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 Due per implementare dei programmi che interagiscono con un Display TFT e che riescano a eseguire le seguenti operazioni

- Timer
- Stazione multi-sensore
- Sensore di distanza ultrasuoni (facoltativo)

Strumentazione necessaria:

- Scheda Arduino Due, breadboard e cavi
- Computer con software Arduino IDE + Cavo USB - Type A
- 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

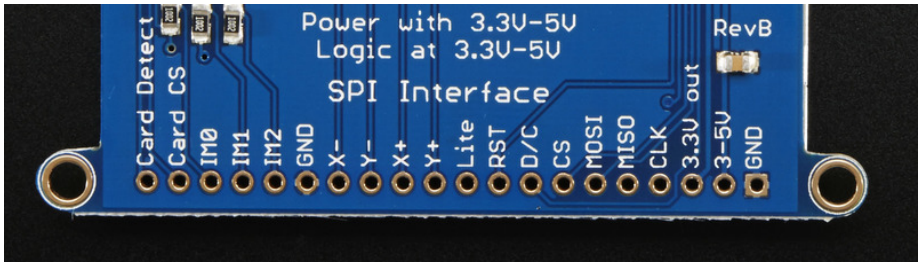

Figure 1: Pinout del Display HX8357

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

- alimentazioni: `VCC` (3.3V) e `GND`
- linee comuni: `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 `SPI` 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 e resistenza $R = 10$ kΩ

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
```

```
bool buttonState = LOW;
bool lastButtonState = LOW;
int stato = 0;
int reset = 0;

unsigned long startTime = 0;

void setup(){
  pinMode(buttonPin, INPUT);

  tft.begin(); // Inizializzazione display
  tft.setRotation(3); //Orientamento orizzontale
  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

2 Secondo esperimento

Nel secondo esperimento si vuole realizzare una semplice stazione di monitoraggio ambientale gestota dalla scheda Arduino Due che tramite un display TFT, riporta i dati da sensori analogici di umidità e temperatura 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 e resistenza $R = 10$ kΩ

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).

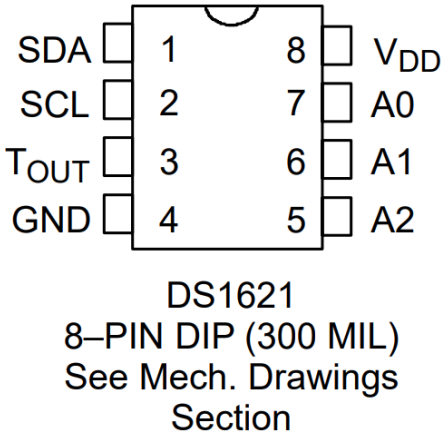

Figure 2: Pinout sensore DS1621

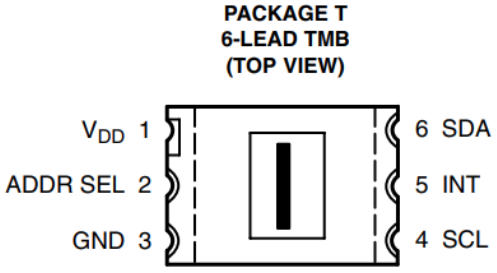

Figure 3: Pinout sensore TSL2561

Abbiamo collegato a GND i pin A_0, A_1, A_2 del sensore DS1621 in modo che il suo indirizzo sia `0x48`. Lasciando flottante il pin `ADDR SEL` del sensore TSL2561 il suo indirizzo è `0x39`.

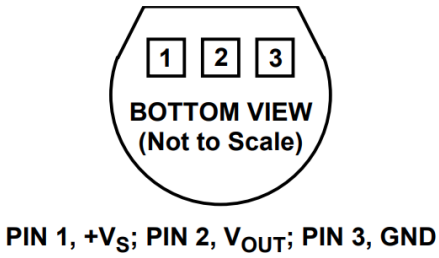

Figure 4: Pinout sensore TMP36

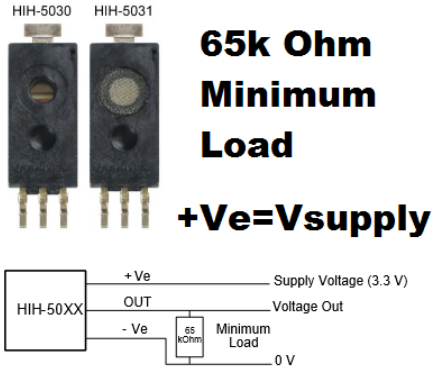

Figure 5: Pinout sensore HIH-5030

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

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); // Connect
    Wire.write(0xAC);   // Access config
    Wire.write(0x02);   // Set for continuos conversion
    Wire.endTransmission();

    Wire.beginTransmission(TEMP_DEV_ID);
    Wire.write(0xEE);   // Start conversion
    Wire.endTransmission();

    // Configurazione del sensore di luce digitale TSL2561
    Wire.beginTransmission(LIGHT_DEV_ID);
    Wire.write(0x00);   // Access Configuration options
    Wire.write(0x03);   // Sets power up
    Wire.endTransmission();

    Wire.beginTransmission(LIGHT_DEV_ID);
    Wire.write(0x81);   // Command to se timing
    Wire.write(0x02);   // Low Gain (1x), 402ms (default)
    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 necessaria per cambiare quale sensore si vuole visualizzare.

```
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");
            stato_precedente = 2;
        }
        break;
```

```
        case 3:{
            if (stato != stato_precedente) {
              tft.fillScreen(HX8357_BLACK);
              tft.setCursor(30,60);
              tft.println("Humidity");
            }
            float hum = AnalogHumidity(HUMIDITY_DEV_PIN);
            String txt = String(hum);
            tft.setCursor(60,120);
            tft.fillRect(60,120,480,320, HX8357_BLACK);
            tft.println(txt + "␣%");
            stato_precedente = 3;
        }
    }
    delay(500);
}
```

2.2 Codice - lettura sensori

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

Il sensore TMP36 restituisce in output una tensione proporzionale alla temperatura.

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 ottenere il valore di umidità corretto è necessario modificare la lettura del sensore in base alla temperatura utilizzando le formule fornite dal datasheet del componente

Listing 2: Lettura dal sensore HIH-5030

```
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, 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);   // Access LSB of ADC0
    Wire.endTransmission();
    Wire.requestFrom(DEV_ID2, 1);

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

    Wire.beginTransmission(DEV_ID2);
    Wire.write(0x8D);   // Access MSB of ADC0
    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

3 Terzo esperimento

In questa esperienza si vuole realizzare un misuratore di distanza utilizzando il sensore di distanza a ultrasuoni HC-SR04, il cui pinout è riportato in Figura 6. Si vuole utilizzare il sensore in modo "nativo", senza utilizzare librerie già scritte. Sono stati utilizzati i seguenti componenti:

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

Il circuito è alimentato mediante porta USB del PC, la quale eroga circa ($\sim 5V$). Il sensore HC-SR04 è alimentato con una tensione di $V_{CC} = 5$ V. Per garantire la sicurezza degli ingressi della scheda Arduino Due, che operano a una tensione di 3.3 V, si è realizzato un semplice partitore di tensione collegato all'uscita del sensore (ECHO). Il partitore di tensione riduce la tensione di uscita del sensore a un valore prossimo a 3.3 V, consentendo così all'Arduino Due di rilevare correttamente il segnale.

Figure 6: 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 HC-SR04

```
long distance(){
    long d = 0;
    long duration = 0;
    // Invio dell' inpulso di TRIG
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    // Lettura della durata dell' inpulso da ECHO
    duration = pulseIn(ECHO_PIN, HIGH);
    d = (duration * 100) / 5830;
    delay(25);
    return d;
}
```

La funzione integrata `pulseIn(pin,value)` ritorna il tempo in microsecondi in cui un determinato `pin` mantiene il livello `value`, legge quindi la durata di un inpulso su un pin e ne ritorna la lunghezza in microsecondi. In questo caso fa partire un timer quando il pin `ECHO_PIN` commuta a livello `HIGH` e interrompe il timer quando il pin ritorna a livello `LOW`, ritornando poi il valore del timer in microsecondi

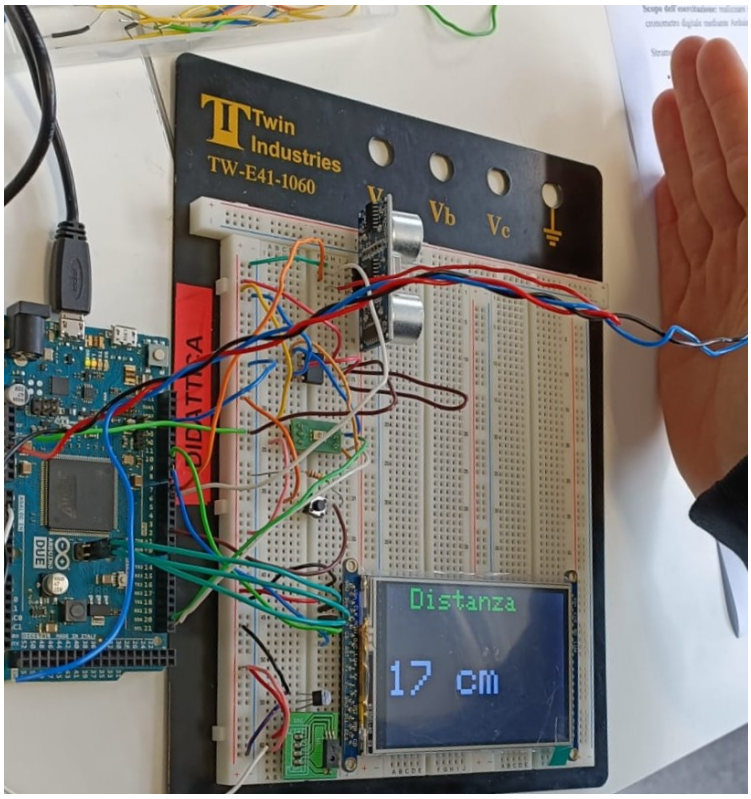

Figure 7: Prova del funzionamento del sensore