
# FPGA-driven system for safe radiation monitoring

**GROUP ID: IT005**

# 1. Design Introduction

*Please give some general information of your design, e.g. purpose of the design, application scope, and targeted users. Please also include a detailed description of why you used Altera FPGA devices to do the design.*

Con il diffondersi in molti Paesi degli impianti per la produzione di energia nucleare e l'invecchiare di quelli già esistenti, è aumentato il rischio di incidenti e di conseguenza la possibilità di fughe radioattive, basti citare il recente disastro di ***Fukushima***.

Per stabilire la gravità dei danni, dato l'alto rischio di contaminazione, si necessita di sistemi di monitoraggio delle radiazioni che coinvolgano il meno possibile l'intervento diretto di esseri umani.
In questo contesto è inserito il nostro progetto, che si propone di creare una piattaforma per il monitoraggio delle radiazioni, pilotata tramite scheda ***FPGA***.

È necessario a tal fine disporre di strumenti per la misurazione della radioattività (dose equivalente di radiazione) come ***contatori Geiger***. A tal proposito sono nati, proprio alla luce dei recenti eventi, numerosi progetti *"open hardware"* che hanno contribuito alla diffusione di contatori a basso costo, che necessitano però di ulteriore hardware realizzato in proprio o sfruttando soluzioni commerciali esistenti sul mercato.

L'utilizzo dell'FPGA è una valida alternativa alla realizzazione di hardware home-made, grazie anche alla flessibilità e alla riprogrammabilità via software di questi dispositivi. Questo aspetto è un grande vantaggio se si tiene conto della rapida evoluzione di sensori e in generale delle soluzioni open hardware.

L'idea generale è quella di utilizzare un sensore per la detection delle radiazioni, implementando l'interfaccia necessaria all'elaborazione dei segnali, trasmessi dal sensore, direttamente su FPGA.

I dati ottenuti vengono poi post-processati dalla scheda e mostrati all'utente. Tali dati potrebbero poi essere memorizzati in un supporto di memoria al fine di conservare un andamento storico dei livelli di radioattività riscontrati.

Questa idea apre la strada a molteplici possibili estensioni, come ad esempio la possibilità di installare il dispositivo di monitoraggio su di una piattaforma mobile

radiocomandata al fine di evitare l'esposizione di operatori ad eventuali rischi di contaminazione.

Tali estensioni potrebbero coinvolgere una seconda FPGA, utilizzata dall'operatore a distanza di sicurezza, preposta a ricevere tramite comunicazione wireless il valore della misura rilevata dal dispositivo mobile, ed eventualmente trasmettere i dati e il log ad un computer che possa visualizzarne l'andamento graficamente.

Tra i possibili contesti di utilizzo nei quali si può sfruttare il sistema di monitoraggio delle radiazioni (inviando la piattaforma radiocomandata dotata di FPGA e contatore Geiger) oltre ai già citati problemi di ***eventi eccezionali*** quali ***disastri naturali*** o incidenti in centrali nucleari, rientrano anche:

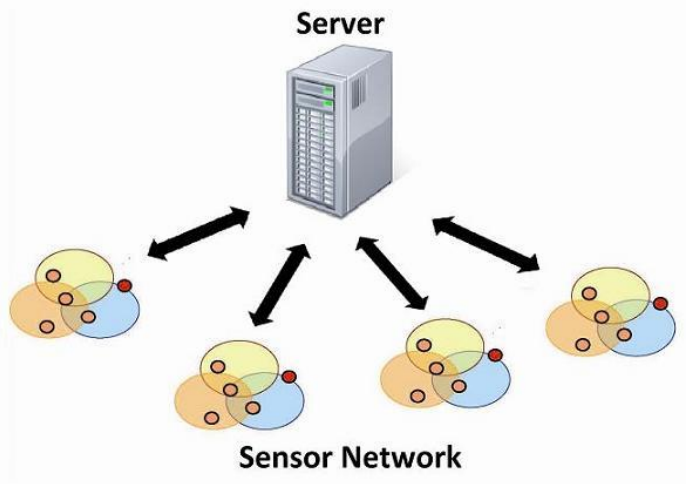
- ***controlli doganali*** - ricerca di tracce di radioattività tra le merci in transito o in container provenienti da Paesi con scarsi controlli di sicurezza;
- ***contrasto allo smaltimento illecito di rifiuti speciali*** - verifica della presenza in discariche ordinarie di rifiuti pericolosi o speciali, al fine di contenere il fenomeno dello smaltimento illecito di rifiuti provenienti da ospedali o industrie;
- ***controllo dei materiali di costruzione*** - spesso nel rimodernamento di stabili o fabbriche possono essere presenti materiali radioattivi o pericolosi;
- ***applicazioni militari*** - mappatura di territori sensibili o interessanti dal punto di vista bellico.

Numerosi sono anche i possibili sviluppi futuri come l'aggiunta di ulteriori sensori di sicurezza, telecamere per un miglior controllo remoto e/o chip GPS per la tracciabilità.

## 2. Function Description

*Please give detailed information to show the functionality of your design and how to implement it.*

Il nostro progetto si propone di creare una piattaforma per il monitoraggio delle radiazioni, pilotata tramite scheda ***FPGA***. L'idea di base si pone nel contesto delle reti di sensori wireless.




Una rete di sensori wireless, o **w**ireless **s**ensor **n**etwork (**WSN**), è una tipologia di rete avente architettura distribuita, realizzata da un insieme di dispositivi autonomi in grado di rilevare dati provenienti da misure effettuate nell'ambiente circostante, e di collaborare tra di loro. Tali dispositivi sono caratterizzati dal basso costo di produzione, dimensione e peso ridotti e bassi consumi. In genere ogni sensore è poi provvisto di un processore on-board, che permette di effettuare una elaborazione preliminare dei dati, che poi vengono inviati ai cosiddetti **nodi sink**, nodi "coordinatori" che raccolgono i dati dai diversi sensori e li trasmettono ad un server o ad un calcolatore. Il tipo di trasmissione dei dati è generalmente un *wireless a corto raggio*. I dati rilevati da tali sistemi sono utilizzati in diversi campi tra cui:

- ambito medico-sanitario (body-networks per il monitoraggio dei parametri fisiologici);
- domestico (reti di sensori per la domotica);
- rilevamento del traffico;

- ambientale (rilevamento dei livelli di inquinamento ed entità dei fenomeni atmosferici e monitoraggio di siti a rischio contaminazione) come nel nostro caso.

Poiché ci concentreremo sulla misurazione del livello di radiazioni ionizzanti in ambienti aperti o chiusi, la tipologia di sensore utilizzata è il contatore Geiger.

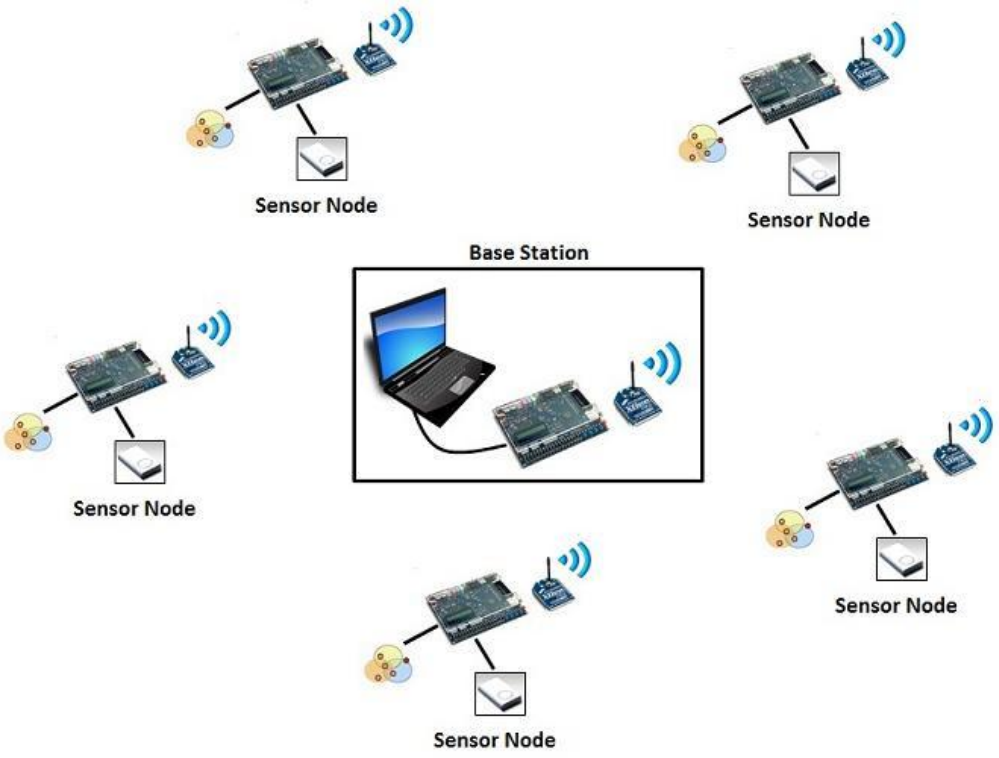
Esso è costituito da una camera a deriva, in cui è presente un gas che, a contatto con una radiazione ionizzante, genera una coppia ione-elettrone che, tramite una reazione a valanga, si moltiplica dando vita ad un impulso elettrico rilevato da un circuito elettronico. Contando quanti impulsi si verificano in un dato intervallo di tempo, è possibile stimare l'entità della radiazione presente nell'ambiente. L'unità di misura della dose equivalente di radiazione è il **Sievert/ora [Sv/h]**, che misura la quantità di energia assorbita per unità di massa nell'unità di tempo (dove 1 Sv = 1 J/kg ) , ed è legata al conteggio degli impulsi (**cpm - count per minute**) da un'equazione lineare i cui coefficienti dipendono dal tipo di dispositivo utilizzato per le misure. Poichè il Sv è un'unità di misura molto grande, nella pratica si usano i suoi sottomultipli; in condizioni normali si ha una radiazione di fondo dell'ordine dei micro-Sievert (uSv).
La copertura della misurazione è fortemente localizzata (il contatore va posto vicino all'oggetto da monitorare) e la durata della misurazione è inversamente proporzionale all'errore statistico di misura: questo è dovuto al fatto che le particelle ionizzanti colpiscono il sensore in istanti di tempo casuali trasportando un'energia media che determina appunto il valore di radiazione misurata. Di conseguenza maggiore è la durata della misurazione, più accurati saranno i risultati poiché l'errore statistico viene ammortizzato nel tempo (generalmente vengono consigliati tempi dell'ordine dei minuti o decine di minuti).

Il contatore da noi utilizzato deriva da un progetto open-hardware giapponese, ***Pocket-Geiger***, nato nel 2011 subito dopo il disastro di Fukushima, per dare la possibilità alla popolazione di poter disporre di uno strumento a basso costo per verificare di persona la pericolosità degli ambienti e degli oggetti con cui essi entravano in contatto.

Sensore Pocket-Geiger

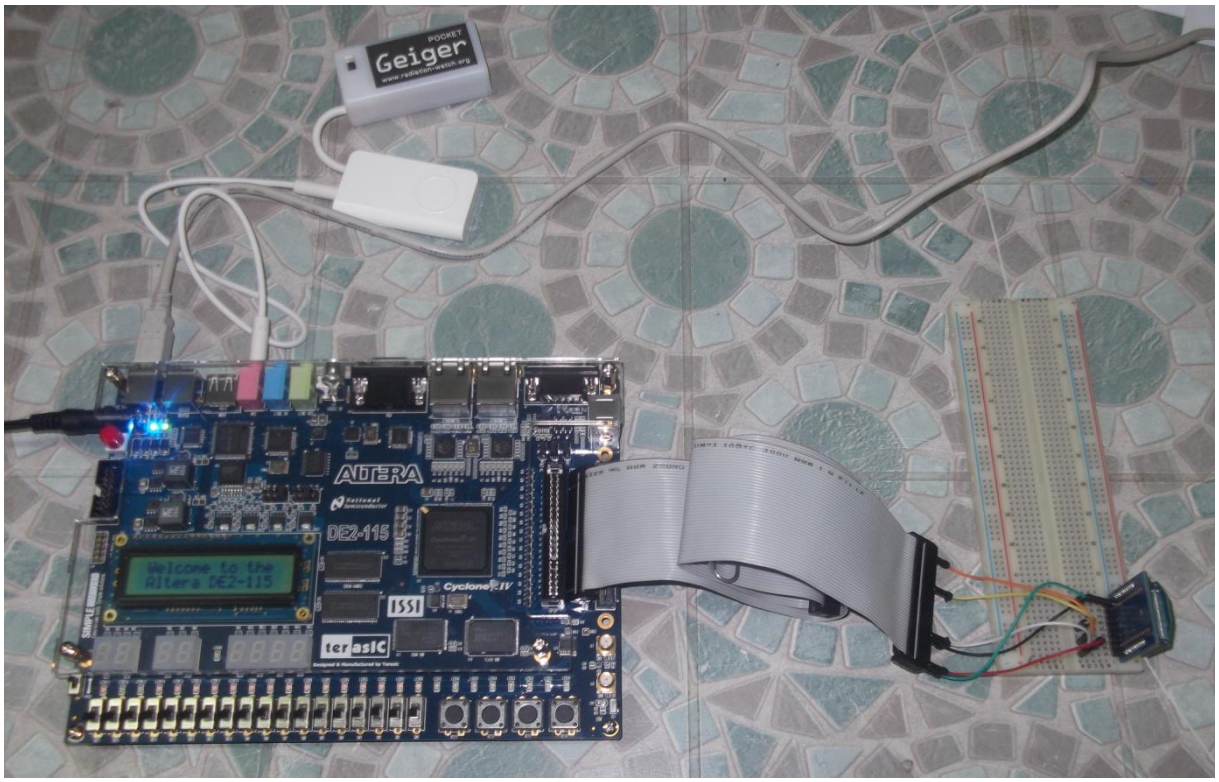
Esso viene fornito insieme agli schemi di progetto in modo tale da poter essere interfacciato ad hardware esterno (che disponga di un'unità di elaborazione).
Per il corretto funzionamento è quindi necessario hardware e software aggiuntivo. La stessa *Pocket-Geiger* consiglia di utilizzare un tablet/smartphone come dispositivo di elaborazione e una loro applicazione proprietaria (chiusa e a pagamento) come software per l'analisi dei dati grezzi ottenuti dal sensore. Tali dati vengono inviati al dispositivo di elaborazione, attraverso un segnale audio proveniente dal contatore, tramite il classico jack audio da 3.5 mm.
Nel nostro progetto l'idea è quella di utilizzare una scheda FPGA come hardware per l'acquisizione ed elaborazione del segnale audio, utilizzando anche il **soft-processor Nios II**, sul quale verrà fatta girare una applicazione per l'analisi **DSP real time**.

La nostra rete di sensori è costituita da una "configurazione base", eventualmente estensibile, composta da un nodo-sensore e un nodo-sink che rappresenta la stazione di raccolta dati.

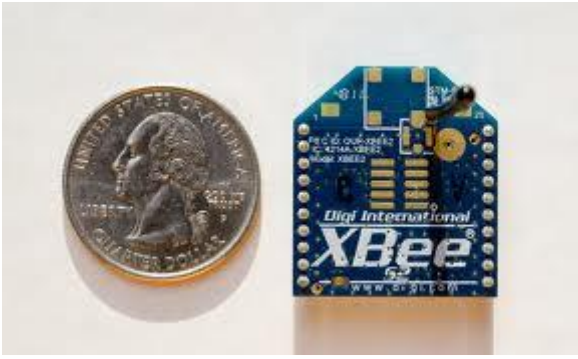



Modello della rete di sensori progettata

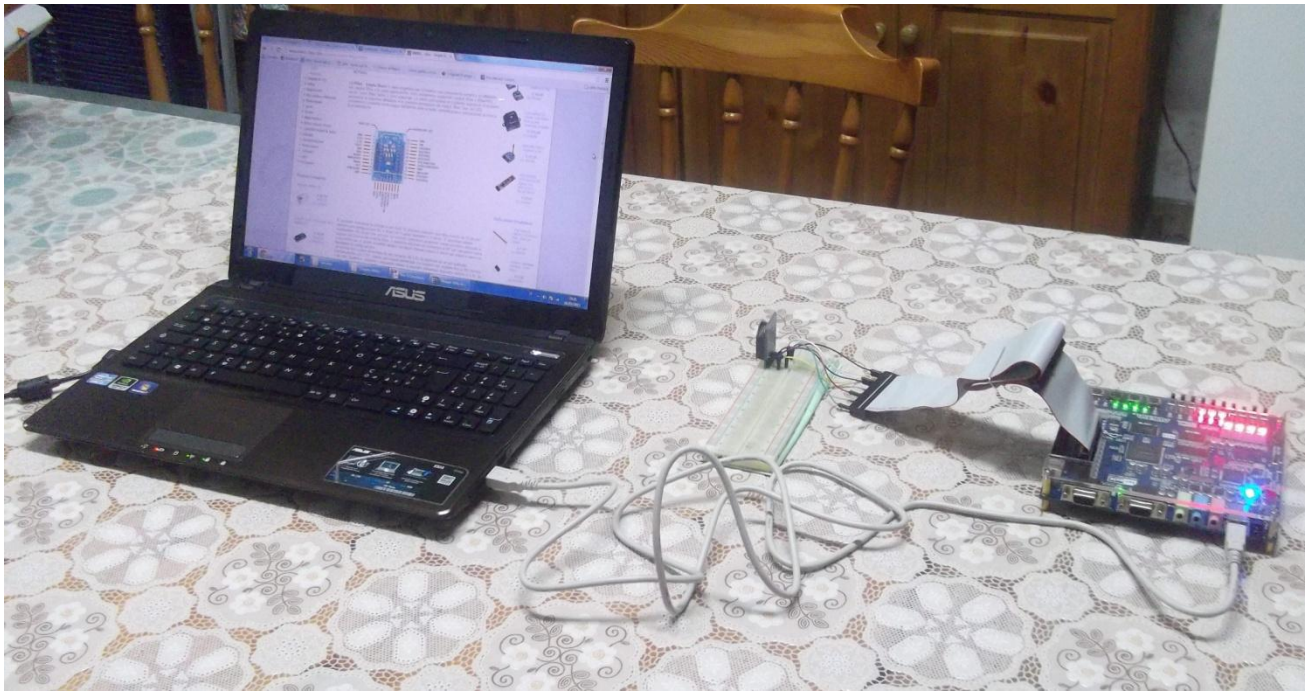
Il nodo dotato di sensore è costituito dal contatore geiger che è collegato tramite jack audio ad una scheda FPGA (**Altera DE2-115**), la quale elabora i dati al fine di generare il valore della dose equivalente di radiazioni, e tramite moduli wireless invia tale valore alla stazione base.



Per la comunicazione tra i due nodi sono stati usati dei **moduli Xbee**, moduli wireless (2.4 GHz) a basso consumo energetico ma che garantiscono alte prestazioni unitamente ad una comunicazione sicura (comunicazione autenticata e criptata). Essi si basano sul protocollo di comunicazione **Zigbee** (**IEEE 802.15.4**), e risultano di facile utilizzo in quanto la comunicazione avviene in modalità "black box" (potremmo immaginarli come il prolungamento wireless di un cavo seriale).



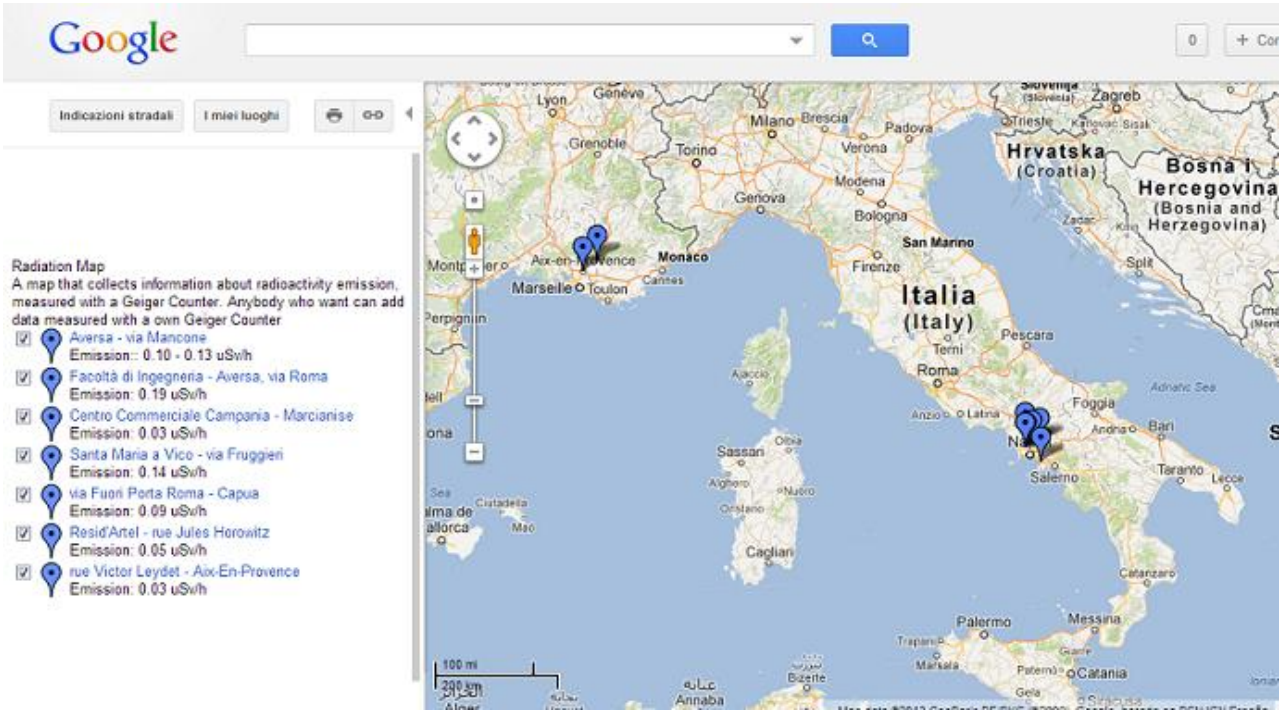

La stazione base è costituita da una seconda scheda FPGA (**Altera DE1**) collegata ad un computer, la quale ha il compito di ricevere i dati raccolti dal sensore (via wireless tramite XBee), mostrando sul display della board i risultati della misurazione, per poi trasferire i dati ottenuti ad un elaboratore che provvederà a garantire il requisito di persistenza dei dati, salvandoli in un file di log.




Stazione base

La stazione base viene controllata tramite un'unica applicazione (da noi realizzata) che ha il compito di:

- comunicare con l'FPGA: inizializzazione, caricamento del bitstream sull'FPGA, protocollo di handshake con il nodo-sensore e trasferimento dei dati dall'FPGA ad un file di log;
- eseguire un'operazione di geolocation che consente di realizzare il discovery della posizione (latitudine e longitudine) in cui è stata registrata la misura;
- effettuare l'upload su una Google-map condivisa e liberamente consultabile da qualsiasi dispositivo dotato di connessione internet (smartphone / tablet / pc / ecc.).




Alcune misurazioni effettuate con il sistema finale ed automaticamente caricate dalla nostra applicazione

Disponibile all'indirizzo: http://goo.gl/W5I45

Ovviamente, per sua natura, una rete di sensori ha come caratteristica principale quella di essere facilmente estensibile. Per tale motivo sono stati utilizzati i moduli wireless Xbee, i quali, anche in caso di più di 2 nodi, possono essere organizzati in un'unica rete sicura ed autenticata. Ciò non significa che non possano essere utilizzati anche differenti moduli wireless o diverse tipologie di sensori aggiuntivi come ad esempio web-cam, rilevatori di campi elettrici, GPS ecc. Inoltre, il fatto di avere un'unica mappa condivisa ed accessibile via internet da tutti i dispositivi e da eventuali altre stazioni base, è un elemento di vantaggio nell'operazione di **data-fusion**.

## 3. Performance Parameters.

*Please enumerate some performance parameters that the design needs to reach. If possible, please compare the actual performance realized in your design with the design parameter, and then appraise the function of Altera FPGA devices in the design.*

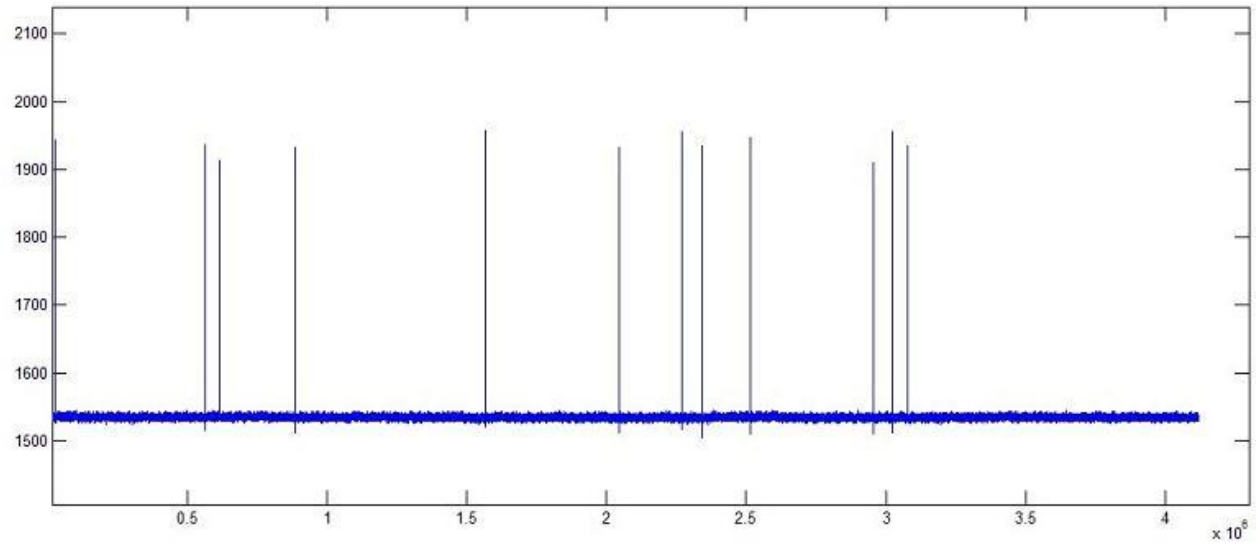
Il requisito principale del sistema proposto è dato dalla necessità di dover lavorare in **real-time** eseguendo task differenti contemporaneamente, pilotando e interfacciandosi con dispositivi hardware eterogenei. Tali problemi sono concentrati soprattutto nei nodi-sensori della rete, in quanto le operazioni delegate alla stazione base, ovvero l'autenticazione di tutti i nodi-sensori effettuando un handshake per preparare la comunicazione, la ricezione dei dati e l'aggiornamento del file di log e del display a 7 segmenti, non necessitano di essere eseguite simultaneamente.
Nello specifico, possiamo identificare i task di competenza dell'FPGA del **nodo-sensore** nei seguenti punti:

- Campionamento ed acquisizione del segnale audio proveniente dal sensore (contatore geiger);
- Elaborazione numerica del segnale (DSP - rilevamento degli impulsi e calcolo del valore CPM);
- Gestione della comunicazione con il dispositivo di trasmissione wireless (conversione dei dati per l'adattamento alla comunicazione).

Per ciascuno dei suddetti punti nasce una serie di problematiche da risolvere al fine di rispettare il requisito di esecuzione real-time, tenendo conto che la misurazione deve essere accurata se paragonata a sistemi proprietari esistenti. Per poter validare il secondo requisito, i risultati delle misurazioni effettuate dalla nostra rete di sensori sono stati paragonati, come vedremo più avanti, con quelli ottenuti da un secondo contatore geiger collegato ad hardware e software proprietari.

Per quanto riguarda il campionamento e l'acquisizione del segnale, il problema centrale è quello di dover scegliere opportunamente la frequenza di campionamento dell'ADC. Tale frequenza deve essere sufficientemente elevata da catturare i singoli impulsi, dei quali non è nota a priori né la durata né la frequenza, perché sono legate a fenomeni pseudo-aleatori dipendenti anche dalla realizzazione fisica del sensore. A tal proposito, sono state effettuate delle prove sperimentali preliminari, tramite software Matlab, utilizzando un segnale registrato in un intervallo di tempo sufficientemente grande, il tutto in modalità off-line.

Il segnale registrato ha un andamento che é quello mostrato in figura:




Effettuando diverse valutazioni a differenti frequenze di campionamento, si è ritenuto necessario utilizzare un sampling rate di almeno 32 kHz che è un giusto tradeoff per garantire la detection dei singoli impulsi, e allo stesso tempo per evitare overflow dei buffer e sovraccaricamento del Nios processor.

Per quanto riguarda il secondo punto, ovvero le operazioni di DSP effettuate sul Nios II, per garantire il requisito di esecuzione in tempo reale, sono state utilizzate le funzionalità di multithreading offerte dal sistema operativo real time (RTOS) uC-OS II. Tale OS è risultato molto utile nel gestire il multi-threading insito nel nostro sistema, perché riduce al minimo i tempi di latenza di interruzione e di thread-switching. Inoltre, esso è facilmente invocabile dal Nios tramite opportune primitive. Il multi-threading è necessario in quanto bisogna suddividere le operazioni di digital signal processing e acquisizione dei dati, dalle operazioni di trasmissione wireless, che vengono quindi delegate ad un task differente. I dettagli della ripartizione dei compiti tra i task saranno trattati in maniera approfondita nella sezione 4.

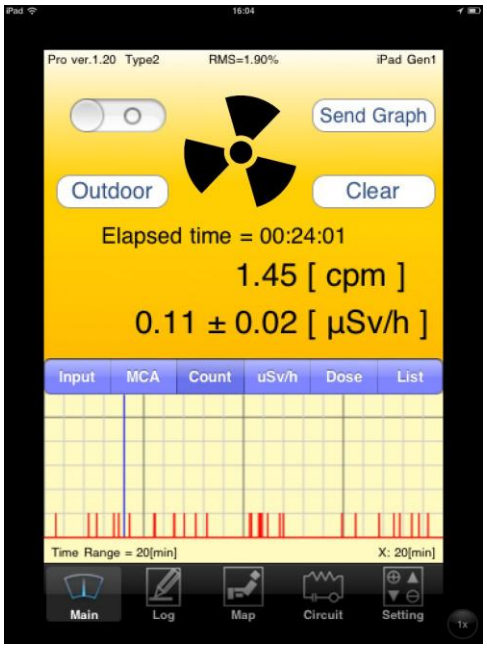
Oltre a fissare la frequenza di campionamento, un ulteriore problema da risolvere per ottenere il corretto rilevamento di tutti gli impulsi è stato la scelta della soglia necessaria a distinguere il segnale utile dal rumore sottostante, ossia il valore della soglia oltre il quale una variazione del segnale di fondo può essere considerata o meno un impulso. Come vedremo nella sezione successiva, in cui saranno mostrati i dettagli dell'algoritmo utilizzato, la soluzione proposta è un algoritmo **double-threshold**, ovvero un algoritmo che fa uso di due soglie differenti al fine di minimizzare gli errori dovuti a possibili oscillazioni degli impulsi, che porterebbero ad un conteggio falsato.

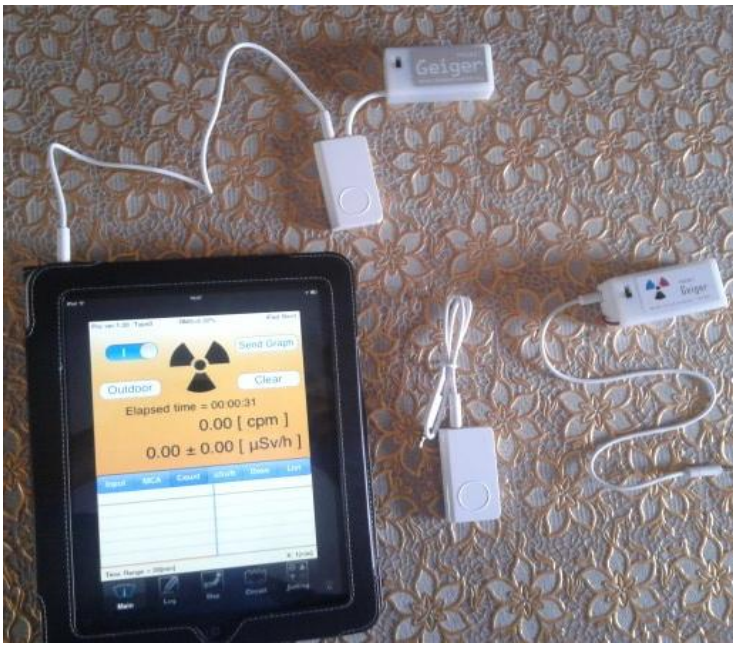
Infine, per quanto riguarda la gestione della comunicazione, quest'ultimo punto viene delegato ad un task dedicato, che adatta i dati ottenuti dalla misurazione e li trasmette una volta al secondo via wireless al nodo della stazione base.

Le caratteristiche delle schede Altera che sono risultate utili ai fini del conseguimento delle performance richieste sono le seguenti:

- **Nios II Fast** - la versione fast del soft-processor ha aiutato nel raggiungimento delle prestazioni richieste, in particolare per la capacità di effettuare operazioni logico-aritmetiche in tempi ridotti, utili per l'analisi DSP;
- **RTOS uC-OS II** - come già detto, questo sistema operativo è particolarmente leggero e veloce, soprattutto nelle operazioni di multitasking;
- **Qsys IP cores** - la presenza di numerosi blocchi Qsys, già disponibili per l'utilizzo, semplifica il design rendendo modulare il progetto e accorciando i tempi di sviluppo. In particolare utili al nostro progetto sono risultati i blocchi per la gestione dell'UART e del campionatore del device audio;
- **Expansion header** - la presenza di una porta di espansione con connettore standard a 40 pin per accedere all'fpga, ci ha permesso di inserire hardware aggiuntivo come i **moduli tx/rx** (Xbee) per la comunicazione wireless, utilizzando per il cablaggio componenti facilmente reperibili sul mercato. Inoltre l'interfacciamento di tali moduli è stato notevolmente semplificato dalla possibilità di poter regolare la tensione di uscita dei pin dell'fpga in modo da renderli compatibili con la tensione di lavoro degli Xbee. Questa caratteristica risulta utile perché assicura la compatibilità con dispositivi eterogenei aventi tensioni di funzionamento diverse.

A questo punto, presentiamo una tabella comparativa tra le misurazioni effettuate utilizzando il nostro sistema e quelle effettuate, contemporaneamente e negli stessi luoghi, mediante la soluzione proposta da PocketGeiger. Tale soluzione consiste in un contatore geiger che, a differenza del sensore utilizzato nel nostro progetto, risulta utilizzabile direttamente da un utente finale che possiede un tablet ed una copia a pagamento del software proprietario.




| Luogo | Durata [s] | PocketGeiger [uSv/h] | Sistema Altera [uSv/h] |
|---|---|---|---|
| Aulario Ingegneria SUN, Aversa | 300 | 0.14 | 0.14 |
| Aulario Ingegneria SUN, Aversa | 150 | 0.12 | 0.14 |
| Centro Commerciale Campania (CE) | 300 | 0.09 | 0.09 |
| Centro Commerciale Campania (CE) | 150 | 0.08 | 0.09 |
| Via Stazio 2, Aversa | 300 | 0.14 | 0.14 |
| Via Fuori Porta Roma 66, Capua | 300 | 0.11 | 0.10 |
| Aulario Ingegneria SUN, Aversa (Potassio) | 300 | 0.20 | 0.21 |

Nella tabella sovrastante sono mostrate solo alcune delle misurazioni di prova effettuate. Come è possibile notare, le misure sono sufficientemente coerenti e tendono a convergere allo stesso valore all'aumentare del tempo di rilevamento. Per evidenziare il corretto funzionamento del sistema, sono state eseguite anche delle misurazioni in vicinanza di un materiale che emette radiazioni leggermente superiori rispetto a quelle del fondo naturale, ma comunque innocue per la salute dell'uomo (il materiale in questione è un comune sale dietetico ad alto contenuto di potassio il cui isotopo 40K è radioattivo).


Sale con contenuto di potassio al 28,5%

Si osserva, infatti, nell'ultima riga della tabella che le misure ottenute mettendo a contatto il contatore geiger con il sale sono superiori, anche se di poco, rispetto agli altri valori rilevati.

---

## 4. Design Architecture

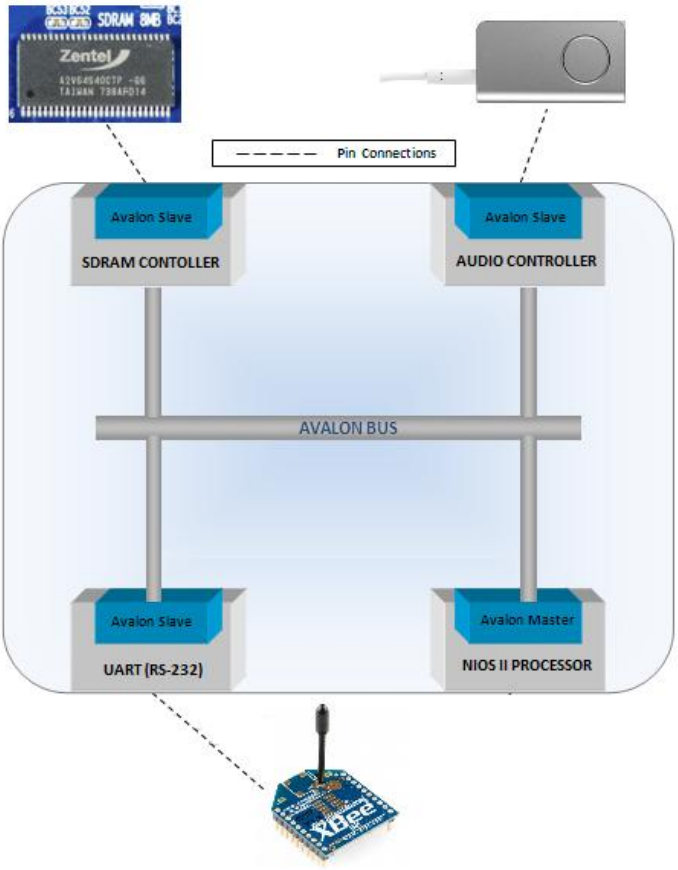
*Please give the system design scheme of your design or both hardware design block diagram and software flow chart.*

Di seguito saranno illustrate le architetture hardware e software del sistema realizzato sia per il nodo-sensore che per la stazione base con l'ausilio degli hardware design block diagrams (schemi bdf) e dei software flow-chart. Inoltre, saranno mostrate le procedure per la configurazione e il corretto interfacciamento e utilizzo dei dispositivi di trasmissione wireless (Xbee) con le schede FPGA.

**Hardware nodo-sensore**

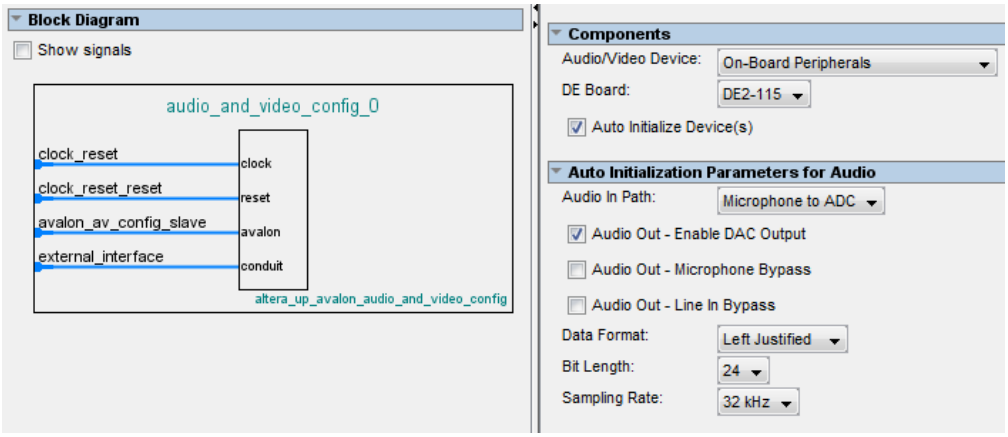
L'immagine sottostante rappresenta una vista di alto livello della configurazione hardware del nodo sensore. Nella figura sono evidenziate le principali periferiche che compongono l'architettura.

Disponendo di due diverse board (DE1 e DE2), è stato deciso di utilizzare per lo sviluppo del nodo sensore la scheda più performante (la board DE2) poichè esso richiede una capacità computazionale maggiore rispetto a quella richiesta dalla stazione base. Il soft processor utilizzato è un Nios-II versione fast (è stata scelta la versione fast al fine di migliorare le prestazioni in real time), caratterizzato da una frequenza di clock di 50 MHz. Abbiamo poi fissato la dimensione delle cache a 4 kB per la cache istruzioni e 2 kB per quella dati; inoltre, date le dimensioni piuttosto limitate della on-chip memory, abbiamo deciso di utilizzare la memoria SDRAM presente sulla board, la quale ha una dimensione di 128 MB. Per accedere alla SDRAM è stato necessario inserire un controller, così come mostrato in figura, scelto tra quelli disponibili nella libreria di componenti di Qsys.



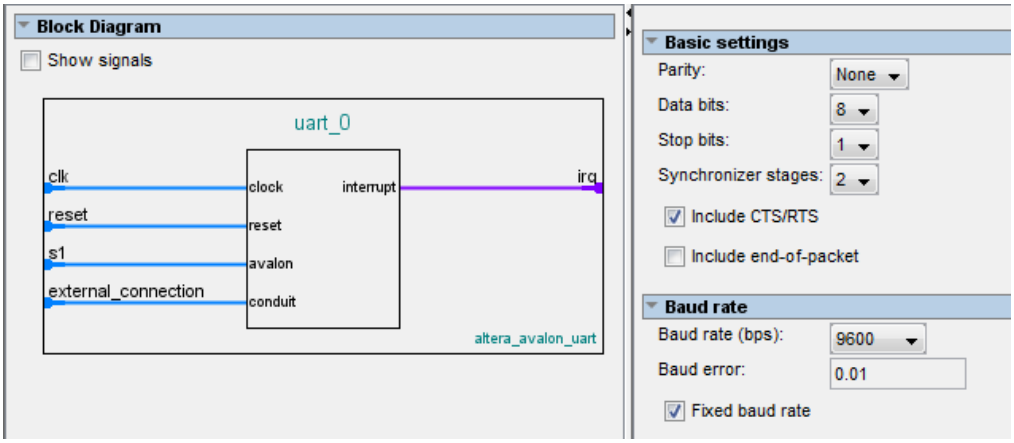

Per utilizzare il sensore per la rilevazione delle radiazioni è necessario utilizzare il jack audio da 3.5 mm (ingresso microfono). Per gestire il segnale in ingresso e permettere la lettura dei campioni acquisiti dal DAC, è stato opportuno inserire 2 ip-core forniti dall'Altera University Program: l' *Audio and video configuration e* l' *Avalon Audio core*.

*Audio and video configuration* - le periferiche audio e video presenti sulla board necessitano di una configurazione iniziale attraverso 2 pin. L'*Audio and video configuration* IP offre una modalità semplice per l'inizializzazione dei device. In particolare, abbiamo scelto il path di input (ovvero il microfono collegato all'ADC), il formato dei dati (nel nostro caso left-justified) e la dimensione del singolo campione (24 bit). Infine, la frequenza di campionamento è stata impostata a 32 kHz, per le motivazioni riportate nella precedente sezione. Il blocco è mostrato nella figura seguente.

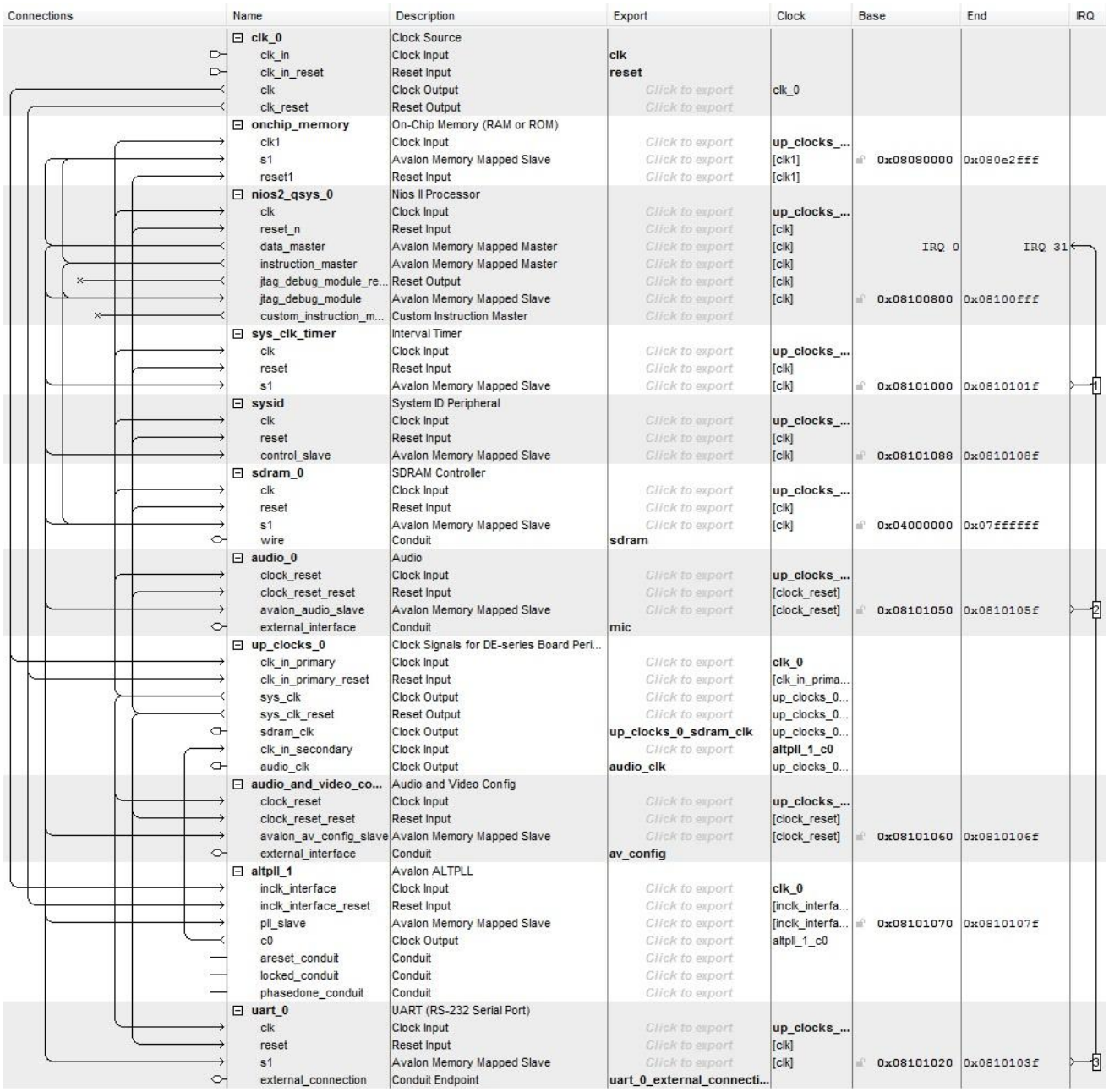



*Avalon Audio core* - questo blocco serve per l'interazione con il codec audio (WM8731) e per la gestione delle code FIFO dei due canali audio (left/right). Esso offre l'interfaccia vera e propria tra il Nios II e il device audio, invocabile dal codice C tramite delle apposite primitive.

I due blocchi sopraelencati vanno a costituire il controller audio mostrato nella prima figura. Il blocco UART è invece necessario per la configurazione della comunicazione tra l'FPGA e il modulo Xbee, che avviene tramite protocollo RS232. Esso fornisce, quindi, il controller della comunicazione seriale dalla parte dell'FPGA e permette di specificare i parametri della comunicazione, tra i quali la presenza di bit di parità, del bit di stop, l'ampiezza del dato trasmesso e il baud-rate. Come vedremo in seguito, i pin necessari alla comunicazione seriale provengono direttamente dall'FPGA tramite l'expansion header, che, rispetto all'utilizzo del connettore standard DB9, è più facile da interfacciare al trasmettitore (esso, infatti, non possiede connettori seriali e sarebbe stato necessario l'utilizzo di hardware esterno per l'adattamento). Nella figura sottostante, è mostrato il dettaglio delle impostazioni dell'IP-core UART.




Vediamo adesso il design hardware completo, mostrato secondo il formalismo grafico di Qsys.

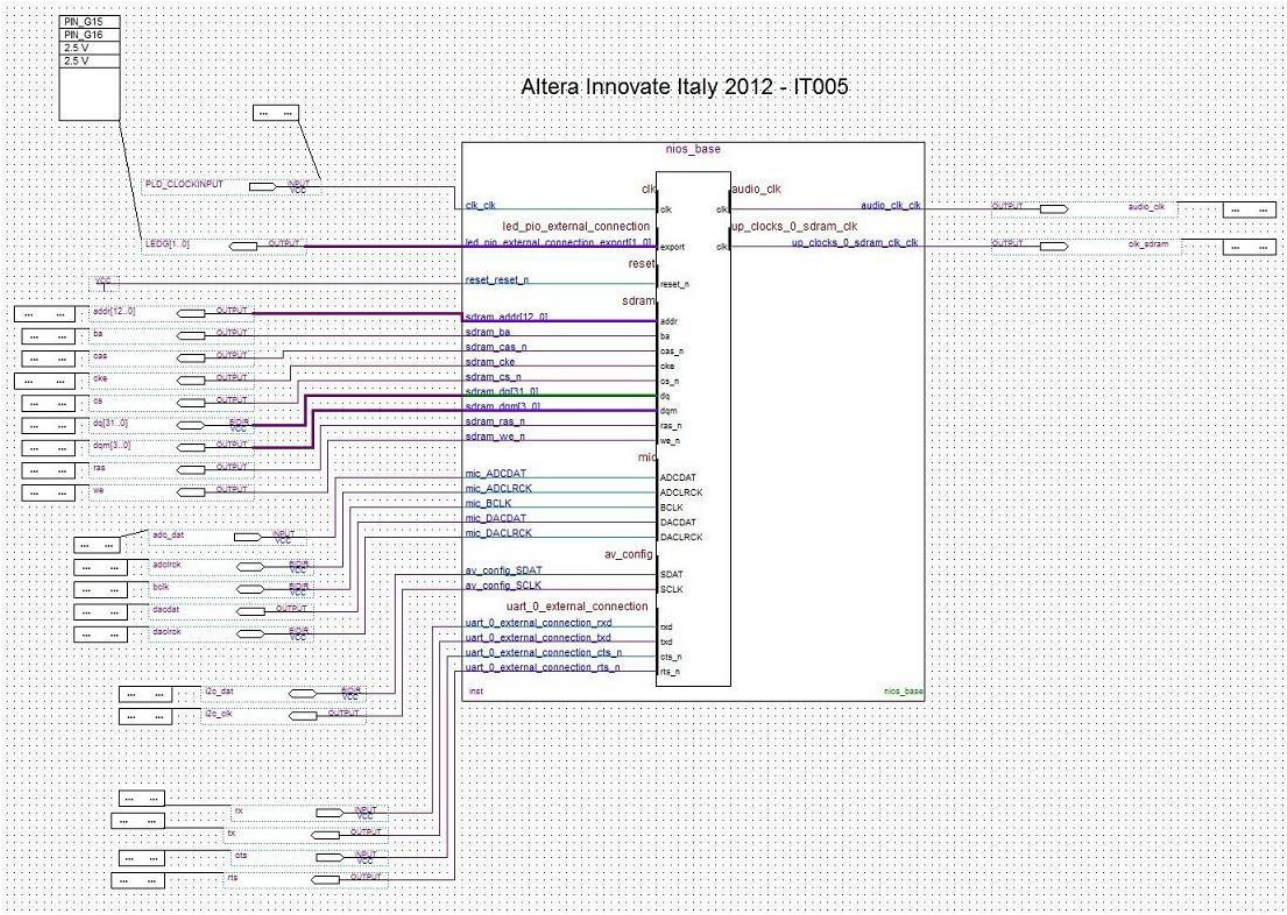


| Connections | Name | Description | Export | Clock | Base | End | IRQ |
|---|---|---|---|---|---|---|---|
| | **clk_0** | Clock Source | | | | | |
| | clk_in | Clock Input | **clk** | | | | |
| | clk_in_reset | Reset Input | **reset** | | | | |
| | clk | Clock Output | Click to export | clk_0 | | | |
| | clk_reset | Reset Output | Click to export | | | | |
| | **onchip_memory** | On-Chip Memory (RAM or ROM) | | | | | |
| | clk1 | Clock Input | Click to export | **up_clocks_...** | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk1] | 0x08080000 | 0x080e2fff | |
| | reset1 | Reset Input | Click to export | [clk1] | | | |
| | **nios2_qsys_0** | Nios II Processor | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset_n | Reset Input | Click to export | [clk] | | | |
| | data_master | Avalon Memory Mapped Master | Click to export | [clk] | | IRQ 0 | IRQ 31 |
| | instruction_master | Avalon Memory Mapped Master | Click to export | [clk] | | | |
| | jtag_debug_module_re... | Reset Output | Click to export | [clk] | | | |
| | jtag_debug_module | Avalon Memory Mapped Slave | Click to export | [clk] | 0x08100800 | 0x08100fff | |
| | custom_instruction_m... | Custom Instruction Master | Click to export | | | | |
| | **sys_clk_timer** | Interval Timer | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk] | 0x08101000 | 0x0810101f | 1 |
| | **sysid** | System ID Peripheral | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | control_slave | Avalon Memory Mapped Slave | Click to export | [clk] | 0x08101088 | 0x0810108f | |
| | **sdram_0** | SDRAM Controller | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk] | 0x04000000 | 0x07ffffff | |
| | wire | Conduit | **sdram** | | | | |
| | **audio_0** | Audio | | | | | |
| | clock_reset | Clock Input | Click to export | **up_clocks_...** | | | |
| | clock_reset_reset | Reset Input | Click to export | [clock_reset] | | | |
| | avalon_audio_slave | Avalon Memory Mapped Slave | Click to export | [clock_reset] | 0x08101050 | 0x0810105f | 2 |
| | external_interface | Conduit | **mic** | | | | |
| | **up_clocks_0** | Clock Signals for DE-series Board Peri... | | | | | |
| | clk_in_primary | Clock Input | Click to export | **clk_0** | | | |
| | clk_in_primary_reset | Reset Input | Click to export | [clk_in_prima... | | | |
| | sys_clk | Clock Output | Click to export | up_clocks_0... | | | |
| | sys_clk_reset | Reset Output | Click to export | up_clocks_0... | | | |
| | sdram_clk | Clock Output | **up_clocks_0_sdram_clk** | up_clocks_0... | | | |
| | clk_in_secondary | Clock Input | Click to export | **altpll_1_c0** | | | |
| | audio_clk | Clock Output | **audio_clk** | up_clocks_0... | | | |
| | **audio_and_video_co...** | Audio and Video Config | | | | | |
| | clock_reset | Clock Input | Click to export | **up_clocks_...** | | | |
| | clock_reset_reset | Reset Input | Click to export | [clock_reset] | | | |
| | avalon_av_config_slave | Avalon Memory Mapped Slave | Click to export | [clock_reset] | 0x08101060 | 0x0810106f | |
| | external_interface | Conduit | **av_config** | | | | |
| | **altpll_1** | Avalon ALTPLL | | | | | |
| | inclk_interface | Clock Input | Click to export | **clk_0** | | | |
| | inclk_interface_reset | Reset Input | Click to export | [inclk_interfa... | | | |
| | pll_slave | Avalon Memory Mapped Slave | Click to export | [inclk_interfa... | 0x08101070 | 0x0810107f | |
| | c0 | Clock Output | Click to export | altpll_1_c0 | | | |
| | areset_conduit | Conduit | Click to export | | | | |
| | locked_conduit | Conduit | Click to export | | | | |
| | phasedone_conduit | Conduit | Click to export | | | | |
| | **uart_0** | UART (RS-232 Serial Port) | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk] | 0x08101020 | 0x0810103f | 3 |
| | external_connection | Conduit Endpoint | **uart_0_external_connecti...** | | | | |

Come si può apprezzare dall'immagine, sono presenti, oltre ai componenti di cui abbiamo già parlato, altri moduli:

- un Interval Timer per lo scheduler del sistema operativo;
- il core System ID peripheral per il check delle versioni;
- un dispositivo PLL che, a partire dal segnale di clock principale, permette di ottenere un nuovo clock a diversa frequenza da applicare in ingresso al blocco up_clocks (Clock Signals). In questo modo, si posssono derivare automaticamente altri segnali di clock necessari per il campionatore audio e per l'SDRAM.

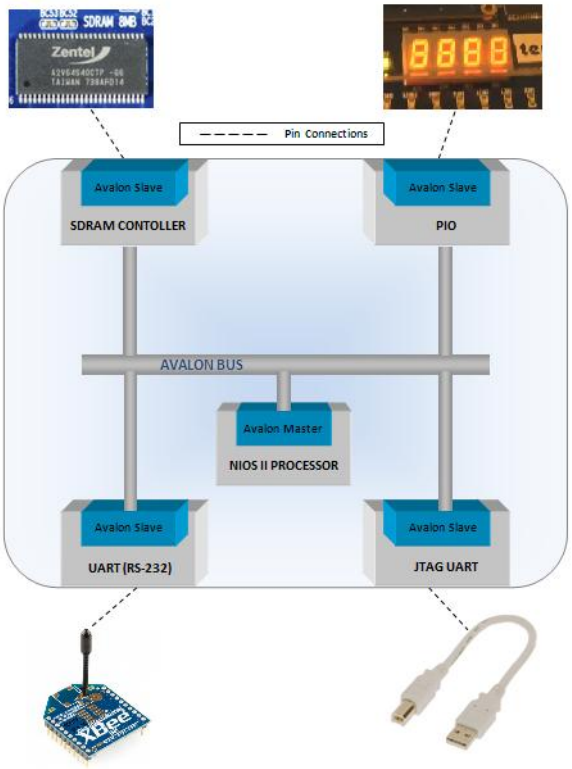
La schermata di sopra mette in evidenza anche gli indirizzi base dei registri tramite i quali è possibile accedere a ciascuna periferica.




Nell'immagine sovrastante è mostrato il Block Diagram File (.bdf) del sistema Nios realizzato, così come presentato dal tool di sviluppo Quartus II. La figura mostra i segnali di input-output del SOPC, i quali andranno associati tramite un'operazione di pin assignment ai pin fisici realmente presenti sulla board.

## Hardware stazione-base

Analogamente a prima mostriamo lo schema di alto livello del nodo che costituisce la stazione base del sistema.




Avendo preferito riservare la scheda DE2 per l'implementazione del nodo-sensore per i motivi già citati in precedenza, per la progettazione della stazione base è stata utilizzata la board DE1.

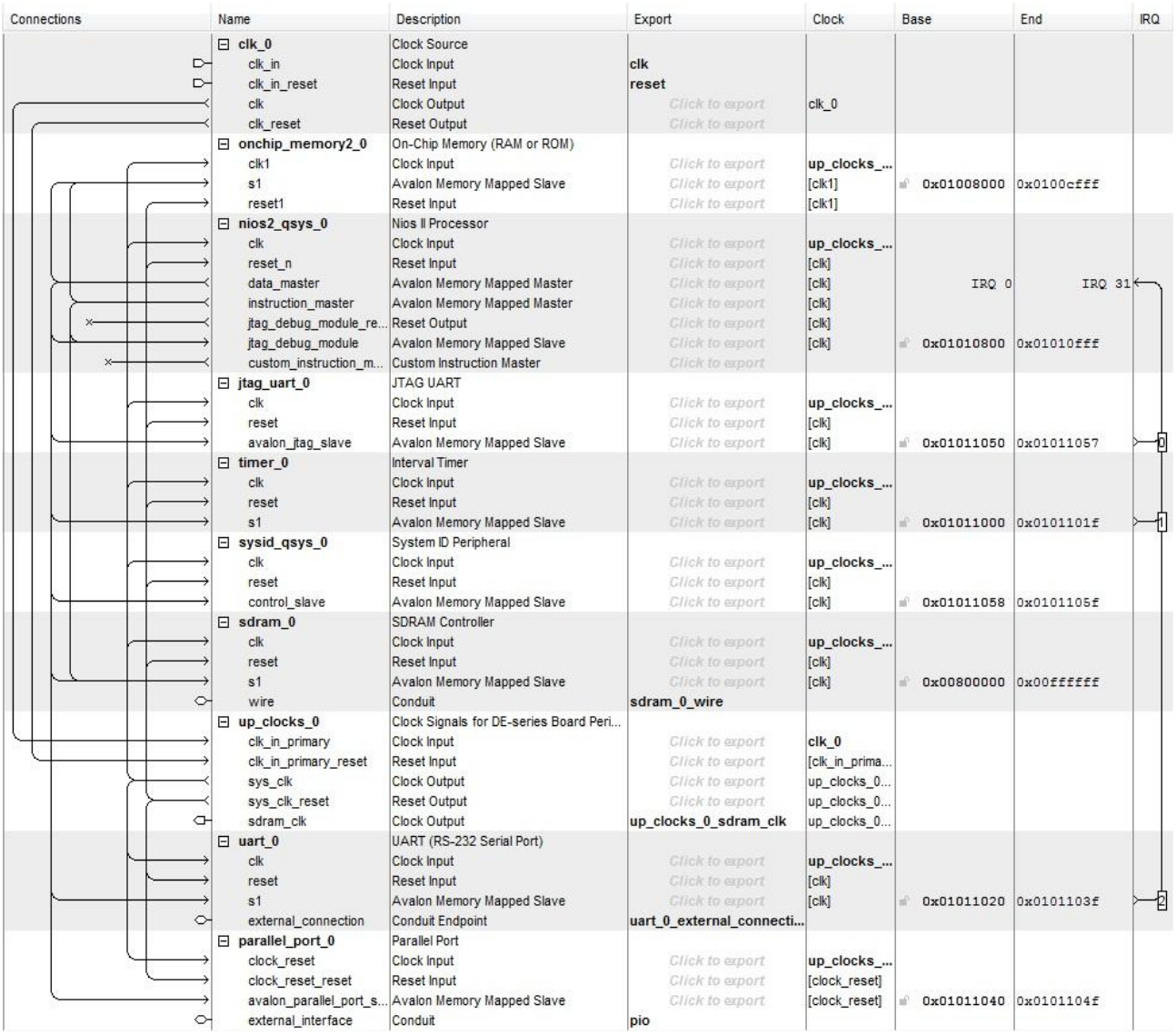
Sulla FPGA della stazione base, è stato installato un Nios-II versione standard, anche in questo caso con frequenza di clock di 50 MHz. Le dimensioni delle cache sono state fissate a 4 kB per la cache istruzioni e 2 kB per quella dati. Anche qui è stato preferito l'uso della SDRAM in aggiunta alla on-chip memory.

Il controller JTAG, necessario per la comunicazione tra Nios e Pc Host, ci ha permesso di utilizzare l'USB Blaster come STDIN ed STDOUT del programma.

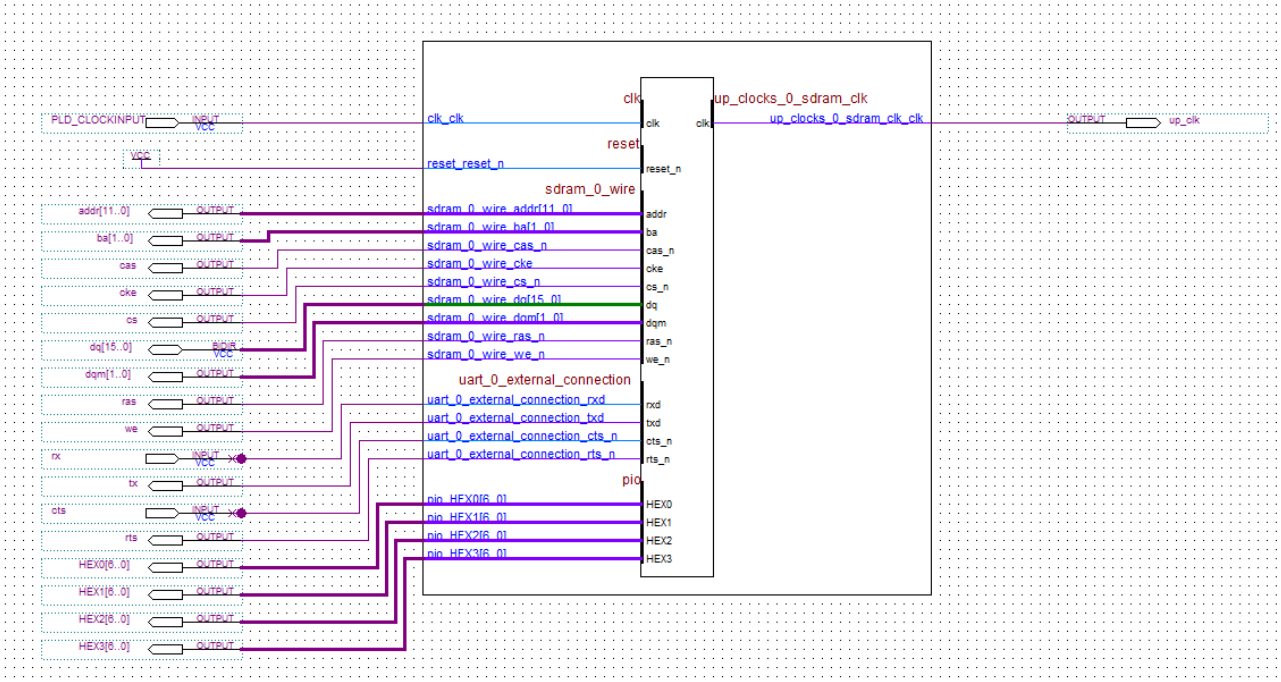
Il controller UART è stato configurato allo stesso modo del nodo-sensore, dovendo controllare anch'esso un modulo Xbee per la ricezione.

La PIO è stata introdotta per poter visualizzare, tramite il display a 7-segmenti della stazione base, il valore della radiazione in uSv/h in tempo reale (durante la misurazione).

Riportiamo anche qui lo schema Qsys del SOPC e il relativo Block Diagram File:



| Connections | Name | Description | Export | Clock | Base | End | IRQ |
|---|---|---|---|---|---|---|---|
| | **clk_0** | Clock Source | | | | | |
| | clk_in | Clock Input | **clk** | | | | |
| | clk_in_reset | Reset Input | **reset** | | | | |
| | clk | Clock Output | Click to export | clk_0 | | | |
| | clk_reset | Reset Output | Click to export | | | | |
| | **onchip_memory2_0** | On-Chip Memory (RAM or ROM) | | | | | |
| | clk1 | Clock Input | Click to export | **up_clocks_...** | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk1] | 0x01008000 | 0x0100cfff | |
| | reset1 | Reset Input | Click to export | [clk1] | | | |
| | **nios2_qsys_0** | Nios II Processor | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset_n | Reset Input | Click to export | [clk] | | | |
| | data_master | Avalon Memory Mapped Master | Click to export | [clk] | IRQ 0 | IRQ 31 | |
| | instruction_master | Avalon Memory Mapped Master | Click to export | [clk] | | | |
| | jtag_debug_module_re... | Reset Output | Click to export | [clk] | | | |
| | jtag_debug_module | Avalon Memory Mapped Slave | Click to export | [clk] | 0x01010800 | 0x01010fff | |
| | custom_instruction_m... | Custom Instruction Master | Click to export | | | | |
| | **jtag_uart_0** | JTAG UART | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Click to export | [clk] | 0x01011050 | 0x01011057 | 0 |
| | **timer_0** | Interval Timer | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk] | 0x01011000 | 0x0101101f | 1 |
| | **sysid_qsys_0** | System ID Peripheral | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | control_slave | Avalon Memory Mapped Slave | Click to export | [clk] | 0x01011058 | 0x0101105f | |
| | **sdram_0** | SDRAM Controller | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk] | 0x00800000 | 0x00ffffff | |
| | wire | Conduit | **sdram_0_wire** | | | | |
| | **up_clocks_0** | Clock Signals for DE-series Board Peri... | | | | | |
| | clk_in_primary | Clock Input | Click to export | **clk_0** | | | |
| | clk_in_primary_reset | Reset Input | Click to export | [clk_in_prima... | | | |
| | sys_clk | Clock Output | Click to export | up_clocks_0... | | | |
| | sys_clk_reset | Reset Output | Click to export | up_clocks_0... | | | |
| | sdram_clk | Clock Output | **up_clocks_0_sdram_clk** | up_clocks_0... | | | |
| | **uart_0** | UART (RS-232 Serial Port) | | | | | |
| | clk | Clock Input | Click to export | **up_clocks_...** | | | |
| | reset | Reset Input | Click to export | [clk] | | | |
| | s1 | Avalon Memory Mapped Slave | Click to export | [clk] | 0x01011020 | 0x0101103f | 2 |
| | external_connection | Conduit Endpoint | **uart_0_external_connecti...** | | | | |
| | **parallel_port_0** | Parallel Port | | | | | |
| | clock_reset | Clock Input | Click to export | **up_clocks_...** | | | |
| | clock_reset_reset | Reset Input | Click to export | [clock_reset] | | | |
| | avalon_parallel_port_s... | Avalon Memory Mapped Slave | Click to export | [clock_reset] | 0x01011040 | 0x0101104f | |
| | external_interface | Conduit | **pio** | | | | |

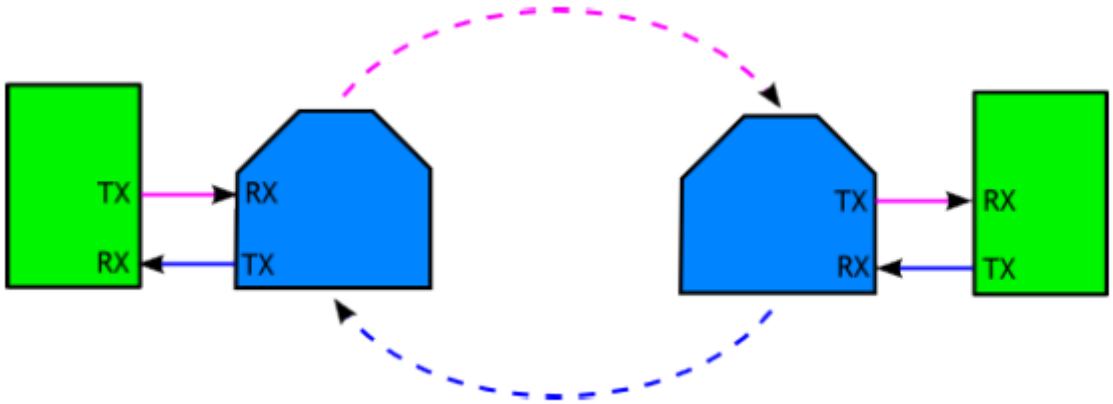



## Configurazione dei moduli Xbee

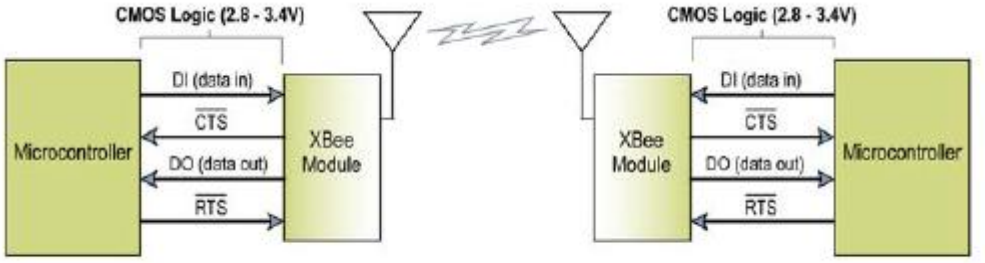
Descriviamo ora i settaggi che permettono ai moduli Xbee di funzionare instaurando una comunicazione autenticata, crittografata e priva di interferenze.

I moduli wireless Xbee possono essere immaginati come il prolungamento di un cavo seriale. La comunicazione tra di essi avviene tramite il protocollo ZigBee ed è gestita autonomamente dai dispositivi, mentre la comunicazione con le FPGA avviene tramite protocollo RS232.

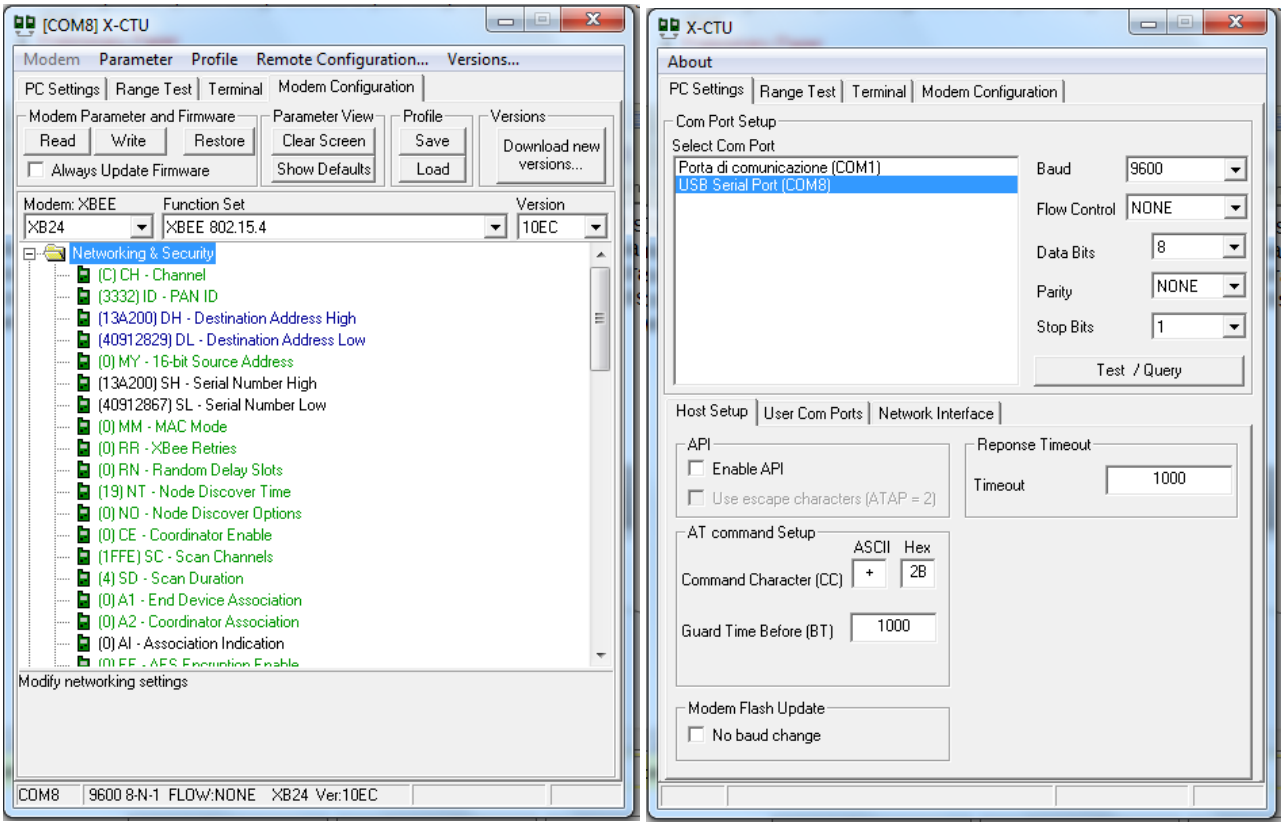
Lo schema generale di funzionamento è mostrato nella seguente figura, in cui i blocchi di colore verde rappresentano gli expansion header delle due schede e i blocchi di colore azzurro rappresentano i dispositivi wireless.



In realtà, per la comunicazione con le FPGA non è stato utilizzato il set minimale di segnali previsto dallo standard RS232 (RX, TX), ma sono state aggiunte altre due linee dedicate all' hardware flow-control (CTS, RTS), che permettono di gestire i problemi di interferenza tramite un handshake, in modo da evitare la trasmissione/ricezione di caratteri casuali dovuti al rumore esterno. Di seguito, è mostrato lo schema precedente in maggiore dettaglio:




Per quanto riguarda invece l'autenticazione, ogni dispositivo Xbee dispone di uno Unique Serial Number (USN), cablato in hardware, che permette di identificarlo univocamente. Ciò consente di creare delle reti permettendo una comunicazione punto-punto, broadcast o multicast, semplicemente specificando nei registri di controllo di ogni dispositivo i campi *Destination Address* e *Source Address,* il cui settaggio può essere effettuato facilmente tramite un apposito programma fornito dal costruttore (X-CTU) , come mostrato nelle seguenti schermate:



La prima schermata contiene tutti i settaggi del dispositivo, tra cui la possibilità di effettuare una comunicazione sicura mediante l'utilizzo dell'algoritmo di cifratura AES, permettendo di specificare anche la relativa chiave. La seconda schermata permette di specificare impostazioni generali del protocollo RS232, tra cui baud rate, parity e stop bit e data width, i quali devono essere coerenti con quanto specificato in Qsys nel controller UART delle FPGA.

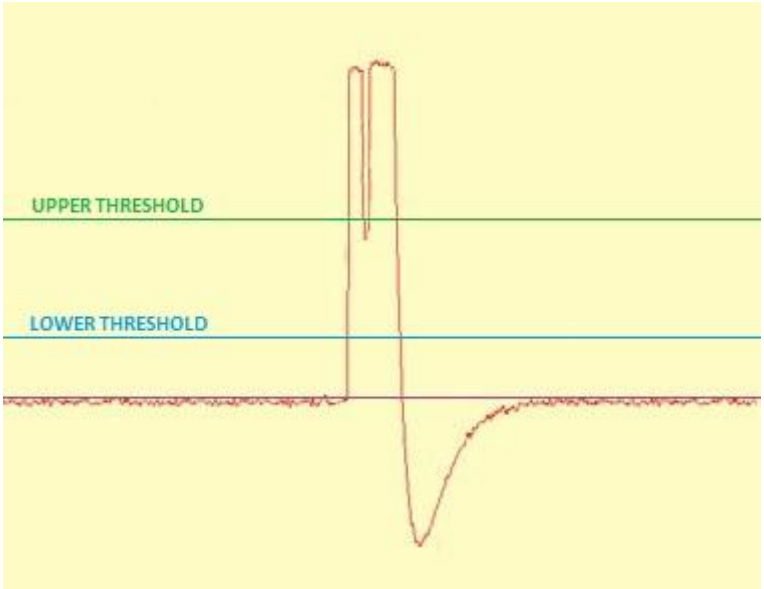
**Software nodo sensore**

Prima di mostrare i flow-chart degli algoritmi per il Nios, illustriamo di seguito la struttura dell'algoritmo a doppia soglia utilizzato per rilevare gli impulsi provenienti dal contatore Geiger.

Non sono conosciuti a priori i dettagli sulla forma d'onda di tali impulsi, poichè essi dipendono anche dall'implementazione hardware del sensore; quindi, non possiamo fare alcuna assunzione sull'energia del segnale. Per questo motivo l'impulso viene rilevato solo quando l'ampiezza del segnale in ingresso oltrepassa una data soglia.

Uno dei problemi principali che possono verificarsi in questo caso è che, dopo la fase di salita, l'impulso oscilli ripetutamente a causa di rumore o perturbazioni esterne prima di iniziare la fase di discesa. Se tali oscillazioni portano ad un superamento ripetuto della soglia, il conteggio degli impulsi risulta errato, in quanto ne verrebbero rilevati più di quelli effettivi. Per eliminare questi impulsi fittizi si è ricorso ad un'ulteriore soglia, fissata ad un livello più basso della precedente, che viene utilizzata per determinare la fase di discesa.

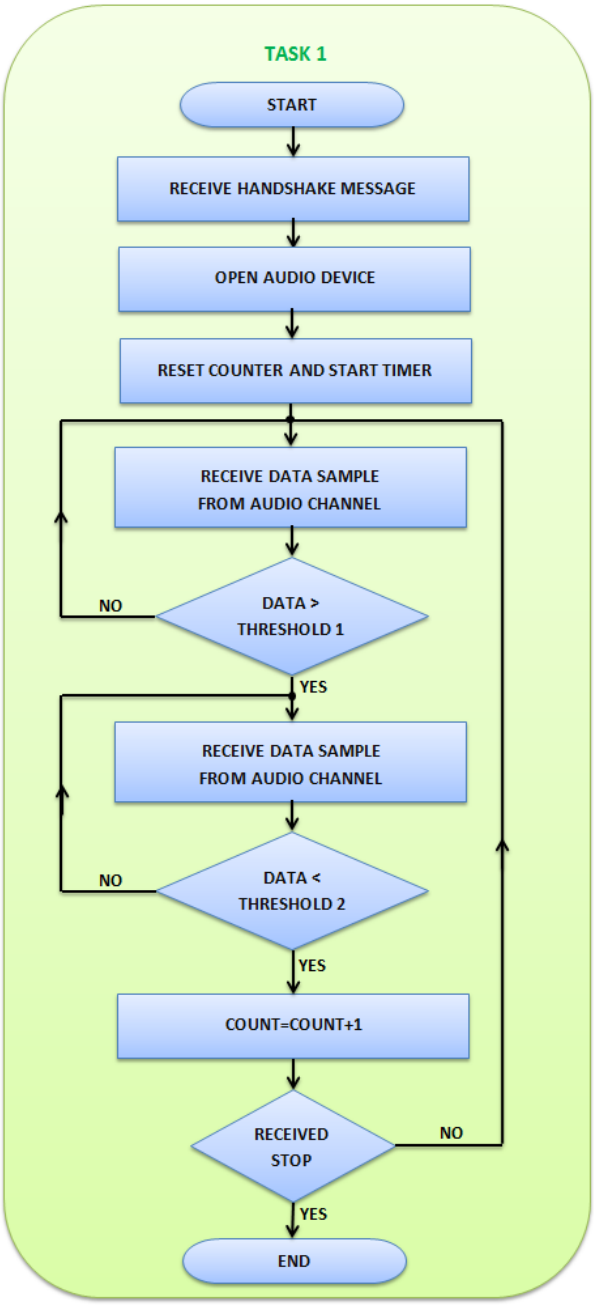
Di seguito una figura che rende più chiara la situazione:

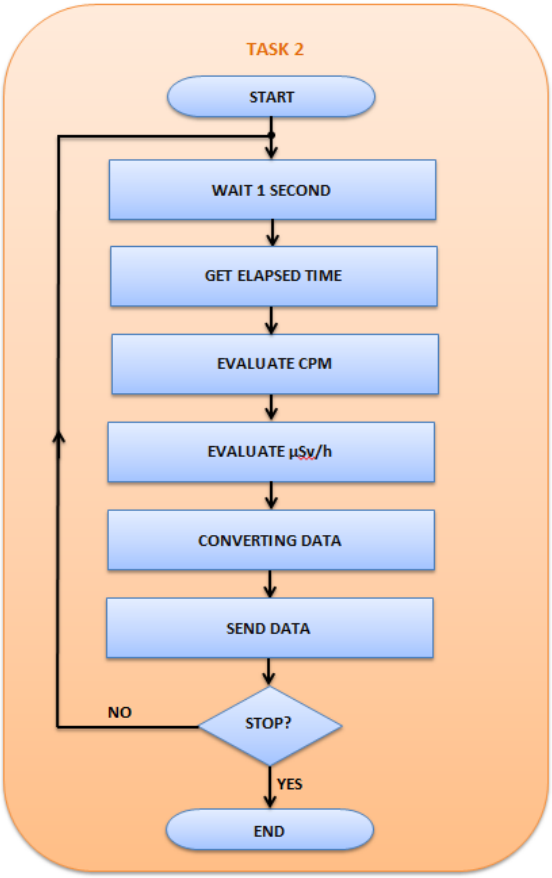


Il valore per l'upper threshold è stato fissato al 50% del valore massimo rappresentabile dal singolo campione, come suggeritoci anche da alcuni componenti del team di sviluppo del contatore Geiger. La lower threshold, da noi aggiunta per garantire maggiore robustezza all'algoritmo, è stata invece fissata intorno al 10%. Infatti, nell'esempio mostrato in figura, in assenza della lower threshold l'algoritmo avrebbe rilevato, erroneamente, due impulsi invece che uno.

A questo punto analizziamo i flow-chart relativi al software implementato per il Nios del nodo sensore. Come già detto nelle precedenti sezioni, è stato sfruttato il sistema operativo real-time uC-OS II in quanto dotato di meccanismi di gestione veloce del multithreading.

Nell'applicazione specifica sono stati realizzati due task: il primo dedicato alle operazioni di detection degli impulsi e al conteggio degli stessi, ottenuti interfacciandosi con il contatore Geiger; il secondo, invece, è rivolto alla trasformazione dei dati grezzi acquisiti dal primo task e all'interfacciamento con il dispositivo Xbee che li trasmette via wireless verso la stazione base.




Il ***task 1*** innanzitutto attende la ricezione di un segnale di handshake da parte della stazione base (il cui algoritmo verrà mostrato successivamente), che dà il via alle operazioni. Appena ricevuta l'abilitazione, viene inizializzato il controller audio collegato al contatore geiger e vengono resettate eventuali misure precedenti. Inoltre, viene avviato un timer che, come vedremo, sarà necessario al task 2. A questo punto, si passa all'esecuzione vera e propria, che consiste nell'algoritmo double-threshold per la detection degli impulsi illustrato in precedenza. Ad ogni impulso identificato (che corrisponde a rilevare il passaggio del segnale attraverso le due soglie in successione), viene incrementata una variabile globale di conteggio (count). Il task procede iterativamente fin quando non riceve un

segnale di stop dalla stazione base (i segnali di handshake consistono nella ricezione via wireless di caratteri speciali).

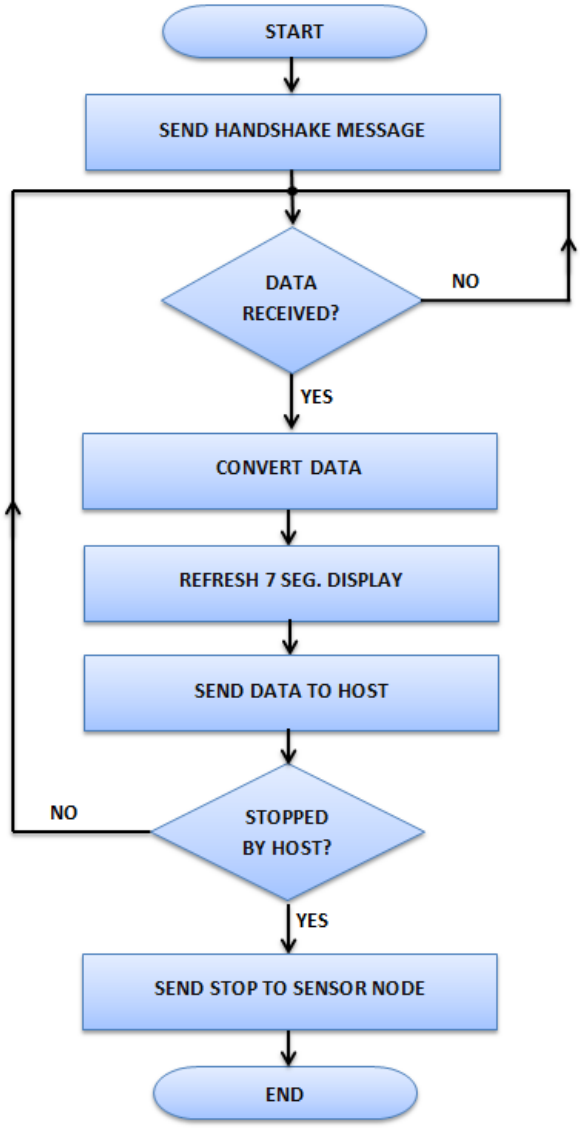
Il ***task 2*** parte in contemporanea con il task 1 e procede parallelamente risvegliandosi ad intervalli temporali di un secondo. Il compito di questo task è quello di utilizzare la variabile globale 'count' aggiornata dal primo task per calcolare il valore della misura in uSv/h, tenendo conto dell'equazione lineare che li lega. Per fare questo calcolo è necessario conoscere il tempo trascorso dall'inizio dell'elaborazione (questo spiega la presenza del timer inizializzato dal task 1), tenendo conto naturalmente del tempo di sistema e non del tempo logico del singolo task. Una volta ottenuto il valore, è necessario inviarlo utilizzando l'Xbee. A tal fine bisogna tener conto del fatto che l'RS232 permette di inviare un byte alla volta, che possiamo immaginare come un carattere ASCII. E' quindi necessaria una conversione dei dati prima dell'invio per adattarli a tali requisiti. Infine il dato viene scritto nel buffer dell'UART e trasmesso. Il task termina anch'esso alla ricezione del carattere speciale di stop.

**Software stazione base**

Passiamo ora ad illustrare i flow-chart del software della stazione base. Ricordiamo che essa è composta da un laptop al quale è collegata una scheda FPGA dotata di dispositivo Xbee. Nel seguito ci riferiremo al laptop con il termine *host*.

Analizziamo, in primo luogo, il flow chart dell'applicazione eseguita sul soft-processor dell'FPGA.

All'avvio del processo, la stazione base provvede all'invio al nodo sensore del segnale di start, dopodichè resta in attesa di ricevere un dato. Quando questo viene ricevuto (una volta al secondo), si effettuano le operazioni di riconversione per ricostruire il valore originario della misura, e si aggiorna la visualizzazione sul display a 7 segmenti.



Inoltre, tale valore viene anche ottenuto dall'host (via usb-blaster) per la registrazione in un file di log, realizzata effettuando il redirect dello stdout dal terminale al file stesso.

Quando l'utente vuole, può digitare un carattere speciale sul terminale dell'host per ordinare la terminazione delle operazioni. Ciò comporta l'invio del segnale di stop al nodo sensore.

Per rendere ulteriormente autonomo il sistema, le operazioni di inizializzazione (trasferimento del bitstream sull'FPGA, invio dell'eseguibile al soft-processor Nios, ecc.) che, normalmente vengono eseguite utilizzando IDE grafici, come ad esempio Eclipse e Quartus, sono state delegate ad un'unica applicazione software da noi sviluppata in Java. Tale applicazione, oltre alle suddette operazioni, ha anche il compito di interfacciarsi con l'utente che gestisce la stazione base. In seguito, abbiamo deciso di migliorare l'intero sistema integrando i dati raccolti dall'FPGA con un meccanismo di geolocalizzazione, basato o sulla conoscenza dell'indirizzo in cui si trova il sensore, o attraverso la localizzazione della connessione all'ISP.

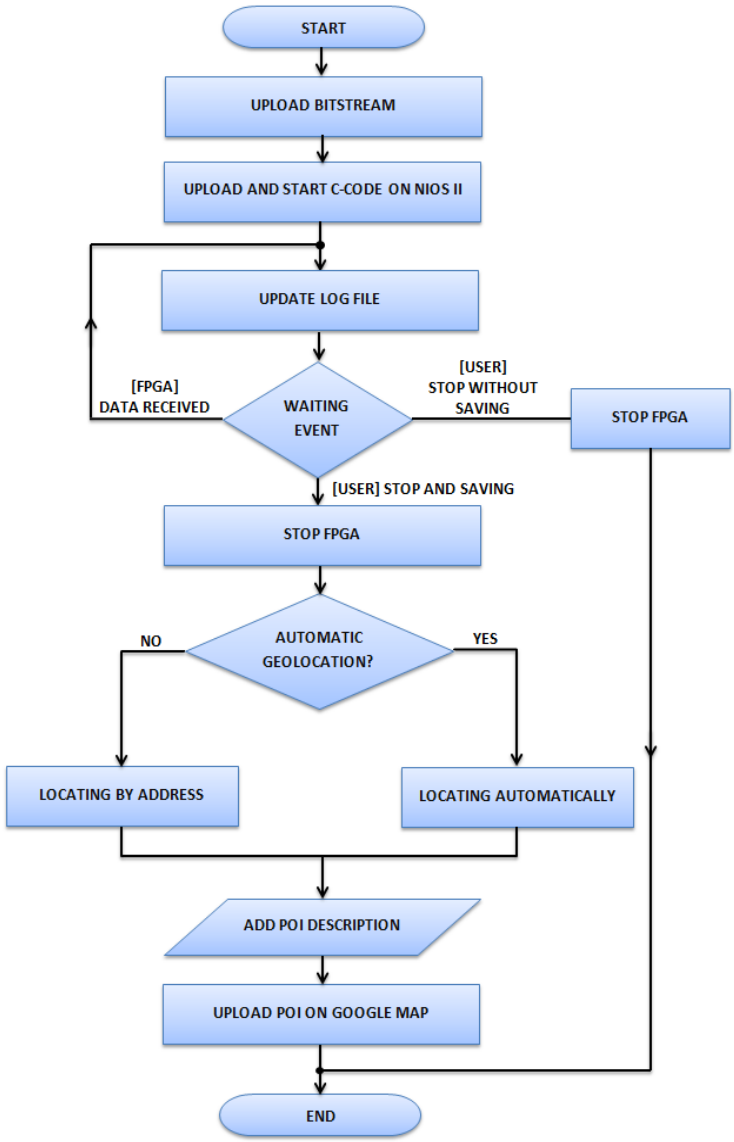
Inoltre, ogni volta che si effettua una misurazione in un determinato luogo, è possibile riportare automaticamente i risultati della stessa sotto forma di Point of Interest (POI) su di una Google map accessibile in remoto da qualsiasi dispositivo (tablet, smartphone, laptop) che desideri visualizzare le informazioni.

In questo modo è possibile mantenere un database aperto garantendo a chiunque disponga di un contatore Geiger di contribuire al progetto fornendo le misure rilevate per la pubblicazione on-line. Questo può essere un utile strumento per il monitoraggio dei livelli di radioattività in aree contaminate (discariche, container, centrali nucleari).

E' anche possibile, per evitare modifiche improprie dei dati, limitare la modifica della mappa solo ad utenti autorizzati, mantenendo però la visualizzazione accessibile a tutti. Nel sistema da noi realizzato, abbiamo deciso di utilizzare una mappa completamente libera.

Illustriamo adesso il flow-chart dell'applicazione host.

La prima operazione è il caricamento del bitstream (.sof) che inizializza l'FPGA.



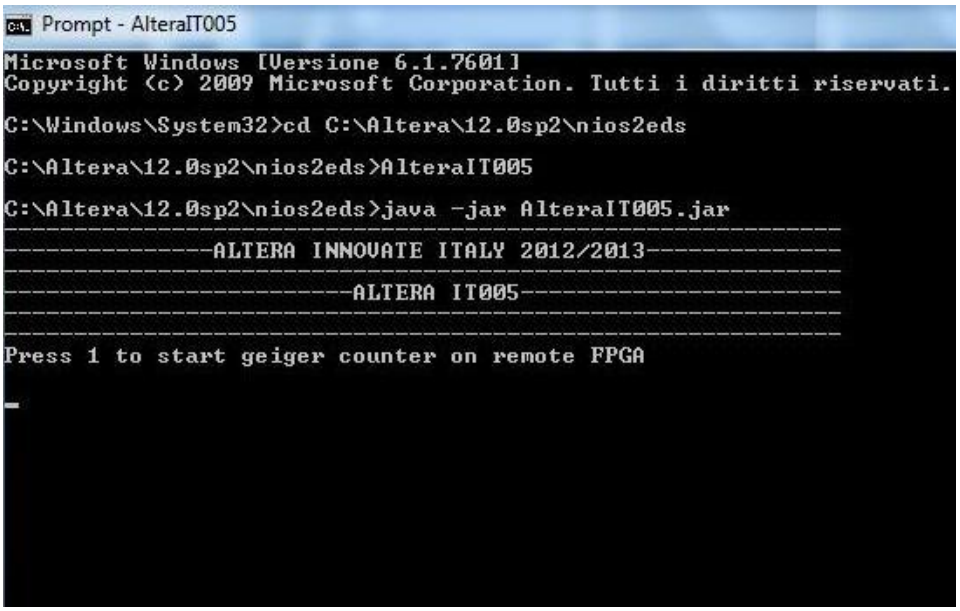

Successivamente viene inviato il codice C da eseguire sul Nios II (.elf relativo al flow-chart precedente). Quindi, l'applicazione si mette in attesa di ricevere periodicamente i dati dal Nios, aggiornando così il file di log.

All'utente viene mostrata un'interfaccia tramite la quale può decidere in qualsiasi momento di terminare le operazioni di misurazione. In questo caso è possibile scegliere due alternative:

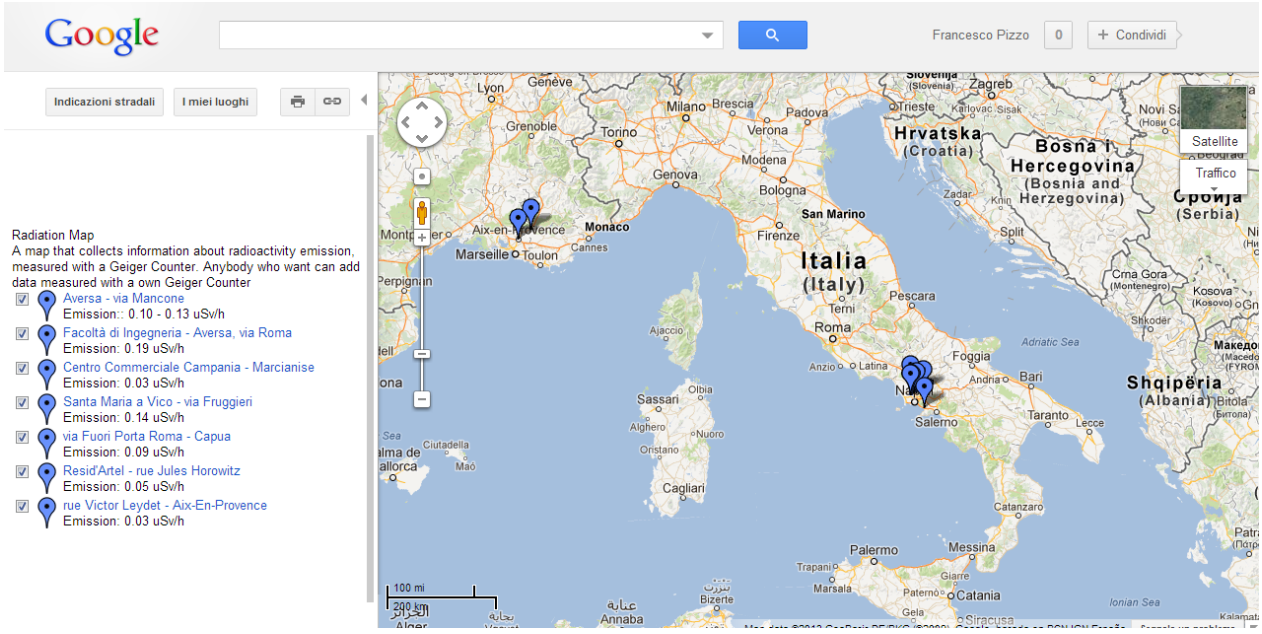
- terminare il processo e uscire dall'applicazione;
- aggiornare la mappa con i dati raccolti effettuando una geolocalizzazione automatica o tramite indirizzo (se noto).

Scegliendo la seconda alternativa, l'applicazione recupera le informazioni relative a latitudine e longitudine del POI da salvare, e richiede all'utente di inserire una breve descrizione. Infine, il POI viene aggiunto alla mappa utilizzando le API di Google e il processo termina, aprendo a video una finestra del browser all'indirizzo della mappa condivisa (http://goo.gl/W5I45).

Di seguito una figura che mostra uno screenshot dell'applicazione:




La figura che segue, invece è relativa alla Google Map utilizzata nell'applicazione realizzata:



# 5. Design Methodology

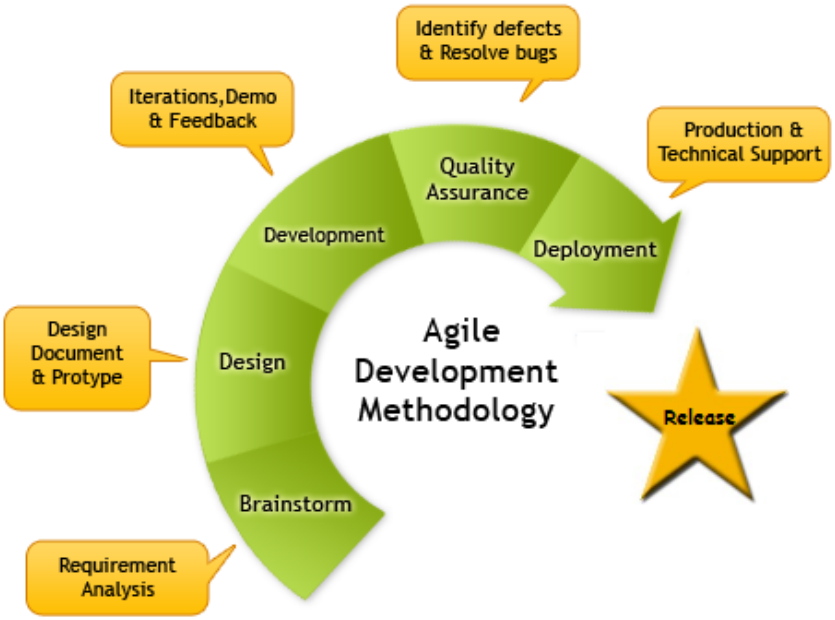
*Please give the detailed description of the implementation method and steps of the design, especially how the design is implemented using the concept of SOPC.*

Il sistema è stato realizzato adottando un approccio evolutivo con il quale di volta in volta sono state implementate diverse versioni, introducendo nuove caratteristiche e devices rispetto alle precedenti.

In particolare, per la realizzazione dell'intero sistema, abbiamo utilizzato un metodo di sviluppo ispirato alle "metodologie agili", dalle quali sono state ereditate alcune importanti caratteristiche come:

- la suddivisione del lavoro in sottoproblemi (secondo lo spirito *divide et impera);*
- l'interscambiabilità dei ruoli tra i diversi membri del team;
- la programmazione a coppie (tipica dell' *extreme programming*);
- l'approccio di tipo incrementale;
- daily briefings necessari per discutere degli obiettivi da raggiungere e di eventuali problemi da risolvere.

A partire dall'abstract iniziale proposto, abbiamo cercato di individuare, sulla base delle funzionalità richieste, i principali moduli che compongono il sistema come sottoproblemi di complessità comparabile tra loro. A questo punto, tra essi sono stati individuati i moduli necessari alla realizzazione di una release minimale (funzionalità base del nodo sensore) ed eseguibile. Definita quest'ultima, il sistema è stato sviluppato secondo un tradizionale modello a cascata che prevede le fasi di progettazione, implementazione, testing e controllo dei risultati. Nel momento in cui la release realizzata si mostrava correttamente funzionante, restituendo risultati coerenti con quelli attesi, abbiamo proceduto secondo l'approccio incrementale con l'aggiunta di nuove funzionalità alla stessa (come, ad esempio, la comunicazione wireless) per ottenere una nuova release. Questo procedimento è stato poi ripetuto iterativamente ottenendo cosi varie versioni del sistema fino alla release finale.




Nello specifico, l'approccio iniziale è stato quello di prendere confidenza con le funzionalità avanzate della board e con i vari componenti hardware (sensore Pocket-Geiger, coppia di XBee). A tale scopo sono state effettuate diverse semplici prove ed analisi dei segnali restituiti dal sensore, reperendo informazioni utili da datasheet e documentazioni varie.

Acquisite le conoscenze necessarie ai nostri scopi, abbiamo provveduto alla realizzazione di un sistema minimale, formato dalla scheda Altera DE2 e dal sensore, sviluppando un semplice programma C eseguibile su un sistema costituito dal Nios e dalle opportune periferiche (come l'IP Core del device audio), che effettua l'acquisizione dei dati restituiti dall'ADC e il digital signal processing degli stessi necessario a fornire la misura della dose equivalente di radiazione, mostrando infine i risultati ottenuti sul terminale (*Versione 1*).

Successivamente, il sistema è stato ampliato con l'aggiunta del modulo wireless collegato all'fpga tramite l'expansion header. In questa release (*Versione 2*) la comunicazione avveniva tra il nodo-sensore ed un laptop, senza utilizzare ancora la seconda FPGA e senza l'utilizzo dell'RTOS. Una volta messa a punto la comunicazione wireless, impostando gli Xbee in modo da essere mutuamente autenticati e specificando l'utilizzo della crittografia simmetrica AES, ci si è resi conto che l'invio dei dati via wireless dopo ogni campione processato, risultava poco performante e non adatto a soddisfare il requisito di esecuzione in real-time.

A tal fine, si è reso necessario l'utilizzo di un sistema multi-tasking, e quindi dell'RTOS uC-OS II, per separare le operazioni di Digital Signal Processing dalle

operazioni di comunicazione senza fili, realizzando così un'ulteriore versione del nostro sistema (*Versione 3*).

A questo punto, terminato lo sviluppo delle funzionalità del nodo-sensore, si è passati alla realizzazione della stazione base, ponendoci nel contesto delle reti di sensori. Abbiamo collegato la seconda fpga ad un ulteriore dispositivo Xbee e abbiamo sviluppato il codice per l'handshake e per la ricezione dei dati dal nodo con il contatore geiger (*Versione 4*).

Infine, dopo esserci accertati del corretto funzionamento dell'intero sistema, abbiamo deciso di realizzare un applicazione finale (*Versione 5*) che, a partire da un unico file eseguibile, effettua tutte le operazioni necessarie all'inizializzazione della board della stazione base ed alcune operazioni aggiuntive:

- download del bitstream sull'FPGA (.sof);
- invio e lancio dell'eseguibile sul NIOS II (.elf);
- lancio dell'interfaccia per l'utente;
- generazione file di log;
- operazioni di geo-location;
- upload del POI sulla Google Map del progetto.

Questa applicazione va quindi a formare, insieme all'intera configurazione hardware della rete di sensori, il nostro progetto finale.

Di seguito mostriamo una tabella riepilogativa delle funzionalità aggiunte in ogni release:

| | Analisi DSP | Comunicazione Wireless | RTOS (Multitask) | Rete Sensori (Più nodi) | Geolocation (POI su GMap) | Esecuzione Automatizzata |
|---|---|---|---|---|---|---|
| Versione 1 | ✓ | | | | | |
| Versione 2 | ✓ | ✓ | | | | |
| Versione 3 | ✓ | ✓ | ✓ | | | |
| Versione 4 | ✓ | ✓ | ✓ | ✓ | | |
| Versione 5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

N.B. Per ogni versione, in tutte le schede fpga presenti è stato utilizzato il soft processor Nios II.

## 6. Design Features

*Please enumerate the outstanding features of your design and what aspects of Nios helped you to implement them.*

Nella realizzazione del progetto sono state implementate diverse caratteristiche volte a garantire requisiti tipici dei sistemi informatici.

Uno degli obiettivi principali che abbiamo cercato di perseguire è stato quello di rendere il sistema completamente **autonomo** nella sua versione definitiva.
La scheda FPGA si presta particolarmente a questo compito, grazie anche alla semplicità con la quale è possibile interfacciarla a stazioni di calcolo e a differenti moduli di comunicazione. In particolare il nodo sensore, in grado di acquisire e processare dati in real-time provenienti dal contatore geiger, si basa sulla sola FPGA senza la necessità di potenza di calcolo esterna. Inoltre vi è una stazione base capace di comunicare con il nodo-sensore per la ricezione dei dati elaborati, con la possibilità di pubblicare i dati raccolti dal sensore su una mappa accessibile e consultabile liberamente tramite internet.
E' proprio grazie all'FPGA che si realizza l'autonomia del sistema in quanto, in uno scenario esteso con molti rilevatori, l'unico terminale che necessiterebbe di un PC sarebbe la stazione base, mentre i nodi con i sensori risultano essere completamente **stand-alone**.

In secondo luogo, il sistema è stato progettato nell'ottica di poterlo estendere con ulteriori moduli. L'integrazione di nuovi componenti fornisce, infatti, la possibilità di aggiungere nuove funzionalità e di migliorare, in termini prestazionali o qualitativi, quelle già esistenti. Da questo punto di vista, le schede FPGA permettono facilmente di inserire nel sistema altri dispositivi (ad esempio, nuovi tipi di sensori) che possono essere utilizzati anche per la raccolta e l'elaborazione di dati eterogenei. Il conseguimento del requisito di **estensibilità** è facilitato dalla presenza di tool dedicati offerti da Altera (come Qsys) con i quali è possibile sviluppare o modificare in maniera intuitiva SOPC complessi utilizzando i controller per la gestione delle periferiche presenti nelle librerie IP Core Altera.

Un altro requisito fondamentale che è soddisfatto dalla nostra applicazione è quello di elaborazione e processamento dei dati in **real-time**. A tal proposito, la possibilità offerta dal NIOS II Processor di utilizzare come linguaggio di programmazione il C ha facilitato lo sviluppo di un opportuno algoritmo di DSP specifico per il nostro

sistema. Inoltre, il tool Qsys permette di personalizzare il processore NIOS specificandone i parametri architetturali in base alle caratteristiche del sistema da sviluppare. Tale caratteristica è stata sfruttata utilizzando due diverse versioni del NIOS sulle due FPGA del sistema (NIOS Standard e NIOS Fast rispettivamente per la stazione base e per il nodo-sensore).

L'esigenza di processare i dati raccolti dal sensore e, simultaneamente, di inviare gli stessi alla stazione base, determina invece la necessità di utilizzare task differenti per la ripartizione di questi compiti. Questo aspetto è garantito dall'uso dell'RTOS che offre una vasta libreria per il supporto alla programmazione multi-threading.

Il sistema si basa, infine, su una **comunicazione affidabile** e sicura tra le due FPGA coinvolte che è garantita dall'utilizzo dei moduli transceiver XBee attraverso un algoritmo di crittografia AES, così come illustrato nella descrizione architetturale del sistema (sezione 4).

---

## 7. Conclusion

*What you learned during the design? During this contest, you certainly increased your understanding of Altera FPGA devices and made some conclusions. These conclusions will be useful for others who are learning about Altera FPGA devices or using Altera FPGA devices as reference. Please tell us what you learned during the design.*

La partecipazione a questo contest è stata un'esperienza molto formativa che ci ha permesso di accrescere le nostre conoscenze sui dispositivi FPGA e perfezionare l'utilizzo dell'hardware Altera.

La realizzazione di questo lavoro ci ha consentito inoltre di mettere in pratica le conoscenze acquisite durante il nostro percorso di studi universitari, integrandole con le nuove competenze assunte durante lo sviluppo del progetto stesso.

Un aspetto molto interessante, che è emerso dal lavoro svolto, è legato alla possibilità di estendere facilmente con hardware aggiuntivo il design sviluppato sulla scheda FPGA, incrementando in questo modo a dismisura i possibili scenari di utilizzo.

La caratteristica di riprogrammabilità dell'hardware per via software rende l'FPGA un dispositivo particolarmente adatto a tutte quelle situazioni nelle quali è necessario realizzare un prototipo di un determinato sistema. Inoltre, nei casi in cui sia necessario aggiornare l'hardware per miglioramenti o correzioni di eventuali errori, basterà modificare o riscrivere il codice VHDL riprogrammando il chip senza dover sostituire fisicamente il dispositivo, garantendo rapidità nelle fasi di testing e manutenzione.

L' *Altera Innovate Italy Contest* ha anche contribuito a maturare le nostre capacità di organizzazione e collaborazione all'interno di un team. Infatti, la necessità di dover raggiungere un obiettivo prefissato in un tempo limitato ci ha spinto ad organizzare in maniera accurata e puntuale il lavoro, attraverso una suddivisione equa dei compiti tra i componenti del team e nell'arco temporale a disposizione.

Il confronto con queste problematiche ci ha permesso di acquisire abilità che potranno rivelarsi utili nel momento in cui bisognerà approcciarsi al mondo del lavoro.