Πανεπιστήμιο Θεσσαλίας, 2012-2013

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Μεταπτυχιακή εργασία

Θέμα:

« Ανάπτυξη πλατφόρμας ασύρματων αισθητήρων και κατάλληλου λογισμικού διαχείρισης »

Καζδαρίδης Ιωάννης

Επιβλέπων καθηγητής:

Λέανδρος Τασιούλας (Καθηγητής)

Συνεπιβλέπων καθηγητές:

Κουτσόπουλος Ιορδάνης (Επίκουρος Καθηγητής)

Αργυρίου Αντώνιος (Λέκτορας)

Βόλος, Μάρτιος 2013

Ευχαριστίες

Με την εκπόνηση της παρούσας μεταπτυχιακής εργασίας, φέρνω εις πέρας τις μεταπτυχιακές μου σπουδές στο Τμήμα Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

Θα ήθελα αρχικά να ευχαριστήσω θερμά τον Λέκτορα του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων κ. Κοράκη Αθανάσιο, και τον καθηγητή του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων κ. Λέανδρο Τασιούλα, για τις χρήσιμες συμβουλές και υποδείξεις του καθώς και για την υποστήριξη που μου προσφέρανε κατά τη διάρκεια της φοίτησής μου, αλλά και κατά την εκπόνηση της διπλωματικής μου εργασίας. Επιπροσθέτως να τους ευχαριστήσω για την δυνατότητα που μου δίνουν να βρίσκομαι στην ομάδα του NITlab και να ασχολούμαι συνεχώς με νέες τεχνολογίες και να κάνω έρευνα στον χώρο τον ασύρματων δικτύων.

Από καρδιάς να ευχαριστήσω όλη την ομάδα του NITlab μέσα από την οποία μαθαίνω διαρκώς καινούργια πράγματα και κάνω σημαντικές συνεργασίες. Ιδιαιτέρως να ευχαριστήσω τον υποψήφιο διδάκτορα του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων Κερανίδη Στράτο για την βοήθειά του στις μετρήσεις κατανάλωσης ενέργειας.

Τέλος, ευχαριστώ θερμά την οικογένειά μου για την αμέριστη συμπαράσταση που μου παρείχε όλα αυτά τα χρόνια για την ολοκλήρωση των μεταπτυχιακών και προπτυχιακών σπουδών μου.

Αφιερωμένο στην θεία μου,
Καζδαρίδου Γεωργία

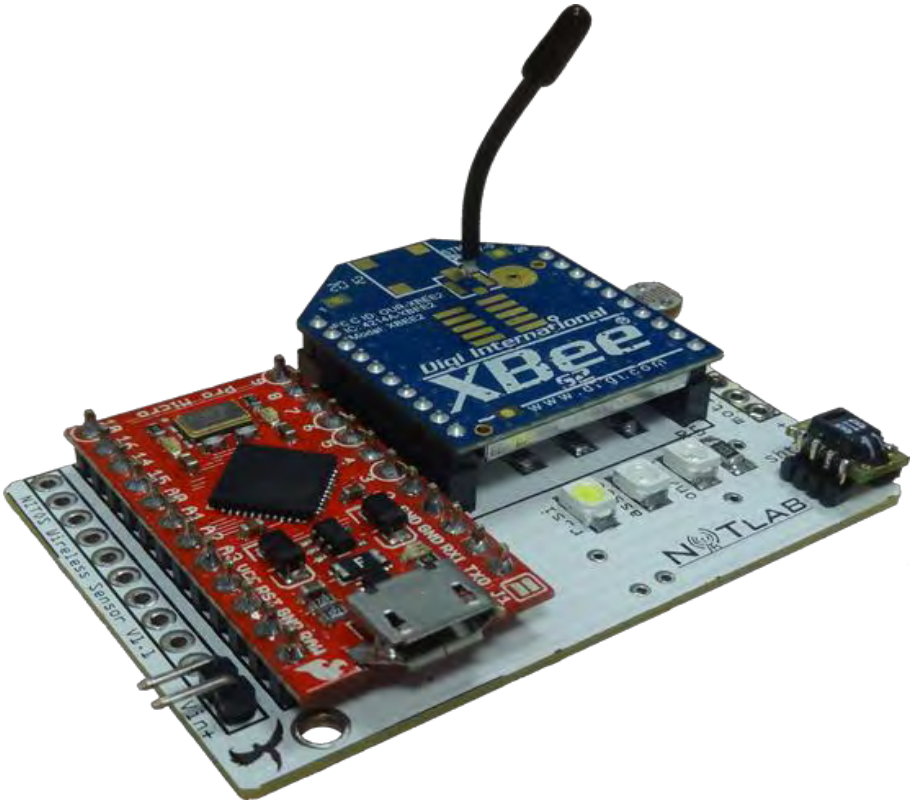
CONTENTS

ABSTRACT

Towards the direction of enabling sensing based on WSN solutions at NITOS, this master thesis presents a developed prototype wireless sensor platform comprised of open-source and configurable modules. The main part of the aforementioned platform is a Pro Micro [16] board, fabricated by Sparkfun Electronics [17], a well-known industrial company, which provides open-source hardware platforms that are fully configurable by Arduino's [18] open-source software. The on-board AVR micro-controller developed by Atmel [27], runs at 8Mhz and coordinates the overall platform operation. Moreover, the platform is equipped with an Xbee [28] radio interface that enables communication with the respective gateway. The Xbee module is a tiny device ideal for setting up mesh networks and has a defined rate of 250 kbps. This module uses the IEEE 802.15.4 stack (the basis for Zigbee[6]) and wraps it into a simple to use serial command set, allowing a very reliable and simple communication with Pro Micro's micro-controller. The developed platform currently features a number of sensing modules, such as air temperature and humidity, light intensity and human presence. Various types of sensing modules and actuators can be further integrated utilizing Arduino libraries that implement several existing communications protocols, such as I2C, LIN, SPI, TWI, USI, etc. Firmware can be easily uploaded through the on-board USB connection. In addition to this, the developed platform supports over-the-air-programming achieved by a special circuit already integrated in the PCB, allowing firmware to be uploaded wirelessly. Apart from the pluggable Xbee module, the developed platform can be equipped with WiFi or Bluetooth radio interfaces compatible with Xbee footprint, thus enabling communication utilizing different technology standards. By exploiting open-source firmware, developers can define different network topologies according to their experimentation setup, while numerous aggregation schemes can be used for gathering information reducing the total communication cost. Furthermore, the developed platform is a low-cost and small-sized one that can be powered by low-voltage sources, making it ideal for extended deployment at almost any place.

1 Introduction

Wireless sensor devices or networks are becoming popular day by day and becoming realism in different platforms such as scientific, medical, research, manufacturing, commercial, industry and so on. In this master thesis, we present a custom-made Wireless Sensor platform that can feature various types of sensors, while it can be programmed in a very easy way. In the second chapter we present 802.15.4 MAC and ZigBee network protocol. In the third chapter we present existing commercial wireless sensor platforms. Finally, in the fourth chapter we present the modules that we used to build the NITOS Wireless Sensor Platform, the configuration process, a demo and some measurements.



2 Existing MAC and Network Protocols

2.1 802.15.4 MAC Protocol

This standard defines a communication layer at level 2 in the OSI (Open System Interconnection) model. Its main purpose is to let the communication between two devices.It was created by the Institute of Electrical and Electronics Engineers (IEEE) [5], entity which main task is to set standards so that technological developments can count with a common platform of rules to be set over.

2.1.1 Understanding 802.15.4

As mentioned before this protocol lies over the level 2 of the OSI. This layer is called the Data Link. Here the digital information units (bits) are managed and organized to become electromagnetic impulses (waves) on the lower level, the physical one. This layer is similar to others known ones such as the 802.11 (commercially named under Wifi technologies) or the common Ethernet (802.3). The frequencies defined in the standard are spread among 27 different channels divided in three main bands and three different bit rates.

Bands:

- 868.0 - 868.6MHz -> 1 channel (Europe)
- 902.0-928.0MHz -> 10 channels (EEUU)
- 2.40-2.48GHz -> 16 channels (Worldwide)

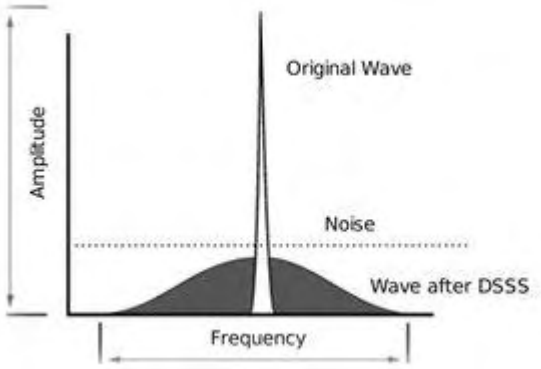
Bit Rates:

- 868.0 - 868.6MHz -> 20/100/250 Kb/s
- 902.0-928.0MHz -> 40/250 Kb/s
- 2.40-2.48GHz -> 250 Kb/s

2.1.2 802.15.4 against noise

It used Direct Sequence Spread Spectrum (DSSS) [2] to modulate the information before being sent to the physical layer. Basically, each bit of information to be transmitted is modulated into 4 different signals (another kind of bits alphabet), this process causes the total information to be transmitted to occupy a larger bandwidth but it uses a lower spectral power density for each signal.

This causes less interference in the frequency bands used and improves the Signal to Noise Ratio (SNR) in the receiver due to the fact that is easier to detect and decode the message which is being sent by the transmitter.



There are different DSSS modulations depending on the hardware physical limits of the circuit and number of symbols which can be processed at a given time. Binary Phase Shift Keying (BPSK), Offset Quadrature Phase Shift Keying (O-QPSK) and Parallel Sequence Spread Spectrum (PSSS) let communication from bandwidths from 20Kb/s to 250Kb/s.

2.1.3 802.15.4 against interferences

802.15.4 uses two techniques to avoid all the nodes start emitting at the same time: CSMA-CA [3] and GTS [4].

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 Guarantee Time Slots (**GTS**). These systems use 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. There are some special frames like the **ACK** packets which doesn't require this method to be performed.

One of the functionalities implemented in **802.15.4** is the **channel energy scan** (PLME-ED request). The idea is to be able to know how much energy (activity/noise/interferences) there is in one (or several channels) prior to start using it. This way we can save energy choosing free channels when setting the network. There are three different behaviors when facing the energy detection issue:

- **Energy**: scan the channels and report the energy found. It doesn't matter if it is caused by other ZigBee nodes or by another technology or noise. Just report if the spectrum is being used. Only when the value received is below a certain threshold we will transmit.
- **Carrier Sense (CCA)**: scan the medium and report if there are 802.15.4 transmissions. Only when the channel is free we will transmit.
- **CCA + Energy**: scan the medium and report if there are 802.15.4 transmissions above the energy threshold specified. If not we will use the channel.

2.1.4 802.15.4, a low consumption protocol

It is ready to work with **low-duty cicles**. 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.

2.1.5 Transmission Power and Reception Sensibility for 802.15.4 transceivers

802.15.4 sets the minimum amount of energy needed to transmit in **-3dBm**, (**0.5 mW**) and the minimum sensibility in the receiver is **-92dBm** (**6.3 * e-10 mW**). Regarding the reception sensibility, the XBee shows **-92Bm** (**6'3 * e-10 mW**) and **-100dBm** (**1 * 1e-10mW**) the **XBee-Pro** flavor.
This means we will detect any packet which reach us with a energy of as low as 0.000000000063mW for the XBee and 0.00000000001mW for the XBee-Pro.

2.1.6 On top of 802.15.4

There are several protocols which use 802.15.4 as its MAC layer. The most known is ZigBee [6], although there are a lot of them:

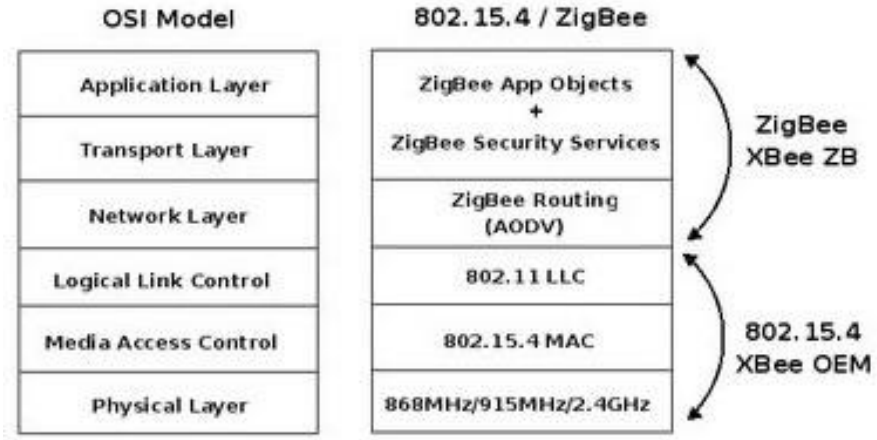
- Wireless HART [7]: It is the wireless version of the HART protocol which is the most used in the automation and industrial applications which require real time. It uses Time Synchronized Mesh Protocol (TSMP). A "time coordinator" node is required in order to assign the time slot to all the motes.
- ISA - SP100 [8]: It also centers in the process and factory automation. It is being developed by the Systems and Automation Society (ISA) and tries to be an standard for this kind of projects.

- IETF IPv6 - LoWPAN [9]: As the same name points it is the implementation of the IPv6 stack on top of 802.15.4 to let any device be accessible and access from and to Internet.
- **New Mesh protocols**: A lot of different mesh protocols are being implemented by companies over the 802.15.4 MAC layer. One example is the Digi's own mesh protocol (DigiMesh [10]), which is though to be a completely distributed mesh protocol where all the nodes can sleep and route their brother's packets.

Reference guide to 802.15.4 description: [1].

2.2 ZigBee Routing Protocol

This standard defines a 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 in the upper layer application services. It was created by a set of companies which form the ZigBee Alliance [6].



2.2.1 Understaning ZigBee

ZigBee offers basically four kinds of different services:

- **Extra Encription services** (application and network keys implement extra 128b AES encryption)
- **Association and authentication** (only valid nodes can join to the network).
- **Routing protocol**: AODV [11], a reactive ad hoc protocol has been implemented to perform the data routing and forwarding process to any node in the network.
- **Application Services**: An abstract concept called **"cluster"** is introduced. Each node belongs to a predefined cluster and can take a predefined number of

actions. Example: the "house light system cluster" can perform two actions: "turn the lights on", and "turn the lights off".

ZigBee is a layer thought to organize the network. The first thing a node (route or end device) which want to join the network has to do is to ask to the coordinator for a network address (**16b**), as part of the **association** process. All the information in the network is routed using this address and not the 64b MAC address. In this step authentication and encryption procedures are performed.

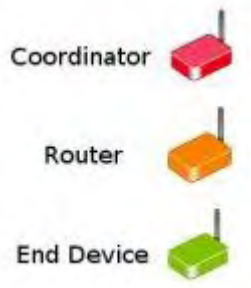
Once a node has joined to the network can send information to its brothers through the routers which are always awake waiting for the packets. When the router gets the packet and the destination is in its radio of signal, the router first looks if the destination end device is awake or slept. In the first case the router sends the packet to the end device, however if it is sleeping, the router will bufferize the packet untill the end device node gets awake and ask for news to the router.

2.2.2 Is ZigBee a mesh protocol?

To answer this question we have to clarify what the mesh concept means. Let's see first how a ZigBee network works.

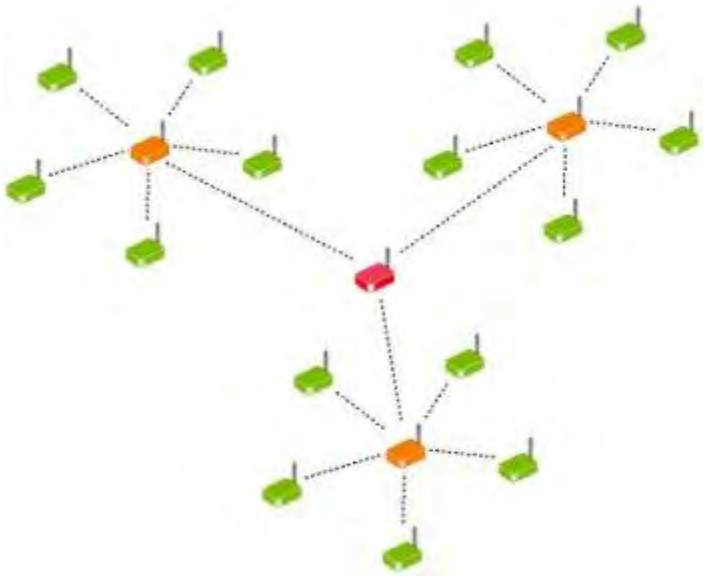
There are three kinds of nodes in a ZigBee network:

- **Coordinator**: is the "master" device, it governs the entire network.
- **Routers**: they route the information which sent by the end devices.
- **End device**: (the motes): they are the sensor nodes, the ones which take the information from the environment.



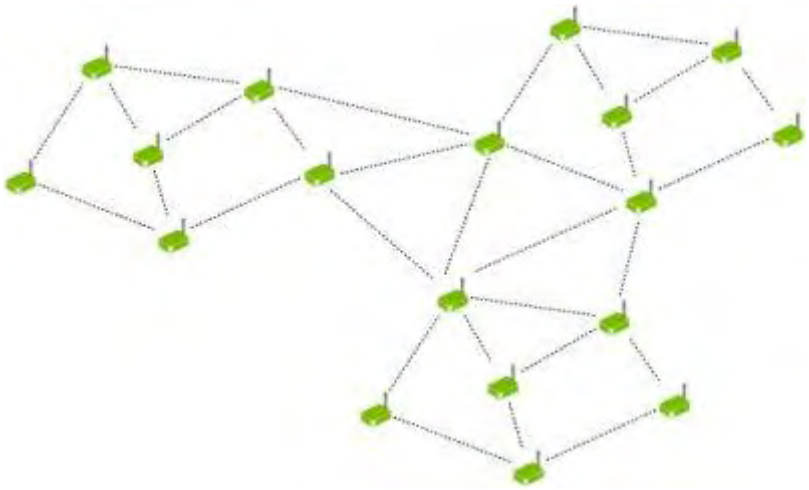
Coordinator and routes cannot be battery powered, motes can. ZigBee creates **star topologies**. There are some basic rules:

- The end devices connect to a router or a coordinator.
- The routers can connect among them and with the coordinator.
- The routers and coordinators cannot sleep. They have to save in their buffer the packets which go to the end devices.
- The end devices can sleep.



The concept "Mesh Network" relays in the Ad hoc communications, also called peer to peer (P2P). This means all the devices in the network can communicate with each other directly. They have to be able to discover each other and send broadcast messages to all the brothers (***"hello! is anybody out there?"***). They have to be able to create networks like the one represented in the image below.

ZigBee creates start network topologies, not mesh ones. To create a completely mesh network such as the one showed in the image below all the nodes have to have the same **role**, all of them have to be *"end devices + routers"* so that they can route their brothers information and sleep when no action is required (saving energy). The **DigiMesh** protocol (over 802.15.4) sets a completely distributed network where all the nodes talk among them using p2p (equal to equal) datagrams.



Reference guide to ZigBee protocol description: [1].

3 Related Wireless Sensor Platforms

3.1 Intel mote 2

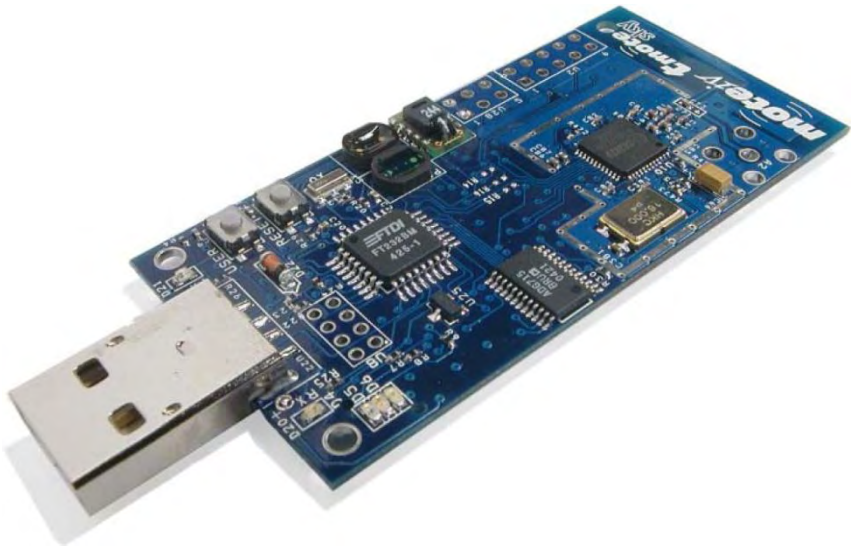
The Imote2 [24] is an advanced wireless sensor node platform developed at Intel Research and supports TinyOS software. It is built around the low-power PXA271 XScale CPU and also integrates an IEEE 802.15.4 compliant radio. The design is modular and stackable with interface connectors for expansion boards on both the top and bottom sides, providing a standard set of I/O signals as well as additional high-speed interfaces for application specific I/O.

Key Features:

- PXA271 XScale Processor
    - Core Frequency: 13/104/208/312/416 MHz
    - 256 KB SRAM
    - 32 MB SDRAM
    - 32 MB Flash
- Zigbee (IEEE 802.15.4) Radio (TI CC2420)
- Mini-USB Client (slave)
    - RS232 console over USB
    - power
- I-Mote2 Basic Sensor connector (31+ 21 pin connector)
- Indicators: Tri-color status LED; Power LED; battery charger LED, console LED
- Switches: on/off slider, Hard reset, Soft reset, User programmable switch

3.2 Tmote Sky

Tmote Sky [25] 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 includes increased performance, functionality, and expansi on. With TinyOS support out-of-the-box, Tmote 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

3.3 Waspmote

Waspmote [26] is a sensor device developers oriented. It works with different communicaction protocols (ZigBee, Bluetooth and GPRS) and frequencies (2.4GHz, 868MHz, 900MHz) and create links up to 12Km. Using the hibernate low power mode (0.7uA) it can save battery energy when not transmitting and be able to work even for years. Waspmote is compatible with more than 50 sensors and its open source IDE makes really easy to start working with it.

Key Features :

- Microcontroller ATmega1281
- Power 3.3V - 4.2V
- Frequency 8MHz
- Input / Output
    - 7 x analog
    - 8 x digital (I/O)
    - 1 x PWM
    - 2 x UART
    - 1 x I2C
    - 1 x USB
- SRAM 8KB

- FLASH 128KB
- EEPROM 4KB
- Power consumption
    - Hibernate g 0.7uA
    - Sleep g 62uA
    - Deep Sleep g 62uA
    - ON g 9mA
- SD card 2GB
- Waspmote sensors on board
    - Temperature [-40ΊC, 85ΊC]
    - Accelerometer ± 2g
    - (1024LSb/g) / ± 6g
    - (340LSBb/g)

4 Configurations and Implementation steps

4.1 XBee Modules, Characteristics and configuration Commands

The XBee and XBee-PRO RF Modules were engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices. The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

4.1.1 Key Features

Long Range Data Integrity XBee

- Indoor/Urban: up to 100’ (30 m)
- Outdoor line-of-sight: up to 300’ (90 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO

- Indoor/Urban: up to 300’ (90 m), 200' (60 m) for International variant
- Outdoor line-of-sight: up to 1 mile (1600 m), 2500' (750 m) for International variant
- Transmit Power: 63mW (18dBm), 10mW (10dBm) for International varian
- Receiver Sensitivity: -100 dBm RF Data Rate: 250,000 bps

Low Power XBee

- TX Peak Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µA

XBee-PRO

- TX Peak Current: 250mA (150mA for interna-tional variant)
- TX Peak Current (RPSMA module only): 340mA (180mA for international variant)
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10 µA

ADC and I/O line support

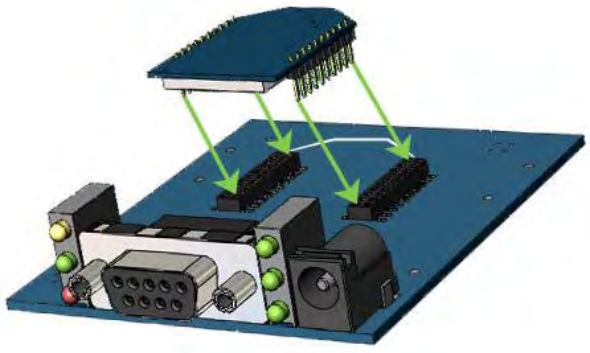
- Analog-to-digital conversion, Digital I/O I/O Line Passing

Advanced Networking & Security

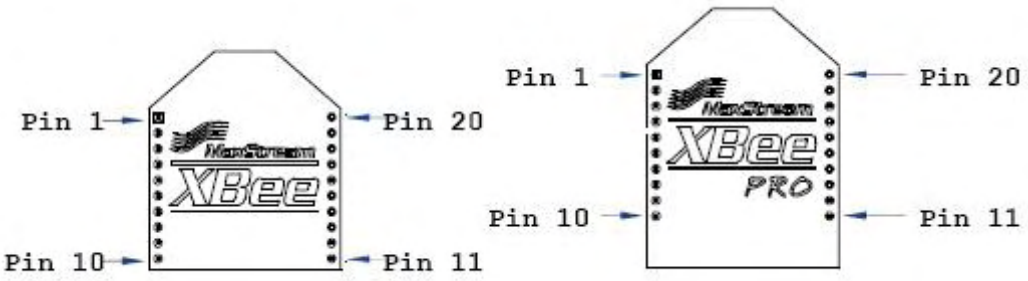
- Retries and Acknowledgements
- DSSS (Direct Sequence Spread Spectrum)
- Each direct sequence channels has over 65,000 unique network addresses available
- Source/Destination Addressing
- Unicast & Broadcast Communications Point-to-point, point-to-multipoint and peer-to-peer topologies supported
- Coordinator/End Device operations
- Transparent and API Operations
- 128-bit Encryption

4.1.2 Mounting Considerations

The XBee®/XBee-PRO® RF Module was designed to mount into a receptacle (socket) and there-fore does not require any soldering when mounting it to a board. The XBee Development Kits con-tain RS-232 and USB interface boards which use two 20-pin receptacles to receive modules.



4.1.3 Pin Signals



Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP**	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	**RTS** / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS & DTR
- Signal Direction is specified with respect to the module
- Module includes a 50k pull-up resistor attached to **RESET**
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

The only required pin connections are VCC, GND, DOUT and DIN. To support serial firmware updates, VCC, GND, DOUT, DIN, RTS, and DTR should be connected. All unused pins should be left disconnected. All inputs on the radio can be pulled high with internal pull-up resistors using the PR software command. No specific treatment is needed for unused out-puts. Other pins may be connected to external circuitry for convenience of operation including the Asso-ciate LED pin (pin 15) and the commissioning button pin

(pin 20). The Associate LED will flash differently depending on the state of the module, and a pushbutton attached to pin 20 can enable various deployment and troubleshooting functions without having to send UART commands. If analog sampling is desired, VRef (pin 14) should be attached to a voltage reference.

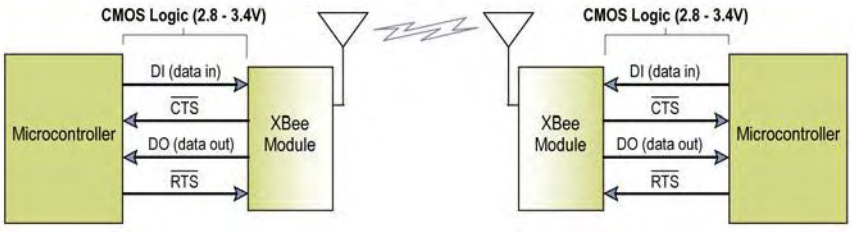
4.1.4 Antenna Performance

Antenna location is an important consideration for optimal performance. In general, antennas radiate and receive best perpendicular to the direction they point. Thus a vertical antenna's radiation pattern is strongest across the horizon. Metal objects near the antenna may impede the radiation pattern. Metal objects between the transmitter and receiver can block the radiation path or reduce the transmission distance, so antennas should be positioned away from them when possible. Some objects that are often overlooked are metal poles, metal studs or beams in structures, concrete (it is usually reinforced with metal rods), vehicles, elevators, ventilation ducts, refrigerators, microwave ovens, batteries, and tall electrolytic capacitors. If the XBee is to be placed inside a metal enclosure, an external antenna should be used. XBee units with the Embedded PCB Antenna should not be placed inside a metal enclosure or have any ground planes or metal objects above or below the antenna. For best results, place the XBee at the edge of the host PCB on which it is mounted. Ensure that the ground, power and signal planes are vacant immediately below the antenna section. Digi recommends allowing a “keepout” area, which is shown in detail on the next page.

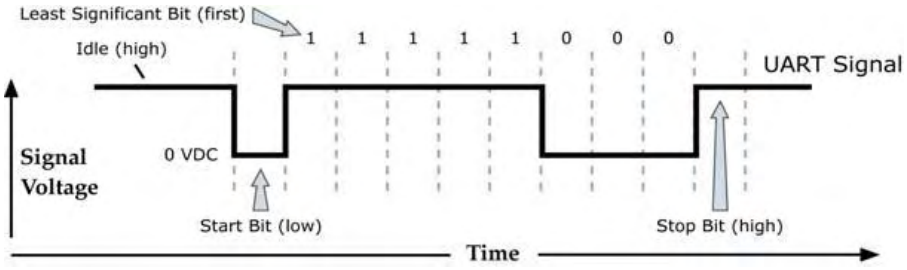
4.1.5 Serial Communications

The XBee®/XBee-PRO® RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage com-patible UART; or through a level translator to any serial device (For example: Through a Digi pro-prietary RS-232 or USB interface board).

Devices that have a UART interface can connect directly to the pins of the RF module as shown in the figure below.



Data enters the module UART through the DI pin (pin 3) as an asynchronous serial signal. The sig-nal should idle high when no data is being transmitted. Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.



Serial communications depend on the two UARTs (the microcontroller's and the RF module's) to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits). The UART baud rate and parity settings on the XBee module can be configured with the BD and NB commands, respectively.

4.1.6 Serial-to-RF Packetization

Data is buffered in the DI buffer until one of the following causes the data to be packetized and transmitted:

- No serial characters are received for the amount of time determined by the RO (Packetiza-tion Timeout) parameter.
- If RO = 0, packetization begins when a character is received. 2. The maximum number of characters that will fit in an RF packet (100) is received. 3.
- The Command Mode Sequence (GT + CC + GT) is received. Any character buffered in the DI buffer before the sequence is transmitted.

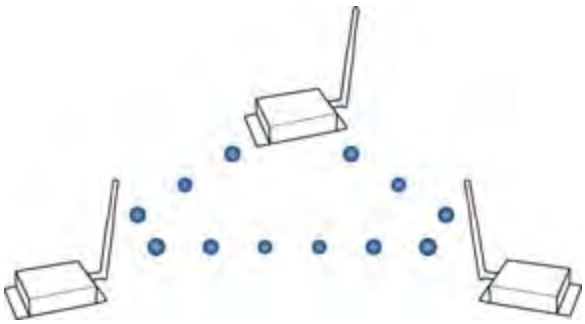
If the module cannot immediately transmit (for instance, if it is already receiving RF data), the serial data is stored in the DI Buffer. The data is packetized and sent at any RO timeout or when 100 bytes (maximum packet size) are received. If the DI buffer becomes full, hardware or software flow control must be implemented in order to prevent overflow (loss of data between the host and module).

4.1.7 Xbee / Xbee Pro Networks

4.1.7.1 Peer-to-Peer

By default, XBee®/XBee-PRO RF Modules are configured to operate within a Peer-to-Peer network topology and therefore are not dependent upon Master/Slave

relationships. NonBeacon systems operate within a Peer-to-Peer network topology and therefore are not dependent upon Master/ Slave relationships. This means that modules remain synchronized without use of master/server configurations and each module in the network shares both roles of master and slave. Digi's peer-to-peer architecture features fast synchronization times and fast cold start times. This default con-figuration accommodates a wide range of RF data applications.



A peer-to-peer network can be established by configuring each module to operate as an End Device (CE = 0), disabling End Device Association on all modules (A1 = 0) and setting ID and CH parameters to be identical across the network.

4.1.7.2 NonBeacon (w/ Coordinator)

A device is configured as a Coordinator by setting the CE (Coordinator Enable) parameter to “1”. Coordinator power-up is governed by the A2 (Coordinator Association) parameter. In a Coordinator system, the Coordinator can be configured to use direct or indirect transmissions. If the SP (Cyclic Sleep Period) parameter is set to “0”, the Coordinator will send data immediately. Otherwise, the SP parameter determines the length of time the Coordinator will retain the data before discarding it. Generally, SP (Cyclic Sleep Period) and ST (Time before Sleep) parameters should be set to match the SP and ST settings of the End Devices.

Association is the establishment of membership between End Devices and a Coordinator. The establishment of membership is useful in scenarios that require a central unit (Coordinator) to relay messages to or gather data from several remote units (End Devices), assign channels or assign PAN IDs. An RF data network that consists of one Coordinator and one or more End Devices forms a PAN (Personal Area Network). Each device in a PAN has a PAN Identifier [ID (PAN ID) parameter]. PAN IDs must be unique to prevent miscommunication between PANs. The Coordinator PAN ID is set using the ID (PAN ID) and A2 (Coordinator Association) commands. An End Device can associate to a Coordinator without knowing the address, PAN ID or channel of the Coordinator. The A1 (End Device Association) parameter bit fields determine the flexibility of an End Device

during association. The A1 parameter can be used for an End Device to dynamically set its destination address, PAN ID and/or channel.

To configure a module to operate as a Coordinator, set the CE (Coordinator Enable) parameter to ‘1’. Set the CE parameter of End Devices to ‘0’ (default). Coordinator and End Devices should con-tain matching firmware versions. **NonBeacon (w/ Coordinator) Systems** The Coordinator can be configured to use direct or indirect transmissions. If the SP (Cyclic Sleep Period) parameter is set to ‘0’, the Coordinator will send data immediately. Otherwise, the SP parameter determines the length of time the Coordinator will retain the data before discarding it. Generally, SP (Cyclic Sleep Period) and ST (Time before Sleep) parameters should be set to match the SP and ST settings of the End Devices.

4.1.8 Xbee / Xbee Pro Addressing

Every RF data packet sent over-the-air contains a Source Address and Destination Address field in its header. The RF module conforms to the 802.15.4 specification and supports both short 16-bit addresses and long 64-bit addresses. A unique 64-bit IEEE source address is assigned at the fac-tory and can be read with the SL (Serial Number Low) and SH (Serial Number High) commands. Short addressing must be configured manually. A module will use its unique 64-bit address as its Source Address if its MY (16-bit Source Address) value is “0xFFFF” or “0xFFFE”. To send a packet to a specific module using 64-bit addressing: Set the Destination Address (DL + DH) of the sender to match the Source Address (SL + SH) of the intended destination module. To send a packet to a specific module using 16-bit addressing: Set DL (Destination Address Low) parameter to equal the MY parameter of the intended destination module and set the DH (Destina-tion Address High) parameter to '0'.

4.1.8.1 Unicast mode

By default, the RF module operates in Unicast Mode. Unicast Mode is the only mode that supports retries. While in this mode, receiving modules send an ACK (acknowledgement) of RF packet reception to the transmitter. If the transmitting module does not receive the ACK, it will re-send the packet up to three times or until the ACK is received.

Short 16-bit addresses. The module can be configured to use short 16-bit addresses as the Source Address by setting (MY < 0xFFFE). Setting the DH parameter (DH = 0) will configure the Destination Address to be a short 16-bit address (if DL < 0xFFFE). For two modules to communi-cate using short addressing, the Destination Address of the transmitter module must match the MY parameter of the receiver.

Long 64-bit addresses. The RF module's serial number (SL parameter concatenated to the SH parameter) can be used as a 64-bit source address when the MY (16-bit Source Address) parame-ter is disabled. When the MY parameter is disabled (MY = 0xFFFF or 0xFFFE), the module's source address is set to the 64-bit IEEE address stored in the SH and SL parameters. When an End Device associates to a Coordinator, its MY parameter is set to 0xFFFE to enable 64- bit addressing. The 64-bit address of the module is stored as SH and SL parameters. To send a packet to a specific module, the Destination Address (DL + DH) on the sender must match the Source Address (SL + SH) of the desired receiver.

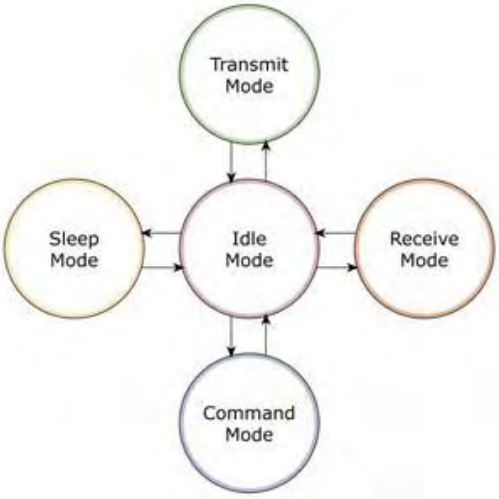
4.1.8.2 Broadcast mode

Any RF module within range will accept a packet that contains a broadcast address. When config-ured to operate in Broadcast Mode, receiving modules do not send ACKs (Acknowledgements) and transmitting modules do not automatically re-sent packets as is the case in Unicast Mode. To send a broadcast packet to all modules regardless of 16-bit or 64-bit addressing, set the desti-nation addresses of all the modules as shown below. Sample Network Configuration (All modules in the network):

- DL (Destination Low Address) = 0x0000FFFF If RR is set to 0, only one packet is broadcast. If RR > 0, (RR + 2) packets are sent in each broadcast. No acknowl-edgements are returned. See also the RR command description.
- DH (Destination High Address) = 0x00000000 (default value)

4.1.9 Modes of Operation

XBee®/XBee-PRO® RF Modules operate in five modes.



4.1.9.1 Idle mode

When not receiving or transmitting data, the RF module is in Idle Mode. The module shifts into the other modes of operation under the following conditions:

- Transmit Mode (Serial data is received in the DI Buffer)
- Receive Mode (Valid RF data is received through the antenna)
- Sleep Mode (Sleep Mode condition is met)
- Command Mode (Command Mode Sequence is issued)

4.1.9.2 Transmit/receive mode

Each transmitted data packet contains a Source Address and Destination Address field. The Source Address matches the address of the transmitting module as specified by the MY (Source Address) parameter (if MY >= 0xFFFE), the SH (Serial Number High) parameter or the SL (Serial Number Low) parameter. The <Destination Address> field is created from the DH (Destination Address High) and DL (Destination Address Low) parameter values. The Source Address and/or Destination Address fields will either contain a 16-bit short or long 64-bit long address. The RF data packet structure follows the 802.15.4 specification.

There are two methods to transmit data:

- Direct Transmission - data is transmitted immediately to the Destination Address
- Indirect Transmission - A packet is retained for a period of time and is only transmitted after the destination module (Source Address = Destination Address) requests the data. Indirect Transmissions can only occur on a Coordinator. Thus, if all nodes in a network are End Devices, only Direct Transmissions will occur.

Indirect Transmissions are useful to ensure packet delivery to a sleeping node. The Coordinator currently is able to retain up to 2 indirect messages.

If the transmission is not a broadcast message, the module will expect to receive an acknowledgement from the destination node. If an acknowledgement is not received, the packet will be resent up to 3 more times. If the acknowledgement is not received after all transmissions, an ACK failure is recorded.

4.1.9.3 Sleep mode

Sleep Modes enable the RF module to enter states of low-power consumption when not in use. In order to enter Sleep Mode, one of the following conditions must be met (in addition to the module having a non-zero SM parameter value):

- Sleep_RQ (pin 9) is asserted and the module is in a pin sleep mode (SM = 1, 2, or 5)
- The module is idle (no data transmission or reception) for the amount of time defined by the ST (Time before Sleep) parameter.

The SM command is central to setting Sleep Mode configurations. By default, Sleep Modes are dis-abled (SM = 0) and the module remains in Idle/Receive Mode. When in this state, the module is constantly ready to respond to serial or RF activity.

4.1.9.4 Command mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming characters are interpreted as commands. Two Command Mode options are supported:

- AT Command Mode
- API Command Mode

Reference guide for Xbee features and characteristics are provided through Digi [12], manufacture of Xbee module.

4.2 Cool Term – Configuring Xbee Series 1 Modules

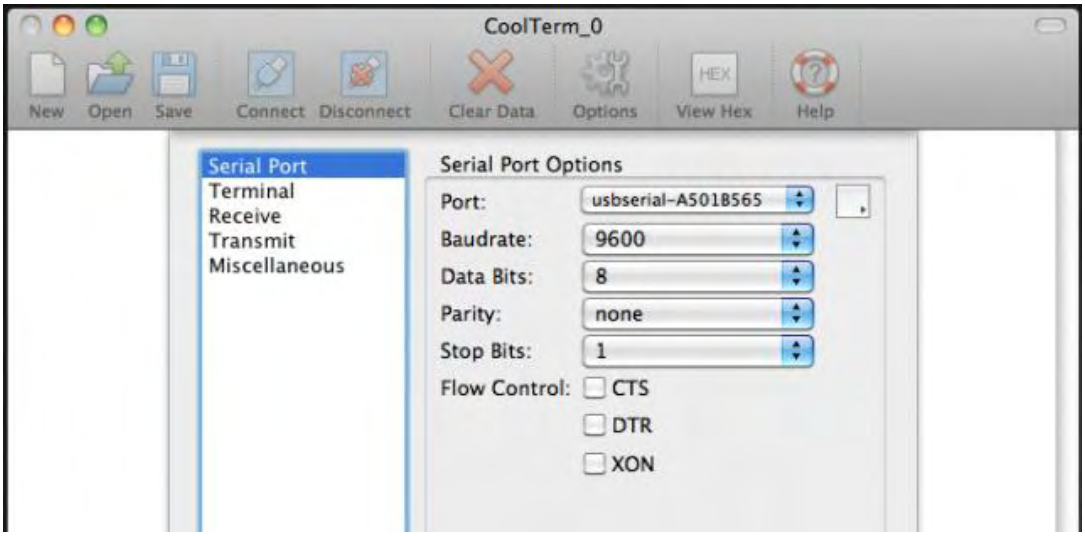
CoolTerm [13] is free software that helps you to configure Xbee modules and to enable communication between two or more Xbee interfaces. Using a serial connection from your computer, the text you type to one XBee will be wirelessly transmitted to the other XBee, which will send the text via serial to your other computer.

In order to communicate with Xbee modules you need a USB Explorer [14] and to mount over it the Xbee module that you want to parameterize.

To use the XBee Explorer, you may need to install FTDI drivers onto your computer, which will let your computer talk via serial to the board.

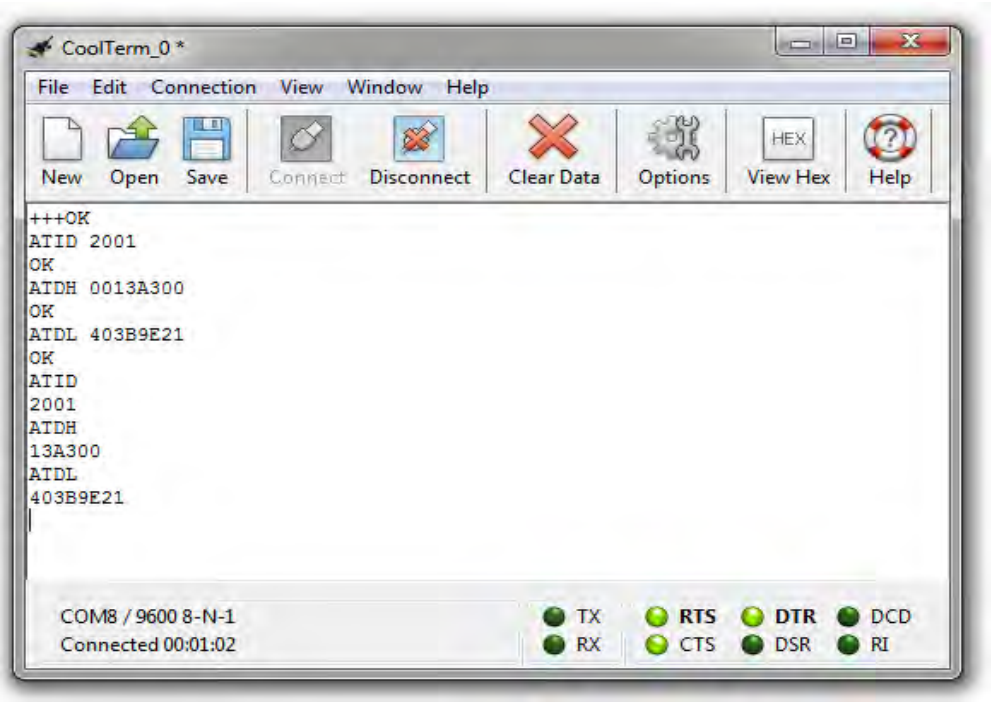
After mounting Xbee module over USB Explorer and connect the last to your computer you can open CoolTerm program and to connect through serial to the desired available device you need to configure. Follow the steps below in order to connect it to your device:

- Open CoolTerm and click “Options”
- Choose the serial port that starts with “usbserial…” If you’re not sure which serial port to select, look at the list of available ports and then disconnect the XBee Explorer from your computer. Click “Re-Scan Serial Ports” and check which serial port disappeared. Reconnect your XBee Explorer, click “Re-Scan Serial Ports” and choose that port.
- Be sure Baudrate is set to 9600.
- Be sure Data Bits is set to 8.
- Be sure Parity is set to “none.”
- Be sure Stop Bits is set to 1.
- In the list on the left side of the options window, click “Terminal.”
- Be sure “Local Echo” is activated. This will allow you to see what you’re typing into the terminal.



- Click OK to save those settings and close the Options window.
- Click “Connect” on the toolbar.
- You should see “Connected” on the status bar at the bottom of the window.
- Type “+++” to enter command mode. You should see the reply “OK.”
- Type in each command followed by its parameter and hit enter.
- You can verify the setting by typing the command without a parameter.
- Your settings aren’t saved yet! Type ATWR to save the settings.
- Here’s how the terminal session will look, starting with the “+++” to enter command mode.

```
+++
OK
ATID 2001
OK
ATMY 1
OK
ATDH 0
OK
ATDL 2
OK
ATID
3001
ATMY
1
ATDH
0
ATDL
2
ATWR
OK
```



In order to enable communication between two Xbee Series 1 modules you need to configure properly the network id, Xbee's address and destination high and low addess.

- **PAN ID:** PAN stands for Personal Area Network. This is a unique identifier for your network. XBees assigned to a particular PAN ID can only communicate with each other. This lets you set up separate networks in the same location.
- **MY Address:** This is the source address for an XBee, it's a unique address for this particular radio. It's how other radios will send messages to it.
- **Destination address high:** This represents the first half of the address we want to talk to. XBee radios can have a 64-bit address, so this is the higher 32-bit part of that address number. Since we don't need so many addresses, we'll set this to 0 and only use the low setting.
- **Destination address low:** This is the address we'll use to locate the other XBee. Make sure it matches the ATMY setting of the XBee you want to talk to.

An example of proper configurations for two Xbee modules will be :

Xbee 1

Function	Command	Parameter
PAN ID	ATID	3001 (any address from 0 to FFFE will do)
MY Address	ATMY	1
Destination address high	ATDH	0 (indicates a 16-bit address)
Destination address low	ATDL	2

Xbee 2

Function	Command	Parameter
PAN ID	ATID	3001 (any address from 0 to FFFE will do)
MY Address	ATMY	2
Destination address high	ATDH	0 (indicates a 16-bit address)
Destination address low	ATDL	1

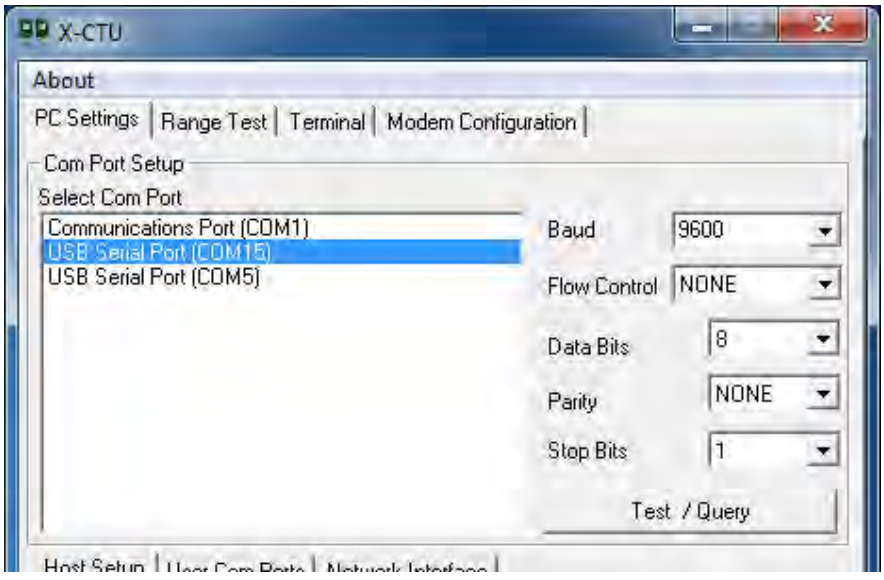
If everything is set up properly, the text that you type in the serial terminal program on the first computer will be relayed to the second computer and appear on its serial terminal screen as well.

If you want to enable communication between more than two XBees, set them all to the same PAN ID (with ATID) and then set the destination low address on the broadcaster to FFFF (with ATDL FFFF). Now, when you type into the broadcaster's terminal, you should see the text appear on all the other terminals.

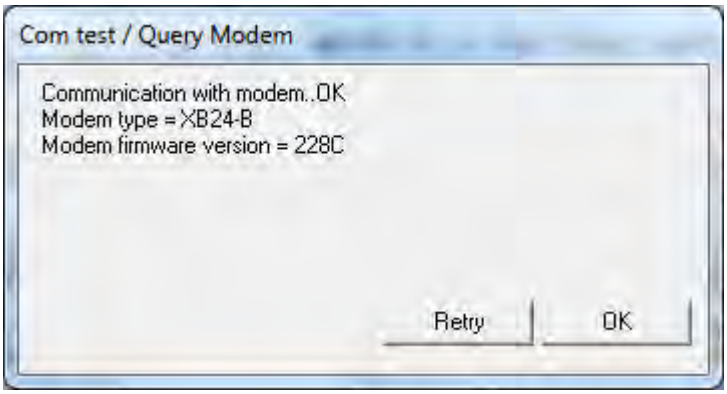
4.3 XCTU– Configuring Xbee Series 2 Modules

A similar to CoolTerm program for configuring Xbee modules is X-CTU [15], provided by Digi. This program provides more features and configures Xbee Series 2 modules (CoolTerm cannot do that).

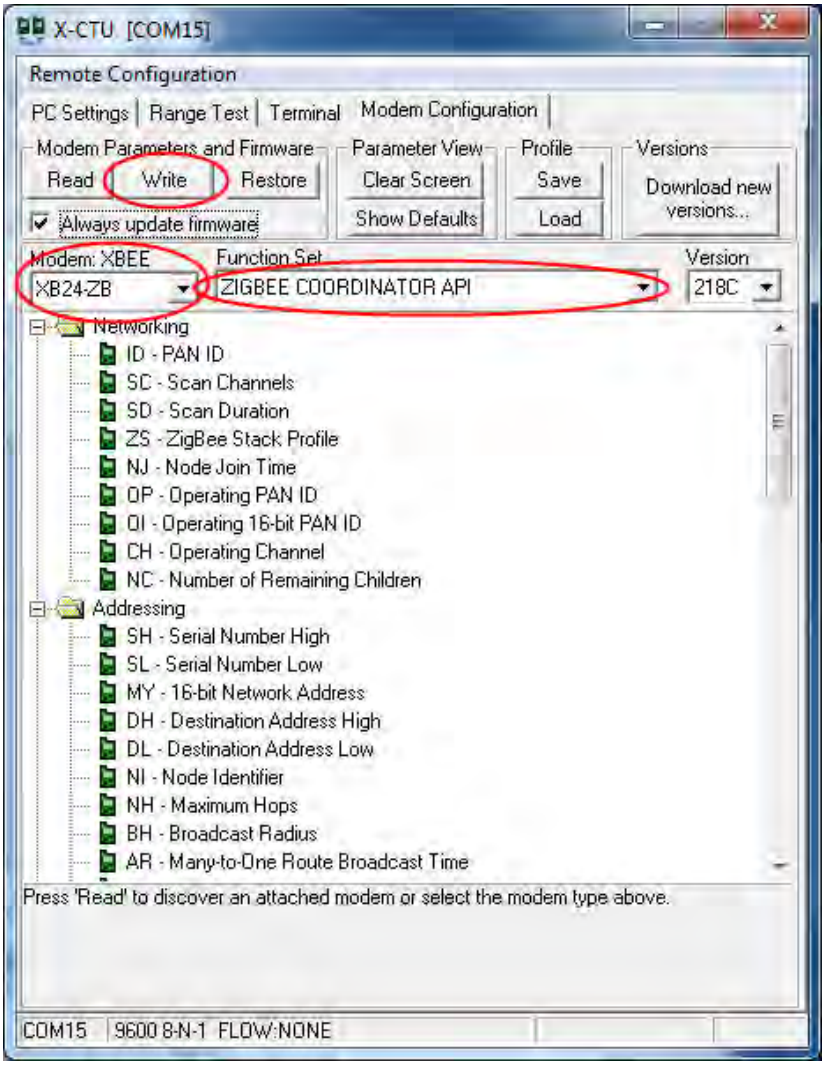
In the same way as with CoolTerm you have to connect your Xbee modules to USB Explorers and to your PC through Serial. After opening X-CTU program and selecting the device you wish to configure



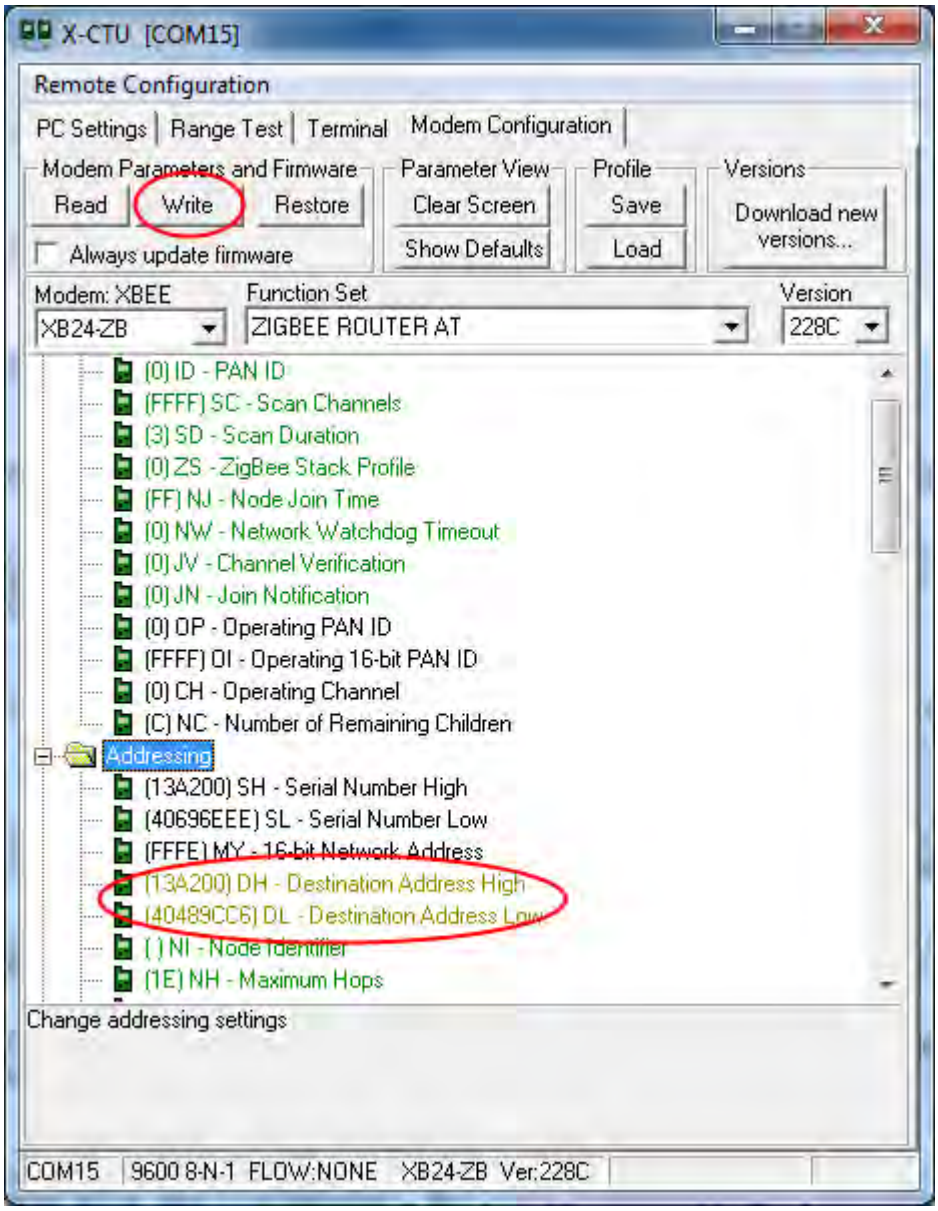
you can also make a test to your Xbee module and it will report you if the connection is OK and the current firmware of your Xbee module.



The special feature that X-CTU provides is that can set your Xbee Series 2 module to act as Coordinator or Router. In tab Modem Configuration you can select the desired option.

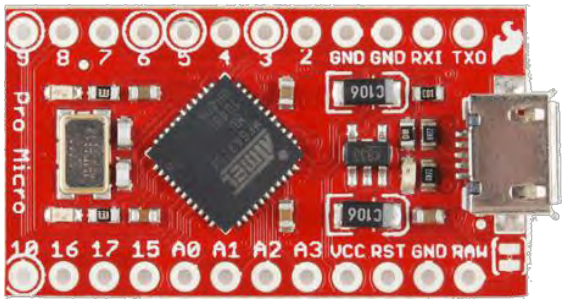


After selecting Coordinator or Router functionality and the mode type you have to press write button in order to load your settings into Xbee module. By pressing the read button you can read the existing configurations.



4.4 Sparkfun's Pro Micro board

SparkFun electronics[17] manufactures and provides Pro Micro [16] board which is an Arduino[18] compatible device. It features Atmel's Atmega 32U4 [19] microcontroller that operates at 8Mhz and 3.3V.



The USB transceiver inside the 32U4 allows adding USB connectivity on-board and doing away with bulky external USB interface.

This tiny little board does all of the neat-o Arduino tricks that you're familiar with: 4 channels of 10-bit ADC, 5 PWM pins, 12 DIOs as well as hardware serial connections Rx and Tx. Due to the fact that is a 3.3V version provides ease of use with many common 3.3V sensors. There is a voltage regulator on board so it can accept voltage up to 12VDC.

Dimensions: 1.3x0.7 inches

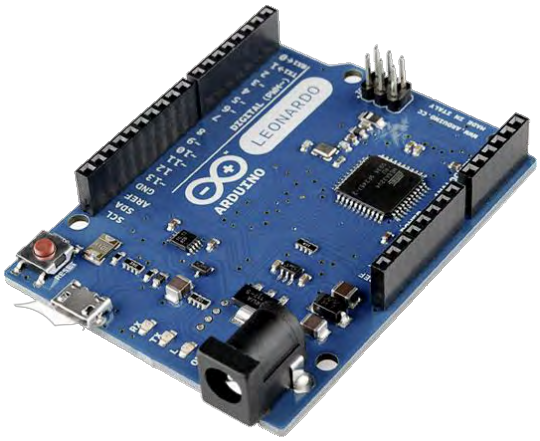
Features:

- ATMega 32U4 running at 3.3V/8MHz
- Supported under Arduino IDE v1.0.1
- On-Board micro-USB connector for programming
- 4 x 10-bit ADC pins
- 12 x Digital I/Os (5 are PWM capable)
- Rx and Tx Hardware Serial Connections
- Smallest Arduino-Compatible Board Yet

4.5 Arduino

Arduino is a single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.

Arduino boards can be purchased pre-assembled or do-it-yourself kits. Hardware design information is available for those who would like to assemble an Arduino by hand. There are sixteen official Arduinos that have been commercially produced to date.



4.5.1 Arduino Hardware

An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I^2C serial bus, allowing many shields to be stacked and used in parallel. Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a simple inverter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.)

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs. These pins are on the top of the board, via female 0.1 inch headers. Several plug-in application shields are also commercially available.

The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board to be plugged into solderless breadboards.

There are a great many Arduino-compatible and Arduino-derived boards. Some are functionally equivalent to an Arduino and may be used interchangeably. Many are the basic Arduino with the addition of commonplace output drivers, often for use in school-level education to simplify the construction of buggies and small robots. Others are electrically equivalent but change the form factor, sometimes permitting the continued use of Shields, sometimes not. Some variants even use completely different processors, with varying levels of compatibility.

4.5.2 Arduino Software

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit makefiles or run programs on a command-line interface.

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

setup(): a function run once at the start of a program that can initialize settings

loop(): a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks an LED on and off. In the Arduino environment, the user might write a program like this:

```
#define LED_PIN 13

void setup () {
 pinMode (LED_PIN, OUTPUT); // enable pin 13 for
digital output
}

void loop () {
 digitalWrite (LED_PIN, HIGH); // turn on the LED
 delay (1000); // wait one second (1000 milliseconds)
 digitalWrite (LED_PIN, LOW); // turn off the LED
 delay (1000); // wait one second
}
```

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground, a convenient feature for many simple tests. The previous code would not be seen by a standard C++ compiler as a valid program, so when the user

clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program.

The Arduino IDE uses the GNU toolchain and AVR Libc to compile programs, and uses avrdude to upload programs to the board.

As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

4.6 Sensors and actuators

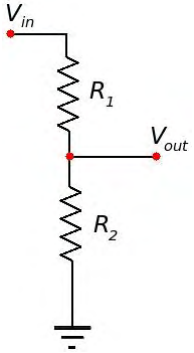
The goal of the developed platform is to collect measurements from the attached sensors and to provide the available measurements. Apart from this, the developed platform can feature various types of actuators and can remotely control them by receiving respective signals.

4.6.1 Integrated Sensors

The sensors that are integrated with the platform are:

- A light intensity sensor:

This is essentially a photo resistor that changes its resistance depending to the light that it receives. To measure the light through this sensor we have designed a simple voltage divider, where R2 is a 10KOhm fixed resistor and R1 is the photo resistor (the light sensor), and we are sampling in the Vout pin.



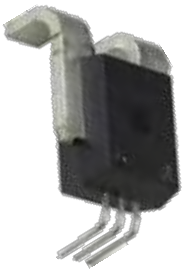
- A human presence sensor:

Passive Infrared sensor [29] can detect motion since it measures infrared (IR) light radiation from objects in its field of view.

- A digital temperature and humidity sensor :

SHT11 [30] digital humidity and temperature sensor is the all-round version of the reflow solderable humidity sensor series that combines decent accuracy at a competitive price. The capacitive humidity sensor is available up to high volumes and as every other sensor type of the SHTxx family, it is fully calibrated and provides a digital output. It implements 2-wire protocol for serial communication with microcontrollers.

- A power consumption sensor :



The Allegro ACS756 [31] family of current sensor ICs provides economical and precise solutions for AC or DC current sensing in industrial, automotive, commercial, and communications systems. The device consists of a precision, low-offset linear Hall circuit with a copper conduction path located near the die. Applied current flowing through this copper conduction path generates a magnetic field which the Hall IC converts into a proportional voltage.

- Electrical Relays:

Electrical relays can control electrical appliances by turning on/off their status.

4.6.2 Additional Compatible Sensors

Many other sensors are compatible with Arduino devices and can be connected to the developed platform. Some of them are:

PH Sensor

Magnetic field

Location Sensor - GPS

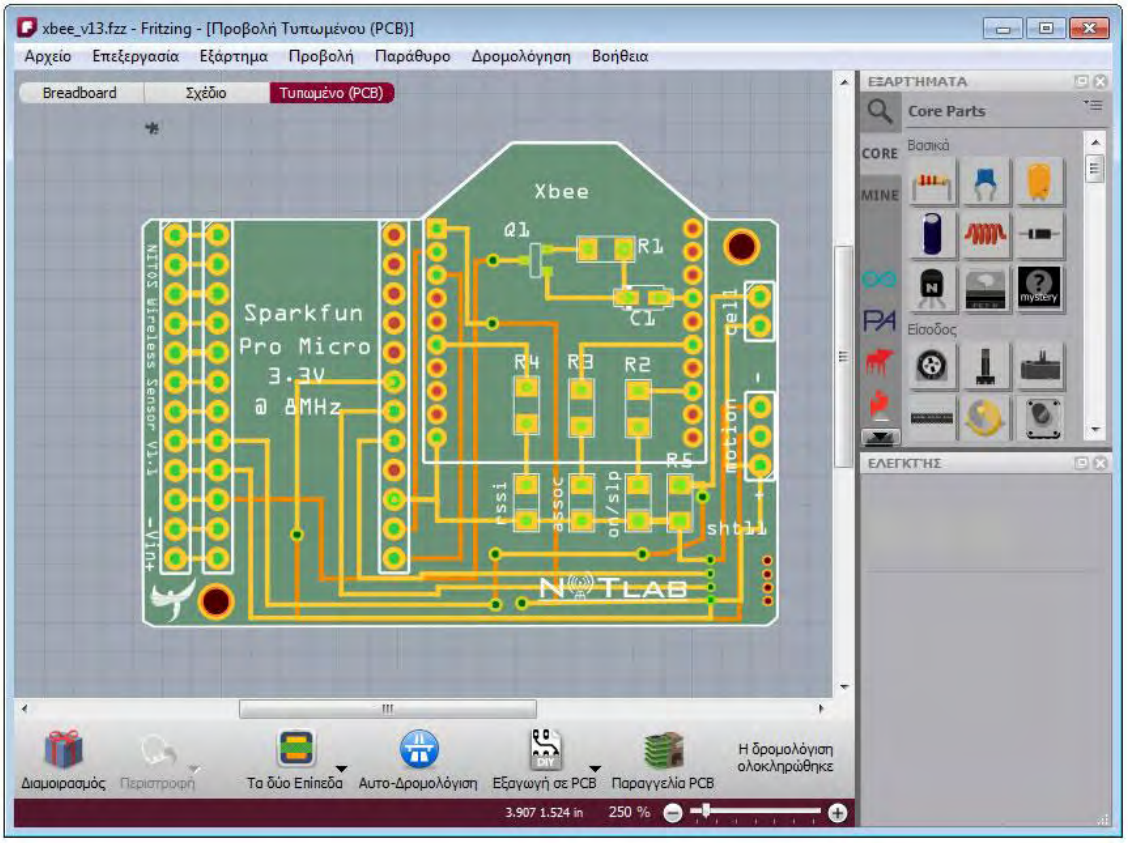
Camera

Air Quality Sensors - Gas Sensor

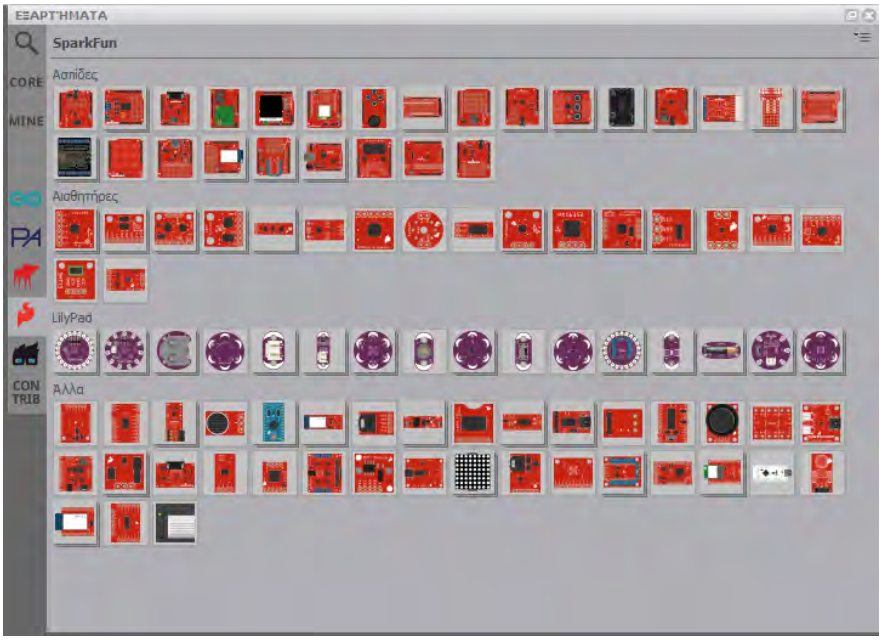
Door Status Sensors

4.7 Fritzing Software

Fritzing [20] is an open-source hardware initiative to support designers, artists, researchers and hobbyists to work creatively with interactive electronics. Through the developed program users can create their electronic circuits and then to design their Printed Circuit Board in order to assemble it and create their project.

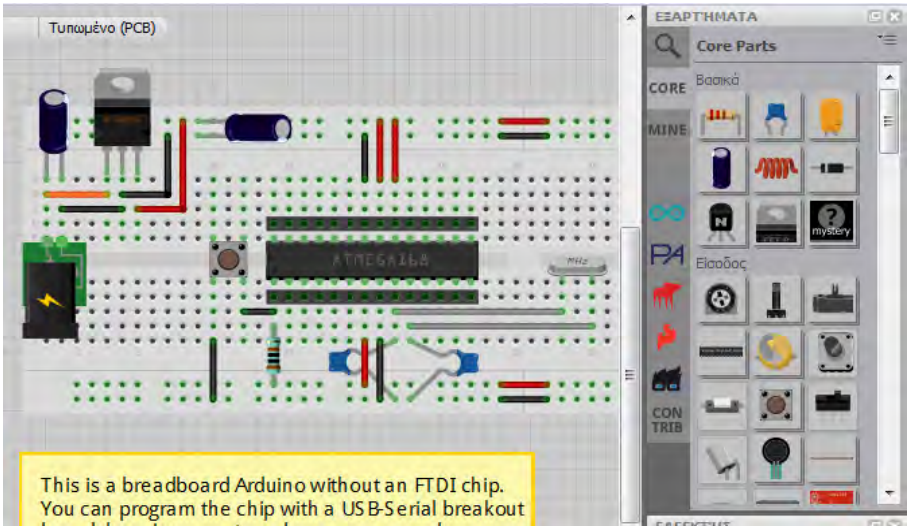


Apart from providing common electronics elements, Fritzing software also provides a variety of Arduino compatible boards. Exploiting Fritzing users can add to their circuit usual components such as resistors capacitors transistors but the key novelty of Fritzing is the existing layout of the most Arduino platforms. This feature helps users to easily create their own designs by using the existing formats and pinouts, without having to design from scratch each platform they need. Below are depicted the available components from Sparkfun and Arduino while also the core components (the common).

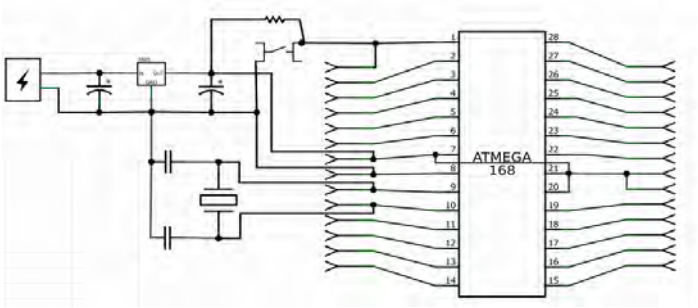


Fritzing software features three different tabs:

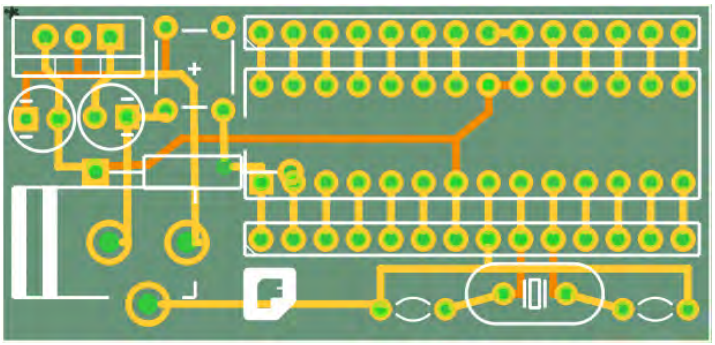
- **Breadboard**: Users can add to their breadboard every element and connect it wherever, in the same way they do with their real accessories.



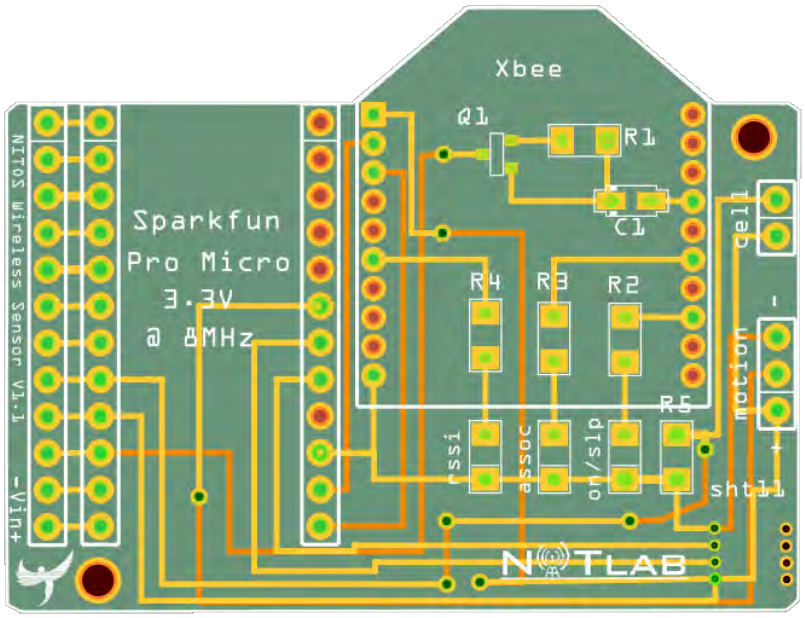
- **Design**: Users can see the electronic design (the connection among the picked elements)



- **PCB**: Users can design the route of the wires and the location of each element on the board. After designing the PCB board users can order their boards to be printed through Fritzing Fab [21].



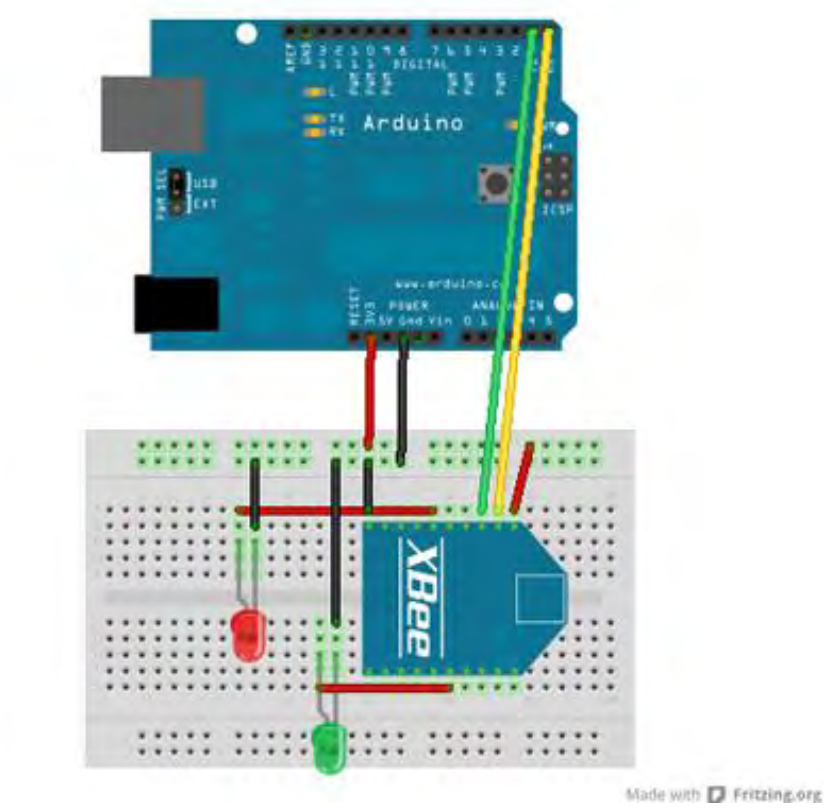
In the context of the present project the PCB for wireless sensors was designed through Fritzing software and was fabricated through Fritzing Fab [21]. Below is a schematic of the PCB as it is shown through Fritzing software at PCB tab.



Below are depicted some PCB boards that was fabricated through Fritzing Fab.

4.8 NITOS Wireless Sensor Platform

The first step for creating NITOS Wireless Sensor platform was to enable communication between a simple Arduino board and an Xbee module. This was accomplished by connecting Arduino's and Xbee's serial ports and by powering Xbee module though Arduino's 3.3V power pin. Below an image with the aforementioned connection is depicted.



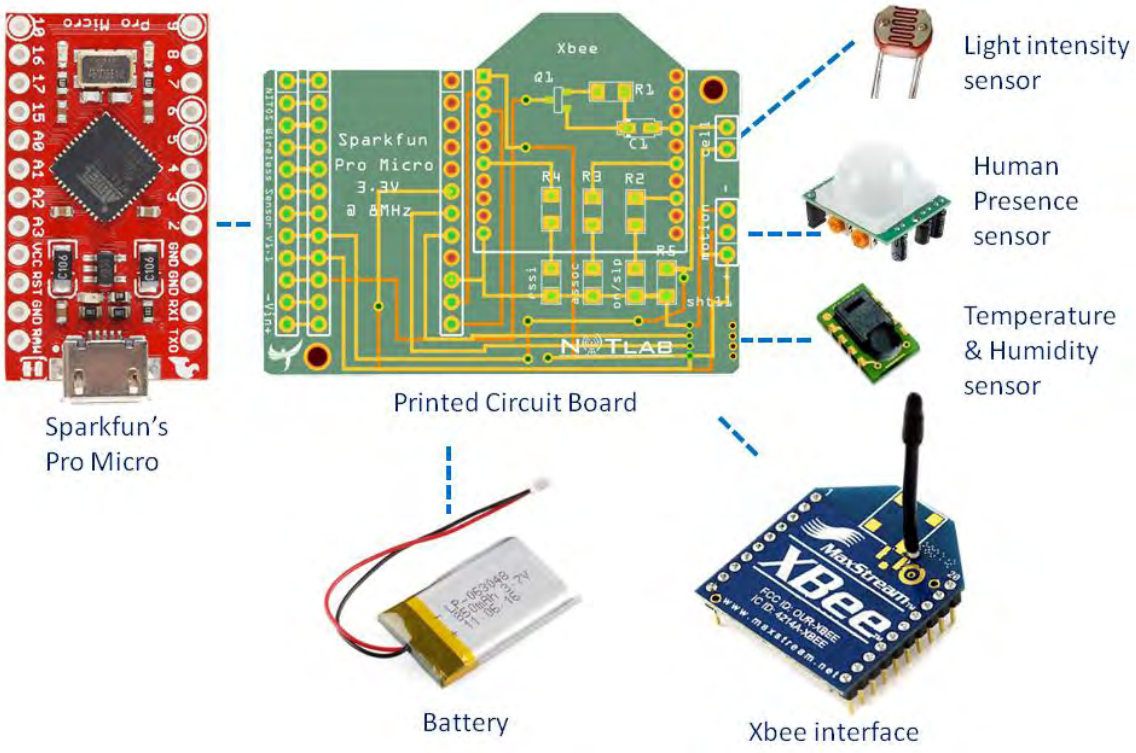
Even if Arduino Leonardo is working at 5V and Xbee module at 3.3V, communication through serial was successful. The correct way to connect two devices that work in different voltage levels is by using a logic converter.

After establishing connection between Arduino Leonardo and Xbee module, we could send simple packets over the air, and we used an other Xbee module attached to a USB Explorer and connected to a PC with CoolTerm program running, in order to receive and monitor the packets that was captured. The Arduino code that was running on Arduino Leonardo in order to send simple "Hello" packets every second is shown below.

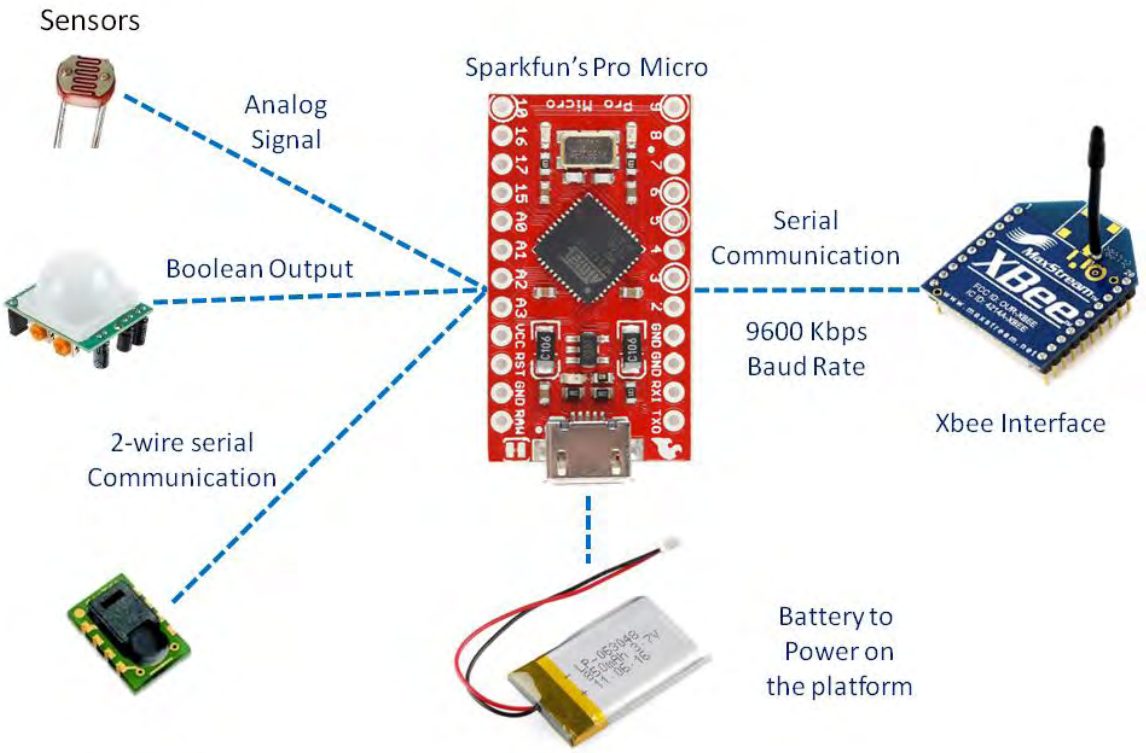
```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Hello XBee Network!");
  delay(1000);
}
```

Arduino Leonardo board was used only for testing purposes and not for being the board that will be used for the wireless sensor platform. Towards the direction of creating a small-size wireless sensor platform we choose to use SparkFun's Pro Micro. Pro Micro is a very small device that has the same characteristics and abilities with the rest Arduino compatible devices. Apart from that, Pro Micro operates at 3.3V which is ideal for serial communication with Xbee modules without the need of logical converter. In addition to the aforementioned, Atmega 32U4 integrates two serial ports in contrast with the most Arduino platforms that feature only one serial connection. This is a key issue, since Pro Micro can support simultaneously a serial connection at the USB socket and a serial connection at the Rx and Tx pins that has on board. The idea here is that Pro Micro can be re-programmed without having to be de-attached from the existing Xbee that might be connected to the serial port in order to flash new firmware onto it, because it uses a different socket for this procedure.

The final sensor platform is a mixture of different components. The basic component is the Pro Micro, which is the "heart" of the platform, since it controls all the functionalities on the board. Another key component is the wireless interface that enables communication with the respective gateway. This is the Xbee module manufactured by Digi. In order to receive some measurements we have chosen to add a digital environmental sensor that measures temperature and humidity, a light intensity analog sensor and a human presence sensor. All of these components are assembled in the produced by Fritzing board, plus some resistors some sockets and some leds that are used for indication purposes. Below are images showing the developed wireless sensor platform and the components that we have used.

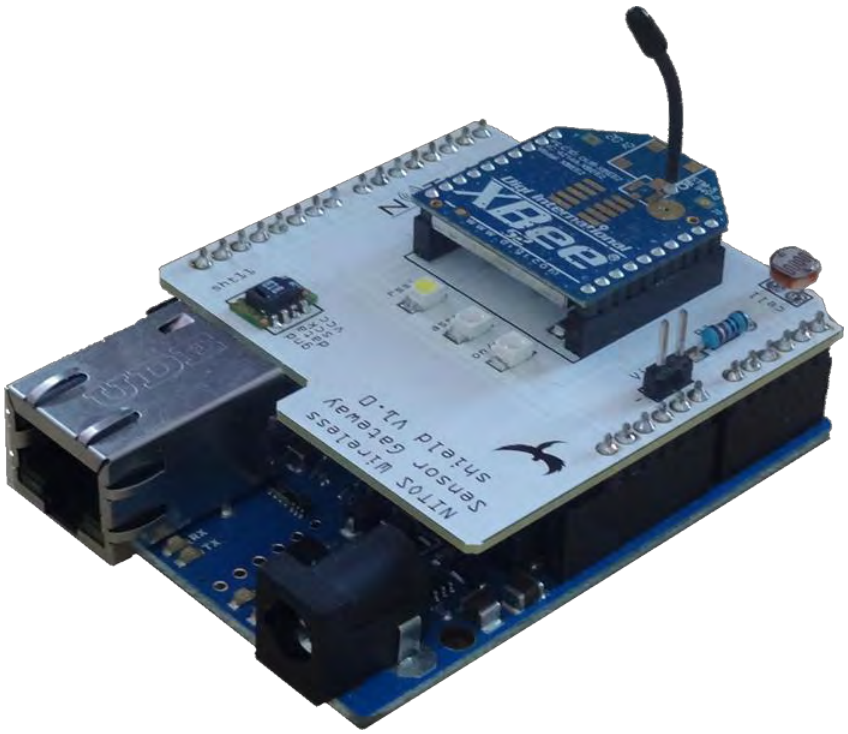


Each sensor on the board is connected with the Pro Micro and communicates with the microcontroller. Atmega 32U4 can implement a variety of existing protocols that commercial sensors supports, as a result to be able to communicate and monitor the values of each attached sensor. After monitoring the values of each sensor, it sends the measurements through 9600 baud rate serial channel to the Xbee module for wireless transmission. The developed sensor platform can be powered either by a simple lithium battery or by a PC through USB connection. Below the architecture of the developed sensor is shown.



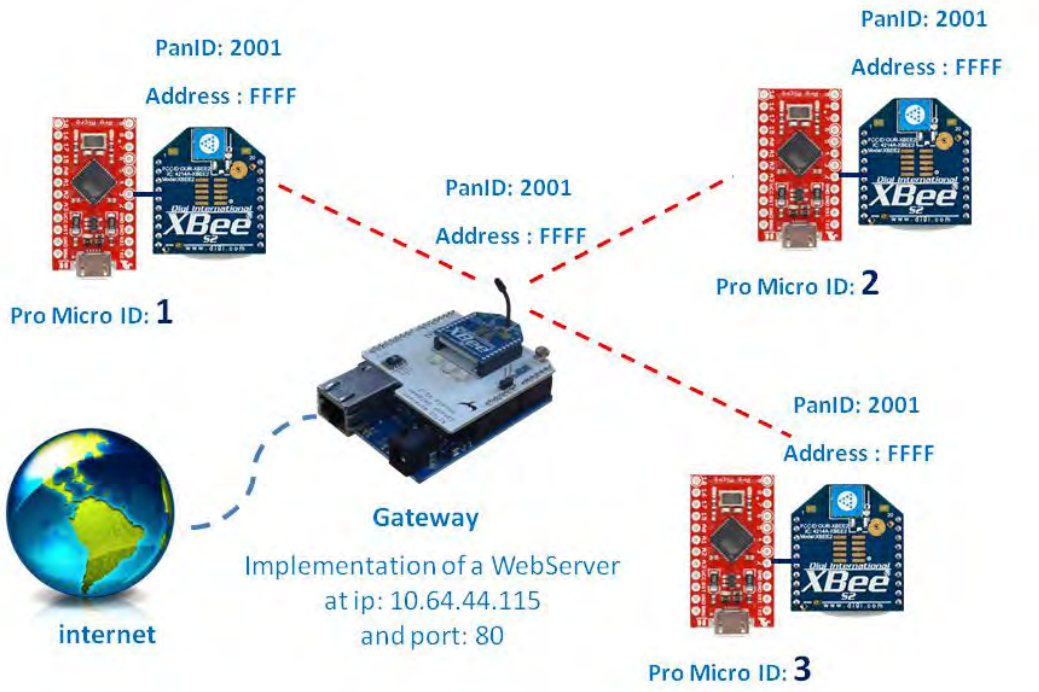
4.9 NITOS Wireless Gateways

In order to enable communication with the developed wireless sensor platform we have created a respective gateway node. To this aim we have exploited Arduino Etherner board w/o PoE [22], which is a common Arduino board that features an Ethernet Interface that communicates with the Arduino Microcontroller(Atmega328p) through Wiznet 5100 [23]. The developed gateway can be programmed through Arduino IDE to act as a Web Server that serves http requests. Exploiting this feature we can send http requests through the Web and ask to learn the values of the sensors that are deployed around the gateway and belong to its network. Then the gateway node will send a packet through its Xbee interface, asking to receive the desired measurement. After receiving the desired measurement it will forward it back to the user through http response.



4.10 Demo SetUp

For demonstration purposes we have set up a simple demo, where each mote reports periodically its measurements to the respective gateway. To do so, we have configured every Xbee interface to the broadcast address “FFFF” and each Pro Micro has a unique id. When Pro Micro creates a packet to be sent, indicates also the id of the mote, which is its own id. In this way when the gateway receives a packet can understand from where it received the packet from. Below the network set up is depicted.



Wireless sensors devices, can also receive commands when they operate. To enable this feature we have programmed Pro Micro devices to listen all the time the air for any possible transmission and to transmit periodically its measurements. For demonstration purposes we have developed an application that blinks the integrated leds of Pro Micro when it receives a respective command. Below is the respective Arduino code for Pro Micro:

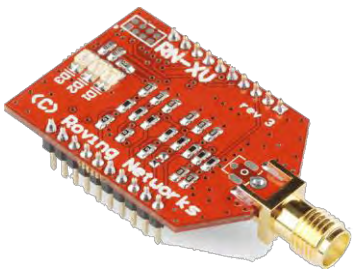
```
#include <SPI.h>
#include "Sensirion.h"

int RXLED = 17;
int r=0;
```

```
const uint8_t dataPin  =  2;
const uint8_t clockPin =  3;

float temperature;
float humidity;
float dewpoint;

Sensirion tempSensor = Sensirion(dataPin, clockPin);



void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);

}

void loop() {
  int sum_photo =0;
  int photo =0;
  double  div_photo=0.0;



  while(Serial1.available()){

            char getData = Serial1.read();
            Serial.println(getData);



      if(getData == 'a'){

            digitalWrite(RXLED, HIGH);
            Serial1.println("done");

      }else if(getData == 'b'){

            digitalWrite(RXLED, LOW);
            Serial1.println("done");

      }

        if(getData == 'c'){

        TXLED1;
        Serial1.println("done");

      }else if(getData == 'd'){

            TXLED0;
            Serial1.println("done");

      }

  }
```

```
  if(r > 500){

      tempSensor.measure(&temperature, &humidity, &dewpoint);
      sum_photo = 0;

  for (int i = 0; i < 5; i++) {
       photo = analogRead(3);
       sum_photo += photo ;

  }
  div_photo = (int) sum_photo / 51.15;

  Serial1.print("Temp: ");
  Serial1.print(temperature);
  Serial1.print("C Hum: ");
  Serial1.print(humidity);
  Serial1.print("Light: ");
  Serial1.println(div_photo);
  r=0;

  }

  delay(1);
  r++;
}
```

4.11 WiFi Enabled Sensor motes

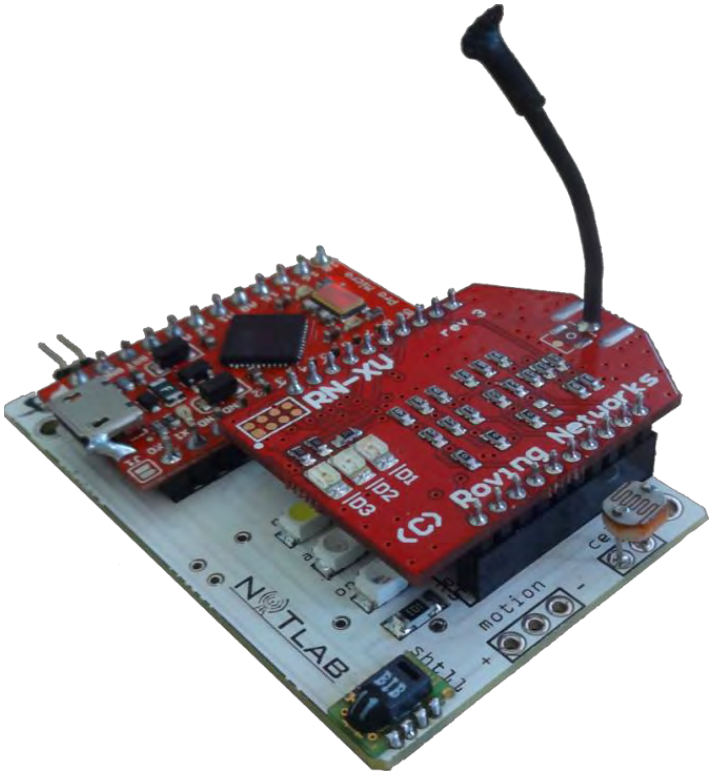
The developed platform has the flexibility to feature various types of wireless interfaces. The basic modules that we have used are Zigbee and 802.15.4 compatible devices (Xbee S1 and Xbee S2). Except of these modules that implements low power protocols and are ideal for distributed sensing applications we can add modules that supports different technologies such as WiFi and Bluetooth. In order to use WiFi technology we have exploited WiFly module that Roving provides. This module implements WiFi protocol and communicates with the Pro Mico in the same way as the Xbee module.



To configure WiFly module we need to use CoolTerm and a USB Explorer. We can give commands to the module in order to configure the settings we want, such as the network id we desired to be connected to, the password of the network, etc. Below basic commands to configure a WiFly module are shown.

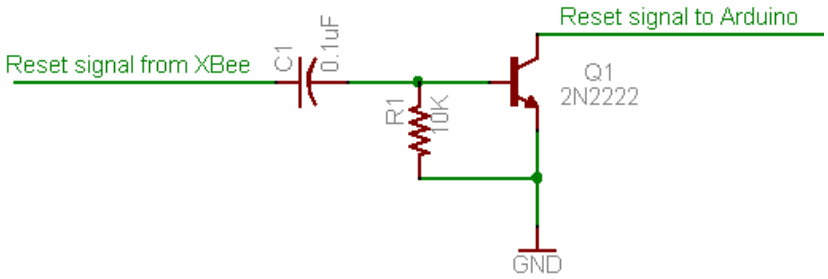
```
scan     #returns available AP's

set wlan auth 4    #for wpa2-psk encryption

set wlan phrase "password"

set wlan ssid "yourssid"

save

reboot

get wlan   #returns configurations settings
```

In order to communicate with the WiFi enabled sensor we need to build a Web Server into the Pro Micro. The WiFly module can obtain an ip address and will operate in a specific port. Below is an image of the WiFi enabled sensor.



4.12 Over the Air Programming

The developed wireless sensor platform features a special circuit that enables a unique functionality. It permits Pro Micro to be flashed over the air with new firmware. Exploiting Xbee I/O pins we have configured pin 6, to act as output ant to activate the below circuit, which act as a switch.

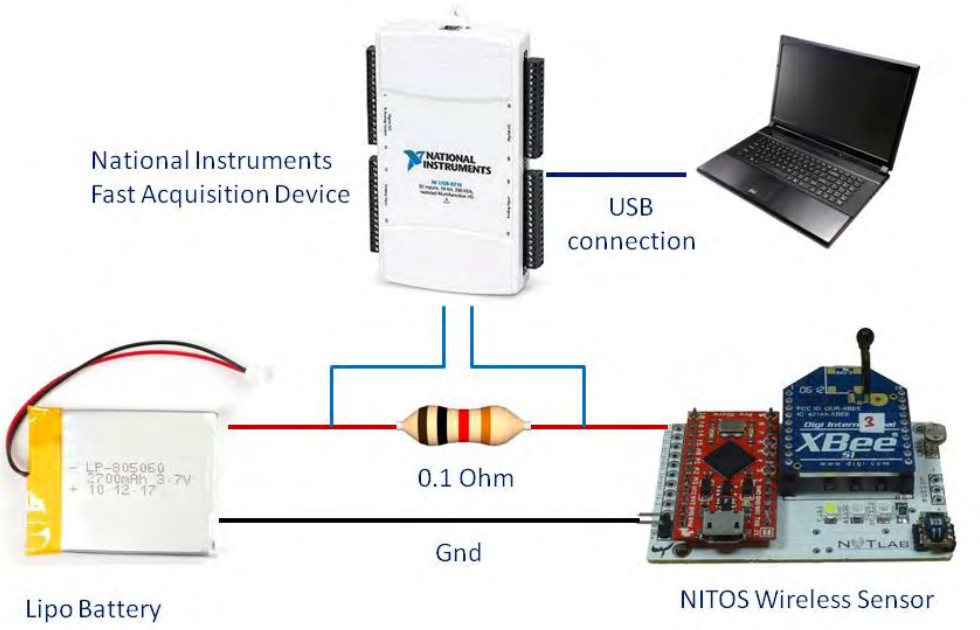


When this circuit is activated (a High signal comes in the left side of the circuit) it resets the connected to it Arduino board. We have integrated this circuit to the developed wireless sensor and we can reset Pro Micro board through Xbee module (just by sending a respective command to enable the pin 6). In this way, we can send our new firmware wirelessly to Pro Micro, through Xbee module and then to send a reset command to Xbee, in order to reset Pro Micro. After resetting, Pro Micro will start running the new firmware we just uploaded.

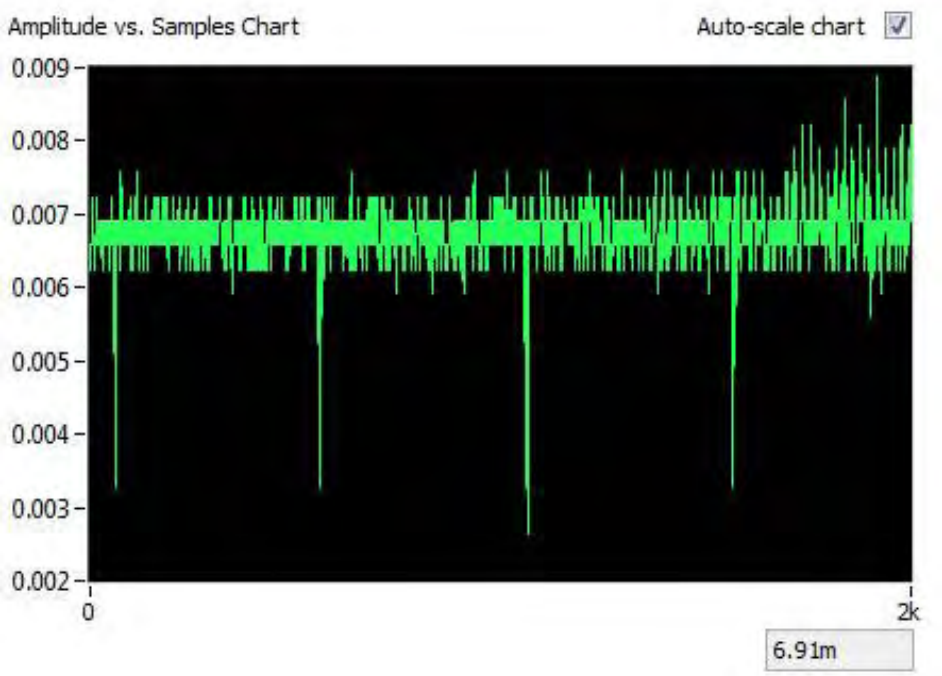
4.13 Power Consumption Measurements

Towards the direction of measuring the power consumption of the developed wireless sensor we have created a special setup where we monitor the voltage on a current shunt resistor that is mounted subsequent to the positive cable of power supply. The current shunt resistor has a very low resistance (0.1 Ohm) and we sample it continuously, in order to read the voltage that on it. To this aim we relied on a high acquisition device manufactured by National Instruments that can achieve high sampling rate.

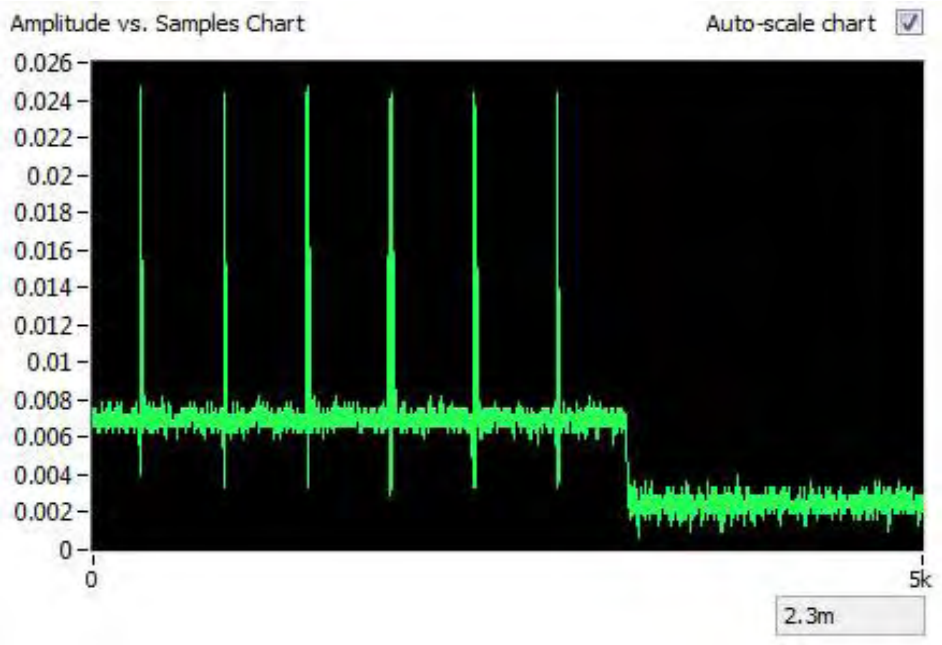
By reading the voltage values National Instruments device, can export the current that the sensor platform consumes, since we define the value of the current shunt resistor. In this way it is simple also to calculate the power (in watts) that is consumed by multiplying the current with the voltage that this platforms needs to operate, which is 3.3V in our case. Below is depicted the wiring we used to measure the current.



The results we obtained from this process were that, when a simple Xbee transmits there isn't any difference in the power consumption. In addition, when Xbee Pro transmits we can easily observe the difference in the power consumption. The reason is that a simple Xbee transmits either with 1mW or 2mW, while Xbee Pro transmits with 60mW.



Simple Xbee power consumption



Xbee Pro power consumption

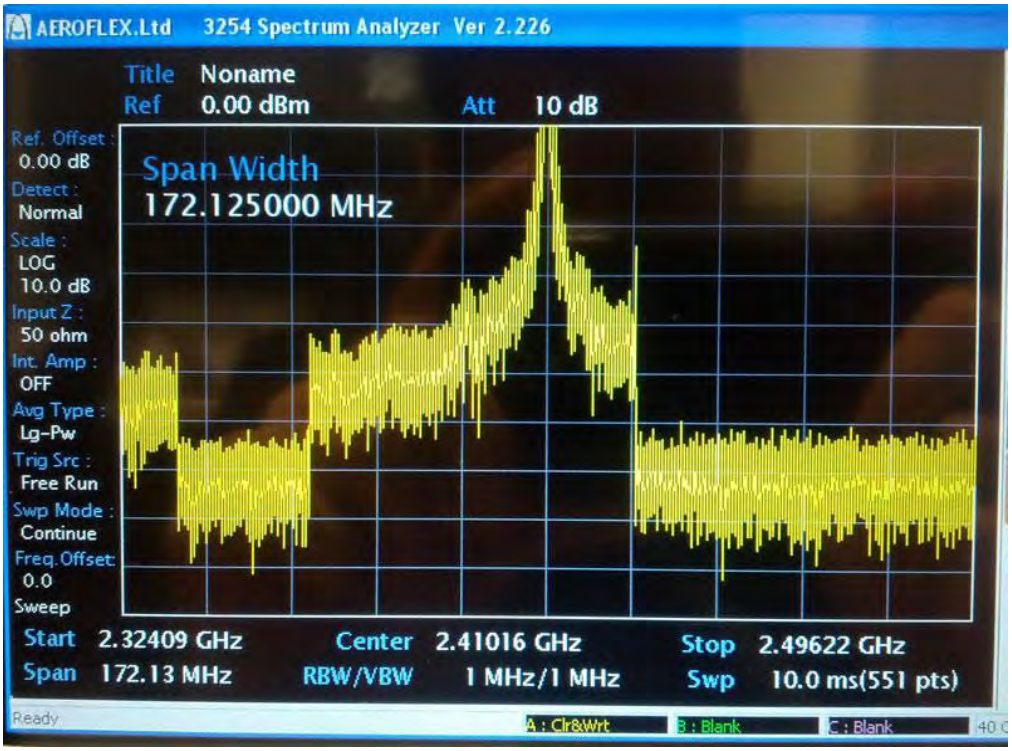
The power consumption of Simple Xbee module is about 70mA.

The power consumption of Xbee Pro module is the same, when it's not transmitting and is about 240mA when it transmits. In addition the power consumption only of Pro Micro Board is about 20mA as it can shown in the last part of Xbee Pro power consumption

graph, since we removed the Xbee Pro from the platform during the measurement process.

4.14 Spectrum Measurements

In order to monitor the transmission frequency of the developed platform we used a spectral spectrum analyzer by Aeroflex.



We can observe in the above image that the center transmission frequency is on 2.41016 GHz, which is the channel we have configured Xbee during the set up. (channel C, frequency 2.410 GHz)

5 References

[1] 802.15.4 vs ZigBee : http://www.sensor-networks.org/index.php?page=0823123150
[2] DSSS modulation : http://en.wikipedia.org/wiki/Direct-sequence_spread_spectrum
[3] CSMA-CA: http://en.wikipedia.org/wiki/CSMA_CA
[4] GTS : http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?reload=true&arnumber=4455149
[5] Institute of Electrical and Electronics Engineers : http://www.ieee.org/index.html
[6] ZigBee protocol : http://www.zigbee.org/
[7] Wireless HART : http://www.hartcomm2.org/index.html
[8] ISA-SP100 : http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891
[9] IETF IPv6 LoWPAN : http://ietfreport.isoc.org/ids-wg-6lowpan.html
[10] Digi Mesh : http://www.digi.com/technology/digimesh/
[11] AODV Routing Protocol : http://en.wikipedia.org/wiki/AODV
[12] Digi : http://www.digi.com/
[13] CoolTerm program : http://freeware.the-meiers.org/
[14] USB Explorer : https://www.sparkfun.com/products/8687
[15] X-CTU program : http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities
[16] Pro Micro Board : https://www.sparkfun.com/products/10999
[17] Sparkfun Electronics : https://www.sparkfun.com/
[18] Arduino : http://www.arduino.cc/
[19] Atmega 32U4 Microcontroller : http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Dev/Arduino/Boards/ATMega32U4.pdf
[20] Fritzing Software : http://fritzing.org/
[21] Fritzing Fab : http://fab.fritzing.org/fritzing-fab
[22] Arduino Ethernet Board : http://arduino.cc/en/Main/ArduinoBoardEthernet
[23] Wiznet 5100 Ethernet Controller : http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=5&cate2=7&cate3=26&pid=1011
[24] Intel mote 2 : http://docs.tinyos.net/tinywiki/index.php/Imote2
[25] Tmote Sky : http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf
[26] Waspmote : http://www.libelium.com/products/waspmote/
[27] Atmel Corporation : http://www.atmel.com/
[28] Xbee radio interfaces : http://www.digi.com/xbee/
[29] PIR Motion Sensor : https://www.sparkfun.com/products/8630
[30] sensirion sht11 sensor : http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Hu

midity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf
[31] Alegro ACS756 Currrent Sensor :
http://www.digchip.com/datasheets/parts/datasheet/029/ACS756.php