IC chip using the content of the commercial node. After that, we checked that the data was really there, reading it again. As the IC was now properly programmed, we plugged it into the computer we were going to check if the original problem was solved. When the PC detected it, the new “*Crossbow Telos Revision B*” name was shown instead of “*Serial <---> USB converter*”. When tried to programming, *ERROR 1* again.

We get out all the active components remaining on the partly disassembled board and also the original IC that was moved to our most new non-working node. On this latter board, the face opposite to the commercial microcontroller was made first, thus the microcontroller was not glued. The unsoldering operation was clearly easier as we were getting more used to it. Anyway, the removing of components with glue on its base caused problems again. We broke one IC and had to get it from a different commercial board.

With all the active components unsoldered, we started building a new node. It used all the commercial IC's but all our brand new passive components. Sadly, the new board returned the same *ERROR 1*, unable to program again. It was a fault caused by the passive components or maybe a defective PCB? We got the schematic from the original Berkeley University website but, who knows?

The circle was getting tighter. We fully disassembled a commercial board. Every single component, one by one by using the soldering iron and indexed them over a sheet of paper. It would be a passive component or the PCB itself?



**Fig.3.23.** Commercial PCB with the removed components.

We mounted a new board using blank, new components on the commercial PCB. Result: *ERROR 1*. We mounted one of our PCB's with all the commercial components. Result: Our board was programming perfectly. Clearly, a passive component was the guilty of all our headaches.

This time it had to work. There was nothing else to discard. We mounted a new board with all the passive components and our active blank components. IT DID NOT WORK. *ERROR 1*! At this point we could not be more frustrated. What was that hell *ERROR 1*? At first we did think on leaving the node construction but after calming out we understood we were too far to abandon.

There was something we avoiding all the time because it would be incredible time consuming, the 100% definitive diagnostic method. Replace one by one, using the soldering iron every single piece on the board until it started working properly. At some component it had to let us program. The following procedure was repeated until the board stopped returning *ERROR 1*:

- 1: Unsolder piece X on the commercial board.
- 2: Unsolder piece X on our board.
- 3: Solder piece X from the commercial board on our non-working board.
- 4: Connect our board to the USB and check if it is can program.
- 5: If it is not working, go back to step 1 and proceed with the next piece.

It was so boring but we were encouraged because it was the last stage, the end. This time it could not fail. After an eternity, the board programmed. What we did not like is that the component that did bring it to life was the MSP430 Microcontroller. We already had done a board before using a commercial microcontroller and at that time, it did not work. We also already did a board using all active components from a commercial node, and also did not work! It really made no sense.

Anyway, we made again that same board, using all our components excepting the microcontroller. With our latest technique, finishing a board using the oven was taking just about 4 hours by side. After two days we had another *ERROR 1* node for collecting. After all that time, we had nothing. We were hating that *ERROR 1* screen.

We decided to try the last thing and if it was not working, abandon. No more trials after that. It is hard to imagine how furiously frustrated we were at that time. We repeated one by one once more, but this time using the non-working with the commercial IC board instead of a completely blank one.

After another eternity of boring and repetitive component-by-component replacing, the board finally programmed. This time, it started working when we replaced the microcontroller's 32.768 kHz clock!!

This time, somehow more encouraged, we started making a node that had all our components except the microcontroller and the 32.768 kHz clock, that were coming from a commercial board. After two days, we got the final board finished. We connected it and.... It programmed! YES! We found it!

At that moment we saw heaven. Our *ERROR 1* was coming from the combination of two components: The microcontroller and its clock. After some research, we found the exact model the original board was made with CMR200T32 and bough some units. We replaced it on a non-working board along with a commercial microcontroller and it nicely programmed.



**Fig.3.24.** CMR200T32

The problem was that to make a working node, we had to use microcontroller coming from a commercial board. The original firmware image was not available on the Internet and we had no success getting it from a commercial IC by using the manufacturer provided tools. We needed some alternative that let us extract this firmware image from a microcontroller that was containing it.

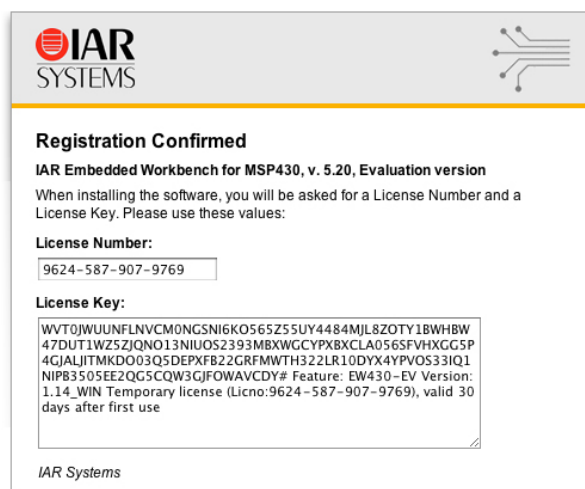
After researching, we found exactly what we needed. A commercial program specific for the MSP430 claiming that for the cases the microcontroller's lock fuse is not enabled, the whole image can be directly extracted and written on a blank IC. It uses special hardware for making the operation but there was a new version supposed to be compatible with the original Texas Instruments programmer that was already available in our lab.



**Fig.3.25.** MSP430 in the programing socket

Even better, there was a free functional demo version. Preparing the software environment for the original Texas Instruments programmer required special care because the needed driver was not available in the lab on the Internet. After some research, we were able to extract a compatible driver from another commercial software, the version for the MSP430 of IAR WORKBENCH.



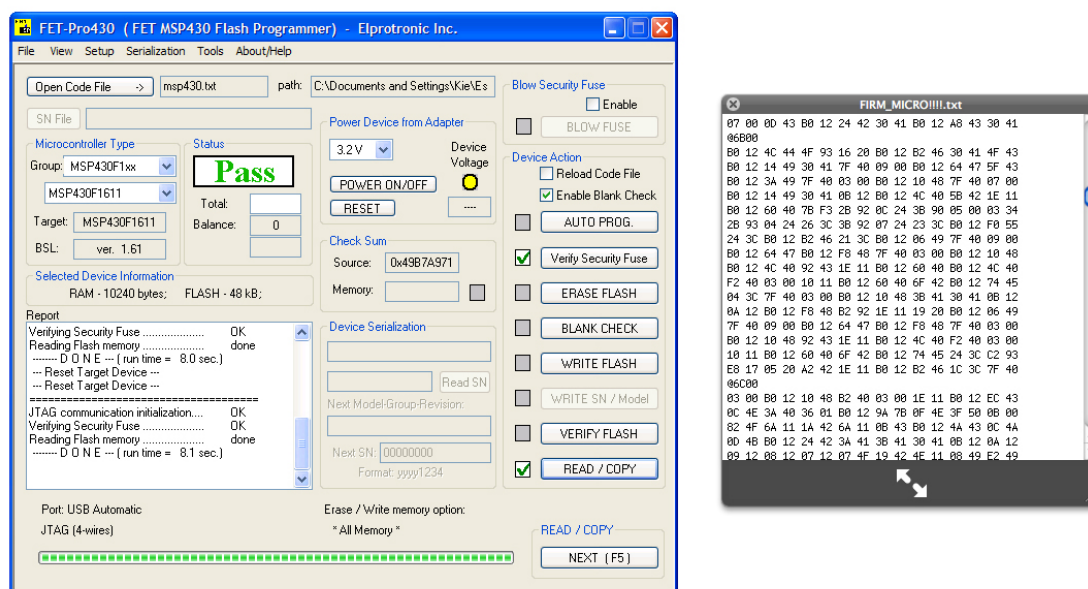


**Fig.3.26. IAR WORKBENCH**

To obtain the driver, we installed the demo version of this software and finally found it on the folder

*"C:\Program Files\IAR Systems\Embedded Workbench5.4\430\drivers\TIUSBFET\WinXP"*

Then installed the driver from the windows device manager. We were ready to get the firmware image from inside the original MSP430 using the now fully functional *ELProtronic FETPRO430* software.



**Fig.3.27. Firmware reading using ELPROTRONIC FET430 Software**



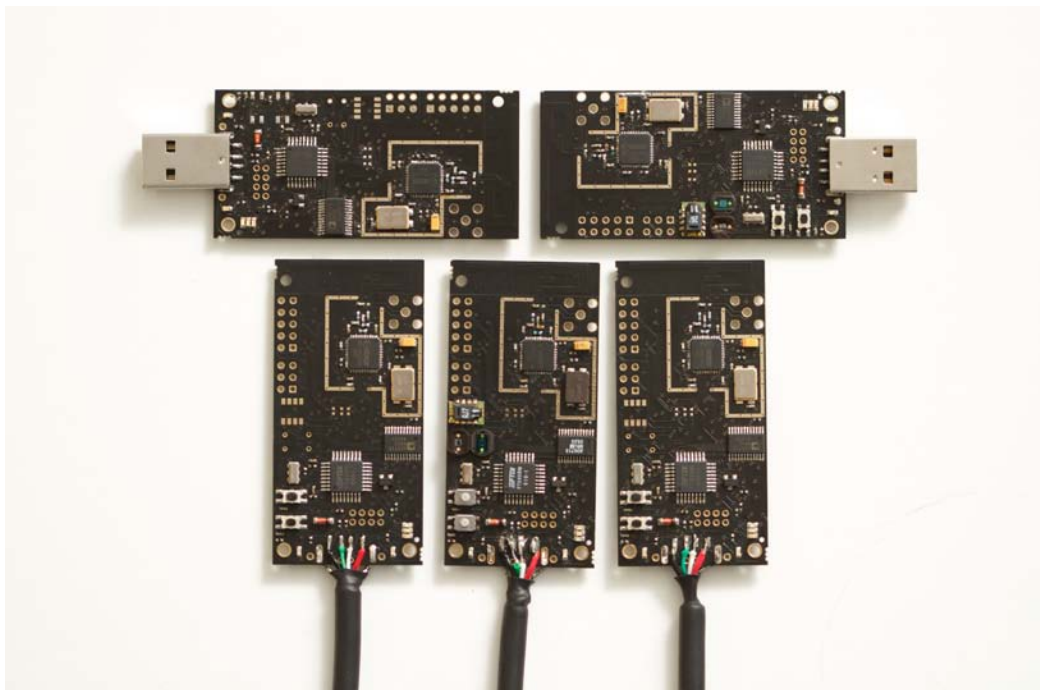
Once the Firmware was in our power, we written it into a blank IC.



**Fig.3.28.** Programed MSP430

We built a node using all commercial components, including the originally blank, now programmed MSP430. Combining that moment's excitement and previous training, we finished our last node in 7 hours and 30 minutes. A trick that helped speeding up the process consisted on making the gel shots placing the needle as perpendicular as possible to the board. This made the gel shot stick to the board properly and not to the needle. When plugged to the USB port and sent the programming command, it worked PERFECT!

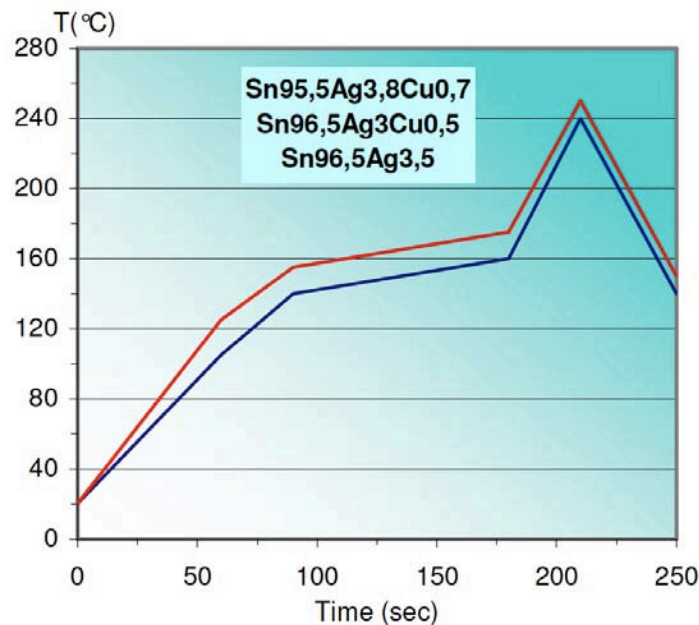
After that, we repaired the remaining boards that survived from previous quartering by replacing the MSP430F1611 and its clock.



**Fig.3.29.** Our finished MAGmotes

We were so happy with the boards but one of our requirements for MAGCLOUD wasn't yet accomplished. Our nodes had to be environmental friendly, and they were not because of the tin. Our gel tin was really tin-lead. This kind of pure tin was also used on the commercial boards and it caused hard trouble when trying to unsolder even with the station at its maximum power. The lead is used to reduce the melting point, so this pure tin requires much higher temperature to fully melt.

We made a board using a new lead-free tin gel syringe but the oven using its default program was not enough powerful to melt it. The datasheet of the lead-free tin gel included a graphic with the optimal heat curve the gel has to receive to give long-term professional results. Checking the default oven curve it's clear that is so far from this lead-free tin heat curve. Some days later, a new custom program with the required curve was introduced and the lead-free tin finally melt the way the old tin used to.



**Fig.3.30.** Lead-free gel tin heating curve

The last step, and just to make sure the boards are as close as possible to the commercial ones, we programmed all the USB FT232BL IC's with the original image we previously extracted in Fig.3.22 but customizing the nodes name to "MAGmote".

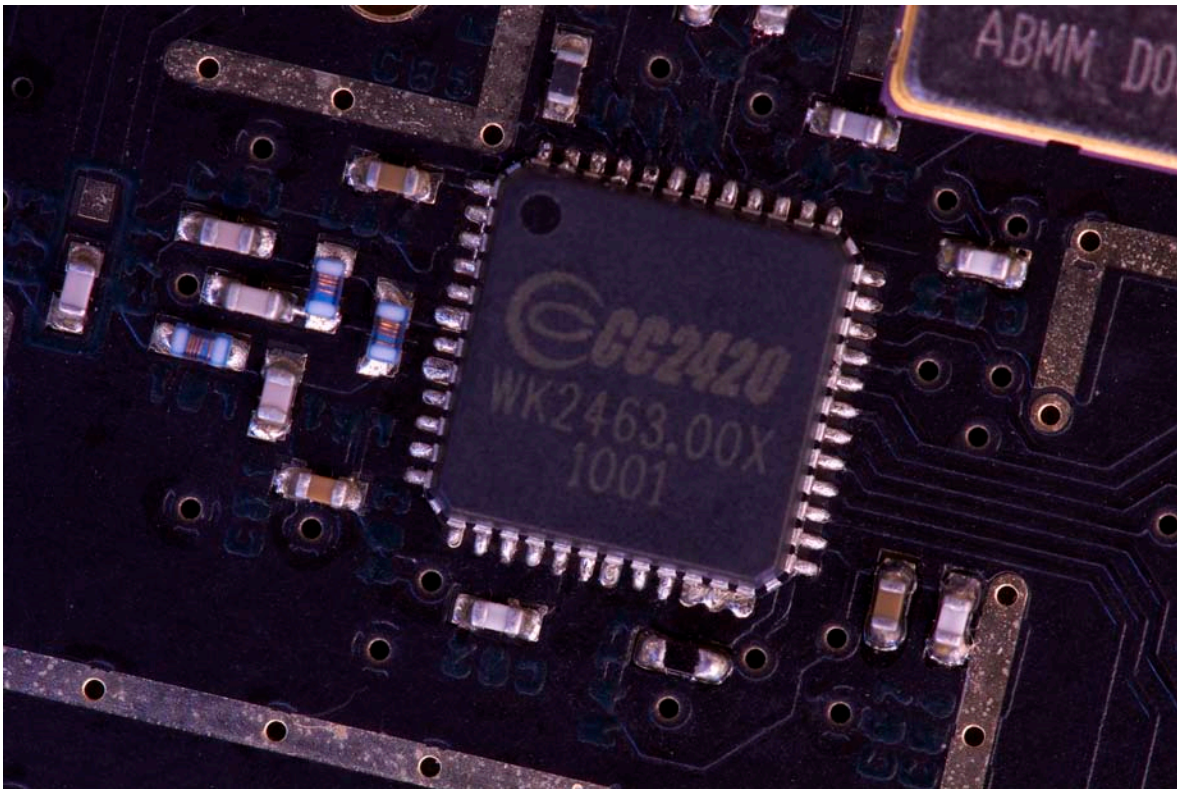


**Fig.3.31.** First time plugged final mote

### 3.3 STEP-BY-STEP MAGCLOUD NODE CONSTRUCTION

- 1- Program the microcontroller using the *EIProtronic* software with the corresponding image contained on the DVD.
- 2- Mount the 0,41 mm diameter needle on the syringe and attach the pressure timer adapter at its back.
- 3- Configure the timer at 15 full seconds and press the pedal once. At the moment the gel starts flowing out of the needle, press the stop button. If the timer expires but no gel has flown out, keep pressing the pedal until it does.
- 4- Shorten the timer to about 1 second. Using the microscope, keep reducing the time until you get the desired size shot. This parameter tends to had to be lowered after a while to obtain the initial shot size.
- 5- Place the PCB under the microscope and start covering the contacts with gel on the whole side. To shape thinner lines, paint using a very pointed tweezers or a needle.
- 6- Using the thermal glue syringe, place a convenient-sized dot on the places the largest components will reside to avoid them falling inside the oven.
- 7- Using a pointed tweezers, carefully place every component where it will end soldered. If a component is placed misaligned, the gel shape will get bad. Try to redistribute using the pointed tweezers or clean the gel on the area and repaint, whatever seems faster.
- 8- Turn on the oven and start the proper program. The default setting is useful for tin-lead gel, but if tin-free wants to be used a custom profile has to be chosen. If this lead-free custom program it is not stored in the memory of the oven, it will have to be introduced manually using the heat curves provided from the manufacturer.
- 9- When the oven requests it, open the lid and adjust the size for your PCB. Then, place the largest size of the board between the rails and fix very tight to avoid vibration. Do not forget to wear protective gloves during this operation.
- 10- Close the door of the oven and wait until the process ends. You will be requested by the oven to open the lid and let the PCB cold.

- 11-** Remove the PCB (do not forget the protective gloves), close the lid and turn off the oven.
- 12-** Go back to STEP4 and proceed with the remaining face of the board. STEP5 is not required this time.
- 13-** Program the FTDI USB Controller using *FT Prog* software and the corresponding image provided on the DVD.
- 14-** When finished soldering, clean the needle, seal the syringe and store it on the refrigerator.



**Fig.3.32.** Detail of our oven-soldered board

## CHAPTER 4. THE SOFTWARE IMPLEMENTATION

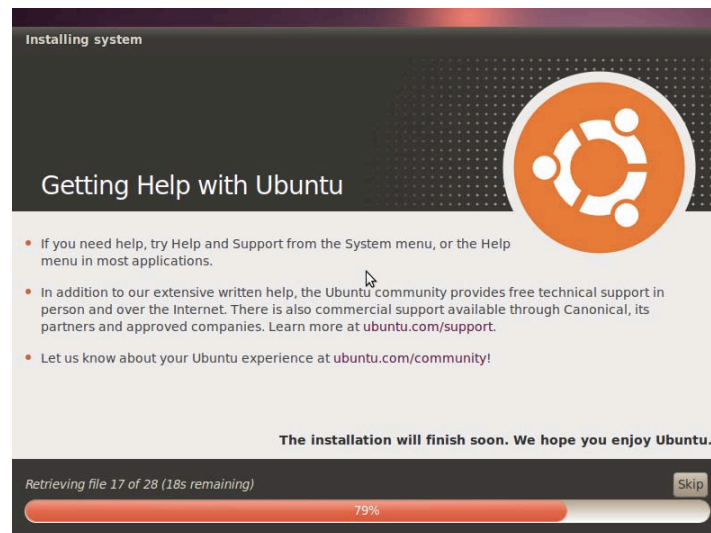
### 4.1 Ubuntu

We want the software of our project to be 100% free and open, with the source code publicly available. That is the reason we decided to work with Linux. The first step is choosing between all the free distributions the one that better fits our needs.

Ubuntu is the Linux distribution that better supports TinyOS. It is a computer operating system originally based on Debian GNU/Linux, distributed as free and open source. It provides a stable platform for the average user, strongly focusing on usability and seamless installation.

Ubuntu is composed of multiple packages distributed under either free software or an open source license. Different Ubuntu-based operative systems have developed: Kubuntu and Xubuntu.

XunbunTOS is version of Xubuntu that includes a pre-built TinyOS environment for providing an easy and fast starting point. At first we were thinking on using it for the project but the option was refused when we noticed XubunTOS wasn't in development anymore. The version of TinyOS it was using was obsolete. We preferred to manually install TinyOS 2.1.1 directly from the sources into a regular Ubuntu image, concretely into Ubuntu 10.04.2 LTS. (The latest release at publication date).



**Fig.4.1.** Ubuntu installation



### 4.1.1 Installing TinyOS

Once Ubuntu 10.04 is correctly installed, it is necessary to add different packages and tools. Some libraries are required to compile and install TinyOS and its dependencies.

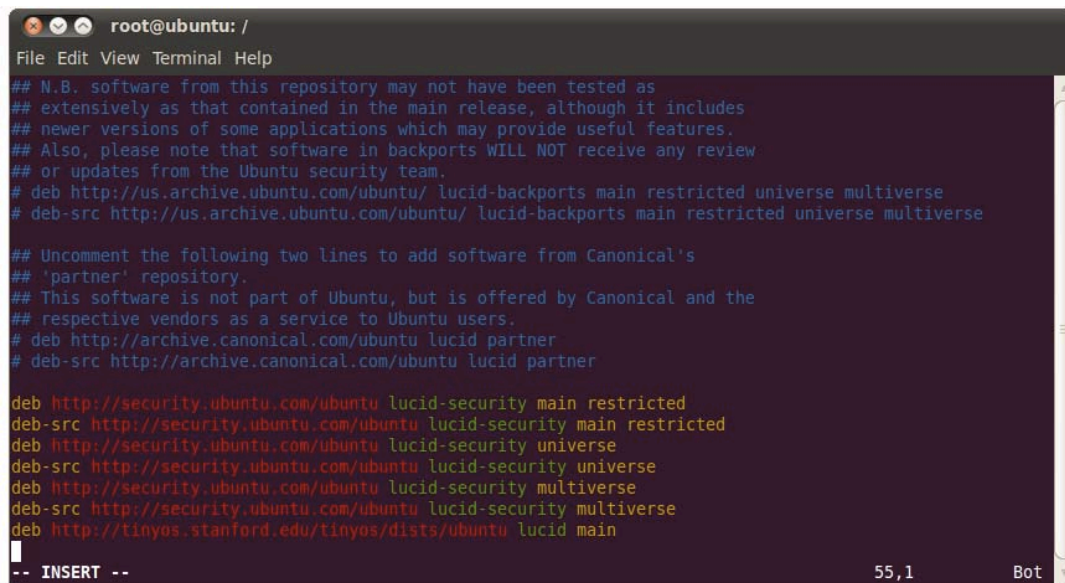
First, the Linux build essentials and some other basic components are installed.

```
➤ sudo apt-get install m4 autoconf libtool gawk bzip2 bison  
flex gettext textinfo zlib1g-dev  
➤ sudo apt-get install build-essential  
➤ sudo apt-get install libmpc-dev libmpfr-dev
```

Most of the following commands must be executed as super.

```
➤ sudo su
```

The next step is to add the TinyOS apt sources to `/etc/apt/sources.list`.



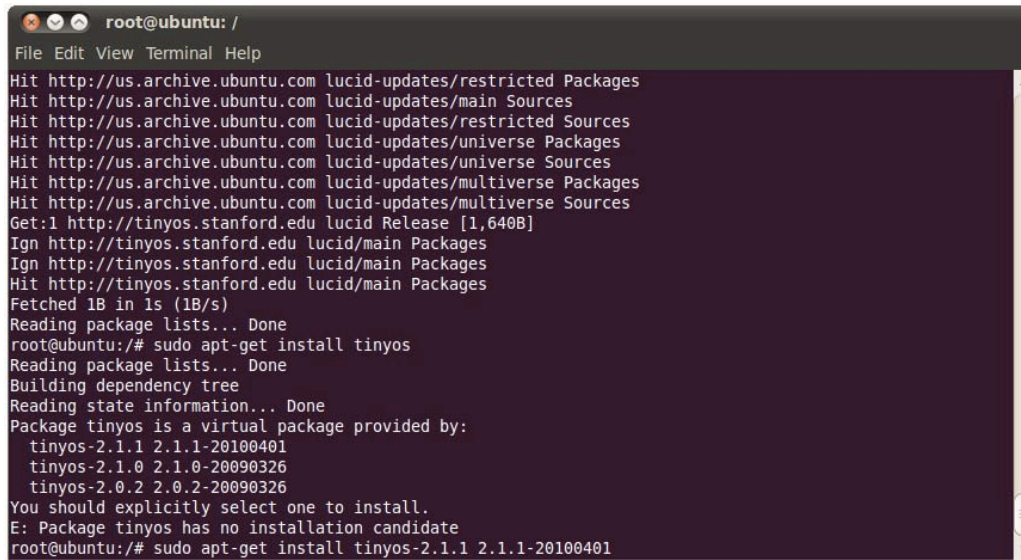
```
root@ubuntu: /  
File Edit View Terminal Help  
## N.B. software from this repository may not have been tested as  
## extensively as that contained in the main release, although it includes  
## newer versions of some applications which may provide useful features.  
## Also, please note that software in backports WILL NOT receive any review  
## or updates from the Ubuntu security team.  
# deb http://us.archive.ubuntu.com/ubuntu/ lucid-backports main restricted universe multiverse  
# deb-src http://us.archive.ubuntu.com/ubuntu/ lucid-backports main restricted universe multiverse  
  
## Uncomment the following two lines to add software from Canonical's  
## 'partner' repository.  
## This software is not part of Ubuntu, but is offered by Canonical and the  
## respective vendors as a service to Ubuntu users.  
# deb http://archive.canonical.com/ubuntu lucid partner  
# deb-src http://archive.canonical.com/ubuntu lucid partner  
  
deb http://security.ubuntu.com/ubuntu lucid-security main restricted  
deb-src http://security.ubuntu.com/ubuntu lucid-security main restricted  
deb http://security.ubuntu.com/ubuntu lucid-security universe  
deb-src http://security.ubuntu.com/ubuntu lucid-security universe  
deb http://security.ubuntu.com/ubuntu lucid-security multiverse  
deb-src http://security.ubuntu.com/ubuntu lucid-security multiverse  
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu lucid main  
-- INSERT --
```

Fig.4.2. TinyOS apt sources

Next is to update the apt cache:

```
➤ sudo apt-get update
```

After the cache have been updated successfully with all the newest versions it's time to try and install TinyOS. There are several different versions of TinyOS but for this project, the latest available has been used (TinyOS 2.1.1).



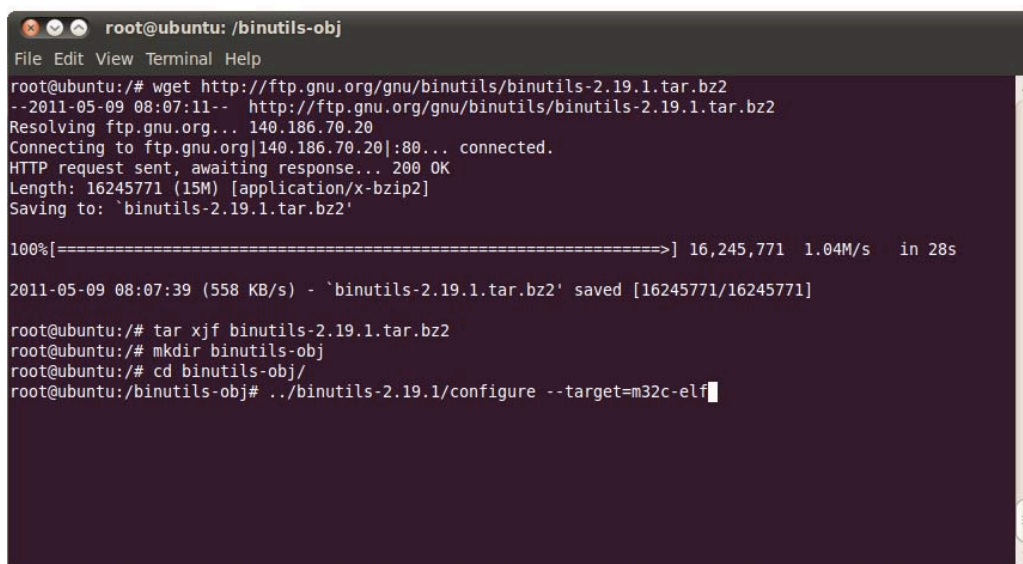
```

root@ubuntu: /
File Edit View Terminal Help
Hit http://us.archive.ubuntu.com lucid-updates/restricted Packages
Hit http://us.archive.ubuntu.com lucid-updates/main Sources
Hit http://us.archive.ubuntu.com lucid-updates/restricted Sources
Hit http://us.archive.ubuntu.com lucid-updates/universe Packages
Hit http://us.archive.ubuntu.com lucid-updates/universe Sources
Hit http://us.archive.ubuntu.com lucid-updates/multiverse Packages
Hit http://us.archive.ubuntu.com lucid-updates/multiverse Sources
Get:1 http://tinyos.stanford.edu lucid Release [1,640B]
Ign http://tinyos.stanford.edu lucid/main Packages
Ign http://tinyos.stanford.edu lucid/main Packages
Hit http://tinyos.stanford.edu lucid/main Packages
Fetched 1B in 1s (1B/s)
Reading package lists... Done
root@ubuntu:/# sudo apt-get install tinyos
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package tinyos is a virtual package provided by:
  tinyos-2.1.1 2.1.1-20100401
  tinyos-2.1.0 2.1.0-20090326
  tinyos-2.0.2 2.0.2-20090326
You should explicitly select one to install.
E: Package tinyos has no installation candidate
root@ubuntu:/# sudo apt-get install tinyos-2.1.1 2.1.1-20100401

```

**Fig.4.3.** TinyOS packages

Next step is to install a compiler, as when installing TinyOS from the repository is not included. First, it is necessary to download *binutils* (collection of binary tools). When provided from [gnu.org](http://gnu.org), the *binutils* installation works as follows:



```

root@ubuntu: /binutils-obj
File Edit View Terminal Help
root@ubuntu:/# wget http://ftp.gnu.org/gnu/binutils/binutils-2.19.1.tar.bz2
--2011-05-09 08:07:11-- http://ftp.gnu.org/gnu/binutils/binutils-2.19.1.tar.bz2
Resolving ftp.gnu.org... 140.186.70.20
Connecting to ftp.gnu.org|140.186.70.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16245771 (15M) [application/x-bzip2]
Saving to: `binutils-2.19.1.tar.bz2'

100%[=====>] 16,245,771  1.04M/s  in 28s

2011-05-09 08:07:39 (558 KB/s) - `binutils-2.19.1.tar.bz2' saved [16245771/16245771]

root@ubuntu:/# tar xjf binutils-2.19.1.tar.bz2
root@ubuntu:/# mkdir binutils-obj
root@ubuntu:/# cd binutils-obj/
root@ubuntu:/binutils-obj# ../binutils-2.19.1/configure --target=m32c-elf

```

**Fig.4.4.** Binutils from gnu

At this point, to compile TinyOS, GCC (Gnu Compiler Collection) is required. There are a lot of versions but the most stable at this document publication date is 4.3.3.

```
➤ wget      ftp://ftp.nluug.nl/mirror/languages/gcc/releases/gcc-4.3.3/gcc-4.3.3.tar.bz2

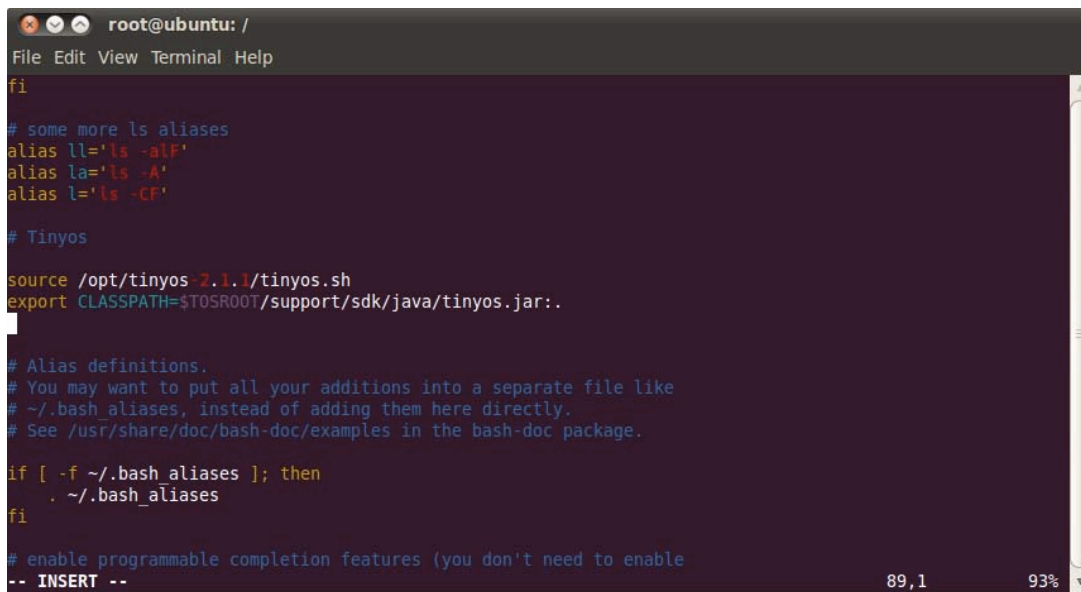
➤ wget      ftp://ftp.nluug.nl/mirror/languages/gcc/releases/gcc-4.3.3/gcc-core-4.3.3.tar.bz2
➤ wget ftp://sources.redhat.com/pub/newlib/newlib-1.17.0.tar.gz
➤ tar xjf gcc-4.3.3.tar.bz2
➤ tar xjf gcc-core-4.3.3.tar.bz2
➤ tar xzf newlib-1.17.0.tar.gz
➤ cd gcc-4.3.3
➤ ln -s ../newlib-1.17.0/newlib/
➤ ln -s ../newlib-1.17.0/libgloss
➤ cd ..
➤ mkdir gcc-obj
➤ cd gcc-obj/
➤ ../gcc-4.3.3/configure --target=m32c-elf --with-newlib --enable-languages="c c++"
➤ make
➤ sudo make install
➤ cd ..
```

To flash the micro controller, the *sm16cf* component is required. *sm16cf* is a python flasher. It can be found at sourceforge.

```
➤ wget      http://sourceforge.net/projects/sm16cf/1.0.0-rc5/sm16cf-1.0.0-rc5.tar.bz2
➤ wget http://sourceforge.net/projects/sm16cf/files/sm16cf/1.0.0-rc5/sm16cf-1.0.0-rc5.tar.bz2
➤ tar xjf sm16cf-1.0.0-rc5.tar.bz2
➤ cd sm16cf-1.0.0-rc5
➤ sudo python setup.py install
```

The last step consists on modifying the *.bashrc* in the */etc* to load TinyOS upon the initialization of the bash shell.





```

root@ubuntu: /
File Edit View Terminal Help
fi

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Tinyos

source /opt/tinyos-2.1.1/tinyos.sh
export CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar:.

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

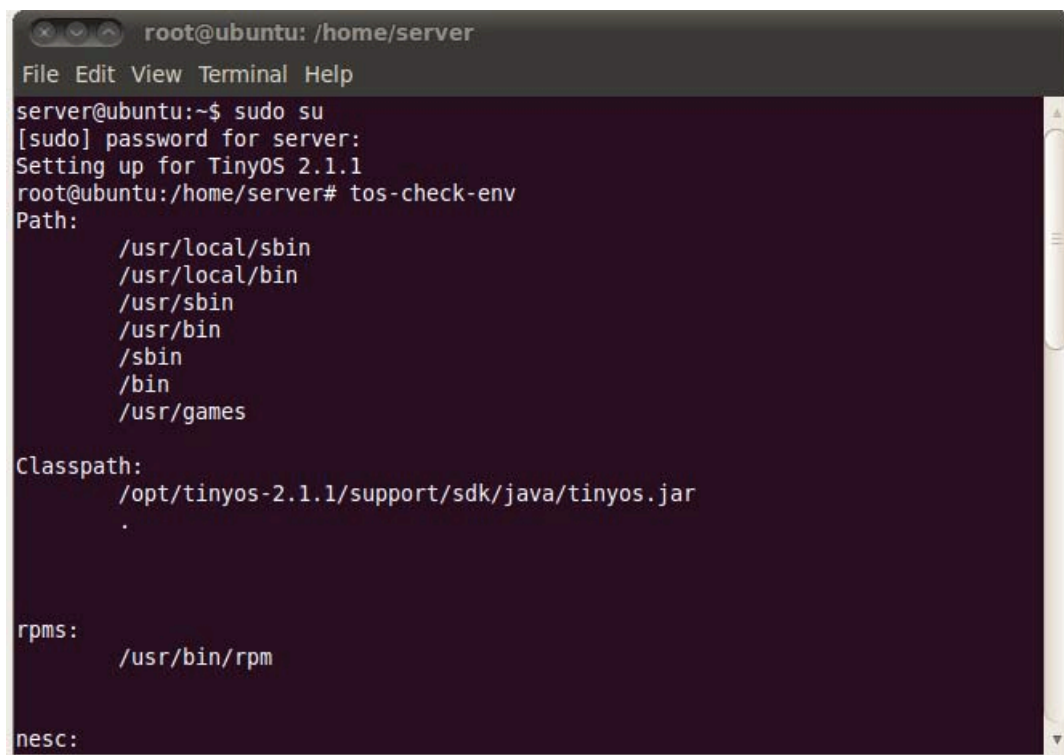
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
-- INSERT --
89,1 93%

```

Fig.4.5. TinyOS.sh

After that, each time that a shell is started, the system executes the *tinyos.sh* file. If this step is omitted, the changes will only be effective until restarting the computer.



```

root@ubuntu: /home/server
File Edit View Terminal Help
server@ubuntu:~$ sudo su
[sudo] password for server:
Setting up for TinyOS 2.1.1
root@ubuntu:/home/server# tos-check-env
Path:
    /usr/local/sbin
    /usr/local/bin
    /usr/sbin
    /usr/bin
    /sbin
    /bin
    /usr/games

Classpath:
    /opt/tinyos-2.1.1/support/sdk/java/tinyos.jar
    .

rpms:
    /usr/bin/rpm

nesc:

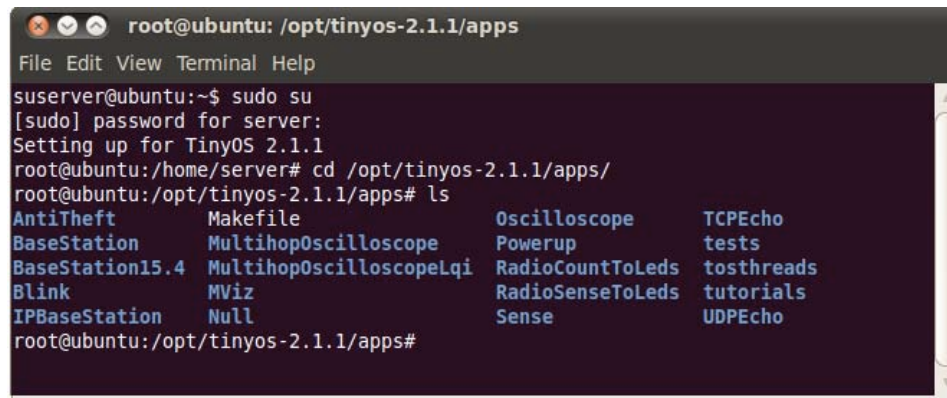
```

Fig.4.6. TinyOS installed

### 4.1.2 Using TinyOS

Now, all the packages have been installed and the TinyOS environment is working properly. Along with the installation packages, some example code is included.

Those examples are also guide to get used to TinyOS.



```

root@ubuntu: /opt/tinyos-2.1.1/apps
File Edit View Terminal Help
suserver@ubuntu:~$ sudo su
[sudo] password for server:
Setting up for TinyOS 2.1.1
root@ubuntu:/home/server# cd /opt/tinyos-2.1.1/apps/
root@ubuntu:/opt/tinyos-2.1.1/apps# ls
AntiTheft      Makefile      Oscilloscope  TCPEcho
BaseStation    MultihopOscilloscope  Powerup       tests
BaseStation15.4 MultihopOscilloscopeLqi RadioCountToLeds tosthreads
Blink          MViz          RadioSenseToLeds tutorials
IPBaseStation  Null          Sense         UDPEcho
root@ubuntu:/opt/tinyos-2.1.1/apps#

```

**Fig.4.7.** TinyOS example applications

One can start understanding TinyOS by following the tutorials found at the TinyOS Wiki.

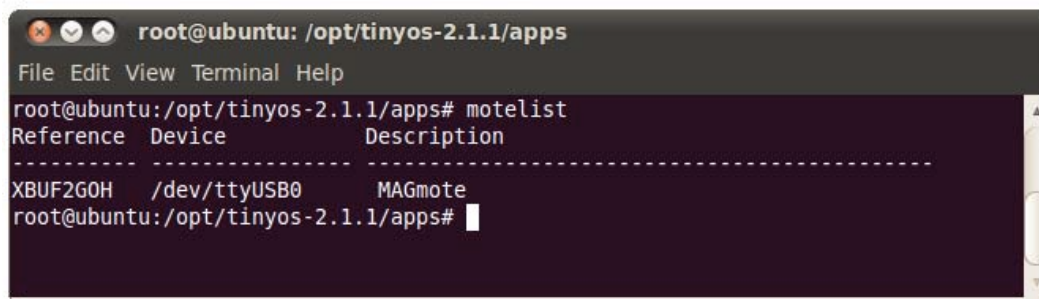
At those tutorials it is explained how execute the code, the relation between the different files and how nesC should be used. Some basic programs can be built into the devices to allow the user checking its nodes proper functioning.

On these tutorials also are listed the main TinyOS commands usable through the shell.

#### 4.1.2.1 Main commands

The main commands used to build the programs into the devices are:

- **motelist**: Lists which USB ports have motes attached along with their name.



```

root@ubuntu: /opt/tinyos-2.1.1/apps
File Edit View Terminal Help
root@ubuntu:/opt/tinyos-2.1.1/apps# motelist
Reference  Device      Description
-----
XBUF2G0H  /dev/ttyUSB0  MAGmote
root@ubuntu:/opt/tinyos-2.1.1/apps#

```

**Fig.4.8.** Motelist command

- ***make 'platform'***: Using this command the program is compiled but not uploaded to the devices, it just creates a TOS image. It is used to check possible errors before installing it inside a mote.

```
➤ make tmote;
```

- ***make 'platform' install,ID bsl,/dev/ttyUSBx***: Compiles and uploads the program to the device attached into the USBx port. It will take a few seconds and after that, the device will start executing the code.

```
➤ make tmote install,1 bsl/dev/ttyUSB0;
```

#### 4.1.2.2 Example applications

As noted before, there are some useful applications coming along the TinyOs environment (Fig.4.7.). On this project we used the following ones:

- ***Blink***: is a simple application that blinks the 3 mote LEDs. It tests that the boot sequence and timers are working properly. The three LEDs blink at 1Hz, 2Hz, and 4Hz. Because an independent timer drives each led, visual inspection can determine whether there are some bugs in the boot or in the timer system.
- ***Null***: Is an application useful to test if the build environment is functional in its most minimal sense, i.e., if you can correctly compile an application. It is also useful to test the minimum power consumption of a node when it has no interrupts or resources active.
- ***BaseStation***: Is an application that acts as a simple Active Message bridge between the serial and radio links. On the serial link, BaseStation sends and receives simple active messages (not particular radio packets): on the radio link, it sends radio active messages, whose format depends on the network stack being used. BaseStation will copy its compiled-in group ID to messages moving from the serial link to the radio, and will filter out incoming radio messages that do not contain that group ID.

BaseStation includes queues in both directions, with a guarantee that once a message enters a queue, it will eventually leave on the other interface. The queues allow the BaseStation to handle load spikes gracefully.

BaseStation acknowledges a message arriving over the serial link only if that message was successfully queued for delivery to the radio link.

- **Oscilloscope:** The oscilloscope program is a simple data-collection demo for the default sensor. It periodically samples this default sensor and broadcasts a message with the info over the radio every 10 readings. These readings can be received by a BaseStation mote and displayed by the Java "Oscilloscope" application, found in the java subdirectory. The sampling rate starts at 4Hz, but can be changed.

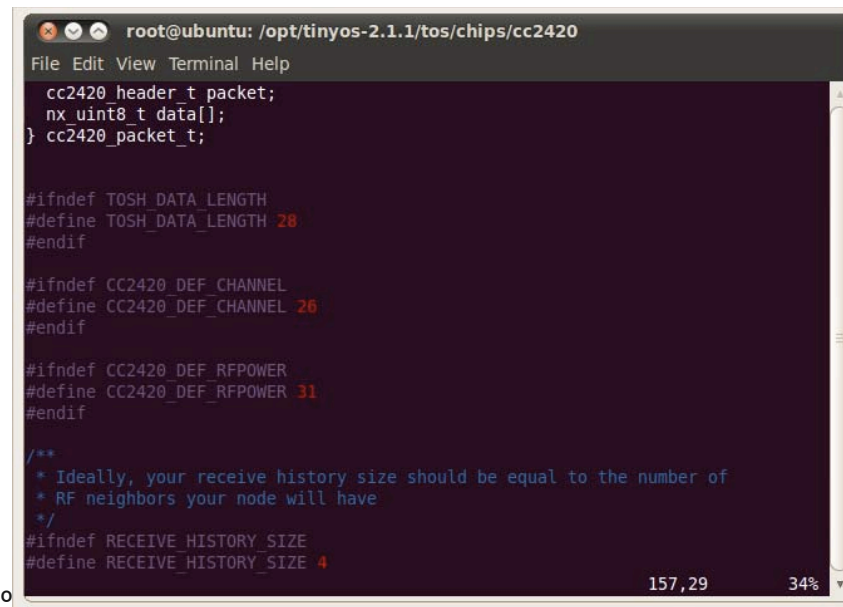
#### 4.1.2.3 CC2420.h

TinyOS allows us to modify the radio parameters of our motes. Doing that, we can choose the channel and transmission power, that best fits our needs.

Because of that, our motes work in the channel 26 since is the one where there is less interference with other wireless technologies that also transmit in 2.4 GHz frequency.

Another significant variable is the one that defines the power which nodes are transmitting. During the tests, the power was modified to check its behavior but finally the maximum power (0dBm) was chosen in order to achieve the longest range.

To change the channel or the transmission power, the file *CC2420.h* included in */opt/tinyos-2.1.1/tos/chips/cc2420* must be modified.



```
root@ubuntu: /opt/tinyos-2.1.1/tos/chips/cc2420
File Edit View Terminal Help
cc2420_header_t packet;
nx_uint8_t data[];
} cc2420_packet_t;

#ifndef TOSH_DATA_LENGTH
#define TOSH_DATA_LENGTH 28
#endif

#ifndef CC2420_DEF_CHANNEL
#define CC2420_DEF_CHANNEL 26
#endif

#ifndef CC2420_DEF_RFPOWER
#define CC2420_DEF_RFPOWER 31
#endif

/**
 * Ideally, your receive history size should be equal to the number of
 * RF neighbors your node will have
 */
#ifndef RECEIVE_HISTORY_SIZE
#define RECEIVE_HISTORY_SIZE 4
#endif
```

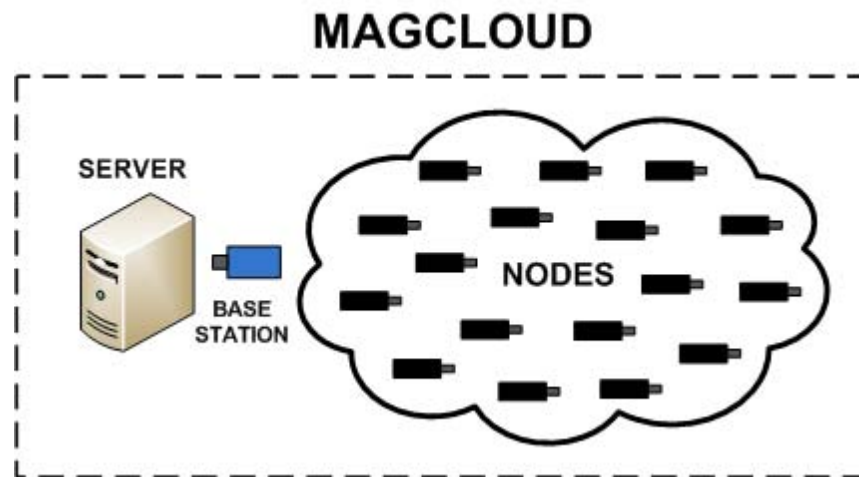
Fig.4.9. CC2420.h

Another important factor in wireless communications are the RSSI (Receive Signal Strength Indication) and the PER (Packet Error Rate). The RSSI is a measurement of the power present in a received radio signal, and the PER is the number of incorrectly received data packets divided by the total number of received packets.

For further information on RSSI and PER, it can be found a PFC made in this University (EETAC) providing a TMote application that can measure those parameters.

## 4.2 Our Software

With the environment set up, it is time to understand the way it works our software. In the MAGCLOUD system, three different kinds of entities can be distinguished:



**Fig.4.10.** MAGCLOUD complete system

- **Sensor nodes:** All of them together form the sensor cloud and to those corresponds the task of measuring, encapsulating and sending the magnitudes to the main node.
- **Base station:** Is the node that acts as a gateway between the radio cloud and the server.
- **The server:** Is the machine that can render and store the data received from the cloud.

It is mandatory for the software driving our system to be robust, powerful and easy to use.

## The sensor nodes

The main function of the software inside the nodes is taking raw reads from the sensor inputs and transmitting this info to the network.

*Easy-Sen* is a hardware corporation specialized in sensor modules for the *TMote Sky* platform. For the use of their devices, they offer fully tested software that is capable of working in the most demanding environments. The great news for us is that this software does exactly what we need in the way we want and it is open and free. This is the software we will use.

The first step is downloading and unzipping the package on the hard drive. It is easily installed executing the following command from the folder containing the software:

```
➤ make tmote install,X
```

Where *X* is the identifier of the node on the cloud. Every node will have a different identifier. Root permissions are required for this operation.

## The Base Station

The central node or Base Station is the responsible of passing all the cloud packets to the server. The program entrusted of this task is the included in the examples package of TinyOS and fore sure is fully robust, tested and supported. It installs using the same command we used for the previous program but typed from its corresponding folder. In our case is located in:

```
➤ /opt/tinyos-2.1.1/apps/BaseStation
```

## The server

The server is the responsible of receiving, representing, storing and offering to external networks the data from the cloud. The program installed on the nodes also included a server module capable of doing such tasks.

To start representing the data received from the network, the program must be executed as follows: (When the central node is on USB0)

```
➤ java net/tinyos.tools.MsgReader SenseToRadioMsg -comm  
serial@/dev/ttyUSB0:telosb
```

The Easy-Sen Server module produces the following output:

```

root@ubuntu: /home/ubuntu/Desktop/SenseToRadioMsg-1
File Edit View Terminal Help
[data=0x692 0x409 0x627 0x505 0x3fd 0x3e3 0x594 0x56f ]
[counter=0x1772]
[nodeid=0x9]
1305047096070: Message <SenseToRadioMsg>
[data=0x693 0x414 0x63a 0x529 0x436 0x3a0 0x561 0x58d ]
[counter=0x1776]
[nodeid=0x9]
1305047098006: Message <SenseToRadioMsg>
[data=0x697 0x41b 0x64c 0x53f 0x448 0x3af 0x581 0x53c ]
[counter=0x177a]
[nodeid=0x9]
1305047099963: Message <SenseToRadioMsg>
[data=0x6ba 0x43d 0x6a9 0x534 0x443 0x3e4 0x5c3 0x5a6 ]
[counter=0x177e]
[nodeid=0x9]
1305047101913: Message <SenseToRadioMsg>
[data=0x6bb 0x444 0x6a4 0x525 0x41f 0x3a9 0x562 0x589 ]
[counter=0x1782]
[nodeid=0x9]
1305047103859: Message <SenseToRadioMsg>
[data=0x6b1 0x43c 0x695 0x51d 0x423 0x3b3 0x569 0x5a6 ]
[counter=0x1786]
[nodeid=0x9]
1305047105810: Message <SenseToRadioMsg>
[data=0x6b0 0x432 0x685 0x4fd 0x478 0x408 0x585 0x574 ]
[counter=0x178a]
[nodeid=0x9]
^Croot@ubuntu:/home/ubuntu/Desktop/SenseToRadioMsg-1# java net.tinyos.tools.MsgReader SenseToRadioMsg -comm serial@/dev/tty
B0:telosb | java ReadLine

```

**Fig.4.11.** Easy-Sen server module default output

The output structure is quite complex. It contains the value from the magnitude read coded between the 0x0 and 0xFF (hex).

We are interested on rendering only the value metered from the inputs, in a more clear way. Doing so, the adaptation to a remote monitoring and graphing platform should be easy. Plugins included with most of those platforms usually allow querying by executing a file that returns just a value or a very simple structured string, not the kind of output we are getting.

In order to adapt this output to meet our requirements, we developed the following Java program:

```

import java.io.*;
import java.lang.*;

class ReadLine {
    public static void main(String[] args) throws IOException {
        String CurLine = "";

        InputStreamReader converter = new InputStreamReader(System.in);
        BufferedReader in = new BufferedReader(converter);
        int nLine=2;

        while (CurLine!=null) {
            CurLine = in.readLine();
            if (CurLine!=null) {

                if(nLine%5==0) {
                    String[] vector = CurLine.split(" ");

                    vector[2]=vector[2].substring(8);
                    vector[3]=vector[3].substring(2);
                    vector[4]=vector[4].substring(2);
                    vector[5]=vector[5].substring(2);
                    vector[6]=vector[6].substring(2);
                    vector[7]=vector[7].substring(2);
                    vector[8]=vector[8].substring(2);
                    vector[9]=vector[9].substring(2);

                    int S1 = Integer.parseInt(vector[2],16);
                    int S2 = Integer.parseInt(vector[3],16);
                    int S3 = Integer.parseInt(vector[4],16);
                    int S4 = Integer.parseInt(vector[5],16);
                    int S5 = Integer.parseInt(vector[6],16);
                    int S6 = Integer.parseInt(vector[7],16);
                    int S7 = Integer.parseInt(vector[8],16);
                    int S8 = Integer.parseInt(vector[9],16);

                    S1 = (S1*1000/2750)-3;
                    S2 = (S2*1000/1644)-3;
                    S3 = (S3*1000/1644)-3;
                    S4 = (S4*1000/1369)-2;
                    S5 = (S5*1000/1644)-3;
                    S6 = (S6*1000/1644)-3;
                    System.out.println( "Primero:"+S1+"mV
                                         Segundo:"+S2+"mV
                                         Tercero:"+S3+"mV
                                         Cuarto:"+S4+"mV
                                         Quinto:"+S5+"mV
                                         Sexto:"+S6+"mV");

                }
                nLine++;
            }
        }
    }
}

```

To execute it, the following command has to be invoked from the folder containing both programs:



```
➤ java net.tinyos.tools.MsgReader SenseToRadioMsg -comm
  serial@/dev/ttyUSB0:telosb | java ReadLine
```

It translates the metered values to decimal and removes all the clutter to render the output in the desired clear way. The program can be easily customized to return the value of just one sensor or any other required option.

```
serial@/dev/ttyUSB1:115200: resynchronising
Primero:1582 Segundo:1138 Tercero:1798 Cuarto:1291 Quinto:1015 Sexto:880
Primero:1600 Segundo:1156 Tercero:1586 Cuarto:1307 Quinto:1035 Sexto:793
Primero:1555 Segundo:1094 Tercero:1696 Cuarto:1298 Quinto:1027 Sexto:780
Primero:1529 Segundo:1207 Tercero:1714 Cuarto:1345 Quinto:1082 Sexto:852
Primero:1598 Segundo:1164 Tercero:1607 Cuarto:1346 Quinto:1083 Sexto:839
Primero:1589 Segundo:1140 Tercero:1790 Cuarto:1458 Quinto:1087 Sexto:844
Primero:1592 Segundo:1141 Tercero:1820 Cuarto:1338 Quinto:1075 Sexto:833
Primero:1593 Segundo:1159 Tercero:1596 Cuarto:1316 Quinto:1042 Sexto:797
Primero:1546 Segundo:1084 Tercero:1671 Cuarto:1270 Quinto:1119 Sexto:889
Primero:1622 Segundo:1190 Tercero:1668 Cuarto:1258 Quinto:1098 Sexto:865
Primero:1570 Segundo:1122 Tercero:1752 Cuarto:1397 Quinto:1150 Sexto:914
Primero:1587 Segundo:1146 Tercero:1812 Cuarto:1325 Quinto:1062 Sexto:832
Primero:1608 Segundo:1168 Tercero:1607 Cuarto:1334 Quinto:1069 Sexto:833
Primero:1592 Segundo:1150 Tercero:1824 Cuarto:1334 Quinto:1064 Sexto:825
Primero:1611 Segundo:1176 Tercero:1636 Cuarto:1386 Quinto:1135 Sexto:910
Primero:1595 Segundo:1154 Tercero:1590 Cuarto:1310 Quinto:1046 Sexto:801
Primero:1593 Segundo:1159 Tercero:1589 Cuarto:1322 Quinto:1044 Sexto:800
Primero:1578 Segundo:1126 Tercero:1770 Cuarto:1426 Quinto:1052 Sexto:818
Primero:1586 Segundo:1136 Tercero:1780 Cuarto:1460 Quinto:1091 Sexto:863
Primero:1609 Segundo:1180 Tercero:1647 Cuarto:1410 Quinto:1031 Sexto:788
Primero:1542 Segundo:1086 Tercero:1671 Cuarto:1264 Quinto:1113 Sexto:877
Primero:1586 Segundo:1146 Tercero:1820 Cuarto:1330 Quinto:1077 Sexto:850
Primero:1550 Segundo:1086 Tercero:1685 Cuarto:1296 Quinto:1019 Sexto:886
Primero:1576 Segundo:1124 Tercero:1748 Cuarto:1392 Quinto:1140 Sexto:903
Primero:1611 Segundo:1170 Tercero:1629 Cuarto:1382 Quinto:1131 Sexto:904
Primero:1583 Segundo:1144 Tercero:1798 Cuarto:1286 Quinto:1011 Sexto:871
Primero:1593 Segundo:1163 Tercero:1613 Cuarto:1362 Quinto:1102 Sexto:872
Primero:1603 Segundo:1162 Tercero:1606 Cuarto:1339 Quinto:1078 Sexto:841
Primero:1599 Segundo:1158 Tercero:1600 Cuarto:1318 Quinto:1053 Sexto:823
Primero:1594 Segundo:1155 Tercero:1591 Cuarto:1315 Quinto:1040 Sexto:792
Primero:1538 Segundo:1082 Tercero:1678 Cuarto:1273 Quinto:1128 Sexto:881
Primero:1603 Segundo:1164 Tercero:1604 Cuarto:1330 Quinto:1059 Sexto:830
Primero:1608 Segundo:1168 Tercero:1616 Cuarto:1346 Quinto:1084 Sexto:833
```

**Fig.4.12.** Customized output

If the output wants to be stored in a “log.txt” file, the following command can be used:

```
➤ java net.tinyos.tools.MsgReader SenseToRadioMsg -comm
  serial@/dev/ttyUSB0:telosb | java ReadLine > log.txt
```

If monitoring from another machine is required, the computer connected to the WSN can be remotely reached through SSH service.

## CHAPTER 5. RESULTS

It is time to see how the nodes behave in the real world. In this chapter, mote accuracy and electric consumption are measured.

### 5.1 Accuracy:

The original TMote manufacturer provides the following theoretical range for each input:

**Table 5.1.** Inputs theoretical range

| Input | Range      |
|-------|------------|
| 1     | 0 to 1,5 V |
| 2     | 0 to 2,5 V |
| 3     | 0 to 2,5 V |
| 4     | 0 to 3,0 V |
| 5     | 0 to 2,5 V |
| 6     | 0 to 2,5 V |

It is important to know how close our boards are to those theoretical values. To measure the behaviour of our very last board, the maximum possible range (0 to 3 volts) was divided on 15 parts (200mV increment) and for every part, 100 samples were taken.

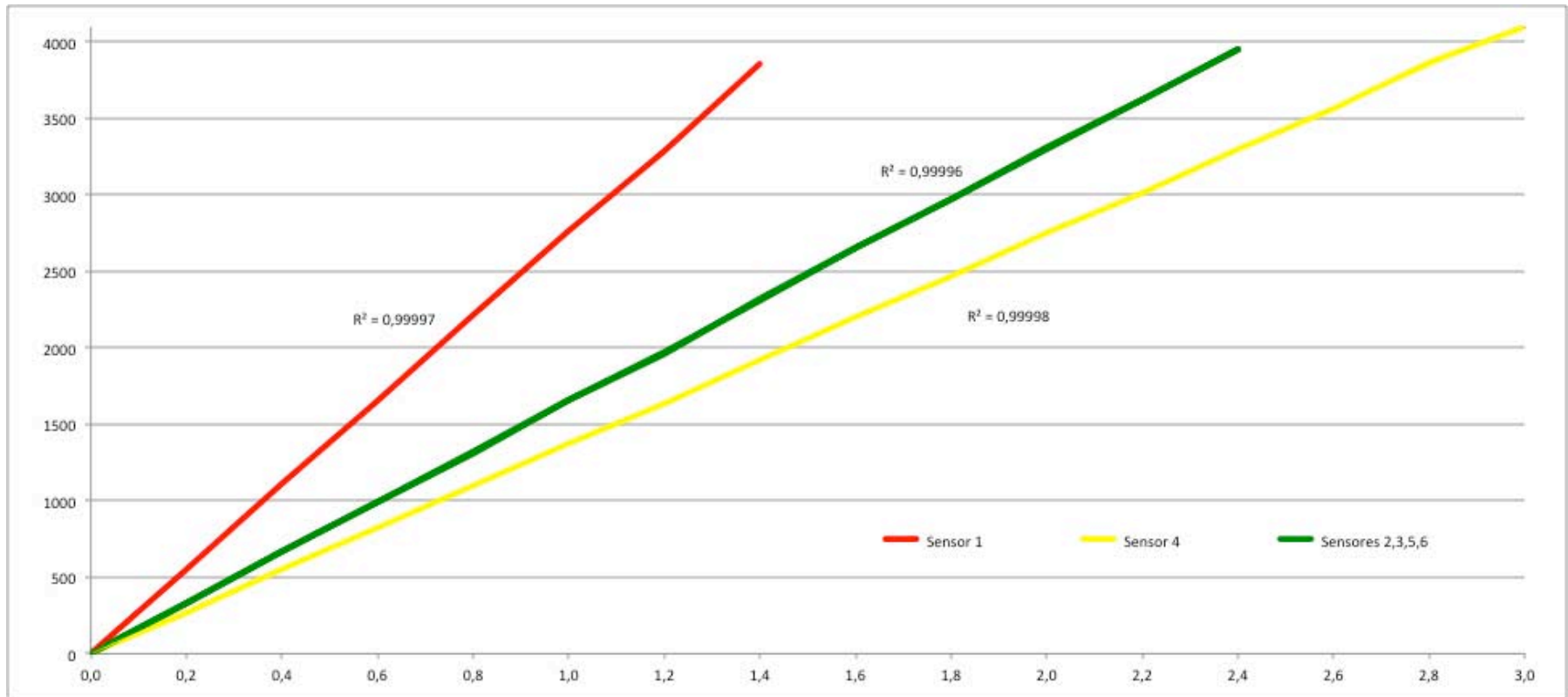
The procedure was the following:

- 1- Precisely adjust the voltage source to required value using a voltage meter.
- 2- Connect to the corresponding input of the node.
- 3- Start the transmission to the central node.
- 4- On the destination computer, take and store 100 samples.
- 5- Go back to step 1.

This loop was repeated 15 times for a single sensor, 90 times for the full board.

1500 samples for every sensor are too many to do by hand. We designed a program that was taking the samples and storing them inside a CSV file that we could later import to Excel.

After some days, we had 9.000 samples distributed over 90 CSV files. We then imported to Excel and calculated the mean and standard deviation for all of them. The results can be found on the following page.



**Fig.5.1.** Metered inputs sensitivity

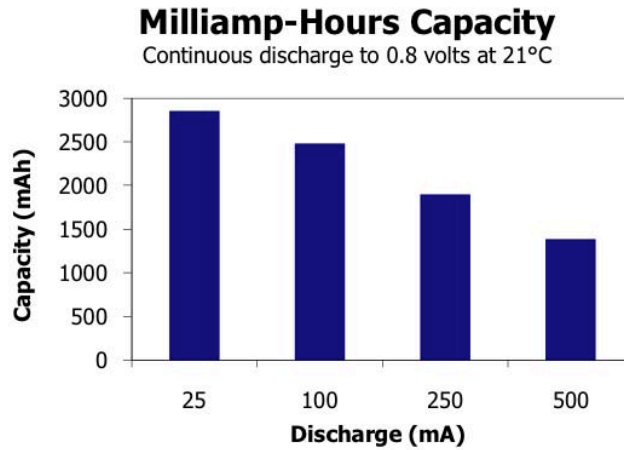
Our boards are precise enough for most of the possible real word applications if using the theoretical curves. Whenever more precision is required, the node can be calibrated at any time by using this procedure.



## 5.2 Consumption:

It is very important that nodes that have to operate in an autonomous way require the least possible amount of energy. In our case, nodes have to survive for periods of over one year with just two AA batteries.

We chose two easily available Energizer AA alkaline batteries for the calculations. Considering a drain rate less than 25 mA, the datasheet claims a capacity of about 2850mAh.



**Fig.5.2.** Energizer alkaline AA capacity

When looking at the CC2420 datasheet, one notices that it would take less than a week to fully exhaust both batteries for this component alone (RX).

$$\frac{\left(\frac{2850 \text{ mAh}}{19,7 \text{ mA}}\right)}{24 \text{ h}} < 7 \text{ days}$$

To last for a full year, the maximum node consumption should be:

$$\frac{2850 \text{ mAh}}{(365 \text{ days} \times 24 \text{ h})} = 0,32 \text{ mA}$$

The only way of using just this tiny amount of energy is turning the unnecessary components off whenever possible. The Duty Cycle gives idea of the fraction of the total time the node will be fully enabled or in sleep mode.

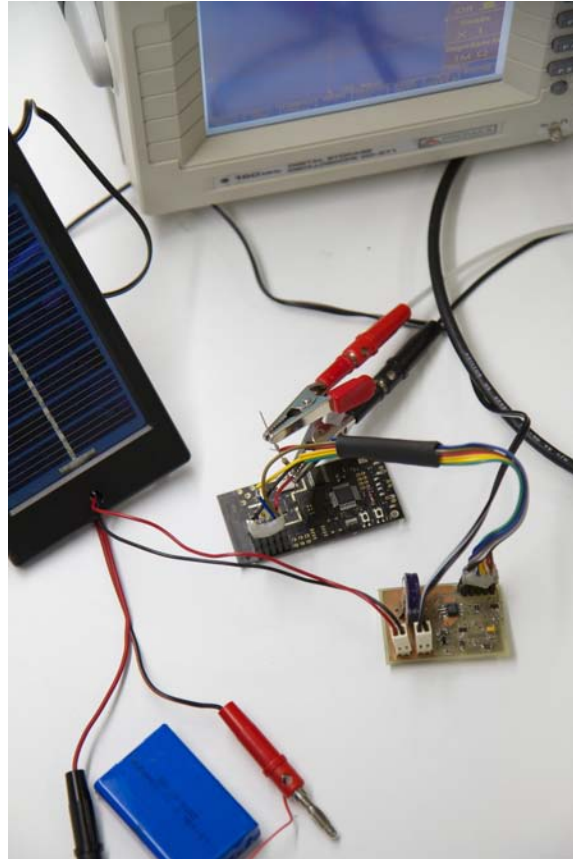
If we want our node to last over one year while just using two AA batteries:

$$(ISL \times (1 - DC)) + (ION \times DC) < 0,32 \text{ mA}$$

Being DC the Duty Cycle, ION the consumption when the node is ON and ISL the consumption when the node is in Sleep or standby.

To measure the consumption of the node when in Sleep mode, we used:

- A node
- An energy source
- An oscilloscope
- A  $10\Omega$  resistor



**Fig.5.3.** Measured *Null* software consumption

The first step is installing the program *Null* on the node. This software turns off the radio, leds and any unnecessary part. As the author explains on the *Null* program help, it was designed for measuring consumption of a node in Standby mode.

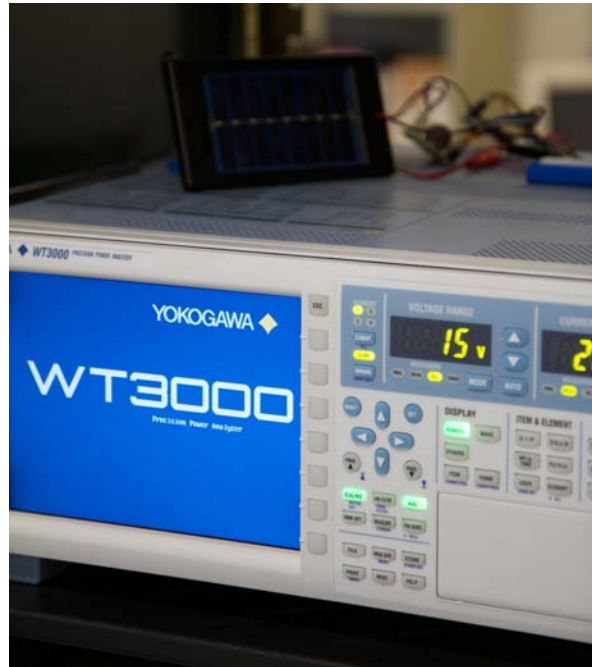
The second step is connecting the node to the energy source while placing the  $10\Omega$  resistor in series. In our case, we used a solar powered source we were testing, but any other suitable source could be used.

Third step is measuring the voltage falling in the  $10\Omega$  resistor. Knowing those two values, one can easily calculate the current crossing the resistor and thus also flowing through the node:



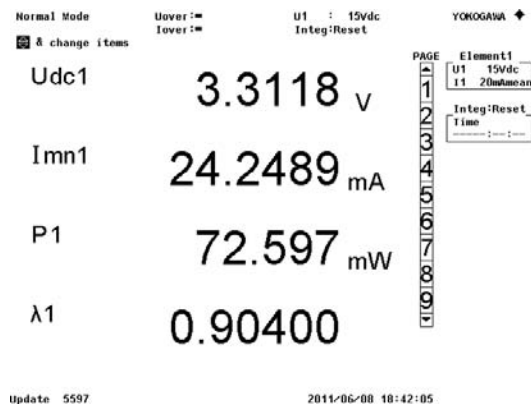
$$I_{SL} = \frac{V_{SL}}{R} = \frac{1,25 \text{ mV}}{10 \Omega} = 125 \mu\text{A}$$

To calculate the consumption of a node when in full power, we used a different approach. It was after we noticed there was a device on the lab specifically suited for this kind of task: The YOKOGAWA WT 3000. On the back of this precision power analyser there are a pair of terminals for the current and another for the voltage.



**Fig.5.4.** Measured full power consumption

We slowly increased the cadency of sent packets until the node consumption stopped going up. At this maximum consumption setting, the current was about 25mA.



**Fig.5.5.** Maximum consumption

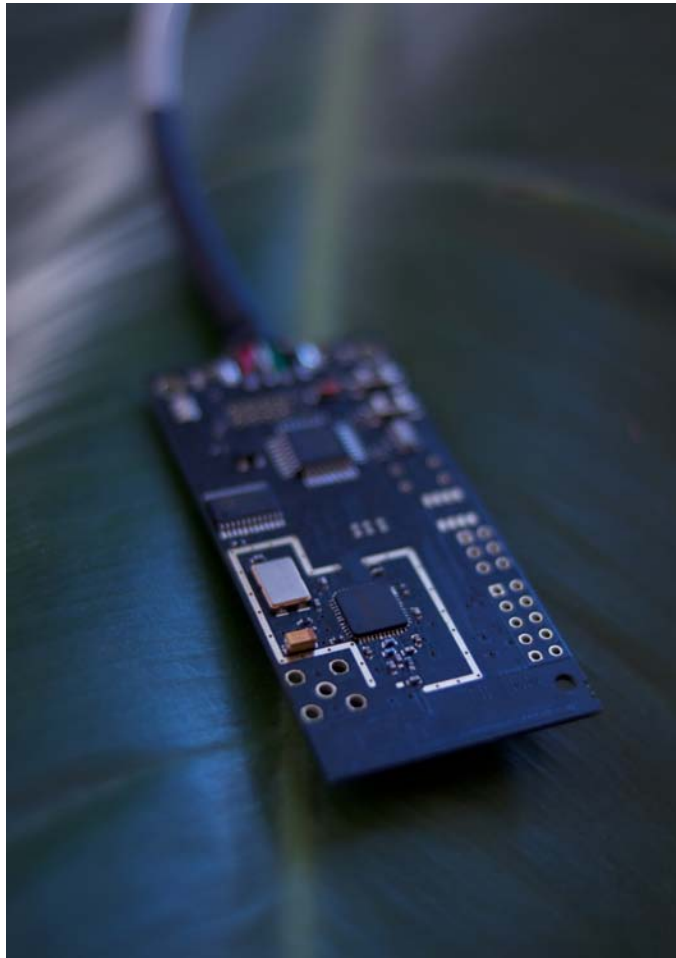


If we consider a Duty Cycle of 1%, our nodes would last for:

$$\begin{aligned}
 \textit{Autonomy} &= \frac{\textit{Capacity}}{(ISL \times (1 - DC)) + (ION \times DC)} \\
 &= \frac{2850 \text{ mAh}}{(125 \mu A \times (1 - 0,01)) + (25 \text{ mA} \times 0,01)} \approx 7625 \text{ h} \approx \mathbf{317 \text{ days}}
 \end{aligned}$$

### 5.3 Time and Costs

In the following table is shown the cost of every component used in the project. As we said before, nodes time construction when proper technique is learned is about 8 hours for both sides.



**Fig.5.6.** Finished node

| QUANT. | REFERENCE                                       | VALUE              | PACKAGE               | SUPPLIER CODE  | COST €<br>(10 units) | COST €<br>(100 units) | COST €<br>(+200 units) |
|--------|---|--------------------|-----------------------|----------------|----------------------|-----------------------|------------------------|
| 1      | C84   | 10n                | SM/C_0402             | 1650806        | 0,052                | 0,021                 | 0,011                  |
| 1      | C21   | 33n                | SM/C_0402             | 1758892        | 0,009                | 0,005                 | 0,002                  |
| 2      | C81,C61   | 0.5p +/-0.25p np0  | SM/C_0402             | 1650777        | 0,054                | 0,041                 | 0,037                  |
| 2      | C71,C73   | 5.6p +/-0.25p np0  | SM/C_0402/DUAL/INLINE | 1650796        | 0,127                | 0,125                 | 0,095                  |
| 2      | C381,C391                                       | 22p 5% np0         | SM/C_0402             | 1327634        | 0,115                | 0,022                 | 0,019                  |
| 3      | C82,C83,C86                                     | 68p                | SM/C_0402             | 1414596        | 0,031                | 0,009                 | 0,006                  |
| 14     | C1,C2,C4,C5,C6,C7,C8,C9,C20,C22,C23,C24,C85,C87 | 0.1u               | SM/C_0402             | 1650809        | 0,021                | 0,018                 | 0,011                  |
| 3      | C3,C25,C64                                      | 10uF LowESR < 5ohm | SM/TANTANUM           | 197130         | 0,49                 | 0,3                   | 0,184                  |
| 2      | D4,D21  | Red Clear          | SMD 0603 NUM          | 1685073        | 0,23                 | 0,164                 | 0,147                  |
| 2      | D5,D20  | Green Clear        | SMD 0603 NUM          | 1685076        | 0,4                  | 0,26                  | 0,24                   |
| 1      | D6  | Blue Clear         | SMD 0603 NUM          | 1685096        | 0,4                  | 0,3                   | 0,22                   |
| 1      | D22   | 1N5817             | SSS Mini 2P           | 621-LLSD103A-7 | 0,396                | 0,222                 | 0,141                  |
| 3      | L2,L3,L20                                       | F Bead 240-1035-1  | SM/C_0805             | 240-2389-6-ND  | 0,08                 | 0,06                  | 0,03                   |
| 1      | L62   | 5.6n 5%            | SM/C_0402             | 1265406        | 0,44                 | 0,28                  | 0,21                   |
| 2      | L81,L61   | 7.5n 5%            | SM/C_0402             | 1265409        | 0,44                 | 0,28                  | 0,21                   |
| 1      | R475  | 0                  | SM/C_0402             | 1469661        | 0,033                | 0,02                  | 0,01                   |
| 2      | R22,R23   | 27                 | SM/C_0402             | 1738829        | 0,028                | 0,015                 | 0,008                  |
| 3      | R9,R25,R26                                      | 100                | SM/C_0402             | 1738837        | 0,022                | 0,015                 | 0,008                  |
| 1      | R8  | 220                | SM/C_0402             | 1738841        | 0,028                | 0,015                 | 0,008                  |
| 2      | R7,R20  | 470                | SM/C_0402             | 1738845        | 0,028                | 0,015                 | 0,008                  |
| 1      | R24   | 1.5k               | SM/C_0402             | 1653201        | 0,074                | 0,037                 | 0,019                  |
| 2      | R2,R27  | 2.2k               | SM/C_0402             | 1738854        | 0,028                | 0,015                 | 0,008                  |

|           |                               |                 |                         |                      |       |       |       |
|-----------|-------------------------------|-----------------|-------------------------|----------------------|-------|-------|-------|
| 6         | R10,R28,R29,R30,<br>R474,R476 | 10k             | SM/C_0402               | 1469669              | 0,031 | 0,018 | 0,009 |
| 1         | R451                          | 43k 1%          | SM/C_0402               | 1358086              | 0,044 | 0,032 | 0,017 |
| 3         | R1,R11,R12                    | 100k 1%         | SM/C_0402               | 1738877              | 0,028 | 0,015 | 0,008 |
| 1         | R4                            | 470k            | SM/C_0402               | 9239553              | 0,062 | 0,03  | 0,015 |
| 3         | R32,R472,R473                 | 1m              | SM/C_0402               | 1469667              | 0,031 | 0,018 | 0,009 |
| 1         | R5                            | 5.1m            | SM/C_0402               | 1458808              | 0,02  | 0,013 | 0,009 |
| 2         | SW2,SW1 (3x4mm)               | EVQ-P2K02Q      | SW/EVQP2                | 1437637              | 0,48  | 0,36  | 0,192 |
| 1         | U0                            | TI_MSP430_F1611 | QUAD.50M/64/W<br>G12.00 | 1470487              | 12,12 | 10,28 | 9,23  |
| 1         | U5                            | M25P80          | SOIC 8PIN<br>150MIL     | 1734968              | 1,23  | 1,02  | 0,82  |
| 1         | U9                            | DS2411          | SOT321                  | 700-<br>DS2411RT&R   | 1,03  | 0,62  | 0,484 |
| 1         | U10                           | CC2420          | QLP/48                  | 839-CC2420R          | 8,15  | 6,02  | 4,4   |
| 1         | U20                           | FT232BM         | LQFP/32-LD              | 895-FT232BL          | 5,27  | 4,76  | 3,56  |
| 1         | U22                           | USB A           | USB-A                   | 1760681              | 5,5   | 4,51  | 4,12  |
| 1         | U23                           | 93C46           | SOIC 8PIN<br>150MIL     | 1607557              | 0,21  | 0,189 | 0,189 |
| 1         | U25                           | MCP1700T-3302TT | SOT231                  | 1605554              | 0,87  | 0,52  | 0,52  |
| 1         | U27                           | ADG715BRU       | TSSOP/24                | 9604375              | 3,78  | 2,92  | 2,92  |
| 1         | U29                           | NC7WZ126        | US/8                    | 1417647              | 0,65  | 0,25  | 0,25  |
| 1         | X0                            | 32kHz           | CMR200TB                | 1652560              | 0,79  | 0,56  | 0,5   |
| 1         | X1                            | 16MHZ - 16pf    | HK-XTAL-7X5             | 815-ABMM-16-<br>B2-T | 0,7   | 0,632 | 0,581 |
| 1         | X3                            |                 |                         | 81-<br>CSTCR6M00G53  | 0,363 | 0,231 | 0,182 |
| 1         | PCB BOARD                     |                 |                         |                      | 4,89  | 2,84  | 1,73  |
| TOTAL (€) |                               |                 |                         |                      | 50,5  | 37,7  | 31,3  |

## CHAPTER 6. UPGRADES

One of the main characteristics of MAGCLOUD is that it's an almost infinitely upgradeable platform. Following are shown some possible upgrades.

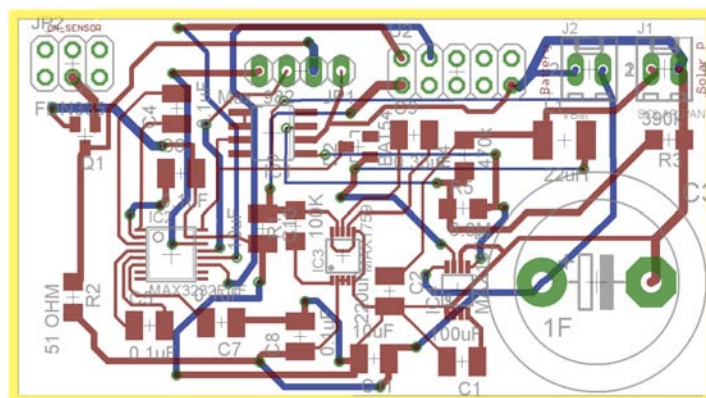
### 6.1 Hardware Upgrades

#### Solar Panel

With the purpose of giving our system an unlimited autonomy, we can add a solar panel to go recharging the battery. We have several circuit designs made in the EETAC to adapt diverse solar panels to the input of TMote nodes. Specifically, Jorge Eduardo Higuera provided us three types of modules.



**Fig.6.1.** Solarex MSX-005



**Fig.6.2.** Power submodule

See *Annex C* for more information.

## Infinite sensing magnitudes

MAGCLOUD works with any sensor whose response can be adapted to the incoming voltage range of our nodes.

Because of that, the system can be upgraded to monitor with precision the majority of measureable magnitudes. There are many schematics in the internet with different types of sensors adapted to TMote Sky boards that can be easily integrated in MAGCLOUD.

## Universal sealed boxes

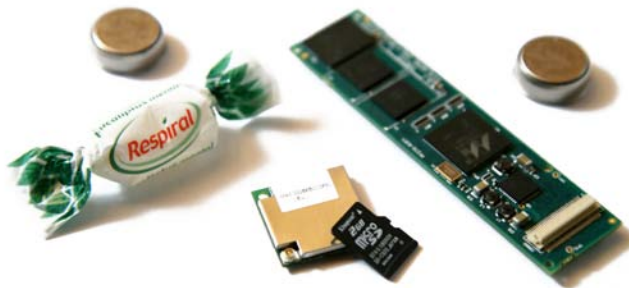
A sealed box with upper and lower connectors can be designed to pile up motes, different energy sources and sensors. Those boxes can plug one to the other to form more complex systems quickly, without need to take the components out of their boxes. The modules would be provided with the necessary connectors to work in multitude of environments.

An example could be a central box with a MAGmote, and upper one with a solar panel and the lower one containing some sensors.

## Remote Base Station

In certain situations, it can be interesting to separate the cloud subsystem from the main server. It can be done by connecting the central node with a GPRS/3G router. This router will be the responsible of transmitting all the cloud data to the server.

There are many solutions on the market that can realize this operation. One possible option due to its powerful features would be connecting our mote to a GPRS capable GUMSTIX. It is a very robust platform and, as can be seen in the picture, due to its tiny size it can be adapted to the most environments.



**Fig.6.3.** Gumstix

## 6.2 Software Upgrades

### 6.2.1 IPV6

One of many TinyOS characteristics is the support for 6LoWPAN. BLIP (Berkeley Low-power IP) stack, is an implementation in TinyOS of a number of IP-based protocols.



Using this implementation, one can form multi-hop IP networks consisting of different nodes communicating over shared protocols. Due to the rapid evolution of the IETF and IEEE, blip is currently not completely compliant; however, it does provide significant interoperability with other IP networks.

The blip router is only supported on the latest version of TinyOS 2.1.1. This is exactly the version we are using so installing it on our system results easy. There are many manuals on the Internet explaining its installation. There also can be found some PFCs based in this implementation. (See *Annex D*)

### 6.2.2 Monitoring Software

The perfect option to complete MAGCLOUD would be blending it with a remote monitoring environment. The chosen platform should accomplish certain characteristics to guarantee an effective management of the cloud data.

- It must have a neat and clear GUI (Graphical User Interface)
- It must permit to easily draw graphics with the magnitude values
- It must permit to monitor other problems or alerts from MAGCLOUD system.
- It also must have many other useful submodules available.

As the rest of the used software, the monitoring system must be open, free and must be able to run over UNIX-LINUX systems as Ubuntu. Nowadays exist a great variety of powerful and easy tools that can fit our needs.



**Fig.6.4.** Monitoring tools

We think that NetMRG is great candidate. Having worked with this program before in our TFC, we can say that it has the required characteristics.

## CHAPTER 7. ENVIRONMENTAL IMPACT

From the beginning of the project we have taken into account the environmental impact of the manufacture or the extensive running that our nodes could cause.

The energy saving is fundamental. Because of that, our nodes use the technologies allowing a minimum consumption. By default, they are capable of using rechargeable batteries but they can also mount a solar panel to use renewable energy.

As said before, we have used lead-free tin for the construction of the boards, having prepared for that a custom profile in the oven.

All MAGCLOUD used components are also environmentally friendly; every single one accomplishes with RoHS (Restriction of Hazardous Substances) WEEE regulations. RoHS restricts the use of six hazardous substances helping the efficient recycling process of the products at its end of life.

Those substances are:

- Lead (Pb)
- Mercury (Hg)
- Cadmium (Cd)
- Hexavalent chromium (Cr6+)
- Polybrominated biphenyls (PBB)
- Polybrominated diphenyl ether (PBDE)



In addition, it has to be taken into account that one of the largest application fields of MAGCLOUD is the environmental monitoring itself. Because of that, ours is a system that keeps in mind a sustainable future not only because it will be causing minimum environmental impact but also because it could be really helpful in the race for its care.

In closing, every provided copy of this memory is printed on recycled paper. We also prepared a custom profile for the printer to allow the high-resolution output the pictures require while keeping a very low ink density for minimizing the consumption.



## CHAPTER 8. CONCLUSIONS

MAGCLOUD was born from our dream of interfacing any magnitude with the world in an easy way. This project is the evidence of how close has come the real implementation to the original idea.

We have physically assembled some pro-looking hardware, enough rugged for real world deployment. Every single requirement we initially stated has been reached.

Now, we have EVIDENCE to assert that MAGCLOUD:

- Can provide connectivity on almost any environment.
- Can be upgraded to reach almost any future requirement.
- Supports extensive ready to use software and sensing schematics.
- Provides massive free support documents and help.
- Is easy to use and easy to manage.
- Takes care of the environment.
- Nodes can be assembled within our actual lab equipment.
- Nodes final cost is the cost of the hardware plus 8 hours of a student work.

MAGCLOUD is real world stuff. MAGCLOUD is only limited by imagination.

From our personal point of view, we loved realising this project. It tremendously improved our skills in:

- Dealing with real enterprises
- Researching
- Prototyping
- Working with lab equipment
- Soldering
- Testing and measuring

But above all, TROUBLE SOLVING.

We are sure this experience made us better engineers.

**Thank you MAGCLOUD.**

## CHAPTER 9. REFERENCES

Pallás Areny, Ramón. *Analog signal processing*. New York [etc.] John Wiley & Sons, cop. 1999.-- XV, 586 p.

Pallás Areny, Ramón. *Sensors and signal conditioning*; 2<sup>nd</sup> ed. New York [etc.]: John Wiley & Sons, cop. 2001.-- xiii, 587 p.

Universitat Politècnica de Catalunya. *Revistes* [On-line]: Various searches. Servei de Biblioteques i Documentació. Available in  
<<http://biblioteca.upc.es/revistes/inici.asp>>

Reinhard Bischoff, Jonas Meyer and Glauco Feltrin. *Wireless Sensor Network Platforms*. New York [etc]. Chapter 69.

Various authors. *WSN* [On-line]: Wireless Sensor Network [Query: June, 2011] Available in < [http://en.wikipedia.org/wiki/Wireless\\_sensor\\_network/](http://en.wikipedia.org/wiki/Wireless_sensor_network/)>, <[http://wsn.oversigma.com/wiki/index.php?title=WSN\\_Platforms/](http://wsn.oversigma.com/wiki/index.php?title=WSN_Platforms/)>

Howitt, I., Gutierrez, J.A. *IEEE 802.15.4 low rate - wireless personal area network coexistence issues*. Charlotte, NC, USA

Various authors. *IEEE 802.15.4* [On-line]: IEEE 802.15.4 [Query: June, 2011] Available in <[http://en.wikipedia.org/wiki/IEEE\\_802.15.4/](http://en.wikipedia.org/wiki/IEEE_802.15.4/)>

Sinem Coleri Ergen. *ZigBee/IEEE 802.15.4 Summary*; California, September 10, 2004

Various authors. *ZigBee* [On-line]: ZigBee [Query: June, 2011] Available in <<http://en.wikipedia.org/wiki/ZigBee/>>

Various authors. *6LoWPAN* [On-line]: The Internet of things [Query: June, 2011] Available in <<http://en.wikipedia.org/wiki/6LoWPAN/>>, <<http://www.cs.berkeley.edu/~jwhui/6lowpan.html/>>

Various authors. *TinyOS* [On-line]: TinyOS [Query: June, 2011] Available in <<http://www.tinyos.net/>>, <[http://docs.tinyos.net/tinywiki/index.php/Main\\_Page/](http://docs.tinyos.net/tinywiki/index.php/Main_Page/)>

Various authors. *Installing TinyOS* [On-line]: GCC for TinyOS [Query: June, 2011] Available in <<http://codefat.com/tag/gcc-for-tinyos/>>

Various authors. *Contiki a lightweight and flexible operating system for tiny networked sensors*. Washington, DC, 2004; pp. 455–462.

Various authors. *MANTIS: system support for multimodal networks of insitu sensors*. *Proceedings*. New York, 2003; pp. 50–59.

Various authors. *SOS: a dynamic operating system for sensor nodes*. ACM Press: New York, 2005; pp. 163–176.

Various authors. *SNM* [On-line]: The Sensor Network Museum [Query: June, 2011] Available in < <http://www.snm.ethz.ch/Main/HomePage/> >

Various authors. *BSN* [On-line]: Body Sensor Networks [Query: June, 2011] Available in <<http://ubimon.doc.ic.ac.uk/bsn/m206.html> />

Sentilla. *Moteiv* [On-line]: Tmote Sky Documentation [Query: June, 2011] Available in  
< <http://www.sentilla.com/moteiv-transition.html/> >

Providers. [On-line]: Providers [Query: June, 2011] Available in  
<<http://farnell.com/>>, <<http://de.mouser.com/>>, <<http://www.digikey.com/>>

Various authors. *Linux Basis* [On-line]: Linux Distributions [Query: June, 2011] Available in <<http://www.linuxbasis.org/linux-distributions.html/>>

Easy Sen. [On-line]: Smart Sensing Solutions [Query: June, 2011] Available in  
<<http://www.easysen.com/>>

Various authors. *Berkeley Webs* [On-line]: Blip [Query: June, 2011] Available in  
<<http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip/>>

Various authors. *Berkeley Webs* [On-line]: Epic: An Open Mote Platform for Application-Driven Design [Query: June, 2011] Available in  
<<http://www.cs.berkeley.edu/~prabal/projects/epic/>>

Various authors. *Monitoring Tools* [On-line]: Comparison of network monitoring systems [Query: June, 2011] Available in <  
[http://en.wikipedia.org/wiki/Comparison\\_of\\_network\\_monitoring\\_systems/](http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems/)>

Blackburn, James A. *Modern instrumentation for scientists and engineers*. New York; Barcelona [etc.]: Springer, 2001.-- XV, 319 p.

Fowler, Kim R. *Electronic instrument design: architecting for the life cycle*. New York [etc.] : Oxford University Press, 1996.-- xvii, 552 p.

Various authors. *Sensors, about*. [On-line]: Inside Sensors. Questex U.S. Technology Group. Available in <<http://www.sensorsmag.com/sensors/>>

De Pablo Escolà, Ana. *Development of a wireless sensor network with 6LoWPAN support*. EETAC Masther Thesis, July 2009.

Jose Lopez Lopez and Sergio Mena Doce. *Desarrollo de un demostrador para evaluar técnicas Cross-Layer en sistemas de comunicaciones inalámbricos*. EETAC TFC, March 2008.

Narciso Cuartero Moya and Sergio Quintana Alcaraz. *Monitorización Remota de una red 1-Wire*. EETAC TFC, February 2008.



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEXES

**TITLE: THE MAGCLOUD WSN**

**MASTER DEGREE: Master in Science in Telecommunication Engineering & Management**

**AUTHORS: Narciso Cuartero Moya  
Sergio Quintana Alcaraz**

**DIRECTOR: José Polo Cantero  
2nd DIRECTOR: Jorge Eduardo Higuera Portilla**

**DATE: July, 4th 2011**



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEX A

**TITLE:** PCB Suppliers Conversation

**MASTER DEGREE:** Master in Science in Telecommunication Engineering  
& Management

**AUTHORS:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

**DIRECTOR:** José Polo Cantero  
**2nd DIRECTOR:** Jorge Eduardo Higuera Portilla

**DATE:** July, 4th 2011

## ANNEXE A. PCB SUPPLIERS CONVERSATION

<http://www.accent-systems.es/>



After filling their online form, they responded to our mail:

Jordi Casamada · Accent Systems jcasamada@accent-systems.com

Hola Sergio,

Me puedes hacer llegar los gerbers, la lista de componentes y toda la información que tengas sobre la placa?

En breve te haremos llegar una oferta.

Muchas gracias,  
Jordi.

We reply:

Hola, buenos días, le adjunto los archivos gerbers y un pdf con la descripción.

Un Saludo;

Sergio Quintana

They reply:

Hola Sergio,

Te llamé ayer al móvil y te dejé un mensaje en el contestador.

Tengo una duda sobre tu placa. Esta placa es una placa que se puede comprar, ya que se está vendiendo. Por que motivo quieres fabricarla tu a tan pocas cantidades? no te saldría mejor comprarla a Texas?

Espero tu respuesta,

Muchas gracias,  
Jordi



We reply:

Hola Jordi, ya vi tu mensaje ayer y esta tarde te iba a llamar en cuanto me reuniera con mi compañero.

Somos 2 estudiantes de Ingeniería Superior de Telecomunicaciones de la UPC que estamos realizando el PFC sobre estos nodos; el proyecto se basa en hacer tanto el hardware como el software con distintas mejoras respecto a la versión comercial que se vende actualmente, sobretodo en el caso del software. Es por ello que necesitamos producir estas placas en una pequeña escala y si todo va correcto, ampliaríamos la tirada y llenaríamos todo el campus de la universidad con estos nodos.

Con el material que te adjunte, podrías producir las placas, o necesitas algún archivo más? En caso afirmativo, cuanto coste supondría?

Para cualquier otra duda o lo que sea, puedes contactar via movil o mail.

Esperamos tu respuesta,  
Muchas Gracias;

Sergio

After a week, we try again:

Hola Jordi, buenos días, quería saber como llevan el presupuesto de las placas que solicite o si ha surgido algún problema.

Un Saludo;

Sergio Quintana

They have not responded yet.

<http://www.cipsacircuits.com/>



After filing their form, they asked for an initial fee too expensive.

Estimado Sr.Sergio,

Para poder pasarle la oferta con el formato standard necesitaría sus datos para darle de alta como posible cliente, de todas formas se la paso via email

Circuito 4 capas Ref-Pack

25 unidades-- 8.30 eur/unidad

50 unidades- 4.89 eur/unidad

100 unidades- 2.82 eur/unidad

Gtos iniciales :420 euros ( solo se pagan la primera vez, salvo que no se modifique el diseño deñ circuito)

Saludos Cordiales

P.D.- Le dejo mi número de móvil por si necesita preguntarme alguna cosa más 629 632 643

<http://www.diselec.com/>

We filed the online form but they have not respondet yet.

<http://www.folegax.com/>



After filling their online form, they responded to our mail:

Buenos días Sergio  
Gracias por darnos la oportunidad de poder ofertar la fabricación del circuito impreso  
Para ello precisamos nos envíes los archivos en formato gerbers  
Gracias  
Gonzalo Navarro

We reply:

Hola, buenos días, le adjunto los archivos gerbers y un pdf con la descripción.  
  
Un Saludo;  
  
Sergio Quintana

They reply:

Hola, buenos días, quería saber como llevan el presupuesto de las placas que solicite.  
  
Un Saludo;  
  
Sergio Quintana

They have not responded yet.

<http://www.lpkf.es>



They do not do multiplayer PCBs.

Buenos días,

Siento comunicarte que no podemos realizar circuitos multicapa.

Un saludo.

## **FINAL CHOSEN ENTERPRISE**

<http://www.2cisa.com/>





Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEX B

**TITLE:** 2Cisa Budget

**MASTER DEGREE:** Master in Science in Telecommunication Engineering  
& Management

**AUTHORS:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

**DIRECTOR:** José Polo Cantero  
**2nd DIRECTOR:** Jorge Eduardo Higuera Portilla

**DATE:** July, 4th 2011



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEX C

**TITLE:** Solar submodules

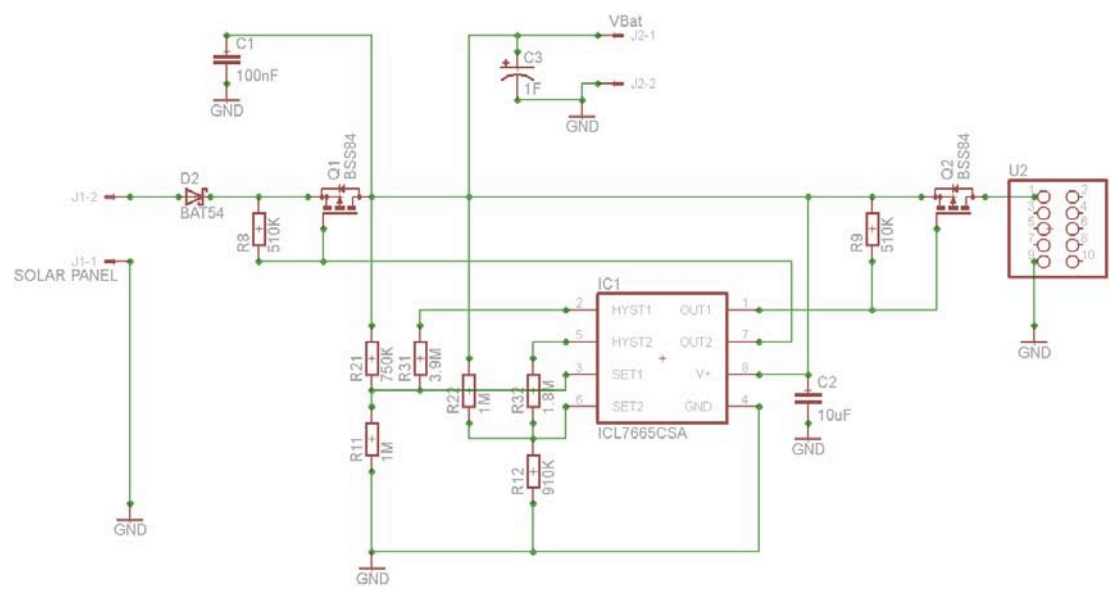
**MASTER DEGREE:** Master in Science in Telecommunication Engineering  
& Management

**AUTHORS:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

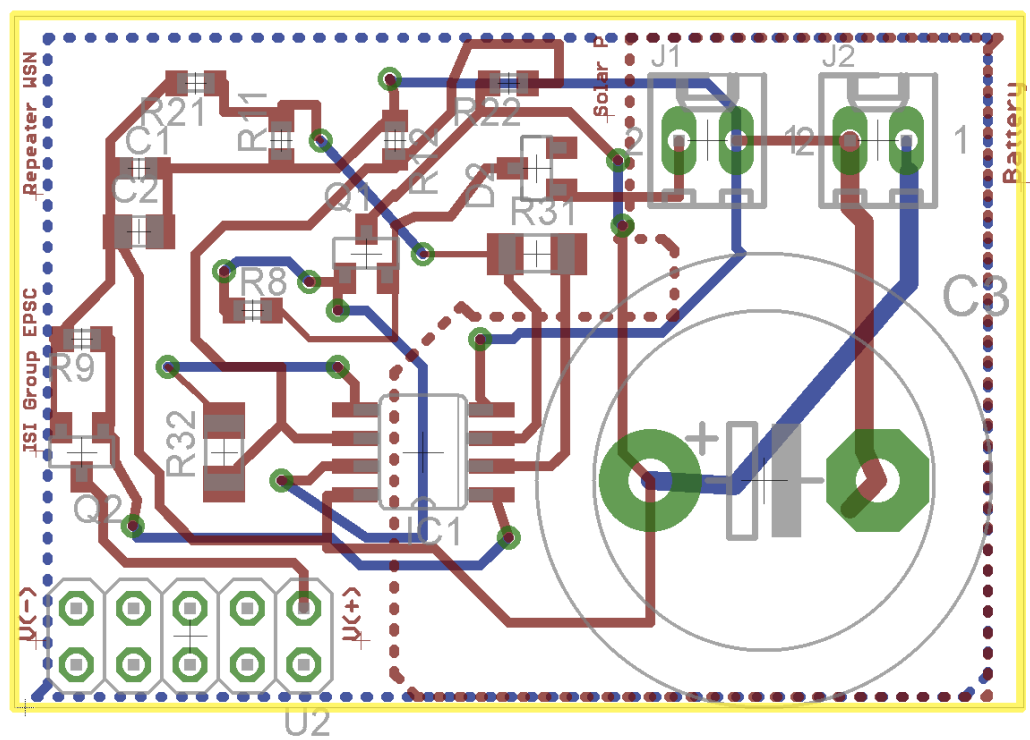
**DIRECTOR:** José Polo Cantero  
**2nd DIRECTOR:** Jorge Eduardo Higuera Portilla

**DATE:** July, 4th 2011

ANNEXE C. SOLAR SUBMODULE

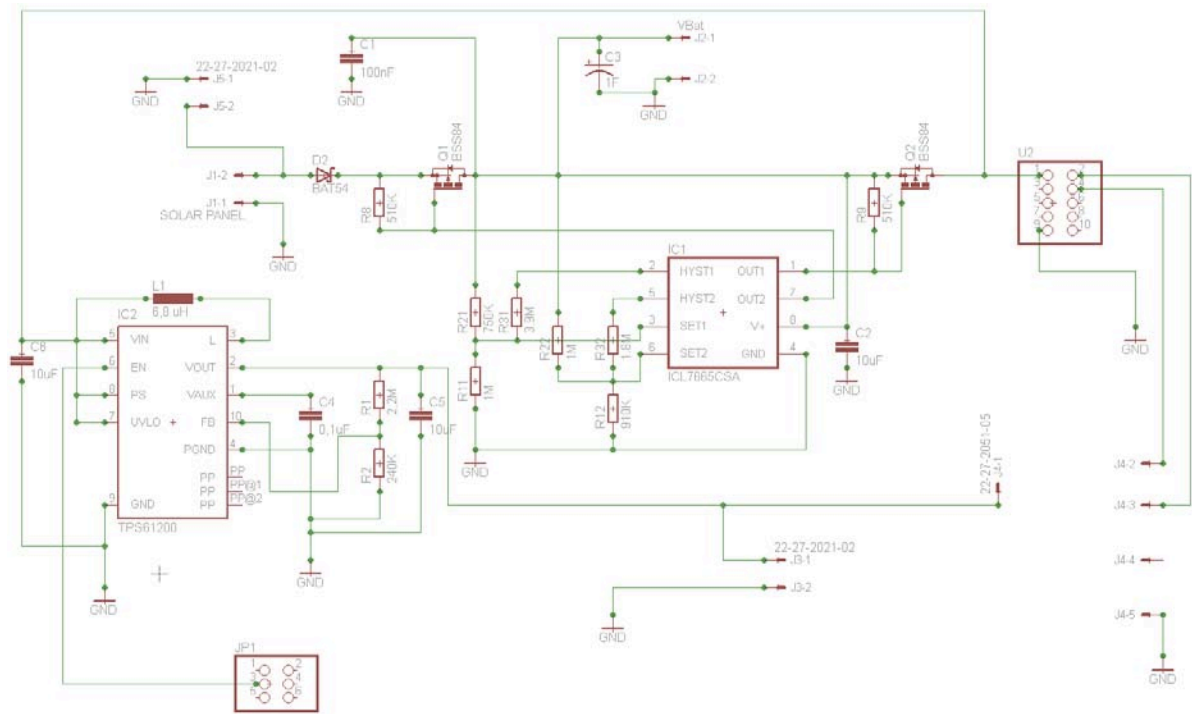


Schematics 1

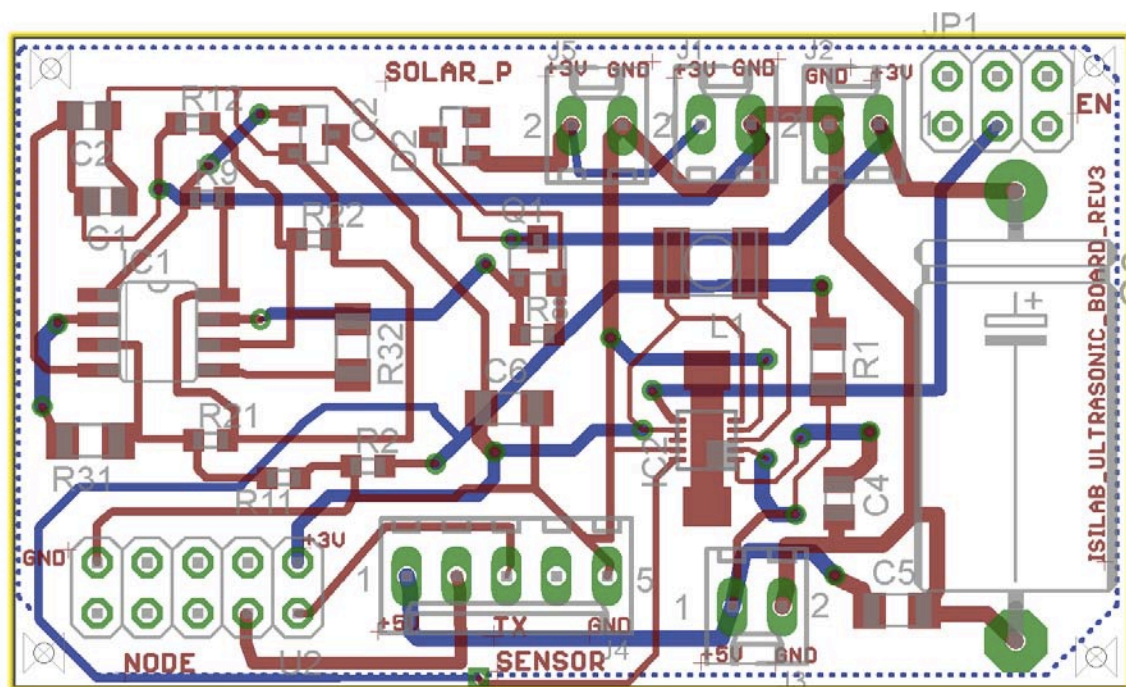


Board 1

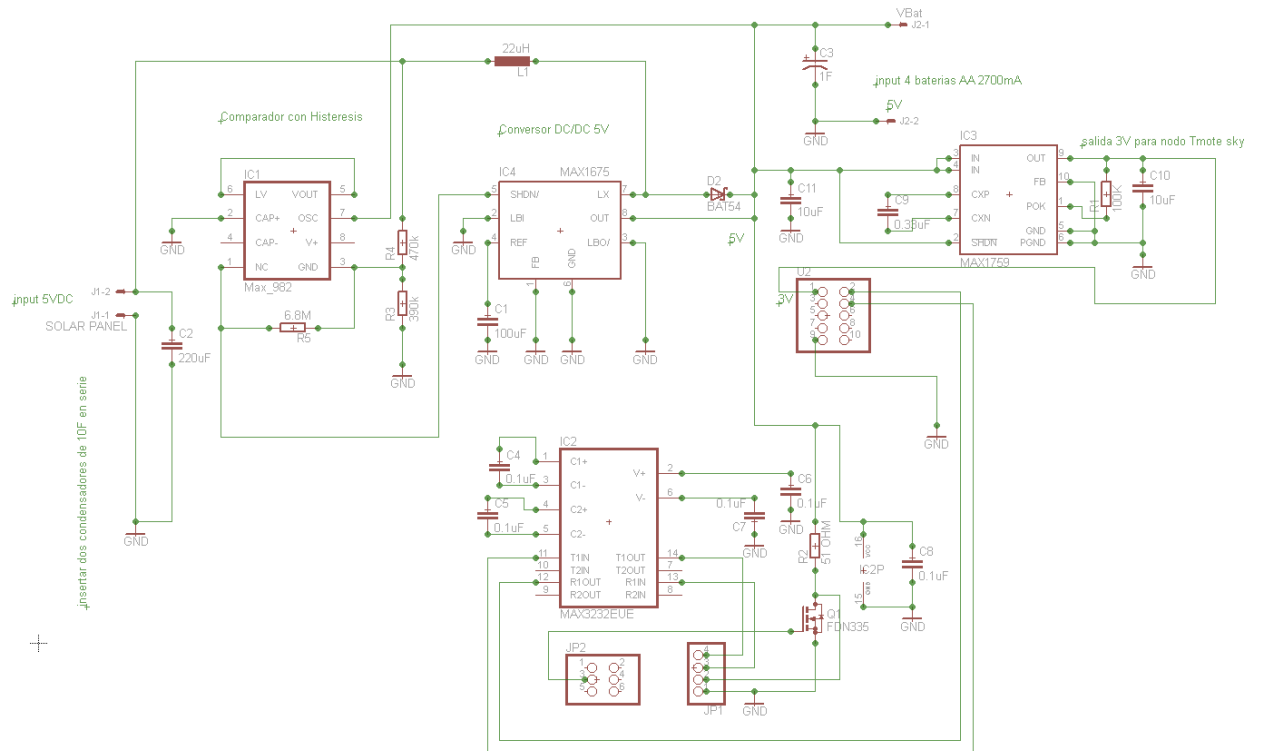




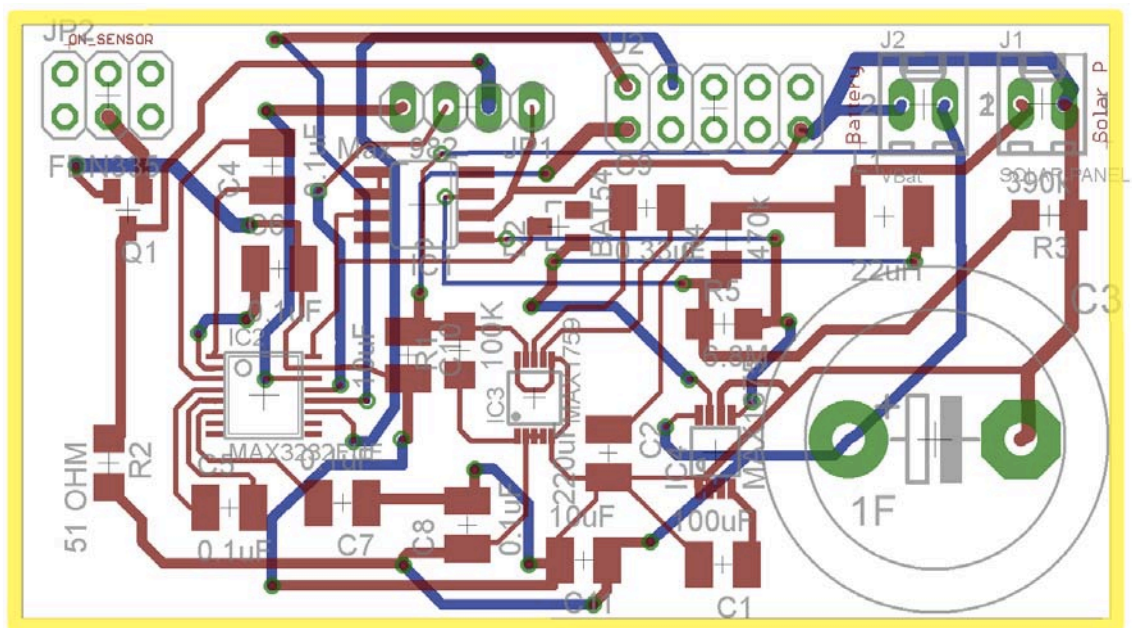
**Schematics 2**



**Board 2**



**Schematics 3**



**Board 3**



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEX D

**TITLE:** Installation of blip under Ubuntu 10.04

**MASTER DEGREE:** Master in Science in Telecommunication Engineering  
& Management

**AUTHORS:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

**DIRECTOR:** José Polo Cantero  
**2nd DIRECTOR:** Jorge Eduardo Higuera Portilla

**DATE:** July, 4th 2011

# Installation of blip under Ubuntu 10.04

---

## About blip

BLIP, the Berkeley Low-power IP stack, is an implementation in TinyOS of a number of IP-based protocols. Using blip/TinyOS, you will be able to form multi-hop IP networks consisting of different motes communicating over shared protocols.



## About TinyOS 2.1.1

The 2.1.1 (April 2010) version of TinyOS includes very important features:

- Support for 6LoWPAN, an IPv6 networking layer within the mote network
- Support for simple, uniform low-power networking across many protocols



## 6LoWPAN and Ipv6 Basics

The Maximum Transfer Unit (MTU) refers to the largest payload which may be sent in a link-layer data frame. Most blip node routers use the IEEE 802.15.4 physical layer, which limits the MTU to around 100 bytes. To provide the larger payloads to upper-layer protocols, blip implements 6LoWPAN layer 2.5 fragmentation. As a result, the maximum size IP-layer datagram is 1280 octets.

## Prerequisites

### Node module

I used a Crossbow Telos Rev. B generic device called FTDI (Future Technology Devices International).



Figure 1: Crossbow Telos Rev. B mote

## Installation of TinyOS

Install the latest TinyOS version, the version 2.1.1 was used for this test.

In a terminal, you have to add a new repository:

```
sudo gedit /etc/apt/sources.list
```

Add this line by replacing the <distribution> field:

```
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu <distribution> main
```

<distribution> could be : gutsy, karmic, lucid...

Then, reload the repository:

```
sudo aptitude update
```

Install the latest version:

```
sudo aptitude install tinyos-2.1.1
```

Replace 2.1.1 by your newer version.

Once the installation is finished, set the shell environment. Type:

```
gedit ~/.bashrc
```

And add following lines at bottom:

```
#TinyOS  
export TOSROOT=/opt/tinyos-2.1.1  
export TOSDIR=$TOSROOT/tos  
export CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar  
export MAKERULES=$TOSROOT/support/make/Makerules  
export PATH=/opt/msp430/bin:$PATH
```

Replace 2.1.1 by your newer version.

It's OK. TinyOS is set up.

## Installation of VMware and Xubuntos

Because Ubuntu 10.04 have some problems to correctly compile code for Crossbow Telos B node, you have to virtualize a compatible Ubuntu. I choose Xubuntos because there already are dependencies.

Strangely, the compilation of code for other nodes such as MicaZ is working.

VMware is a virtualization tools. You can install it easily. Go on [www.wmware.com](http://www.wmware.com) and download the latest version of VMware Player for Linux.

To install it, use these commands in the VMware's folder:

```
chmod 777 Vmware-Player-*  
./Vmware-Player-*
```

Follow the instructions.

Download Xubuntos.

Extract it.

Virtualize it with VMware.

Install (again) the latest TinyOS on your Xubuntos.

## First test

Try to compile and to implement a little program. Type:

```
cd $TOSROOT/apps/Blink  
make telosb
```

telosb means Crossbow Telos rev. B, you have to change it if you have a different node.

Plug in your node. Type:

```
motelist
```

With this command, you can see your node and its path (/dev/ttyUSB0 for instance)

Type:

```
make telosb install bsl,ttyUSB0
```

If it works, your node should blinking.

## Install the IP Base Station node

Plug in your node. Type:

```
cd $TOSROOT/apps/IPBaseStation  
make telosb blip install
```

## Install few UDPEcho nodes

```
cd $TOSROOT/apps/UDPEcho  
make telosb blip install.ID
```

ID is the end of the Ipv6 address of the node. For instance, if the prefix specified in the configuration file was fec0:: and the node id as 1, the mote would have address fec0::1. It's a hexadecimal conversion, if the ID is 101, the mote address should be fec0::65.

## Tunnel Driver Installation

We need to build the routing driver. It has one dependency, the TinyOS serial library. To build it, do the following:

```
sudo aptitude install automake autoconf  
cd $TOSROOT/support/sdk/c/sf  
./bootstrap  
./configure  
make
```

Next, repeat the same steps for the driver:

```
cd $TOSROOT/support/sdk/c/blip  
./bootstrap  
./configure  
make
```

Plug in your IP Base Station node.

In this directory, you can begin running the driver using the following:

```
sudo driver/ip-driver /dev/ttyUSB0 telosb
```

In my case, the driver was not very stable. It failed many times before succeeded to start.

```
lavren@lavren-laptop:/opt/tinyos-2.1.1/support/sdk/c/blip/driver$ sudo ./ip-driver -c  
../serial_tun.conf /dev/ttyUSB0 telosb  
2010-07-02T21:58:43.086CST: INFO: Read config from '../serial_tun.conf'
```

```
2010-07-02T21:58:43.086CST: INFO: Using channel 15
2010-07-02T21:58:43.086CST: INFO: Retries: 5
2010-07-02T21:58:43.086CST: INFO: telnet console server running on port 6106
2010-07-02T21:58:43.089CST: INFO: created tun device: tun0
2010-07-02T21:58:48.191CST: FATAL: configuring interface failed! aborting!
```

It failed!

```
lavren@lavren-laptop:/opt/tinyos-2.1.1/support/sdk/c/blip/driver$ sudo ./ip-driver -c
../serial_tun.conf /dev/ttyUSB0 telosb
2010-07-02T21:59:03.288CST: INFO: Read config from '../serial_tun.conf'
2010-07-02T21:59:03.288CST: INFO: Using channel 15
2010-07-02T21:59:03.288CST: INFO: Retries: 5
2010-07-02T21:59:03.288CST: INFO: telnet console server running on port 6106
2010-07-02T21:59:03.290CST: INFO: created tun device: tun0
2010-07-02T21:59:03.564CST: INFO: interface device successfully initialized
2010-07-02T21:59:03.564CST: INFO: starting radvd on device tun0
2010-07-02T21:59:04.291CST: INFO: Starting to proxy for 0x1
2010-07-02T21:59:23.951CST: INFO: Starting to proxy for 0x2
2010-07-02T21:59:52.731CST: INFO: driver shutting down
```

It succeeded!

## Executing a Network test

After installing the base station, starting the tunnel driver and installing at least one sensor node with a test application, the sensor network can be tested.

Power on your UDPEcho nodes and try to ping them. Example:

```
lavren@lavren-laptop:~$ ping6 fec0::1
PING fec0::1(fec0::1) 56 data bytes
64 bytes from fe80::1: icmp_seq=7 ttl=240 time=443 ms
64 bytes from fe80::1: icmp_seq=8 ttl=240 time=104 ms
64 bytes from fe80::1: icmp_seq=9 ttl=240 time=91.8 ms
64 bytes from fe80::1: icmp_seq=10 ttl=240 time=114 ms
64 bytes from fec0::1: icmp_seq=11 ttl=65 time=173 ms
64 bytes from fec0::1: icmp_seq=12 ttl=65 time=119 ms
```

You can also communicate by UDP on port 2000 using this command:

```
lavren@lavren-laptop:~$ nc6 -u fec0::2 2000
help
sdsh-0.9      builtins: [help, echo, ping6, uptime, ident, led1, led2]
uptime
up 81 seconds
```



I add two more actions named “led1” and “led2” in my UDPEcho program to switch off or switch on these 2 LEDs.

## A little more

Modifying the file `$TOSROOT/support/sdk/c/blip/serial_tun.conf` allows someone to use its own networking configuration.

In the case the message **sendmsg: Operation not permitted** is shown and the tunnel driver is not correctly started, IPv6 is disabled in the firewall and is only allowed to be used in combination with the loopback interface. This configuration is currently standard in Ubuntu. To enable the IPv6 traffic under Ubuntu, the line **IPv6=yes** has to be added to the file `/etc/default/ufw`. After restarting, the tunnel driver should work fine.



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANNEX E

**TITLE:** Tmote Sky datasheet

**MASTER DEGREE:** Master in Science in Telecommunication Engineering  
& Management

**AUTHORS:** Narciso Cuartero Moya  
Sergio Quintana Alcaraz

**DIRECTOR:** José Polo Cantero  
**2nd DIRECTOR:** Jorge Eduardo Higuera Portilla

**DATE:** July, 4th 2011

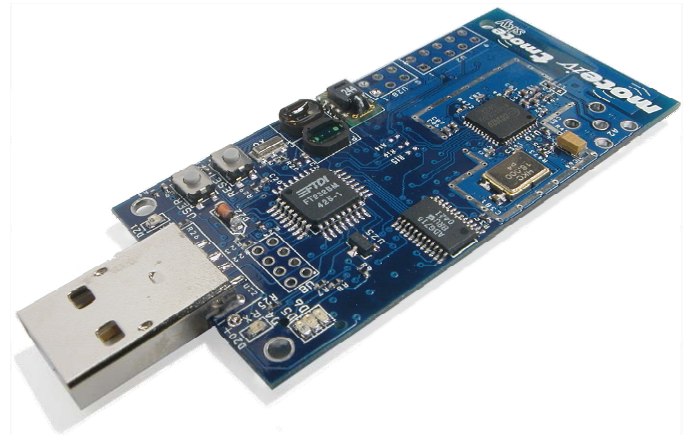


## **Ultra low power IEEE 802.15.4 compliant wireless sensor module**

Humidity, Light, and Temperature sensors with USB

### **Product Description**

Tmote Sky is an ultra low power wireless module for use in sensor networks, monitoring applications, and rapid application prototyping. Tmote Sky leverages industry standards like USB and IEEE 802.15.4 to interoperate seamlessly with other devices. By using industry standards, integrating humidity, temperature, and light sensors, and providing flexible interconnection with peripherals, Tmote Sky enables a wide range of mesh network applications. Tmote



Sky is a drop-in replacement for Moteiv's successful Telos design. Tmote Sky includes increased performance, functionality, and expansion. With TinyOS support out-of-the-box, Tmote Sky leverages emerging wireless protocols and the open source software movement. Tmote Sky is part of a line of modules featuring on-board sensors to increase robustness while decreasing cost and package size.

### **Key Features**

- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver
- Interoperability with other IEEE 802.15.4 devices
- 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
- Integrated ADC, DAC, Supply Voltage Supervisor, and DMA Controller
- Integrated onboard antenna with 50m range indoors / 125m range outdoors
- Integrated Humidity, Temperature, and Light sensors
- Ultra low current consumption
- Fast wakeup from sleep (<6μs)
- Hardware link-layer encryption and authentication
- Programming and data collection via USB
- 16-pin expansion support and optional SMA antenna connector
- TinyOS support : mesh networking and communication implementation
- Complies with FCC Part 15 and Industry Canada regulations
- Environmentally friendly – complies with RoHS regulations

## Table of Contents

|  |    |
|--|----|
| Product Description.....                         | 1  |
| Key Features.....                                | 1  |
| Table of Contents.....                           | 2  |
| Module Description .....                         | 3  |
| Power .....                                      | 4  |
| Typical Operating Conditions .....               | 4  |
| Mechanical Characteristics .....                 | 5  |
| Block Diagram.....                               | 6  |
| Schematic .....                                  | 7  |
| Microprocessor .....                             | 9  |
| Description .....                                | 9  |
| Typical Operating Conditions .....               | 9  |
| PC Communication .....                           | 9  |
| Programming.....                                 | 10 |
| Block Diagram.....                               | 12 |
| Radio.....                                       | 13 |
| Description .....                                | 13 |
| Typical Operating Conditions .....               | 14 |
| Measured Output Power .....                      | 14 |
| Antenna.....                                     | 15 |
| Internal Antenna without Battery Pack.....       | 15 |
| Internal Antenna with Battery Pack.....          | 15 |
| Radiation Pattern .....                          | 16 |
| External Flash .....                             | 17 |
| Typical Operating Conditions .....               | 17 |
| Flash Hardware Write Protection .....            | 18 |
| Sensors.....                                     | 19 |
| Humidity/Temperature Sensor .....                | 19 |
| Light Sensors .....                              | 20 |
| Expansion Connector.....                         | 21 |
| Internal Temperature and Voltage Monitoring..... | 23 |
| Agency Certification .....                       | 24 |
| FCC Certification.....                           | 24 |
| OEM Labeling requirement .....                   | 24 |
| FCC Notices.....                                 | 25 |
| General Information .....                        | 25 |
| Document History.....                            | 26 |
| Product Status Definitions.....                  | 26 |
| Disclaimer .....                                 | 27 |
| Address Information .....                        | 28 |
| Headquarters .....                               | 28 |

## Module Description

The Tmote Sky module is a low power “mote” with integrated sensors, radio, antenna, microcontroller, and programming capabilities.

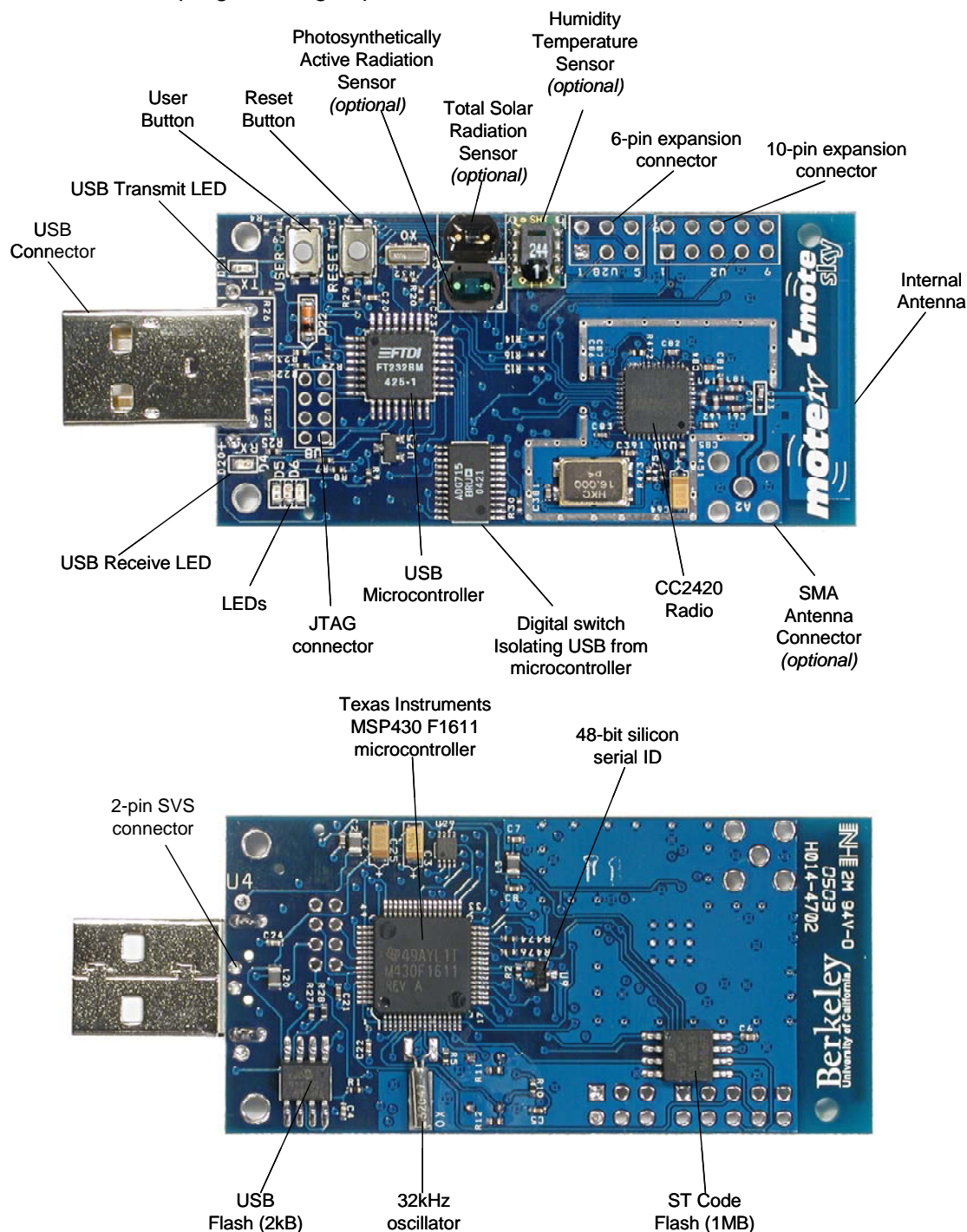


Figure 1 : Front and Back of the Tmote Sky module

## Power

Tmote Sky may be powered by two AA batteries. The module was designed to fit the two AA battery form factor. AA cells may be used in the operating range of 2.1 to 3.6V DC, however the voltage must be at least 2.7V when programming the microcontroller flash or external flash.

If the Tmote Sky module is plugged into the USB port for programming or communication, it will receive power from the host computer. The mote operating voltage when attached to USB is 3V. If Tmote will always be attached to a USB port, no battery pack is necessary.

The 16-pin expansion connector (described in the Section on page 17) can provide power to the module. Any of the battery terminal connections may also provide power to the module. At no point should the input voltage exceed 3.6V—doing so may damage the microcontroller, radio, or other components.

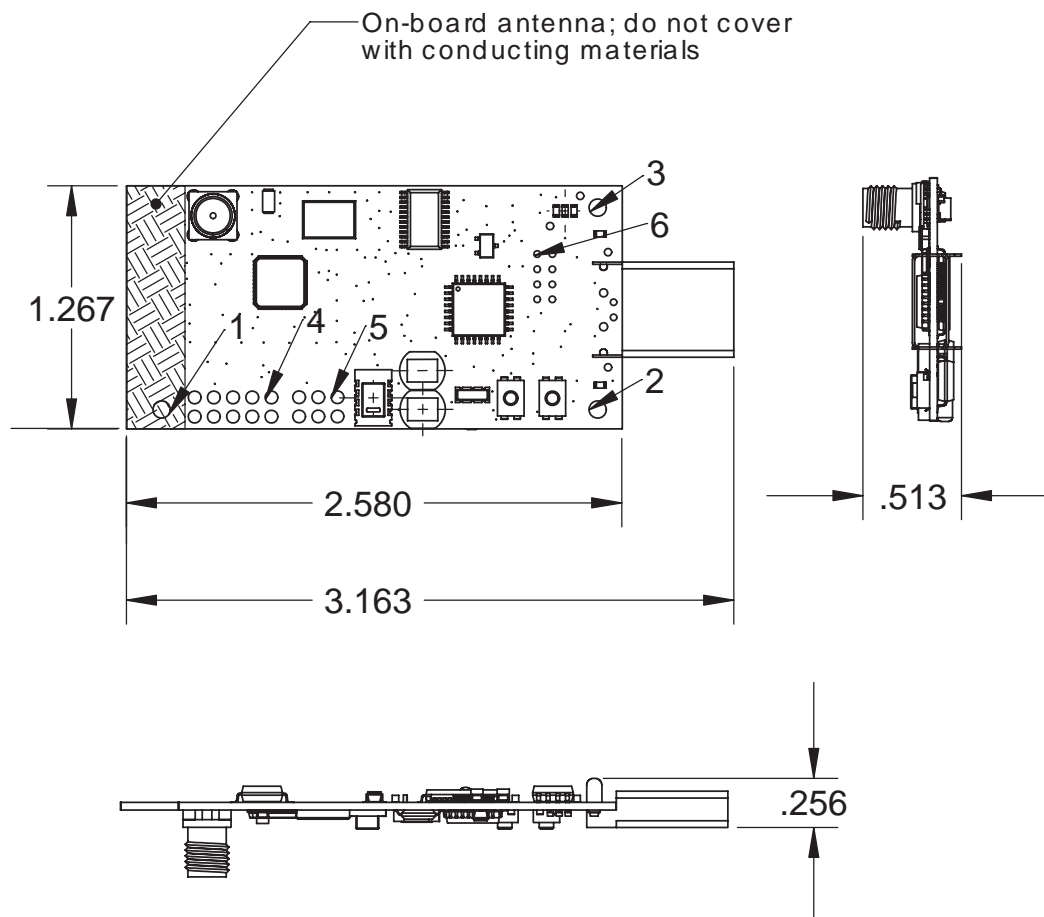
## Typical Operating Conditions

|  | MIN | NOM  | MAX  | UNIT |
|--|-----|------|------|------|
| Supply voltage                                 | 2.1 |      | 3.6  | V    |
| Supply voltage during flash memory programming | 2.7 |      | 3.6  | V    |
| Operating free air temperature                 | -40 |      | 85   | °C   |
| Current Consumption: MCU on, Radio RX          |     | 21.8 | 23   | mA   |
| Current Consumption: MCU on, Radio TX          |     | 19.5 | 21   | mA   |
| Current Consumption: MCU on, Radio off         |     | 1800 | 2400 | μA   |
| Current Consumption: MCU idle, Radio off       |     | 54.5 | 1200 | μA   |
| Current Consumption: MCU standby               |     | 5.1  | 21.0 | μA   |



**Caution!** ESD sensitive device.  
Precaution should be used when handling  
the device in order to prevent permanent  
damage.

## Mechanical Characteristics



| Tag | X     | Y     | Size    | Notes                                    |
|-----|-------|-------|---------|--|
| 1   | 0.183 | 0.099 | Ø 0.090 | Mounting hole, do not use metal fixture  |
| 2   | 2.454 | 0.099 | Ø 0.090 | Mounting hole                            |
| 3   | 2.454 | 1.151 | Ø 0.090 | Mounting hole                            |
| 4   | 0.755 | 0.162 | Ø 0.066 | Pin 1 of 10-pin 0.1in rect IDC connector |
| 5   | 1.099 | 0.163 | Ø 0.066 | Pin 1 of 6-pin 0.1in rect IDC connector  |
| 6   | 2.139 | 0.909 | Ø 0.034 | Pin 1 of 8-pin 2mm rect JTAG connector   |

**Figure 2 : Physical dimensions of Tmote Sky.**  
All units are in inches unless otherwise noted.

|   | MIN  | NOM  | MAX  | UNIT |
|---|------|------|------|------|
| Width   | 1.24 | 1.26 | 1.29 | in   |
| Length  | 2.55 | 2.58 | 2.60 | in   |
| Height (without battery pack and SMA antenna) | 0.24 | 0.26 | 0.27 | in   |

## Block Diagram

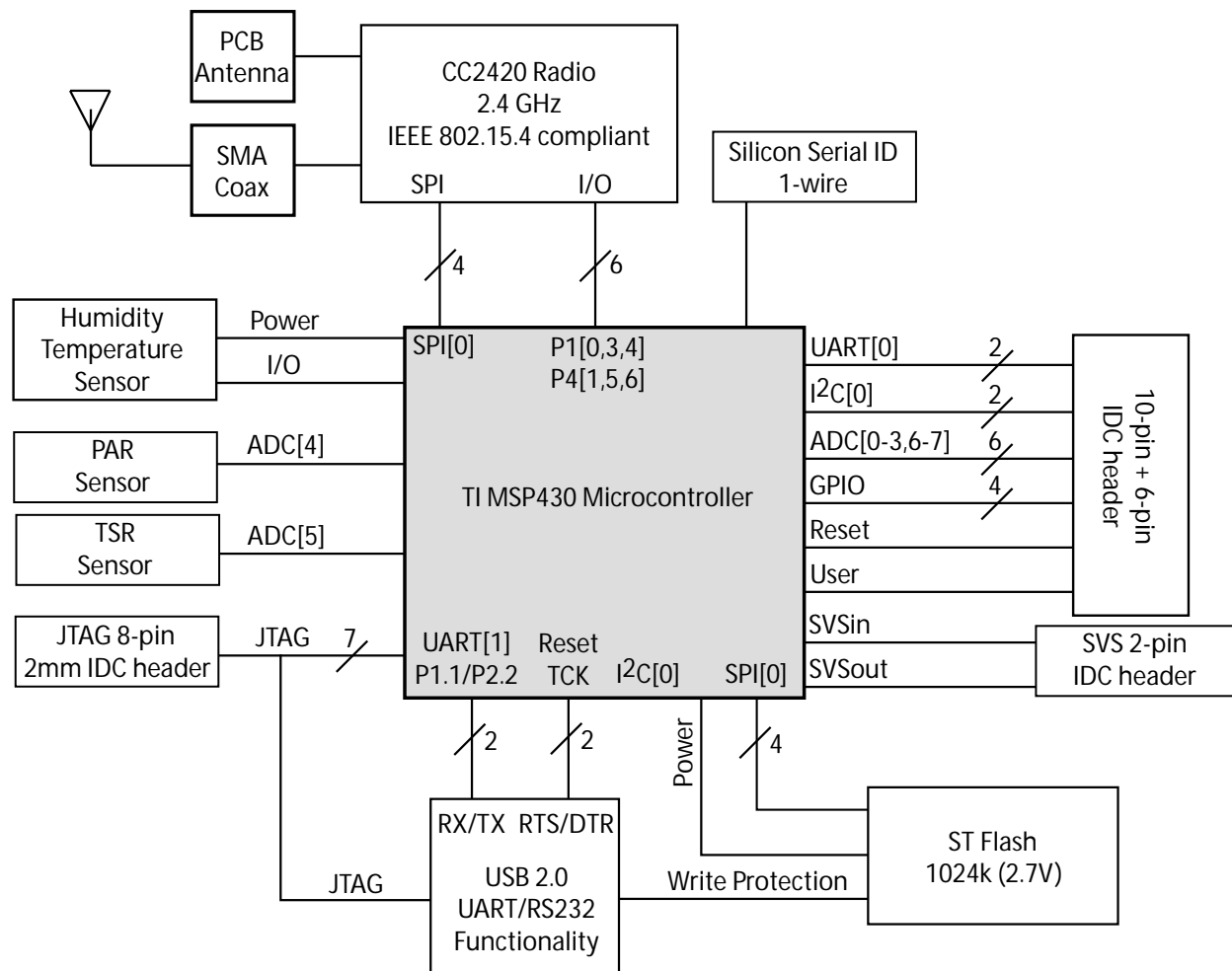
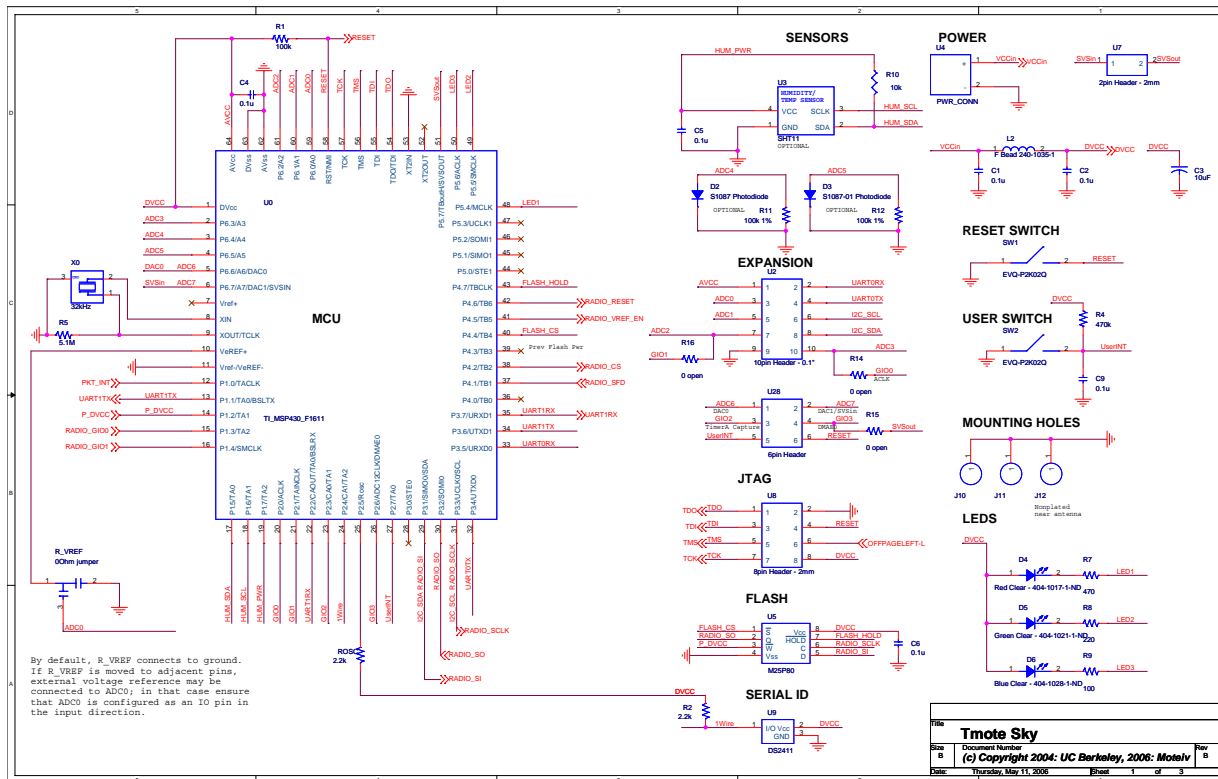


Figure 3 : Functional Block Diagram of the Tmote Sky module, its components, and buses



## Schematic



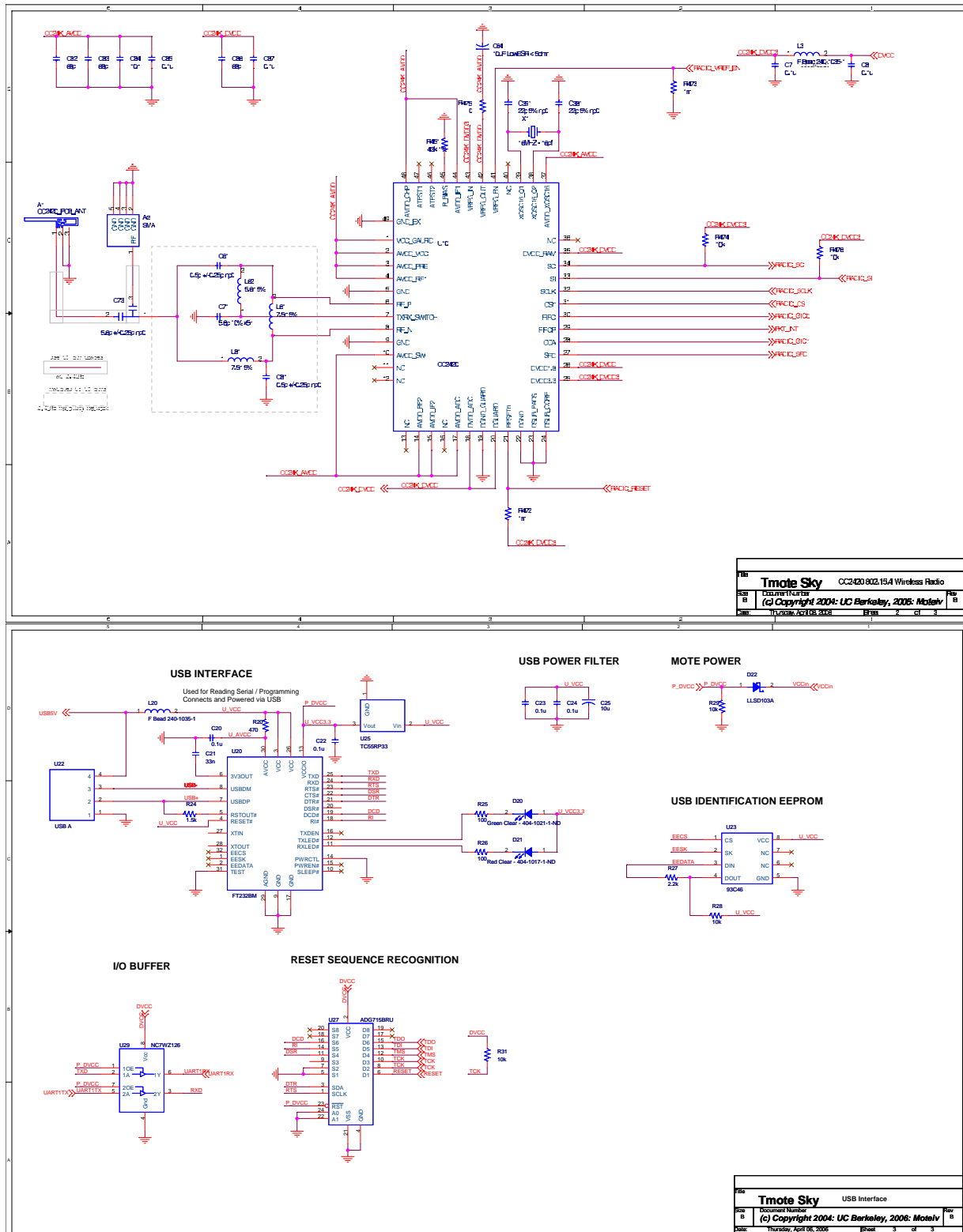


Figure 4 : Schematics for the Tmote Sky module

## Microprocessor

### Description

The low power operation of the Tmote Sky module is due to the ultra low power Texas Instruments MSP430 F1611 microcontroller featuring 10kB of RAM, 48kB of flash, and 128B of information storage. This 16-bit RISC processor features extremely low active and sleep current consumption that permits Tmote to run for years on a single pair of AA batteries. The MSP430 has an internal digitally controlled oscillator (DCO) that may operate up to 8MHz. The DCO may be turned on from sleep mode in 6 $\mu$ s, however 292ns is typical at room temperature. When the DCO is off, the MSP430 operates off an eternal 32768Hz watch crystal. Although the DCO frequency changes with voltage and temperature, it may be calibrated by using the 32kHz oscillator.

In addition to the DCO, the MSP430 has 8 external ADC ports and 8 internal ADC ports. The ADC internal ports may be used to read the internal thermistor or monitor the battery voltage. A variety of peripherals are available including SPI, UART, digital I/O ports, Watchdog timer, and Timers with capture and compare functionality. The F1611 also includes a 2-port 12-bit DAC module, Supply Voltage Supervisor, and 3-port DMA controller.

The features of the MSP430 F1611 are presented in detail in the Texas Instruments MSP430x1xx Family User's Guide available at <http://ti.com/msp430>.

### Typical Operating Conditions

|  | MIN | NOM    | MAX | UNIT    |
|--|-----|--------|-----|---------|
| Supply voltage during program execution          | 1.8 |        | 3.6 | V       |
| Supply voltage during flash memory programming   | 2.7 |        | 3.6 | V       |
| Operating free air temperature                   | -40 |        | 85  | °C      |
| Low frequency crystal frequency                  |     | 32.768 |     | kHz     |
| Active current at Vcc = 3V, 1MHz                 |     | 500    | 600 | $\mu$ A |
| Sleep current in LPM3 Vcc = 3V, 32.768kHz active |     | 2.6    | 3.0 | $\mu$ A |
| Wake up from LPM3 (low power mode)               |     |        | 6   | $\mu$ s |

### PC Communication

Tmote Sky uses a USB controller from FTDI to communicate with the host computer. In order to communicate with the mote, the FTDI drivers must be installed on the host. FTDI provides drivers for Windows, Linux, BSD, Macintosh, and Windows CE. These drivers are included on the Moteiv CD shipped with your order. Windows users will need the Virtual Com Port (VCP) drivers. They may also be downloaded from FTDI's website at: <http://www.ftdichip.com/>

Tmote Sky appears as a COM port in Windows' device manager (or as a device in /dev in Linux, OSX, and BSD). Multiple Tmote Sky motes may be connected to a single computer's USB ports at the same time. Each mote will receive a different COM port identifier. In the example below, one Tmote is connected and assigned COM6 "USB Serial Port".

An application may read from Tmote Sky by opening the COM port assigned to the Tmote Sky mote. Tmote communicates with the host PC through USART1 on the TI MSP430.

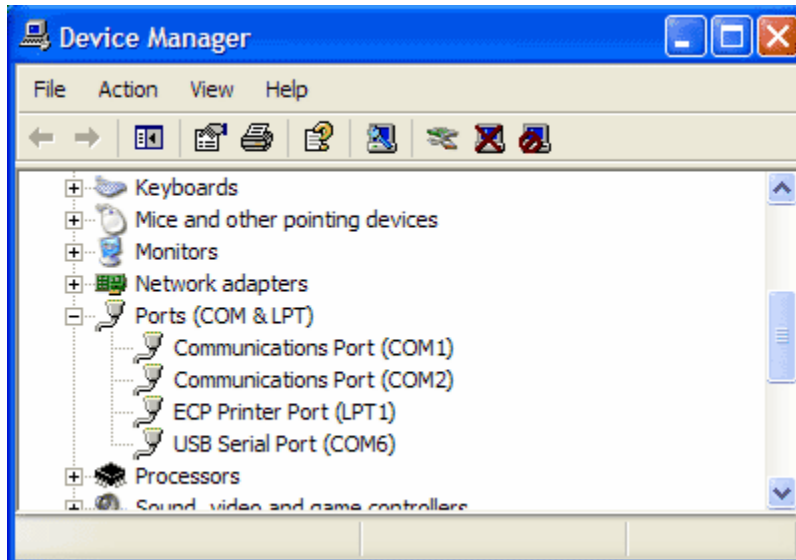


Figure 5 : Device Manager showing Tmote Sky installed as COM6

The `motelist` command line utility lists all of the Tmote Sky motes currently connected to a computer. This utility optionally lists previously connected motes that the system has cached. Invoke `motelist` with the `-h` option for more information.

```
> motelist
```

| Reference | CommPort | Description      |
|-----------|----------|------------------|
| M49WD0S6  | COM6     | Moteiv tmote sky |



**NOTE:** Tmote Sky uses an I<sup>2</sup>C digital switch to prevent unwanted conventional serial port signals from reaching the TI microcontroller. The I<sup>2</sup>C protocol must be implemented and sent over the RTS and DTR lines in order to obtain direct access between the Tmote Sky and USB controller. The UART lines do not use the I<sup>2</sup>C switch allowing direct communication (but not programming or JTAG) without additional software.

## Programming

The Tmote Sky module is programmed through the onboard USB connector. A modified version of the MSP430 Bootstrap Loader, `mbsp430-bsl`, programs the microcontroller's flash. Tmote has a unique hardware circuit that prevents the mote from spuriously resetting. This hardware circuit makes it necessary to have a special sequence sent to the module in order to program it.

By invoking `mbsp430-bsl`, verify you have the patched BSL by looking for the "telos" keyword. Version 1.39-telos-7 or later is required for Tmote Sky.

```
> mbsp430-bsl
```

```
MSP430 Bootstrap Loader Version: 1.39-telos-7
Use -h for help
```

To communicate with Tmote Sky, the MSP430 Bootstrap Loader requires a set of options to provide the proper signals to the microcontroller to initiate programming. For convenience, the options have been folded into a single Tmote flag:

```
--tmote
```

To program a Tmote Sky module on COM3 (or /dev/ttyUSB2 in Linux) with an application image named `app.ihex`, invoke the MSP430 Bootstrap loader with the following options.

```
> msp430-bsl --tmote -c 2 -r -e -I -p app.ihex
MSP430 Bootstrap Loader Version: 1.39-telos-7
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
2742 bytes programmed.
Reset device ...
```

If you are using TinyOS, it has support for programming Tmote Sky. After compiling your application, you may install it with the following command

```
> make tmote install.x bsl,n
```

Where `x` is the 16-bit address assigned to the mote and `n` is the COM port that Tmote Sky is currently using. Note that not including “`bsl`” or “`bsl,n`” will program automatically using the `bsl` to the first Telos mote found on the USB bus using the `motelist` command.

For more information about the options in the MSP430 Bootstrap loader, invoke `msp430-bsl` with the `-h` option to display the help information.

`Motelist` and `msp430-bsl` are available from Moteiv Corporation at <http://www.moteiv.com> in the “Support” section.



**NOTE:** `msp430-bsl` starts counting from 0, but COM ports in Windows start counting at 1. If Tmote Sky is connected to COM3 in Windows, you must program it using “`-c 2`” or “`bsl,2`” when invoking `msp430-bsl`. In Linux, Tmote Sky will appear as `/dev/ttyUSB2` and may be programmed using “`-c 2`” or “`bsl,2`”.

## Block Diagram

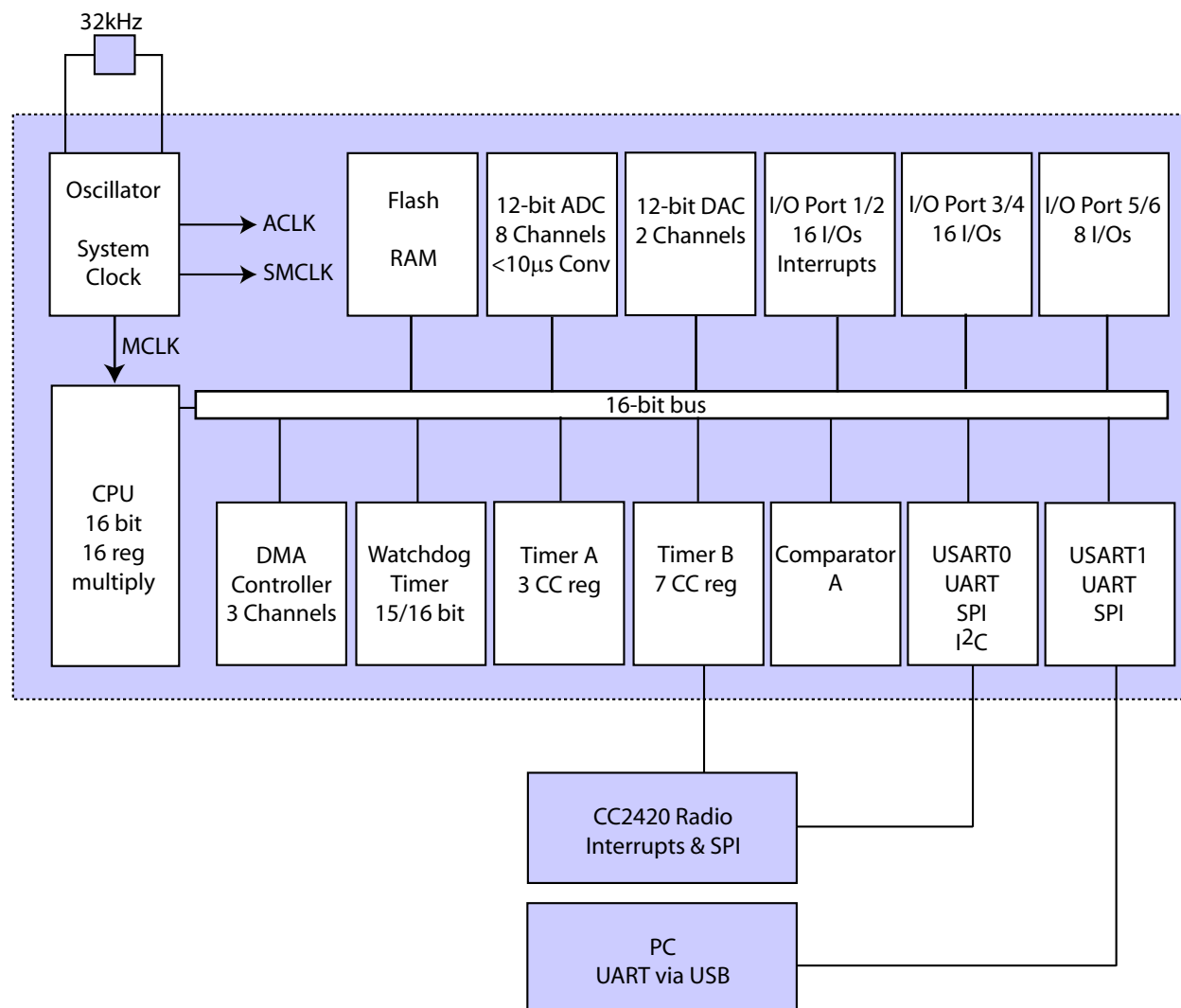


Figure 6 : Block diagram of the TI MSP430 microcontroller and its connection to other peripherals in the Tmote Sky module

## Radio

### Description

Tmote Sky features the Chipcon CC2420 radio for wireless communications. The CC2420 is an IEEE 802.15.4 compliant radio providing the PHY and some MAC functions. With sensitivity exceeding the IEEE 802.15.4 specification and low power operation, the CC2420 provides reliable wireless communication. The CC2420 is highly configurable for many applications with the default radio settings providing IEEE 802.15.4 compliance. Features and usage of the CC2420 is available in Chipcon's datasheet at <http://www.chipcon.com>

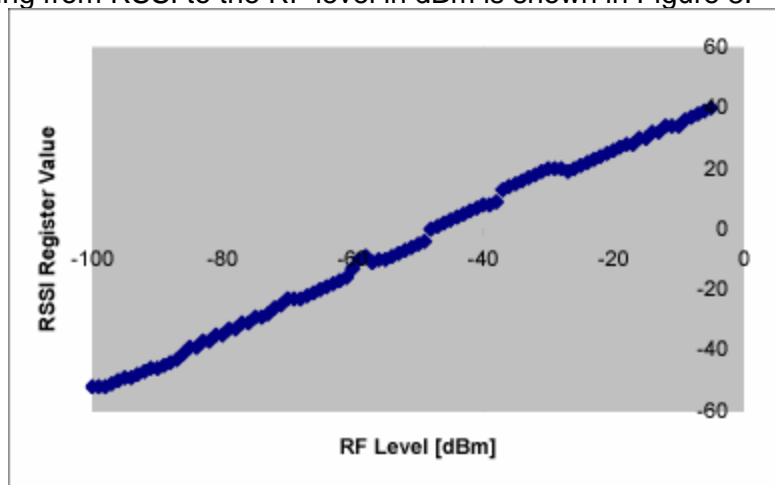
The CC2420 is controlled by the TI MSP430 microcontroller through the SPI port and a series of digital I/O lines and interrupts (see the Schematics on page 7 for more information). The radio may be shut off by the microcontroller for low power duty cycled operation.

The CC2420 has programmable output power. Common CC2420 register values and their corresponding current consumption and output power are shown in Figure 7.

| PA_LEVEL | TXCTRL register | Output Power [dBm] | Current Consumption [mA] |
|----------|-----------------|--------------------|--------------------------|
| 31       | 0xA0FF          | 0                  | 17.4                     |
| 27       | 0xA0FB          | -1                 | 16.5                     |
| 23       | 0xA0F7          | -3                 | 15.2                     |
| 19       | 0xA0F3          | -5                 | 13.9                     |
| 15       | 0xA0EF          | -7                 | 12.5                     |
| 11       | 0xA0EB          | -10                | 11.2                     |
| 7        | 0xA0E7          | -15                | 9.9                      |
| 3        | 0xA0E3          | -25                | 8.5                      |

**Figure 7 : Output power configuration for the CC2420**

The CC2420 provides a digital received signal strength indicator (RSSI) that may be read any time. Additionally, on each packet reception, the CC2420 samples the first eight chips, calculates the error rate, and produces a link quality indication (LQI) value with each received packet. A mapping from RSSI to the RF level in dBm is shown in Figure 8.



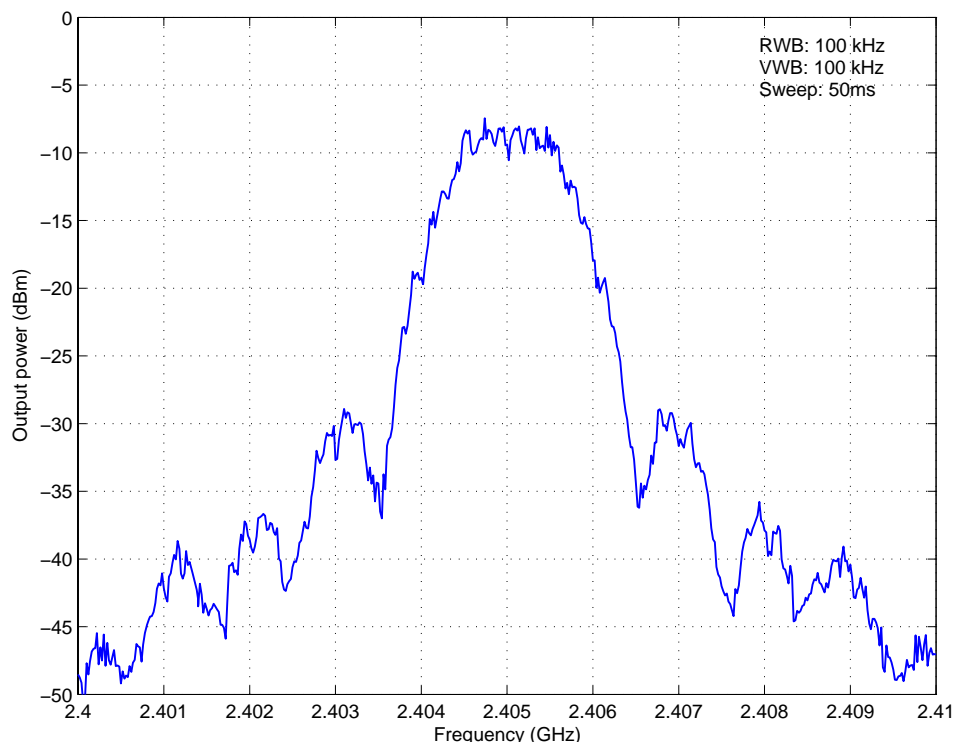
**Figure 8 : Received Signal Strength Indicator mapping to RF Power [dBm]**

## Typical Operating Conditions

|  | MIN  | NOM  | MAX    | UNIT |
|--|------|------|--------|------|
| Supply voltage during radio operation (Vreg on)  | 2.1  |      | 3.6    | V    |
| Operating free air temperature                   | -40  |      | 85     | °C   |
| RF frequency range                               | 2400 |      | 2483.5 | MHz  |
| Transmit bit rate                                | 250  |      | 250    | kbps |
| Nominal output power                             | -3   | 0    |        | dBm  |
| Programmable output power range                  |      | 40   |        | dBm  |
| Receiver sensitivity                             | -90  | -94  |        | dBm  |
| Current consumption: Radio transmitting at 0 dBm |      | 17.4 |        | mA   |
| Current consumption: Radio receiving             |      | 19.7 |        | mA   |
| Current consumption: Radio on, Oscillator on     |      | 365  |        | μA   |
| Current consumption: Idle mode, Oscillator off   |      | 20   |        | μA   |
| Current consumption: Power Down mode, Vreg off   |      |      | 1      | μA   |
| Voltage regulator current draw                   | 13   | 20   | 29     | μA   |
| Radio oscillator startup time                    |      | 580  | 860    | μs   |

## Measured Output Power

The RF output power of the Tmote Sky module from the CC2420 radio is shown in Figure 9. For this test, the Tmote Sky module is transmitting at 2.405GHz (IEEE 802.15.4 channel 11) using the O-QPSK modulation with DSSS. The CC2420 programmed output power is set to 0 dBm. The measured output power of the entire modulated spectrum is 2.4 dBm.



**Figure 9 : Measured RF output power over the modulated spectrum from the Tmote Sky module**



## Antenna

Tmote Sky's internal antenna is an Inverted-F microstrip design protruding from the end of the board away from the battery pack. The Inverted-F antenna is a wire monopole where the top section is folded down to be parallel with the ground plane. Although not a perfect omnidirectional pattern, the antenna may attain 50-meter range indoors and upwards of 125-meter range outdoors. Measurements of the internal antenna's performance with and without a battery pack are shown in Figure 10 and Figure 11. Approximate radiation patterns for the Inverted-F antenna as provided by Chipcon AS are shown in Figure 12 and Figure 13.

### Internal Antenna without Battery Pack

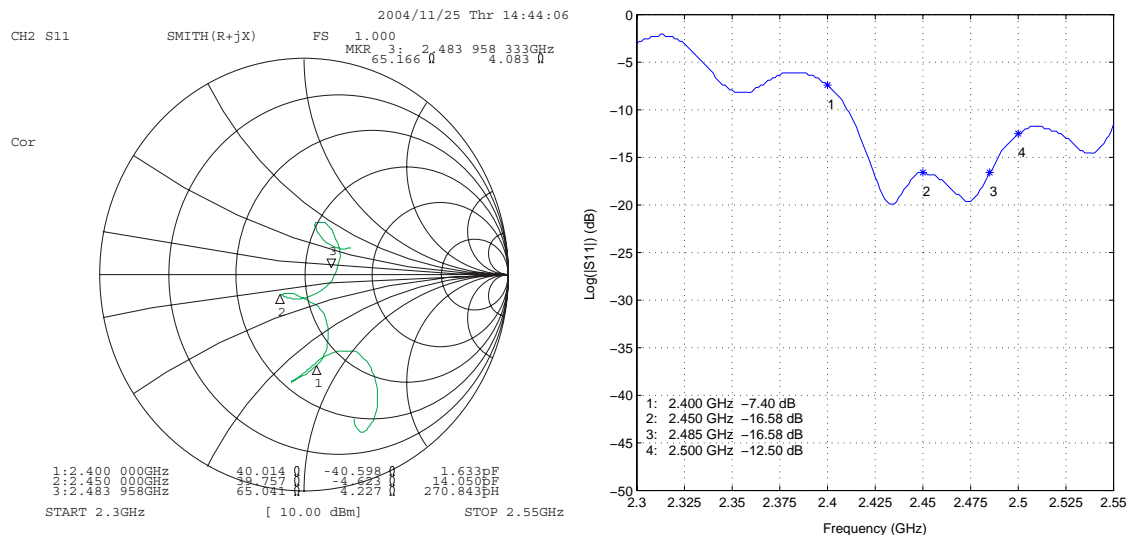


Figure 10 : S11 measurements for the internal inverted-F antenna when no battery pack is present

### Internal Antenna with Battery Pack

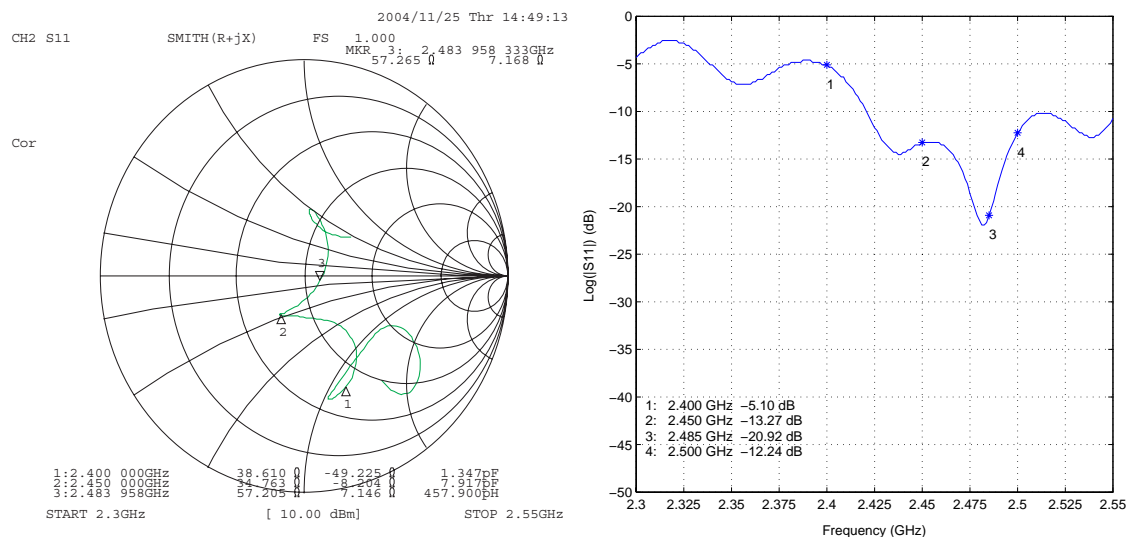


Figure 11 : S11 measurements for the internal inverted-F antenna with battery pack underneath

## Radiation Pattern

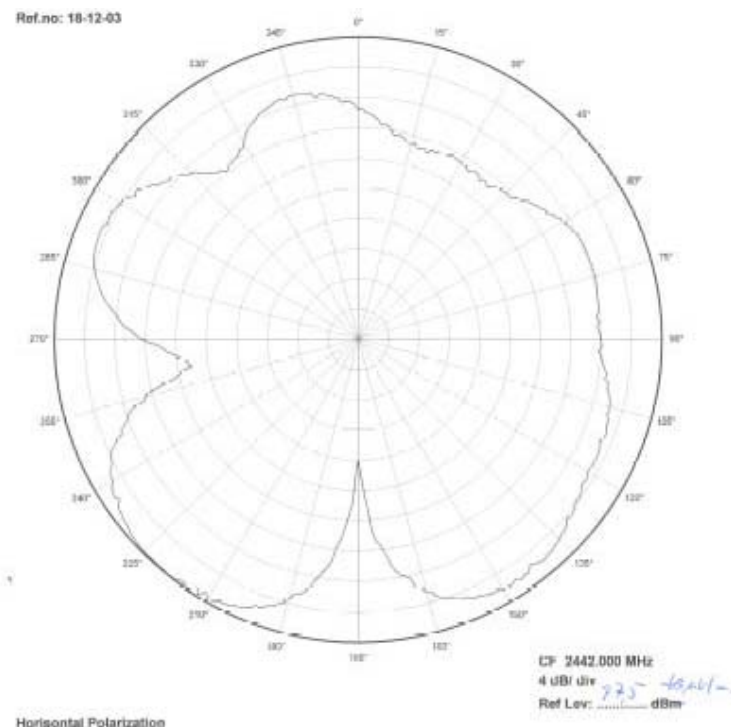


Figure 12 : Radiated pattern of the Inverted-F antenna with horizontal mounting (from Chipcon AS)

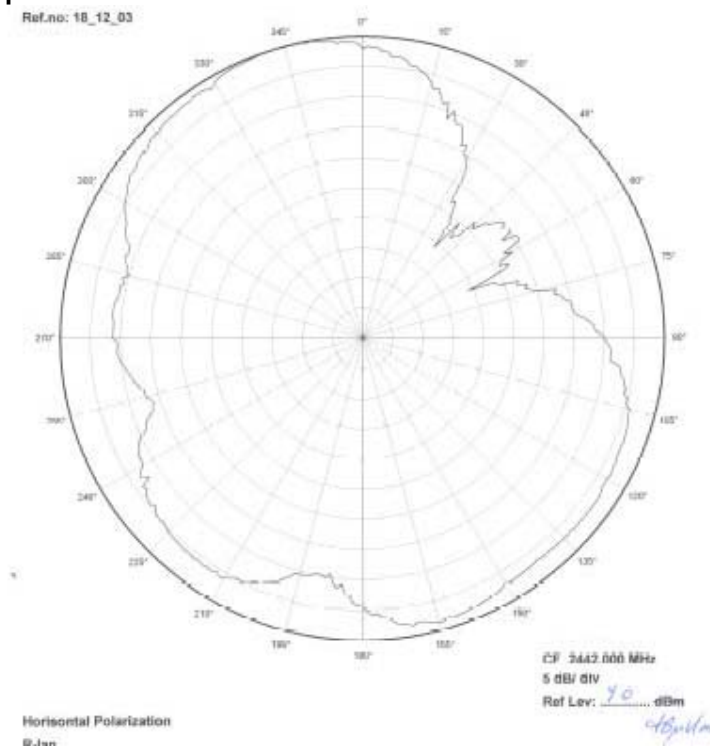
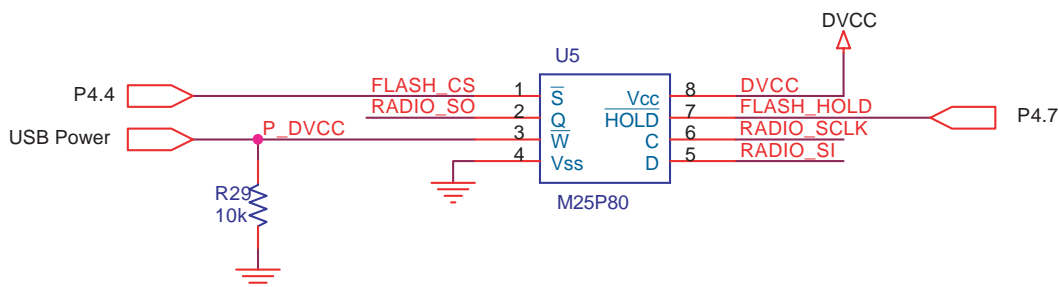


Figure 13 : Radiated pattern of the Inverted-F antenna with vertical mounting (from Chipcon AS)

## External Flash

Tmote Sky uses the ST M25P80 40MHz serial code flash for external data and code storage. The flash holds 1024kB of data and is decomposed into 16 segments, each 64kB in size. The flash shares SPI communication lines with the CC2420 transceiver. Care must be taken when reading or writing to flash such that it is interleaved with radio communication, typically implemented as a software arbitration protocol for the SPI bus on the microcontroller.



**Figure 14 : External serial flash schematic**

## Typical Operating Conditions

|  | MIN | NOM | MAX     | UNIT   |
|--|-----|-----|---------|--------|
| Supply voltage during flash memory programming | 2.7 |     | 3.6     | V      |
| Operating free air temperature                 | -40 |     | 85      | °C     |
| Erase/Programming cycles                       |     |     | 100,000 | cycles |
| Data Retention                                 |     |     | 20      | years  |
| Active current (READ)                          |     |     | 4       | mA     |
| Active current (WRITE/ERASE)                   |     |     | 20      | mA     |
| Standby current                                |     | 8   | 50      | μA     |
| Deep Power Down current                        |     | 1   | 10      | μA     |



**NOTE:** The ST M25P-series of code flash always starts in the standby state. For low power applications, the flash must be sent a command at boot time to place it in the deep power down mode. If using TinyOS, the flash is automatically put into deep power down mode and must be instructed to exit deep power down mode the first time the flash is accessed. See the ST M25P80 datasheet for more information.  
<http://www.st.com/stonline/books/pdf/docs/8495.pdf>

## Flash Hardware Write Protection

The flash includes hardware write protection functionality. The write protection exists on a sector basis as shown in Figure 16. The hardware write protection pin (Pin 3 of the M25P80 shown in Figure 14) only disables write protection when the module is powered by the USB port. When connected to USB, the status register must be updated by removing the write protect and block protect bits in Figure 15. The write protected segments may only be changed after the module connected to USB and the write protect bit is cleared.

Tmote Sky ships with sector 15 (the upper sixteenth sector) write protected and the SRWD bit set. In sector 15 is the “Golden Image” and factor metadata. The “Golden Image” is a factory program image that includes network reprogramming so that Tmote Sky may always return to a known good state, even if loaded with a malfunctioning program image. When Tmote Sky is connected to the USB, the “Golden Image” may be changed. See the TinyOS Deluge documentation in `tinyos-1.x/docs` of the TinyOS distribution included with Tmote.

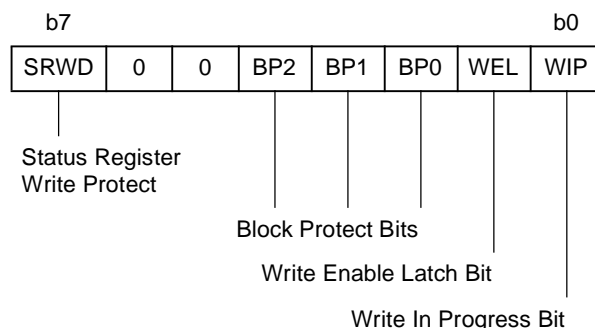


Figure 15 : ST M25P80 Status Register contents (from ST).

| Status Register Content |         |         | Memory Content (Sectors) |                                 |
|-------------------------|---------|---------|--------------------------|---------------------------------|
| BP2 Bit                 | BP1 Bit | BP0 Bit | Protected Area           | Unprotected Area                |
| 0                       | 0       | 0       | None                     | All sectors (0-15)              |
| 0                       | 0       | 1       | Upper sixteenth (15)     | Lower fifteen-sixteenths (0-14) |
| 0                       | 1       | 0       | Upper eighth (14-15)     | Lower seven-eighths (0-13)      |
| 0                       | 1       | 1       | Upper quarter (12-15)    | Lower three-quarters (0-11)     |
| 1                       | 0       | 0       | Upper half (8-15)        | Lower half (0-7)                |
| 1                       | 0       | 1       | All sectors (0-15)       | None                            |
| 1                       | 1       | 0       | All sectors (0-15)       | None                            |
| 1                       | 1       | 1       | All sectors (0-15)       | None                            |

Figure 16 : Write protection settings for the ST M25P80 flash.  
Tmote Sky modules are shipped with the gray setting (001).



**NOTE:** When programming data to write protected segments of external flash, do not disconnect the module before the programming is completely. If the module is disconnected from the USB, the write may be interrupted or the status register may not be updated to reflect the new write protection settings.

## Sensors

### Humidity/Temperature Sensor

The optional humidity/temperature sensor is manufactured by Sensirion AG. The SHT11 and SHT15 models may be directly mounted on the Tmote Sky module in the U3 component position.

The SHT11/SHT15 sensors are calibrated and produce a digital output. The calibration coefficients are stored in the sensor's onboard EEPROM. The difference between the SHT11 and SHT15 model is that the SHT15 produces higher accuracy readings as shown in Figure 18. The sensor is produced using a CMOS process and is coupled with a 14-bit A/D converter. The low power relative humidity sensor is small in size and may be used for a variety of environmental monitoring applications.

More information can be found in the SHT1x datasheet available at <http://www.sensirion.com>

| Parameter          | MIN  | TYP  | MAX   | Units |
|--------------------|------|------|-------|-------|
| <b>Humidity</b>    |      |      |       |       |
| Resolution         | 0.5  | 0.03 | 0.03  | %RH   |
|                    | 8    | 12   | 12    | Bit   |
| Repeatability      |      | ±0.1 |       | %RH   |
| Range              | 0    |      | 100   | %RH   |
| <b>Temperature</b> |      |      |       |       |
| Resolution         | 0.04 | 0.01 | 0.01  | °C    |
|                    | 0.07 | 0.02 | 0.02  | °F    |
|                    | 12   | 14   | 14    | bit   |
| Repeatability      |      | ±0.1 |       | °C    |
|                    |      | ±0.2 |       | °F    |
| Range              | -40  |      | 123.8 | °C    |
|                    | -40  |      | 254.9 | °F    |

Figure 17 : Sensirion relative humidity and temperature performance specifications

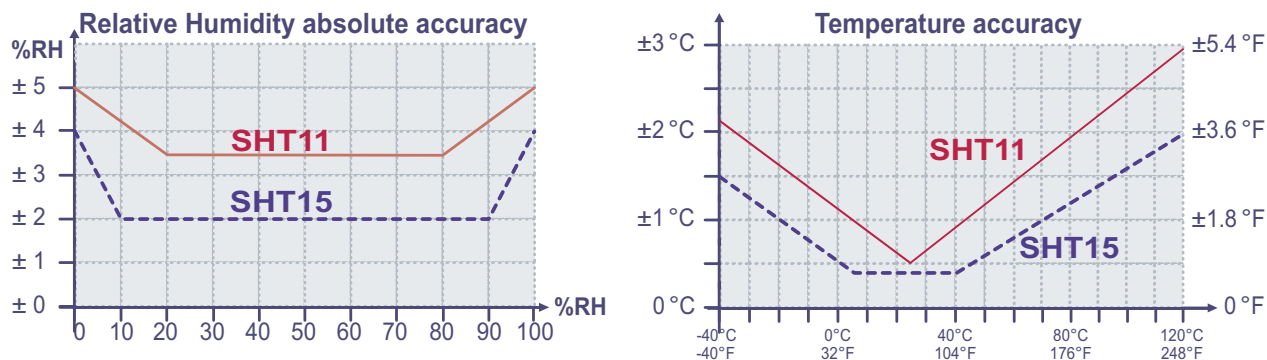


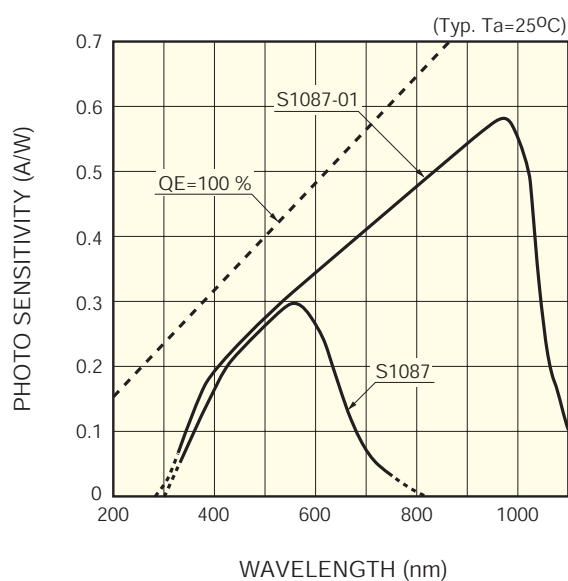
Figure 18 : Accuracy of Sensirion relative humidity and temperature sensors (courtesy Sensirion)

## Light Sensors

A variety of light sensors may be used with Tmote Sky. Tmote Sky has connections for two photodiodes. Moteiv currently uses photodiodes from Hamamatsu Corporation (<http://www.hamamatsu.com>)

If your mote is populated with light photodiodes, the default diodes are the S1087 for sensing photosynthetically active radiation and the S1087-01 for sensing the entire visible spectrum including infrared

Although these photodiodes from Hamamatsu have been tested with Tmote Sky, any photodiode with similar physical dimensions may be used with Tmote Sky.



**Figure 19 : Photo Sensitivity of the Light sensors on Tmote Sky(from Hamamatsu)**

## Expansion Connector

Tmote Sky has two expansion connectors and a pair of onboard jumpers that may be configured so that additional devices (analog sensors, LCD displays, and digital peripherals) may be controlled by the Tmote Sky module. On the far side of the board from the USB connector is a 10-pin IDC header at position U2 and a 6-pin IDC header at U28. The 10-pin connector has the same connections as Tmote Sky and is the primary connector. It provides digital input and output signals as well as analog inputs. Peripherals may be connected to the 10-pin connector using an IDC header, an IDC ribbon cable, or by designing a printed circuit board that solders directly on to the IDC header providing a robust connection to the module. An additional 6-pin (U28) header provides access to the exclusive features of Tmote Sky. Two additional ADC inputs are provided that may be reconfigured by software to be two 12-bit DAC outputs. ADC7 may also act as the input to the supply voltage supervisor. The user interface elements—the reset and user buttons—are exported by the 6-pin header for use in external interfaces and packaging.

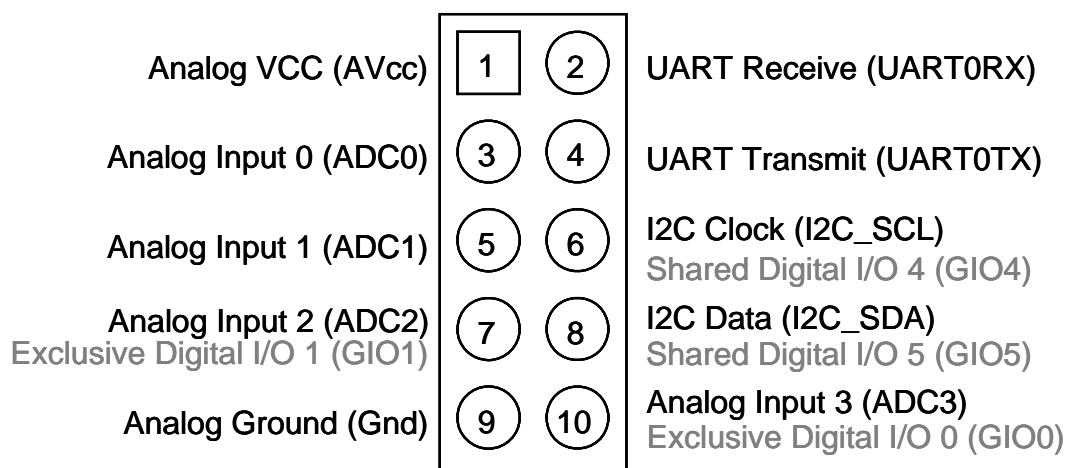


Figure 20 : Functionality of the 10-pin expansion connector (U2).  
Alternative pin uses are shown in gray.

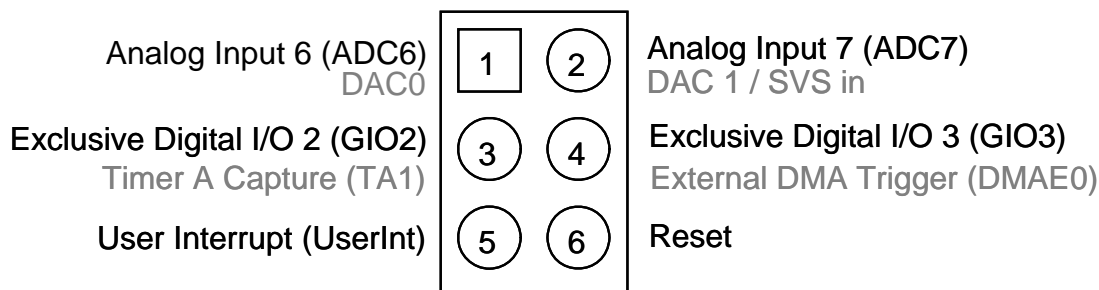


Figure 21 : Functionality of the 6-pin expansion connector (U28).



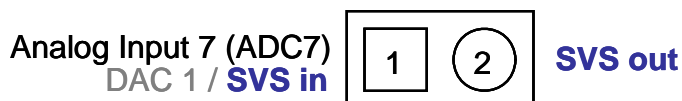
**NOTE:** The I<sup>2</sup>C pins are shared with the radio's data input pin and the radio clock. Care must be taken by application developers to multiplex operations on the I<sup>2</sup>C bus and the radio.

If expansion pin 10 (ADC3) is used to generate microcontroller interrupts instead of digital I/O and analog input functionality, R14 must be populated with a 0 ohm resistor to enable the pin for digital I/O (GIO0) on the microcontroller. R16 must be populated with a 0 ohm resistor to enable GIO1. R14 and R16 are located on the top side of Tmote Sky between the USB controller and the radio.

**NOTE:** When R14/R16 is populated (GIO0/GIO1 enabled), ADC3/ADC2 will not provide reliable readings if an application reverts to using the ADC input instead of the digital I/O port input on the microcontroller. R14/R16 are not required for digital I/O functionality from ADC3/ADC2; they are only needed if an interrupt request must be generated by an external device on these pins. R14/R16 should be removed when using ADC3/ADC2 for analog input.

The 6-pin IDC header also has an optional jumper, R15. By installing a 0 ohm resistor at R15, GIO3 is directly connected to SVSout. By making GIO3 an input and using the SVS features of the microcontroller, the SVSout function can be exported via pin 4 of U28.

A separate Supply Voltage Supervisor (SVS) 2-pin IDC header is provided underneath the USB connector at position U7. The SVS header allows add-on boards to be built that connect to the positive and negative battery terminals and the SVS pins in order to provide power the module and use the microcontroller's advanced SVS functionality for boost converters, solar systems, and rechargeable systems. The SVS header is shown in Figure 22 and includes the SVSin and SVSout pins from the microcontroller.



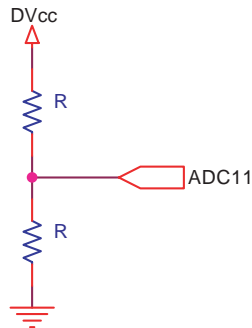
**Figure 22 : Functionality of the 2-pin Supply Voltage Supervisor connector (U7).**



## Internal Temperature and Voltage Monitoring

The MSP430 microcontroller has internal temperature and voltage sensors that may be used through the microcontroller's ADC interface.

The voltage port (input 11) on the 12-bit ADC monitors the output from a voltage divider.

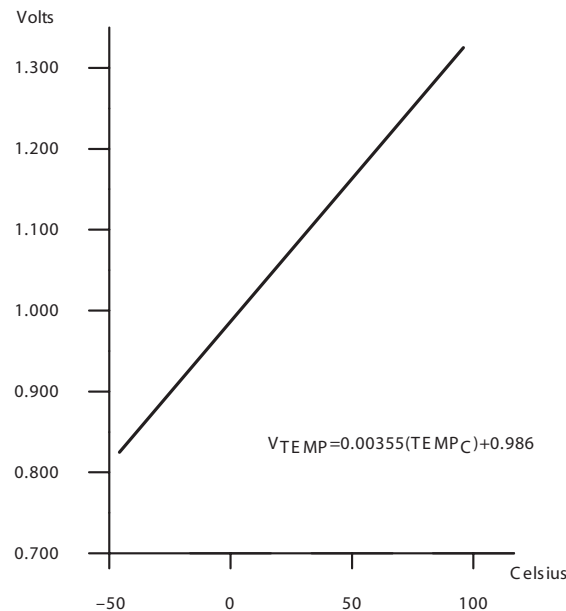


Voltage monitoring for Tmote Sky modules.

Converting the ADC units to a voltage reading can be done with the following formula:

$$DV_{cc} = \frac{ADCCounts}{4096} \times V_{ref} \times \frac{2R}{R}$$

The temperature input is a temperature diode connected to internal ADC port 10. When using the temperature sensor, the sample period must be greater than 30  $\mu$ s. The temperature sensor offset error can be large, and may need to be calibrated for most applications. The typical response of the temperature sensor is shown in Figure 23.



**Figure 23 : Typical response of the internal temperature sensor. Results vary and the sensor should be calibrated for most applications. Response curve from Texas Instruments.**