Daniele Ortu , Luca Putzolu, Gabriele Merlin

DJANGI GAMES

Sviluppo di un videogame per Arduino con l'utilizzo di un display LCD e touchscreen

Università degli studi di Cagliari
Corso di Informatica
Proggetto ARE

Indice

Introduzione

In occasione del corso di Architettura degli Elaboratori, tenuto dal prof. Reforgiato, abbiamo pensato di realizzare un progetto utilizzando il micro controllore Arduino UNO.

Il nostro intento era quello di realizzare qualcosa di particolare, diverso dai soliti progetti sulla domotica o sulla robotica che si reperiscono facilmente su internet.
Vedendo alcuni progetti per Arduino ci siamo interessati riguardo l'utilizzo e la gestione dei display di vario tipo. In particolare ci ha incuriosito un display LCD con touch screen incorporato visto su Youtube.
Ci siamo informati meglio al riguardo e, vedendo le caratteristiche e le possibilità che ci avrebbe fornito questo display, abbiamo pensato di creare un piccolo videogioco per Arduino, giocabile direttamente tramite touch screen.

Qualcuno prima di noi ha replicato il famoso videogioco Flappy Bird utilizzando, per l'appunto, un display lcd e touch screen. Il progetto di Flappy Bird, oltre a essere più semplice da realizzare, presenta meno caratteristiche rispetto al nostro e un diverso hardware.

Abbiamo sviluppato la nostra idea non tenendo conto di questo esempio (visto che il gioco che abbiamo realizzato è completamente diverso), ma studiando i vari componenti e le varie librerie, provando innanzitutto a sviluppare qualche semplice progetto (tavoletta da disegno, stampe di scritte e figure ecc.), poi componendo pezzo per pezzo il nostro videogioco.

Esistono degli schermi con processori grafici che si possono programmare ad alto livello e che permettono alte velocità di esecuzione. Tuttavia ci piaceva l'idea di agire a basso livello e con delle componenti basilari e poco costose. Programmare un videogioco agendo pixel per pixel, con hardware tutt'altro che veloce non è stato semplice ma è stato utile per comprendere meglio il funzionamento a basso livello di uno schermo.

Il videogioco "Djanni Bombs" ha come protagonista Djanni, un gatto. Il giocatore dovrà muovere il gatto per evitare delle bombe che avranno diversa frequenza a seconda del tempo trascorso. Più tempo si rimane in gioco più il punteggio sale. Quando una bomba colpisce il gatto allora il gioco sarà concluso.
Il progetto vuole essere una rivisitazione (più dinamica) del progetto di Programmazione 1 Exploding Djanni, replica del famoso gioco di carte Exploding Kittens, in cui il protagonista era sempre lo stesso gatto.

Componenti

Geekcreit® 2.8 Inch TFT LCD Shield Touch Display Module Link[1]

Caratteristiche

- 2.8 inch LCD TFT display
- Bright, 4 white-LED backlight
- Colorful, 18-bit 262,000 different shades
- 4-wire resistive touchscreen 240 x 320 pixels with individual pixel control
- No wiring, no soldering.
- Simply plug it in and load library
- On-board 3.3V 300mA LDO regulator
- 5V compatible, use with 3.3V or 5V logic
- Support 2GB micro SD TF card
- Size: 7.8 x 5.3cm 320x240 pixel

Il display LCD, utilizzato in questo progetto, non è altro che una shield da inserire direttamente sui pin di Arduino, senza l'utilizzo di breadboard. Ciò ha permesso una certa compattezza delle varie componenti, in modo da permettere una maggiore maneggiabilità. Il display, inoltre, dispone di un touch screen resistivo, che ha permesso di creare il videogioco utilizzando direttamente il touch screen per i comandi, senza il supporto di tasti o interruttori. Il display offre la possibilità di utilizzare 262000 tonalità di colore, potendo gestire ogni singolo pixel. Lo schermo, da 2.8 pollici, fornisce una risoluzione di 320x240 pixel. Attraverso lo schermo si possono visualizzare immagini bitmap, inserendole in una micro SD da inserire nell'apposito slot collocato nella shield.

Arduino UNO rev3 Link[2]

Caratteristiche

- Tipo Microcontrollore: Atmel ATmega328
- Tensione di lavoro: 5Vdc
- Tensione di alimentazione consigliata: da 7Vdc a 12Vdc
- Pin digitali: 14 configurabili come ingressi o uscite
- Pin analogici: 6 ingressi
- Massima corrente per pin digitale: 40mA massima
- Memoria Flash: 32KB
- Memoria Sram: 2KB
- Memoria EEPROM: 1KB
- Velocità di clock del microcontrollore: 16MHz

Arduino UNO rev3 è un microcontrollore molto semplice da utilizzare. Sul sito ufficiale è presente un IDE ufficiale per lo sviluppo su questa piattaforma. Si programma il microcontrollore in linguaggio C/C++, collegandolo direttamente a una porta usb del computer. Maggiori informazioni sono presenti sul sito ufficiale e su manuali specifici.

Scelta delle componenti

A partire dall'idea originaria potevano essere scelti centinaia di componenti utili allo scopo: una diversa versione di Arduino, un display monocromatico, un display di diversa dimensione, oppure non touch screen. Tuttavia si sono scelti questi due componenti per svariati motivi:

Arduino Uno rev. 3

Vantaggi:

- Ergonomicità.
- Spazio occupato in un eventuale case.

Svantaggi:

- Poca memoria disponibile.
- Nessun pin rimasto inutilizzati.

Geekcreit® 2.8 Inch TFT LCD Shield Touch Display Module

Vantaggi:

- Si tratta di una shield da montare direttamente sui pin di Arduino senza l'utilizzo di breadboard e cavi.
- Spazio occupato in un eventuale case.
- Stesse dimensioni di Arduino Uno.

Svantaggi:

- Animazioni e figure da programmare a basso livello (pixel per pixel).
- Nessuna regolazione per la luminosità.
- Difficoltà nella gestione dell'orientamento del display con quello delle coordinate del touch screen.

Nonostante i vari limiti delle componenti, abbiamo cercato di sfruttarle al massimo, cercando di arginare i difetti e di sfruttare al massimo l'hardware. Utilizzare un Arduino Mega (l'unico, oltre ad Arduino UNO, compatibile col display) avrebbe comportato la disponibilità di una memoria più capiente, cioè si sarebbero potuti inserire più elementi grafici, ma ci sarebbe stato un grosso spreco di pin, oltre che una minore maneggevolezza. Le migliorie sarebbero state poche, magari si sarebbe potuto introdurre un altoparlante oppure altre funzionalità, ma in ogni caso sarebbero state secondarie in confronto al lavoro svolto e ai nostri interessi.

Librerie

Per l'utilizzo del display abbiamo usufruito di due librerie, una per la gestione dell'LCD, l'altra per la gestione del touch screen.

Adafruit GFX **Link[3]**

La libreria Adafruit GFX gestisce le funzioni relative al display LCD. Permette la gestione di ogni singolo pixel, la creazione di forme geometriche, la stampa di scritte e di caratteri.
Di seguito riportiamo alcune tra le funzioni più utilizzate:

tft.drawPixel(int16_t x, int16_t y, uint16_t color);
tft.dawFastHLine(int16_t x0, int16_t y0, int16_t w, uint16_t color);
tft.drawFastVLine(int16_t x0, int16_t y0, int16_t h, uint16_t color);
tft.fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t c);
tft.drawRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t radius, uint16_t color);
tft.fillRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t radius, uint16_t color);
*tft.*drawChar(uint16_t x, uint16_t y, char c, uint16_t color, uint16_t bg, uint8_t size);
*tft.*setCursor(uint16_t x0, uint16_t y0);
*tft.*setTextColor(uint16_t color);
*tft.*setTextColor(uint16_t color, uint16_t backgroundcolor);
*tft.*setTextSize(uint8_t size);
*tft.*setTextWrap(boolean w);
tft.print(char)*
tft.fillScreen(uint16_t color);
tft.reset(void);
tft.setRotation(uint8_t x);

I colori utilizzati sono nel formato RGB365, perciò ci siamo serviti di un convertitore online (**Link[4]**) per la conversione dal formato classico (componenti RGB 0-255) al formato richiesto dalla libreria.

Touchscreen **Link[5]**

La libreria per la gestione del touchscreen è stata utilizzata per ricevere in input i punti dello schermo toccati dall'utente, attraverso un sistema di coordinate composto da 320x240 pixel selezionabili. Toccando un punto si deve in seguito "mappare" attraverso una funzione apposita che si occupa di cambiare i valori di tale punto in relazione all'altezza e laghezza dello schermo. La libreria fornisce un nuovo tipo di dato chiamato TSPoint, che rappresenta quindi un punto con coordinate x e y e un indicatore per la pressione, z.
Di seguito riportiamo alcune tra le funzioni più utilizzate:

ts.getPoint(void)
ts.pressureThreshhold(void)

Precisazioni

- Per garantire una corretta esecuzione del codice è necessario inizializzare il touchscreen e il display attraverso le funzioni riportate di seguito:

Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

```
//_____________________________________________________________________LCD
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0
#define LCD_RESET A4
//--------------------------------------------------------------------

//_____________________________________________________________TOUCHSCREEN
#define YP A2  // must be an analog pin, use "An" notation!
#define XM A3  // must be an analog pin, use "An" notation!
#define YM 8   // can be a digital pin
#define XP 9   // can be a digital pin


#define TS_MINX 914// must be an analog pin, use "An" notation!
#define TS_MINY 118 // must be an analog pin, use "An" notation!
#define TS_MAXY 912// can be a digital pin
#define TS_MAXX 124
//--------------------------------------------------------------------
```

- Dopo aver richiesto in input un punto tramite la funzione *ts.getPoint,* è necessario reimpostare i pin utilizzati per tale richiesta e “mappare” il punto inserito.

```
p = ts.getPoint();
pinMode(XM, OUTPUT);
pinMode(YP, OUTPUT);
p.x = map(p.x, TS_MAXX, TS_MINX, 0, 320);
p.y = map(p.y, TS_MAXY, TS_MINY, 0, 240);
```

Fasi del progetto

Step 1:

In una prima fase abbiamo recuperato tutto il materiale necessario: Arduino, display, librerie, IDE Arduino e alcuni esempi di utilizzo **Link**[6]. In seguito abbiamo fatto alcune prove per verificare e conoscere meglio il funzionamento del display e di Arduino. In seguito abbiamo sviluppato qualche semplice programma per utilizzare il touch screen.

Step 2:

In un secondo tempo abbiamo ideato il videogioco, pensato a quante schermate dovesse avere, quali elementi grafici servissero e deciso l'orientamento del display col quale avremo lavorato. A tal proposito è stato necessario prendere un asse di coordinate per i punti del touch screen diverso da quello per i pixel del display. Per questo in alcune parti del codice in cui ci dovrebbe essere la coordinate indicante le x abbiamo dovuto inserire il valore y e viceversa.

Step 3:

In seguito abbiamo implementato una bozza delle varie schermate e del campo da gioco.

Step 4:

Ci siamo poi interessati alla stampa di un'immagine. Quest'ultimo passo è stato più problematico rispetto agli altri per due motivi:

1. Nonostante il display disponga di un ingresso micro SD attraverso il quale, tramite un'apposita funzione, è possibile realizzare la stampa, quest'ultima risulta essere molto lenta.
2. La memori di Arduino utilizzabile è di 32 Kb.

La nostra esigenza è stata quella di stampare un'immagine e, ogni volta che il giocatore avesse cliccato su un punto dello schermo, tale immagine doveva essere cancellata e stampata nella nuova posizione. Per questo fine ci serviva velocità e, utilizzare immagini bitmap, impiegava troppo tempo, invece doveva essere qualcosa di quasi istantaneo. Infatti, abbiamo optato per la traduzione della nostra immagine (**Link**[7])in una matrice di pixel da dichiarare all'inizio del programma come variabile globale. Ogni campo rappresenta il colore di ogni pixel dell'immagine. Purtroppo, avendo poca memoria a disposizione abbiamo inserito una sola immagine.
Finché si fosse trattato di un'immagine 5x5 sarebbe stato semplice campionare ogni pixel manualmente, il nostro sprite prevedeva però una dimensione di 20x25.
Ci siamo serviti perciò del sito internet Piskel (**Link**[8]) che, tra le altre funzioni, permette, a partire da un'immagine, di creare una matrice di colori di pixel, già tabulata secondo il linguaggio C.
L'unica cosa che abbiamo dovuto fare è convertire ogni singolo colore (nel gatto ce ne sono stati circa 30), presente nella matrice, nel formato RGB365, come spiegato nel paragrafo Librerie.
Tramite la funzione tft.drawPixel abbiamo stampato ogni singolo pixel della matrice.

```
const uint16_t   spriteCat[20][25]  =
{
  {0xF800    , 0xF800    , 0xF800    , 0xF800    , 0xF800        .....
  {0xF800    , 0xF800    , 0xF800    , 0xF800    , 0xF800        .....
  {0xF800    , 0xF800    , 0x10A2    , 0x528E    , 0x10A2        .....
  {0xF800    , 0x10A2    , 0x6B50    , 0x8C75    , 0x10A2        .....
  {0xF800    , 0x10A2    , 0x5AEF    , 0x10A2    , 0xF800        .....
  {0xF800    , 0x10A2    , 0x528E    , 0x10A2    , 0xF800        .....
  {0xF800    , 0x10A2    , 0x6B50    , 0x10A2    , 0xF800        .....
  {0xF800    , 0xF800    , 0x10A2    , 0x6B50    , 0x10A2        .....
  {0xF800    , 0xF800    , 0x10A2    , 0x10A2    , 0x39CB        .....
  {0xF800    , 0xF800    , 0xF800    , 0x39CB    , 0x4A2C        .....
  {0xF800    , 0xF800    , 0x10A2    , 0x5AEF    , 0xFFFF
  {0xF800    , 0xF800    , 0x10A2    , 0xFFFF    , 0xFFFF        .....
  {0xF800    , 0xF800    , 0x10A2    , 0xEF5E    , 0xFFFF        .....
  {0xF800    , 0xF800    , 0x10A2    , 0xEF5E    , 0xEF5E        .....
  {0xF800    , 0xF800    , 0x10A2    , 0xEF5E    , 0xFFFF
  {0xF800    , 0x10A2    , 0x10A2    , 0xEF5E    , 0xFFFF        .....
  {0xF800    , 0x10A2    , 0x39CB    , 0xAD78    , 0xEF5E        .....
  {0x10A2    , 0x73B2    , 0x528E    , 0x10A2    , 0x10A2        .....
  {0x10A2    , 0x7C13    , 0x528E    , 0xF800    , 0xF800
  {0xF800    , 0x10A2    , 0x10A2    , 0xF800    , 0xF800        .....
 };
```

Step 5:

Un ulteriore step è stato quello di aggiornare la posizione dello sprite a seconda del punto toccato dal giocatore. Ogni volta che si aggiornava l'immagine rallentava l'esecuzione di tutti gli altri elementi del gioco, perciò abbiamo utilizzato dei delay affiché non si notasse la differenza, e in modo tale che il refresh non sia veloce a tal punto da non distinguere chiaramente l'immagine. Oltre a questo, una condizione da utilizzare sempre, è stata quella che la pressione sia superiore a un certo valore minimo datoci dalla funzione *ts.pressureThreshhold(void).*

Step 6:

In seguito allo sprite del gatto ci siamo occupati delle bombe. Esse sarebbero dovute fuoriuscire dalla parte sinistra dello schermo e arrivare sino alla parte destra. Abbiamo realizzato un array di un nuovo tipo "Bomba", di 12 elementi (il numero 12 è dovuto alla distanza tra proiettili e alla loro dimensione 10x10) contenente coordinate x e y di ogni bomba e un booleano "on" indicante lo stato della bomba (attiva o disattiva). A seconda del tempo trascorso dall'inizio del gioco ci sarà una generazione più frequente di bombe, sarà più probabile dunque incappare in "muri di proiettili". Per problemi di velocità di Arduino non è stato possibile modificare la velocità dei proiettili. Lo stato della bomba ritorna a false una volta che essa ha raggiunto la fine dello schermo. Per complicare ulteriormente il gioco sarebbe sufficiente creare un altro (o più) array di 12 bombe da generare in parallelo, sempre a seconda del tempo trascorso, in questo modo i proiettili risulteranno uno dietro l'altro senza la necessità di aspettare l'arrivo di una bomba, della stessa linea, alla fine dello schermo.

Step 7:

Dopo aver rifinito il timer e i vari elementi grafici essenziali per il gioco ci siamo concentrati sul punteggio. A seconda del tempo trascorso si avrà un punteggio maggiore, inoltre abbiamo implementato una tabella dei record che si conserva anche dopo aver spento Arduino, mediante il salvataggio dei dati su eeprom. La tabella è resettabile tramite un apposito tasto. È stato complicato realizzare la stampa dei punteggi, in quanto la funzione di stampa del display non ha la possibilità di stampare direttamente il valore delle variabili. Abbiamo provveduto quindi, a partire da ogni punteggio, estrapolare ogni singola cifra e, convertendola in un carattere hash, a stamparla

tramite la funzione *tft.drawChar(uint16_t x, uint16_t y, char c, uint16_t color, uint16_t bg, uint8_t size);*
Per il salvataggio su eeprom abbiamo utilizzato la libreria apposita e due funzioni per il salvataggio e la scrittura di qualunque valore su di essa.

```
//Score's print
tft.setCursor(45, 140);
tft.setTextColor(WHITE);
tft.setTextSize(2);
tft.print("Il tuo punteggio e':");

stampa.x=123;
for(i=4;i>=0;i--)
{
  array[i]=score/(pow(10,i));//extract the digit
  score=score-(array[i]*pow(10,i));//update the score
  charpunteggio[i]=array[i]+48;//convert the digit in a char
  tft.drawChar(stampa.x, 170, charpunteggio[i], WHITE, BLACK, 2);//print the digit
  stampa.x=stampa.x+14;//traslate the position of the next digit
}
```

Step 8:

Come ultima fase ci siamo occupati del case che avrebbe dovuto contenere Aduino e il display. Il contenitore è stato realizzato con del lamierino di alluminio tagliato e modellato a mano. Abbiamo provveduto a praticare il foro per il display, dopodiché abbiamo pensato di inserire anche una batteria e un interruttore. A tale scopo abbiamo recuperato l'interruttore da un vecchio caricatore non funzionante e, collegandolo a una batteria da 9 volt tramite l'apposito clip, abbiamo saldato i fili a un connettore 2,1×5 mm apposito per Arduino. Dopo aver fatto il foro per l'interruttore nella parte superiore, avendo verificato il corretto funzionamento, abbiamo creato una piccola prolunga per la presa usb di Arduino, utilizzando i connettori adatti, cercando di impiegare il minore spazio possibile. Sarà quindi possibile collegare Arduino al computer, per modificare il programma, senza aprire il case. Abbiamo regolato l'altezza di Arduino tramite un piccolo rialzo in legno e, praticando dei fori per le viti, lo abbiamo ancorato al case. Dopo aver levigato la custodia, l'abbiamo verniciata.

Link

[1] Display Bangood:
https://www.banggood.com/it/2_8-Inch-TFT-LCD-Shield-Touch-Display-Module-For-Arduino-UNO-p-989697.html?p=DQ30066511122014069J&utm_campaign=educ8stv&utm_content=huangwenjie

[2] Caratteristiche Arduino:
https://store.arduino.cc/arduino-uno-rev3

[3] Download Adafruit gfx library:
https://github.com/adafruit/Adafruit-GFX-Library
Manuale Adafruit gfx library:
https://learn.adafruit.com/adafruit-gfx-graphics-library#

[3] RGB 365 converter:
http://www.rinkydinkelectronics.com/calc_rgb565.php

[5] Download Touchscreen library:
https://github.com/adafruit/Touch-Screen-Library
Manuale Touchscreen library:
https://learn.adafruit.com/2-8-tft-touchscreen/touchscreen

[6] Esempi programmazione display:
Flappy bird: https://www.youtube.com/watch?v=jPU4iv378ig
Exemples: https://www.youtube.com/watch?v=IcIY2pWursc

[7] Sprites Djanni
http://finalfantasy.wikia.com/wiki/File:TAY_PSP_Cat_Lover_No1.png

[8] Piskel:
http://www.piskelapp.com/