Esperienza di laboratorio

Arduino DUE
Display LCD e sensori

Gruppo A6
Giacomo Calabria - 2007964
Daniele Venturini - 1195858

19 May 2023

Contents

INTRODUZIONE

Lo scopo dell'esperienza di laboratorio è utilizzare la scheda Arduino per implementare dei programmi che riescano a fare le seguenti operazioni

- Timer con Display TFT
- Stazione meteo
- Sensore di distanza ultrasuoni (facoltativo)

Strumentazione necessaria:

- Scheda Arduino Due
- Computer con software Arduino IDE
- Cavo USB - Type A
- Breadboard
- Display TFT 3.5" 320x480, *HX8357* Adafruit

Il Display TFT *HX8357*

In Figura 1 è stato riportato il pinout del Display HX8357 dal lato dell' interfaccia SPI

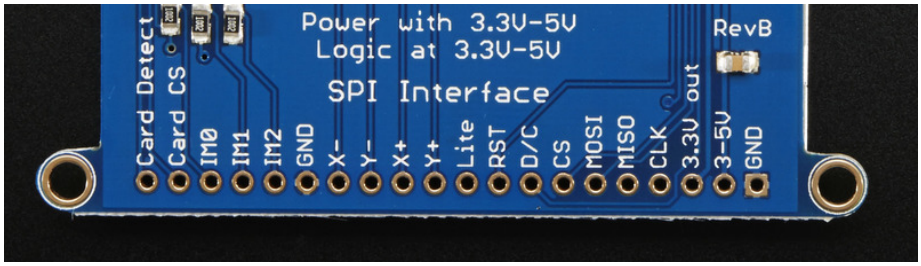

Figure 1: Pinout del Display HX8357

Il Display viene controllato mediante l'interfaccia Serial Peripheral Interface (SPI) la quale richiede il collegamento dei seguenti pin:

- alimentazioni: `VCC` e `GND`
- linee comuni per i dati: `MISO`, `MOSI` e `SCK`
- linea per il *chip select* (`CS`)

Il Display HX8357 richiede il collegamento di un ulteriore pin `D/C` per la selezione della modalità data/command. Per i pin `MISO`, `MOSI` e `SCK` utilizziamo l'header `ICSP` di Arduino Due integrato nella scheda.

1 Primo esperimento

Il primo esperimento ha lo scopo di introdurre le procedure di comando di un display TFT mediante scheda Arduino DUE. Si scriverà un programma che realizzi un semplice cronometro a tre stati di funzionamento

- **Stato 0:** il circuito attende la pressione del tasto T visualizzando il tempo "0.00 s". Durante l'attesa viene visualizzata la scritta "Press to Start"
- **Stato 1:** nel momento in cui il tasto T viene premuto, il timer comincia a misurare il tempo, a step di 50 ms. La misura finisce quando il tasto viene rilasciato. Durante la misura appare la scritta "Release to Stop"
- **Stato 2:** quando il tasto T viene rilasciato, il timer si ferma, e visualizza il tempo totale durante cui il tasto è rimasto premuto. Viene inoltre visualizzata la scritta "Press to Reset". Una ulteriore pressione del tasto T resetta il timer e fa tornare il circuito allo stato 0

I componenti necessari a questa esperienza sono:

- Scheda Arduino DUE
- Breadboard e cavi
- Display TFT 3.5" 320x480, *HX8357* Adafruit
- Interruttore a bottone, FSM2JART, Cod. RS 745-5185
- Resistenza $R = 10\ \text{k}\Omega, 0.25\ \text{W}$

Il circuito è alimentato mediante porta USB del PC, la quale eroga circa ($\sim 5V$).

1.1 Codice

La libreria `Adafruit_HX8357.h`, insieme alle librerie `Adafruit_GFX.h` e `SPI.h` permette alla scheda arduino di comunicare con il Display HX8357 tramite le funzioni dedicate.

```
1  #include <SPI.h>
2  #include "Adafruit_GFX.h"
3  #include "Adafruit_HX8357.h"
4
5  #define TFT_CS 10
6  #define TFT_DC 9
7
8  Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, -1);
9  // TFT_RST set to -1 to tie it to Arduino's reset
10
11 #define buttonPin 2
12
13 int buttonState = LOW;
14 int lastButtonState = LOW;
15
```

```
unsigned long startTime = 0;

int stato = 0;
int reset = 0;

void setup(){
  pinMode(buttonPin, INPUT);
  tft.begin();
  tft.setRotation(3);
  tft.fillScreen(HX8357_BLACK);
  tft.setTextColor(HX8357_GREEN);
  tft.setTextSize(8);
  tft.setCursor(100,0);
  tft.println("TIMER");
  tft.setTextColor(HX8357_BLUE);
  tft.setCursor(100, 80);
  tft.print("0.00 s");
  tft.setCursor(20, 180);
  tft.setTextColor(HX8357_RED);
  tft.setTextSize(5);
  tft.print("Press to Start");
}
```

```
void loop() {
  buttonState = digitalRead(buttonPin);
  if(buttonState == HIGH && lastButtonState == LOW && reset == 1){
    startTime = 0;
    reset = 0;
    stato = 0;
    tft.fillRect(0, 60, 480, 320, HX8357_BLACK);
    tft.setTextSize(8);
    tft.setTextColor(HX8357_BLUE);
    tft.setCursor(100, 80);
    tft.print("0.00 s");
    tft.setCursor(20, 180);
    tft.setTextColor(HX8357_RED);
    tft.setTextSize(5);
    tft.print("Press to Start");
  }
```

```
  if (buttonState == HIGH && lastButtonState == LOW && stato == 0){
    lastButtonState = buttonState;
    stato = 1;

    if(startTime == 0){
      startTime = millis();
    }

    tft.fillRect(0, 60, 480, 320, HX8357_BLACK);
    tft.setCursor(20, 180);
    tft.setTextSize(5);
    tft.setTextColor(HX8357_RED);
    tft.print("Release␣to␣Stop");
  }
  if (stato == 1 && buttonState == HIGH){
    unsigned long centiseconds = (millis() - startTime) / 10;
    unsigned long seconds = centiseconds / 100;
    centiseconds %= 100;

    tft.setTextSize(8);
    tft.setTextColor(HX8357_BLUE);
    tft.setCursor(100, 80);
    tft.print(String(seconds));
    tft.print(".");
    if (centiseconds < 10) {
      tft.print("0");
    }
    tft.print(String(centiseconds));
    tft.println("␣s");
    delay(50);
    tft.fillRect(0, 60, 480, 80, HX8357_BLACK);
  }
```

```
  if (buttonState == LOW && lastButtonState == HIGH){
    lastButtonState = buttonState;

    tft.fillRect(0, 60, 480, 320, HX8357_BLACK);
    tft.setCursor(20, 180);
    tft.setTextSize(5);
    tft.setTextColor(HX8357_RED);
    tft.print("Press to Reset");

    unsigned long centiseconds = (millis() - startTime) / 10;
    unsigned long seconds = centiseconds / 100;
    centiseconds %= 100;

    tft.setTextSize(8);
    tft.setTextColor(HX8357_BLUE);
    tft.setCursor(100, 80);
    tft.print(String(seconds));
    tft.print(".");
    if (centiseconds < 10) {
      tft.print("0");
    }
    tft.print(String(centiseconds));
    tft.println(" s");
    reset = 1;
    stato = 3;
    startTime = 0;
  }
}
```

E' possibile apprezzare il funzionamento del circuito dal video al seguente link errato

2 Secondo esperimento

Nel secondo esperimento si vuole realizzare una semplice stazione di monitoraggio ambientale basata su scheda Arduino, display TFT, sensore analogico di umidità e sensori digitali I^2C di temperatura e luminosità.

Per questo esperimento sono stati utilizzati i seguenti componenti:

- Sensore di temperatura analogico, codice *TMP36*, Analog Devices
- Sensore di umidità analogico, codice *HIH-5030-001*, Honeywell
- Sensore di temperatura digitale, codice *DS1621*, Maxim
- Sensore di luce ambientale, codice *TSL2561*, TAOS
- Display TFT 3.5" 320x480, *HX8357* Adafruit
- Scheda Arduino DUE
- Breadboard e cavi
- Interruttore

Il circuito è alimentato mediante porta USB del PC, la quale eroga circa ($\sim 5V$). La scheda Arduino due ha due bus I^2C, utilizzeremo i pin 20(SDA) e 21(SCL).

2.1 Codice - interfaccia grafica

Per il funzionamento del Display HX8357 con Arduino è necessario importare le librerie `Adafruit_HX8357.h`, `Adafruit_GFX.h` e `SPI.h`. Per comunicare con i dispositivi I^2C è necessario utilizzare la libreria `Wire.h`.

```
#include <SPI.h>
#include <Wire.h>
#include "Adafruit_GFX.h"
#include "Adafruit_HX8357.h"

#define TEMP_DEV_ID 0x48
#define LIGHT_DEV_ID 0x39

#define HUMIDITY_DEV_PIN A1
#define TEMP_PIN A0

#define TFT_CS 10
#define TFT_DC 9

Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, -1);
// TFT_RST set to -1 to tie it to Arduino's reset

#define buttonPin 2
```

Variabili globali

```
bool pulsante_precedente_alto = LOW;
bool stateButton = LOW;

int stato = 0;
int stato_precedente = 0;
float temperatura_precedente_analog = 0;
```

```
void setup(){
    analogReadResolution(12);
    Wire.begin();

    // Configurazione del sensore di temperatura digitale DS1621
    Wire.beginTransmission(TEMP_DEV_ID);
    Wire.write(0xAC);
    Wire.write(0x02);
    Wire.endTransmission();

    Wire.beginTransmission(TEMP_DEV_ID);
    Wire.write(0xEE);
    Wire.endTransmission();

    // Configurazione del sensore di luce digitale TSL2561
    Wire.beginTransmission(LIGHT_DEV_ID);
    Wire.write(0x00);
    Wire.write(0x03);
    Wire.endTransmission();

    Wire.beginTransmission(LIGHT_DEV_ID);
    Wire.write(0x81);
    Wire.write(0x02);
    Wire.endTransmission();

    // configurazione del Display TFT
    tft.begin();
    tft.setRotation(3);
    tft.fillScreen(HX8357_BLACK);
    tft.setTextSize(5);

    attachInterrupt(digitalPinToInterrupt(2),change_state,RISING);
}
```

Funzione interrupt

```
void change_state(){
  stato++;
```

```
  if(stato >= 4)
    stato = 0;
}
```

```
void loop(){
    switch(stato){
        case 0:{
            if (stato != stato_precedente) {
              tft.fillScreen(HX8357_BLACK);
              tft.setCursor(30,60);
              tft.println("Temperatura 1");
            }
            float temp = AnalogTemperature(TEMP_PIN);
            String txt = String(temp);
            tft.setCursor(60,120);
            tft.fillRect(60,120,480,320, HX8357_BLACK);
            tft.println(txt + " C");
            stato_precedente = 0;
        }
        break;
        case 1:{
            if (stato != stato_precedente) {
              tft.fillScreen(HX8357_BLACK);
              tft.setCursor(30,60);
              tft.println("Temperatura 2");
            }
            float temp = DigitalTemperature();
            String txt = String(temp);
            tft.setCursor(60,120);
            tft.fillRect(60,120,480,320, HX8357_BLACK);
            tft.println(txt + " C");
            stato_precedente = 1;
        }
        break;
        case 2:{
            if(stato != stato_precedente) {
              tft.fillScreen(HX8357_BLACK);
              tft.setCursor(30,60);
              tft.println("Luminanza");
            }

            int lux = DigitalLight();
            String txt = String(lux);
            tft.setCursor(60,120);
            tft.fillRect(60,120,480,320, HX8357_BLACK);
            tft.println(txt + " lux");
```

```
43              stato_precedente = 2;
44          }
45          break;
46          case 3:{
47              if (stato != stato_precedente) {
48                tft.fillScreen(HX8357_BLACK);
49                tft.setCursor(30,60);
50                tft.println("Humidity");
51              }
52              float hum = AnalogHumidity(HUMIDITY_DEV_PIN);
53              String txt = String(hum);
54              tft.setCursor(60,120);
55              tft.fillRect(60,120,480,320, HX8357_BLACK);
56              tft.println(txt + "␣%");
57              stato_precedente = 3;
58          }
59      }
60      delay(500);
61  }
```

2.2 Codice - lettura sensori

Qui di seguito riportiamo le funzioni che leggono i dati dai vari sensori. Per le letture dai sensori analogici abbiamo scelto di impostare il DAC di Arduino Due a 12 bit, in modo da avere una risoluzione maggiore con 4096 livelli

Listing 1: Lettura dal sensore TMP36

```
int AnalogTemperature(int pin) {
    float analogValue = analogRead(pin);
    float temperatura = (analogValue - 155) *330.0/4095.0; //12 bit DAC
    return temperatura;
}
```

Per ricavare il valore di umidità e correggerlo in base alla temperatura dal sensore a abbiamo utilizzato le formule fornite dal datasheet del componente

```
int AnalogHumidity(int pin) {
    float analogValue = analogRead(pin);
    float rhvoltage = (analogValue/4095.0)*3.3; // 12 bit DAC
    float umidita = ((rhvoltage/3.3)-0.1515)/0.00636;
    umidita = umidita/(1.0546-0.00216*AnalogTemperature(A0));
    return umidita;
}
```

```
int DigitalTemperature(){
    int firstByte;
    int secondByte;
    float temp = 0;

    Wire.beginTransmission(DEV_ID);
    Wire.write(0xAA); // Temperature register
    Wire.endTransmission();
    Wire.requestFrom(DEV_ID, 2); // request 2 bytes from slave device

    firstByte = Wire.read(); // read first byte
    secondByte = Wire.read(); // read second byte

    temp = firstByte;

    if(secondByte){
        temp += 0.5;
    }

    return temp;
}
```

```
int DigitalLight(){
    byte firstByte, secondByte;
    int lux = 0;

    Wire.beginTransmission(DEV_ID2);
    Wire.write(0x8C); // Temperature register
    Wire.endTransmission();
    Wire.requestFrom(DEV_ID2, 1);

    firstByte = Wire.read(); // read first byte

    Wire.beginTransmission(DEV_ID2);
    Wire.write(0x8D); // Temperature register
    Wire.endTransmission();
    Wire.requestFrom(DEV_ID2, 1);

    secondByte = Wire.read(); // read second byte

    lux = 256 * int(secondByte) + int(firstByte);
    // convert the two bytes to int
    return lux;
}
```

E' possibile apprezzare il funzionamento del circuito dal video al seguente link errato

3 Terzo esperimento

In questa esperienza si vuole realizzare un misuratore di distanza, utilizzando il sensore di distanza a ultrasuoni. Sono stati utilizzati i seguenti componenti:

- Scheda Arduino DUE
- Display TFT 3.5" 320x480, *HX8357* Adafruit
- Breadboard e cavi
- Sensore ultrasonico distanza HC-SR04

Il circuito è alimentato mediante porta USB del PC, la quale eroga circa ($\sim 5V$)

Figure 2: Pinout del sensore HC-SR04

3.1 Codice - interfaccia grafica

```
#include <SPI.h>
#include "Adafruit_GFX.h"
#include "Adafruit_HX8357.h"

#define TRIG_PIN 7
#define ECHO_PIN 6

#define TFT_CS 10
#define TFT_DC 9

Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, -1);

void setup(){
    pinMode(ECHO_PIN, INPUT);
    pinMode(TRIG_PIN, OUTPUT);

    // Configurazione del display
    tft.begin();
    tft.setRotation(3);
```

```
    tft.fillScreen(HX8357_BLACK);
    tft.setTextColor(HX8357_GREEN);
    tft.setTextSize(6);
    tft.setCursor(60,0);
    tft.println("Distanza");
    tft.setTextColor(HX8357_WHITE);
    tft.setTextSize(10);
}

void loop(){
    String DistTxt = String(distance());

    tft.fillRect(0,120,480,320,HX8357_BLACK);
    tft.setCursor(0,150);
    tft.println(DistTxt + "␣cm");

    delay(100);
}
```

3.2 Codice - lettura sensore ultrasonico

Scrivere cosa fa il codice

```
long distance(){
    long d = 0;
    long duration = 0;

    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);

    duration = pulseIn(ECHO_PIN, HIGH);
    d = (duration * 100) / 5830;

    delay(25);
    return d;
}
```

E' possibile apprezzare il funzionamento del circuito dal video al seguente link errato