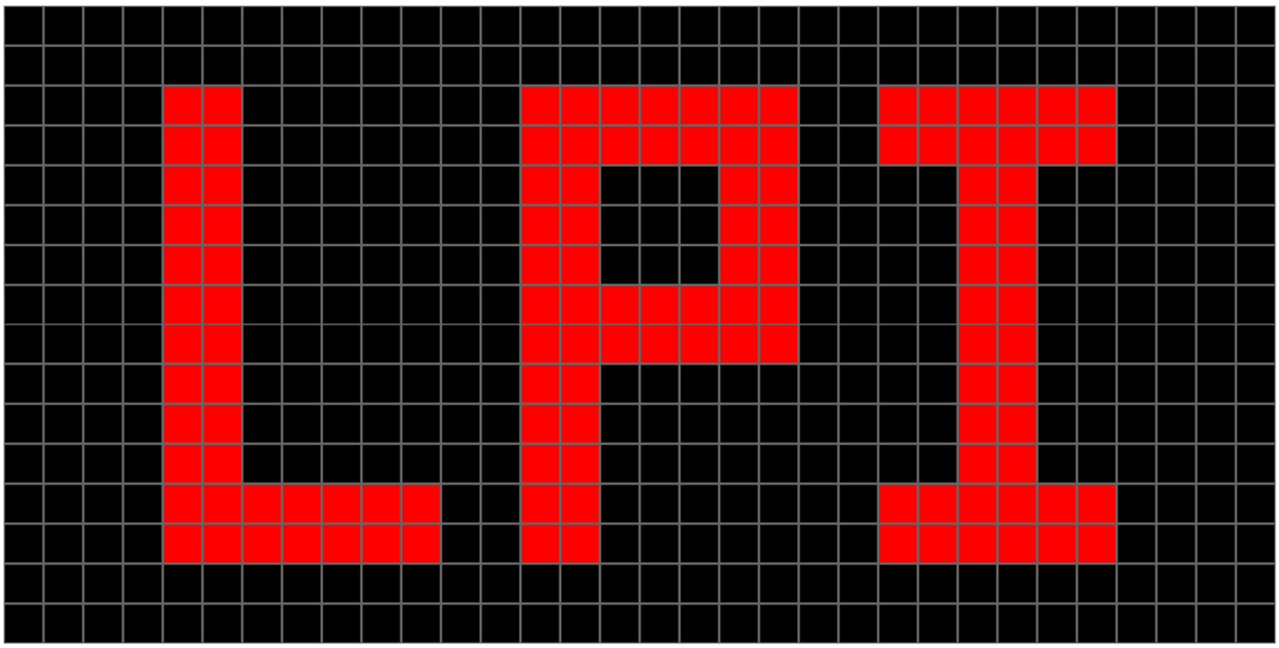
Repubblica e Cantone Ticino
Dipartimento dell'educazione, della cultura e dello sport
Divisione della formazione professionale

Anno scolastico 2016-17

Lavoro a progetti

Nome Cognome	**Patrik Randjelovic**
Professione:	**Elettronico**
Titolo del progetto	**Gestione matrici Dot LED**



Azienda:	**CPT** Centro professionale tecnico Viale S.Franscini 25 6500 Bellinzona		
Formazione approfondita:	**a.2 Sviluppare prototipi**		
Formatori:	**Lucchini M., Kamm D., Uboldi N.**		
Data inizio lavoro:	06.04.1017	Ore a disposizione:	86
Data fine lavoro:	16.04.2017	Ore effettive:	84

Sommario

Compito

Contesto generale

Come LPI, ho ricevuto l'incarico di adattare una libreria, già presente per Arduino, per essere utilizzata con MPLab v3.40 e il PIC18F45k22. Questa libreria facilita la visualizzazione di testi ed immagini sulle matrici 16x32 della Sure electronics.
Lo scopo è quello di creare una serie di esercizi didattici da proporre agli allievi del 3° anno e di portare una “demo” alle porte aperte.

Hardware utilizzato

- Due varianti della matrice 16x32 della Sure: una con i LED da 3mm e una da 5mm.
- PIC16F886 e PIC18F45k22
- EasyPIC v7
- PICkit 3

Software utilizzato

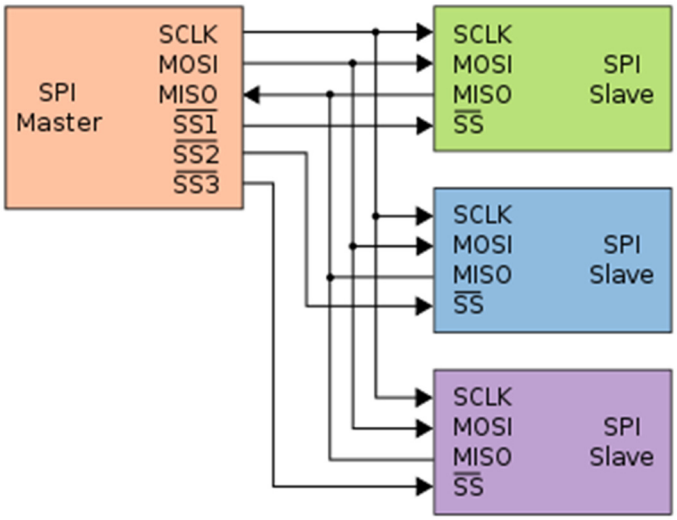
- MPLab X IDE v3.40

Messa in servizio

Dopo aver convertito la libreria HT1632 originariamente implementata per Arduino, ho caricato un breve programma di prova sul microcontrollore. Ho collegato la matrice 16x32 della SURE, alimentata a 12V. Il PIC è montato sulla piastra EasyPic v7 ed è alimentato a 5V tramite la presa USB del computer. Il microcontrollore e la matrice comunicano tramite il protocollo SPI con i fili GND, chipSelect, write, data e clock. Per programmare il PIC ho utilizzato PICKit 3 della Microchip.
Non ho riscontrato anomalie quali correnti eccessive o cali di tensione quindi ho considerato la messa in servizio conclusa.

Protocollo SPI

Il protocollo SPI (Serial Peripheral Interface) è uno standard di tipo master slave ed è seriale, sincrono e full-duplex. I bit vengono inviati serialmente grazie alla logica shift register.
Una caratteristica interessante è che la frequenza di trasmissione non è definita. Ovviamente ci sono dei limiti e sono dati dalle caratteristiche HW dei dispositivi interessati.
Per comunicare il protocollo utilizza 5 fili: GND, chipSelect (SS), data (MOSI), write (MISO) e clock (SCL).
ChipSelect è unica per ogni dispositivo slave. Il master attiva la linea dello slave a cui vuole inviare i dati o riceverli.
Data è la linea in uscita del master.
Write è la linea in entrata del master.
Il clock è definito dal master e definisce la frequenza di comunicazione con lo slave.



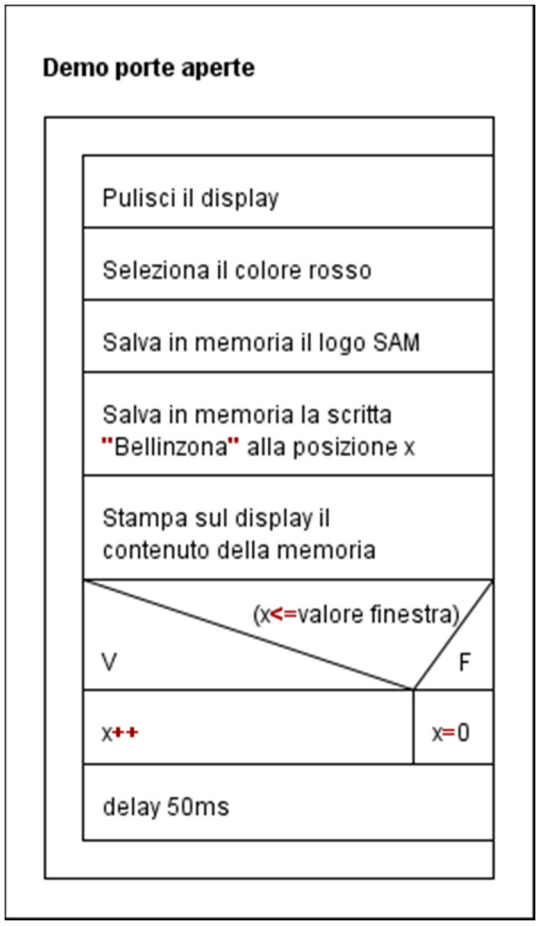
1 Esempio di collegamento

Demo per le porte aperte

Per le porte aperte ho realizzato un programma che stampa sulla matrice Dot Led il logo della SAM e la scritta “Bellinzona” che scorre.
Ho riproposto questa demo come esercizio “4.1-Logo SAMB” in quanto abbina un'immagine e una scritta sullo stesso display.

Il programma

Il programma salva nella memoria interna del PIC, appositamente dedicata, un immagine.
Di seguito sovrascrive la scritta “Bellinzona” in base alla posizione x.
Infine invia il contenuto della memoria alla matrice.



Le varie funzioni della libreria

Descrizione

Void beginCS ()

Inizializza le impostazioni della libreria.

Esempio sintassi:
beginCS(); //Inizializzazione della libreria HT1632.h

Void selectChannel(channel)

Permette di selezionare il colore della matrice.

Parametri:

uint8_t channel	Canale colore. 1=rosso 0=verde

Esempio sintassi:
selectChannel(0); //Seleziona il colore verde

Void renderTarget (target)

Nel caso in cui vi siano più matrici utilizzate, questa funzione permette di selezionare quella desiderata. Di default è selezionata la matrice 0.

Parametri:

uint8_t target	Numero della matrice da utilizzare

Esempio sintassi:
renderTarget(1); //Seleziona la seconda matrice collegata

Void render ()

Invia alla matrice selezionata con la funzione renderTarget () il contenuto salvato in memoria per poi visualizzarlo.

Esempio sintassi:
render(); //Invia il contenuto salvato in memoria alla matrice per visualizzarlo

Void clear ()

Libera la memoria.

Esempio sintassi:
Clear(); //Libera la memoria

Void drawImage (img, width, height, x, y, img_offset, offsetLeft, offsetRight)

Copia in memoria l'immagine rispettando i vari parametri.

Parametri:

Const byte * img	Puntatore dell'array dove è salvata l'immagine
Uint8_t width	Larghezza dell'immagine
Uint8_t height	Altezza dell'immagine
Uint8_t x	Posizione x
Uint8_t y	Posizione y
Int img_offset	Offset dei byte per colonna
Uint8_t offsetLeft	Inizio restrizione campo finestra
Uint8_t offsetRight	Fine restrizione campo finestra

Esempio sintassi:

```
//Sala in memoria l'immagine da visualizzare
drawImage(nomeImg, larghezzaImg, altezzaImg, 0, 0, 0, 0, larghezzaMatrice);
```

Int getTextWidth (text [], font_end [], font_height, gutter_space)

Restituisce la lunghezza in pixel della stringa contenente la scritta che si vuole visualizzare sul display.

Parametri:

Const char text []	Stringa contenente una scritta
Int font_end []	Array contenente la posizione finale dei font utilizzati per i caratteri
Uint8_t font_height	Altezza dei font utilizzati
Uint8_t gutter_space	Spaziatura tra i caratteri

Esempio sintassi:

```
//Richiedi la lunghezza in pixel della stringa testo
larghezzaTesto = getTextWidth(testo, FONT_5X4_END, FONT_5X4_HEIGHT, 1);
```

Void drawText (text[], x, y, font[], font_end[], font_height, gutter_space, offsetLeft, offsetRight)

Copia in memoria la stringa contenente la scritta che si vuole visualizzare sul display rispettando i vari parametri.

Parametri:

Const char text []	Stringa contenente una scritta
Int x	Posizione x
Int y	Posizione y
Const byte font []	Nome font utilizzati
Int font_end []	Array contenente la posizione finale dei font utilizzati per i caratteri
Uint8_t font_height	Altezza dei font utilizzati
Uint8_t gutter_space	Spaziatura tra i caratteri
Uint8_t offsetLeft	Inizio restrizione campo finestra
Uint8_t offsetRight	Fine restrizione campo finestra

Esempio sintassi:

```
//Salva in memoria la stringa testo da visualizzare
drawText(testo, 0, 0, FONT_5X4, FONT_5X4_END, FONT_5X4_HEIGHT, 1, 0,
OUT_SIZE);
```

Void setPixel(x, y, channel)

Salva in memoria il pixel desiderato.

Parametri:

uint8_t x	Posizione x
uint8_t y	Posizione y
uint8_t channel	Canale colore. 1=rosso 0=verde

Esempio sintassi:

```
setPixel(5,6,1);        //Salva in memoria il pixel rosso alla posizione x=5 e y=6
```

Void clearPixel(x, y, channel)

Pulisce dalla memoria il pixel alla coordinata x y sul canale del colore selezionato.

Parametri:

uint8_t x	Posizione x
uint8_t y	Posizione y
uint8_t channel	Canale colore. 1=rosso 0=verde

Esempio sintassi:

```
clearPixel(5,6,1);        // Pulisce dalla memoria il pixel rosso alla posizione x=5 e y=6
```

Uint8_t getPixel(x, y, channel)

Ritorna il valore del pixel alla coordinata x y sul canale del colore selezionato.

Parametri:

uint8_t x	Posizione x
uint8_t y	Posizione y
uint8_t channel	Canale colore. 1=rosso 0=verde

Esempio sintassi:

```
statoPixel = getPixel(5,6,1);        //Richiede lo stato del pixel alla posizione x=5, y=6 e al
                                      canale del colore rosso
```

Void fill()

Riempe l'intera matrice con il colore selezionato dalla funzione selectChannel(channel).

Esempio sintassi:

```
fill();       //Riempi la memoria con il colore selezionato
```

Void fillAll()

Riempe l'intera matrice con i colori rosso e verde (giallo).

Esempio sintassi:

```
fillAll();       // Riempi la memoria con il colore rosso e verde
```

Collegamenti fisici

- CS1 PORTC.2
- CLK PORTC.3
- WR PORTC.4
- DATA PORTC.5

Funzioni implementate

La funzione drawText() permette di salvare sulla memoria interna del PIC una stringa da visualizzare sulla matrice. Inizialmente si poteva scrivere solo sull'intera riga del display, aggiungendo 2 parametri alla funzione (offsetLeft e offsetRight) ho aggiunto la possibilità di limitare lo spazio di visualizzazione della stringa.
Essendo drawText() dipendente da drawImage(), ho dovuto modificare anche quest'ultima aggiungendo gli stessi parametri.

Conclusioni

Ho trovato il lavoro macchinoso per via che i compiti richiesti sono stati di mia competenza. Si è trattato quindi, ad eccezione di alcuni casi, di applicare le conoscenze acquisite in questi anni scolastici. Non ho riscontrato particolari problemi durante lo svolgimento del LPI tranne che per il PIC16F886 e un imprevisto del 12 maggio.
Il PIC16F886, inizialmente scelto per provare le funzioni della libreria HT1632.h, si è rivelato avere poca memoria. Quindi, dopo aver consultato il professore Lucchini, è stato deciso di non tenerlo in considerazione.
Di venerdì 12 maggio c'è stato un blackout, le valvole su cui sono collegati i computer dei banchi della mia fila sono saltate. Il file su cui stavo lavorando, quello degli esercizi didattici, per motivi che possono essere vari, è stato perso.

Allegati

Documenti

- Creare e configurare un progetto in MPLab v3.40
- HT1632-Reference
- Esercizi
- Diario giornaliero
- Pianificazione

Datasheet

- 41412F.pdf (PIC18(L)F2X/4Xk22)
- DE-DP14211_Ver1.1_EN (Sure electronics Dot Matrix)

Librerie e file base

- HT1632.h
- Images.h
- Font_5x4.h
- Main.c
- MyConfig.mc3

Programmi e struttogrammi

- Soluzioni degli esercizi didattici
- Demo porte aperte (4.1-Logo SAMB)

Altro

- Tool creazione immagini

Note