Università Politecnica delle Marche

Facoltà di Ingegneria

Laurea Specialistica in Ingegneria Elettronica

ARCHITETTURE, SVILUPPO, IMPLEMENTAZIONE E VERIFICA DI ALGORITMI DI VISIONE STEREOSCOPICA PER SISTEMI EMBEDDED.

Relatore:
Prof. Aldo Franco DRAGONI

Candidato:
Pierpaolo SCORRANO

Correlatore:
Dr. Paolo PAGANO

A.A. 2008-2009

Ai miei genitori

Il Serpente che solleva la testa si diverte nelle brume,
ll Drago volante cavalca le nubi.
Ma quando le nuvole se ne vanno e le brume si dissolvono,
essi non sono differenti dai vermi della terra.

Antico proverbio cinese

Indice

1 Cenni di teoria di visione stereoscopica **2**
1.1 Telecamera pinhole 2
1.2 Triangolazione con due telecamere 4
1.3 Geometria epipolare 5
1.3.1 Matrice essenziale E 6
1.3.2 Matrice fondamentale F 7
1.3.3 Matrice intrinseca della telecamera 8
1.3.4 Rettificazione epipolare 8
1.4 Algoritmo di stereovisione 9
1.4.1 Depth Discontinuities by Pixel to Pixel Stereo 10

2 Ambiente di Sviluppo **17**
2.1 Piattaforma Hardware 17
2.1.1 FLEX embedded board 17
2.1.2 Microcontrollore dsPIC33FJ256MC710 20
2.2 Camera CMOS 24
2.2.1 Driver per la Camera CMOS HV7131GP 24
2.3 Piattaforma Software 26
2.3.1 Supporto Microchip 27
2.3.2 *ERIKA Enterprise* 27
2.3.3 μWireless 31
2.4 Strumenti di debugging 32
2.4.1 DAQ 32
2.4.2 Chipcon CC2420 Packet Sniffer 34

3 Architettura Hardware **36**
3.1 Sistema di stereovisione 36
3.1.1 Camera Board 36
3.1.2 Camera Control 37
3.2 Stereo Coordinator 38

4 Implementazione **39**
4.1 Porting dell'algoritmo di stereovisione su dsPIC 39
4.1.1 Eliminazione dell'allocazione dinamica della memoria 39
4.1.2 Eliminazione del calcolo delle discontinuità di profondità . . . 40
4.1.3 Riduzione delle strutture allocate in memoria 40
4.2 Protocollo per la trasmissione wireless 42
4.2.1 Meccanismo di gestione dei pacchetti tra coordinator e device 42
4.3 Postprocess: elaborazione della disparità su PC 43

5 Risultati sperimentali **45**
5.1 Introduzione 45
5.2 Qualità della disparità elaborata embedded 45
5.3 Velocità di trasmissione 47

Conclusioni **53**

A Gestione del protocollo wireless **54**
A.1 Task lato coordinator 54
A.2 Task lato device 56

B Schematici del sistema di Stereovisione **58**
B.1 Schematico Camera Board 58
B.2 Schematico Camera Control 58

Glossario **61**

Bibliografia **62**

Ringraziamenti **64**

Elenco delle figure

1.1 Proiezione di un oggetto sul piano immagine [1] 3
1.2 Proiezione del punto P sul piano immagine [2] 3
1.3 Proiezione del punto P su due piani immagine allineati 4
1.4 Elementi fondamentali della geometria epipolare 6
1.5 Piani immagine non allineati lungo la *baseline* [2] 8
1.6 Piani immagine allineati lungo la *baseline* [2] 9
1.7 Esempio di *match sequence* 11
1.8 Esempio di interpolazione dei pixel 12
1.9 Griglia di ricerca sulle *scanline epipolari* 13
1.10 Esempio di pruning 14
1.11 Tempo di elaborazione standard e con pruning 14

2.1 Architettura modulare della *FLEX* board 18
2.2 *FLEX* Base boards 19
2.3 *FLEX* Demo Board 21
2.4 schema generale del dsPIC®33 DSC 22
2.5 Schema a blocchi della camera CMOS 25
2.6 Filtro a mosaico per la rilevazione dei colori 25
2.7 Architettura di *ERIKA* RTOS 28
2.8 Screenshot del framework Eclipse con il plugin RT-Druid 30
2.9 Architettura stratificata del sistema di comunicazione μWireless . . . 31
2.10 Diagramma delle interazioni delle classi del DAQ [3] 33
2.11 La board Texas Instruments CC2420 EB con l'estensione CC2420EM 34
2.12 Interfaccia del Chipcon CC2420 Packer Sniffer 35

3.1 Schema implementativo della Stereo Camera 37
3.2 Stereo Coordinator 38

4.1 Esempio di ritrasmissione 43

5.1 Immagini e misure prova 1 48
5.2 Immagini e misure prova 2 49

5.3 Immagini e misure prova 3 50
5.4 Immagini e misure prova 4 51
5.5 Tempi per trasmettere un frame 52

B.1 Schematico della scheda Camera Board 59
B.2 Schematico della scheda Camera Control 60

Introduzione

L'obbiettivo della tesi è quello di progettare ed implementare un sensore di stereovisione wireless. Tale sensore serve ad estrarre l'informazione della distanza degli oggetti in una scena utilizzando due telecamere per calcolare la disparità. Tale tipologia potrebbe, ad esempio, trovare impiego in applicazioni robotiche, nella ricostruzione 3D della scena ed in applicazioni di infomobilità.
I dispositivi di stereovisione attualmente in commercio utilizzano componenti hardware costosi ad elevate prestazioni e spesso l'elaborazione viene eseguita su computer; l'idea innovativa del lavoro proposto è quella di realizzare un sistema con hardware minimale che utilizza microcontrollori, una coppia di camere CMOS a basso costo ed elaborazione embedded. Lo sviluppo di software per piattaforme embedded richiede una buona conoscenza dell'hardware e le relative limitazioni, non è sempre possibile realizzare il porting di un algoritmo sviluppato per un computer su dispositivo embedded. La limitazione hardware più stringente è la limitata memoria del dispositivo. Il porting dell'algoritmo richiede una drastica riduzione della memoria RAM utilizzata, tale problema viene risolto eliminando le funzioni non necessarie con le relative strutture dati e ridefinendo quelle indispensabili. La mappa di disparità contenente l'informazione sulla profondità subisce una compressione senza perdite per risparmiare memoria e tempi di trasmissione. La trasmissione wireless avviene sfruttando lo standard IEEE 802.15.4 con lo stack μWireless: l'implementazione di tale stack non prevede ancora la pacchettizzazione di array di grandi dimensioni, tale funzionalità è realizzata nel seguente lavoro implementando un protocollo apposito. L'analisi delle prestazioni è valutata utilizzando un programma appositamente sviluppato.
Si inizia la trattazione con dei cenni di teoria di stereo visione come le regole della geometria epipolare e le tipologie di algoritmi presenti nella computer vision; si descrivono gli strumenti hardware e software utilizzati, in particolare le schede *FLEX*, il microcontrollore e le telecamere; il sistema operativo *ERIKA* e il protocollo wireless. Si discutono le strategie con cui si è operata una riduzione della memoria per poter realizzare il porting dell'algoritmo su dsPIC®.
Infine si riportano i risultati sperimentali sul sistema in varie situazioni, e si valutano le velocità di trasmissione; si discutono le performance a livello di funzionalità e nei confronti di un'architettura distribuita su nodi connessi a radiofrequenza.

Capitolo 1

Cenni di teoria di visione stereoscopica

In questo capitolo vengono descritti gli aspetti teorici relativi alla visione stereoscopica, si inizia parlando del modello della telecamera pinhole e della triangolazione con due telecamere, si introducono i concetti fondamentali di geometria epipolare per arrivare alla descrizione dell'algoritmo di stereovisione utilizzato.

1.1 Telecamera pinhole

Una fotocamera realizza la proiezione prospettica del mondo reale 3D nel piano immagine 2D, il modello più semplice per descrivere questo comportamento è il *pinhole* (fig.1.1). La telecamera *pinhole* è un modello ideale che non tiene conto della distorsione introdotta dalle lenti e della quantizzazione dei pixel. Considerando in una scena un punto P di coordinate (X_p, Y_p, Z_p) la sua proiezione sul piano immagine sarà p' di coordinate (x_p, y_p), come mostrato in figura 1.2. La distanza focale f è definita come la distanza tra il centro della proiezione ed il piano immagine. Utilizzando il *teorema delle lenti sottili* si ottengono le seguenti relazioni:

$$x_p = -\frac{fX_p}{Z_p} \qquad y_p = -\frac{fY_p}{Z_p} \tag{1.1}$$

Le relazioni (1.1) evidenziano che l'immagine risulta invertita sia orizzontalmente che verticalmente. I valori profondità Z_p e distanza focale f determinano la dimensione della proiezione degli oggetti sul piano immagine.

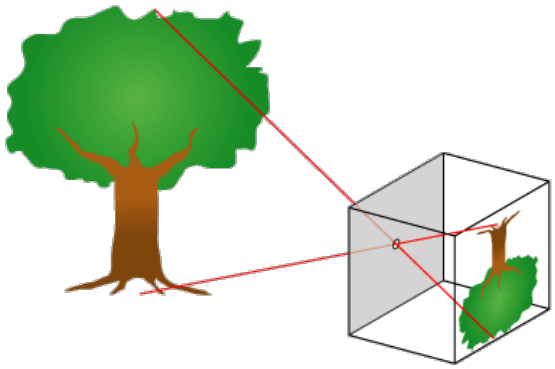


Figura 1.1: Proiezione di un oggetto sul piano immagine [1]

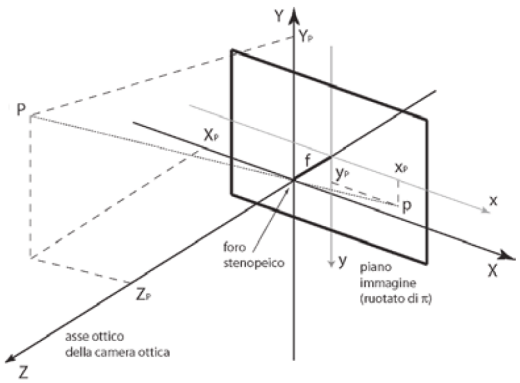


Figura 1.2: Proiezione del punto P sul piano immagine [2]

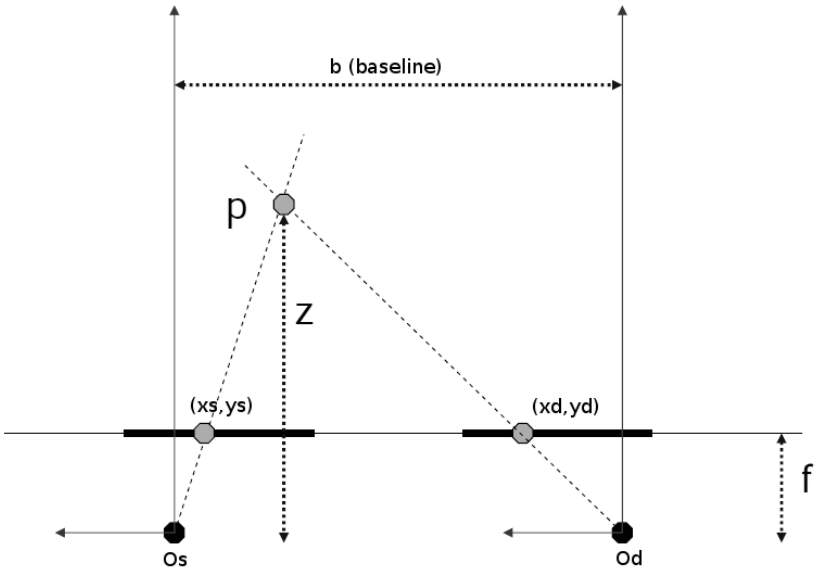


Figura 1.3: Proiezione del punto P su due piani immagine allineati

1.2 Triangolazione con due telecamere

Utilizzando due telecamere è possibile effettuare una triangolazione del punto P osservato per determinare la sua distanza. Per semplicità si descrive la procedura di triangolazione utilizzando due telecamere *pinhole* allineate lungo l'asse x come mostrato in figura 1.3. Le proiezioni del punto P sui piani immagine sinistro e destro generano due coppie di punti, rispettivamente (x_s, y_s) e (x_d, y_d). I centri ottici dei piani sono O_s e O_d, risultano allineati sull'asse x e si trovano ad una distanza b. Analogamente al caso della telecamera pinhole si ottengono le seguenti relazioni di proiezione:

$$x_s = -\frac{f}{Z_p}\left(X_p + \frac{b}{2}\right) \qquad x_d = -\frac{f}{Z_p}\left(X_p - \frac{b}{2}\right) \tag{1.2}$$

$$y_s = -\frac{fY_p}{Z_p} \qquad y_d = -\frac{fY_p}{Z_p} \tag{1.3}$$

Utilizzando le equazioni 1.2 e 1.3 è possibile ottenere due nuove grandezze δ_o e δ_v chiamate *disparità* e definite nel caso specifico:

$$\delta_o = x_d - x_s = -\frac{f}{Z_p}b \qquad \delta_v = 0 \tag{1.4}$$

La disparità orizzontale δ_o genera nell'osservatore la percezione della profondità e permette di calcolare la distanza dell'oggetto. La disparità verticale δ_v è uguale a 0 perché i centri ottici sono allineati lungo l'asse y; quando questa condizione non è verificata si introduce la disparità verticale che genera nell'osservatore affaticamento visivo.

1.3 Geometria epipolare

La geometria epipolare analizza le configurazioni geometriche nella visione stereoscopica, individuando le relazioni presenti tra i punti della scena tridimensionale e le relative proiezioni sulle superfici dei sensori. In generale il piano immagine si trova dopo il centro focale, ma utilizzando la geometria epipolare si preferisce introdurre la semplificazione di un piano immagine virtuale prima del centro focale eliminando così il capovolgimento dell'immagine.
Per comprendere il funzionamento della geometria epipolare è necessario definire tre elementi fondamentali (fig.1.4):

- **Epipoli.** Gli epipoli sono i punti di intersezione dei piani immagine con la retta passante per i centri ottici. La retta su cui giacciono i punti epipolari (E_L, E_R) e quindi anche i centri ottici (O_L, O_R) prende il nome di *baseline*.

- **Linea epipolare.** Il punto p_L rappresenta la proiezione sul piano immagine sinistro del punto P, ed è posizionato sull'intersezione della retta O_L-P con il piano immagine. La proiezione della retta O_L-P genera sul piano immagine destro un segmento passante per E_R su cui giace il punto p_R. Tale segmento prende il nome di linea epipolare. Lo stesso ragionamento può essere fatto partendo dalla proiezione di P sul piano immagine destro (p_R) per determinare la linea epipolare $E_L - p_L$ sul piano sinistro.

- **Piano epipolare.** I punti P, O_L e O_R determinano il piano π chiamato piano epipolare. Le intersezioni di tale piano con i piani immagine dei sensori corrispondono alle linee epipolari.

Dato un punto p_L è possibile ricavare p_R utilizzando le regole della geometria epipolare e ridurre l'area di ricerca sul piano immagine destro effettuando la ricerca lungo la linea epipolare E_R-p_R. Il rispetto di questo vincolo ha il vantaggio di ridurre l'area per il matching eliminando molti punti che genererebbero delle corrispondenze spurie. Quando si determinano p_L e p_R, chiamati punti corrispondenti, è possibile calcolare la posizione di P con una triangolazione come descritto nella sezione 1.2.

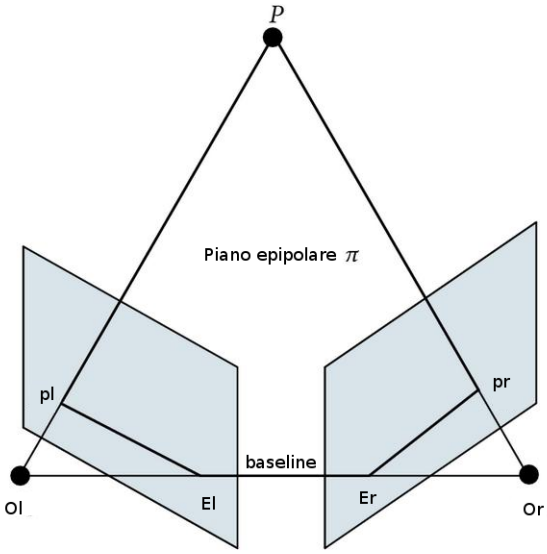


Figura 1.4: Elementi fondamentali della geometria epipolare

1.3.1 Matrice essenziale E

La matrice essenziale **E** è una matrice di dimensioni 3x3 e rango 2 che mette in relazione i punti corrispondenti di una coppia di immagini stereoscopiche. La matrice essenziale **E** contiene le informazioni relative ad una rotazione **R** e a una traslazione **T**. Considerando un punto P nello spazio (X,Y,Z), la sua proiezione sul piano immagine sinistro sarà rappresentata dal punto p_L, il punto p_R sul piano immagine destro sarà ricavato dalla seguente relazione:

$$p_R = \mathbf{R}(p_L - \mathbf{T})\,. \tag{1.5}$$

Se $\mathbf{P_L}$ è il vettore che unisce il centro ottico sinistro O_L al punto p_L e se $\mathbf{P_R}$ è il vettore che unisce il centro ottico destro O_R al punto p_R [4], risulta:

$$\mathbf{P_R} = \mathbf{R}(\mathbf{P_L} - \mathbf{T})\,. \tag{1.6}$$

dove **T** rappresenta la traslazione (*baseline*) e **R** la rotazione. Per ogni punto x in un piano con vettore normale **n** e passante attraverso il punto a si ottiene il seguente vincolo:

$$(x - a) \cdot \mathbf{n} = 0\,. \tag{1.7}$$

Considerando i vettori del piano epipolare $\mathbf{P_L}$ e $\mathbf{T}$, il loro prodotto vettoriale è perpendicolare ad entrambi proprio come $\mathbf{n}$. Si ottiene la seguente relzione:

$$(\mathbf{P_L} - \mathbf{T})^T(\mathbf{T} \times \mathbf{P_L}) = 0\,. \tag{1.8}$$

La relazione 1.5 può essere riscritta come:

$$(\mathbf{P_L} - \mathbf{T}) = \mathbf{R}^{-1}\mathbf{P_R}\,. \tag{1.9}$$

tenendo conto che $\mathbf{R}^T = \mathbf{R}^{-1}$ si ottiene:

$$(\mathbf{R}^T\mathbf{P_R})^T(\mathbf{T} \times \mathbf{P_L}) = 0\,. \tag{1.10}$$

Il prodotto vettoriale può essere riscritto nel seguente modo $\mathbf{T} \times \mathbf{P_L} = \mathbf{S}\mathbf{P_L}$ con $\mathbf{S}$ così definita:

$$\mathbf{S} = \begin{pmatrix} 0 & -T_z & -T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix} \tag{1.11}$$

Dalla 1.10 si ottiene la relazione $(\mathbf{P_R})^T\mathbf{RSP_L} = 0$ dove il prodotto $\mathbf{RS}$ rappresenta la matrice essenziale $\mathbf{E}$.

1.3.2 Matrice fondamentale F

La matrice essenziale $\mathbf{E}$ non contiene nessuna informazione intrinseca relativa alle due telecamere. Senza questa informazione non è possibile mappare un pixel dell'immagine di sinistra su quella destra. Per ottenere queste informazioni si considera un punto p (contenente le coordinate del pixel) che si può scrivere come $P = \mathbf{M}^{-1}p$ dove M rappresenta la matrice intrinseca della telecamera. La relazione $(\mathbf{P_R})^T\mathbf{E}\mathbf{P_L} = 0$ diventa

$$p_R^T(\mathbf{M_R}^{-1})^T\mathbf{E}\mathbf{M_L}^{-1}p_L = 0\,. \tag{1.12}$$

ed è possibile definire la matrice fondamentale $\mathbf{F}$ come:

$$\mathbf{F} = (\mathbf{M_R}^{-1})^T\mathbf{E}\mathbf{M_L}^{-1}\,. \tag{1.13}$$

Con la proprietà fondamentale 1.12, la matrice $\mathbf{F}$ permette di verificare se una coppia di punti estratti da una coppia di immagini stereoscopiche corrispondono allo stesso punto della scena reale. La matrice fondamentale può essere stimata utilizzando almeno sette coppie di punti corrispondenti p_L e p_R. I suoi sette parametri rappresentano l'unica informazione geometrica relativa alle camere che può essere ottenuta utilizzando solo punti corrispondenti.

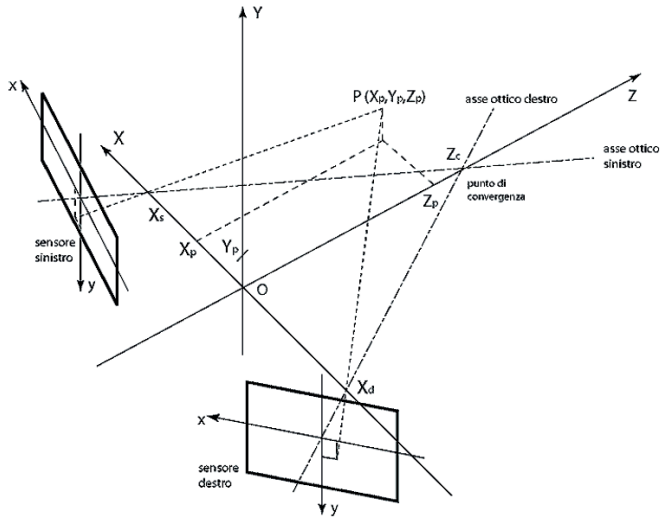


Figura 1.5: Piani immagine non allineati lungo la *baseline* [2]

1.3.3 Matrice intrinseca della telecamera

La matrice intrinseca della telecamera è definita a partire dalle lunghezze focali f_x ed f_y relative agli assi x ed y, e ai parametri c_x e c_y relativi alle coordinate del centro del sensore. Nel caso di due telecamere identiche e perfettamente allineate si avranno due matrici $\mathbf{M}$ che differiscono solo per l'elemento c_x.

$$\mathbf{M} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{1.14}$$

1.3.4 Rettificazione epipolare

In natura le rette che uniscono i centri ottici degli occhi con un oggetto osservato non sono parallele (fig.1.5). Le regole di geometria epipolare appena introdotte si possono applicare al caso generico di una coppia di telecamere con un'inclinazione qualsiasi e si possono semplificare i calcoli orientando parallelamente i sensori.

Nel caso in cui non sia possibile modificare la posizione delle camere la semplificazione può essere ottenuta operando una rettificazione epipolare, equivalente ad una rotazione virtuale. In questo caso (fig.1.6) le linee epipolari coincidono e sono parallele alla retta che contiene i due punti focali O_L e O_R. Ciò implica che per ogni punto di un'immagine il suo punto corrispondente può essere trovato cercando solo lungo le righe dell'altra.

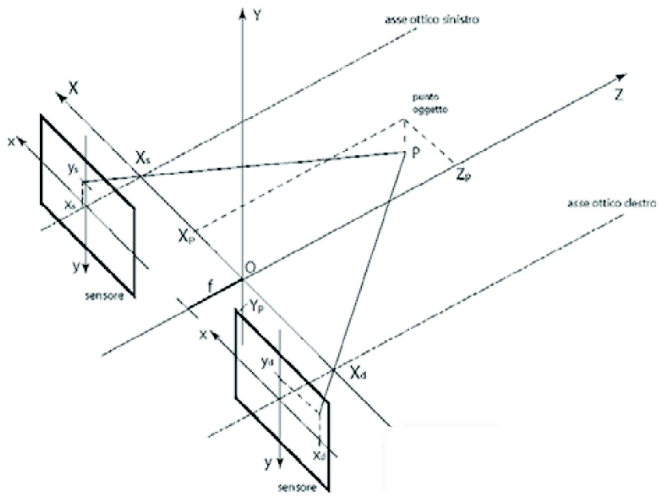


Figura 1.6: Piani immagine allineati lungo la *baseline* [2]

1.4 Algoritmo di stereovisione

Processando una coppia di immagini stereo si ottiene un'immagine in scala di grigio chiamata mappa di disparità, l'informazione sulla distanza è contenuta nel livello di luminosità di ogni pixel.
Gli algoritmi di stereovisione si dividono in due famiglie principale, *feature-based* ed *area-based* [5]. I primi operano in due fasi, nella prima vengono rilevate le feature interessanti come contorni o segmenti e nella seconda fase si procede ai loro abbinamenti con quelli presenti nell'altra immagine. Questo approccio permette di elaborare selettivamente solo alcune porzioni dell'immagine riducendo i tempi di calcolo e inoltre fornisce stime affidabili di distanza se l'abbinamento è corretto. Operando con le "feature" si incontrano 3 limitazioni principali:

- limitandosi ad alcune porzioni dell'immagine non si possono generano mappe di disparità dense;
- è fondamentale scegliere correttamente le "feature";
- non è possibile trovare "feature" generali per trattare qualsiasi tipo di scena.

Tali limitazioni non fanno funzionare questa tipologia di algoritmi in una scena in cui sono presenti oggetti con forma irregolare, come persone e piante.
Gli algoritmi *area-based* operano in una sola fase, direttamente sui valori di intensità delle immagini cercando le aree corrispondenti. L'immagine viene elaborata in finestre quadrate, nelle quali si cercano le corrispondenze facendo uso di operatori di

confronto, come gli operatori di correlazione, di correlazione normalizzata, di differenza, di distanza quadratica [5]. La scelta dell'operatore dipende dal bilancio tra il costo computazionale e le caratteristiche desiderate come indipendenza dalle differenze di luminosità, sensibilità al rumore, facilità di realizzazione. Le corrispondenze tra le immagini stereo vengono cercate lungo le linee epipolari (1.3), ad ogni pixel in un'immagine corrisponde un insieme di possibili pixel disposti lungo la linea epipolare dell'altra.
Gli algoritmi *area-based* migliorano le criticità presenti in quelli *feature-based*, possono lavorare con qualsiasi tipo di immagine e generano delle mappe di disparità dense.

1.4.1 Depth Discontinuities by Pixel to Pixel Stereo

L'algoritmo di stereovisione utilizzato in questo algoritmo è il *Depth Discontinuities by Pixel to Pixel Stereo* [6] ed è di tipo *area-based* [5]. La scelta di tale algoritmo è dovuta alla sua semplicità implementativa, alla ridotta richiesta di strutture dati e alla sua capacità di lavorare con qualsiasi scena anche in assenza di texture (informazione che incide sull'aspetto di una superficie, come una trama di tela oppure l'irregolarità di un muro). L'algoritmo si può dividere in 4 "macro" step:

1 - Si elaborano a coppie le colonne di entrambe le immagini, utilizzando una funzione di costo si minimizza la differenza dell'intensità di ogni pixel e il numero e la lunghezza delle occlusioni.

2 - Utilizzando un'immagine con uno sfondo senza texture la localizzazione dei cambiamenti di disparità è fortemente determinata dal rumore. Si può risolvere questo problema notando che le discontinuità di profondità sono accompagnate da una variazione di intensità, ciò implica che un'occlusione a sinistra deve trovarsi immediatamente a sinistra di una variazione di intensità.

3 - Dopo aver stabilito la posizione delle discontinuità di profondità, restano delle macchie nella mappa di disparità dovute al campionamento dei pixel. Per risolvere questo problema si utilizza una funzione intensità indipendente dal campionamento con cui confrontare ogni pixel. Utilizzando questo metodo la mappa di disparità diventa più pulita.

4 - Per rendere il calcolo veloce le colonne della mappa di disparità vengono post-processate indipendentemente. L'idea di base è di propagare i valori di disparità affidabili in regioni in cui non lo sono, il criterio per stabilire l'affidabilità si basa sul numero di pixel in una colonna con un valore simile di disparità. Il processo di propagazione si arresta quando si incontra una variazione d'intensità maggiore di una determinata soglia. La mappa di disparità così ottenuta presenta

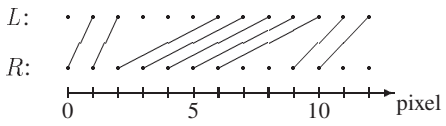


Figura 1.7: Esempio di *match sequence*

ancora degli errori, perché non contiene lungo i confini orizzontali delle variazioni di intensità. Questa situazione contraddice l'assunzione che variazioni di disparità sono accompagnate a variazioni di intensità. Per risolvere il problema si utilizza una propagazione di valori affidabili di disparità in regioni vicine, fino ad incontrare una zona in cui la disparità è di due livelli superiore. Questo vincolo permette di rilevare le superfici inclinate.

Generalità L'algoritmo tratta le immagini come vettori anziché come matrici bidimensionali. Il vettore si ottiene concatenando le colonne dell'immagine (*scanline*), di conseguenza la dicitura *(x,y)* indica una coppia di pixel (immagine sinistra e destra) e non le coordinate su una singola immagine. Elaborando le immagini con questa convenzione i pixel che rappresentano lo stesso oggetto nelle due immagini ($I_l(x)$ e $I_r(y)$) vengono memorizzati in una *match sequence* M. La disparità $\delta(x)$ di un pixel x nell'immagine sinistra abbinato ad un pixel y in quella destra è definita come x-y; ai pixel di una zona occlusa si assegna il valore di disparità massimo degli oggetti vicini. In figura 1.7 è riportata a titolo d'esempio una *match sequence* M di 13 pixel, i 5 abbinamenti centrali rappresentano un oggetto vicino.

Funzione Costo La funzione costo $\gamma(M)$ utilizzata per stabilire la veridicità della *match sequence* è definita come:

$$\gamma(M) = N_{occ}k_{occ} - N_m k_r + \sum_{i=1}^{n} d(x_i, y_i) \tag{1.15}$$

dove N_{occ} rappresenta il numero di occlusioni e N_m il numero di match, k_{occ} rappresenta la costante di penalità per ogni occlusione, k_r la costante di ricompensa per ogni match e d(x,y) la dissimilarità di ogni match.

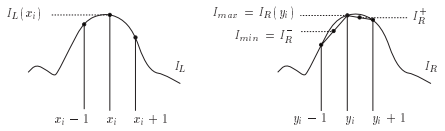


Figura 1.8: Esempio di interpolazione dei pixel

Disparità dei pixel La disparità non può essere ottenuta tramite la differenza di intensità tra $I_l(x)$ e $I_r(y)$ perché le immagini campionate possono rendere questo parametro grande vicino ai margini di intensità. Questo problema può essere ridotto realizzando un interpolazione lineare nei dintorni di tre pixel; in figura 1.8 vengono mostrate le intensità dei pixel dell'immagine sinistra e destra (x_i, y_i) in cui stabilire la disparità. Si definisce $\hat{I_R}$ come l'interpolazione lineare tra i pixel della linea di scansione destra (y_{i-1}, y_i, y_{i+1}), si sceglie il valore di disparità definito nel seguente modo:

$$d(x_i, y_i) = min\{\bar{d}(x_i, y_i, I_L, I_R), \bar{d}(y_i, x_i, I_R, I_L)\} \tag{1.16}$$

dove:

$$\bar{d}(x_i, y_i, I_L, I_R) = \min_{y_i - \frac{1}{2} \leq y_i \leq y_i + \frac{1}{2}} |I_L(x_i) - I_R^{'}(y_i)| \tag{1.17}$$

e in modo simmetrico:

$$\bar{d}(y_i, x_i, I_R, I_L) = \min_{x_i - \frac{1}{2} \leq x_i \leq x_i + \frac{1}{2}} |I_R(y_i) - I_L^{'}(x_i)| \tag{1.18}$$

Il calcolo di $\bar{d}(x_i, y_i, I_L, I_R)$ si divide in 4 step:

$$I_R^- \equiv I_R^{'}(y_i - \frac{1}{2}) = \frac{1}{2}\big(I_R(y_i) + I_R(y_i - 1)\big) \tag{1.19}$$

$$I_R^+ \equiv I_R^{'}(y_i + \frac{1}{2}) = \frac{1}{2}\big(I_R(y_i) + I_R(y_i + 1)\big) \tag{1.20}$$

$$I_{min} = min\big(I_R^-, I_R^+, I_R(y_i)\big) \qquad I_{max} = max\big(I_R^-, I_R^+, I_R(y_i)\big) \tag{1.21}$$

$$\bar{d}(x_i, y_i, I_L, I_R) = max\big\{0, I_L(x_i) - I_{max}, I_{min} - I_L(x_i)\big\} \tag{1.22}$$

$$\tag{1.23}$$

analogamente si procede per il calcolo di $\bar{d}(y_i, x_i, I_R, I_L)$.

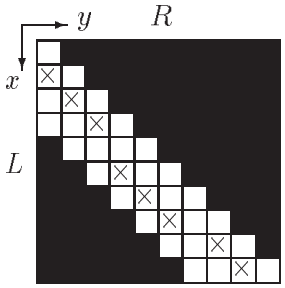


Figura 1.9: Griglia di ricerca sulle *scanline epipolari*

Vincoli Oltre alla misura dello verosimiglianza di una *match sequence* con la sua funzione costo, si richiede il rispetto di alcuni vincoli che permettono di rilevare e processare oggetti nella scena che non presentano texture e di accelerare il processo di ricerca. Molti algoritmi stereo non sono in grado di trattare adeguatamente le scene prive di texture. Spesso tali scene vengono alterate artificialmente ponendo una texture sullo sfondo per poter essere utilizzate dal particolare algoritmo. L'algoritmo scelto nel presente lavoro è in grado di maneggiare degli oggetti nella scena privi di texture utilizzando il seguente vincolo: variazioni di intensità sono accompagnate da discontinuità di profondità. Nell'implementazione pratica si verifica che qualsiasi terna di pixel consecutivi ha una differenza di intensità di almeno cinque livelli. Tale situazione indica la presenza di una debole variazione di profondità. Il secondo vincolo utilizzato riguarda la modalità di ricerca dei match ed è implementato come segue: se (x, y) rappresenta un abbinamento, allora o $x + 1$ o $y + 1$ devono essere abbinati.

Ricerca lungo le scanline epipolari Nella figura 1.9 è illustrato un esempio relativo a due *scanline* di 10 pixel ognuna, utilizzando una disparità massima di 3 pixel. La *match sequence* è marcata con il carattere x negli elementi della griglia di ricerca. Un algoritmo standard trova il percorso migliore esaminando iterativamente tutte le celle che compongono la griglia di ricerca. La scelta migliore è invece un algoritmo pruning che elimina dei percorsi di ricerca utilizzando il seguente criterio: *a parità di riga non si espande la ricerca verso destra nel caso in cui la funzione costo sia maggiore, e con lo stesso criterio a parità di colonna non si espande la ricerca verso il basso.* La complessità computazionale dell'algoritmo completo è di $O(n\Delta^2)$, mentre quella della versione semplificata è di $O(n\Delta log\Delta)$, indicando con n il numero di pixel e con Δ il numero di livelli di disparità utilizzati. Nella figura 1.10 ad esempio si presentano due possibilità, raggiungere c passando per q oppure per p, supponendo che $\gamma(q) < \gamma(p)$ si sceglie il percorso che passa per q. In figura 1.11 viene mostrato il confronto sui tempi di elaborazione dell'algoritmo completo e pruning.

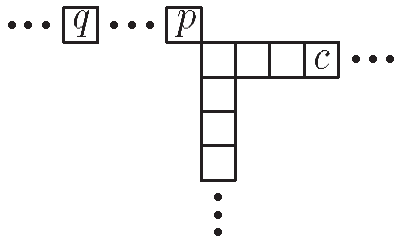


Figura 1.10: Esempio di pruning

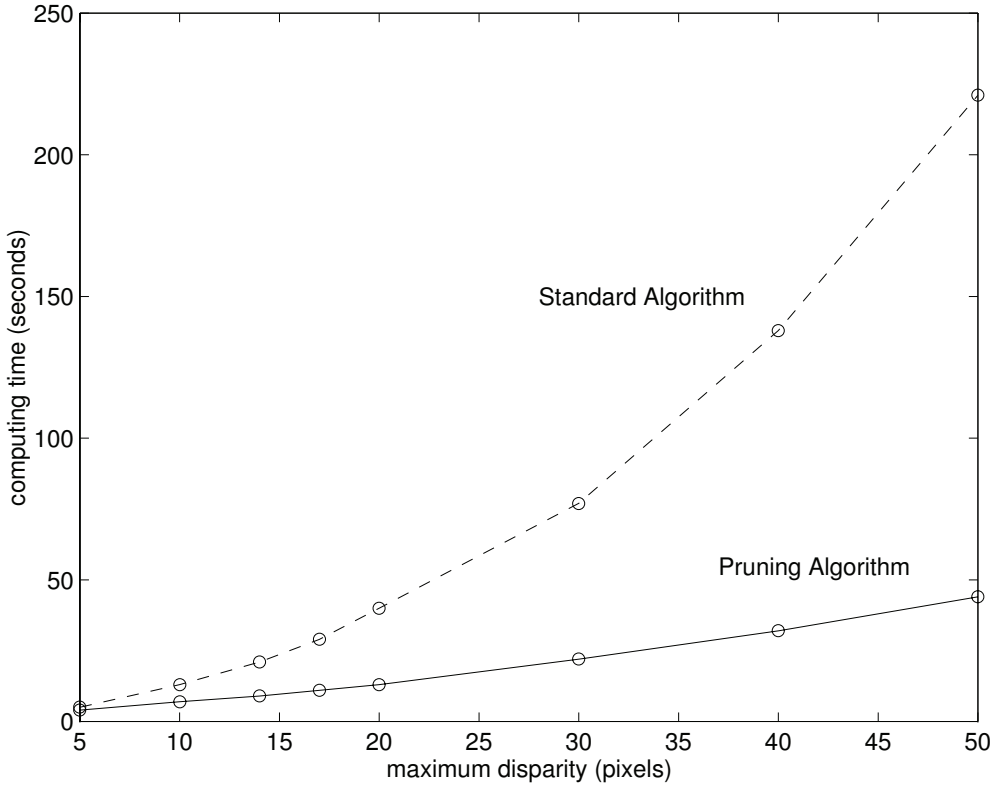


Figura 1.11: Tempo di elaborazione standard e con pruning

Implementazione dell'algoritmo L'algoritmo di stereovisione è stato implementato in linguaggio C e genera la mappa di disparità utilizzando due funzioni principali *matchScanlines* e *postprocess*. La funzione *matchScanlines* genera una prima versione della mappa di disparità affetta da rumore ed imprecisioni, mentre la *postprocess* permette di rendere il risultato dell'elaborazione precedente più preciso. La funzione *matchScanline* richiede in input la coppia di immagini stereo e restituisce in output la mappa di disparità utilizzando al suo interno 2 ulteriori funzioni per effettuare le seguenti pre-elaborazioni:

- si realizza una pre-elaborazione della disparità tra le due coppie di immagini utilizzando la funzione *fillDissimilarityTable* che esegue l'operazione descritta nell'equazione 1.16;
- si calcola il gradiente delle intensità lungo l'asse X con la funzione *computeIntensityGradientsX*. Tale funzione determina quale pixel deve restare accanto ad un gradiente di intensità implementando la regola che un pixel a sinistra della *scanline* viene mantenuto se è presente una variazione di intensità alla sua destra.

La funzione *postprocess* richiede in input l'immagine sinistra e la prima versione della mappa di disparità e restituisce la mappa di disparità definitiva. Utilizza al suo interno le seguenti funzioni, elencate in ordine sequenziale di chiamata:

- *coerceSurroundedPixelsY* rimuove gli errori evidenti dalla mappa di disparità;
- *compute_igxy* calcola il gradiente di intensità;
- *removeIsolatedPixelsX* rimuove i pixel isolati dai gradienti di intensità nella direzione y;
- *compute_igyd* genera una mappa booleana dei gradienti di intensità lungo y;
- *computeReliabilitiesY* genera una mappa di pixel affidabili lungo la direzione y;
- *propagateY* propaga i pixel affidabili lungo la direzione y;
- *removeIsolatedPixelsY* rimuove i pixel isolati dai gradienti di intensità nella direzione x;
- *compute_igxd* genera una mappa booleana dei gradienti di intensità lungo x;
- *computeReliabilitiesX* genera una mappa di pixel affidabili lungo la direzione x;
- *propagateX* propaga i pixel affidabili lungo la direzione x;

- *modefilterY* filtra la mappa di disparità finale lungo la direzione y;
- *modefilterX* filtra la mappa di disparità finale lungo la direzione x.

Capitolo 2

Ambiente di Sviluppo

Nella programmazione di applicazioni su PC si ha normalmente la tendenza ad ignorare le caratteristiche hardware e si implementano algoritmi senza considerare l'occupazione di memoria RAM o la loro complessità computazionale. Quando si lavora con sistemi embedded si ha a che fare con ridotte capacità hardware e diventa così obbligatorio conoscere nel dettaglio le caratteristiche del dispositivo.
In questo capitolo sono illustrate le caratteristiche HW, fornendo una panoramica delle board di sviluppo, del microcontrollore e del transceiver 802.15.4 e le caratteristiche SW, introducendo il sistema operativo real-time *ERIKA* e la libreria μWireless. Sono infine descritti gli strumenti di debugging utilizzati.

2.1 Piattaforma Hardware

Per realizzare il progetto sono state utilizzate quattro schede di sviluppo *FLEX*.

2.1.1 FLEX embedded board

FLEX è una scheda di sviluppo basata sui microcontrollori della famiglia dsPIC®DSC di Microchip [7], nata dalla collaborazione di due aziende italiane, *Evidence* [8] ed *Embedded Solutions* [9]. Si tratta a tutti gli effetti di una evaluation board che permette lo sviluppo e il testing di applicazioni real-time per dsPIC®DSC in breve tempo e in modo relativamente semplice. Ciò è dovuto a due caratteristiche fondamentali che contraddistinguono il sistema:

1 - architettura hardware modulare, unita a una buona robustezza della parte elettronica;

2 - supporto del kernel real-time *ERIKA Enterprise* e del suo ambiente di sviluppo (cfr. 2.3.2).

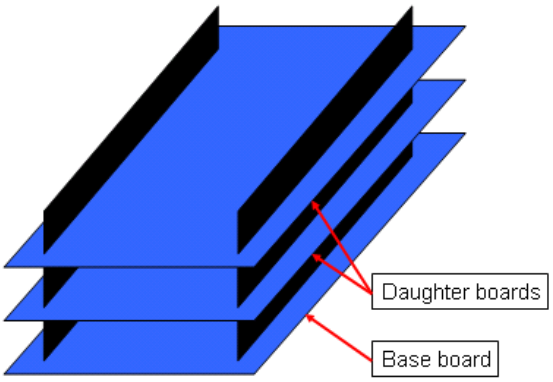


Figura 2.1: Architettura modulare della *FLEX* board

Architettura modulare

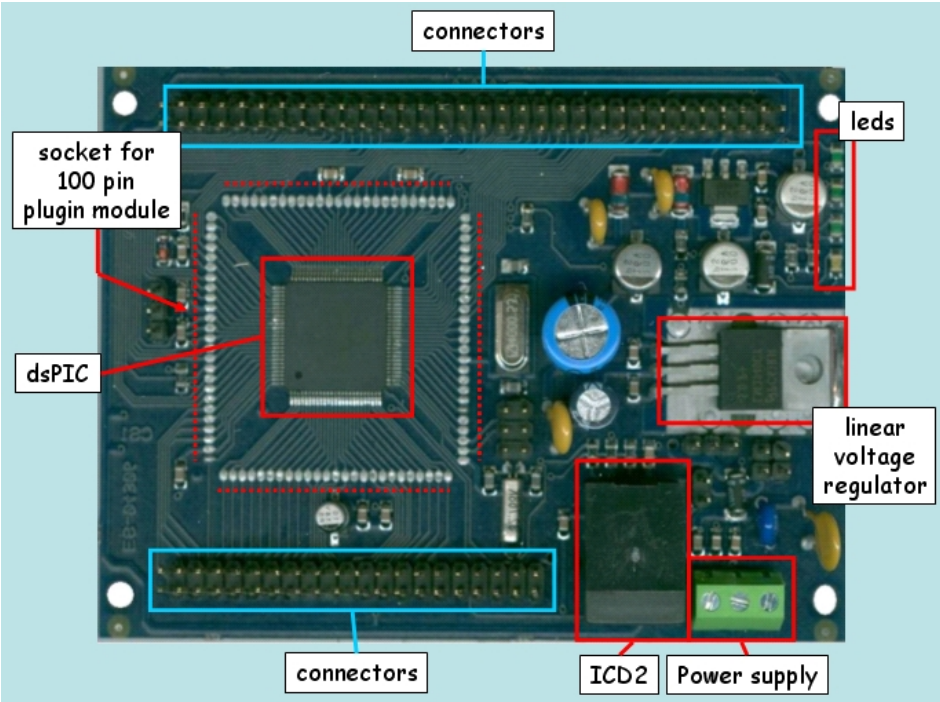
L'architettura *FLEX* prevede la presenza di una "scheda madre" principale e di un insieme di "schede figlie" (*daughter boards*) che possono essere montante in combinazione con la board di base per costituire dei dispositivi complessi. La figura 2.1 riassume simbolicamente l'architettura.

Base board

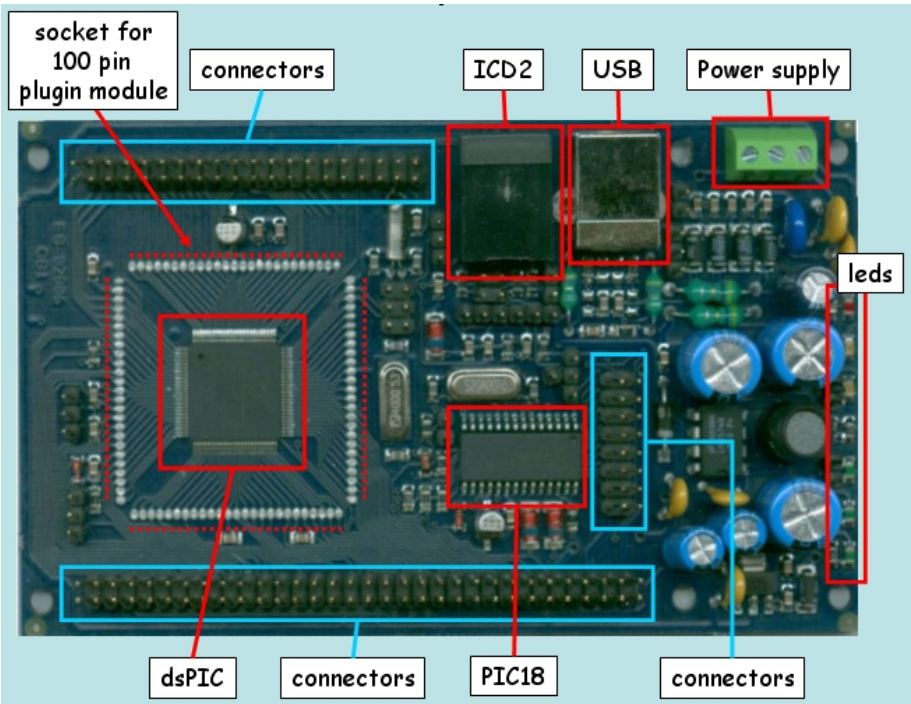
La *FLEX base board* contiene il dispositivo programmabile, lo stadio di alimentazione ed il connettore necessario alla programmazione. Tutti i pin disponibili del dsPIC®vengono esportati per mezzo di connettori standard a passo 2.54mm, semplificando così l'interconnessione con circuiti elettronici esterni.
Esistono due versioni disponibili della scheda: *FLEX Light* e *FLEX Full* (fig. 2.2). Tra le varie caratteristiche oltre al microcontrollore Microchip dsPIC®DSC dsPIC33FJ256MC710 di cui si parlerà tra poco, i connettori di alimentazione, le cui tensioni di ingresso nel caso *Light* sono nel range 9 $\sim$ 12V, e un insieme di led (molto utili in fase di sviluppo). La versione *Full* ha un circuito di alimentazione più complesso e molto più robusto e altri componenti hardware come il connettore USB e il microcontrollore PIC18. Tuttavia la *FLEX Light board* risulta più compatta e adatta per sistemi distribuiti alimentati a batteria.

Daughter boards

Le *daughter board* possono essere montate al di sopra della *base board* sfruttando i connettori standard a passo 2.54mm. Sono delle schede sulle quali possono essere presenti dispositivi di qualunque genere che si interfacciano con il microcontrollore.



(a) *FLEX Light*



(b) *FLEX Full*

Figura 2.2: *FLEX* Base boards

Inoltre è possibile montare una *daughter board* su un'altra in modo da costruire sistemi anche molto complessi.
La *daughter board* utilizzata per realizzare il coordinatore wireless è la *FLEX Demo Board* (fig. 2.3) che permette di aggiungere alla *FLEX* base board diverse funzionalità tra cui:

- 2 output DAC (con risoluzione a 12 bit);
- un accelerometro a 3 assi;
- un set di 4 pulsanti;
- un set di 8 LED;
- un LCD (16 caratteri x 2 linee);
- un buzzer;
- un potenziometro;
- un sensore di temperatura;
- un sensore di luminosità;
- un connettore per transceiver 802.15.4.

In particolare i LED e il display LCD sono stati utilizzati nella fase di implementazione come strumenti di debug, mentre si è fatto uso della comunicazione seriale per interfacciarsi con il PC e del modulo radio per comunicare con il dispositivo stereo. Quest'ultimo è stato realizzato utilizzando tre *FLEX Light* e 2 *daughter board* millefori.

2.1.2 Microcontrollore dsPIC33FJ256MC710

Il microcontrollore della *FLEX* board è il dsPIC33FJ256MC710 di Microchip [7] della famiglia dsPIC®DSC (Digital Signal Controller). È un'architettura a 16 bit dotata di un'unità DSP che permette di ottenere alte prestazioni (40 Mips) nell'elaborazione di segnali digitali. Le periferiche presenti ed un elevato numero di pin digitali a disposizione (fino a 85) rendono questo microcontrollore molto adatto per complesse applicazioni di sensing o di controllo.

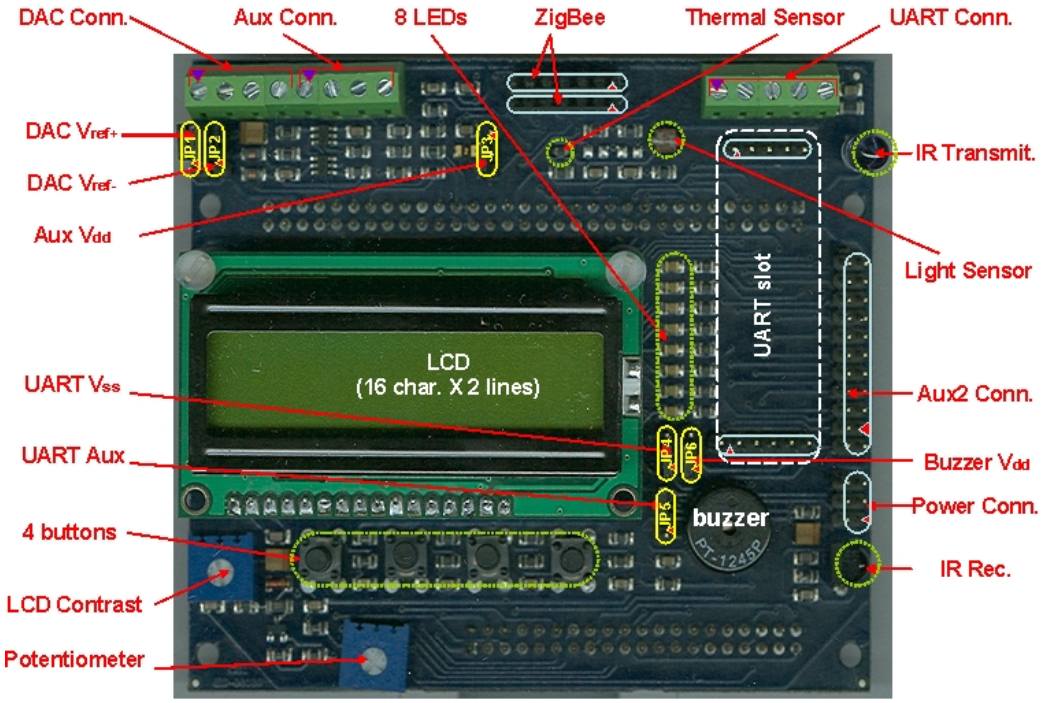


Figura 2.3: *FLEX* Demo Board

Architettura

In figura 2.4, estratta dal manuale relativo al dispositivo, è mostrato lo schema a blocchi generale del dsPIC®in questione. D'ora in avanti si farà sempre riferimento allo specifico modello dsPIC33FJ256MC710.
L'architettura dell'MCU è di tipo Harvard modificata. L'*instruction set* e quindi il bus per la memoria codice sono a 24 bit, il *data path* ed i relativi bus dati sono a 16 bit; il dsPIC®utilizza due bus dati per accedere parallelamente a due zone di memoria. La memoria in particolare è organizzata nel seguente modo:

- Data Memory
  I primi 2 KB servono a mappare in memoria gli *Special Function Register* (SFR) ovvero i registri usati dalla CPU e dalle periferiche. Seguono 30 KB di SRAM che costituiscono spazio per: X Data RAM (accessibile dal relativo X Data Bus), Y Data RAM (accessibile dal relativo Y Data Bus), DMA RAM (2K di memoria dual-port dedicata al DMA). Infine è possibile mappare una parte della memoria codice (di programma) nella memoria dati (Program Visibility Space - PSV).

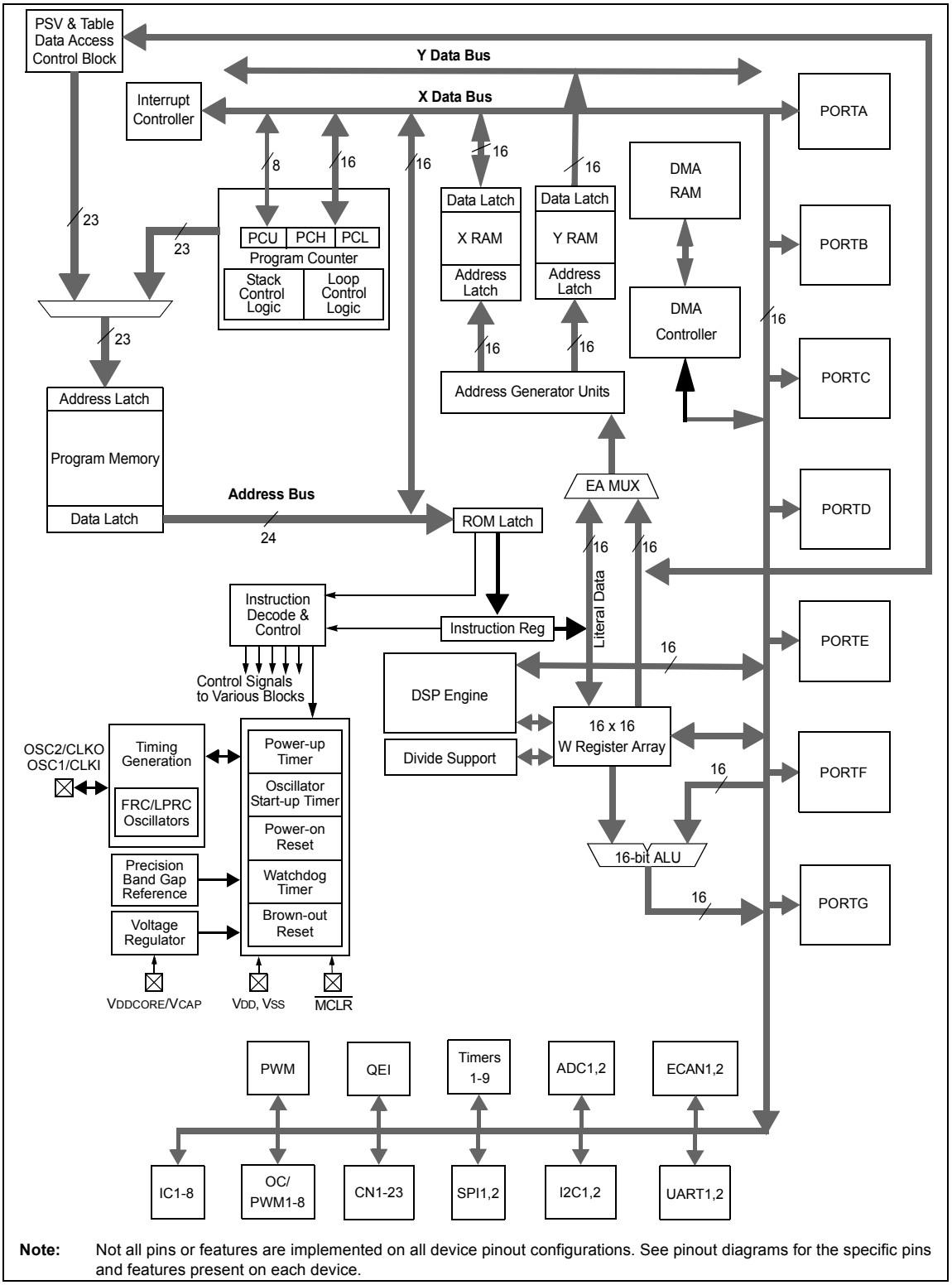


Figura 2.4: schema generale del dsPIC®33 DSC

- Program Memory
  Memoria flash di 256KB per il codice.

La CPU è costituita da una parte MCU e da una DSP. È possibile lavorare con differenti clock, il massimo che si può ottenere con l'oscillatore interno e il PLL è una frequenza di clock di 80MHz che corrisponde ad un rate massimo di istruzioni dichiarato dal costruttore di 40MIPS.
Nel dsPIC®è presente anche un controllore DMA ad 8 canali.

Periferiche

Tra le periferiche presenti sul dsPIC33FJ256MC710 si ricordano:

- 9 Timers a 16 bit (con prescaler programmabile); 8 di questi possono essere usati in coppia per formare timers a 32 bit.
- 8 canali di Input Capture e 8 canali di Output Compare
- 2 SPI, 2 I^2C^{TM}, 2 UART, 2 Enhanced CAN (ECAN™)
- 8 canali Motor Control PWM
- 1 modulo Quadrature Encoder Interface
- 2 moduli ADC

Supporto per Programmazione/Debugging

I microcontrollori dsPIC®33 hanno delle interfacce per la programmazione e la diagnostica del dispositivo stesso. In particolare sono messe a disposizione tre modalità: *In-Circuit Serial Programming™* (ICSP™), *Enhanced ICSP* e *Joint Test Action Group* (JTAG).

La prima è uno strumento proprietario di Microchip e sviluppato per consentire la programmazione dei propri dispositivi. È quindi integrata direttamente nel core del microcontrollore dove è presente una macchina a stati che regola la scrittura in memoria. Il processo deve comunque essere supervisionato dell'esterno con HW e SW opportuno.

ICSP gestisce anche un canale per il debugging sul chip (*in-circuit debugging*). In questo caso è necessario utilizzare un dispositivo esterno appropriato che carichi nel microcontrollore il SW per il debugging.

MPLAB ICD2 è un oggetto in grado di gestire questo processo in modo automatico e che può essere comodamente controllato da PC nell'ambiente di sviluppo di Microchip (*MPLAB IDE*).

2.2 Camera CMOS

La telecamera CMOS HV7131GP è un dispositivo a colori con una risoluzione massima di 652 x 488 pixel e viene prodotta dalla MagnaChip, con la sua tecnologia proprietaria per la realizzazione di sensori CMOS in tecnologia a 0,35μm. Ogni pixel è realizzato con 4 transistor MOS dotati ognuno di una propria microlente necessaria ad aumentare la sensibilità, mentre l'energia luminosa relativa a un colore fondamentale (Red, Green, Blue) è ottenuta con un filtro a mosaico del tipo riportato in figura 2.6. Il valore di luminosità rilevato da un Mosfet posto dopo il filtro viene convertito in segnale elettrico e quantizzato da un ADC a 10 bit. I segnali digitalizzati costituiscono l'input per i circuiti di signal processing che realizzano le seguenti correzioni:

- compensazione del livello del nero;
- autobilanciamento del bianco;
- controllo automatico di esposizione;
- correzione gamma;
- conversione dei colori da RGB a YCbCr;
- interpolazione del colore con matrice 3x3.

Dopo la fase di postprocess il frame viene inviato al dispositivo di controllo in formato RAW bitmap. La comunicazione della camera CMOS con il dispositivo richiede l'utilizzo di un driver descritto in 2.2.1 e di tre componenti hardware fondamentali:

- un clock da 10 Mhz proveniente da un dispositivo esterno e prelevato dal pin MCLK;
- un bus I^2C da cui riceve i parametri di inizializzazione e i comandi;
- un bus ad 8 bit parallelo con cui trasmette i pixel rilevati al dispositivo di controllo.

2.2.1 Driver per la Camera CMOS HV7131GP

La gestione della camera HV7131GP sul dsPIC è affidata ad un apposito driver sviluppato per il sistema operativo *ERIKA Enterprice* (cfr.2.3.2). Il driver mette a disposizione dello sviluppatore l'astrazione del livello hardware e una lista di comandi per gestire la telecamera. È possibile realizzare il porting del driver su altre tipologie di microcontrollore modificando il file relativo all'astrazione del livello hardware. I comandi del driver più rilevanti sono:

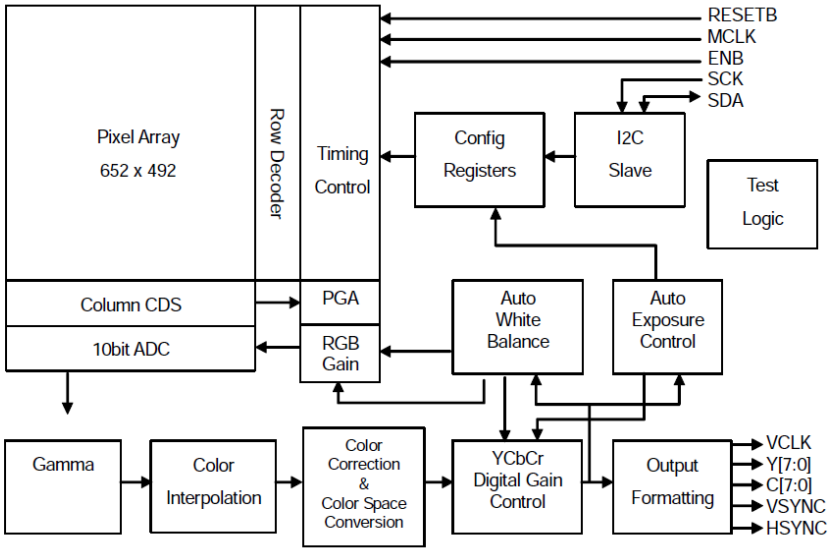


Figura 2.5: Schema a blocchi della camera CMOS

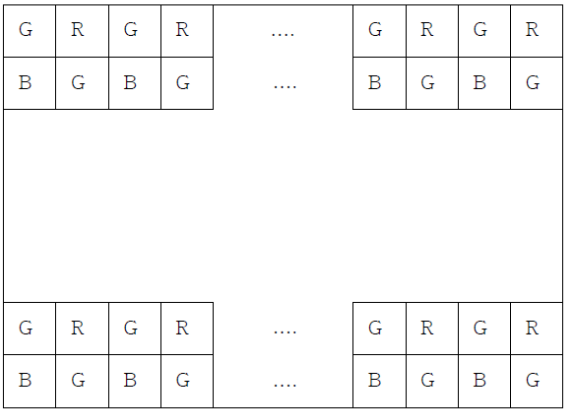


Figura 2.6: Filtro a mosaico per la rilevazione dei colori

- Inizializzazione della camera con un clock a 10Mhz;

```
hv7131gp_status_t hv7131gp_init (void);
```

- Reset della camera;

```
hv7131gp_status_t hv7131gp_init_configure (void);
```

- Configurazione del divisore di tempo;

```
hv7131gp_status_t hv7131gp_configure_time_divisor
(hv7131gp_T_D_Value_t div);
```

- Configurazione della risoluzione;

```
hv7131gp_status_t hv7131gp_configure_resolution
(hv7131gp_R_Value_t res);
```

- Configurazione della finestra di acquisizione;

```
hv7131gp_status_t hv7131gp_window_set
(int width, int height, int x, int y);
```

- Cattura di un frame;

```
hv7131gp_status_t hv7131gp_capture
(uint8_t *image, void(*func)(hv7131gp_status_t));
```

- Modalità sleep;

```
hv7131gp_status_t hv7131gp_set_sleep_status (void);
```

- Riportare la camera in modalità attiva.

```
hv7131gp_status_t hv7131gp_set_active_status (void);
```

2.3 Piattaforma Software

L'ambiente di sviluppo SW è costituito dal supporto del costruttore (Microchip) e dal sistema operativo real-time *ERIKA*.

2.3.1 Supporto Microchip

I componenti principali sono il driver per la gestione del programmatore/debugger *MPLAB ICD2* e il compilatore.

MPLAB IDE

Microchip fornisce un ambiente di sviluppo chiamato *MPLAB IDE*. Questo integra insieme diverse funzionalità: organizzazione dei progetti (gestione dei files), editor per i sorgenti, compilatori, programmazione e debugging con MPLAB ICD2. Per quel che riguarda questo progetto l'IDE è stato utilizzato solo per la gestione dell'MPLAB ICD2 cioè per la fase di programmazione/debugging. Il resto è direttamente gestito nell'ambiente di sviluppo di *ERIKA*.

Compilatore C30

Uno degli strumenti più importanti per la realizzazione di un sistema embedded è il compilatore. *MPLAB C30* è un compilatore ANSI-C compliant che include alcune estensioni nel linguaggio per i microcontrollori della famiglia dsPIC®DSC. È un porting del compilatore GCC della Free Software Foundation e viene chiaramente distribuito insieme ad un assemblatore e al linker. Alcune delle differenze sostanziali rispetto all'ANSI-C sono:

- specifica di attributi per le variabili;
- specifica di attributi per le funzioni;
- definizione di funzioni *inline*;
- associazione di variabili a specifici registri.

Il compilatore è distribuito anche in una versione "student" liberamente scaricabile dal sito di Microchip [7]. Per maggiori informazioni consultare il manuale di riferimento.

2.3.2 *ERIKA Enterprise*

ERIKA (**E**mbedded **R**eal t**I**me **K**ernel **A**rchitecture)[10] è un sistema operativo real-time (RTOS) pensato per soluzioni embedded su piccoli microcontrollori e con una API simile a quelle proposte dal consorzio OSEK/VDX [11]. È distribuito da Evidence s.r.l. [8] in doppia licenza (commerciale e GPL). Il kernel è accompagnato da *RT-Druid* che costituisce un ambiente di sviluppo per il sistema operativo fornendo strumenti per la scrittura, la compilazione e l'analisi dell'applicazione. Per il progetto è utilizzato *ERIKA Enterprise* (EE) con licenza GPL. Alcune delle caratteristiche del sistema operativo sono:

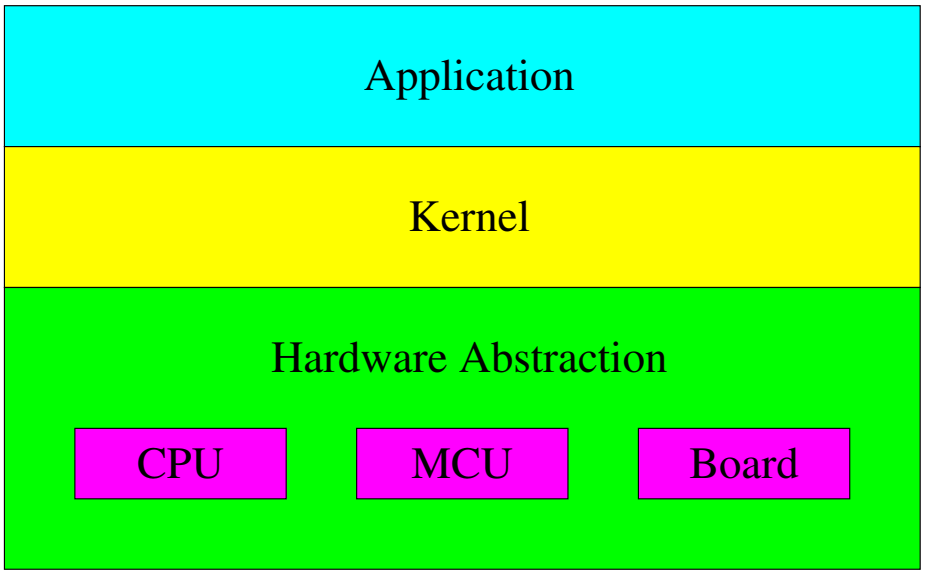


Figura 2.7: Architettura di *ERIKA* RTOS

- quattro classi di conformità OSEK;
- scheduling *Fixed Priority* (FP) e *Earliest Deadline First* (EDF);
- multitasking di tipo preemptive e non-preemptive;
- gestione della condivisione dello stack;
- gestione delle risorse condivise (*Resource*) con *Immediate Priority Ceiling*;
- semafori generalizzati (*Semaphore*);
- attivazione periodica dei task mediante allarmi (*Alarm*).

Architettura del kernel

L'architettura di EE, mostrata in figura 2.7, è costituita da due livelli: *Kernel Layer* e *Hardware Abstraction Layer* (HAL). Il livello del Kernel implementa la gestione dei task e le varie strategie di schedulazione adottate. Esporta per il livello di applicazione una serie di RTOS API per *Task*s, *Alarm*s, *Resource*s e *Semaphore*s. L'HAL si occupa della parte del sistema operativo dipendente dall'hardware, ad esempio la gestione delle interruzioni e delle commutazioni di contesto. Le architetture attualmente supportate sono:

- Microchip dsPIC;
- Atmel AVR5;
- Altera NIOS II;

- ARM7TDMI;
- Tricore;
- Motorola PPC;
- Hitachi H8;
- ST10;

Standard OSEK

Uno degli aspetti più interessanti di *ERIKA* è l'aderenza a quattro classi di conformità OSEK [12]. Lo standard *OSEK* (**O**ffene **S**ysteme und deren schnittstellen für die **E**lektronik im **K**raftfahrzeug - *open system and corresponding interfaces for automotive electronics*) è un progetto congiunto di alcune industrie automobilistiche per lo sviluppo di sistemi di controllo distribuiti nei veicoli. Il consorzio prende il nome di *OSEK/VDX* (VDX **V**ehicle **D**istributed e**X**ecutive) poiché si occupa della definizione di un insieme di API per sistemi operativi real-time (OSEK) e di un sistema per la gestione di rete (VDX). Esempi di applicazioni tipiche dello standard riguardano sistemi di controllo con vincoli temporali molto stretti e orientati a produzione di massa. Ne consegue che OSEK risulta molto orientato all'ottimizzazione (memoria, risorse, prestazioni, ecc). Pertanto le caratteristiche principali sono: scalabilità (supporto per molte architetture HW), portabilità (interfaccia ISO/ANSI-C), configurabilità (tramite *OSEK Implementation Language* - OIL) e allocazione statica (elementi del kernel allocati a tempo di compilazione).

RT-Druid

Come già anticipato *RT-Druid* può essere considerato l'ambiente di sviluppo per EE. È un set di plugin per il framework Eclipse [13](fig. 2.8) ed è composto da una parte per la generazione del codice ed una parte per l'analisi di schedulabilità.
Il *code generator* è disponibile anche in una versione stand-alone indipendente da Eclipse. Si tratta principalmente di un "compilatore" per OIL (dello standard OSEK) che, in base alle specifiche definite nella configurazione del sistema, costruisce la struttura necessaria alla compilazione di *ERIKA* (makefiles, inizializzazione delle strutture dati, ecc). Inoltre fornisce una serie di template project ovvero un insieme di esempi di applicazioni pronte per essere compilate ed eseguite sulla *FLEX* board.

Il plugin per l'analisi della schedulabilità permette di modellizzare, analizzare e simulare il comportamento temporale del sistema embedded che si vuole progettare. È disponibile solo nelle versioni commerciali del sistema operativo.

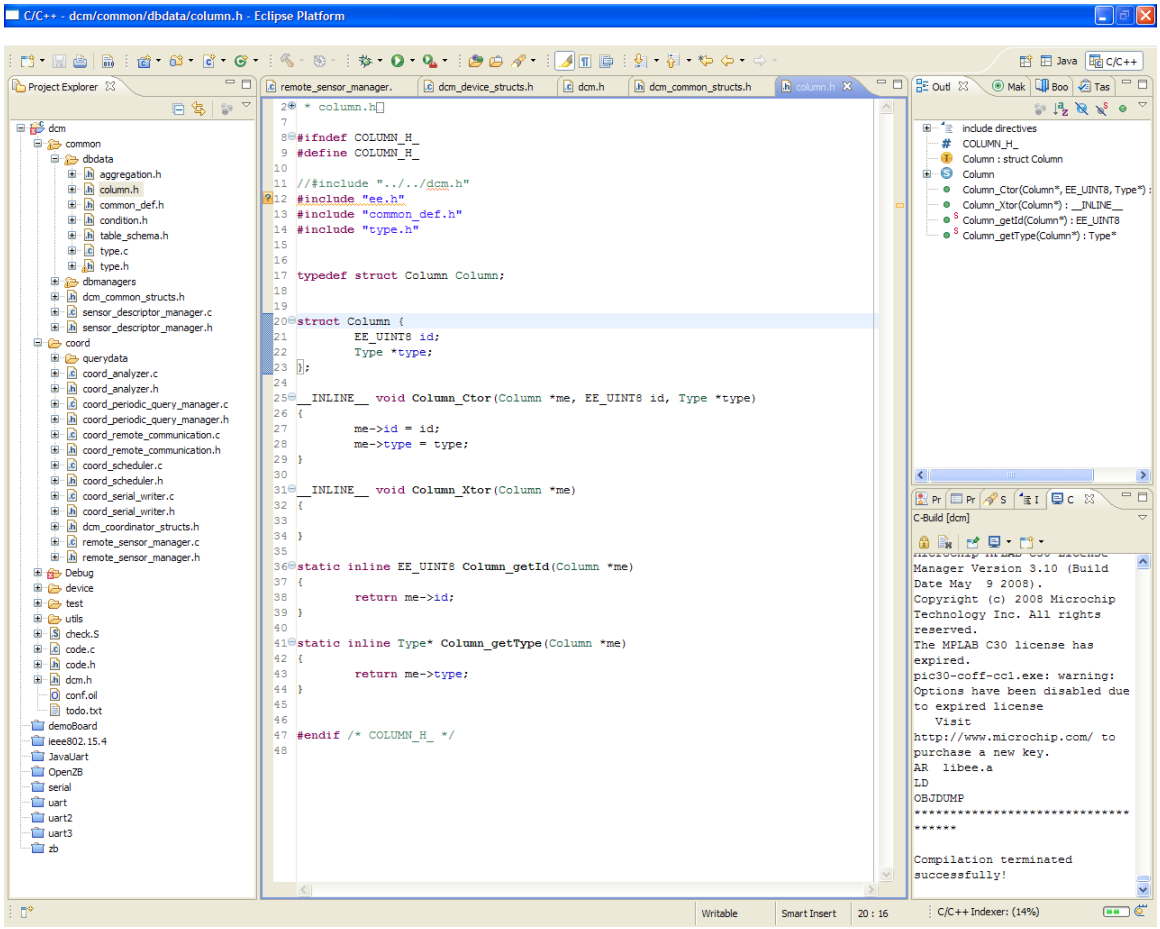


Figura 2.8: Screenshot del framework Eclipse con il plugin RT-Druid

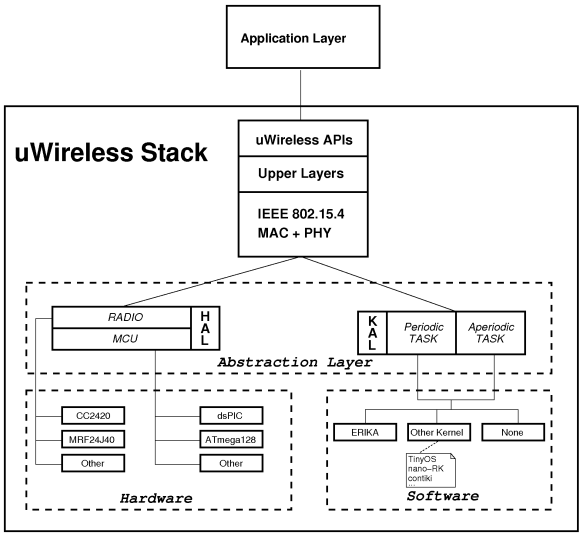


Figura 2.9: Architettura stratificata del sistema di comunicazione μWireless

2.3.3 μWireless

μWireless è un'implementazione (parziale) dello stack 802.15.4 per Erika Enterprise sviluppato dal laboratorio Retis della Scuola Superiore Sant'Anna di Pisa. Esso manca principalmente di un meccanismo per l'associazione dinamica dei dispositivi, ma include il supporto alla modalità beacon-enabled con GTS. Tra le numerose funzionalità esportate al livello applicativo da μWireless si riportano le seguenti:

- possibilità di configurare la durata del superframe e dei timeslot tramite i parametri BO e SO;
- invio in modalità CSMA/CA o GTS;
- possibilità di sincronizzare l'applicazione con gli eventi della rete (ricezione di un frame, inizio e fine del superframe o del singolo time slot, ecc.) tramite la tecnica delle callback;
- possibilità di impostare il payload del beacon frame.

Come si può vedere in figura 2.9, μWireless fa uso delle primitive software messe a disposizione da Erika. Le funzionalità dello stack sono infatti implementate all'interno di appositi task e vengono attivati per mezzo di allarmi periodici.

2.4 Strumenti di debugging

Lo sviluppo del sensore stereo è stato supportato da quattro principali strumenti di debugging. Due di questi, la Flex Demo Board e la coppia MPLAB IDE/MPLAB ICD2 sono stati già presentati (cfr. 2.1.1, cfr. 2.3.1 e cfr. 2.4.1). Nel paragrafo seguente viene presentato il *CC2420 Packet Sniffer*, strumento per l'analisi del traffico di rete ed il DAQ.

2.4.1 DAQ

Il corretto funzionamento di un firmware viene valutato con i classici sistemi di debug, eseguendo nei casi più critici il codice istruzione per istruzione e controllando i valori delle variabili. Nello sviluppo di applicazioni embedded complesse è spesso necessario valutare le prestazioni del firmware confrontandole con un modello simulato presente su PC. Tale operazione si può realizzare con un sistema di acquisizione dati che permetta il salvataggio dei valori generati dal dispositivo embedded su PC [3]. Il sistema di acquisizione utilizzato a tale fine è un *XML-based Data AcQuisition* denominato *DAQ*, sviluppato dal gruppo di lavoro *Wireless Sensor Network* del *Retis Lab* della Scuola Superiore Sant'Anna e distribuito sotto licenza GPL. Il DAQ utilizza le librerie *Apache Xerces C++* e *CERN ROOT* ed è stato recentemente linkato dal sito web del CERN come un'applicazione ROOT-based. L'idea innovativa alla base di questo tool è quella di essere indipendente dal firmware che si intende testare, infatti il DAQ viene inizializzato con un file XML contenente la sorgente dei dati (file, porta seriale, USB, ecc.) ed il loro formato.

Come mostrato in figura 2.10 il file XML viene processato dal metodo *XMLParser* della classe *CXML* per ottenere le informazioni relative alla formattazione dei dati in input con cui inizializzare le strutture *SDef* e *Metadata*. La classe *CSerial* contiene un buffer a basso livello per ricevere i dati in ingresso dal dispositivo embedded e inviarli alla classe *CFormatter* che gestirà la sua serializzazione tramite le classi del software ROOT.

ROOT è un framework standard "de facto" per il data processing in High Energy Physics, tramite la sua classe *TTree* vengono istanziate delle n-tuple in cui ciascuno dei record può avere una struttura più complessa (una classe). Questi *TTree* possono essere serializzati e salvati su disco su contenitori simili a quelli di un filesystem (TFile). I TFile possono essere, in fase di post processing, analizzati attraverso le librerie di *ROOT* utilizzabili su piattaforme Linux e DOS in quanto viene creato all'interno di un TFile un file system indipendente dalla piattaforma.

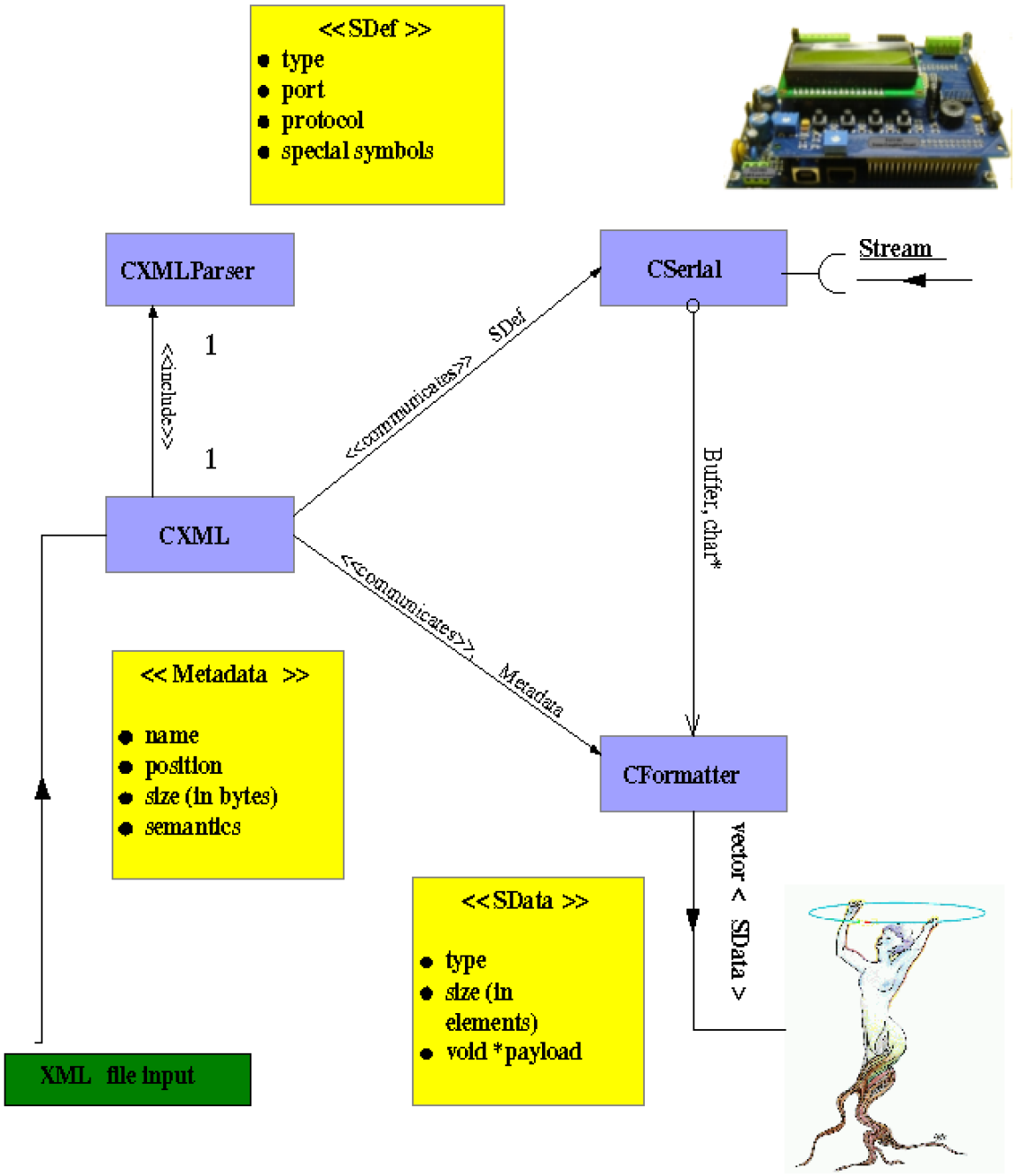


Figura 2.10: Diagramma delle interazioni delle classi del DAQ [3]

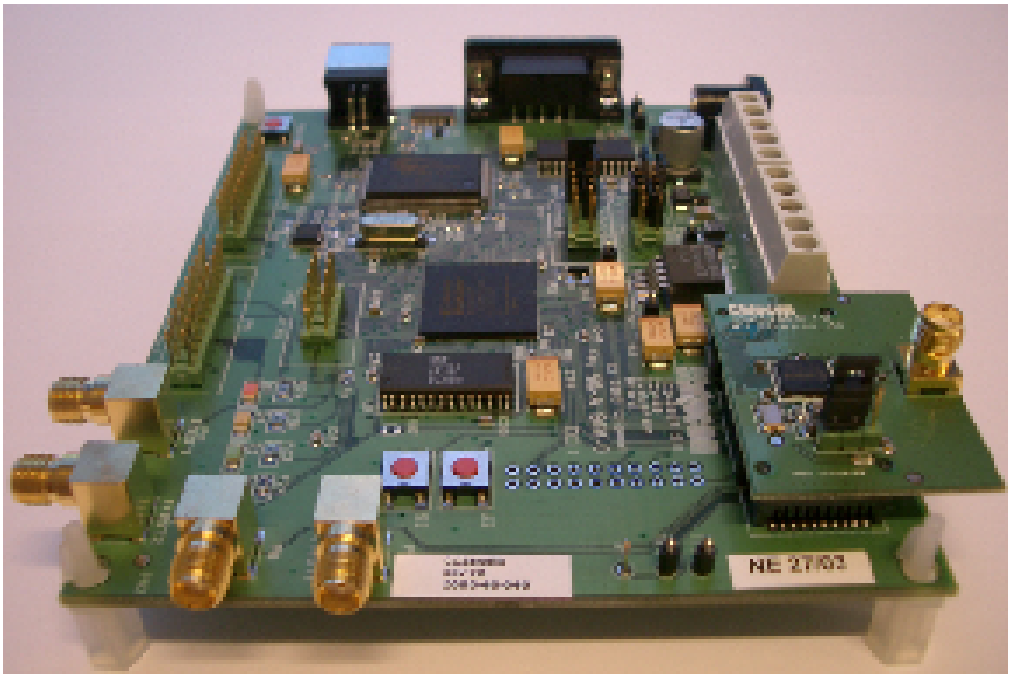


Figura 2.11: La board Texas Instruments CC2420 EB con l'estensione CC2420EM

2.4.2 Chipcon CC2420 Packet Sniffer

L'analizzatore di rete Chipcon CC2420 Packet Sniffer for IEEE 802.15.4 [14] permette di catturare i frame IEEE 802.15.4 per mezzo della scheda Texas Instruments CC2420EB dotata dell'estensione CC2420EM (fig. 2.11).
La sua interfaccia grafica (fig. 2.12) mette a disposizione le seguenti funzionalità:

- elenco dei pacchetti ricevuti con relativi timestamp;
- interpretazione dei pacchetti con evidenziazione dei differenti campi;
- filtraggio dei pacchetti;
- lista dei device presenti nella rete.

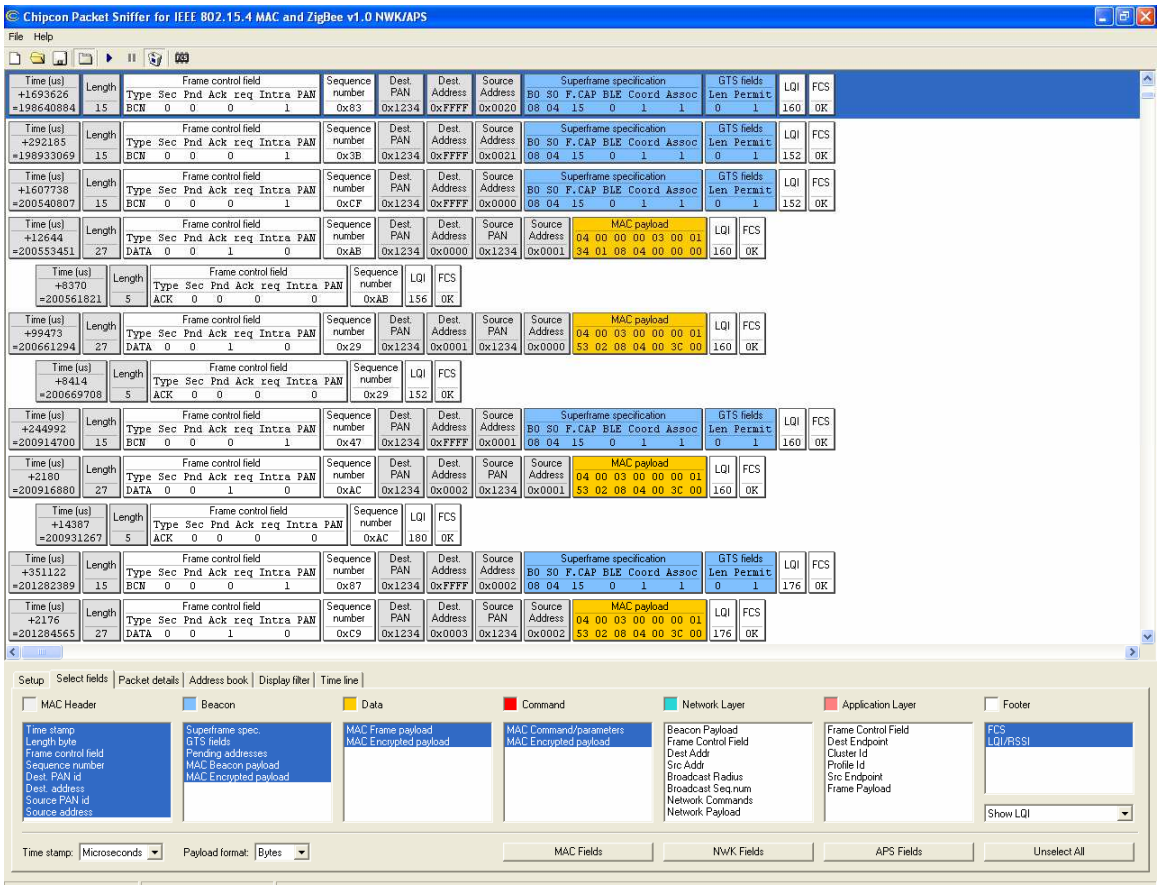


Figura 2.12: Interfaccia del Chipcon CC2420 Packer Sniffer

Capitolo 3

Architettura Hardware

L'intero sistema di stereo visione wireless si divide in due componenti fondamentali, un coordinatore ed una stereo camera connessi dal protocollo μWireless compliant con lo standard IEEE 802.15.4. Il prototipo della stereo camera è stato realizzato con 3 schede *FLEX*, 2 per controllare le telecamere CMOS ed una per elaborare l'immagine stereo e gestire la trasmissione wireless, mentre il coordinatore è stato implementato con una scheda FLEX ed una daughter board a cui è collegato il modulo radio cc2420.

3.1 Sistema di stereovisione

Con il terminte Stereo Camera si indica il sensore di immagini stereo, composto dalle schede Camera Board e Camera Control. In figura 3.1 sono mostrati i componenti base che compongono il dispositivo.

3.1.1 Camera Board

Con il termine Camera Board si indica la scheda FLEX a cui è collegata la telecamera CMOS. La visione stereo è implementata attraverso l'utilizzo di due telecamere e quindi di due Camera Board, collegate alla scheda per l'elaborazione tramite il bus SPI. La camera utilizzata è la Spark Fun SENSE-CCAM, un dispositivo a colori con una risoluzione di 640 x 480 pixel, ma il dsPIC33f a causa dei 30Kb di RAM può gestire la camera con al più una risoluzione di 160 x 120 pixel in scala di grigio. La telecamera operando a 10 Mhz invia un treno di pixel al microcontrollore generando delle interruzioni equidistanziate di 100μs. Con una risoluzione di 160 x 120 la routine di interrupt fine pixel viene richiamata 19200 volte, saturando quasi completamente le risorse del dispositivo. Tale caratteristica non rende possibile il controllo di due telecamere con un solo microcontrollore. Nel testbed si è lasciato alla Camera Board il task di acquisire le immagini demandando alla Camera Control l'algoritmo

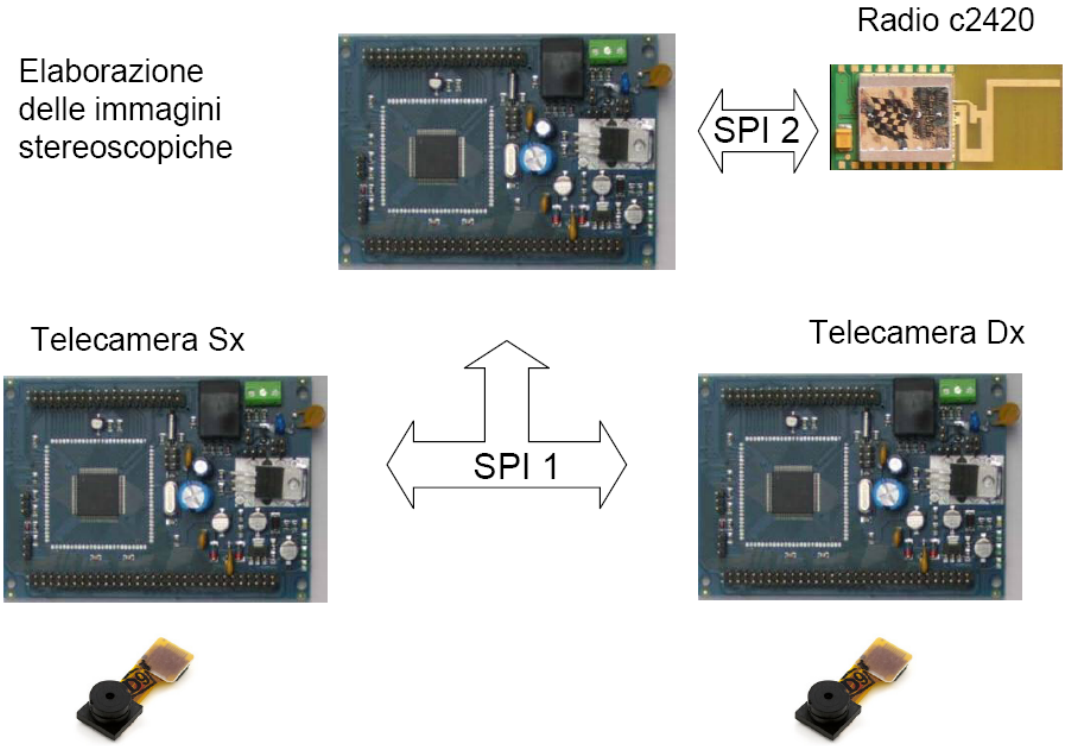


Figura 3.1: Schema implementativo della Stereo Camera

di elaborazione. Lo schema elettrico relativo alla Camera Board è visibile in figura B.1.

3.1.2 Camera Control

Con il termine Camera Control si indica la scheda *FLEX* principale che ha il compito di gestire le 2 schede Camera Board tramite la periferica SPI1 del dsPIC. La scheda Camera Control legge i frame Left e Right acquisiti dalle Camera Board, genera la mappa di disparità e gestisce la sua trasmissione alla scheda Stereo Coordinator tramite il protocollo μWireless. La scheda Camera Control è collegata alle schede Camera Board, con la SPI2 controlla il modulo radio cc2420, mentre con la UART1 può essere collegata al PC per eseguire il data logging su supporto permanente (modalità "debug"). Utilizzando la scheda Camera Control attraverso la porta seriale del PC, è possibile visualizzare anche le immagini delle telecamere. Questa funzionalità è stata sfruttata in fase di test e debug HW/SW. Le immagini stereo acquisite sul computer possono essere elaborate con l'algoritmo originale di stereovisione per effettuare dei

confronti con la versione embedded. Lo schema elettrico relativo alla Camera Control è visibile in figura B.2.

3.2 Stereo Coordinator

La scheda denominata Stereo Coordinator in figura 3.2 è stata realizzata utilizzando la *daughter board* ed il modulo radio cc2420, ed è destinata a svolgere la funzione di Base Station. Essa gestisce la comunicazione wireless con il sistema di stereo visione ed invia i frame ricevuti al computer tramite la porta seriale RS-232 utilizzando la periferica del dsPIC UART2.

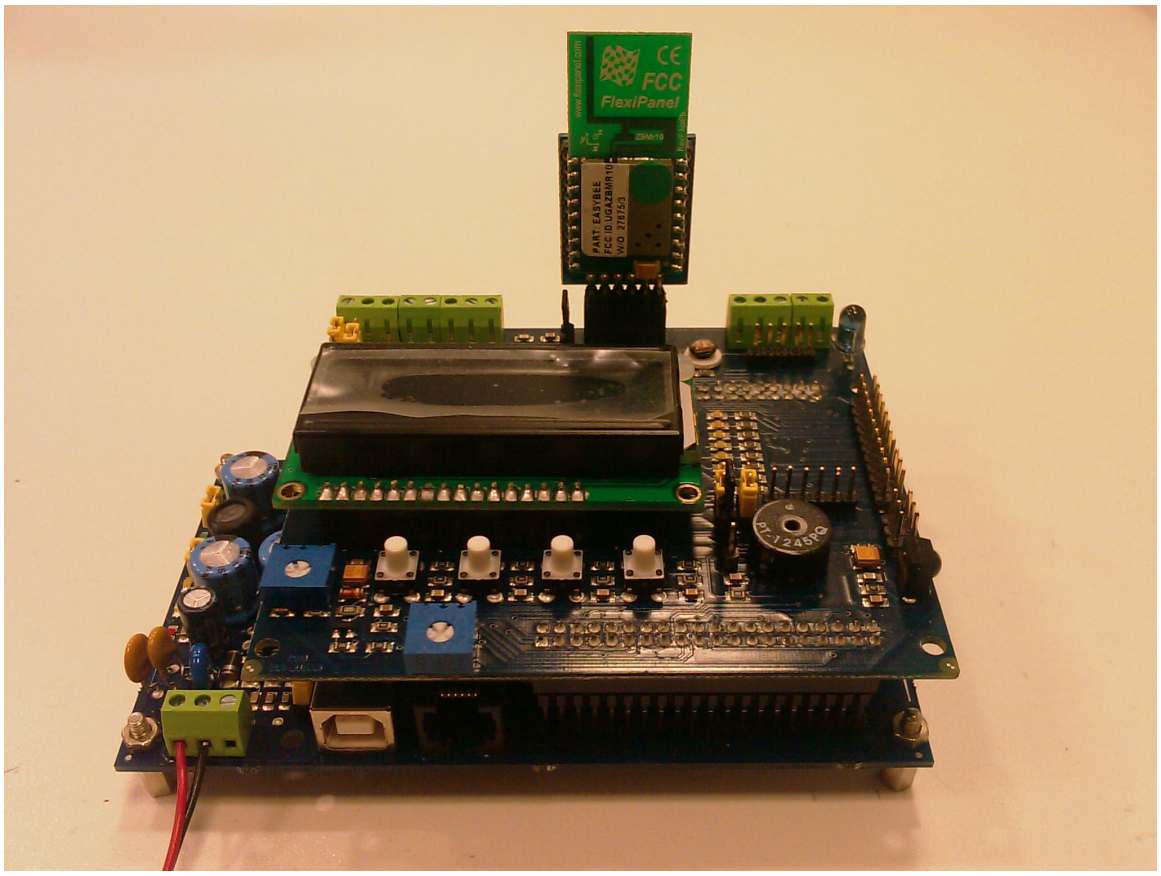


Figura 3.2: Stereo Coordinator

Capitolo 4

Implementazione

In questo capitolo sono discussi aspetti dell'implementazione del sistema di stereovisione wireless, le librerie utilizzate, i vincoli fondamentali.

4.1 Porting dell'algoritmo di stereovisione su dsPIC

L'algoritmo di stereovisione utilizzato non può essere implementato sul dsPIc senza opportune modifiche orientate alla riduzione della RAM richiesta. L'algoritmo *Depth Discontinuities by Pixel to Pixel Stereo* è stato sviluppato e testato dall'autore su una workstation; accetta in input coppie di immagini di dimensioni qualsiasi sfruttando l'allocazione dinamica della memoria. Il processo di *porting* richiede le seguenti semplificazioni:

1 - eliminazione dell'allocazione dinamica della memoria;

2 - eliminazione del calcolo delle discontinuità di profondità;

3 - riduzione del footprint e del numero di strutture allocate in memoria principale.

4.1.1 Eliminazione dell'allocazione dinamica della memoria

La stereocamera processa coppie di immagini stereo con la dimensione predefinita di 69 x 80 pixel. La semplificazione inserita utilizzando un singolo formato permette di inizializzare le dimensioni dei vettori rendendo inutile l'utilizzo di funzioni come *malloc* e *free*.

4.1.2 Eliminazione del calcolo delle discontinuità di profondità

L'algoritmo di stereovisione a partire da una coppia di immagini stereo genera due frame di output, una mappa di disparità contenente l'informazione sulla profondità ed una mappa di discontinuità contenente i contorni degli oggetti presenti nella scena. Il calcolo della mappa di discontinuità non è stato implementato perché non è richiesta tale funzionalità, di conseguenza sono state eliminate nella fase di porting le funzioni e le strutture in memoria non più utilizzate.

4.1.3 Riduzione delle strutture allocate in memoria

Per generare la mappa di disparità l'algoritmo necessita di diverse strutture dati per eseguire i calcoli di stereovisione. Nella funzione

- Funzione per il calcolo dei *match*

```
void matchScanlines(uchar *imgL, uchar *imgR, uchar *disparity_map)
```

sono presenti due array di tipo *integer*

```
  int pie_y[1495];    /*(g_cols+g_slop)*(g_maxdisp + 1)
points to the immediately */
  int pie_d[1495];    /*(g_cols+g_slop)*(g_maxdisp + 1)
preceding match */
```

utilizzati per la ricerca dei *match*, i valori da essi contenuti sono quasi sempre minori di 255 ($2^8 - 1$), in rari casi contengono FIRST_MATCH che è definito come 65535 ($2^{16} - 1$). La soluzione utilizzata è quella di ridefinire lo stato di FIRST_MATCH uguale a 255 e di dichiarare *pie_y* e *pie_d* come unsigned char, con il conseguente aumento del numero delle occorrenze.

```
  uchar pie_y[1425];    /* points to the immediately */
  uchar pie_d[1425];    /*    preceding match */
```

La memoria richiesta per i vettori originali è di 5700 byte, la semplificazione introduce un risparmio del 50% equivalente a 2850 byte. La funzione *postprocess* originale contiene diverse funzioni necessarie a migliorare il calcolo della mappa di disparità, ma la RAM limitata del dsPIC ha imposto di eliminare quelle funzioni non indispensabili che richiedono una grossa quantità di memoria. Le funzioni eliminate sono le seguenti:

1 - Calcolo dei gradienti di intensità

```
static void compute_igxy(uchar *imgL, uchar *igx, uchar *igy)
```

2 - Rimuove i gradienti di intensità isolati lungo la direzione Y

```
static void removeIsolatedPixelsX(uchar *array, int len)
```

3 - Propaga le regioni affidabili nella direzione Y

```
static void compute_igyd(uchar *igy, int *igyd)
static void computeReliabilitiesY(uchar *disp_map,
  uint *reliability_map)
static void propagateY( uchar *disp_map, uchar *igy,
int th_moderately_reliable,
int th_slightly_reliable,
uint *reliability_map,
int *igyd, int max_attraction)
```

4 - Propaga le regioni affidabili nella direzione X

```
static void compute_igxd(uchar *igx, int *igxd)
static void computeReliabilitiesX(uchar *disp_map,
  uint *reliability_map)
static void propagateX(uchar *disp_map, uchar *igx,
       int th_moderately_reliable,
       int th_slightly_reliable,
       uint *reliability_map,
       int *igxd, int max_attraction)
```

Le funzioni eliminate richiedono nel segmento di memoria principale le seguenti strutture dati:

- unsigned int reliability_map[5520]
- unsigned char igx[5520]
- unsigned char igy[5520]
- int igyd[5520]
- int igxd[5520]

ed occupano un totale di 44160 byte. Questa richiesta di risorse renderebbe impossibile il porting sul dsPIC. Avendo operato queste semplificazioni, il footprint richiesto occupa 24 Kb per un totale di 30720 byte presenti sul dispositivo. Senza applicare le semplificazioni proposte l'algoritmo richiederebbe 70 Kb di RAM.

4.2 Protocollo per la trasmissione wireless

La mappa di disparità è per definizione un'immagine grande come quelle di partenza (69 x 80 pixel), e ogni suo pixel è definito su un dominio di 14 valori ($2^4 = 0.5$ byte). Si è scelto di implementare una compressione senza perdite, rappresentando due pixel con un solo byte si ottiene un risparmio del 50% di memoria che equivale a 2760 byte. La compressione del vettore di disparità è stata implementata con le seguenti macro:

```
#define BIT4_WRITE(byte1, select, val) \
 *(byte1) = ((*(byte1))&(~(0x0F << (4 * (select))))) |
    ((val << (4 * (select))) & (0x0F << (4 * (select))))

#define BIT4_READ(byte1, select, val) \
 *(byte1) = ((((val)>> (4 * (select)) ) & 0x0F  ))
```

la prima serve a scrivere due pixel in un solo byte ed è utilizzata nella stereo camera, la seconda realizza l'operazione inversa ed è implementata nella Base Station. L'implementazione dello stack IEEE 802.15.4 permette di inviare pacchetti con una dimensione massima di 118 byte, quindi un'immagine deve essere divisa in pacchetti prima di essere inviata. μWireless non divide automaticamente i vettori che occupano più di 118 byte in pacchetti, tale funzionalità è stata implementata nel presente lavoro. Considerando che la mappa di disparità compressa richiede 2760 byte essa viene divisa in 25 pacchetti da 110 byte ed uno da 10.

$$numb_packet = \lceil \frac{2760}{110} \rceil = 26$$

4.2.1 Meccanismo di gestione dei pacchetti tra coordinator e device

Il dispositivo Stereo Coordinator invia periodicamente alla stereo camera la richiesta di un determinato pacchetto inviando nel *MACpayload* il numero richiesto ed attiva un *timeout*. Se si riceve il pacchetto atteso prima dello scadere del *timeout* si procede alla richiesta del pacchetto successivo, nel caso in cui questa situazione non si verifica perché non è stato ricevuto nessun pacchetto oppure quello ricevuto non è quello atteso viene ripetuta la richiesta dal Task Timeout. Dopo aver completato la lettura di tutti i pacchetti l'immagine viene inviata tramite la porta seriale al computer. Il meccanismo del *timeout* è implementato con un task periodico con periodo di 50 ms attivato al momento della trasmissione. La stereo camera genera pacchetti di risposta al coordinator, inserendo nel primo byte del *MACpayload* il numero della richiesta.

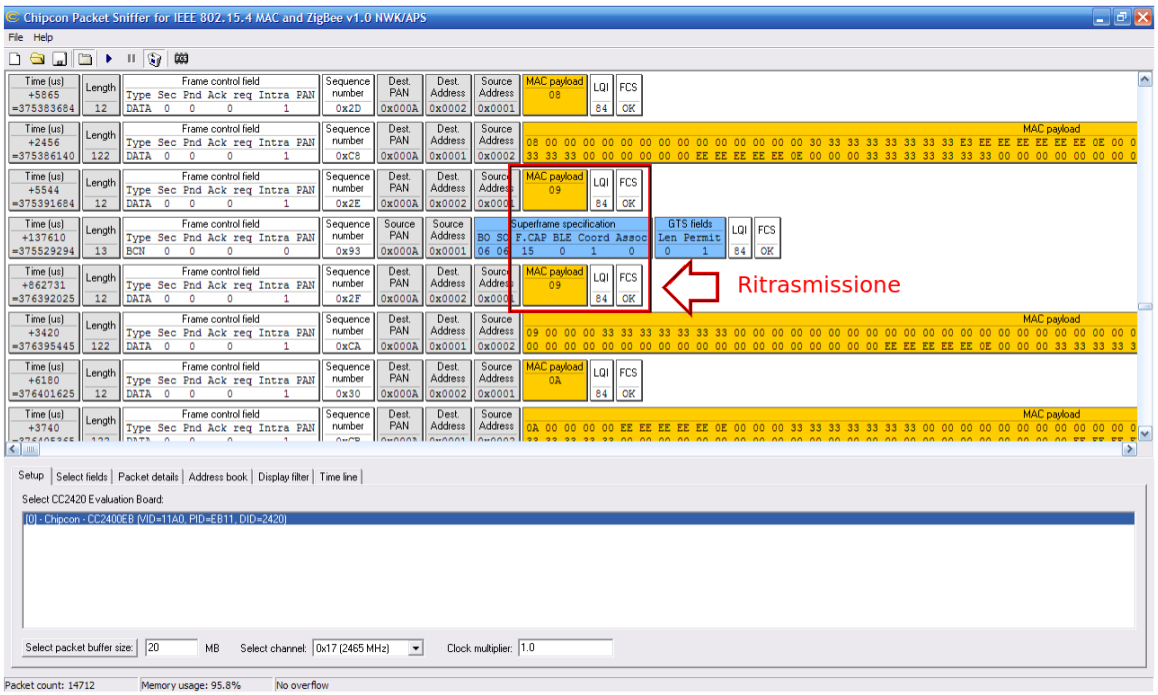


Figura 4.1: Esempio di ritrasmissione

In assenza di richieste da parte del coordinatore non vengono trasmessi pacchetti. Gli esempi di codice relativi alla gestione del protocollo dal lato coordinator e device sono mostrati nell'appendice A.1 e A.2.

4.3 Postprocess: elaborazione della disparità su PC

Le immagini inviate dal dispositivo embedded al computer tramite porta seriale sono acquisite e memorizzate sul disco utilizzando il tool di data acquisition *DAQ* 2.4.1. I dati acquisiti sono analizzati dal programma *LoopDaqEvent_Stereo* per confrontare le prestazioni dell'elaborazione stereo su dispositivo embedded con quella su PC. Il programma *P2P* che implementa l'algoritmo *Depth Discontinuities by Pixel to Pixel Stereo* è stato modificato per non utilizzare in input dei file, ma per operare con le immagini estratte dai TFile di ROOT. Il programma di analisi visualizza sullo schermo 4 finestre contenenti le immagini stereo, la disparità calcolata dalla scheda *FLEX* e quella su PC, necessarie ad effettuare un confronto qualitativo soggettivo tra le due elaborazioni. La visualizzazione avviene frame per frame ed è possibile

rivedere l'intera sequenza acquisita. Il confronto oggettivo viene effettuato calcolando la distanza in norma ed il PSNR della disparità semplificata da quella completa. Il programma *LoopDaqEvent_Stereo* salva su disco i grafici del PSNR e della norma per ogni sequenza di acquisizione.

Capitolo 5

Risultati sperimentali

5.1 Introduzione

In questo capitolo sono descritte le modalità con cui sono state effettuate le misure sul funzionamento del prototipo, sia dal punto di vista della qualità dell'elaborazione e la velocità di trasmissione. È descritto il setup utilizzato nelle misure e gli strumenti software ed hardware utilizzati.

5.2 Qualità della disparità elaborata embedded

Per valutare la qualità dell'elaborazione della stereovisione è necessario effettuare il confronto con un immagine di riferimento. Questa immagine è ottenuta elaborando la mappa di disparità su computer, utilizzando l'algoritmo completo. Il confronto tra la mappa di disparità embedded e su PC può essere affrontato in modo oggettivo calcolando la *norma* ed il *PSNR*. Il calcolo della norma è implementato nel programma di analisi LoopDAQEvent_Stereo con la seguente formula che realizza la distanza tra due matrici:

$$Norma = \frac{1}{numb_pixel}\sqrt{\sum_{k=1}^{numb_pixel} \|disp_i - dispE_i\|^2}$$

mentre per il calcolo del PSNR si utilizza la seguente formula:

$$PSNR = 10\log\left(\frac{MAX^2}{MSE}\right)$$

dove MAX è il valore massimo ammesso per un pixel nell'immagine e vale:

$$MAX = 14^2$$

mentre l'Mean Square Error è così definito:

$$MSE = \frac{1}{numb_pixel} \sum_{k=1}^{numb_pixel} \|disp_i - dispE_i\|^2$$

Le misure sono eseguite in 4 condizioni diverse, sia per le distanze degli oggetti nella scena che per la loro complessità. Gli oggetti più vicini sono individuati sulla mappa di disparità con zone più luminose. Sono state effettuate le seguenti prove sia all'interno che all'esterno degli edifici:

- **Aula con persona che entra nella scena (fig. 5.1).** Nella scena l'oggetto più vicino è una persona e tale situazione è evidenziata da una zona luminosa che ricalca la sagoma.

- **Sedia in una stanza (fig. 5.2)**. Nella ripresa stereo sono presenti due disturbi, il muro in background è illuminato in modo non uniforme e l'immagine relativa alla camera sinistra presenta un disturbo di natura elettromagnetica evidente nella zona centrale. La mappa di disparità è influenzata negativamente da tali disturbi, infatti la disparità relativa alla sedia è collegata a due aloni luminosi sopra ed a destra. Questa sensibilità è visibile sia sull'elaborazione embedded che sul PC.

- **Ripresa di una scena esterna (fig. 5.3).** Dalla prova risulta evidente che l'algoritmo di stereovisione su PC si comporta meglio della versione semplificata per embedded, infatti individua correttamente la posizione dell'edificio.

- **Oggetti su una scrivania (fig. 5.4).** Nella parte destra della mappa di disparità è evidente la presenza di un oggetto vicino, mentre nella sinistra la zona più chiara è prodotta da una bottiglia occlusa alla camera destra.

Nella tabella 5.2 sono riportati i valori medi della Norma e del PSNR frame per ogni prova.

Prova numero	Norma	PSNR (dB)
1	0.067	9.08
2	0.049	12.45
3	0.067	9.64
4	0.079	13.64

5.3 Velocità di trasmissione

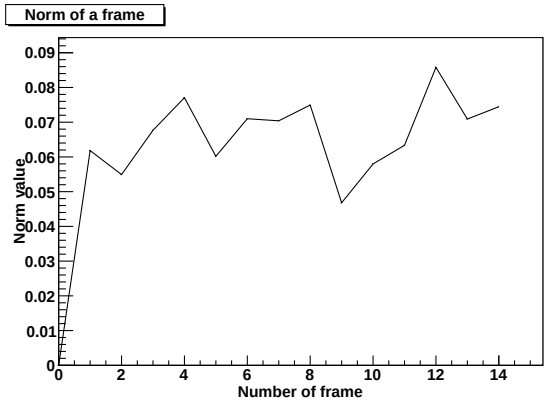
La velocità di trasmissione dei frame tra la stereocamera e la base station può essere valutata utilizzando uno sniffer per il protocollo 802.15.4. In assenza di disturbi ed ostacoli la trasmissione non raggiunge la velocità indicate dallo standard a causa delle limitazioni del transceiver, dell'overhead del protocollo μWireless e in ultimo l'inserimento del *timeout* che introduce un periodo di inattività durante le ritrasmissioni. Considerando la velocità massima dello standard IEEE 802.15.4 che è di 250 kbit/s, la trasmissione di un frame richiede 84 ms. Lo stack μWireless introduce un overhead di 11 byte per ogni pacchetto inviato, che riduce la velocità di trasmissione a 95 ms Dalle misure si evidenzia che l'introduzzione del *timeout* ha un effetto dominante sulla riduzione della velocità complessiva di trasmissione, in presenza di ripetute ritrasmissioni. Il tempo medio necessario a trasmettere un frame di disparità con un *timeout* di 50 ms è di 300 ms. La misura sui tempi di trasmissione è stata effettuata su un numero di 1000 pacchetti, il risultato è stato riportato in figura 5.5. Considerando che il tempo medio per l'invio di un frame è di 300ms ci si aspetta di trovare un picco dovuto ai frame che hanno subito una sola ritrasmissione centrato a 350ms, realizzando il "fit" dell'istogramma si ottiene sulla destra del picco massimo una gaussiana centrata in 326 ms, che è compatibile con il fit gaussiano dell'istogramma centrato a 326 ms con una semilarghezza di circa 30 ms (corrisponde a trovare il valore teorico nell'intervallo di confidenza di 1 δ). L'83% dei frame viene trasmesso con il minimo numero di pacchetti, il 15% subisce una sola ritrasmissione e solo il 2% subisce due ritrasmissioni.

(a) Sinistra

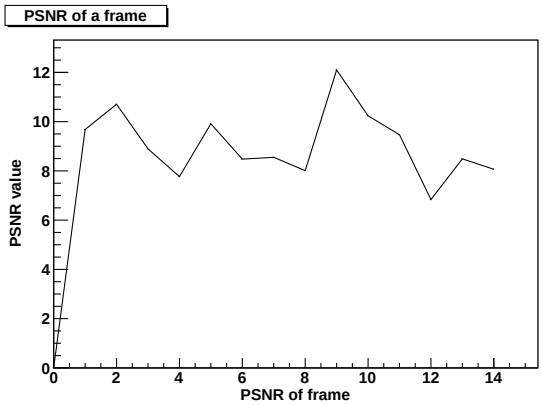
(b) Destra

(c) Embedded

(d) PC



(e) Norma



(f) PSNR

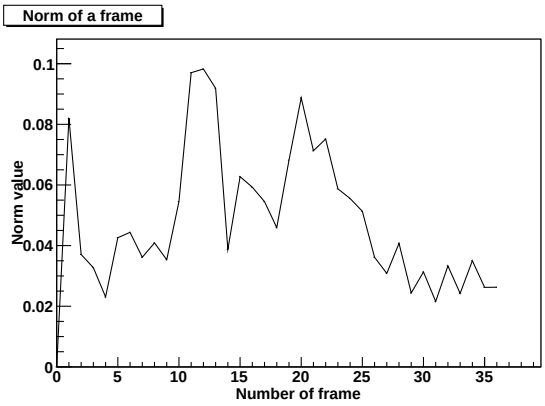
Figura 5.1: Immagini e misure prova 1

(a) Sinistra

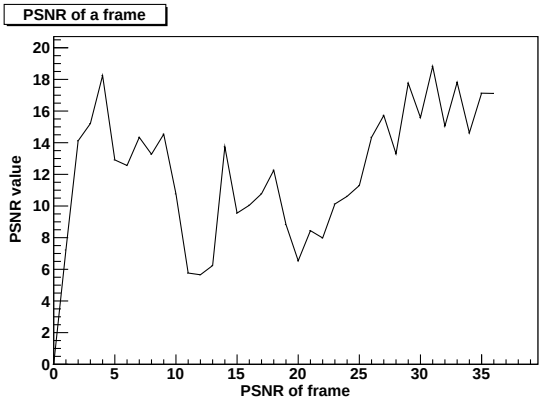
(b) Destra

(c) Embedded

(d) PC



(e) Norma



(f) PSNR

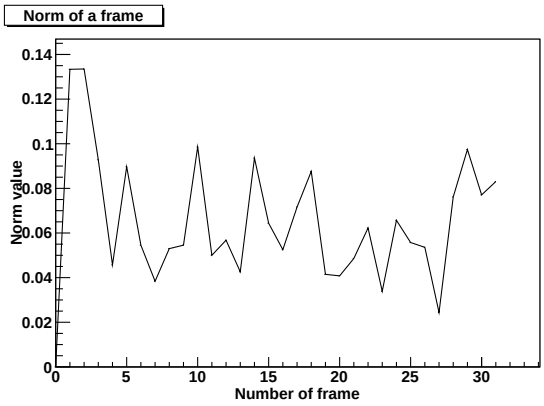
Figura 5.2: Immagini e misure prova 2

(a) Sinistra

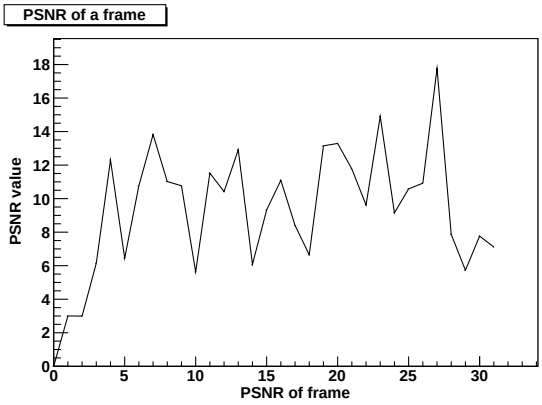
(b) Destra

(c) Embedded

(d) PC



(e) Norma



(f) PSNR

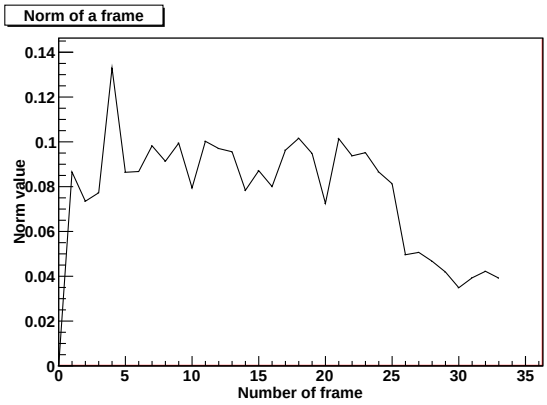
Figura 5.3: Immagini e misure prova 3

(a) Sinistra

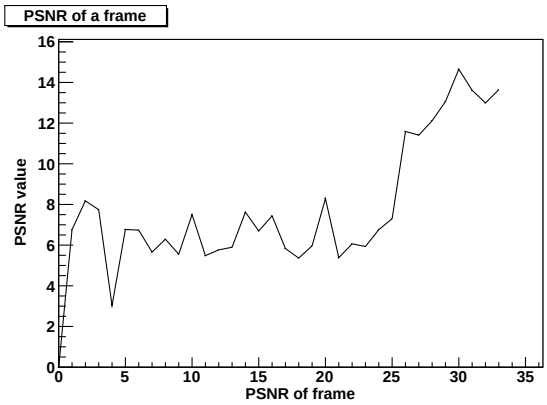
(b) Destra

(c) Embedded

(d) PC



(e) Norma



(f) PSNR

Figura 5.4: Immagini e misure prova 4

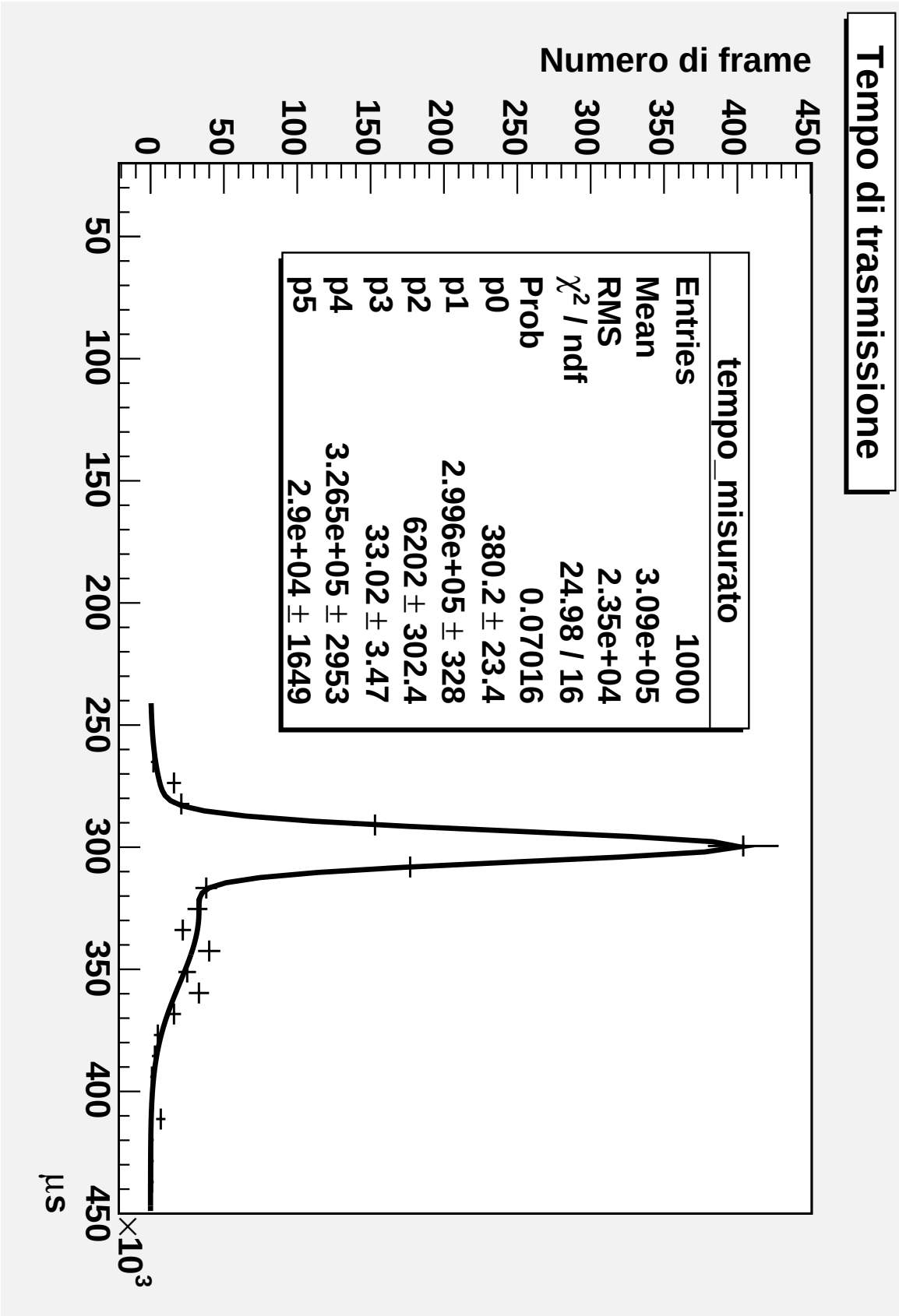


Figura 5.5: Tempi per trasmettere un frame

Conclusioni

In questa tesi è stata presentata l'implementazione di un sistema di stereovisione wireless, che permette di rilevare la distanza di oggetti con diversi livelli di luminosità. L'idea alla base di questo lavoro è quella di realizzare tale sistema con hardware minimale, utilizzando quattro schede di sviluppo equipaggiate con microcontrollore: tre per il sensore ed una per la base station. L'algoritmo di partenza è stato semplificato e testato su PC per verificare il suo funzionamento, ed è stato poi implementato su dsPIC®. Il sistema di stereovisione è stato interfacciato con il programma di analisi LoopDAQEvent_Stereo, che si basa sul framework ROOT, per realizzare delle misure e analizzare le prestazioni qualitative ed oggettive del sistema.
Dalle prove sperimentali emerge che il sistema riesce ad individuare le sagome degli oggetti nella scena, come persone ed edifici, ma entrambe le elaborazioni, sia quella embedded che quella su PC subiscono negativamente gli effetti del rumore presente sulle immagini. La procedura di pacchettizzazione implementata con μWireless permette di trasmettere alla base station ad una velocità limite di 3 fps. Nel presente lavoro sono emerse delle limitazioni di funzionamento imputabili all'hardware. In un eventuale sviluppo futuro si potrebbero apportare i seguenti miglioramenti: sviluppare il sistema sostituendo il dsPIC®con un dispositivo più potente e con una maggiore dotazione di memoria RAM, sviluppare un circuito stampato specifico con criteri di compatibilità elettromagnetica per rendere il sensore immune ai disturbi, e infine sviluppare delle politiche di risparmio energetico indispensabili nelle applicazioni wireless. Il progetto verrà inoltre pubblicizzato ed il codice distribuito su licenza LGPL attraverso il sito web di Evidence srl.

Appendice A

Gestione del protocollo wireless

A.1 Task lato coordinator

```
/* ----------------------------------------------------
|  Timeout Task                                        |
|                                                      |
 ----------------------------------------------------- */
TASK(Timeout)
{
  ActivateTask(BASIC_SEND);
}
/* ----------------------------------------------------
|  Sen Task                                            |
|                                                      |
 ----------------------------------------------------- */
TASK(BASIC_SEND)
{
  static char tx_led = 0;
  packet[0]=packet_number;
  #ifdef __USE_DEMOBOARD__
  if (tx_led) {
    tx_led = 0;
    EE_led_6_off();
    EE_led_7_on();
  } else {
    tx_led = 1;
    EE_led_6_on();
    EE_led_7_off();
```

```
  }
  #endif /* __USE_DEMOBOARD__ */
  uwl_simple154_send(packet, 1, BASIC_DEVICE_ADDR, 0);
  packet_number_old=packet_number;
  if (Iftimeout_on>0){
    CancelAlarm(Timeout_ALARM);
  }
  SetRelAlarm(Timeout_ALARM, TIMEOUT, 0); /* Start Timeout */
  Iftimeout_on=1;
}
/* ----------------------------------------------------
|  Receive Task                                        |
|                                                      |
 ---------------------------------------------------- */
TASK(BASIC_RECEIVE)
{
  static char rx_led = 0;
  int packet_index=0;
  #ifdef __USE_DEMOBOARD__
  if (rx_led) {
    rx_led = 0;
    EE_led_4_off();
    EE_led_5_on();
  } else {
    rx_led = 1;
    EE_led_4_on();
    EE_led_5_off();
  }
  lcd_puts("\nRicezione");
  #endif /* __USE_DEMOBOARD__ */
  EE_led_3_on();
  if ((packet_number_old)==received_packet[0]){
    //se si riceve il pacchetto atteso
    CancelAlarm(Timeout_ALARM); /* Stop Timeout */
    Iftimeout_on=0;
    if (received_packet[0]<26) {
      for (packet_index=1; packet_index<111; packet_index++)
      {
        disparity_map1[110*(int)received_packet[0]+
        packet_index-1]=received_packet[packet_index];
```

```
      }
    }
    if ((packet_number)>25) {
      for (packet_index=1; packet_index<11; packet_index++)
      {
        disparity_map1[110*26+packet_index-1]=
        received_packet[packet_index];
      }
      for (ff=0; ff<5520; ff++){
        BIT4_READ( disp_output, ff % 2 , disparity_map1[ff / 2]);
        disparity_map_out[ff]=disp_output[0];
      }
      pc_serial_write(disparity_map_out, 5520);
    }
    packet_number++;
    if (packet_number>26)
      {
        packet_number=0;
      }
  ActivateTask(BASIC_SEND);
  }
}
```

A.2 Task lato device

```
/* ----------------------------------------------------
|  Rx callback function:                               |
|  Called by the network stack when data have been     |
|  received.                                           |
 ---------------------------------------------------- */

static void rx_data(int8_t status, uint8_t *data, uint8_t len, uint16_t addr)
{
  memset(received_packet, 0, UWL_aMaxMACPayloadSize);
  memcpy(received_packet, data, len);
  received_packet_length = len;
  received_packet_sender = addr;
  ActivateTask(BASIC_RECEIVE);
}
```

```
/* ----------------------------------------------------
|  Receive Task                                        |
|                                                      |
 ---------------------------------------------------- */
TASK(BASIC_RECEIVE)
{
  static int led_status = 0;
  if (led_status) {
    /* if the led is on, turn it off */
    EE_led_off();
    led_status = 0;
  } else {
    /* if the led is off, turn it on */
    EE_led_on();
    led_status = 1;
  }
  int packet_index=0;
  if ( received_packet[0]<26) {
    for (packet_index=1; packet_index<111; packet_index++)
      {
        packet[packet_index]=
        disparity_map1[110*(int)received_packet[0]+packet_index-1];
      }
  } else {
    for (packet_index=1; packet_index<11; packet_index++)
        packet[packet_index]= disparity_map1[110*25+packet_index-1];
    ActivateTask(myTask);
  }
  packet[0]=received_packet[0];
  uwl_simple154_send(packet, packet_index , BASIC_COORD_ADDR, 0);
}
```

Appendice B

Schematici del sistema di Stereovisione

B.1 Schematico Camera Board

B.2 Schematico Camera Control

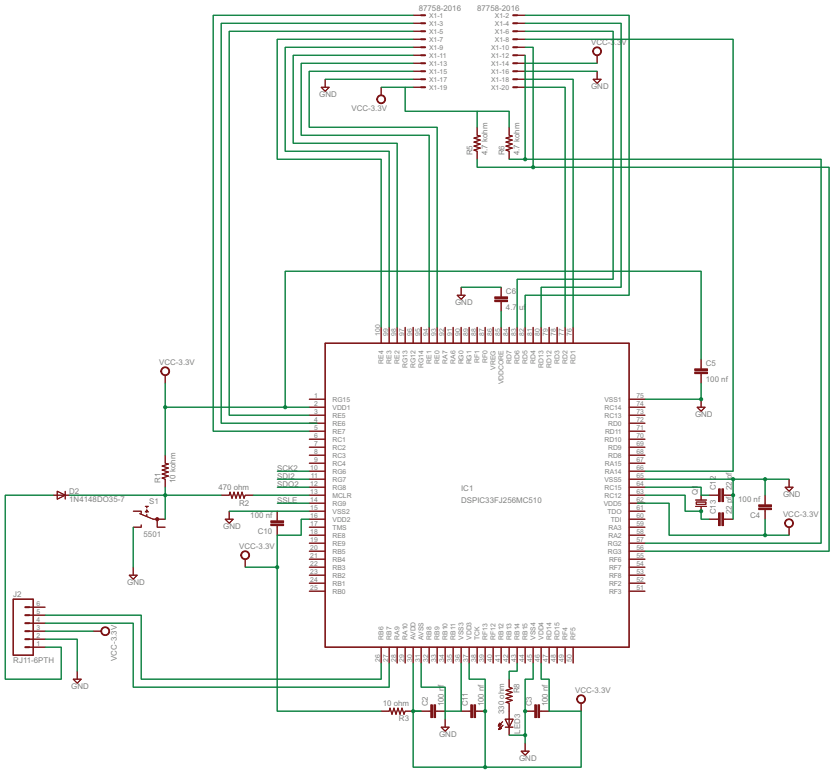


Figura B.1: Schematico della scheda Camera Board

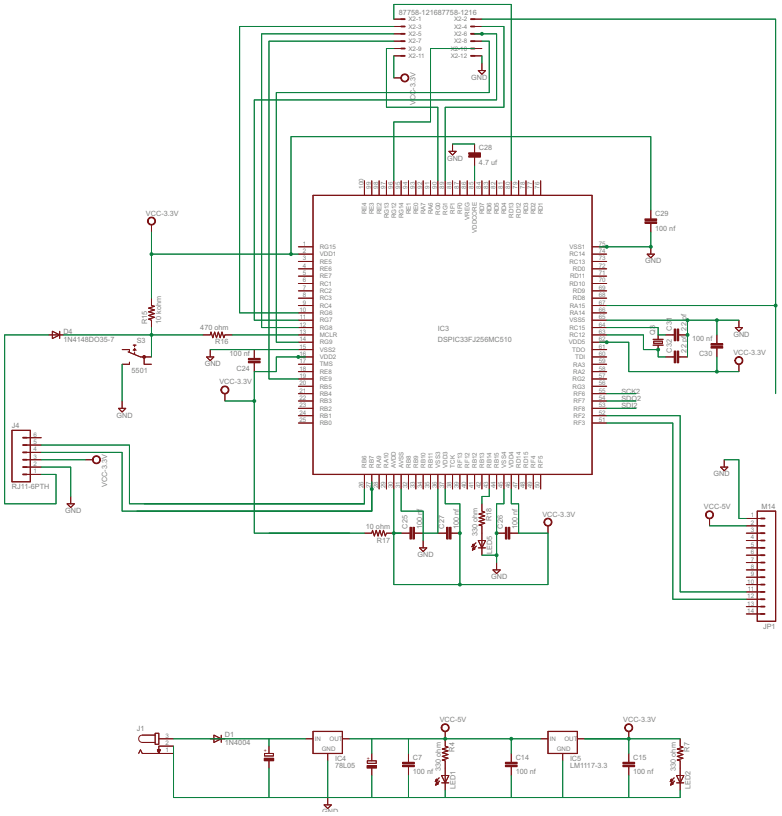


Figura B.2: Schematico della scheda Camera Control

Glossario

Centro focale: punto di incrocio delle linee di proiezione dei punti della scena.

Distanza focale: distanza tra il centro focale ed il piano immagine.

Mappa di disparità: immagine che rappresenta la differenza tra quelle acquisite dalle telecamere e che permette di estrapolare l'informazione relativa alla distanza degli oggetti.

Occlusione: porzione di un'immagine visibile da una sola delle due telecamere stereo.

Rettificazione epipolare: metodo per trasformare le immagini in modo che appaiano come se acquisite da un sistema stereo standard con piani immagine paralleli e sistemi di riferimento allineati.

Stereovisione: tecnica che permette la ricostruzione della struttura tridimensionale della scena osservata da due o più telecamere; si realizza tramite una triangolazione che mette in relazione le proiezioni di un punto della scena sui due (o più) piani immagine delle telecamere.

Triangolazione: procedimento che consente la determinazione indiretta della distanza di un punto utilizzando le proprietà dei triangoli.

Bibliografia

[1] Pinhole camera wiki. http://en.wikipedia.org/wiki/Pinhole_camera.

[2] Mario Muratori. La ripresa stereoscopica. http://www.crit.rai.it.

[3] Paolo Pagano, Claudio Salvadori, and Christian Nastasi. DAQ: a ROOT based Data AcQuisition platform for debugging embedded devices . http://www.evidence.eu.com/content/view/336/266/.

[4] Sebastiano Battiato. Computer Vision, Modulo 2, Stereo-vision. http://www.dmi.unict.it/~battiato/CVision0607/Lez.1.0-StereoVision.pdf.

[5] Algoritmi area-based. http://www.alexcozzi.com/publications/tesi/node26.html.

[6] Stan Birchfield and Carlo Tomasi. Depth discontinuities by pixel-to-pixel stereo. *IEEE*, 1998.

[7] Microchip Technology Inc. http://www.microchip.com/.

[8] Evidence srl, Pisa(I). http://www.evidence.eu.com/.

[9] Embedded Solutions srl, Milan(I). http://www.es-online.it/.

[10] Paolo Gai, Giuseppe Lipari, Luca Abeni, Marco di Natale, and Enrico Bini. Architecture for a portable open source real-time kernel environment. In *Proceedings of the Second Real-Time Linux Workshop and Hand's on Real-Time Linux Tutorial*, November 2000.

[11] OSEK/VDX Portal. http://www.osek-vdx.org/.

[12] Giorgio C. Buttazzo. *Sistemi in Tempo Reale*, chapter 12 Sistemi Operativi Esistenti, pages 341–362. Third edition, 2006.

[13] Eclipse - an open development platform. http://www.eclipse.org/.

[14] CC2420DK Development Kit. `http://focus.ti.com/docs/toolsw/folders/print/cc2420dk.html`.

Ringraziamenti

Ringrazio il prof. Aldo Franco Dragoni per avermi incoraggiato a svolgere questa tesi a Pisa. Ringrazio per il grande aiuto nella preparazione di questa tesi il Dott.Ing.Fis. Paolo Pagano che mi ha accolto nel gruppo WdSN (Wired Sensor Network)... ops ... WSN (wireless sensor network) del Retis Lab della Scuola Superiore Sant'Anna e mi ha iniziato all'acquisizione dei dati con il DAQ. Ringrazio anche i ragazzi che ho conosciuto in laboratorio e che mi hanno aiutato: Christian che mi ha fatto conoscere Erika (Enterprise) ed i segreti del compilatore, Daniele che mi ha trasportato ripetutamente tra Pisa e Ancona, Claudio che mi ha dato consigli preziosi per il progetto e Francesco che mi ha aiutato nel risolvere gli innumerevoli problemi di installazione dei vari software.
Ringrazio tutti i membri militanti della mia amata associazione Università Europea, grazie a loro ho imparato a fare politica universitaria, ad organizzare delle bellissime feste e affrontare senza timore qualsiasi tipo di burocrazia. In associazione ho trovato tanti amici con cui ho affrontato tante sfide.
Ringrazio gli amici incontrati nella mia carriera universitaria, Giulia che mi è stata vicina in questi lunghi 7 anni e mi ha aiutato a correggere questa tesi, i miei genitori che mi hanno sempre sostenuto ed i miei amici di Pescara con cui ho passato delle bellissime serate.