SISTEMI INTEGRATI PER IL MONITORAGGIO, L'EARLY WARNING
E LA MITIGAZIONE DEL RISCHIO IDROGEOLOGICO
LUNGO LE GRANDI VIE DI COMUNICAZIONE

UNIONE EUROPEA
Fondo europeo di sviluppo regionale

PON Ricerca e Competitività 2007-2013

Ministero dell'Istruzione, dell'Università e della Ricerca

Ministero dello Sviluppo Economico

investiamo nel vostro futuro

PROGETTO PON01_01503

Quaderno

04 PON LEWIS

CIRCUITI INTEGRATI A BASSA POTENZA PER SISTEMI DI MONITORAGGIO CON UNITÀ ACCELEROMETRICHE

A cura di Pasquale Corsonello | **DELIVERABLE WP 2.3**

Circuiti integrati a bassa potenza per sistemi di monitoraggio con unità accelerometriche

Sistemi integrati per il monitoraggio, l'early warning e la mitigazione del rischio idrogeologico lungo le grandi vie di comunicazione"

Premessa

Frane e inondazioni sono un problema di grande rilevanza nel nostro Paese. Negli ultimi anni le vittime e i danni dei disastri idrogeologici hanno raggiunto livelli inaccettabili e impongono un grande e immediato impegno della comunità nazionale per cercare di mitigare il livello di rischio, utilizzando strategie articolate ed efficaci capaci di integrare, in una visione organica, interventi strutturali e non strutturali.

Su questi temi l'Università della Calabria è impegnata da anni in attività di studio e di ricerca di rilevanza nazionale e internazionale e nella diffusione e promozione della cultura della previsione e prevenzione del rischio idrogeologico. Nel 2011 insieme ad altri partner, ha promosso un progetto di ricerca triennale, "Sistemi integrati per il monitoraggio, l'early warning e la mitigazione del rischio idrogeologico lungo le grandi vie di comunicazione", finalizzato allo sviluppo di un sistema complesso e articolato di preannuncio delle frane da impiegare per le fasi di previsione/prevenzione del rischio idrogeologico.

Il Progetto, indicato con l'acronimo LEWIS (Landslide Early Warning Integrated System), è stato svolto, nel periodo 2012-2014, nel quadro del Programma Operativo Nazionale 2007-13 "Ricerca e Competitività".

I risultati conseguiti sono descritti in questa collana di Quaderni PON LEWIS.

Il progetto è stato sviluppato dall'Università della Calabria e Autostrade Tech S.p.A. insieme ai partner industriali Strago e TDGroup, alle Università di Catania, di Reggio Calabria e di Firenze e al CINID (Consorzio Interuniversitario per l'Idrologia). Per l'Ateneo calabrese hanno partecipato diversi laboratori e gruppi di ricerca: CAMILab (con funzione di coordinamento), μWave, Geomatica, Nems, Geotecnica, Dipartimento di matematica.

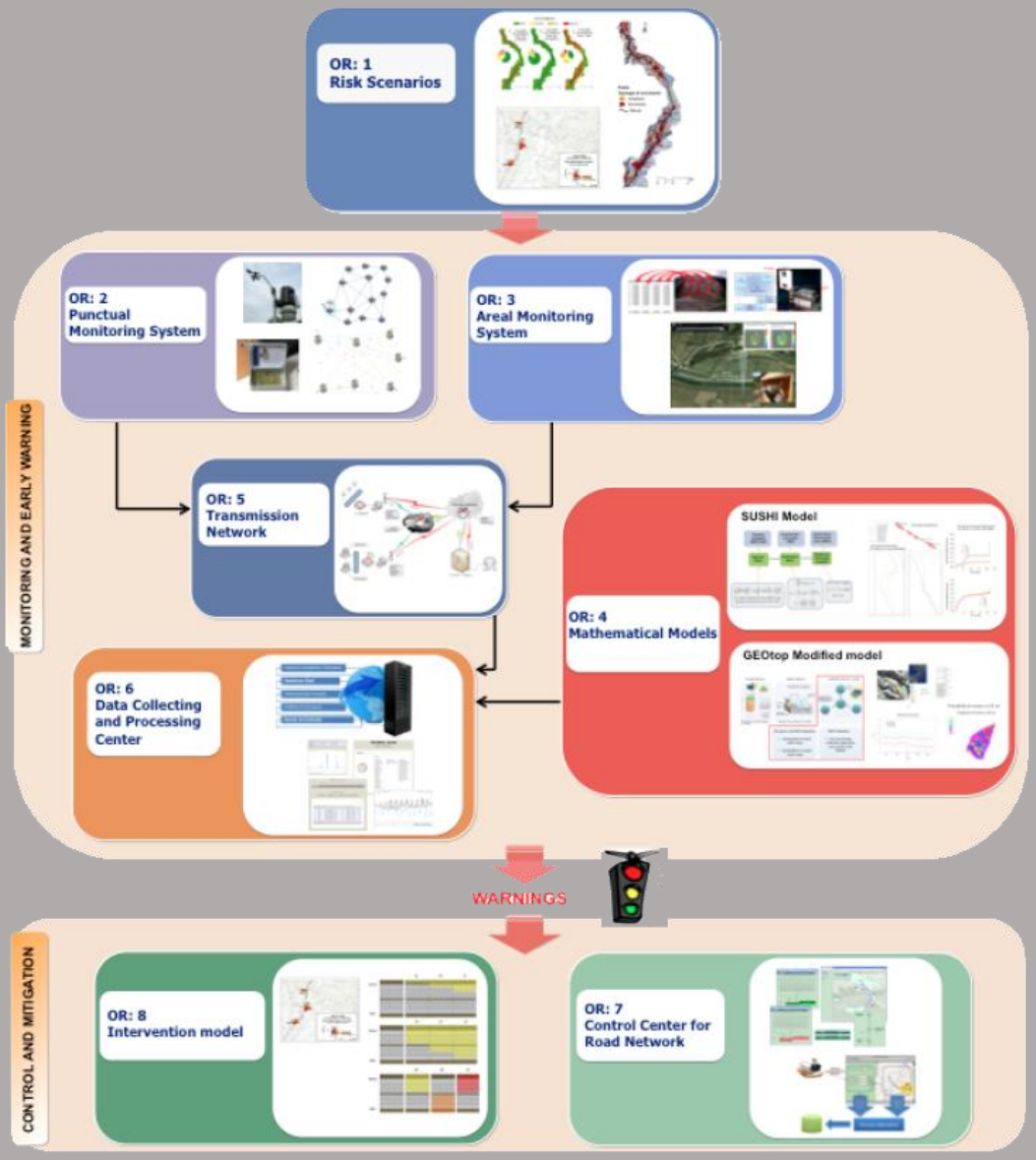

Figura 1 - Articolazione del sistema integrato di monitoraggio dei versanti e di preannuncio dei movimenti franosi

Il progetto è finalizzato allo sviluppo di un sistema di monitoraggio dei versanti e di preannuncio dei movimenti franosi che possono interessare le grandi vie di comunicazione e all'identificazione dei conseguenti interventi non strutturali di mitigazione.

Il sistema è articolato in due sottosistemi (fig. 1):

- ✓ Monitoraggio e preannuncio,
- ✓ Controllo e mitigazione,

che richiedono la preventiva individuazione degli scenari di rischio ossia dei danni che l'eventuale attivazione di una frana può produrre sugli elementi a rischio presenti (infrastruttura viaria, autoveicoli, persone). La procedura originale sviluppata nell'ambito del progetto prevede l'identificazione, lungo il tratto autostradale di interesse, delle aree soggette a movimenti franosi e la conseguente definizione dei relativi scenari di evento e di rischio.

Il sottosistema *Monitoraggio e preannuncio* è formato da diverse componenti: rete di monitoraggio "puntuale" che comprende sensori che misurano localmente l'inizio degli spostamenti superficiali o profondi; rete di monitoraggio "areale" che include sensori che controllano a distanza il fenomeno franoso con tecniche radar; modelli matematici di simulazione dell'innesco e della propagazione dei movimenti franosi. Nel progetto LEWIS sono state sviluppate numerose componenti innovative e sono state modificate e migliorate altre componenti già esistenti. In particolare tra i sensori puntuali sono stati sviluppati i sistemi SMAMID e POIS; tra quelli areali sono stati realizzati un radar in banda L, uno scatterometro, un interferometro; tra i modelli si sono sviluppati e/o migliorati: GEOtop, SUSHI, SCIDDICA.

La raccolta dei dati misurati dai sensori è affidata ad un unico sistema di trasmissione dati che trasmette anche le informazioni necessarie per il funzionamento dei modelli. Il sottosistema è completato da un Centro di acquisizione ed elaborazione dei dati (CAED) che, sulla base dei dati misurati dai sensori e delle indicazioni dei modelli, valuta la situazione di pericolo lungo il tronco autostradale emettendo i relativi livelli di criticità.

I livelli di criticità emessi dal CAED sono l'elemento di collegamento tra il sottosistema *Monitoraggio e preannuncio* e il sottosistema *Controllo e mitigazione*. Gli avvisi di criticità sono acquisiti dal Centro di comando e controllo del traffico (CCCT) che, sulla base di un modello di intervento predefinito, attiva le procedure standardizzate per la mitigazione del rischio, che vanno dalla

sorveglianza diretta del tratto di interesse da parte di squadre tecniche all'interruzione del traffico su entrambe le direzioni di marcia.
Il progetto prevede anche lo sviluppo di attività sperimentali su tre tronchi autostradali lungo la A3, la A16 e la A18, nonché l'erogazione di un Master di secondo livello denominato ESPRI (ESperto in Previsione/Prevenzione Rischio Idrogeologico).
Il progetto di ricerca è stato organizzato in Obiettivi Realizzativi (OR), ciascuno dei quali suddiviso in Work Package (WP), a loro volta articolati in Attività Elementari (AE). In totale erano previste 11 OR, 47 WP e 243 AE. In particolare le OR 1-8 riguardano la ricerca e si articolano in 26 WP e 139 AE. Le OR 9-11 sono dedicate a sperimentazione, governance e trasferimento tecnologico, integrazione e aggiornamento dell'attività di ricerca nella fase di Sviluppo Sperimentale e si articolano complessivamente in 21 WP e 104 AE.
I Quaderni che compongono questa collana sono stati costruiti con riferimento ai singoli WP, per la parte che riguarda la ricerca, e quindi ogni Quaderno contiene la descrizione dei risultati conseguiti nel WP, articolata in base alle AE previste.
Sono, inoltre, previsti altri tre Quaderni:
Quaderno 0 che contiene una descrizione di sintesi, in inglese, dei risultati conseguiti nell'ambito del progetto.
Quaderno 28 che contiene l'informazione relativa alle attività di divulgazione dei risultati scientifici.
Quaderno 29 che contiene la descrizione dei risultati conseguiti con l'attività formativa.
Il Quadro editoriale complessivo è riportato in tabella 1:

QUADERNO	OR	WP	TITOLO
0	-	-	Research outcomes
01 **Parte prima**	1	1.1	Linee guida per l'identificazione di scenari di rischio
01 **Parte seconda**	1	1.1	Linee guida per l'identificazione di scenari di rischio
02	2	2.1	Monitoraggio idrogeologico
03 **Parte prima**	2	2.2	Monitoraggio con unità accelerometriche (Sistema SMAMID)
03 **Parte seconda**	2	2.2	Monitoraggio con unità accelerometriche (Sistema SMAMID)
04	2	2.3	Circuiti integrati a bassa potenza per sistemi di monitoraggio con unità accelerometriche
05	2	2.4	Monitoraggio con sensori puntuali di posizione e inclinazione (Sistema POIS)
06	3	3.1	Sviluppo di uno scatterometro a risoluzione variabile
07	3	3.2	Elettronica di bordo dello scatterometro ed inclinazione
08	3	3.3	Sviluppo di un radar in banda L
09	3	3.4	Tecniche di analisi e sintesi di segnali radar per la simulazione accurata di scenari complessi
10	3	3.5	Elettronica di bordo del radar in banda L

QUADERNO	OR	WP	TITOLO
11	3	3.6	Sistemi interferometrici radar ad apertura sintetica basati a terra
12	4	4.1	Modello areale per il preannuncio delle frane da innesco pluviale (Modello GEOtop)
13	4	4.2	Modelli completi di versante di tipo puntuale per il preannuncio di movimenti franosi (Modello SUSHI)
14	4	4.3	Modelli di propagazione delle frane tipo colate (Modello SCIDDICA)
15	5	5.1	Rete Wireless di Telecomunicazioni: sviluppo e scelta dei parametri di progetto
16	6	6.1	CAED. Acquisizione dati: architettura del sistema
17	6	6.2	CAED. Elaborazione dei dati
18	7	7.1	CCCT. Progettazione
19	7	7.2	CCCT. Interfaccia verso il centro di acquisizione ed elaborazione dati
20	7	7.3	CCCT. Interfaccia con altre centrali operative e canali di diffusione delle notizie
21	7	7.4	CCCT. Modulo per la presentazione e convalida delle allerte
22	7	7.5	CCCT. Modulo per la gestione delle informazioni di traffico
23	7	7.6	CCCT. Integrazioni con moduli speciali

QUADERNO	OR	WP	TITOLO
24	8	8.1	Definizione del modello di intervento e predisposizione del Piano di Emergenza
25	8	8.2	CCCT. Gestione delivery allerte e attivazione squadre d'intervento
26	8	8.3	CCCT. Gestione percorsi alternativi
27	9	9.1 - 9.11	Sperimentazione
28	10	10.1 - 10.2	Piano di comunicazione e diffusione
29	-	-	Master ESPRI (Esperto in Previsione/Prevenzione Rischio Idrogeologico)

Tabella 1 - Quadro editoriale complessivo della collana di Quaderni PON LEWIS

31 dicembre 2014

Il Responsabile Scientifico del progetto PON LEWIS

Pasquale Versace

INDICE

ATTIVITA' ELEMENTARE 2.3.4

45 **Implementazione di una Release Image su Memoria eNVM**

55 APPENDICE

Introduzione

L'attività relativa al WP2.3 ha riguardato lo studio di soluzioni alternative a quelle attualmente utilizzate per la realizzazione del sistema SMAMID. Il sistema SMAMID consiste di una rete wireless di nodi di misura finalizzata al monitoraggio superficiale di rilievi franosi. Peculiari specifiche del sistema sono l'impiego di elettronica a basso costo, per la realizzazione delle capacità di elaborazione dei singoli nodi di misura, e l'utilizzo di soluzioni hardware a ridotto consumo energetico, per garantire il monitoraggio a lungo termine dell'area di interesse. Un nodo di misura del sistema SMAMID consta, al momento, di un sensore accelerometrico, un microcontrollore a 8 bit con limitata capacità di elaborazione (PIC18LF8722), una memoria di supporto all'elaborazione e un modulo radio interfacciato con il microcontrollore.

Il Gruppo di Ricerca ha valutato approcci realizzativi alternativi con particolare interesse verso i corrispondenti rapporti costi/benefici. Si è deciso di affrontare prioritariamente un intervento di miglioramento del *tradeoff* performance/potenza dissipata dei nodi cosiddetti Master.

Deliverable Fisico

L'attività di indagine sui possibili approcci realizzativi alternativi a quelli correntemente utilizzati nella realizzazione del dispositivo SMAMID si è conclusa con l'individuazione dei Field Programmable Gate Array (FPGA) come il più idoneo. Il Gruppo di Ricerca ha identificato nelle FPGA live-at-power-up una soluzione congrua e conveniente. E' stata selezionata la famiglia SmartFusion della Actel comprende chip FPGA con un livello di integrazione adeguato e con un costo relativamente contenuto. Essendo dotati di un processore ARM Cortex-M3, i chip SmartFusion si prestano molto bene alla realizzazione di sistemi embedded. E' stata progettata una architettura prototipale con associato un sistema di sviluppo semplificato per poter sperimentare l'acquisizione di segnali provenienti dai sensori accelerometrici impiegati nel dispositivo SMAMID.
Il prototipo è illustrato nella Figura 0.

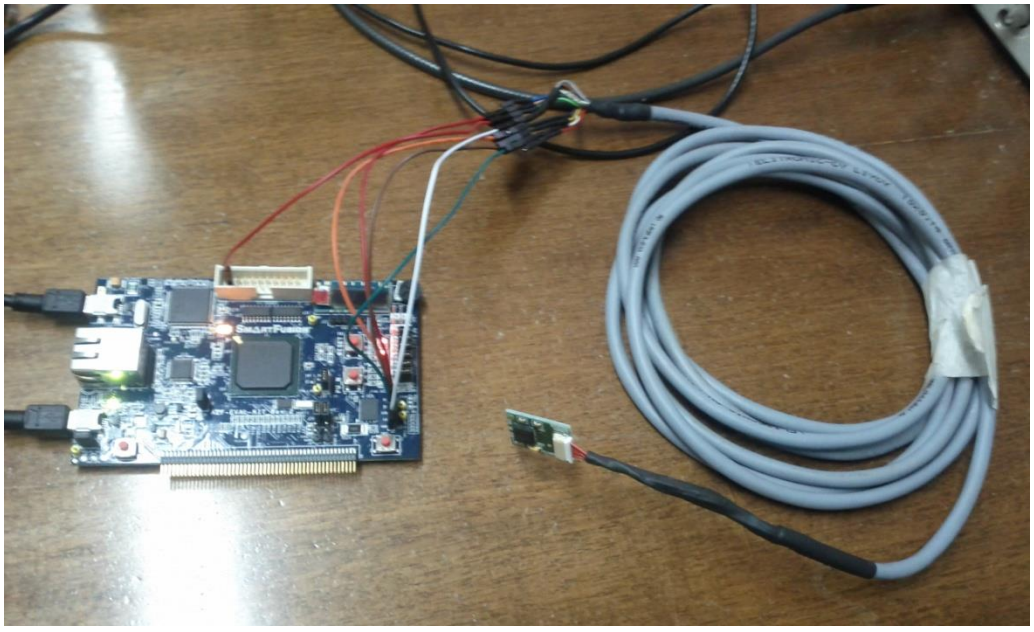

Figura 0

Il prototipo è stato consegnato a Strago S.p.A in data 5/4/2013. Le attività relative al WP sono proseguite fino al 30/6/2013 per consentire a Strago S.p.A. le sperimentazioni nei loro laboratori.

ATTIVITÀ ELEMENTARE 2.3.2

Deliverable Documentale

Il presente Deliverable ha lo scopo di addestrare all'utilizzo del sistema prototipale realizzato nell'ambito del WP2.3. Nel seguito di questo documento si illustra l'utilizzo dei tools Microsemi/Actel per l'implementazione di applicazioni embedded su piattaforma cSoC SmartFusion [1]. In particolare, dopo avere descritto l'architettura hardware del cSoC e del relativo sottosistema microcontrollore (MSS), saranno mostrati i principali step del design flow necessari allo sviluppo di una tipica applicazione software sulla piattaforma prototipale di cui sopra.

REQUISITI HARDWARE

1. SmartFusion Evaluation Kit Board [2]
2. Due cavi mini-USB necessari alla programmazione del chip SmartFusion ed al collegamento della board al PC host su protocollo USB/RS-232.

REQUISITI SOFTWARE

I tools software richiesti dal sistema di sviluppo sono i seguenti:

1. Microsemi Libero SOC v10.1
2. Microsemi Libero SOC v10.1 SP3
3. Drivers CP2102 per il support "USB to RS232 Bridge" .

Tali software, da installare su un sistema Windows (Wins 7 Pro oppure WinXP SP3) sono forniti nella cartella "tools" del DVD allegato. Sarà necessario

registrarsi sul sito Actel e richiedere una licenza “free”. Di seguito sono indicati in dettaglio i passi da effettuare.

2.3.2.1 INSTALLAZIONE DI LIBERO SoC v10.1

Lanciare il programma di installazione LiberoSoCv101DVD.

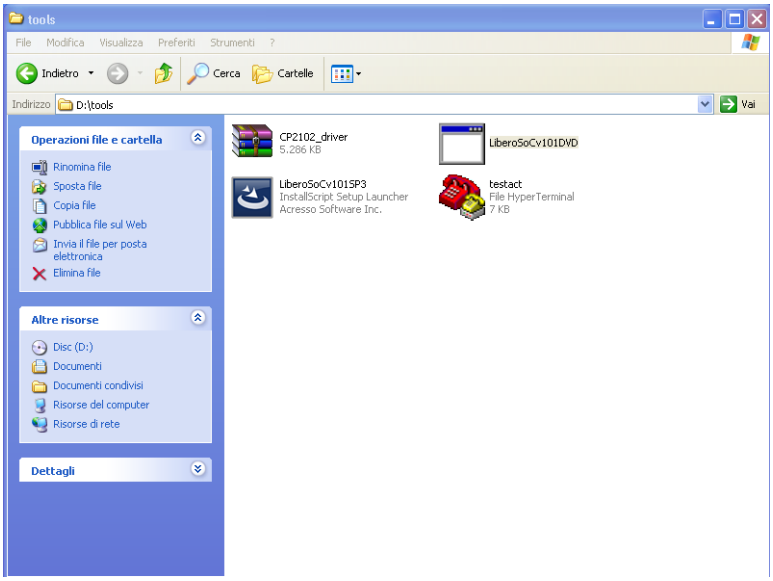

Figura 1

Il programma, prima di avviarsi, richiede un reboot del PC (sotto WinXP).

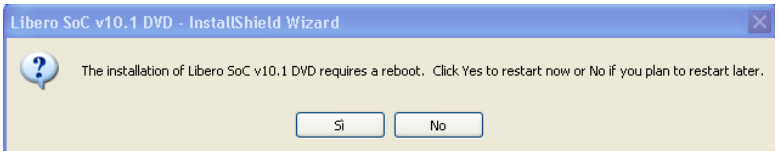

Figura 2

Dopo il reboot del PC, il programma riparte automaticamente.

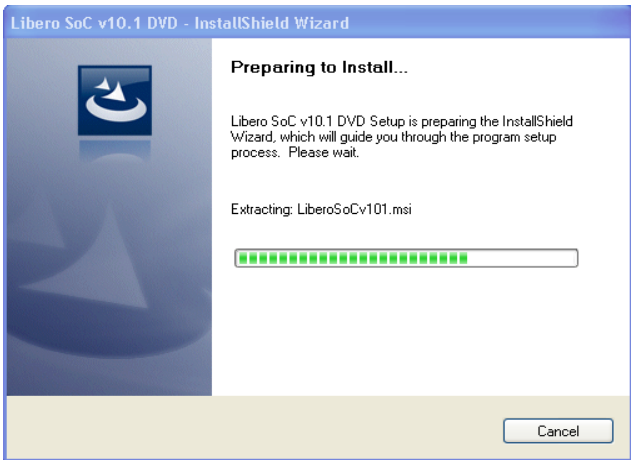

Figura 3

A questo punto viene richiesto di specificare il tipo di installazione, che dovrà essere quella completa come indicato nella figura seguente.

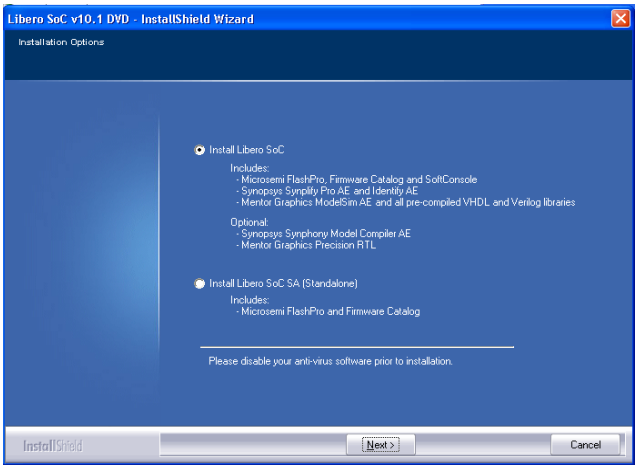

Figura 4

Proseguire con le impostazioni dell'installazione come indicato figure sottostanti.

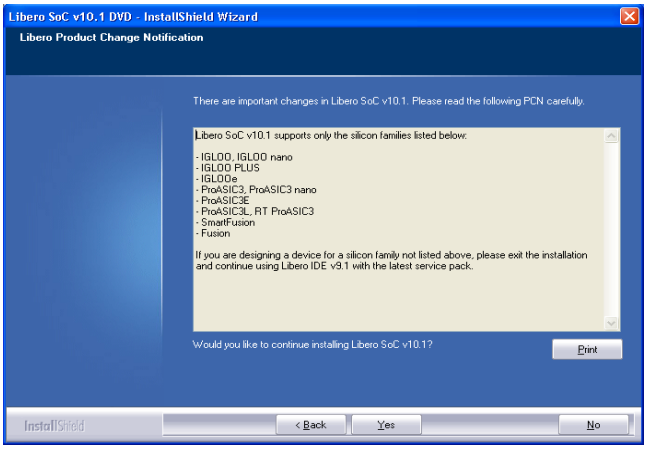

Figura 5

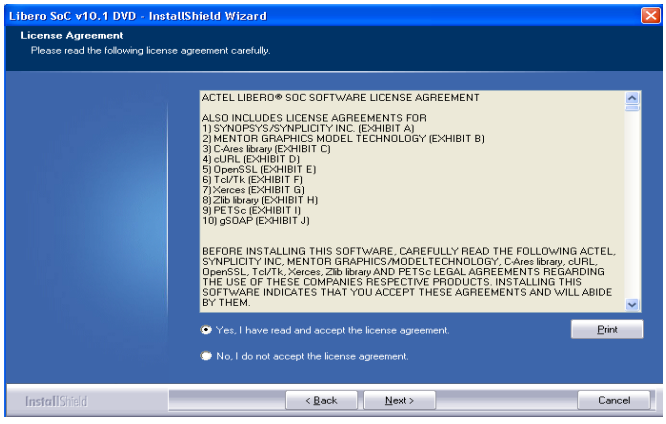

Figura 6

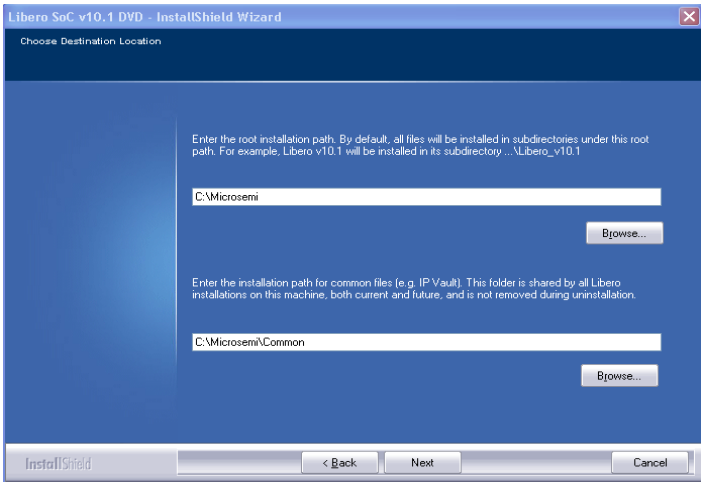

Figura 7

Proseguire come indicato di seguito, escludendo dall'installazione i tools Synphony e Precision RTL. Cliccare infine su "Install".

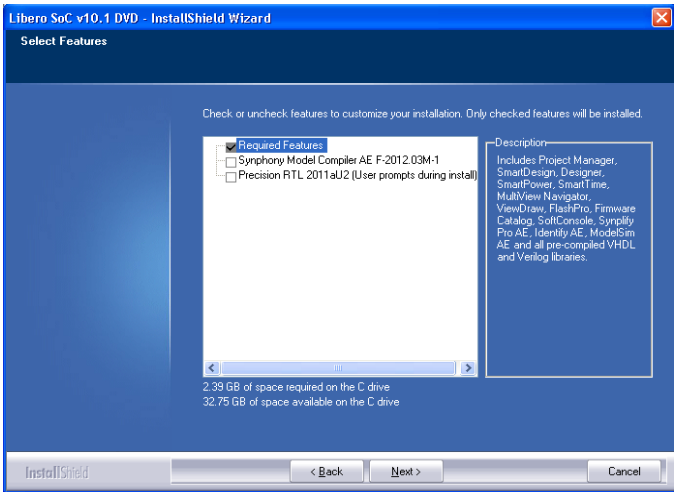

Figura 8

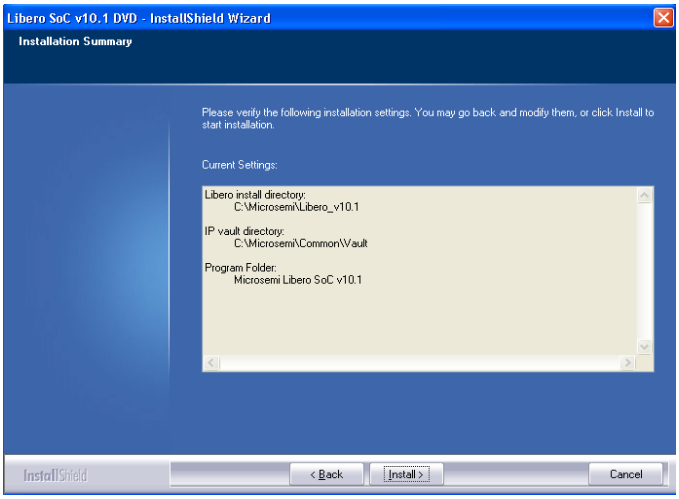

Figura 9

La procedura effettua l'eventuale installazione dei driver per come indicato nelle tre figure sottostanti. Si trascuri il *warning* relativo alla compatibilità con il S.O.

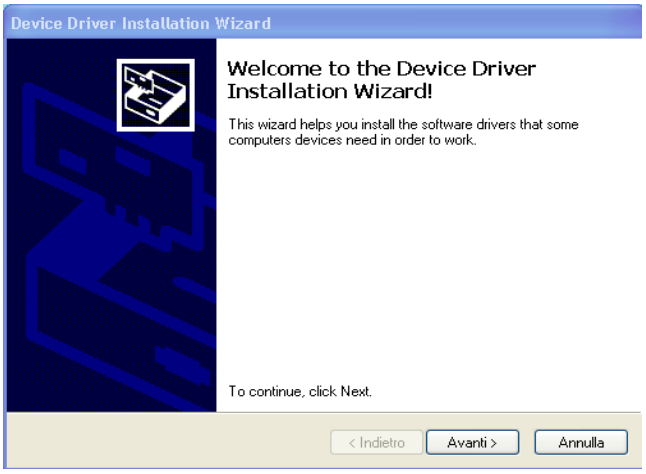

Figura 10

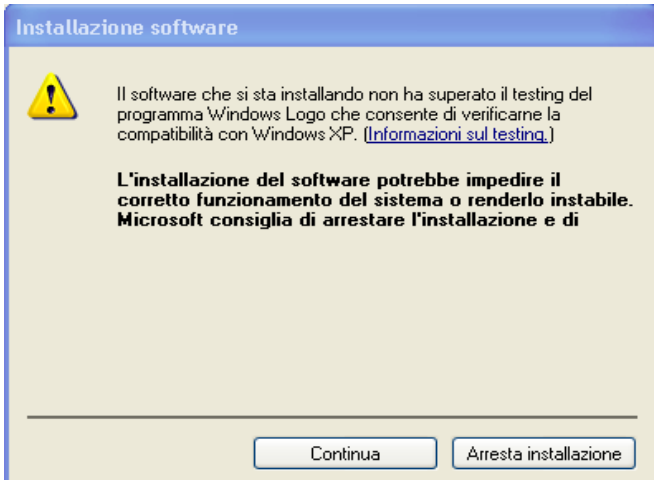

Figura 11

Figura 12

Vengono installati i tools che compongono il pacchetto di Libero.

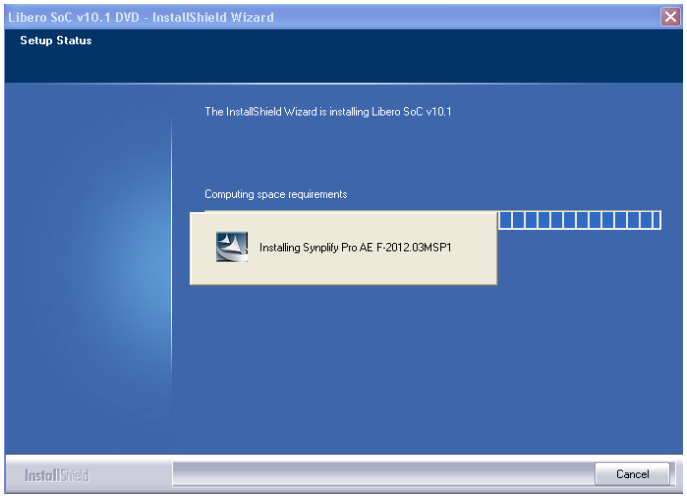

Figura 13

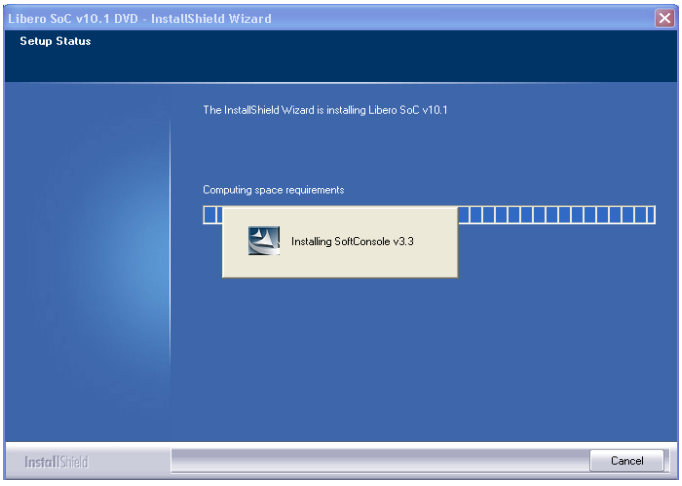

Figura 14

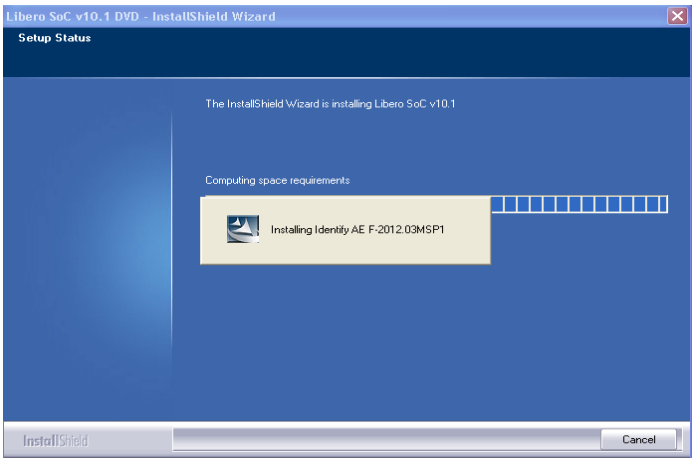

Figura 15

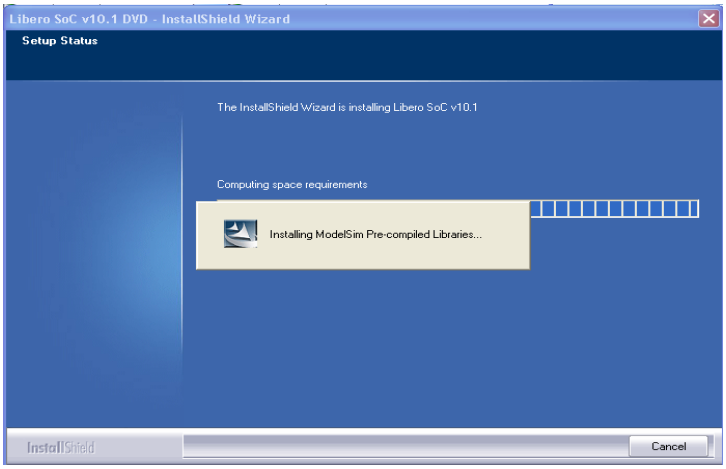

Figura 16

Come illustrato in figura 17, premere OK senza tuttavia inserire i cavi USB nella board.

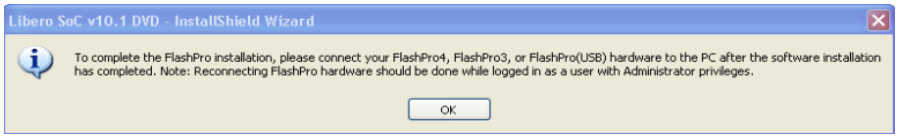

Figura 17

Al termine della procedura di installazione, cliccando sul tasto “Next”, inizia la fase di richiesta di una licenza del software Libero.

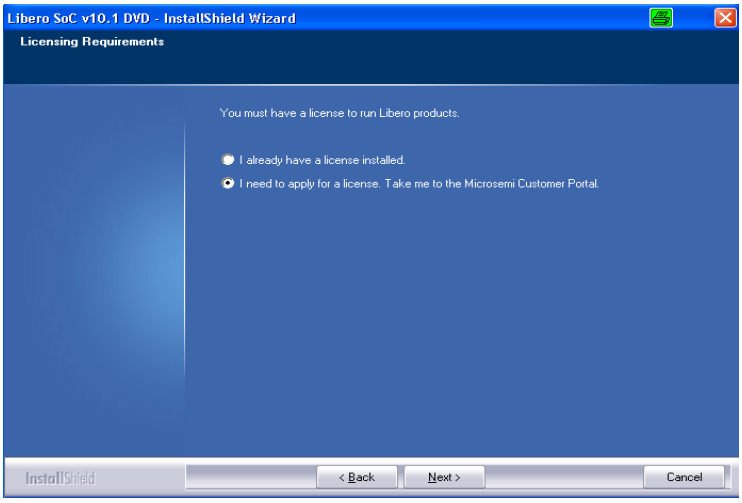

Figura 18

2.3.2.2 RICHIESTA ED INSTALLAZIONE DEL FILE DI LICENZA

Si apre il browser di sistema che si connette automaticamente sul sito di Actel.

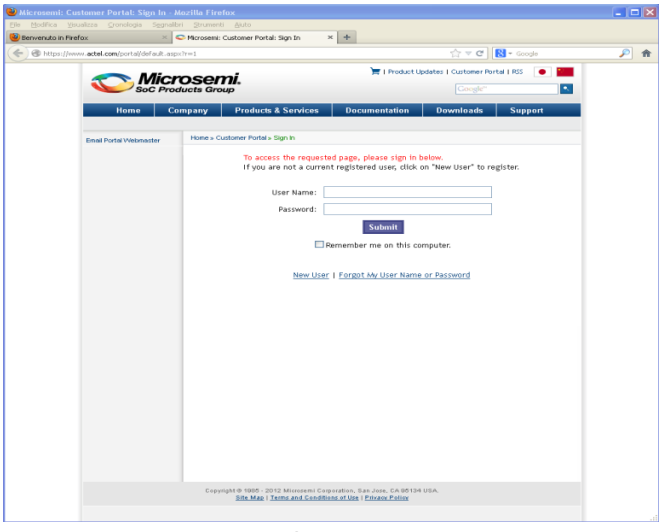

Figura 19

Cliccare su “New user” in modo da avviare la procedura di registrazione e attivazione di un nuovo account sul sito Actel.

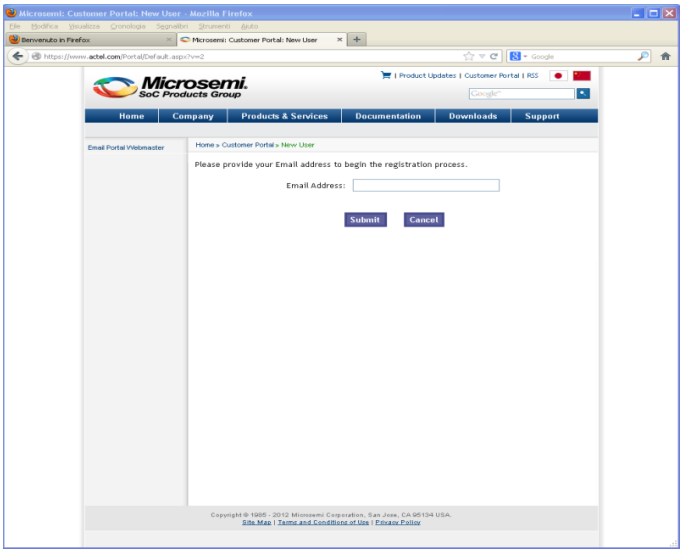

Figura 20

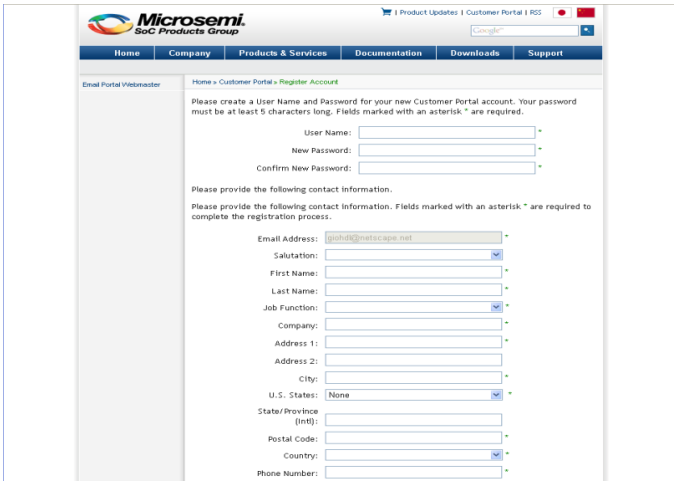

Figura 21

Cliccare su “Pre License Survey” fornendo i dati richiesti.

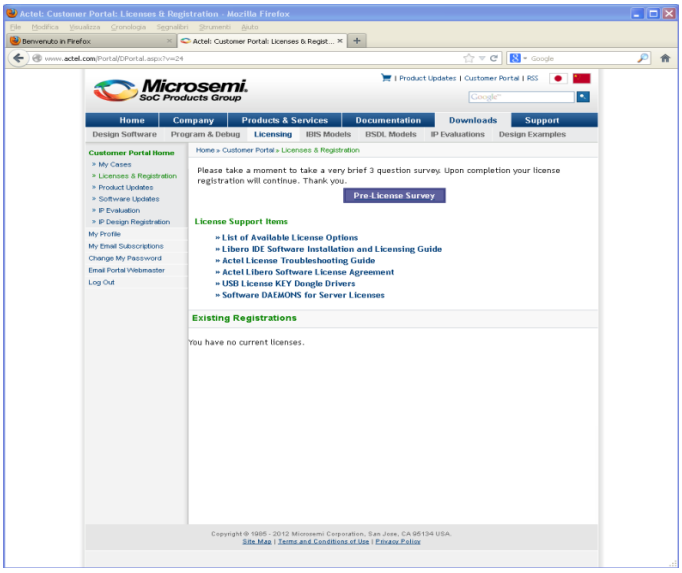

Figura 22

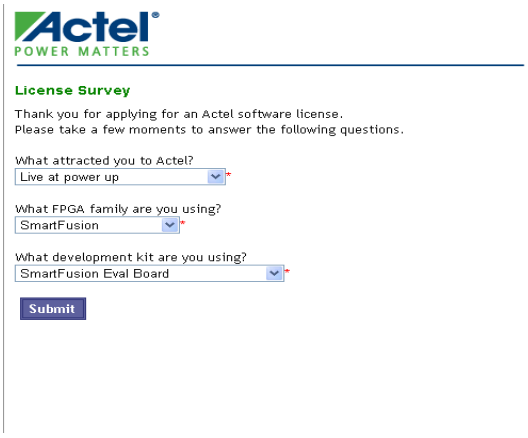

Figura 23

Dopo aver completato la registrazione, si potrà cliccare su “Request Free License”.

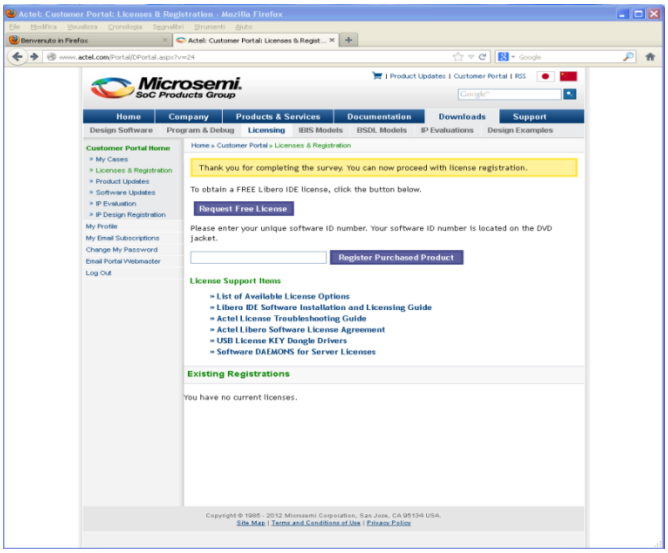

Figura 24

La licenza da richiedere è del tipo “Libero Gold Node Locked for Windows”, di validità annuale.

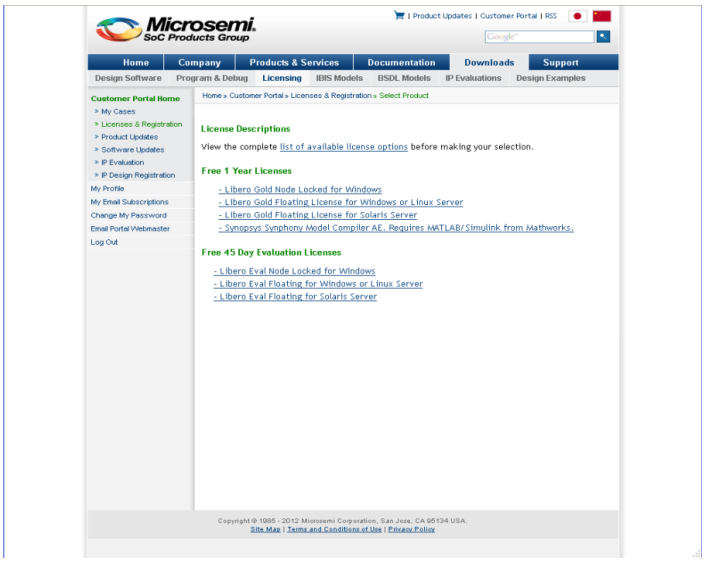

Figura 25

Specificare il valore del numero seriale “DiskID” del proprio disco rigido, che può essere ottenuto con la semplice istruzione “dir”, impartita dalla riga di comando di Windows.

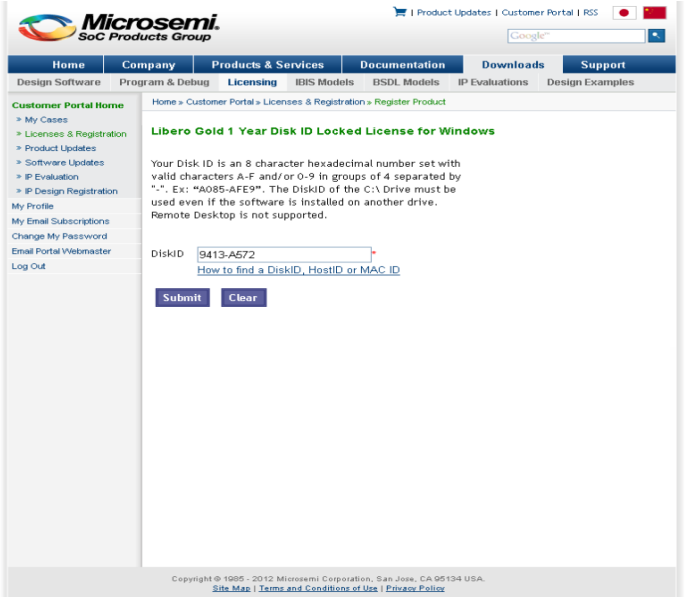

Figura 26

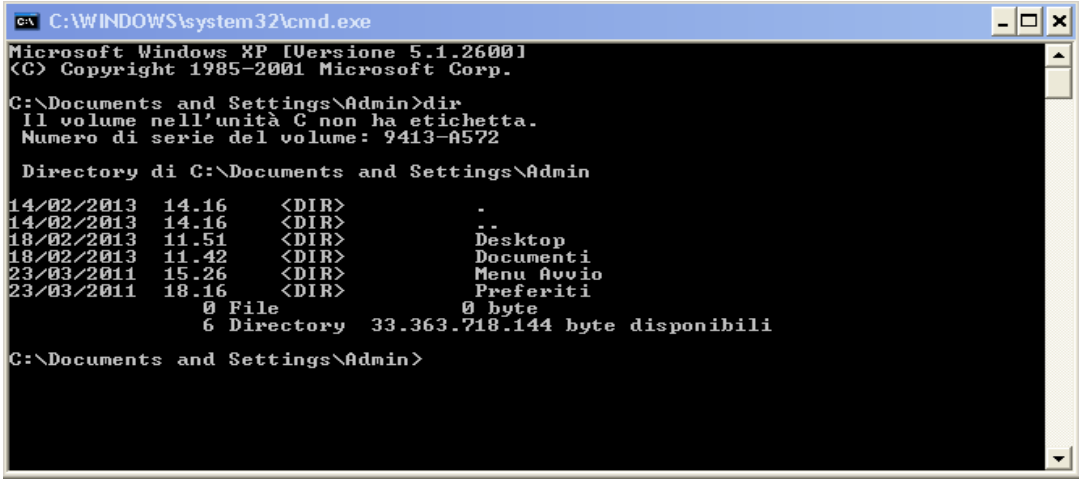

Figura 27

A questo punto, come indicato nella figura seguente, si riceverà sul proprio indirizzo email specificato in fase di registrazione, un file di licenza “license.dat”.

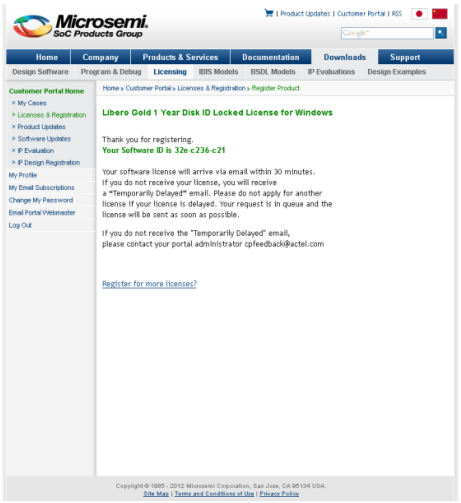

Figura 28

Per completare la fase di installazione della licenza, copiare il file “license.dat” in una cartella opportuna. Impostare quindi dal pannello di controllo di Windows la variabile utente “LM_LICENSE_FILE” al path completo del file di licenza, come illustrato di seguito.

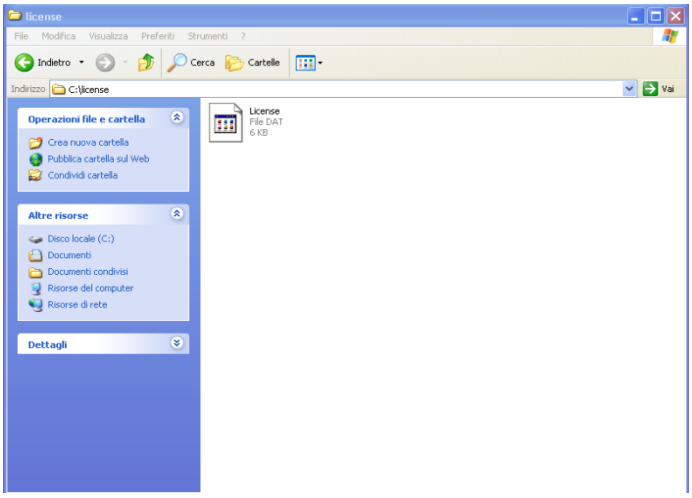

Figura 29

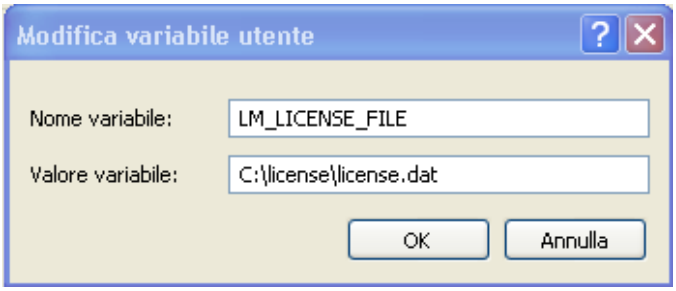

Figura 30

Riavviare a questo punto il PC.

2.3.2.3 INSTALLAZIONE DEL SERVICE PACK LiberoSP3

Lanciare il programma LiberoSoCv101SP3 presente nella cartella “tools”.

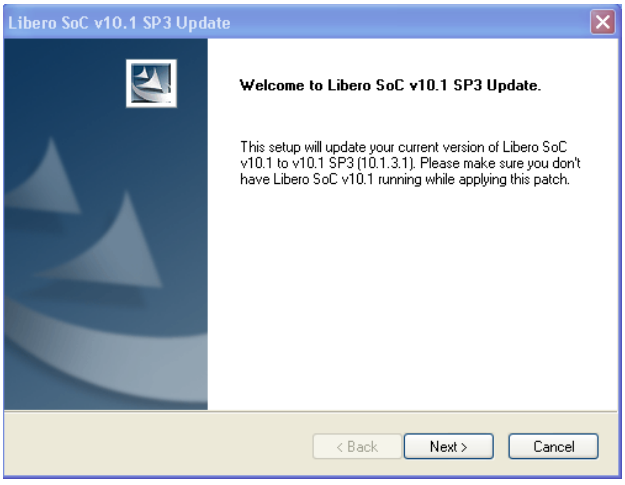

Figura 31

Procedere come illustrato nelle figure seguenti accettando i valori di default forniti dal programma.

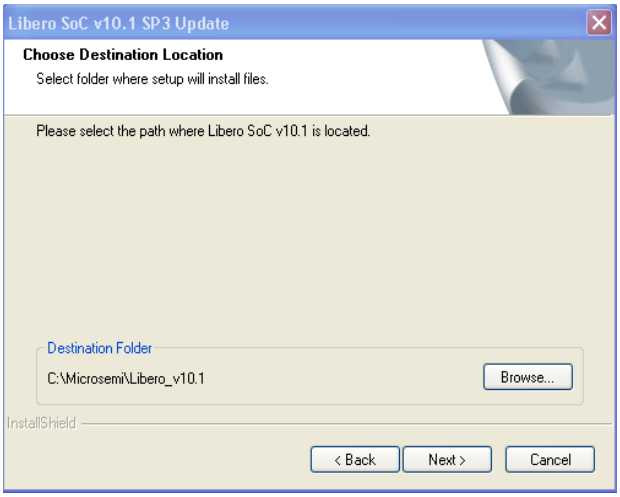

Figura 32

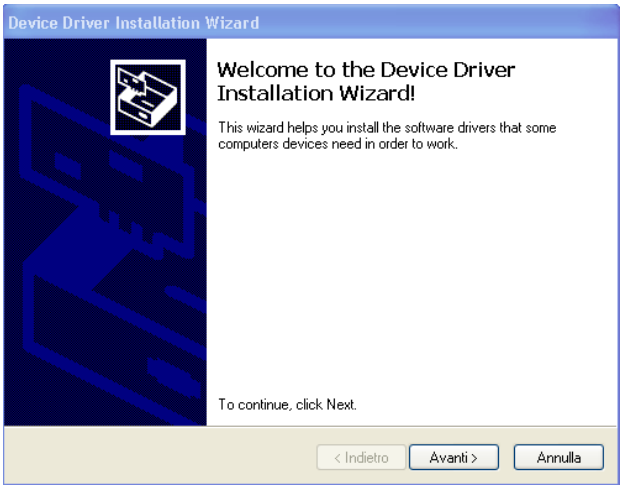

Figura 33

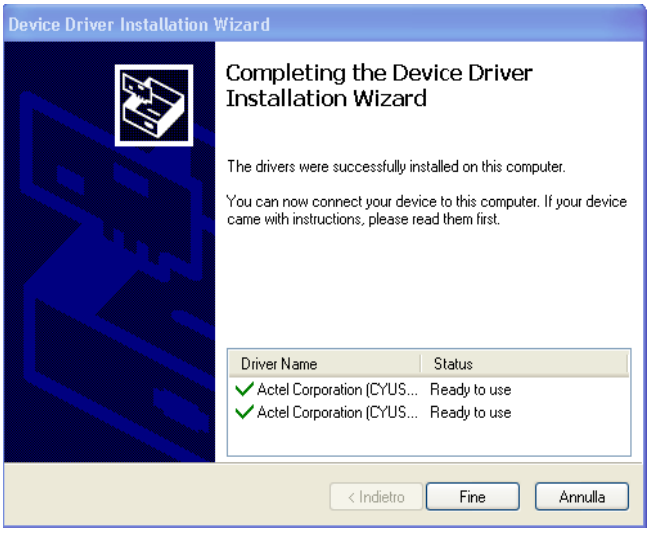

Figura 34

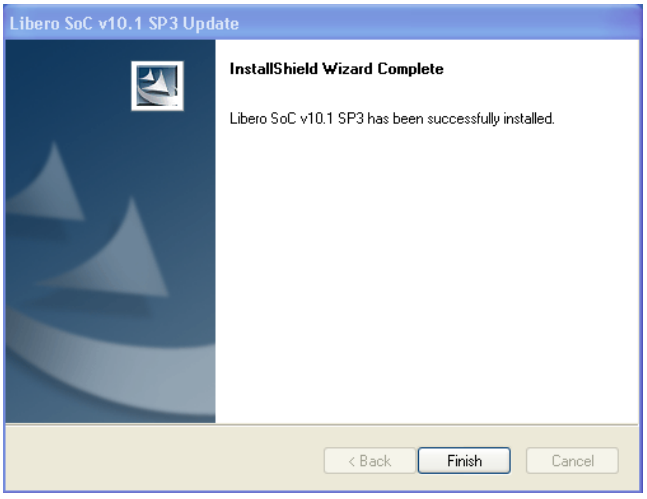

Figura 35

2.3.2.4 INSTALLAZIONE DEI DRIVER USB-RS232 BRIDGE

Copiare il file CP2102_driver presente nella cartella “tools” in una cartella temporanea.

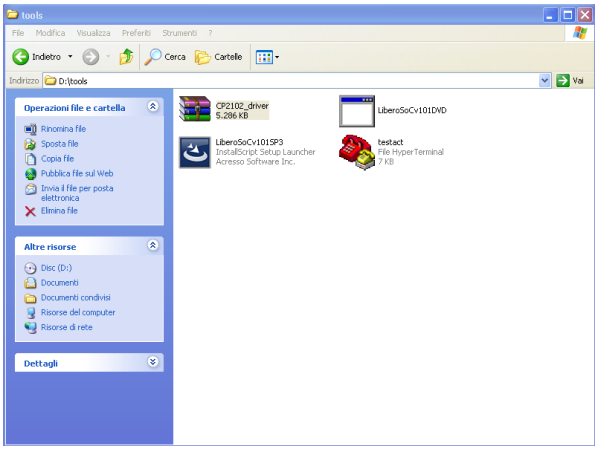

Figura 36

Scompattare il file e lanciare l’installazione.

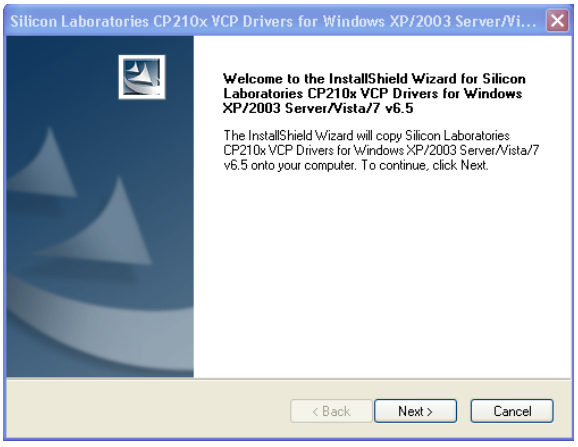

Figura 37

Procedere come indicato nella figura sottostante settando la voce “Launch the CP210x VCP Driver Installer”.

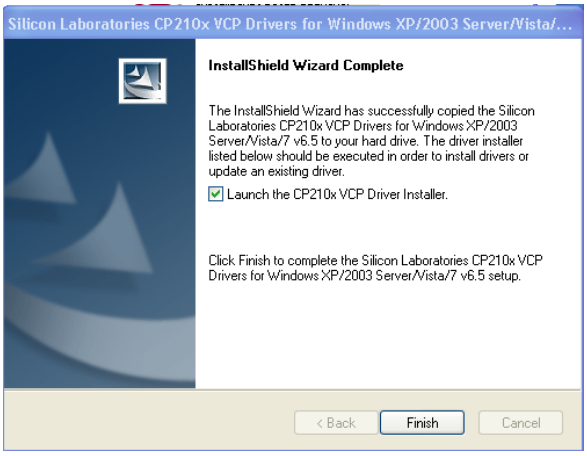

Figura 38

Cliccare infine su “Install” per avviare l’installazione.

Figura 39

Effettuare il reboot del PC.

Per completare l'installazione dei drivers si proceda a collegare la board fornita al PC host mediante i due connettori USB. Appariranno una serie di finestre di Windows. Si scelgano le opzioni indicate nelle figure seguenti.

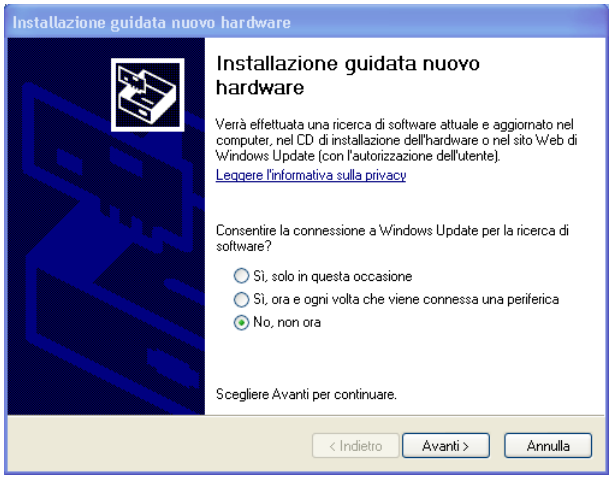

Figura 40

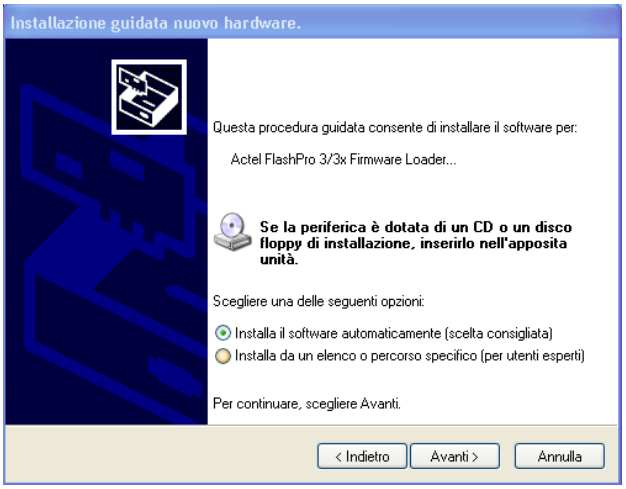

Figura 41

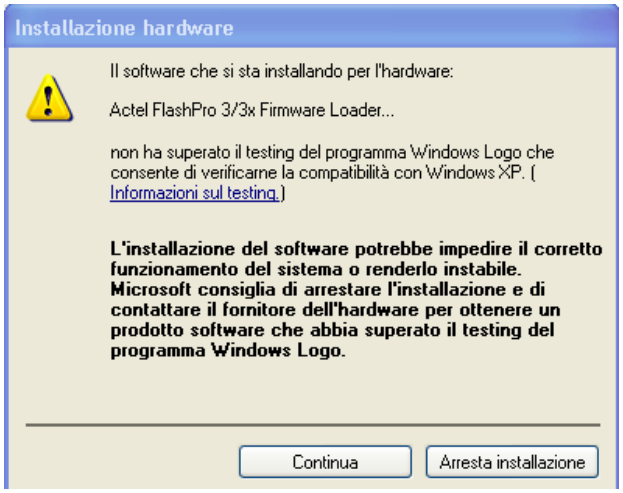

Figura 42

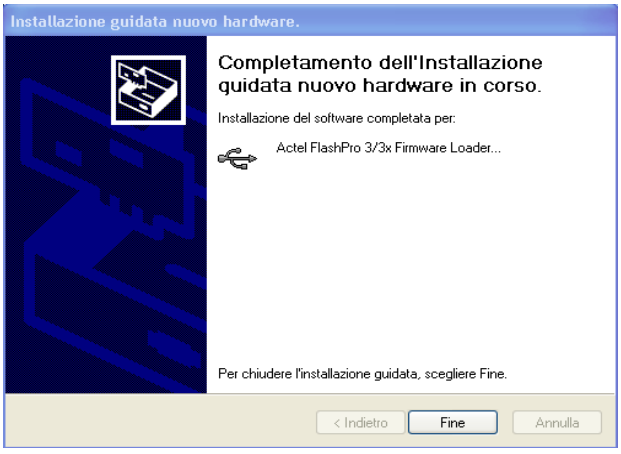

Figura 43

Riavviare nuovamente il PC.

2.3.2.5 DESCRIZIONE DEL PROGETTO

Il progetto "smartdemo" cui si fa riferimento nel seguito, deve essere prelevato dalla cartella “project” del DVD fornito, ed estratto nella cartella C:\Actelprj\kit2\.

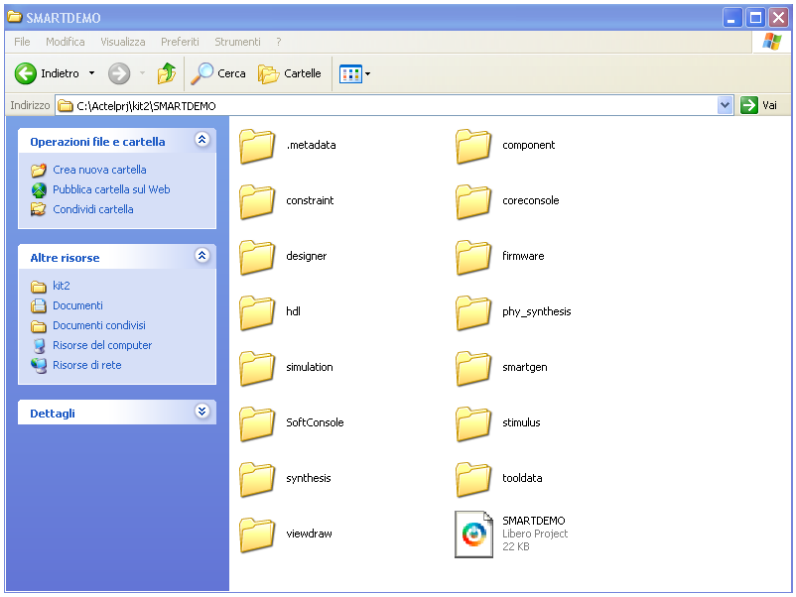

Figura 44

La figura 45 mostra l'albero delle sottodirectory create. Di particolare importanza sono:

- la cartella “designer” che contiene la porzione hardware del progetto ed i files necessari alla programmazione del chip SmartFusion
- la cartella “SoftConsole”, che contiene la porzione software del progetto, include i drivers delle periferiche ed i sorgenti dell’applicazione demo.

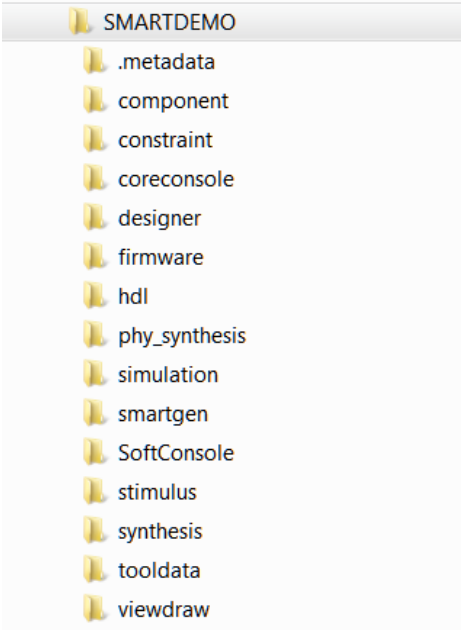

Figura 45

Per quanto riguarda la porzione hardware del progetto, come si può osservare nelle figure 46 e 47, è stato implementato un ampio insieme di periferiche disponibili nell'architettura MSS dello SmartFusion. Tali periferiche consentono di interagire con i componenti presenti sulla board e di comunicare i dati al PC host tramite l'interfaccia USB/RS-232. Tale insieme comprende:

- Processore ARM Cortex-M3 @ 100MHz
- Controller Serial Peripheral Interface (SPI) per la gestione di una memoria EEPROM esterna al chip (SPI_0)
- Secondo controller per la gestione di un eventuale periferica SPI esterna(SPI_1)
- Controller Inter- Integrated Circuit (I2C), che gestisce le funzioni del display OLED (I2C_0)
- Secondo controller per la gestione di un eventuale periferica I2C esterna (I2C_1)
- System timer che genera un interrupt periodico verso il processore
- Modulo Analog Compute Engine (ACE) che implementa due canali di conversione di tensione A/D (TM0_VOLTAGE,MSS_VOL_0)

- Modulo General Purpose Input/Output (GPIO) digitale che consente di gestire:
  - due ingressi provenienti dai push-buttons (SW1, SW2)
  - quattro uscite connesse agli user leds (LED0..LED3)
  - due uscite accessibili dall'esterno (DIGOUT01,DIGOUT1)
- Controller UART (UART_0)
- Controller MAC Ethernet (MAC)

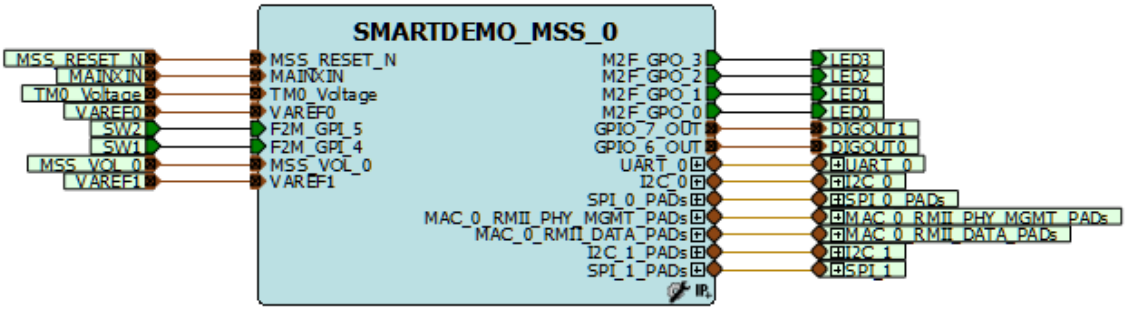

Figura 46

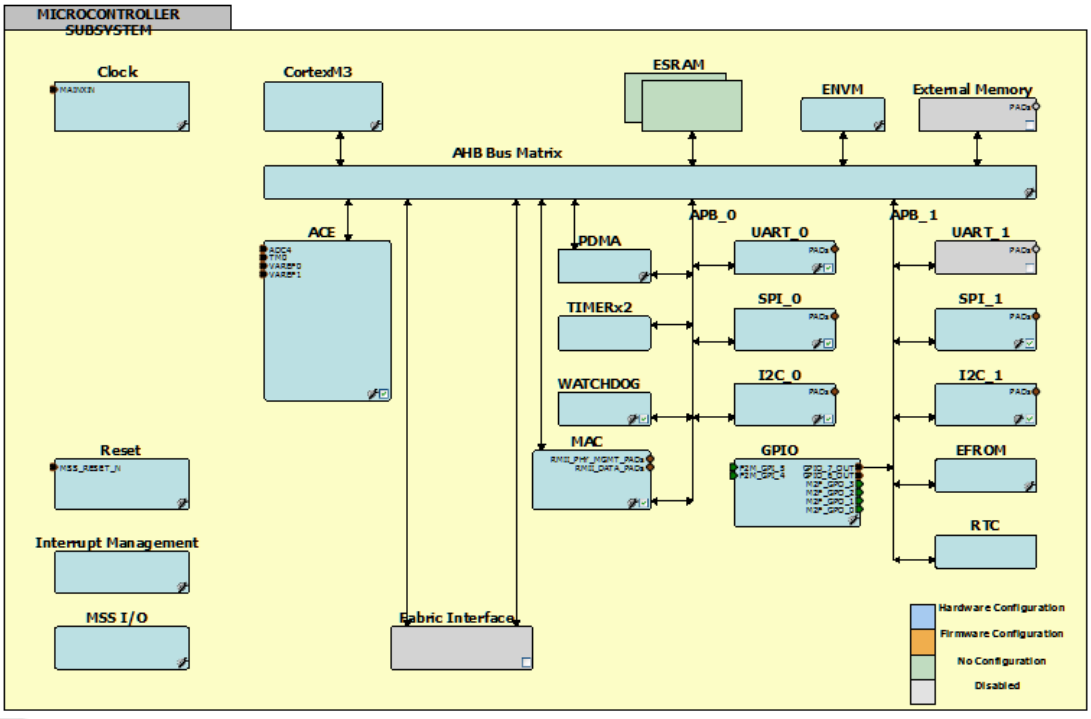

Figura 47

La figura 48 riporta una foto della board presente nel kit.

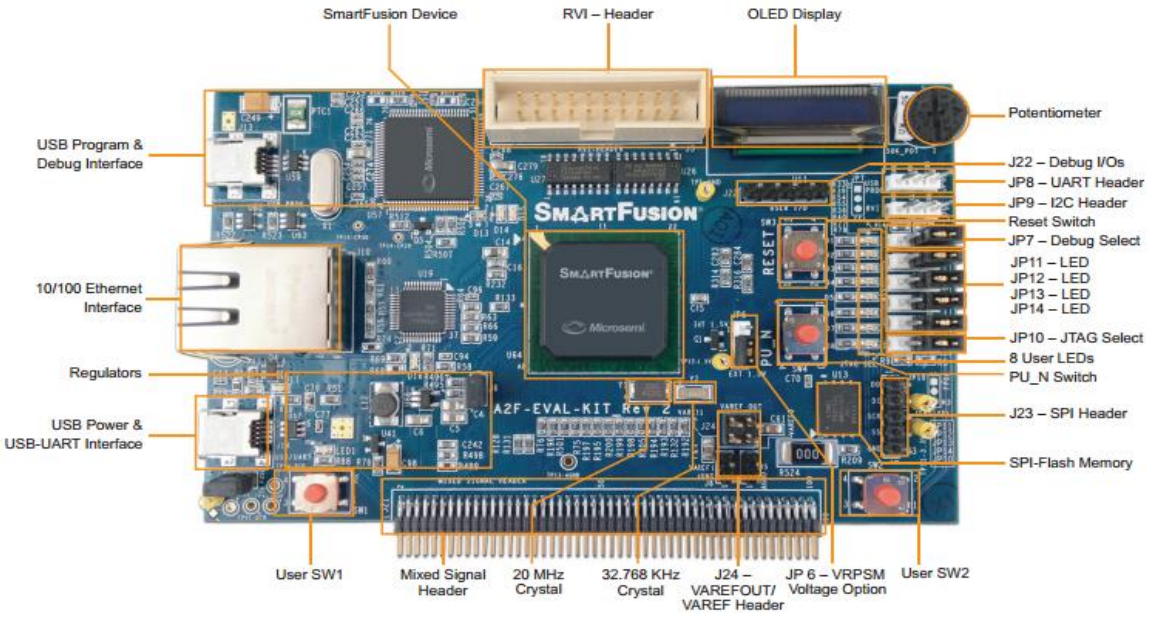

Figura 48

Le linee digitali (GPIO) ed analogiche (ACE), in particolare, sono connesse ai componenti della board [3] per come specificato in tabella 1.

SEGNALE	SMARTFUSION PIN	BOARD
LED0	B19	Led D1
LED1	B20	Led D2
LED2	C19	Led D3
LED3	H17	Led D4
DIGOUT0	V2	Connector mixed–signal J21 - PIN14
DIGOUT1	W2	Connector mixed–signal J21 - PIN16
SW1	G19	Button SW1
SW2	G20	Button SW2
TM0_VOLTAGE	W8	Potentiometer
MSS_VOL_0	U12	Connector mixed–signal J21 - PIN77

Tabella 1

Le connessioni dettagliate dei segnali sui connettori della board sono illustrate nella User Guide del kit [3] fornita nella cartella “doc” del DVD allegato. Il progetto hardware di esempio è stato precaricato nella flash eNVM dello SmartFusion. I passi fondamentali del design flow utilizzati per il test del progetto “smartdemo” sono i seguenti:

a) Esecuzione del codice dell’applicazione. Il programma potrà essere modificato e rieseguito effettuandone il debug tramite il tool SoftConsole.
b) Implementazione di una release image, da caricare nella flash eNVM. In tal modo il programma, senza necessità di debug, sarà eseguito all’accensione della board oppure in seguito alla pressione del tasto reset.

Tali passi sono descritti di seguito.

ATTIVITÀ ELEMENTARE 2.3.3

Esecuzione e Debug dell'Applicazione

Dopo aver installato i tools software e i files di progetto come descritto precedentemente, si proceda a collegare la board al PC host mediante i due connettori USB. Come indicato in figura 48, il connettore in alto viene utilizzato durante la fase di programmazione/debug, mentre quello in basso per alimentare la board ed implementare la connessione RS-232.

1. Lanciare da Windows il tool SoftConsole IDE [6]

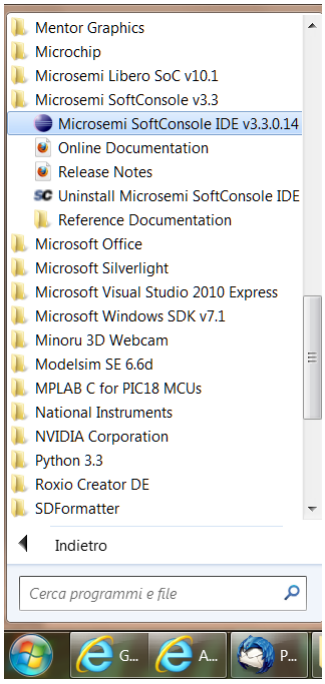

Figura 49

2. Selezionare il workspace relativo al progetto “smartdemo” dalla cartella “C:\Actelprj\kit2\SMARTDEMO\SoftConsole\OLED_MSS_MSS_CM3_0”

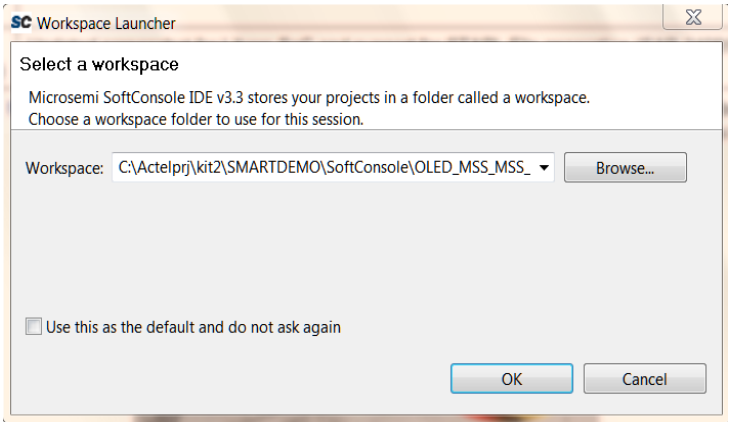

Figura 50

3. Cliccare su “File->Import”

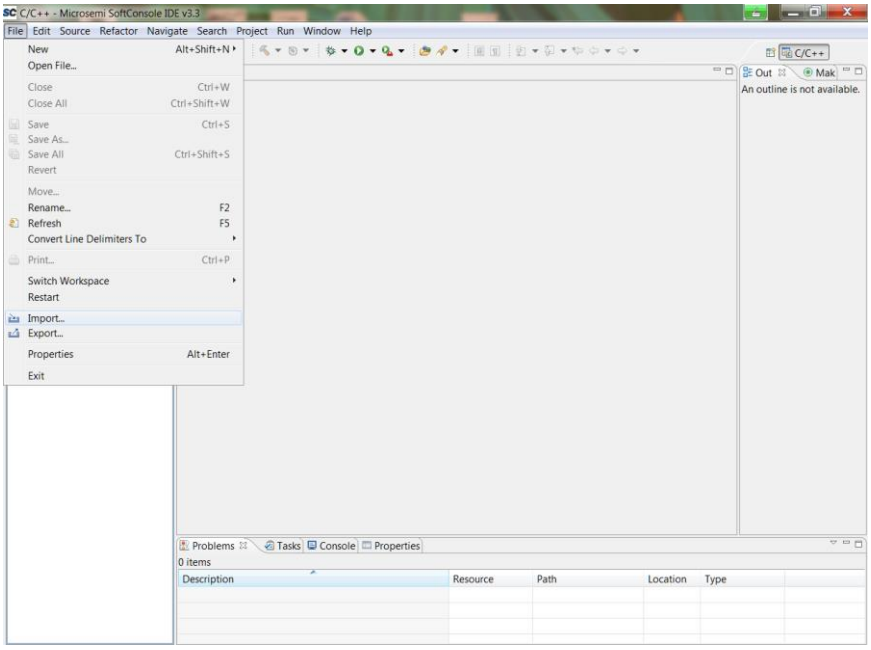

Figura 51

4. Selezionare “Existing Projects into WorkSpace”

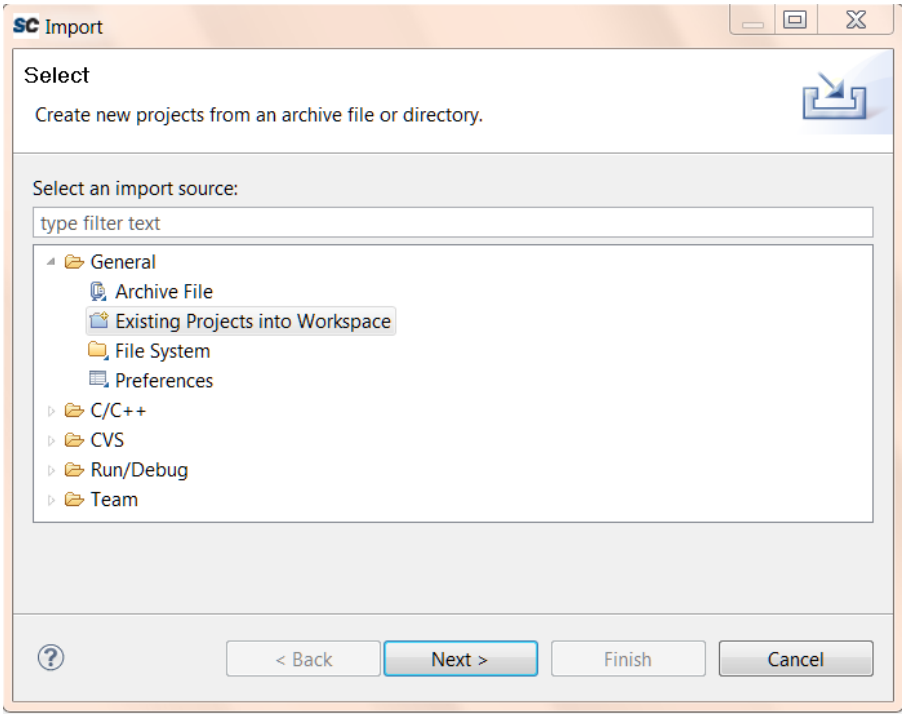

Figura 52

5. Premere “Browse” e selezionare come root directory la cartella “C:\Actelprj\kit2\SMARTDEMO\SoftConsole\OLED_MSS_MSS_CM3_0”. Appariranno i due progetti di interesse con estensione “app” e “hw_platform”. Premere “Finish” in modo da completare la procedura d’importazione.

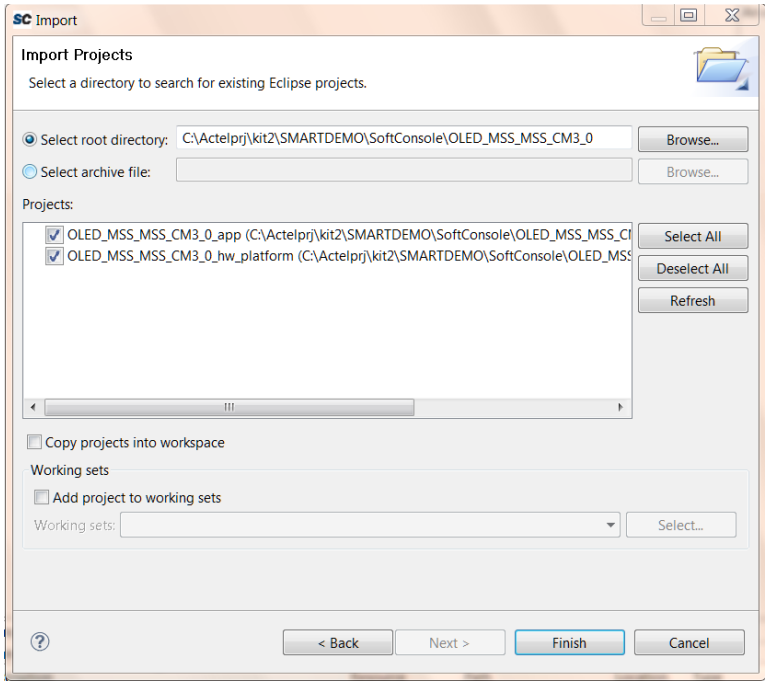

Figura 53

6. In SoftConsole, impostare la “C/C++ perspective” cliccando sul tasto in alto a destra. Selezionare i progetti “hw_platform” e “app” nella finestra “Project Explorer”. Premere col tasto destro e cliccare quindi su “Build Configurations->Set Active->Debug”.

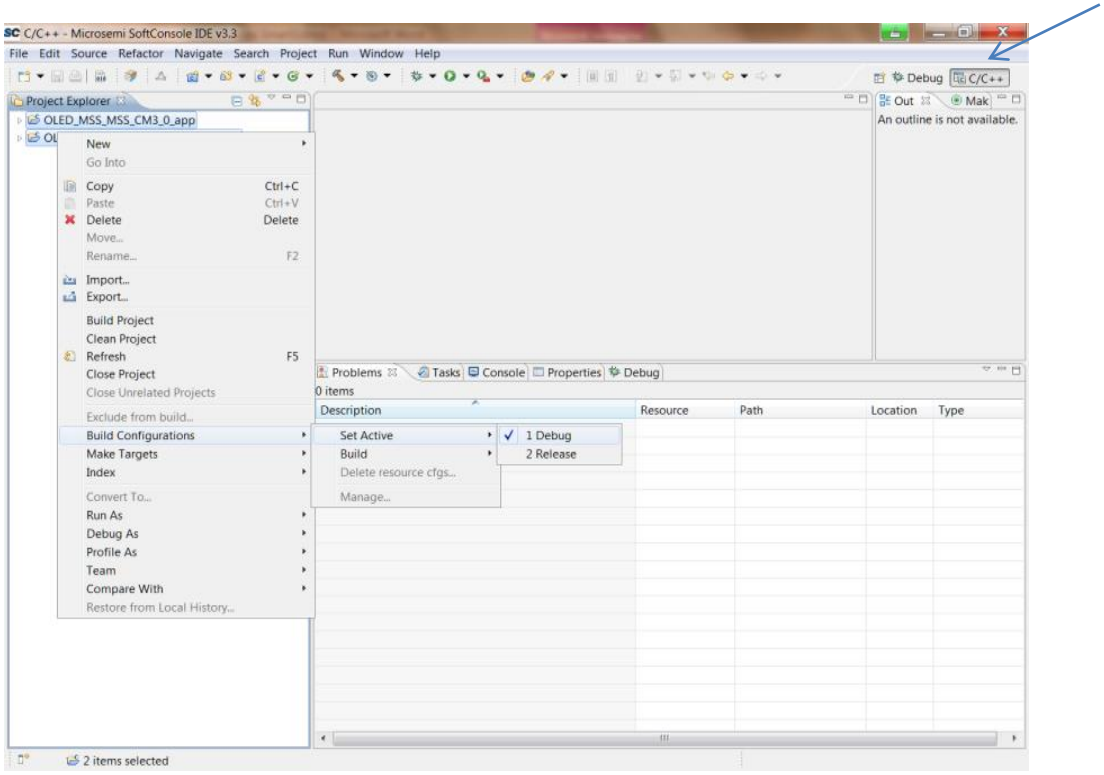

Figura 54

7. Nella finestra “Project Explorer” sarà visualizzata la gerarchia hardware “hw_platform” e quella sofware “app”. All’interno di quest’ultima è presente il codice sorgente dell’applicazione ed in particolare il file top “main.c” che potrà essere aperto cliccando su di esso due volte.

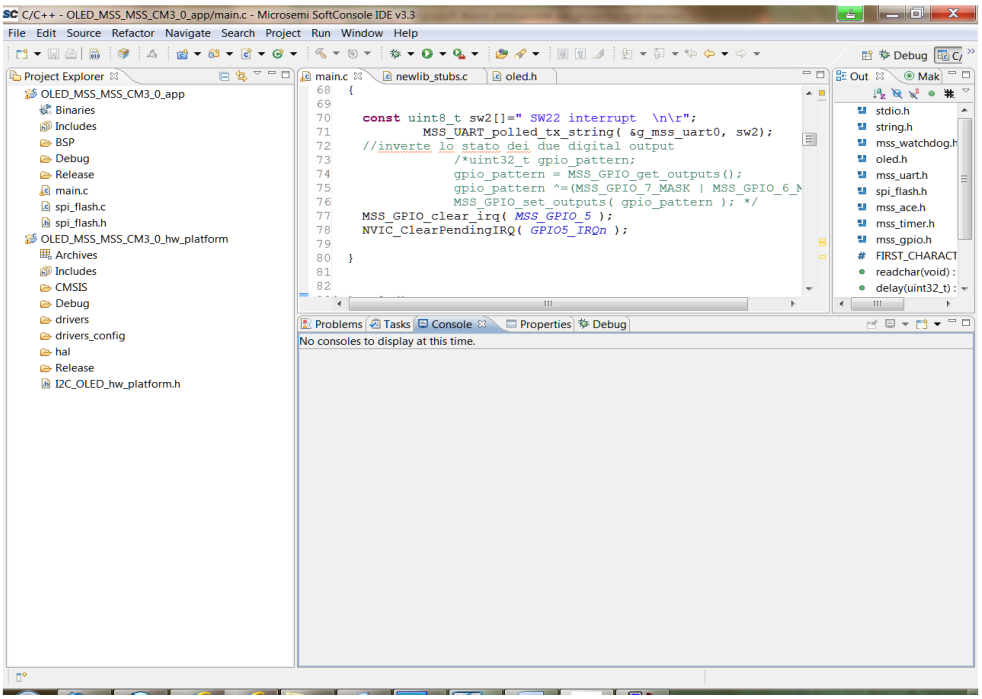

Figura 55

8. Lanciare dal Menu il comando “Project->Clean-> “ Clean all projects”.

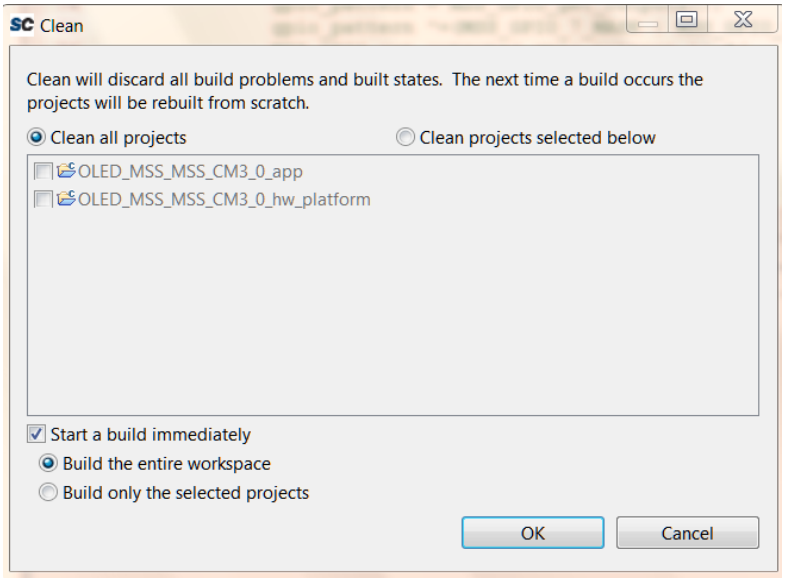

Figura 56

9. Lanciare il software Hyperterminal per la comunicazione su porta seriale. Per i sistemi Win7 si può utilizzare quello fornito nella cartella tools del DVD. Il programma “main.c” effettua una serie di test per ciascuna delle periferiche implementate nel chip SmartFusion. La connessione RS-232 è di tipo bidirezionale e dovrà essere impostata secondo i parametri indicati nelle figure seguenti:

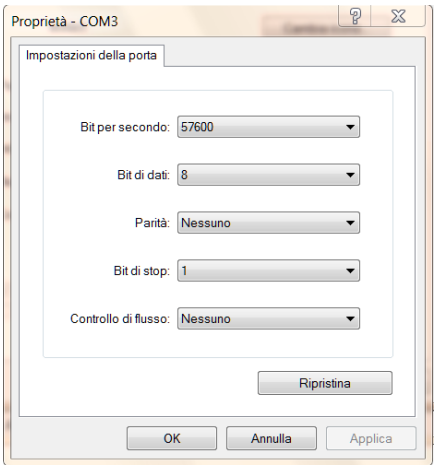

Figura 57

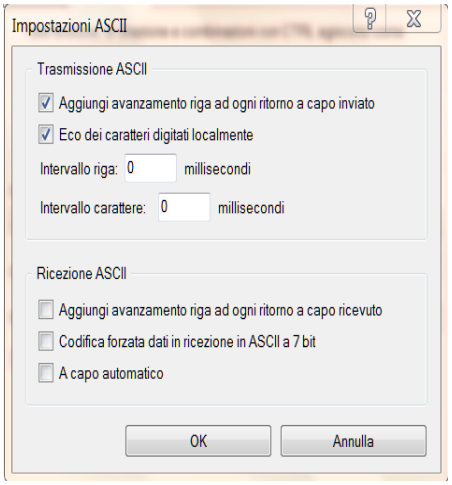

Figura 58

A tal proposito si osservi che è possibile utilizzare la connessione di Hyperterminal “testact” preimpostata presente nella cartella “tools” del DVD fornito.

10. Per avviare la fase di esecuzione/debug del codice, cliccare sul tasto “Debug Perspective” presente nella finestra principale, in alto a destra.

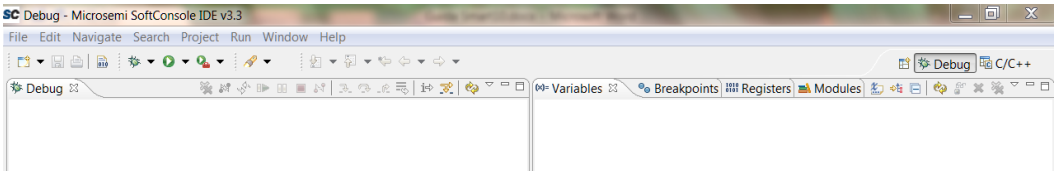

Figura 59

11. Lanciare dal Menu il comando “Run->Debug Configurations” selezionando il target “Cortex-M3 RAM”. Nel caso fosse già presente la configurazione “OLED_MSS_MSS_CM3_0_app Debug” indicata in figura 60, si proceda come indicato al punto 12.

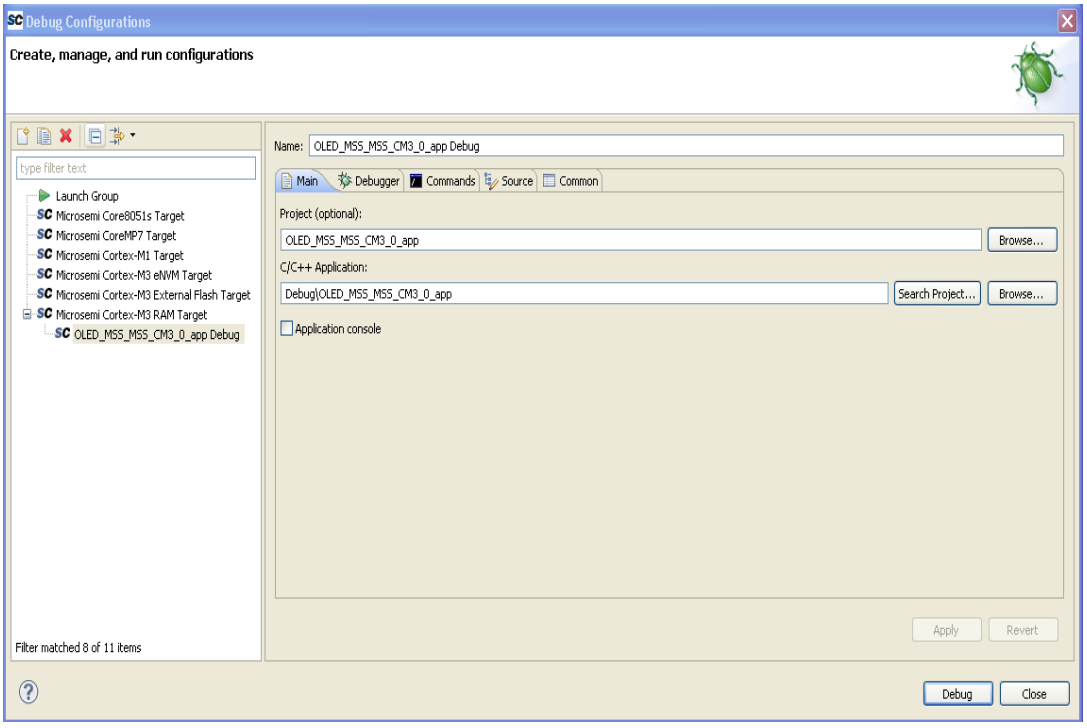

Figura 60

In caso contrario si effettui un doppio click sul target selezionato, compilando i due campi “Project” e “C/C++ Application”.

Per far ciò si prema il tasto “Browse” per la voce “Project” e selezionare quindi “OLED_MSS_MSS_CM3_0_app”.

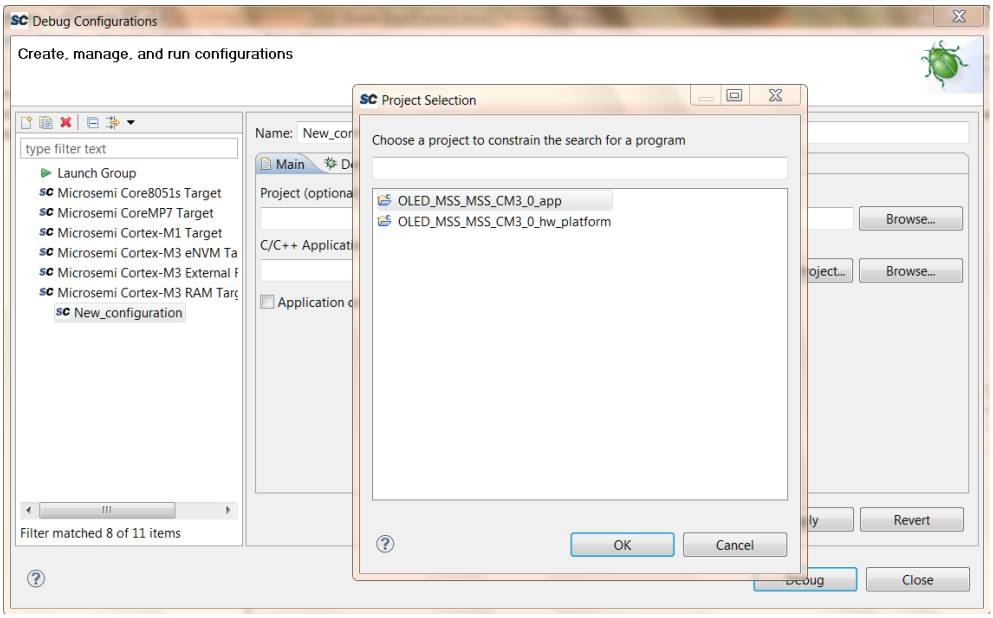

Figura 61

Premere il tasto “Search Project” per la voce “C/C++ Application” e selezionare “/OLED_MSS_MSS_CM3_0_app/Debug/OLED_MSS_MSS_CM3_0_app”.

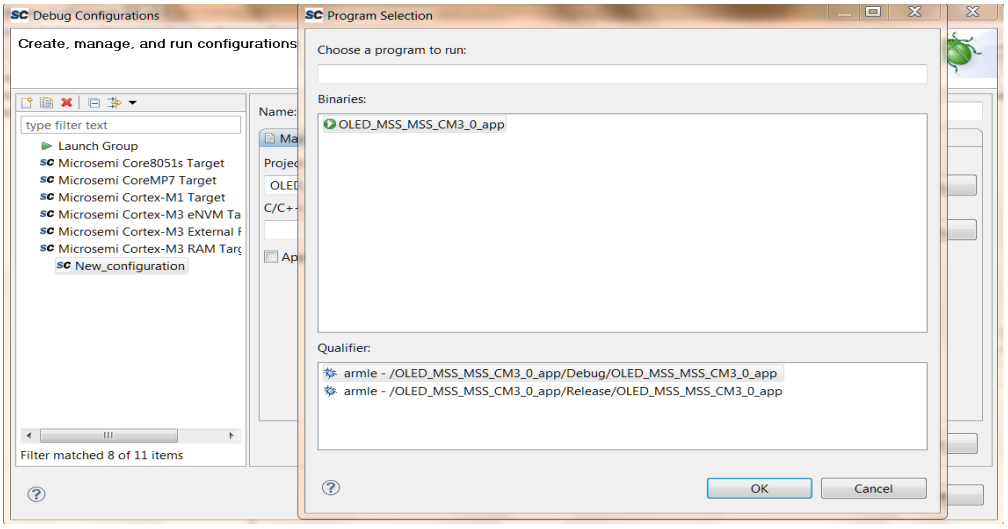

Figura 62

Si prema quindi “Apply”

12. Cliccare quindi su “Debug”; il programma si pone in pausa all’inizio della procedura “main”.

13. Cliccando dal Menu su “Run->Resume” si avvia l’esecuzione del codice e, inserendo gli opportuni “breakpoints”, un eventuale fase di debug. Cliccando su “Run->Terminate” si ottiene l’interruzione dell’esecuzione.

14. E’ possibile effettuare nuovamente l’esecuzione del codice di esempio (eventualmente modificato) ripetendo i passi 8 e 11-14.

2.3.3.1 NOTE SULL'ESECUZIONE DEL PROGRAMMA DEMO

In Appendice A è riportato il codice del programma "main.c". Ciascun test viene segnalato dalla visualizzazione di un messaggio sulla console di Hyperterminal. Come mostrato in figura 63 si effettuano in particolare i seguenti test:

1. Scrittura del display OLED con interfaccia I2C
2. Scrittura/lettura della memoria flash onboard con interfaccia SPI
3. Controllo LED0-LED3
4. Test push-buttons SW1 e SW2 con relative procedure di interrupt
5. Timer con interrupt periodico verso il processore
6. Conversione A/D ciclica della tensione proveniente dal potenziometro onboard e comparazione di tale valore con determinati valori di soglia.

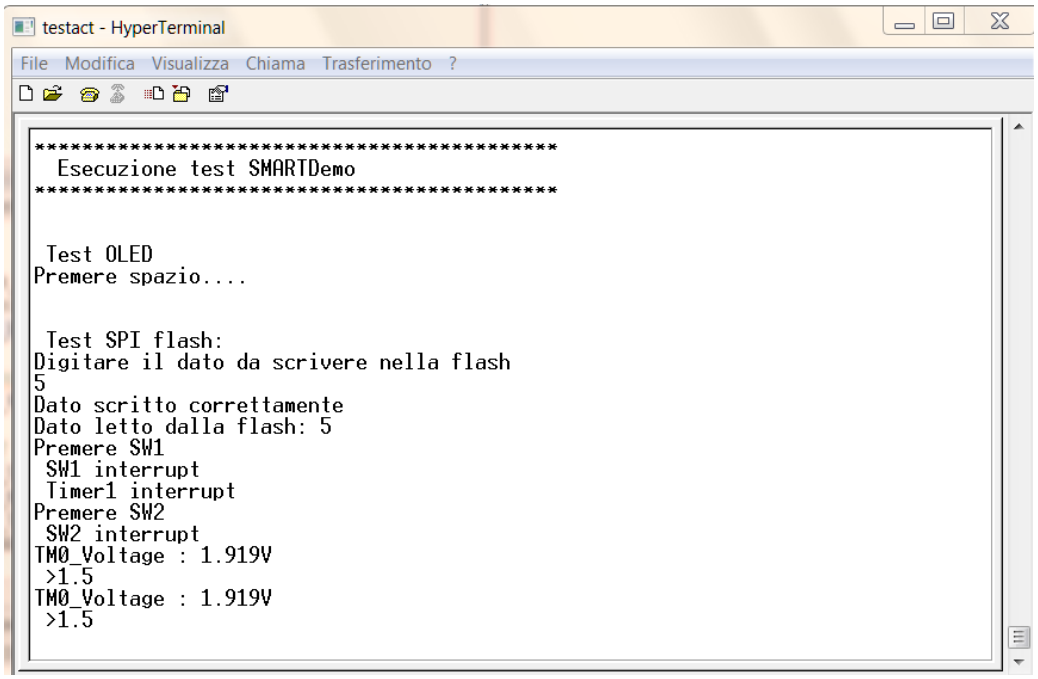

Figura 63

La documentazione relativa alle funzioni ed ai driver delle periferiche utilizzate è disponibile sul sito Actel[4]. Le istruzioni relative al secondo canale di

conversione analogico/digitale “MSS_VOL_0” ed alle due uscite digitali “DIGOUT0” e “DIGOUT1” sono inserite nel codice di esempio ma sono state commentate. Le corrispondenti periferiche possono essere utilizzate effettuando le opportune connessioni ai pin del connettore J1 per come indicato in tabella 1.

2.3.3.2 UTILIZZO DI PERIFERICHE I2C E SPI ESTERNE

La board fornita consente di collegare al chip SmartFusion delle periferiche esterne compatibili con i protocolli seriali I2C ed SPI. In particolare nel progetto SMARTDEMO sono stati abilitati i controller I2C_1 ed SPI_1 che sono connessi ai connettori della board per come specificato nelle tabelle 2 e 3.

SEGNALE	SMARTFUSION PIN	BOARD
SPI_1_DO/GPIO_24	T17	SPI_1_DO, J23 pin 1
SPI_1_DI/GPIO_25	V19	SPI_1_DI, J23 pin 2
SPI_1_CLK/GPIO_26	AA22	SPI_1_CLK, J23 pin 3
SPI_1_SS/GPIO_27	W21	SPI_1_SS, J23 pin 4

Tabella 2

SEGNALE	SMARTFUSION PIN	BOARD
I2C_1_SCL/GPIO_31	U20	SCL1, JP9 pin 2
I2C_1_SDA/GPIO_30	V22	SDA1, JP9 pin 1

Tabella 3

Come indicato di seguito, l’utilizzo di tali controller con delle periferiche esterne richiede l’impiego dei drivers MSS presenti nel sistema di sviluppo.
DRIVER MSS_I2C
Il driver MSS_I2C mette a disposizione un insieme di funzionalità [4] per la gestione del protocollo I2C da parte dello SmartFusion. Il driver è implementato nei due files “mss_i2c.c” ed “mss_i2c.h” presenti nella sottocartella“..OLED_MSS_MSS_CM3_0_hw_platform\drivers\mss_i2c”.
Esso consiste fondamentalmente in:

- Funzioni di inizializzazione e configurazione
  -MSS_I2C_init()
- Funzioni per la gestione di interrupt
  -MSS_I2C_register_write_handler()
- Funzioni di comunicazione I2C “master”
  -MSS_I2C_write()
  -MSS_I2C_read()
  -MSS_I2C_write_read()
  -MSS_I2C_wait_complete()
  -MSS_I2C_get_status()

Funzioni di comunicazione I2C “slave”

-MSS_I2C_set_slave_tx_buffer()
-MSS_I2C_set_slave_rx_buffer()
-MSS_I2C_set_slave_mem_offset_length()
-MSS_I2C_register_write_handler()
-MSS_I2C_enable_slave_rx()

In base alla documentazione specifica del circuito esterno che si intende collegare alla board (sensore, attuatore, display, ecc.) sarà possibile sviluppare un driver *ad hoc* utilizzando le funzioni sopra indicate. Ciò è stato fatto ad esempio per il driver del dispositivo OLED presente nel progetto SMARTDEMO, i cui sorgenti sono disponibili nella sottocartella “..OLED_MSS_MSS_CM3_0_app\BSP\oled_driver”.
Per ulteriori approfondimenti si faccia riferimento alla “MSS_I2C User Driver Guide” presente nella cartella “doc” del DVD allegato.

DRIVER MSS_SPI

Il driver MSS_SPI fornisce un insieme di funzionalità [4] relative al protocollo SPI. Il driver è implementato nei due files “mss_spi.c” ed “mss_spi.h” presenti nella sottocartella“..OLED_MSS_MSS_CM3_0_hw_platform\drivers\mss_spi”.
Le principali funzioni messe a disposizione sono indicate di seguito.

- Funzioni di inizializzazione e configurazione
  -MSS_SPI_init()
  -MSS_SPI_configure_master_mode()
  -MSS_SPI_configure_slave_mode()
- Funzioni di comunicazione SPI “master”
  -MSS_SPI_set_slave_select()

-MSS_SPI_transfer_frame()
-MSS_SPI_clear_slave_select()
-MSS_SPI_set_slave_select()
-MSS_SPI_clear_slave_select()
-MSS_SPI_transfer_block()

- Funzioni di comunicazione SPI “slave”
  -MSS_SPI_set_slave_tx_frame()
  -MSS_SPI_set_frame_rx_handler()
  -MSS_SPI_set_slave_block_buffers()
  -MSS_SPI_set_cmd_handler()
  -MSS_SPI_set_cmd_response()
- Funzioni per la gestione del DMA su SPI
  -MSS_SPI_disable()
  -MSS_SPI_set_transfer_byte_count()
  -MSS_SPI_enable()
  -MSS_SPI_tx_done()

In maniera analoga a quanto detto per il driver I2C, anche per la periferica SPI di interesse indicate sarà possibile sviluppare un driver specifico utilizzando le funzioni sopra indicate. Ciò è stato fatto ad esempio per il driver della memoria flash esterna presente nel progetto SMARTDEMO, i cui sorgenti sono disponibili nella sottocartella “..OLED_MSS_MSS_CM3_0_app”.

Per ulteriori approfondimenti si faccia riferimento alla “MSS_SPI User Driver Guide” presente nella cartella “doc” del DVD allegato.

ATTIVITÀ ELEMENTARE 2.3.4

IMPLEMENTAZIONE DI UNA RELEASE IMAGE SU MEMORIA eNVM

Dopo aver effettuato la fase di debug e messo a punto il codice dell'applicazione (ad esempio modificando il programma demo fornito), sarà possibile costruirne l'immagine in modalità release[5] e caricarla nella flash eNVM dello SmartFusion. In tal modo l'applicazione si avvierà ad ogni accensione/reset della board.

I passi da eseguire sono indicati di seguito.

1. In SoftConsole, impostare la "C/C++ perspective" (tasto in alto a destra) e selezionare i progetti "hw_platform" e "app" nella finestra "Project Explorer". Premere col tasto destro e cliccare quindi su "Build Configurations->Set Active->Release".

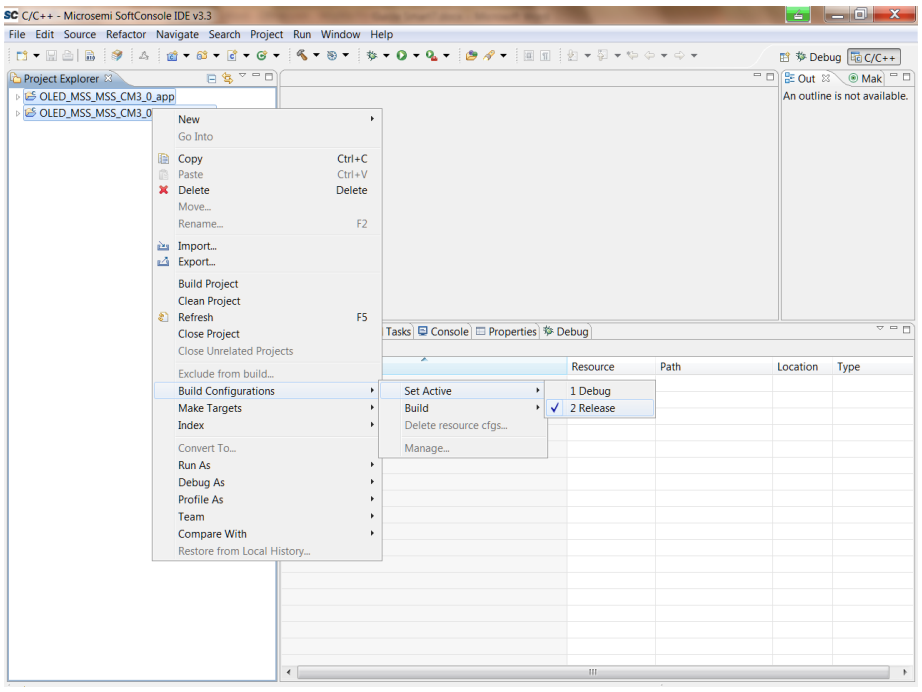

Figura 64

2. Cliccare sul menu “Project->Build all” in modo da generare il file “OLED_MSS_MSS_CM3_0_app.hex” nella sottocartella “…\SoftConsole\ OLED_MSS _MSS _CM3_0\ OLED_MSS _MSS_MSS_CM3_0_app\Release”
3. Avviare il programma Libero SoC v10.1 cliccando sul file di progetto “SMARTDEMO.prjx” inserito nella cartella “C:\actprj\kit2\smartdemo”
4. Apparirà un messaggio che avvisa della necessità di procedere all’aggiornamento dei core usati nel progetto. Disponendo di una connessione ad internet attiva, il programma effettuerà l’aggiornamento dei core in maniera automatica.
5. Attivare il modulo “SMARTDEMO_MSS” cliccando all’interno della finestra ‘Design Hierarchy’. Si aprirà la finestra relativa al “MSS Configurator” come indicato in figura 65.

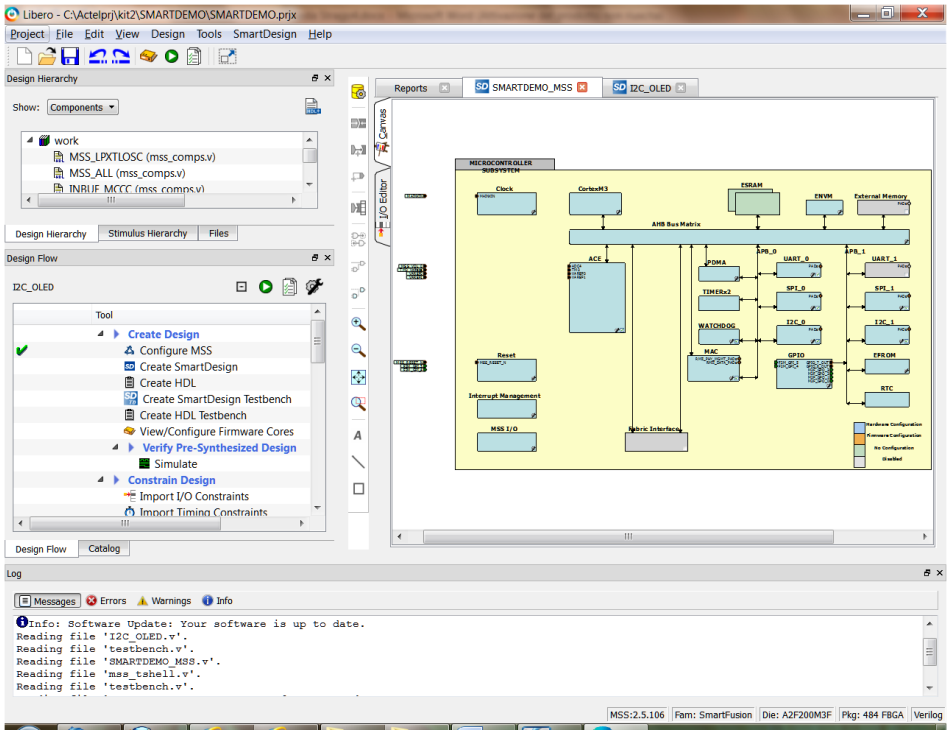

Figura 65

6. Fare doppio clic sul modulo eNVM presente nel MSS Configurator (in alto a destra) per aggiungere il codice dell'applicazione precedentemente generata all'eNVM. La finestra MSS_ENVM_0 configuratore viene visualizzata per come mostrato in figura 66:

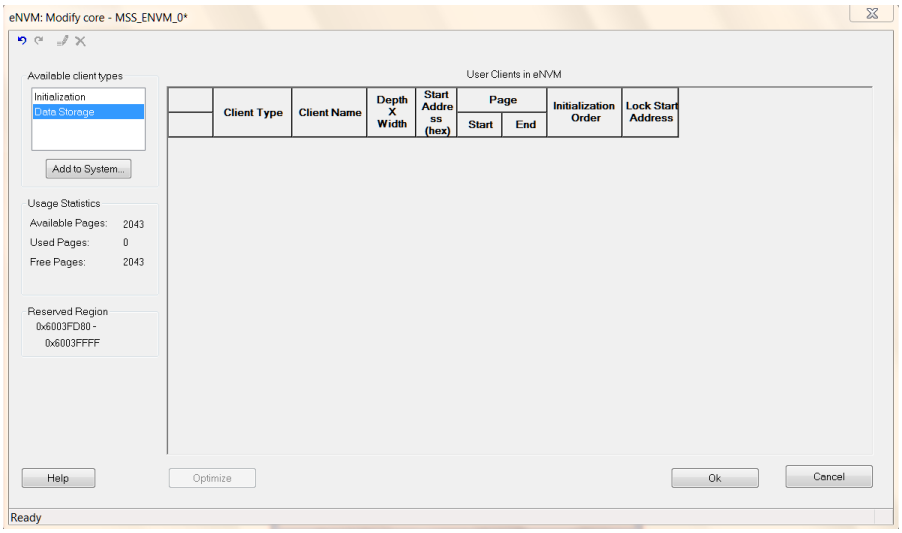

Figura 66

7. Selezionare “Data Storage” nella scheda “Tipi di client” e fare clic su “Aggiungi al sistema”. Inserire i dati per come indicato in figura 67.

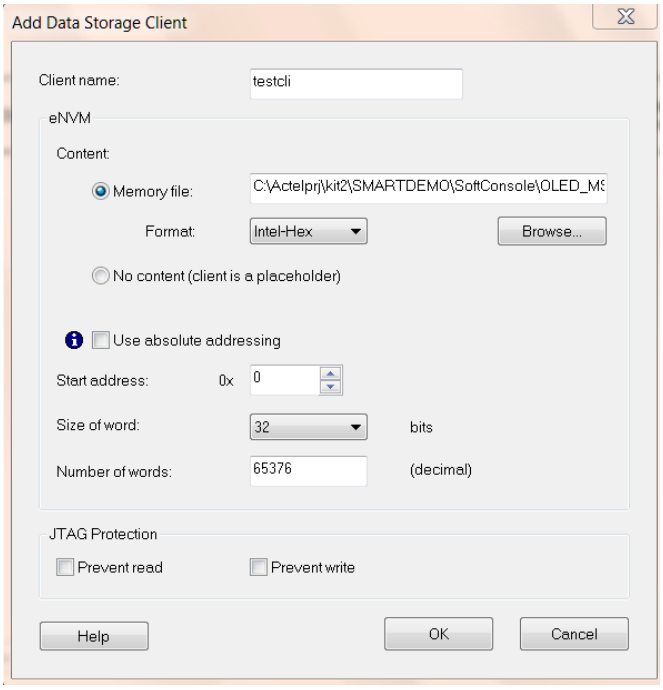

Figura 67

8. In particolare Cliccando su “Browse” sarà possibile specificare il file “OLED_MSS_MSS_CM3_0_app.hex” precedentemente generato (passo 2).
9. Settare il flag “Lock Start Address” come indicato in figura 68 e quindi chiudere la finestra relativa all’eNVM.

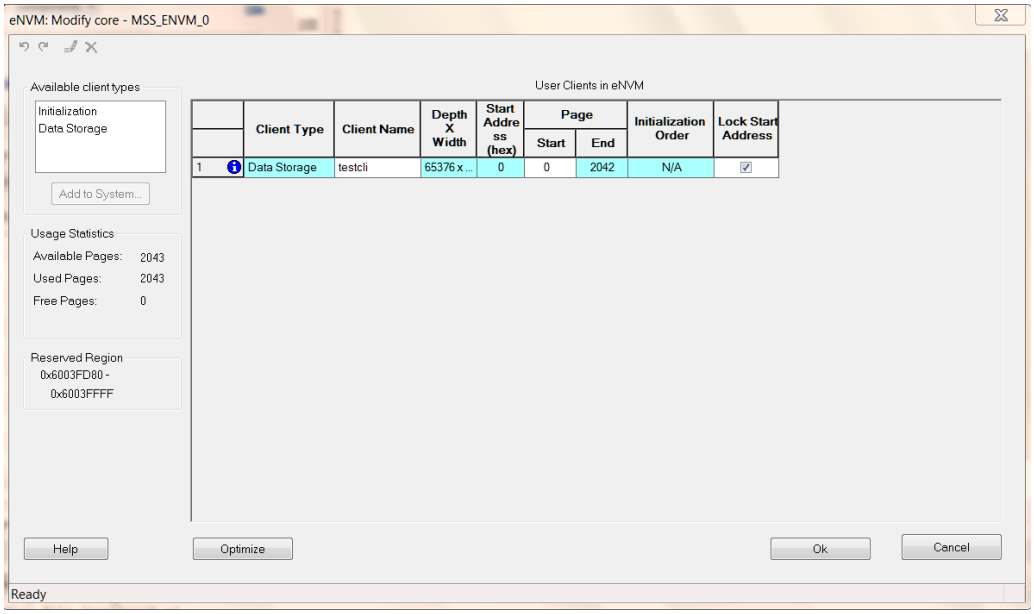

Figura 68

10. Rigenerare il componente SMARTDEMO_MSS cliccando “Smartdesign-> Generate Component”. Cliccare poi su “Generate Programming Data->Update e Run” come indicato in figura 69.

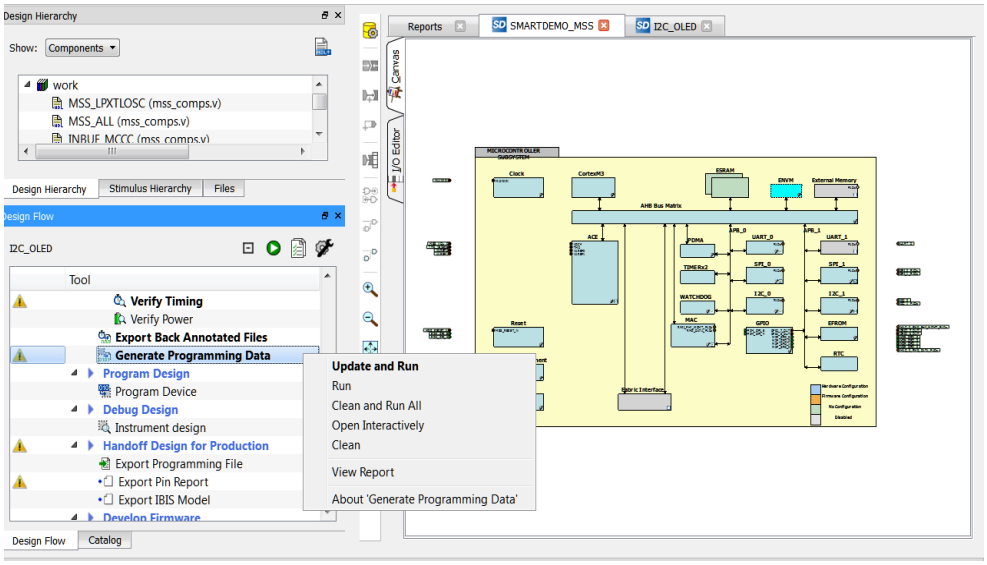

Figura 69

11. A questo punto si può avviare il tool di programmazione FlashPro cliccando su “Program Device-> Open Interactively” come illustrato in figura 70.

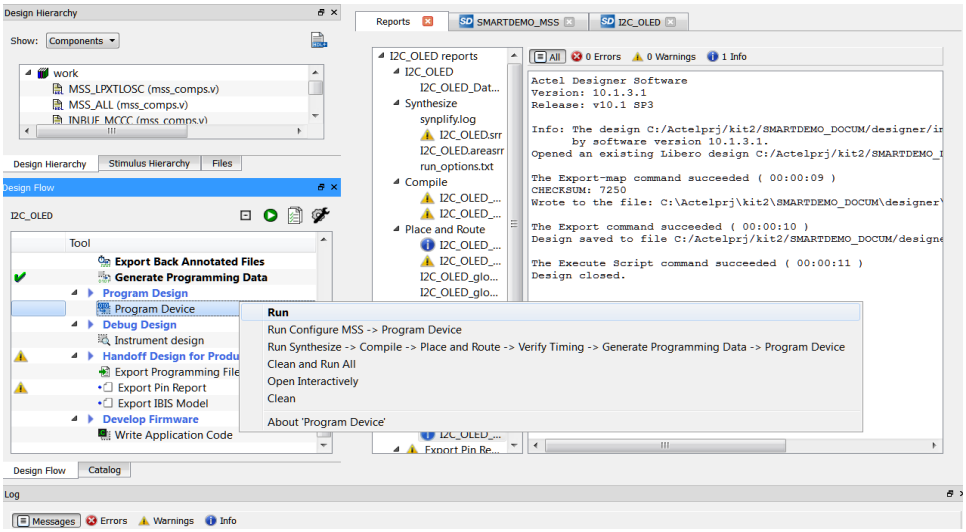

Figura 70

12. Si apre la finestra del tool FlashPro. Cliccare su “File-Export->Single Programming File”.

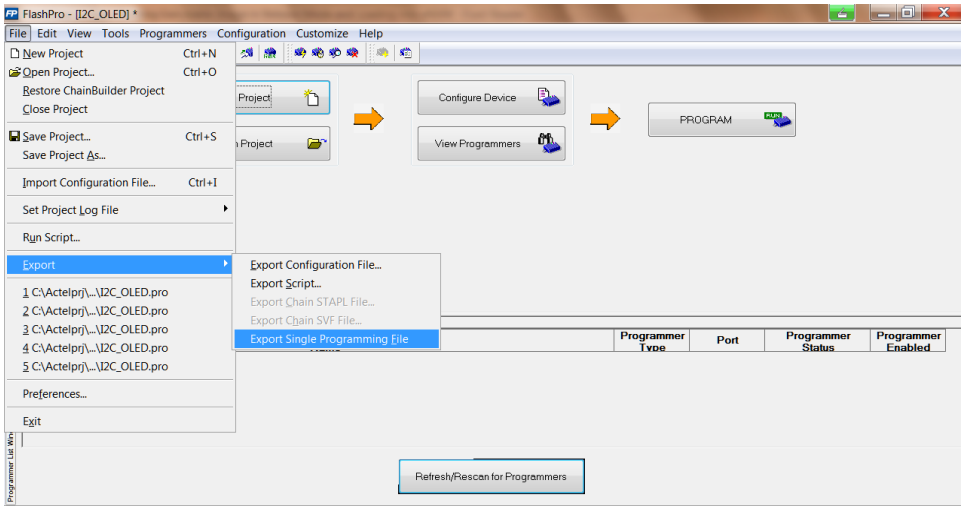

Figura 71

13. Selezionare il solo formato di output “STAPL file” in modo da generare un file di programmazione con estensione “stp”.

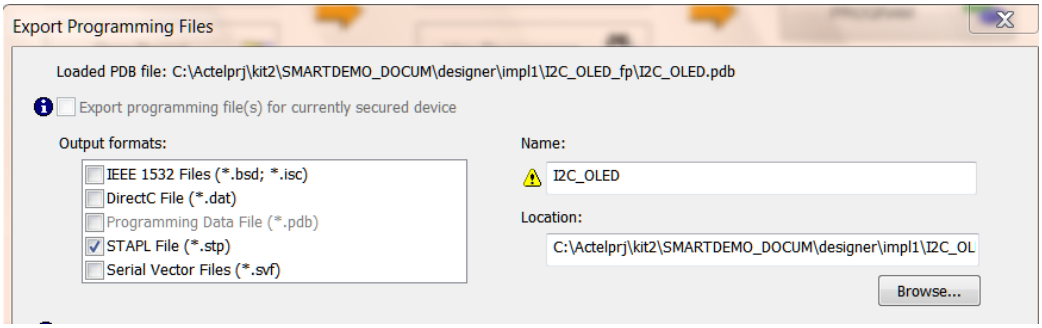

Figura 72

14. Nella finestra principale di FlashPro, cliccare su “Configure Device” e quindi su “Browse” per caricare il file “stp” sopra generato. Tale file è disponibile nella directory “ /designer/impl1/I2C_OLED.fp”.

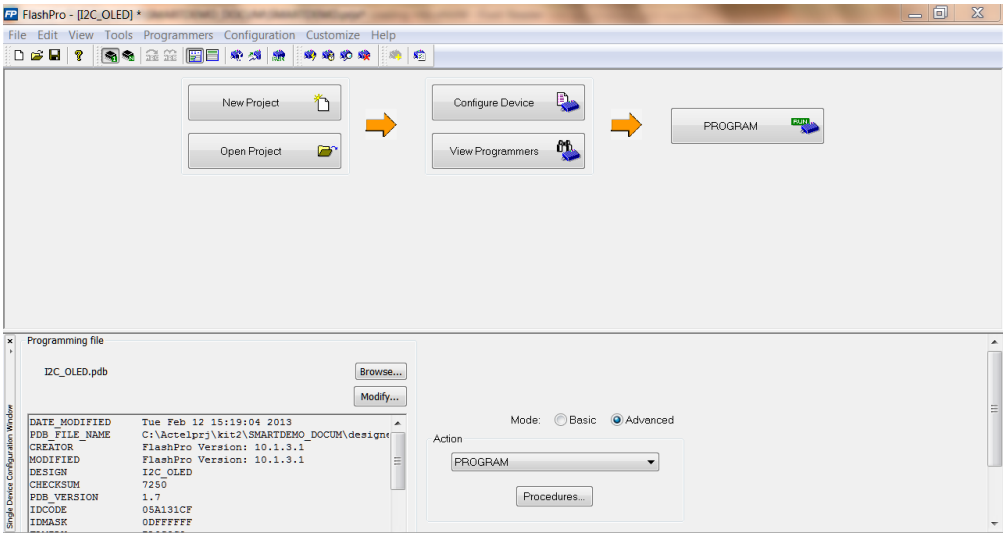

Figura 73

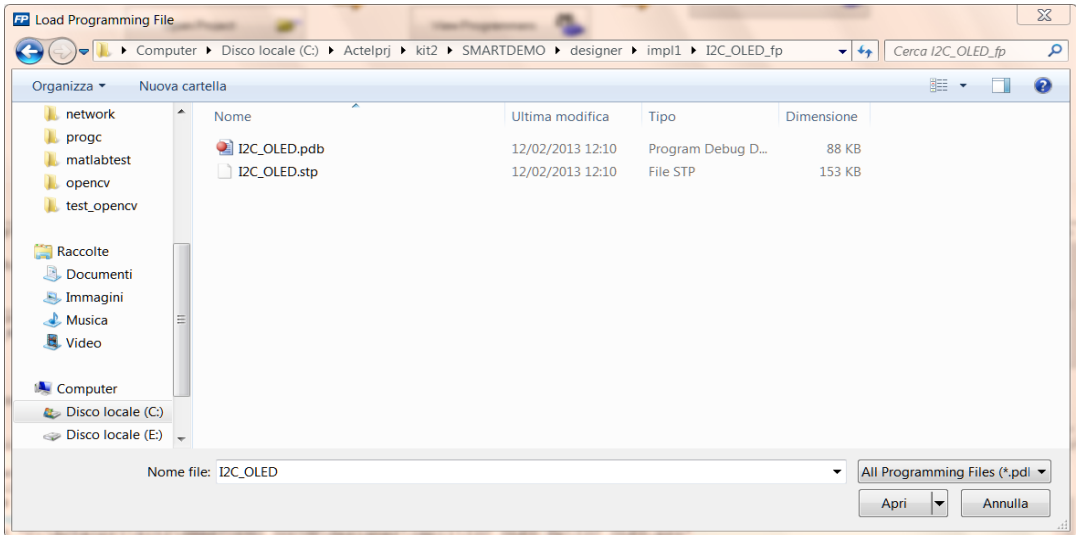

Figura 74

15. Cliccare infine sul tasto “PROGRAM” come indicato in figura 75 in modo da procedere alla programmazione del chip SmartFusion.

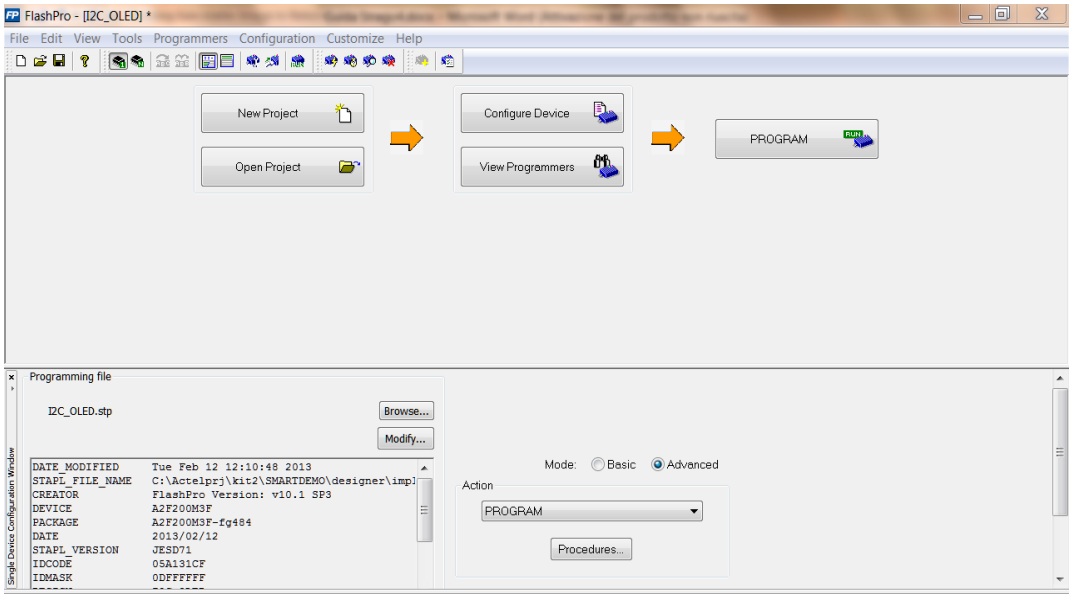

Figura 75

Rimuovere i cavi USB dalla board e riconnettere successivamente il solo cavo di alimentazione. Ad ogni successivo startup/reset del chip, il programma caricato nella memoria eNVM sarà ravviato automaticamente.

Riferimenti

[1]. https://www.actel.com/products/smartfusion/default.asp
[2]. https://www.actel.com/products/hardware/devkits_boards/smartfusion_eval.asp
[3]. http://www.actel.com/documents/A2F_EVAL_KIT_UG.pdf
[4]. http://www.actel.com/products/software/smartdesign/docs_mss.aspx
[5]. http://www.actel.com/documents/SmartFusion_Release_Built_Tutorial.pdf
[6]. http://www.actel.com/documents/SoftConsole_UG.pdf

APPENDICE

```c
#include <stdio.h>
#include <string.h>
#include "mss_watchdog.h"
#include "oled.h"
#include "mss_uart.h"
#include "spi_flash.h"
#include "mss_ace.h"
#include "mss_timer.h"
#include "mss_gpio.h"

#define FIRST_CHARACTER 0

/* Function to read the input character */
char readchar (void)
{
    uint8_t rx_size = 0;
    unsigned char rx_char;
    do
    {
        rx_size = MSS_UART_get_rx( &g_mss_uart0, &rx_char, sizeof(rx_char) );
    }while ( rx_size == 0);

    return(rx_char);
}

/* Function for delay */
void delay ( volatile uint32_t n)
{
    while(n!=0)
    {
        n--;
    }
}
```

```c
/* Timer Interrupt Handler */
void Timer1_IRQHandler( void )
{
   const uint8_t v10[]=" Timer1 interrupt \n\r";
    MSS_UART_polled_tx_string( &g_mss_uart0, v10);

/* inverti lo stato dei led LED0-LED3 */
    MSS_TIM1_clear_irq();
    uint32_t gpio_pattern;
    gpio_pattern = MSS_GPIO_get_outputs();
    gpio_pattern ^= (MSS_GPIO_3_MASK | MSS_GPIO_2_MASK | MSS_GPIO_1_MASK | MSS_GPIO_0_MASK);
    MSS_GPIO_set_outputs( gpio_pattern );
    /*This delay is requied to distinguish the ON and OFF of LEDs */
    delay(100000);
}


/* GPIO 4 Interrupt Handler */
void GPIO4_IRQHandler( void )
  {
    const uint8_t sw1[]=" SW1 interrupt \n\r";
    MSS_UART_polled_tx_string( &g_mss_uart0, sw1);
    MSS_GPIO_clear_irq( MSS_GPIO_4 );
    NVIC_ClearPendingIRQ( GPIO4_IRQn );

  }

/* GPIO 5 Interrupt Handler */
void GPIO5_IRQHandler( void )
  {

    const uint8_t sw2[]=" SW22 interrupt  \n\r";
            MSS_UART_polled_tx_string( &g_mss_uart0, sw2);
    //inverte lo stato dei due digital output
                /*uint32_t gpio_pattern;
                gpio_pattern = MSS_GPIO_get_outputs();
                gpio_pattern ^=(MSS_GPIO_7_MASK | MSS_GPIO_6_MASK);
                MSS_GPIO_set_outputs( gpio_pattern ); */
    MSS_GPIO_clear_irq( MSS_GPIO_5 );
    NVIC_ClearPendingIRQ( GPIO5_IRQn );

  }
```

```c
int main()
{
        uint32_t tx_size;
        size_t rx_size;
        uint8_t rx_buff[1]={0};
        /* UART Initialization and Configuration */
               MSS_UART_init
               (&g_mss_uart0,
               MSS_UART_57600_BAUD,
              MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT);

    /* Init OLED */
        const uint8_t pattern[]  = "\n\r\n\r Test OLED\n\r";
        char *string1=" TEST";
        char *string2=" UNICAL ";
        char *string3=" DIMES ";
        char *string4=" SMARTFUSION";
        char *string5=" FPGA ";
        uint8_t i;
        struct oled_data write_data;

        write_data.line1 = FIRST_LINE;
        write_data.char_offset1 = FIRST_CHARACTER;
        write_data.string1 = string1;

        write_data.line2 = SECOND_LINE;
        write_data.char_offset2 = FIRST_CHARACTER;
        write_data.string2 = string2;

        write_data.line3 = THIRD_LINE;
        write_data.char_offset3 = FIRST_CHARACTER;
        write_data.string3 = string3;

        write_data.line4 = FOURTH_LINE;
        write_data.char_offset4 = FIRST_CHARACTER;
        write_data.string4 = string4;

        write_data.line5 = FIFTH_LINE;
        write_data.char_offset5 = FIRST_CHARACTER;
        write_data.string5 = string5;
        write_data.contrast_val = 0xFF;

      /* Watchdog Disabling function*/
         MSS_WD_disable();
        OLED_init();
```

```
    /* Init SPI     */
        uint8_t init_write_text[] = "Digitare il dato da scrivere nella flash\n\r";
        uint8_t write_success_text[] = "Dato scritto correttamente \n\r";
        uint8_t read_text[] = "Dato letto dalla flash: ";
        const uint8_t pattern_spi[]  = "\n\r\n\r Test SPI flash: \n\r";
        uint8_t tx_buff_spi_flash[4096];
        uint8_t rx_buff_spi_flash[4096];
        uint32_t address=0;uint32_t size  = 0;
        spi_flash_init();
      /*Init ACE*/
        ACE_init( );
        const uint8_t * channel_name;
        channel_name = ACE_get_channel_name( TM0_Voltage );
        const uint8_t * channel_name2;
        channel_name2 = ACE_get_channel_name( MSS_VOL_0);
        const uint8_t v10[]=" >1.0 \n\r";
        const uint8_t    v15[]=" >1.5 \n\r";
        const uint8_t    v20[]=" >2.0 \n\r";
        const uint8_t    v25[]=" >2.5 \n\r";
     /* Init timer1 */
        MSS_TIM1_init( MSS_TIMER_PERIODIC_MODE );
        MSS_TIM1_load_immediate( g_FrequencyPCLK0*20 );
        MSS_TIM1_start();
        MSS_TIM1_enable_irq();

       /* Init LEDS   */
        MSS_GPIO_init();
        MSS_GPIO_config(MSS_GPIO_0 , MSS_GPIO_OUTPUT_MODE );
        MSS_GPIO_config(MSS_GPIO_1 , MSS_GPIO_OUTPUT_MODE );
        MSS_GPIO_config(MSS_GPIO_2 , MSS_GPIO_OUTPUT_MODE );
        MSS_GPIO_config(MSS_GPIO_3 , MSS_GPIO_OUTPUT_MODE );

    /* Init GPIO */
        MSS_GPIO_config(MSS_GPIO_6 , MSS_GPIO_OUTPUT_MODE );
        MSS_GPIO_config(MSS_GPIO_7 , MSS_GPIO_OUTPUT_MODE );
       // MSS_GPIO_config(MSS_GPIO_8 , MSS_GPIO_INPUT_MODE );
       // MSS_GPIO_config(MSS_GPIO_9 , MSS_GPIO_INPUT_MODE );
        MSS_GPIO_set_outputs( 0xAAAAAAAA );

     /* Init SW1 e SW2   */
        NVIC_EnableIRQ(GPIO4_IRQn);
        MSS_GPIO_config( MSS_GPIO_4, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_POSITIVE )
        MSS_GPIO_enable_irq( MSS_GPIO_4 );
        NVIC_EnableIRQ(GPIO5_IRQn);
        MSS_GPIO_config( MSS_GPIO_5, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_NEGATIVE )
        MSS_GPIO_enable_irq( MSS_GPIO_5 );
```

```
////////////////////////////////////////// Esecuzione test
            delay(10000000);
            const uint8_t welcome1[] = "******************************************\n\r";
            const uint8_t welcome2[] = "  Esecuzione test SMARTDemo  \n\r";
            const uint8_t welcome3[] = "******************************************\n\r";
             MSS_UART_polled_tx_string(&g_mss_uart0, welcome1);
             MSS_UART_polled_tx_string(&g_mss_uart0, welcome2);
             MSS_UART_polled_tx_string(&g_mss_uart0, welcome3);

        ////////OLED
                OLED_write_data(&write_data,FIRST_TWO_LINES);

                MSS_UART_polled_tx_string(&g_mss_uart0, pattern);

                    /* Vertical scrolling */
                                write_data.on_off = 0x0;
                                write_data.vertical_on_off = 0x01;
                                for(i=0; i<3; i++)
                                {
                                                        write_data.vert_scroll_offset = 0x1F;
                                                        OLED_vertical_scroll(&write_data);
                                                        delay(8000000);

                                                        write_data.vert_scroll_offset = 0x18;
                                                        OLED_vertical_scroll(&write_data);
                                                        delay(8000000);

                                                        write_data.vert_scroll_offset = 0x10;
                                                        OLED_vertical_scroll(&write_data);
                                                        delay(8000000);

                                                        write_data.vert_scroll_offset = 0x08;
                                                        OLED_vertical_scroll(&write_data);
                                                        delay(8000000);
                                                        write_data.vert_scroll_offset = 0x00;
                                                    OLED_vertical_scroll(&write_data);
                  delay(6000000);
                                }
            uint8_t write_invio[] = "Premere spazio....\n\r";

     MSS_UART_polled_tx( &g_mss_uart0,write_invio , sizeof(write_invio) );
                  while(*rx_buff!= ' ')
                  {
                      rx_size = MSS_UART_get_rx ( &g_mss_uart0, rx_buff, sizeof(rx_buff) );

                 }
```

```
////// spi flash

          MSS_UART_polled_tx_string(&g_mss_uart0, pattern_spi);
        /* Requesting for write data */
        MSS_UART_polled_tx( &g_mss_uart0,init_write_text , sizeof(init_write_text) );
        while(*rx_buff!= '\n')
        {
            rx_size = MSS_UART_get_rx ( &g_mss_uart0, rx_buff, sizeof(rx_buff) );
            if( rx_size > 0 )
            {
                tx_buff_spi_flash[size] = *rx_buff;
                size++;
            }
        }
    /* alliging to 32 bit */
    tx_size = size;
    if ((size%4) != 0)
      {
            size += (4-(size%4));
      }
    /* unprotect the flash sectors*/
      spi_flash_control_hw(SPI_FLASH_SECTOR_UNPROTECT,0,NULL);

      /* Erase first 4Kbye block */
      spi_flash_control_hw(SPI_FLASH_4KBLOCK_ERASE,0,NULL);

      /* Write string data into SPI Flash */
      spi_flash_write(address , tx_buff_spi_flash, size);

      /* Display the write status */
      MSS_UART_polled_tx( &g_mss_uart0,write_success_text , sizeof(write_success_text));

      /* Read string data from SPI Flash */
      spi_flash_read(address ,  rx_buff_spi_flash, size);

      /* Display of read back data */
      MSS_UART_polled_tx( &g_mss_uart0,read_text , sizeof(read_text));
      MSS_UART_polled_tx( &g_mss_uart0,rx_buff_spi_flash , tx_size);
      *rx_buff = NULL;
      size = 0;
      rx_size= 0;
      tx_size= 0;
////////////////////////////////////////////////////////////////////////////////
```

```
// SW1 e SW2
    MSS_UART_polled_tx_string(&g_mss_uart0, "Premere SW1 \n\r");
    delay(50000000);
    MSS_UART_polled_tx_string(&g_mss_uart0, "Premere SW2 \n\r");
    delay(50000000);

////////////////////////////////////////////////////////////////////////////////
    while(1)
    {

    //////////converti, compara e visualizza tensione potenziometro

    uint8_t display_buffer[32];
    uint16_t adc_result;
    int32_t adc_value_mv;
    adc_result = ACE_get_ppe_sample( TM0_Voltage );
    adc_value_mv = ACE_convert_to_mV( TM0_Voltage, adc_result );
    if ( adc_value_mv < 0 )
    {
    snprintf( (char *)display_buffer, sizeof(display_buffer),
    "%s : -%.3fV\r", channel_name, ((float)(-adc_value_mv) / (float)(1000)));
    }
    else
    {
    snprintf( (char *)display_buffer, sizeof(display_buffer),
    "%s : %.3fV\r", channel_name, ((float)(adc_value_mv) / (float)(1000)));
    }
    MSS_UART_polled_tx_string( &g_mss_uart0, display_buffer );
    MSS_UART_polled_tx_string( &g_mss_uart0, "\n\r");


            int32_t flag_status_2p5v = ACE_get_flag_status(TM0_Voltage_over_2p5v);
            int32_t flag_status_2p0v = ACE_get_flag_status(TM0_Voltage_over_2p0v);
            int32_t flag_status_1p5v = ACE_get_flag_status(TM0_Voltage_over_1p5v);
            int32_t flag_status_1p0v = ACE_get_flag_status(TM0_Voltage_over_1p0v);
```

```
            if ( flag_status_2p5v == FLAG_ASSERTED )
            MSS_UART_polled_tx_string( &g_mss_uart0, v25);
            else
            if ( flag_status_2p0v == FLAG_ASSERTED )
            MSS_UART_polled_tx_string( &g_mss_uart0, v20 );
            else
            if ( flag_status_1p5v == FLAG_ASSERTED )
            MSS_UART_polled_tx_string( &g_mss_uart0, v15 );
            else
            if ( flag_status_1p0v == FLAG_ASSERTED )
            MSS_UART_polled_tx_string( &g_mss_uart0, v10);


        delay(20000000);

/*
/// converti tensione tensione adc4 u12

adc_result = ACE_get_ppe_sample( MSS_VOL_0 );
    adc_value_mv = ACE_convert_to_mV( MSS_VOL_0, adc_result );
    if ( adc_value_mv < 0 )
    {
    snprintf( (char *)display_buffer, sizeof(display_buffer),
    "%s : -%.3fV\r", channel_name2, ((float)(-adc_value_mv) / (float)(1000)));
    }
    else
    {
    snprintf( (char *)display_buffer, sizeof(display_buffer),
    "%s : %.3fV\r", channel_name2, ((float)(adc_value_mv) / (float)(1000)));
    }
    MSS_UART_polled_tx_string( &g_mss_uart0, display_buffer );
    MSS_UART_polled_tx_string( &g_mss_uart0, "\n\r");
*/

//////////////////////////////////////////////////////////////////////



  }   //fine ciclo infinito

    return 0;
}
```