Istituto Professionale per L'industria L'Artigianato

“Antonio Guastaferro”

63074 SAN BENEDETTO DEL TRONTO (AP)

Classe : 4A_IPAI -- 5A_IPAI
A.S. : 2017-2018
Docenti : Tufoni Franco -- Enrico Ruggieri
Disciplina : Tecnologie elettriche-elettroniche, dell'automazione e applicazioni

Arduino

Applicazioni Avanzate

Open Source

Vers. 1.2 – 17-03-2018

Indice

Esercizio 1A - Data Logger Ambientale

Indice applicazioni

Dimensionare e programmare un circuito in grado di:

1) visualizzare in tempo reale su un display:
   a. Data, Ora, Temperatura, Umidità e Temperatura di Rugiada;
   b. Indice UV (Raggi Ultravioletti) e livello di protezione
2) ogni 30 secondi registrare i dati su una MicroSd;
3) riportare i dati acquisiti in grafico.

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

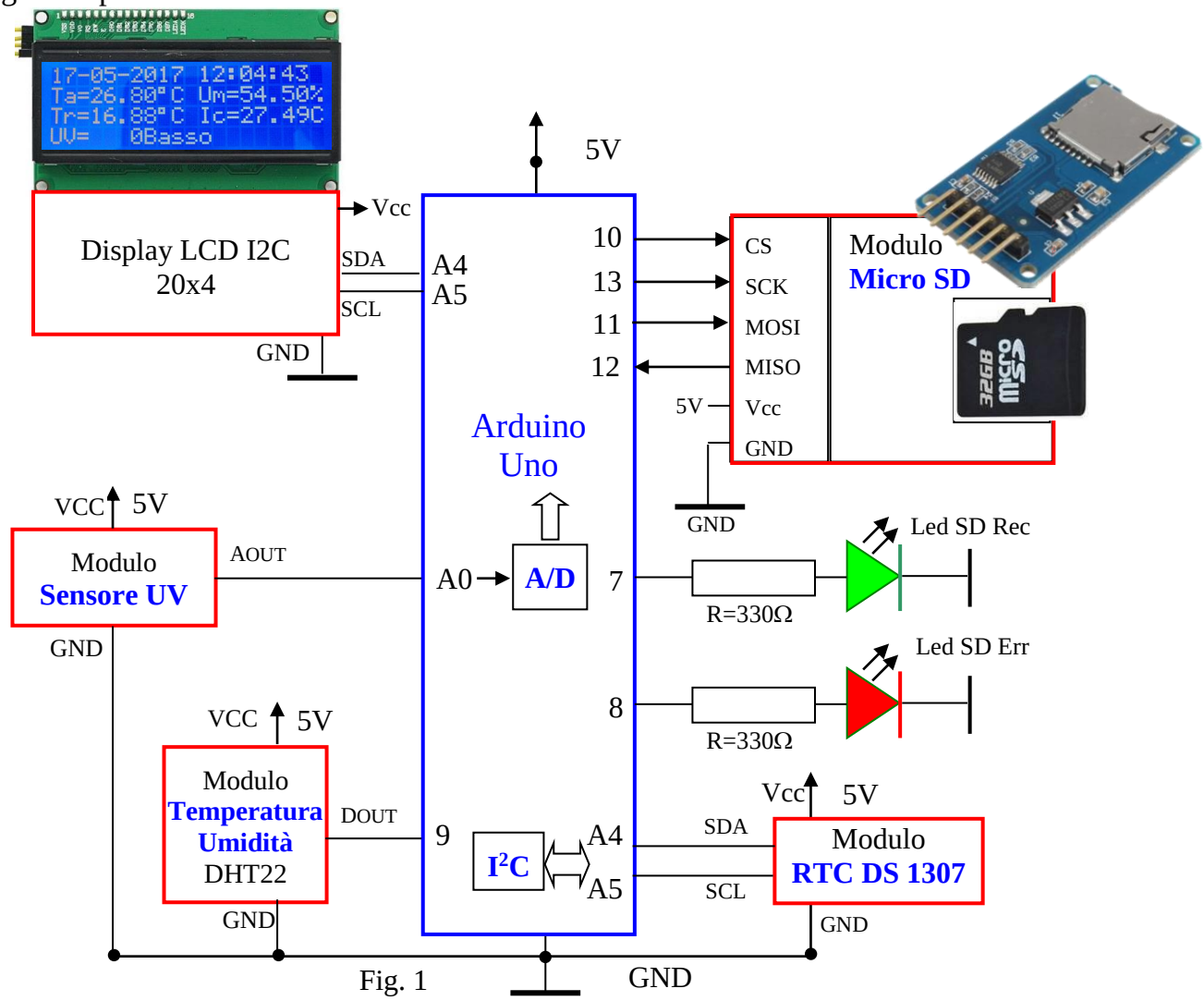


Fig. 1

Il progetto è suddiviso in una parte **Hardware** ed in una parte **Software**.

Hardware	Software
a) Arduino Uno (Sistema di sviluppo programmabile)	Programma/Sketch
b) Modulo sensore UV	
c) Modulo Temperatura Umidità DHT22	
d) Modulo RTC DS 1307	
e) Display LCD I2C 20x4	
f) Modulo MicroSD	
g) Gruppo LED	

Descrizione Data Logger

Il **data logger** oppure **datalogger** è un dispositivo elettronico digitale programmabile in grado di registrare dati provenienti da sensori (Temperatura, umidità, pressione, ecc) su supporti non volatili (Esempio memorie SD Card). Uno dei principali vantaggi di usare dei registratori di dati è la possibilità di memorizzarli in modo automatico su base giornaliera campionandoli nel tempo. Una volta attivati, i registratori di dati possono essere sistemati nel luogo opportuno e lasciati incustoditi a registrare per tutta la durata dell'acquisizione dei dati. Ciò permette un'accurata mappatura delle condizioni ambientali che vengono monitorate come ad esempio parametri esterni meteorologici.

Il sistema, tramite software memorizzato nella memoria programma del microcontrollore:

1. Legge la **Data** e l'**Ora** sul modulo **RTC DS 1307;**
2. legge la **Temperatura** e l'**Umidità** sul modulo **DHT22** e**,** tramite un algoritmo, calcola la **Temperatura di Rugiada;**
3. legge il valore di tensione analogica sul piedino Aout del **Modulo UV** e la trasforma nell'indice **UV** (0 ÷11);
4. visualizza in tempo reale i dati sul display **LCD;**
5. Ogni 30 secondi registra i dati (Data, Ora, Temperatura, Umidità, Temperatura di Rugiada, Indice UV) sulla microSD. Durante la fase di registrazione sul display compare il simbolo "**R**".
6. terminata la registrazione ripete il ciclo dal punto 1.

a) Arduino Uno (Sistema di sviluppo programmabile) Approfondimento in Appendice

Arduino UNO è una scheda basata sul microcontrollore ATmega328, dotata di 14 pin di input/output digitali,6 input analogici, un quarzo a 16MHz, un connettore USB, un jack per l'alimentazione, un connettore per la programmazione ICSP ed un pulsantino per il reset della scheda. Per utilizzare la Arduino UNO è necessario connettere la scheda ad un PC tramite uncavo USB oppure alimentare la scheda tramite un adattatore AC/DC o tramite una semplice batteria.

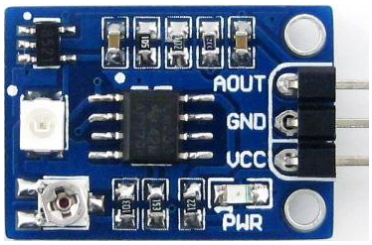
b) Modulo Sensore UV

Il Modulo Sensore UV basato sul diodo GUVA-S12SD consente di misurare la radiazione ultravioletta nel range 200nm-370nm (nanometri).

Nota: Le radiazioni UV coprono quella porzione dello spettro elettromagnetico con una lunghezza d'onda (λ) compresa tra 100 e 400 nanometri (nm) e si dividono in tre categorie:
UVC (100-280 nm) **UVB** (280-315 nm) **UVA** (315-400 nm). (Fig.2)

Fig.2

In Fig 3 è riportato lo schema elettrico suddiviso in quattro parti.



3a) Diodo GUVA-S12SD e circuito di condizionamento con l'amplificatore operazionale SGM8521.
3b) Amplificatore a guadagno regolabile con l'amplificatore operazionale LM358
3c) Diodo Power Led L2
3d) Connettore H1 (Vcc, GND, Aout)

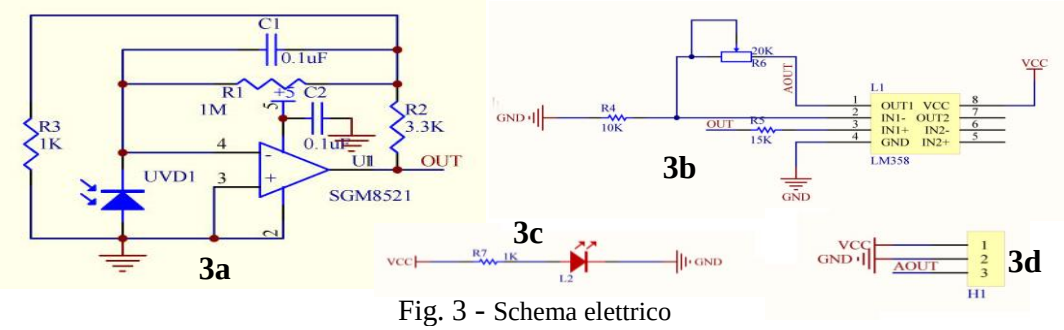


Fig. 3 - Schema elettrico

Tabella 1 -Specifiche tecniche	
Descrizione	**Specifiche**
Spettro UV	200nm-370nm
VCC	3.3V ~ 5.0V
GND:	Power Supply Ground
Temperatura	-20 °C ÷ + 85°C
AOUT	Uscita Analogica 0V ÷1,17V
Applicazioni	Tester Ultravioletti - Luce solare Lampada germicida

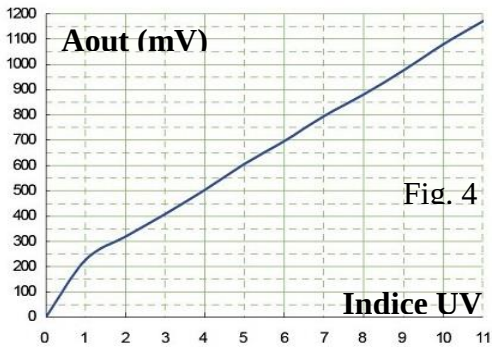


Fig. 4

Il modulo fornisce un'uscita analogica proporzionale all'intensità UV misurata (mW/cm^2) compatibile con l'indice internazionale UV= 0 ÷11.

Aout=0V ÷1,17V Indice UV= 0 ÷11

In Fig 4 e riportato il grafico tensione di uscita in funzione dell'indice UV.
Dal grafico si nota che la risposta non è perfettamente lineare

Il segnale analogico viene inviato all'ingresso A0 (Ch0 del Conv. A/D a 10 bit interno) e convertito in digitale (Fig. 5).

$A/D = \frac{Aout}{q}$ q=Risoluzione analogica del A/D (10bit) $q = \frac{Vfs}{2^{10}} = \frac{5}{1024} = 4{,}88mV$

Esempio:

Indice **UV=8**

Aout=881 mV

$$A/D = \frac{881 \cdot 10^{-3}}{4{,}88 \cdot 10^{-3}} = 180$$

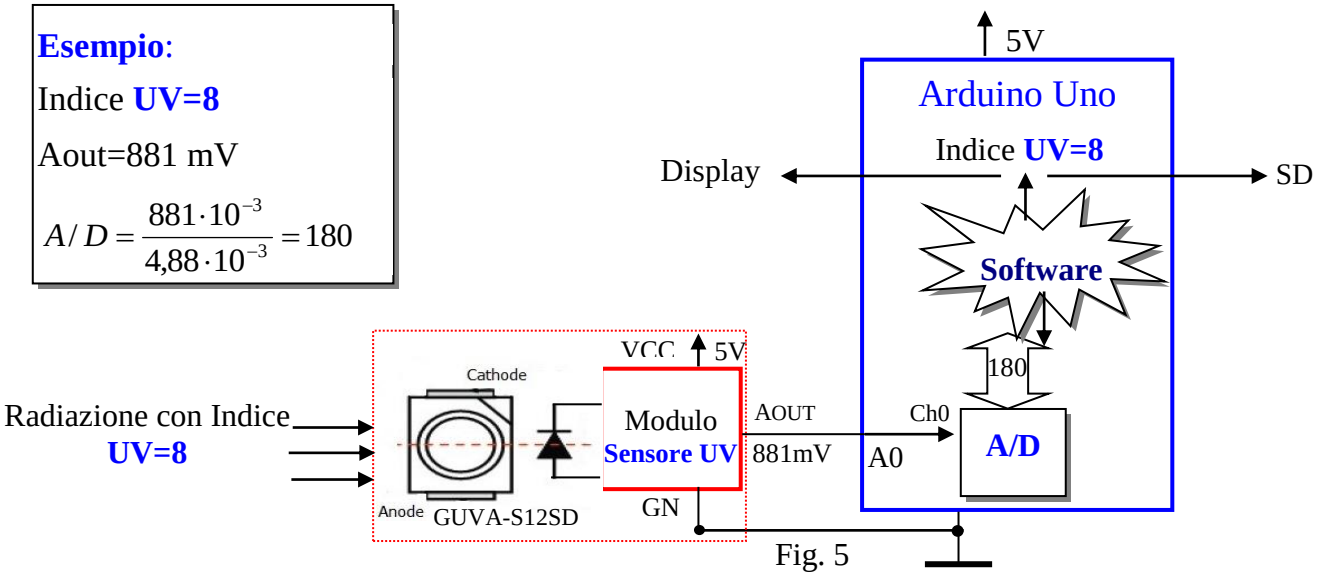


Fig. 5

Nella tabella 2 sono riportati i valori in funzione dell'indice UV con i rispettivi colori che indicano il livello di pericolosità.

Tabella 2												
Indice UV	0	1	2	3	4	5	6	7	8	9	10	11+
Aout(mV)	<50	227	318	408	503	606	696	795	881	976	1079	1170
A/D	<10	46	65	83	103	124	142	162	180	200	221	240
Livello	BASSO			MODERATO			ALTO		MOLTO ALTO			ESTREMO
Protezione	Nessuna protezione			Richiesta protezione					Protezione Extra			
	Esposizione al sole senza rischi			Rimanere all'ombra nelle ore attorno a mezzogiorno. Utilizzare maglietta, crema solare e cappello.					Evitare di uscire nelle ore attorno a mezzogiorno. Stare all'ombra Indispensabile maglietta, crema solare e cappello.			

c) Modulo Temperatura Umidità DHT22

Il sensore DHT22 permette di rilevare l'umidità relativa e la temperatura di un ambiente e di trasmetterla digitalmente attraverso un solo filo (oltre ai 2 necessari all'alimentazione) ad un microcontrollore

Il sensore di umidità è di tipo capacitivo, utilizza al massimo 2.5mA in fase di e permette di acquisire al massimo un campione ogni 2 secondi.

La misurazione dell'umidità varia da 0 a 100% con un'accuratezza tipica pari al ±2%.

Per quanto riguarda la temperatura invece il DHT22 ha un range di misurazione tra -40 e +80°C con un accuratezza di ±0.5°C grazie al sensore DS18B20 integrato al suo interno.

L'alimentazione può variare da 3.3 a 6Vquindi può essere usato sia con microcontrollori che lavorano a 5v sia con quelli a 3.3v.

VCC e GND servono rispettivamente per portare l'alimentazione al sensore, mentre il pin DATA serve per la comunicazione bidirezionale tra sensore e processore.

Una nota molto interessante è che il filo DATA che collega il sensore alla MCU (microcontrollore) può essere lungo fino a 20 metri. (si riduce a 1m se alimentato a 3.3V)

Il DHT22 non necessita di ulteriore componentistica esterna però, specialmente se si utilizzano fili lunghi, potrebbe essere consigliato l'utilizzo di un condensatore da 100nF tra i pin VCC e GND in prossimità del sensore per filtrare eventuali disturbi all'alimentazione.

In alcuni schemi è presente una resistenza detta "di Pull-up" connessa tra la VCC e il pin DATA. Questo per tenere alto lo stato del canale quando non è in corso una trasmissione dati. Il valore della resistenza tipicamente è di 4.7-5.1KΩ a seconda dell'applicazione.

Il DHT22 è molto versatile, infatti può essere impiegato ad esempio in sistemi di deumidificazione, automotive, domotica, monitoraggio ambientale e molti altri ambiti in cui è necessario un controllo elettronico di temperatura e umidità.

Caratteristiche tecniche

- Tensione di alimentazione: da 3 a 5 V e I / O
- Corrente massima: 2,5 mA durante la conversione (durante la richiesta di dati)
- Velocità di campionamento non superiore a 0,5 Hz (una volta ogni 2 secondi)
- Intervallo di temperatura: -40-80 °C risoluzione 0,1 °C errore <± 0,5 °C
- Intervallo di umidità: 0-100% di risoluzione RH0,1% di errore di UR ± 2% di umidità relativa

Comunicazione

La comunicazione avviene attraverso un solo filo ed è bidirezionale (pin2:Data)
Questo sensore, su sollecitazione di Arduino, trasmette sul pin Data (Uscita digitale) un treno di quaranta bit:

- 8 bit per indicare la parte intera del valore di umidità
- 8 bit per indicare la parte decimale dell'umidità (.00 nel modello DHT11)
- 8 bit per indicare la parte intera della temperatura
- 8 bit per indicare la parte decimale della temperatura (.00 nel modello DHT11)
- 8 bit per indicare il numero di controllo (per validare il valore dei precedenti 32 bit)

I dati di umidità e temperatura sono formati da 16bit ciascuno, quindi 32 bit di informazione.

Es:

Umidità = 65.8%RH in binario 00000010 10010010
Temperatura= 26.9%RH in binario 00000001 00001101

La trasmissione avviene a gruppi di 8 bit, quindi ci saranno 4 gruppi di 8 bit (32 bit)
Vengono aggiunti ulteriori 8 bit per verificare che la trasmissione sia avvenuta correttamente.
Questo numero viene calcolato sommando i 4 blocchi:

Verifica = 00000010 + 10010010 + 00000001 + 00001101 = 10100010
Quindi in totale 5 blocchi da 8 bit per un totale di 40 bit.

Sincronizzazione:

Processore porta a 0 lo stato del bus DATA per 800µs che significa "sensore, preparami i dati". Il sensore di risposta porta lo stato del bus a 0 per 80µs per dire "inizio trasmissione"
Poi il sensore inizia a trasmettere i dati di temperatura e umidità:

- Blocco 1: primi 8 bit riguardanti l'umidità, Es. 00000010
- Blocco 2: gli ultimi 8 bit riguardanti l'umidità Es. 10010010
- Blocco 3: primi 8 bit riguardanti la temperatura Es. 00000001
- Blocco 4: ultimi 8 bit riguardanti la temperatura Es. 00001101
- Blocco 5:8bit di checksum/verifica Es. 10100010

La gestione del modulo avviene tramite la libreria (**DHT.h**) che deve essere scaricata ed installata prima della compilazione del programma.

Temperatura di rugiada

Con punto di rugiada o temperatura di rugiada ("**dew point**") si intende la temperatura alla quale, a pressione costante, l'aria (o, più precisamente, la miscela aria-vapore) diventa satura di vapore acqueo. In meteorologia in particolare, indica a che temperatura deve essere portata l'aria per farla condensare in rugiada, senza alcun cambiamento di pressione. Se il punto di rugiada cade sotto 0 °C, esso viene chiamato anche punto di brina.
In estate essa ci fornisce un'idea immediata della sensazione di calore sul nostro organismo: temperature di rugiada superiori ai 17°C sono sintomo di una debole afa, quando invece superano i 21°C, l'afa comincia a diventare fastidiosa.
Inoltre il dew point da una rappresentazione anche di quello che è il "carburante" disponibile per lo sviluppo di temporali, più il dew point è elevato più gli eventuali fenomeni temporaleschi potranno essere intensi.
Nel programma il calcolo viene effettuato dalla funzione **dewPoint**, all'interno l'algoritmo che determina il valore della temperatura di rugiada in base al valore della temperatura e dell'umidità.

d) Modulo RTC DS 1307

Approfondimento in Appendice

Questo modulo è basato sul chip DS1307, orologio/calendario a basso consumo che implementa anche 56 bytes di memoria SRAM non volatile; il modulo RTC fornisce informazioni su secondi, minuti, ore, giorno, mese e anno. La data del fine mese è riconosciuta automaticamente per quei mesi con meno di 31 giorni, inclusi anche gli anni bisestili; l'orologio può operare sia in modalità 24-ore sia in modalità 12-ore con indicazione AM/PM.

Il modulo RTC utilizza il protocollo I2C per comunicare con Arduino tramite i pin analogici **A5** (SCL) e **A4** (SDA). Il DS1307 opera come slave sul bus I^2C ed è allocato all'indirizzo 0x68 (Nel linguaggio C si usa il suffisso 0x per indicare un numero esadecimale).

Impostazione RTC

L'impostazione del RTC avviane tamite le istruzioni:

```
if (!RTC.isrunning())
{
// following line sets the RTC to the date & time this sketch was compiled
RTC.adjust(DateTime(__DATE__, __TIME__));
}
```

Questa parte di programma deve essere inserita in void *setup ()* dopo l'istruzione *RTC.begin();*

La sequenza indicata, se il modulo non è settato (primo utilizzo), imposta nel RTC la Data e l'ora della compilazione.

e) Display LCD I2C 20x4

Approfondimento in Appendice

Modulo LCD 20x4 dispone di 4 righe e 20 colonne è basato sul controller HD44780; dotato di retroilluminazione blu, caratteri bianchi, regolazione del contrasto e di un'interfaccia di comunicazione I2C.

Il modulo è dotato di 4 pin (Alimentazione: **Vcc**, **Gnd**) e (Dati: **SDA**, **SCL**).

f) Modulo MicroSD

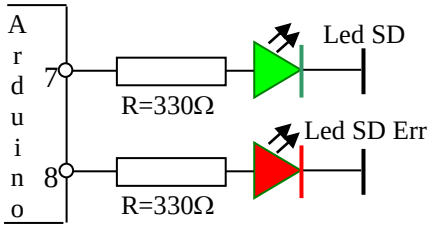
Approfondimento in Appendice

Il modulo MicroSD, tramite l'interfaccia SPI del microcontrollore, consente la lettura e scrittura dei files su una **MicroSD**. Per la gestione del modulo occorre la libreria **SD.h**, richiamabile con il comando **#include <SD.h>,** compresa nell'IDE di Arduino.

Il modulo può essere utilizzato in diversi modi e con diverse funzioni per la gestione dei files sulla scheda microSD.

g) Gruppo LED

Il gruppo LED ri riferisce al funzionamento della SD.



Led SD Err (Rosso) =On - Errore SD (SD non inserita)
=Off - SD inserita (OK)

Led SD (Verde) =On - Fase di registrazione (ogni 30 sec)
=Off - Non registra

Programma/Sketch

```
/* Data Logger Ambientale - Applicazione con Arduino e sensori ambientali
   Data - Ora - Temperatura - Umidità - Indice di calore
   Temperatura di Rugiada - Indice Raggi Ultravioletti
1. Visualizza in tempo reale su un display 20x4 i dati acquisiti
2. Registra ogni 30 secondi i dati acquisiti su una MicroSd.
*/
#include <Wire.h>
#include <RTClib.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,20,4);
#include <SD.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTPIN 9
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);
const int chipSelect = 10;
RTC_DS1307 RTC;
#define DS1307_I2C_ADDRESS 0x68

File myFile;
char ora[9];
char data[12];
String tutruv, B;
byte uv [13]={0,46,65,83,103,124,142,162,180,200,221,240,254};
byte iuv;
byte pulsanteset = 8;
byte ledrec=7;
long tr1,tr2;
int stringStart, stringStop = 0;
int scrollCursor = 16;
int screenWidth = 16;
int screenHeight = 2;
double dewPoint(double celsius, double humidity)  // Calcolo temperature di rugiada
{
	double A0= 373.15/(273.15 + celsius);
	double SUM = -7.90298 * (A0-1);
	SUM += 5.02808 * log10(A0);
	SUM += -1.3816e-7 * (pow(10, (11.344*(1-1/A0)))-1) ;
	SUM += 8.1328e-3 * (pow(10,(-3.49149*(A0-1)))-1) ;
	SUM += log10(1013.246);
	double VP = pow(10, SUM-3) * humidity;
	double T = log(VP/0.61078);   // temp var
	return (241.88 * T) / (17.558-T);
}

void setup ()
```

```
{
  dht.begin();
  pinMode (pulsanteset, INPUT_PULLUP);
  pinMode (ledrec, OUTPUT);
  Wire.begin();
  RTC.begin();

  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(7);
Wire.write(B10010000); // sends 0x10 (hex) 00010000 (binary) to control register - turns on square wave
Wire.endTransmission();
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Data Logger");
lcd.setCursor(0,1);
lcd.print("SD Card: ");
 if (!SD.begin(chipSelect))
  {
    lcd.setCursor(9,1);
    lcd.print("Errore");
  }
  else
  {
  lcd.setCursor(9,1);
  lcd.print("OK    ");
  }
delay (2000);
  lcd.clear();
 }
void loop ()
{
 tr2=millis()-tr1;
  boolean set = digitalRead (pulsanteset);
 if (set == LOW)
 {
 RTC.adjust(DateTime(__DATE__, __TIME__)); // Imposta Data e Ora del Computer
 }
  DateTime now = RTC.now();  // Legge i dati dal modulo RTC DS1307
// Composizione stringa data
  sprintf(data, "%02d-%02d-%d", now.day(),now.month(), now.year());
// Composizione stringa ora
  sprintf(ora,  "%02d:%02d:%02d", now.hour(), now.minute(), now.second());
   float h = dht.readHumidity();     // Lettura Umidità
  float t = dht.readTemperature();// Lettura Temperatura

// Calcolo temperatura di rugiada
```

```
float tr=dewPoint(dht.readTemperature(), dht.readHumidity());
  float hic = dht.computeHeatIndex(t, h, false); // Calcolo Indice di Calore
  int vuv=analogRead(A0);
  for (byte x=0;x<12;x++)
  {
    if (vuv>=uv[x] && vuv<uv[x+1]){iuv=x;}
  }
  if (iuv>=0 && iuv<=2) {B="Basso     ";}
  if (iuv>2 && iuv<=5)  {B="Moderato  ";}
  if (iuv>5 && iuv<=7)  {B="Alto      ";}
  if (iuv>7 && iuv<=10) {B="Molto Alto";}
  if (iuv>10)           {B="Estremo   ";}
   // Visualizza i dati acquisiti ed elaborati sul display LCD 20x4
  lcd.setCursor(0,0);
  lcd.print(data);
  lcd.setCursor(11,0);
  lcd.print(ora);
  lcd.setCursor(0,1);
  lcd.print("Ta=");
  lcd.setCursor(3,1);
  lcd.print(t,2);
  lcd.print(char(223));
  lcd.print("C");
  lcd.setCursor(11,1);
  lcd.print("Um=");
  lcd.setCursor(14,1);
  lcd.print(h,2);
  lcd.print("%");
  lcd.setCursor(0,2);
  lcd.print("Tr=");
  lcd.print(tr,2);
  lcd.print(char(223));
  lcd.print("C");
  lcd.setCursor(11,2);
  lcd.print("Ic=");
  lcd.print(hic,2);
 // lcd.print(char(223));
  lcd.print("C");
  lcd.setCursor(0,3);
  lcd.print("UV=   ");
  lcd.print(iuv,DEC);
  lcd.setCursor(7,3);
  lcd.print(B);
  if (tr2>30000)
{
   digitalWrite(ledrec,HIGH);
   lcd.setCursor(10,0);
   lcd.print("R");
```

```
  // Registra i dati sulla memoria SD
  myFile = SD.open("FileDati.csv", FILE_WRITE);
  myFile.print(data);
  myFile.print("   ");
  myFile.print(ora);
  myFile.print("   ");
  myFile.print(t,2);
  myFile.print("   ");
  myFile.print(h,2);
  myFile.print("   ");
  myFile.print(tr,2);
  myFile.print("   ");
  myFile.print(hic,2);
  myFile.print("   ");
  myFile.print(iuv,DEC);
  myFile.print("   ");
  myFile.println(B);
  myFile.close();
  delay (1500);
  lcd.setCursor(10,0);
  lcd.print(" ");
  tr1=millis();
  digitalWrite(ledrec,LOW);
}
 delay(850);
}
```

In fig. 6 è riportato un estratto dati registrati nella microSD, in fig.7 un estratto dati elaborati con Excel.

	Data	Ora	T(°C)	U(%)	Tr(°C)
2	---	---	---	---	---
3	28-01-2017	18:00	19	32	1.95
4	28-01-2017	18:00	19	32	1.95
5	28-01-2017	18:00	19	32	1.95
6	28-01-2017	18:01	19	32	1.95
7	28-01-2017	18:01	19	32	1.95
8	28-01-2017	18:01	19	32	1.95
9	28-01-2017	18:01	19	32	1.95
10	28-01-2017	18:02	19	32	1.95
11	28-01-2017	18:02	19	32	1.95
12	28-01-2017	18:02	19	32	1.95
13	28-01-2017	18:03	19	32	1.95
14	28-01-2017	18:03	19	32	1.95
15	28-01-2017	18:03	19	31	1.50
16	28-01-2017	18:04	19	32	1.95
17	28-01-2017	18:04	19	31	1.50
18	28-01-2017	18:04	19	31	1.50
19	28-01-2017	18:05	19	32	1.95
20	28-01-2017	18:05	19	32	1.95
21	28-01-2017	18:05	19	32	1.95
22	28-01-2017	18:06	18	29	-0.29
23	28-01-2017	18:06	19	28	0.09
24	28-01-2017	18:06	18	29	-0.29

Fig. 6: Estratto dati registrati nella microSD

	A	B	C	D	E
1	Data	Ora	T(°C)	U(%)	Tr(°C)
2	---	---	---	---	---
3	28/01/2017	18:00	19	32	1,95
4	28/01/2017	18:00	19	32	1,95
5	28/01/2017	18:00	19	32	1,95
6	28/01/2017	18:01	19	32	1,95
7	28/01/2017	18:01	19	32	1,95
8	28/01/2017	18:01	19	32	1,95
9	28/01/2017	18:01	19	32	1,95
10	28/01/2017	18:02	19	32	1,95
11	28/01/2017	18:02	19	32	1,95
12	28/01/2017	18:02	19	32	1,95
13	28/01/2017	18:03	19	32	1,95
14	28/01/2017	18:03	19	32	1,95
15	28/01/2017	18:03	19	31	1,50
16	28/01/2017	18:04	19	32	1,95
17	28/01/2017	18:04	19	31	1,50
18	28/01/2017	18:04	19	31	1,50
19	28/01/2017	18:05	19	32	1,95
20	28/01/2017	18:05	19	32	1,95
21	28/01/2017	18:05	19	32	1,95
22	28/01/2017	18:06	18	29	- 0,29
23	28/01/2017	18:06	19	28	0,09
24	28/01/2017	18:06	18	29	- 0,29

Fig. 7: Estratto dati elaborati con Excel

In fig. 8 è riportato il grafico di tutti i dati acquisiti (1957 record):

1. primo dato 28-01-2017 alle ore 18:00
2. ultimo dato 29-01-2017 alle ore 04:10
3. Esempio Punto A: 28/01/2017 ore 23:50 T=18°C, U=43%, Trugiada=5,35°C

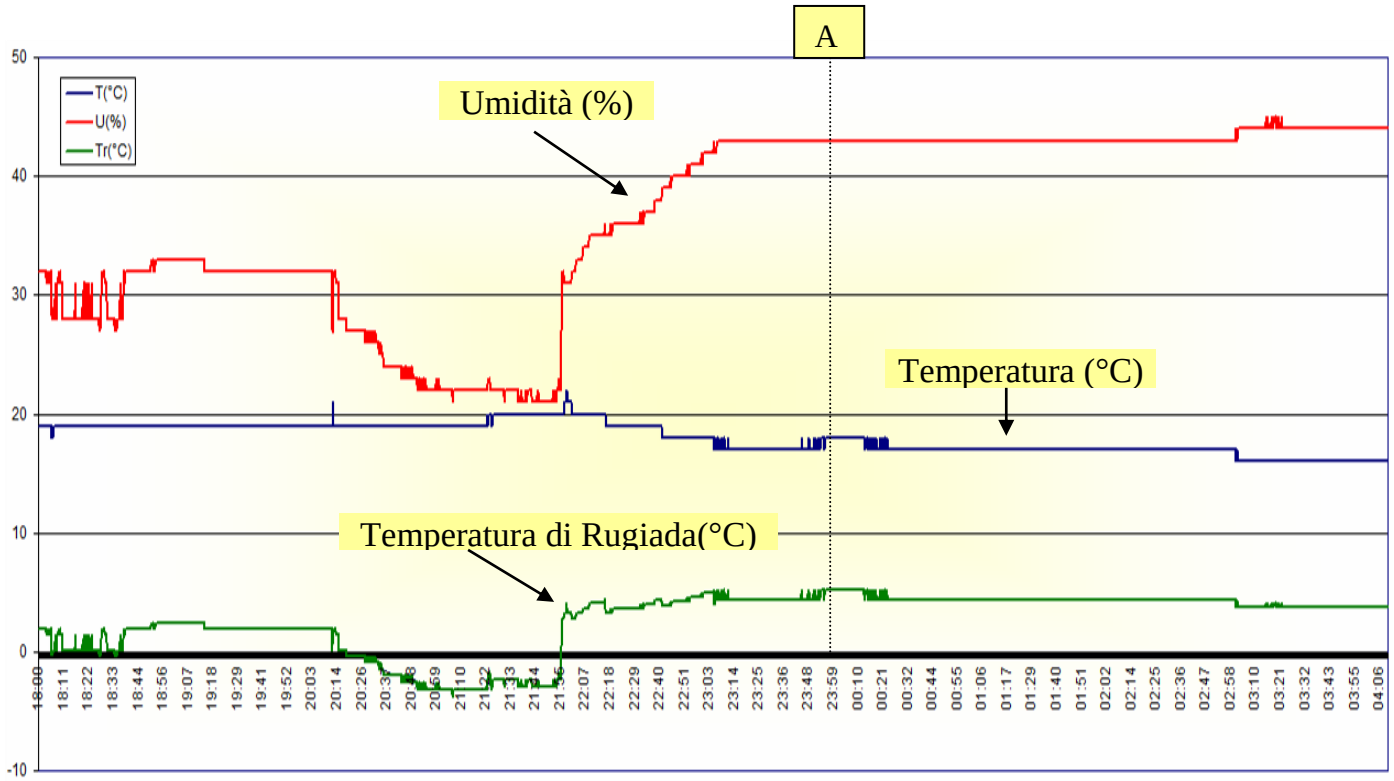


Fig. 8: Grafico di tutti i dati acquisiti

Esercizio 2A - Misuratore distanza con il modulo HC SR04

Dimensionare e programmare un circuito, basato sul modulo ad ultrasuoni HC SR04, in grado di visualizzare sul display 20x4 la distanza in cm tra il modulo ed un oggetto.
Il sistema deve prevedere la correzione della velocita del suono in base alla temperatura e all'umidita dell'ambiente in cui viene effettuata la misura.

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

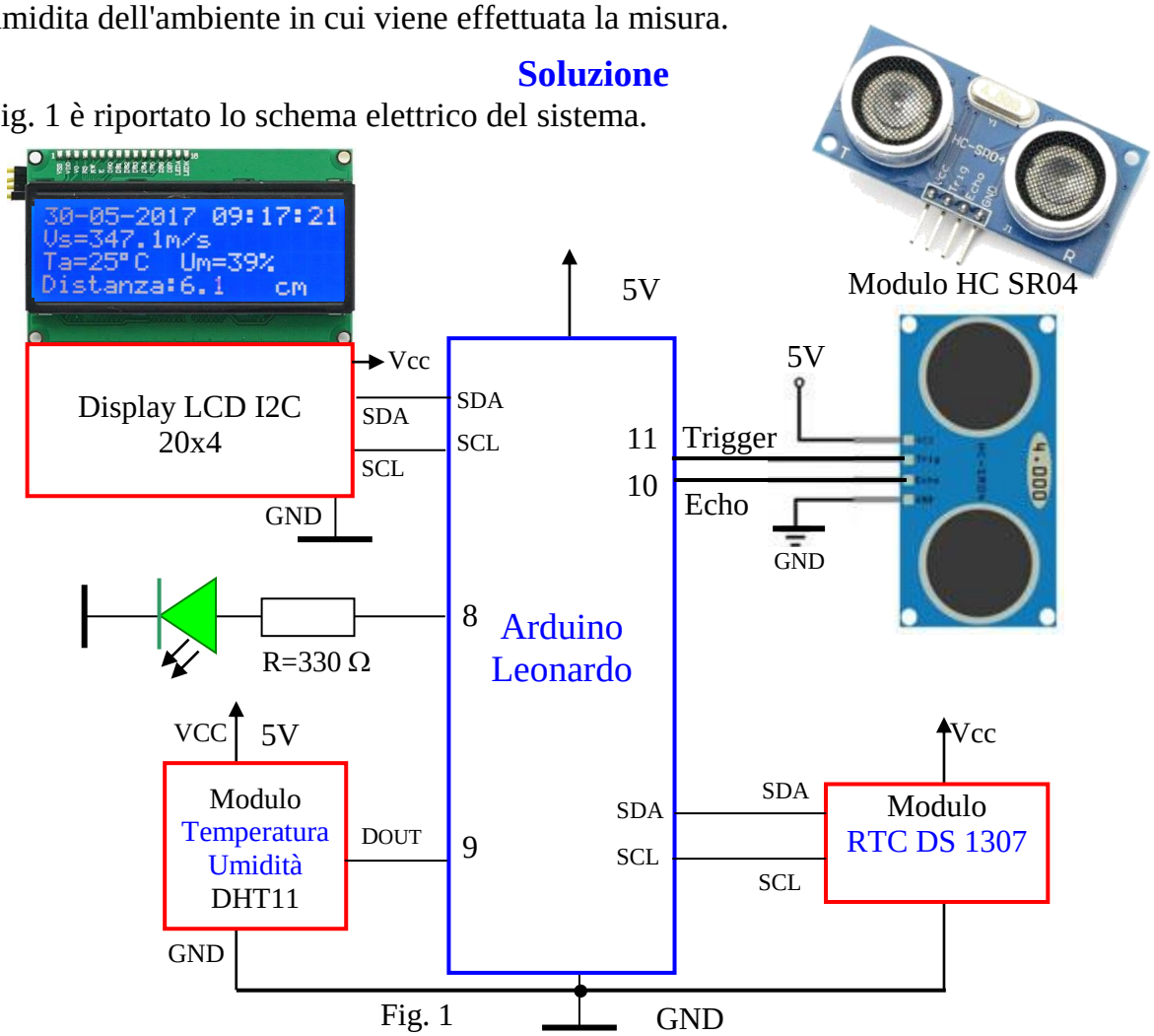


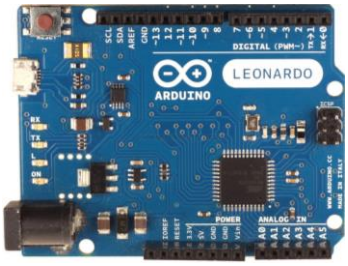
Fig. 1

Il progetto è suddiviso in una parte **Hardware** ed in una parte **Software**.

Hardware	Software
a) Arduino Leonardo (Sistema di sviluppo programmabile)	**Programma/Sketch**
b) Modulo HC-SR04	
c) Display LCD I2C 20x4	
d) Modulo RTC DS 1307	
e) Modulo Temperatura Umidità DHT11	
f)	

a) Arduino Leonardo (Sistema di sviluppo programmabile) Approfondimento in Appendice

Arduino Leonardo è una scheda basata sul microcontrollore ATmega32U4.



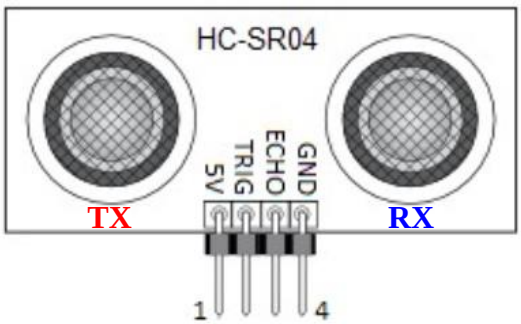
Rispetto ai chip della UNO e della MEGA2560, questo integra il convertitore seriale/USB in hardware per cui può essere collegato direttamente al computer senza microcontrollori intermedi.
Questa caratteristica permette alla scheda di essere riconosciuta dal sistema operativo come una periferica HID (Human Interface Device) tipo un mouse o una tastiera: la LEONARDO ha librerie specifiche che permettono di poter simulare appunto questi dispositivi in modo da poter spedire un tasto al sistema oppure muovere il puntatore del mouse o far leggere un click.

b) Modulo HC-SR04

Questo modulo, con tecnologia ad ultrasuoni, fornisce un metodo semplice per la misurazione di distanze, dove serve un'alta precisione unita ad un elevato range di misura, è capace di misurare una distanza compresa tra 2 cm e 400 cm con una precisione di 3 mm.

Il modulo opera usando la medesima tecnica di rilevamento utilizzata, in natura, dai pipistrelli.
E' ampiamente utilizzato in applicazioni di robotica, sistemi di sicurezza, misuratore di livello nei serbatoi o in sostituzione di unità ad infrarossi, rispetto alle quali offre performance più elevate.
Il sensore HC-SR04 è costituito da una scheda, che presenta nella sua parte posteriore un sofisticato circuito elettronico basato su un microcontrollore della ST (In Fig. 2 lo schema elettrico), e nella parte anteriore sono presenti un quarzo e due cilindri metallici, i trasduttori ad ultrasuoni. Uno di questi invia ultrasuoni (TX) che rimbalzano contro ad un qualunque oggetto posto di fronte ad esso, ed entrano di ritorno nell'altro cilindro (RX).
Il modulo dispone di quattro terminali di collegamento (Connettore J1, Fig 2) due di alimentazione (Vcc, GND) e due di controllo (Trigger, Echo). In tabella sono riportate le caratteristiche tecniche.

Caratteristiche tecniche	
Tensione di lavoro	5 Vdc
Corrente assorbita	>2 mA
Frequenza di lavoro	40 Khz
Distanza max	400 cm
Distanza min	2 cm
Risoluzione	3 mm
Angolo di misura	15°
Ingresso	Trigger 10µs Impulso TTL
Uscita	Echo segnale PWM TTL.
Dimensioni	45,5 x 20,5 x 15,3 mm



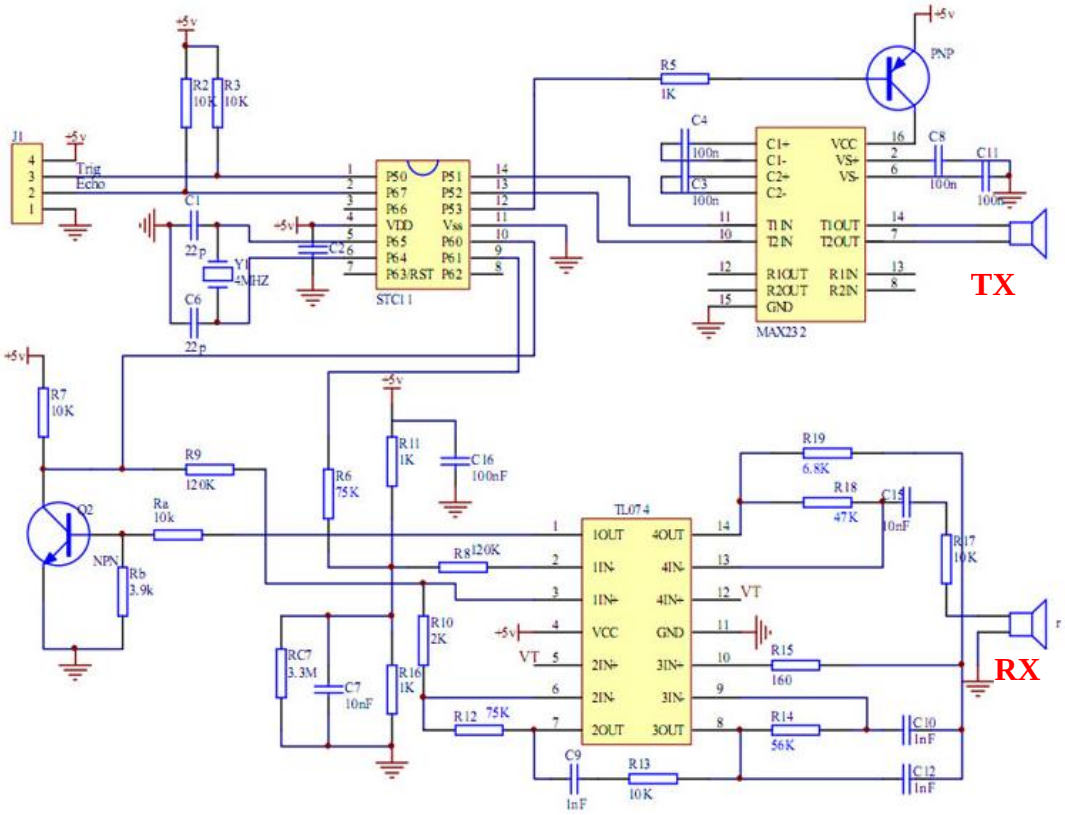
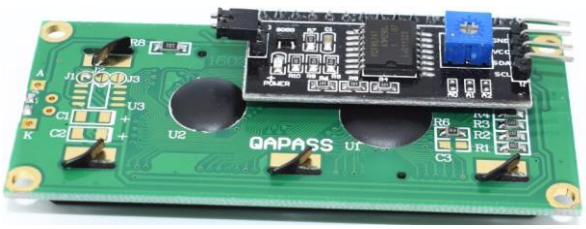


Fig.2 - Schema elettrico del modulo HC SR04

c) Display LCD I2C 20x4

Approfondimento in Appendice

Modulo LCD 20x4 dispone di 4 righe e 20 colonne è basato sul controller HD44780; dotato di retroilluminazione blu, caratteri bianchi, regolazione del contrasto e di un'interfaccia di comunicazione I2C.



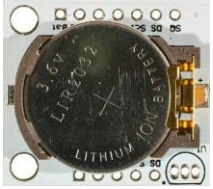
GND
VCC
SDA → A4 Arduino
SCL → A5

Il modulo è dotato di 4 pin (Alimentazione: **Vcc**, **Gnd**) e (Dati: **SDA**, **SCL**).

d) Modulo RTC DS 1307

Approfondimento in Appendice

Questo modulo è basato sul chip DS1307, orologio/calendario a basso consumo che implementa anche 56 bytes di memoria SRAM non volatile; il modulo RTC fornisce informazioni su secondi, minuti, ore, giorno, mese e anno. La data del fine mese è riconosciuta automaticamente per quei mesi con meno di 31 giorni, inclusi anche gli anni bisestili; l'orologio può operare sia in modalità 24-ore sia in modalità 12-ore con indicazione AM/PM.



Il modulo RTC utilizza il protocollo I2C per comunicare con Arduino tramite i pin analogici **A5** (SCL) e **A4** (SDA). Il DS1307 opera come slave sul bus I^2C ed è allocato all'indirizzo 0x68 (Nel linguaggio C si usa il suffisso 0x per indicare un numero esadecimale).

Impostazione RTC

In appendice, oppure Es 1A, sono riportate le modalità per l'impostazione iniziale del RTC.

Funzionamento: procedura di misura

Sequenza di funzionamento (Fig. 3 e Fig. 4):

1. Si invia tramite il Pin 11 di Arduino un impulso di Trigger della durata di 10 µs.
2. Sul fronte di discesa dell'impulso di Trigger il Modulo tramite la capsula TX invia un pacchetto di 8 impulsi alla frequenza di 40 Khz (ultrasuoni) (invia un ping).
3. Sul fronte di discesa dell'ultimo impulso il terminale Echo, collegato al pin 10 di Arduino, si porta a livello alto.
4. Se il segnale emesso da TX incontra un ostacolo torna indietro e viene captato dalla capsula RX.
5. Quando la capsula RX capta il segnale riflesso il segnale Echo torna a livello basso.
6. La durata dell'impulso di Echo si chiama **tempo di volo** (µs) e rappresenta il tempo di viaggio (andata e ritorno) del segnale (alla velocità del suono) impiegato per percorrere lo spazio tra le capsule ed l'ostacolo.
7. Dopo 30 ms si considera che non sia stato incontrato nessun ostacolo, per sicurezza si aspettano 50-60 ms per far si che non vi siano interferenze con la misura successiva.

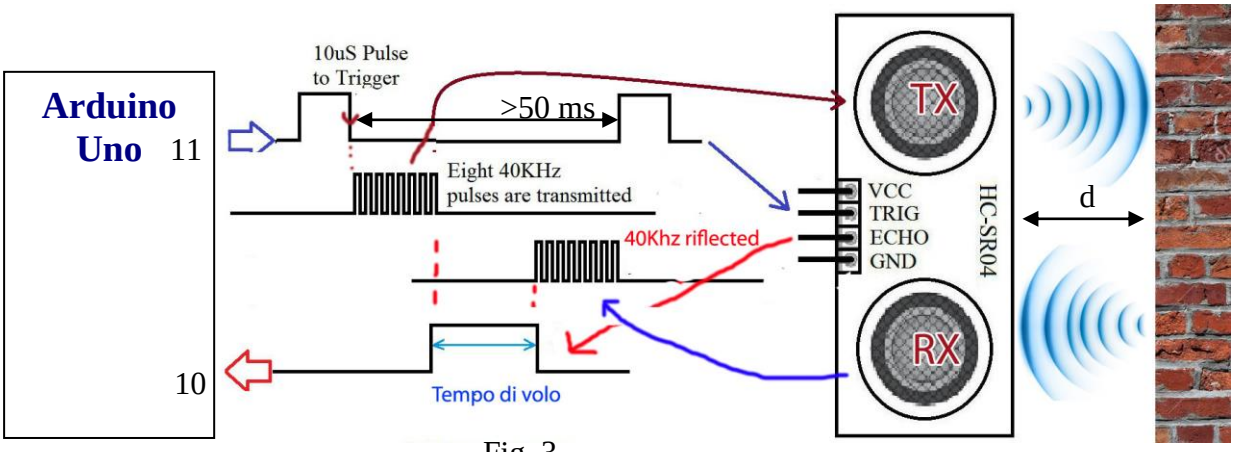


Fig. 3

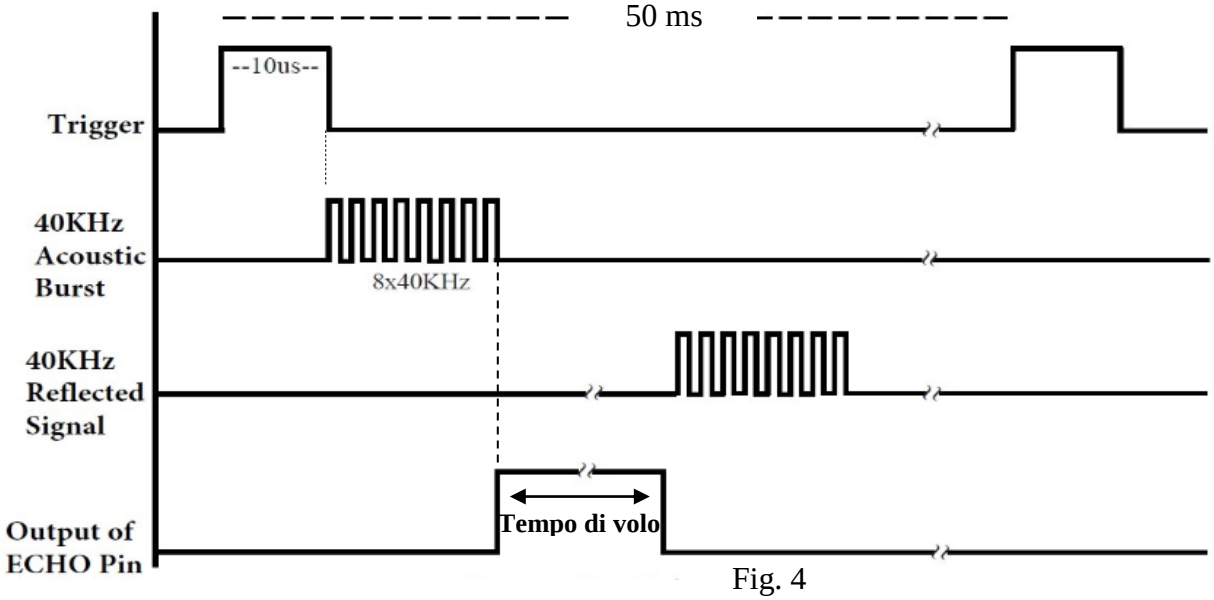


Fig. 4

Conoscendo la velocità del suono (343,8 m/s a 20°C) e sapendo che il "viaggio" dell'onda sonora è stato il doppio della distanza tra il modulo e l'ostacolo (l'onda è andata dal generatore (TX) all'ostacolo e da quì è tornata al sensore (RX)) la distanza è derivata dalla seguente formula:

$$d = \frac{Vs \bullet Tv}{2}$$

dove:
d= Distanza in metri tra il modulo HC-SR04 e l'ostacolo
Tv= Tempo di volo in secondi
Vs= Velocità del suono (343,8 m/s a 20°C)

Il modulo restituisce il tempo di volo in µs, per una semplicità di lettura conviene avere la distanza in cm, si trasforma la velocità del suono da m/s a cm/µs.
L'espressione per il calcolo della distanza in cm diviene:

$$d = \frac{343{,}8 \cdot 10^{-2} \bullet Tv}{10^{6} \bullet 2} \qquad d = \frac{0{,}03438 \bullet Tv}{2} \text{ [cm]}$$

La velocità del suono in aria varia con la temperatura e l' umidità.
Pertanto, al fine di calcolare con precisione la distanza, occorre prendere in considerazione la temperatura ambiente e l'umidità locale.
Nello schema di fig.1 la Temperatura e l'Umidità vengono rilevate dal modulo DHT11 (per il suo funzionamento si veda l'Es. 33).
La formula per la velocità del suono in aria in funzione della temperatura e l'umidità è:

Vs= 331,45 + (0,606 x T) + (0,0124 x H)

Vs = velocità del suono in metri al secondo (m /s)
331,45 = velocità del suono (in m/s) a 0 °C e 0% di umidità
T = temperatura in °C
H =% di umidità (umidità relativa)

Ad esempio, a 20 °C e 50% di umidità il suono viaggia ad una velocità di:
Vs = 331,45 + (0,606 x 20) + (0,0124 x 50) Vs= 344,19 m/s.
Per la conversione da m/s in a cm/µs si divide per 10^4
Vs= 344,19 m/s. Vs=0,034419 cm/µs
Nell'equazione sopra la temperatura ha un'influenza maggiore rispetto l'umidità.

Attenzione: il sensore intercetta i segnali di ritorno con un "angolo di visuale" di circa 15 gradi per cui intercetta anche l'eventuale segnale di ritorno proveniente dal pavimento, dal soffitto o da eventuali pareti laterali. Per ottenere una valida misurazione bisogna quindi tenere il modulo ad una altezza da terra o da ostacoli laterali sufficiente ad evitare interferenze. In linea di massima puo' essere utilizzata la seguente formula per calcolare la posizione del sensore in funzione della distanza massima misurabile.

posizione del sensore = distanza massima misurabile * 0,26 (0,26 è la tangente di 15°)

Questo significa che se vogliamo "vedere" ostacoli ad una distanza massima di un metro dobbiamo tenere il sensore ad una distanza di almeno 26 centimetri da terra o da eventuali pareti.

Programma/Sketch

Per questa soluzione scaricare ed installare la libreria NewPing.

Link Libreria: http://www.ipsiaplcarduinolab.altervista.org/documenti/librerie/NewPing_v1.8.zip

Le funzioni della libreria:

Funzione	Descrizione
sonar.ping()	Manda un ping e determina il tempo di echo come risultato
sonar.ping_in()	Manda un ping e determina la distanza in inches (numero intero)
sonar.ping_cm()	Manda un ping e determina la distanza in cm (numero intero).
sonar.ping_median(iterations)	Effettua un multiplo ping (default=5), e restituisce la media in microsecondi dei risultati ottenuti sui singoli ping.
sonar.convert_in(echoTime)	Converte il tempo di echo da µS a pollici
sonar.convert_cm(echoTime)	Converte il tempo di echo da µS in cm
sonar.ping_timer(function)	Manda un ping e chiama la funzione di test per verificare se un ping è completo
sonar.check_timer()	Check if ping has returned within the set distance limit
NewPing::timer_us(frequency, function)	Chiama la funzione "function" ogni "frequency" microsecondi
NewPing::timer_ms(frequency, function)	Chiama la funzione "function" ogni "frequency" milliecondi
NewPing::timer_stop()	Ferma il timer

```
/* ----------------------------------------------------------------------------------------
- Misuratore distanza con il modulo HC SR04
- Applicazione con Arduino e e sensore ad ultrasuoni HC SR04
- Gestione modulo tramite la libreria NewPing
- correzione della velocità del suono in base alla temperatura ed all'umidità.
- Visualizzazione sul display 20x4: - Distanza in cm, Velocità del suono, Temperatura - Umidità
---- IPSIA - Antonio Guastaferro - San Benedetto del Tronto
----- Vers 2.0  - A.S.2017-2018 - CL:5AIPAI
----------------------------------------------------------------------------------------
*/
#include <NewPing.h>
#include <Wire.h>
#include <RTClib.h>
RTC_DS1307 RTC;
#define DS1307_I2C_ADDRESS 0x68
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,20,4);
#include <dht11.h>
#define TRIGGER_PIN 11 // Pin di Arduino collegato al pin "TRIGGER" del sensore a
ultrasuoni
#define ECHO_PIN 10 // Pin di Arduino collegato al pin "ECHO" del sensore a ultrasuoni
#define MAX_DISTANZA 400 // Massima distanza del ping (cm) - Massima distanza=400
cm
dht11 DHT11;
#define DHT11PIN 9
```

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANZA); // Setup del sensore (pin e massima distanza)
char ora[9];
char data[12];
int pulsanteset = 7;
byte buzzerled=8;

void setup()
{
pinMode (pulsanteset, INPUT_PULLUP);
pinMode (buzzerled, OUTPUT);
Wire.begin();
RTC.begin();
Wire.beginTransmission(DS1307_I2C_ADDRESS);
Wire.write(7);
Wire.write(B10010000); // sends 0x10 (hex) 00010000 (binary) to control register - turns on square wave
Wire.endTransmission();
lcd.init();
lcd.backlight();
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Misura Distanza");
lcd.setCursor(0,1);
lcd.print("Ultrasuoni: SR04");
delay (2000);
lcd.clear();

}

void loop()
{
boolean set = digitalRead (pulsanteset);
if (set == LOW)
{
RTC.adjust(DateTime(__DATE__, __TIME__)); // Imposta Data e Ora del Computer
}
DateTime now = RTC.now(); // Legge i dati dal modulo RTC DS1307
sprintf(data, "%02d-%02d-%d", now.day(),now.month(), now.year()); // Composizione stringa data
sprintf(ora, "%02d:%02d:%02d", now.hour(), now.minute(), now.second()); // Composizione stringa ora
int chk = DHT11.read(DHT11PIN);
int h = DHT11.humidity;
//int t = DHT11.temperature;
int t=18;
float vs = 331.45 + (0.606 * t) + (0.0124 * h); //Velocità del suono in m/s - Correzione con Temp. e Umidità
```

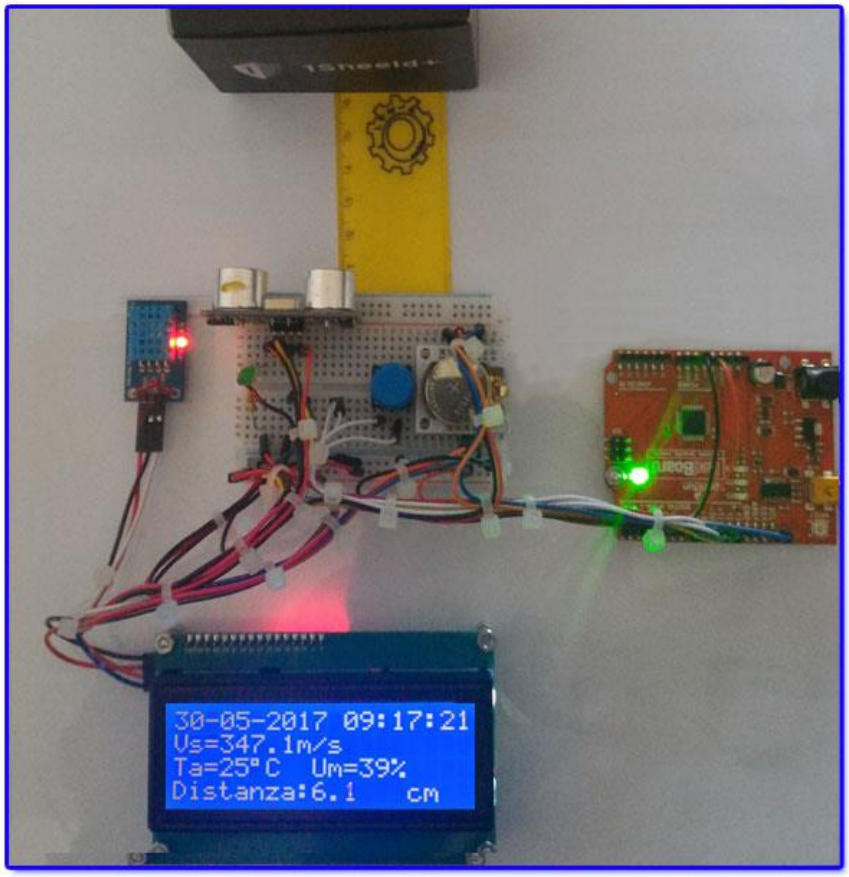
```
unsigned int tempo= sonar.ping_median(40); // Invia un ping multiplo (40 ping),restituisce il
tempo medio di echo in microsecondi
String vstu="Vs="+String(vs,1)+"m/s";// Composizione stringa dati (Velocità del suono e
Temp.)
float d=(vs*tempo/2.0)/10000.0;

lcd.setCursor(0,0);
lcd.print(data);
lcd.setCursor(11,0);
lcd.print(ora);
lcd.setCursor(0,1);
lcd.print(vstu);
lcd.setCursor(0,2);
lcd.print("Ta=");
lcd.setCursor(3,2);
lcd.print(t,DEC);
lcd.print(char(223));
lcd.print("C");
lcd.setCursor(9,2);
lcd.print("Um=");
lcd.setCursor(12,2);
lcd.print(h,DEC);
lcd.print("%");
lcd.setCursor(0,3);
lcd.print("Distanza: ");
lcd.setCursor(9,3);
lcd.print(d,1);
lcd.setCursor(15,3);
lcd.print("cm");
lcd.setCursor(19,3);
lcd.print(" ");

if (d>=10.0)
{
digitalWrite(buzzerled,HIGH);
lcd.setCursor(19,3);
lcd.print("!");
}
else
{
digitalWrite(buzzerled,LOW);
lcd.setCursor(19,3);
lcd.print("%");
}
delay(50);// Ritardo 50ms tra 2 ping (20 ping al secondo). 29ms è il ritardo minimo ammesso
tra 2 ping.
}
```

Conclusioni

Dai test effettuati risulta una precisione pari all'accuratezza del modulo (3 mm).
Non è possibile fare misurazioni con precisione dell'ordine del mm.



Esercizio 3A – Automazione Irrigazione Terreno

Dimensionare e programmare un circuito basato sul modulo Sensore umidità del terreno in grado di:

1. Rilevare e visualizzare su un display Lcd 16x2 l'umidità rilevata dal sensore;
2. se l'umidità è inferiore al 20%:
   a. visualizzare "Asciutto", accendere un Led Rosso e attivare un segnale acustico
   b. attivare una pompa per l'irrigazione goccia a goccia;
3. se l'umidità è compresa tra il 20% e il 60%
   a. visualizzare "Umido" e far lampeggiare il Led verde (situazione ottimale)
4. se l'umidità è superiore al 60%
   a. visualizzare "Bagnato" e far lampeggiare i led rosso e verde

Nei punti 2 e 3 la pompa deve essere disattivata, inoltre deve essere previsto un pulsante per il test del sistema.

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

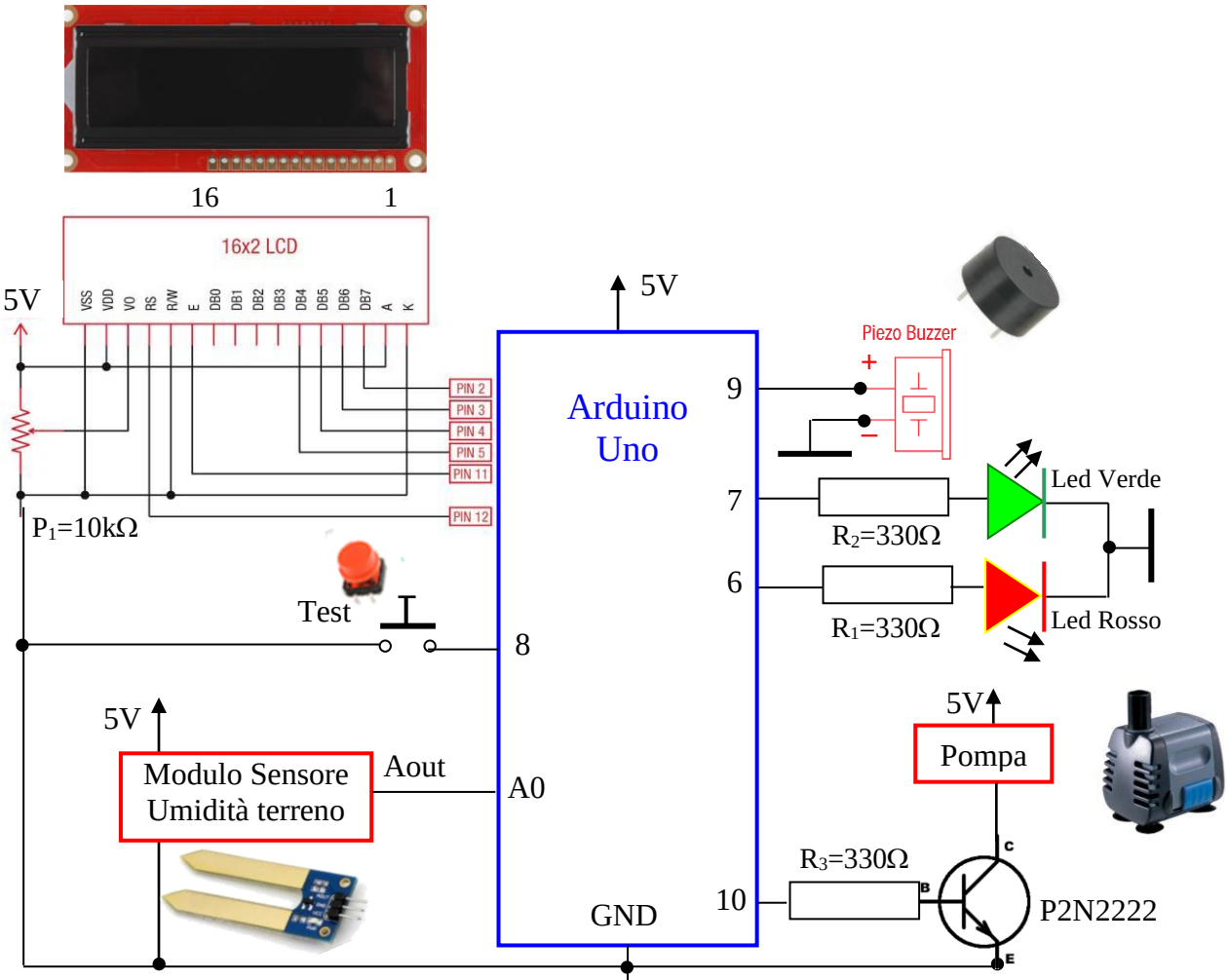


Fig. 1

Modulo sensore umidità terreno

In Fig. 2 è riportato lo schema elettrico del modulo sensore di umidità

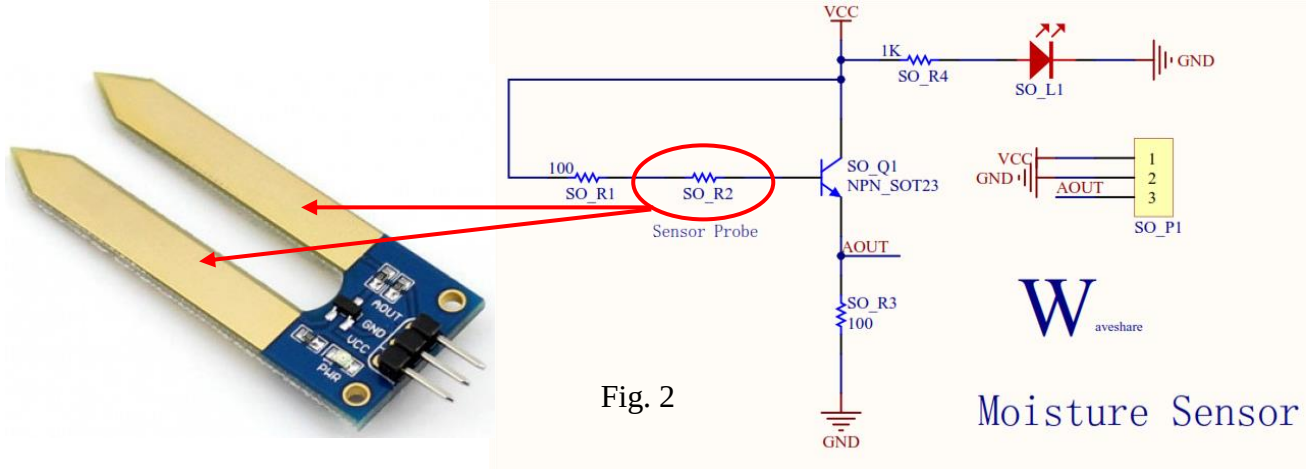


Fig. 2

Lo schema di Fig. 2 mostra un tipico amplificatore di corrente a transistor.
La resistenza di base del transistor è costituita da due resistenze in serie: una fissa da 100 Ω (SO_R1) e da una variabile il cui valore dipende dal grado di umidità del suolo (SO_R2).
La tensione di uscita (AOUT), funzione della corrente di emettitore, è prelevata ai capi della resistenza di emettitore SO_R3.
Se il suolo presenta un grado di umidità basso (terreno secco) il valore di SO_R2 è elevato, la corrente di base del transistor è bassa.
Di conseguenza la corrente di emettitore è bassa e AOUT assume valori bassi.
Man mano che l'umidità aumenta, la resistenza SO_R2 diminuisce, la corrente di base del transistor aumenta facendo aumentare la corrente di emettitore con relativo aumento di AOUT.
Questa tensione viene applicata all'ingresso del convertitore A/D di Arduino tramite il pin A0.
La tensione AOUT è funzione dell'umidità relativa a cui è sottoposto in sensore immerso nel terreno.
Il modulo dispone tre terminali (Vcc, GND, AOUT) e un diodo Led:

1. **Aout**: uscita analogica.
2. **Led SO_L1**: Led Power
3. **Vcc e GND (** Vcc=5V, GND=Massa).

Pulsante test

Il pulsante (Fig. 3) è collegato tra il pin 8 e GND, il pin deve essere abilitato tramite software come input con resistenza di PullUp.

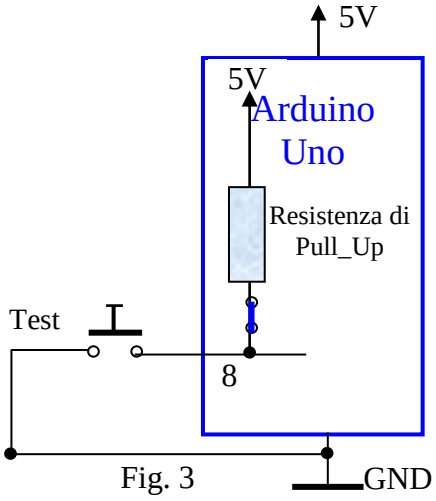


Fig. 3

1) **Test=OFF** (pulsante aperto) il pin 8 si trova a livello alto (5V), il software legge il valore dell'umidità fornita da sensore (ingresso A0).
2) **Test=ON** (pulsante chiuso) il pin 8 si trova a livello basso (GND), il software esclude A0 e tramite un contatore viene simulata l'umidità da 0% a 85% in questo modo è possibile testare il funzionamento delle tre fasi.

Gestione pompa

La pompa alimentata a 5V viene attività tramite il transistor P2N2222 (Fig 4).

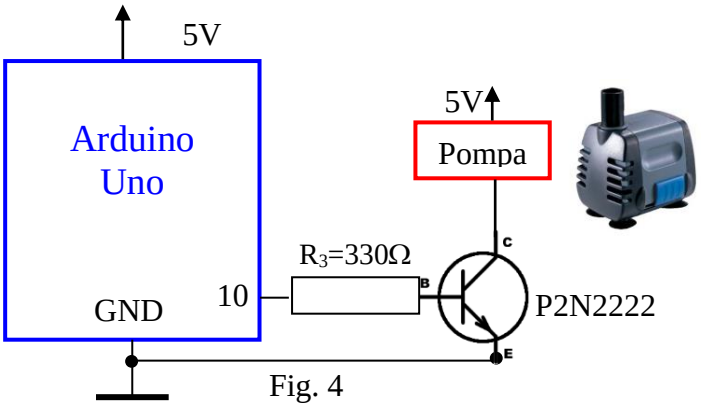
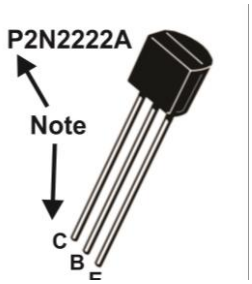


Fig. 4

Si utilizza il circuito di figura perché la corrente di uscita sul Pin 10 non è sufficiente a pilotare la pompa.
Il transistor (NPN) viene polarizzato per il funzionamento come interruttore.



1) **Pin10=HIGH** (*transistor in saturazione*) massima corrente di base, massima corrente di collettore, **pompa on** (attivata).
2) **Pin10=LOW** (*transistor interdetto*) minima corrente di base, minima corrente di collettore, **pompa off** (spenta).

Programma

```
/* Esercizio 36: Modulo Sensore Umidità Terreno
1.  Rileva e visualizza su un display Lcd 16x2 l'umidità rilevata dal sensore;
2.  se l'umidità è inferiore al 20%:visualizza "Asciutto"
     a) accende un Led Rosso e attiva un segnale acustico
     b) attiva una pompa per l'irrigazione goccia a goccia;
3.  se l'umidità è compresa tra il 20% e il 60%
     a) visualizza "Umido"  e  lampeggia  il Led Verde (situazione ottimale)
4.  se l'umidità è superiore al 60%
     a) visualizza "Bagnato" e lampeggia i Led Rosso e Verde
*/
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
int Moisture_ain=A0;
int ledverde=7;
int ledrosso=6;
int ad_value=0;
int buzzer=9;
int pompa=10;
int pulsantetest=8;
int frequenza=700;
void setup()
{
  pinMode(Moisture_ain, INPUT);
  pinMode(ledverde, OUTPUT);
  pinMode(ledrosso, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(pompa, OUTPUT);
  pinMode(pulsantetest, INPUT_PULLUP);
  lcd.begin(16,2);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("U=    %");
  lcd.setCursor(0,1);
  lcd.print("Pompa:");
}
void loop()
{
byte B=digitalRead(pulsantetest);
if (B==0)
{
ad_value=ad_value+2;
if(ad_value>850) {ad_value=0;}
}
else
{
ad_value=analogRead(Moisture_ain);
}
```

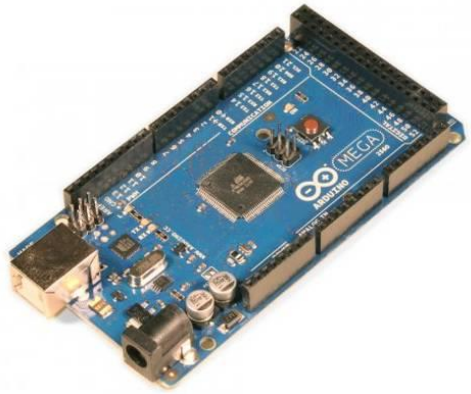
```
int u=map(ad_value,0,1023,0,1000);
float um=u/10.0;
  if(um<20.0)          // Gestione "Asciutto"
  {
    lcd.setCursor(2,0);
    lcd.print("    ");
    lcd.setCursor(2,0);
    lcd.print(um,1);
    lcd.setCursor(8,0);
    lcd.print("Asciutto");
    digitalWrite(ledverde,LOW);
    digitalWrite(ledrosso,HIGH);
    digitalWrite(pompa,HIGH);     // attiva la pompa per l'irrigazione
    lcd.setCursor(6,1);
    lcd.print("ON ");
    tone(buzzer,frequenza,200);
    delay (50);
    frequenza=frequenza+100;
    if (frequenza>3000)    {frequenza=700;}
  }
  else if (um>20.0 && um<60.0)  // Gestione "Umido"
  {
    digitalWrite(pompa,LOW);      // Spegne la pompa per l'irrigazione
    lcd.setCursor(6,1);
    lcd.print("OFF");
    lcd.setCursor(2,0);
    lcd.print("    ");
    lcd.setCursor(2,0);
    lcd.print(um,1);
    lcd.setCursor(8,0);
    lcd.print("Umido    ");
    digitalWrite(ledverde,HIGH);
    digitalWrite(ledrosso,LOW);
    delay (100);
    digitalWrite(ledverde,LOW);
    delay (100);
   }
 else                                 // Gestione "Bagnato"
  {
    digitalWrite(pompa,LOW);      // Spegne la pompa per l'irrigazione
    lcd.setCursor(6,1);
    lcd.print("OFF");
    lcd.setCursor(2,0);
    lcd.print("    ");
    lcd.setCursor(2,0);
    lcd.print(um,1);
    lcd.setCursor(8,0);
    lcd.setCursor(8,0);
```

```
    lcd.print("Bagnato    ");
    digitalWrite(ledverde,LOW);
    digitalWrite(ledrosso,HIGH);
    delay (100);
    digitalWrite(ledverde,HIGH);
    digitalWrite(ledrosso,LOW);
    delay (100);
   }
 }
```

Appendice

a) Arduino (Sistema di sviluppo programmabile)

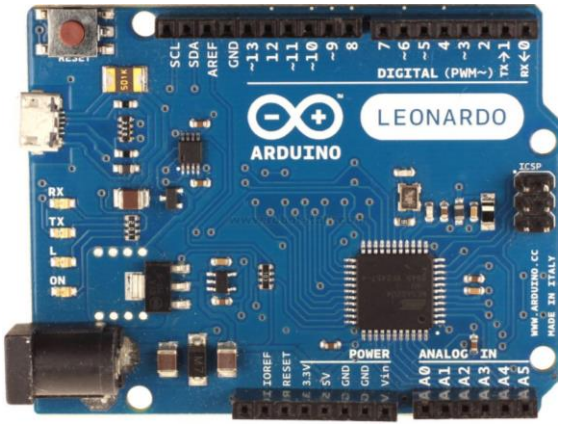
Arduino è un sistema (framework) open source che permette la prototipazione rapida e l'apprendimento veloce dei principi fondamentali dell'elettronica e della programmazione.
È composto da una piattaforma hardware per il physical computing sviluppata presso l'Interaction Design Institute, un istituto di formazione post-dottorale con sede a Ivrea, fondato da Olivetti e Telecom Italia. Questa si basa su un circuito stampato che integra un microcontrollore con pin connessi alle porte I/O, un regolatore di tensione e un'interfaccia USB che permette la comunicazione con il computer. A questo hardware viene affiancato un ambiente di sviluppo integrato (IDE) multipiattaforma (Linux, Apple Macintosh e Windows). Questo software permette di scrivere programmi (**sketch**) con un linguaggio semplice e intuitivo derivato da C/C++ chiamato **Wiring** (cablare, collegare con cavi). Arduino può essere utilizzato per lo sviluppo di oggetti interattivi stand-alone (funzionare da solo)ma può anche interagire, tramite collegamento, con software residenti su computer. La particolarità del progetto è che le informazioni sull'hardware e soprattutto i progetti sono disponibili per chiunque: si tratta quindi di un hardware open source.
In tabella sono riportate le caratteristiche tecniche di alcuni modelli.

Modello	Specifiche Tecniche	
Arduino Uno 	Microcontrollore	ATmega328
	Tensione di funzionamento	5V
	Tensione di Alimentazione (raccomandata)	7-12V
	Massima Tensione supportata (non raccomandata)	20V
	I/O digitali	14 (6 dei quali con uscita PWM)
	Ingressi analogici	6
	Corrente in uscita per I/O Pin	40 mA
	Corrente in uscita per 3.3V Pin	50 mA
	Memoria Flash	32 KB (ATmega328) di cui 0.5 KB usata bootloader
	SRAM	2 KB (ATmega328)
	EEPROM	1 KB (ATmega328)
	Velocità di clock	16 MHz
Arduino Mega 2560 	Microcontrollore	ATmega2560
	Tensione di funzionamento	5 V
	Tensione di ingresso (raccomandato)	7-12V
	Tensione di ingresso (limiti)	6-20V
	I/O digitali	54 (di cui 14 anche come uscite PWM)
	Ingressi analogici	16
	Corrente DC per ogni pin I/O	40 mA
	Corrente DC per pin 3.3V	50 mA
	Memoria flash	256 KB di cui 8 KB utilizzati dal bootloader
	SRAM	8 KB
	EEPROM	4 KB
	Velocità di clock	16 MHz

Arduino Leonardo

Arduino Leonardo è una scheda basata sul microcontrollore ATmega32U4.
Rispetto ai chip della UNO e della MEGA2560, questo integra il convertitore seriale/USB in hardware per cui può essere collegato direttamente al computer senza microcontrollori intermedi.
Questa caratteristica permette alla scheda di essere riconosciuta dal sistema operativo come una periferica HID (Human Interface Device) tipo un mouse o una tastiera: la LEONARDO ha librerie specifiche che permettono di poter simulare appunto questi dispositivi in modo da poter spedire un tasto al sistema oppure muovere il puntatore del mouse o far leggere un click. A parte questa particolarità la LEONARDO è molto compatibile con la UNO: presenta lo stesso layout e la stessa disposizione dei pin. Le uniche differenze riguardano la gestione della seriale, che sulla LEONARDO è fatta, come detto, in maniera diretta per cui per accedere ai pin RX e TX è stata creata una classe Serial1, mentre la Serial classica indica al chip di comunicare col computer. Oltre a questo, il chip ha anche 12 ingressi analogici rispetto ai 6 della UNO: gli ingressi in più sono indicati sul retro della scheda con una didascalia aggiuntiva. Il chip è leggermente differente a livello hardware per cui bisogna prestare attenzione nell'uso di software che accede in maniera diretta all'hardware dell'ATmega328P perché esso potrebbe non girare sull'ATmega32U4. Altra differenza è la porta Micro-USB per la connessione al PC, che necessita dell'uso di un cavetto tipo quello dei cellulari più recenti.

Arduino Leonardo



Microcontrollore	ATmega32U4
Tensione di funzionamento	5 V
Tensione di ingresso (raccomandato)	7-12V
Tensione di ingresso (limiti)	6-20V
I/O digitali	20 (di cui 7 come uscite PWM)
Ingressi analogici	12
Corrente DC per ogni pin I/O	40 mA
Corrente DC per pin 3.3V	50 mA
Memoria flash	32 KB di cui 4 KB utilizzati dal bootloader
SRAM	2,5 KB
EEPROM	1 KB
Velocità di clock	16 MHz

d) Modulo RTC DS1307

Generalità

Un RTC (Real-Time Clock, orologio in tempo reale, è un dispositivo con funzione di orologio, solitamente costituito da un processore a circuito integrato specializzato per questa funzione, il quale conteggia il tempo reale (anno, mese, giorno, ore, minuti e secondi) anche quando l'utilizzatore viene spento. Viene usato in molti tipi di computer ed è presente in tutti i PC. L'RTC è presente anche in molti sistemi embedded nonché viene utilizzato anche in circuiti elettronici dove è necessario avere un preciso riferimento temporale.
Alcuni modelli di RTC integrati sono il DS1307 di Maxim, il PCF8563 di NXP Semiconductors (ex Philips) e le diverse soluzioni di Integrated Device Technology (IDT).
Per poter mantenere il conteggio del tempo anche a circuito non alimentato, i real-time clock hanno un oscillatore al quarzo a loro dedicato e sono alimentati da una speciale batteria autonoma rispetto all'alimentazione principale; in alcuni tipi anche il quarzo è installato all'interno del package. Al contrario, i clock che non sono real-time non funzionano quando il dispositivo è spento.
Gli RTC furono introdotti nei computer agli inizi degli anni ottanta: uno dei primi ad integrare un orologio in tempo reale fu l'Apple III. Successivamente anche IBM utilizzò un RTC nel

suo PC AT del 1984, che integrava un RTC MC146818. Successivamente anche Dallas Semiconductor realizzò degli RTC. Gli orologi in tempo reale erano facilmente individuabili sulle schede madri dei vecchi PC grazie alla batteria tampone che avevano vicino e a dei disegni che indicavano la funzione del chip. Nei computer più recenti gli RTC sono integrati direttamente nel chipset del sistema. Gli RTC non devono essere confusi con il real-time o il clock della CPU.

RTC DS 1307 con interfaccia I2C

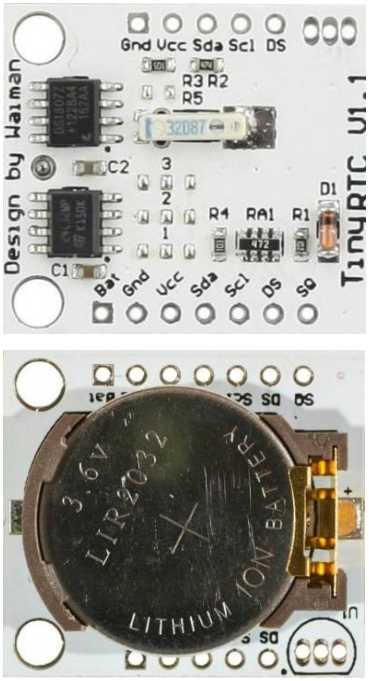
Il modulo si basa sull'integrato **DS1307 I2C** RTC (vedi datasheet) è dotato di EEPROM (**AT24C32** - Erasable and Programmable Read Only Memory) con 32kbyte memoria. Il modello in esame integra il circuito per l'istallazione del sensore di temperatura DS18B20 (Fig. 6).

È presente una batteria tampone ricaricabile LIR2032 al litio che consente di alimentare il dispositivo e garantisce un funzionamento a lungo termine anche quando è disconnesso da Arduino. Il circuito risulta essere così totalmente indipendente: incrementa l'ora senza aver bisogno di un microcontrollore e con la batteria completamente carica è in grado di funzionare per 1 anno intero quando il sensore di temperatura risulta spento oppure non presente.

In Fig 7 è riportato lo schema elettrico del modulo.

L'orario e il calendario sono salvati nei registri in formato BCD. In questo formato ogni cifra di un numero è rappresentata con un codice binario a quattro bit, il cui valore è compreso tra 0000 (0) e 1001 (9). Ad esempio il numero 127 in formato BCD viene registrato in questo modo: 0001 0010 0111.

Sebbene il BCD comporti un notevole spreco di bit (circa 1/6 di memoria inutilizzata in packed BCD), in alcuni casi è preferibile perché ha una diretta corrispondenza con il codice ASCII.



Alloggiamento per **DS18B20**

Fig. 6

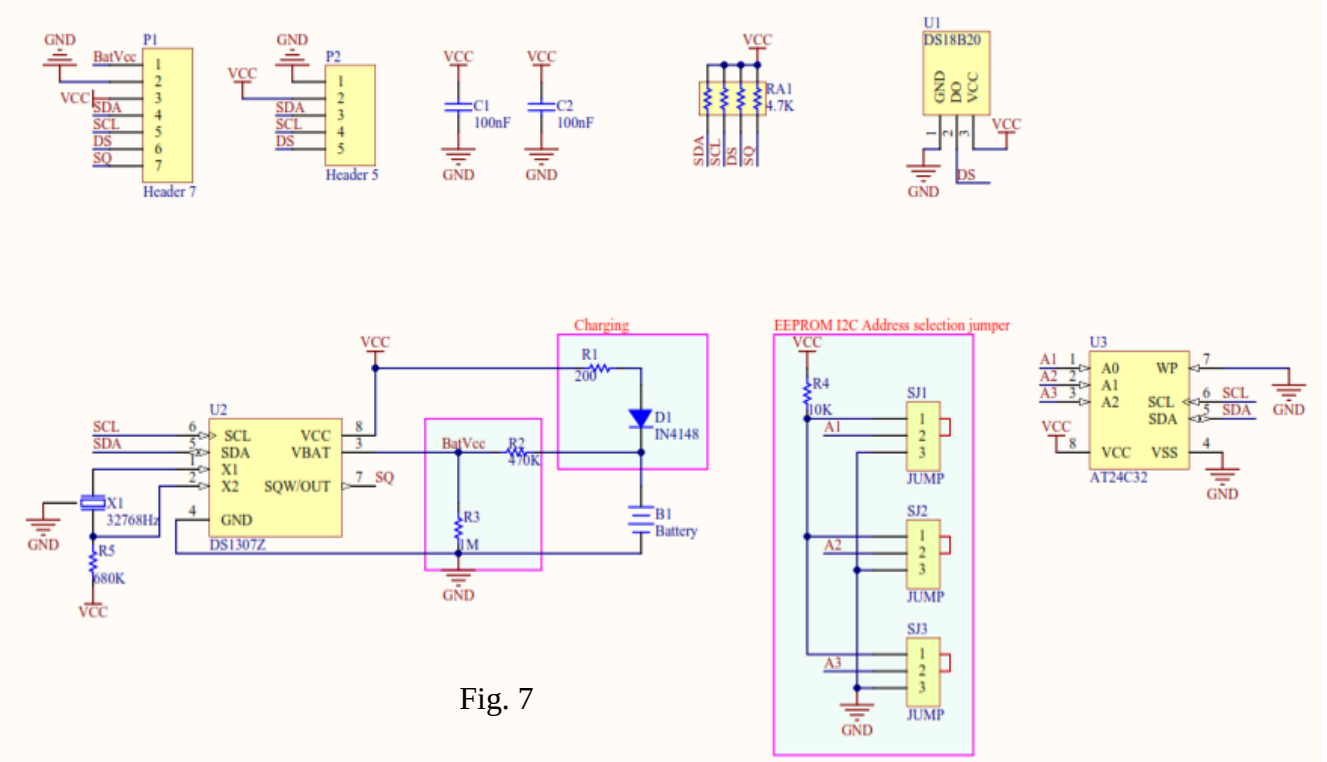


Fig. 7

È sufficiente infatti sostituire i primi quattro bit inutilizzati con 0011 per ottenere il corrispondente ASCII (0011 + BCD => 0011 + 0000 => '0').
Il codice BCD è molto usato in elettronica, specialmente in circuiti digitali privi di microprocessore, perché facilita la visualizzazione di lunghe cifre su display a sette segmenti dove ad ogni display fisico corrisponde esattamente una cifra. Esistono appositi circuiti integrati che effettuano la conversione da BCD nella corrispondente sequenza di accensione dei segmenti. Anche l'esecuzione di semplici calcoli aritmetici è più semplice da effettuarsi su cifre BCD per circuiti logici combinatori.
In Fig. 8 la tabella degli indirizzi dei registri interni dell'RTC.

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	secondi (decine)			secondi (unità)				SECONDI	00–59
01h	0	minuti (decine)			minuti (unità)				MINUTI	00–59
02h	0	12 1	ore (decine)	ore (decine)	ore (unità)				ORE	1–12 +AM/PM 00–23
		24 0	PM/ AM							
03h	0	0	0	0	0	giorno settimana			GG SETTIMANA	01–07
04h	0	0	giorno mese (decine)		giorno mese (unità)				GG MESE	01–31
05h	0	0	0	mese (decine)	mese (unità)				MESE	01–12
06h	Anno (decine)				Anno (unità)				ANNO	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	08h–FFh

Fig. 8

I registri dell'RTC vanno dall'indirizzo 00h a 07h.
I registri utilizzabili come RAM vanno da 08h a 3Fh.
- Il Bit 7 del **registro dei secondi** (00h) è il bit di halt (**CH**) dell'orologio. Quando questo bit è posto a 1 l'oscillatore è spento mentre se posto a 0 l'oscillatore viene riattivato.
- I valori che corrispondono al giorno della settimana vengono definiti dall'utente (solitamente 1 = domenica, 2 Lunedì etc.). Allo scadere della mezzanotte questi vengono incrementati di uno (se vale 7 viene riportato a 1).
- Il Bit 6 del **registro delle ore** (02h) indica se l'orario è definito con modalità 12-ore (quando vale 1) oppure 24-ore (se vale 0). Nel caso si utilizzi la modalità a 12 ore il bit 5 (bit AM/PM) viene posto a 1 se è PM, 0 se è AM. Nel caso sia abilitata la modalità 24-ore il 5 bit è usato per la codifica delle decine delle ore. Se si cambia da 12 a 24 ore l'orario deve essere resettato.
- il **registro di controllo** (08h) è utilizzato per pilotare l'uscita SQ (che è un'onda quadra). Infatti il nostro modulo RTC ha la possibilità di generare un onda quadra di 1Hz, 4096KHz, 8192KHz, 32768KHz. Tramite software viene programmato il registro per generare un'onda quadra ad 1 Hz. Sul Datasheet sono riportati tutti i dettagli per la configurazione dei registri.
Il modulo utilizza i pin analogici Arduino **A5** (SCL) e **A4** (SDA) per la comunicazione seriale "I^2C" (impostazione predefinita per la libreria <wire.h>). (http://arduino.cc/en/Reference/Wire)
Il DS1307 supporta il protocollo I^2C. Un dispositivo che invia i dati sul bus è definito trasmettitore mentre quella che li legge ricevitore. Il dispositivo che controlla gli altri, detti slave, è detto master e deve generare il segnale di clock, controllare l'accesso al bus e generare le condizioni di START e STOP.
Il DS1307 opera come slave sul bus I^2C ed è allocato all'indirizzo 0x68 (Nel linguaccia C si usa il suffisso 0x per indicare un numero esadecimale).

Impostazione RTC

L'impostazione del RTC avviane tamite le istruzioni:

```
if (!RTC.isrunning())
  {
   // following line sets the RTC to the date & time this sketch was compiled
   RTC.adjust(DateTime(__DATE__, __TIME__));
}
```

Questa parte di programma deve essere inserita in void *setup ()* dopo l'istruzione *RTC.begin();*

La sequenza indicata, se il modulo non è settato (primo utilizzo), imposta nel RTC la Data e l'ora della compilazione.

e) Display LCD I2C 20x4

Modulo LCD 20x4 dispone di 4 righe e 20 colonne è basato sul controller HD44780; dotato di retroilluminazione blu, caratteri bianchi, regolazione del contrasto e di un'interfaccia di comunicazione I2C.

Lo schermo a cristalli liquidi (LCD) è un display a schermo piatto, una visualizzazione elettronica o un video che utilizza le proprietà modulanti della luce dei cristalli liquidi.

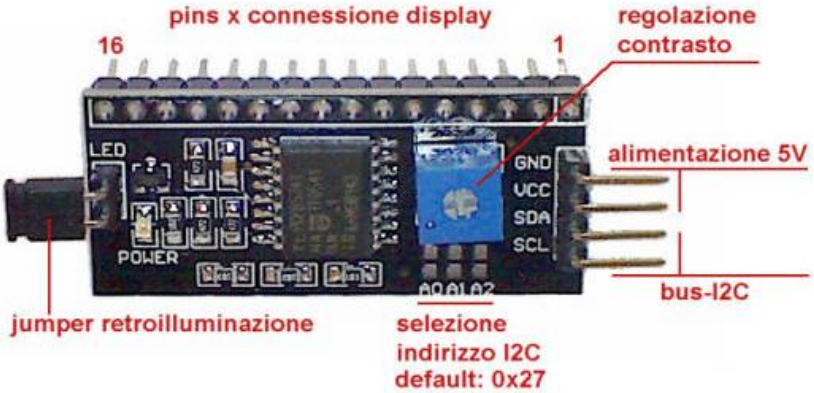
I cristalli liquidi non emettono direttamente la luce.

Come mostrato nell'immagine sotto riportata, un'importante caratteristica di questo modulo LCD è l'interfaccia di comunicazione I2C integrata posta sul retro che ne rende estremamente semplice l'utilizzo con Arduino.

Il modulo è dotato di 4 pin (Alimentazione: **Vcc**, **Gnd**) e (Dati: **SDA**, **SCL**)

Interfaccia di comunicazione I2C

Si tratta di un adattatore/convertitore dal bus seriale I2C al bus parallelo utilizzato dai Display



selezione indirizzo I2C

indirizzi	A0	A1	A2
0x20	chiuso	chiuso	chiuso
0x21	aperto	chiuso	chiuso
0x22	chiuso	aperto	chiuso
0x23	aperto	aperto	chiuso
0x24	chiuso	chiuso	aperto
0x25	aperto	chiuso	aperto
0x26	chiuso	aperto	aperto
0x27	aperto	aperto	aperto

LCD con standard HD44780 e matrice (colonne x righe): 4x2, 8x2, 16x2, 20x2, 16x4, 20x4 ecc

Ad esempio un display LCD 2004(20 colonne, 4 righe) impegna per il suo controllo, a prescindere dall'alimentazione, almeno 6 porte del Microcontroller di sistema per cui il numero delle porte restanti può risultare insufficiente per le necessità del progetto che si vuol

realizzare. Questo modulo permette di comunicare con un display LCD mediante il protocollo I2C che impegna quindi due sole porte. (Dati: **SDA**, **SCL**) oltre ai due terminali di alimentazione (**VCC**, **GND**).

Sul modulo è presente un potenziometro per la regolazione del contrasto ed un jumper rimovibile per l'attivazione della retroilluminazione. Se il jumper viene rimosso e tra i due pin viene inserita una resistenza, si può modificare l'intensità di retroilluminazione (ad esempio con una resistenza da 470 ohm 1/4w l'intensità si dimezza).

Settaggio dell'indirizzo del modulo

Sono inoltre presenti tre connessioni denominate A1, A2 e A3 per il settaggio (a 3-bit) dell'indirizzo I2C tra 0x20 e 0x27. L'indirizzo di default è 0x27 (A0, A1 e A2 lasciati aperti). La chiusura di una coppia si effettua collegando tra loro le due rispettive piazzole. E' ovvio che tale indirizzo hardware deve coincidere con l'indirizzo I2C nel software/libreria di gestione del Modulo. Il display viene gestito tramile la libreria *LiquidCrystal_I2C.h.*

Specifiche:

● Alimentazione: 5V	● Interfaccia: I2C
● Retroilluminazione blu e caratteri bianchi	● Indirizzo I2C: 0x27 (default)
● Retroilluminazione: selezionabile tramite jumper	● Compatibile: Arduino e Raspberry Pi
● Dimensioni: 60 x 90 mm	● Peso: 75,5g

f) Modulo MicroSD

Il modulo MicroSD, tramite l'interfaccia SPI del microcontrollore, consente la lettura e scrittura dei files su una **MicroSD**. Per la gestione del modulo occorre la libreria **SD.h**, richiamabile con il comando **#include <SD.h>,** compresa nell'IDE di Arduino.

Il modulo può essere utilizzato in diversi modi e con diverse funzioni per la gestione dei files sulla scheda microSD.

1. Leggere e scrivere un file, controllare se un file esiste
2. Creare un file di testo, eliminare un file, compilare una lista di tutti i file

In fig.4 il modulo mentre in Fig. 5 lo schema elettrico.

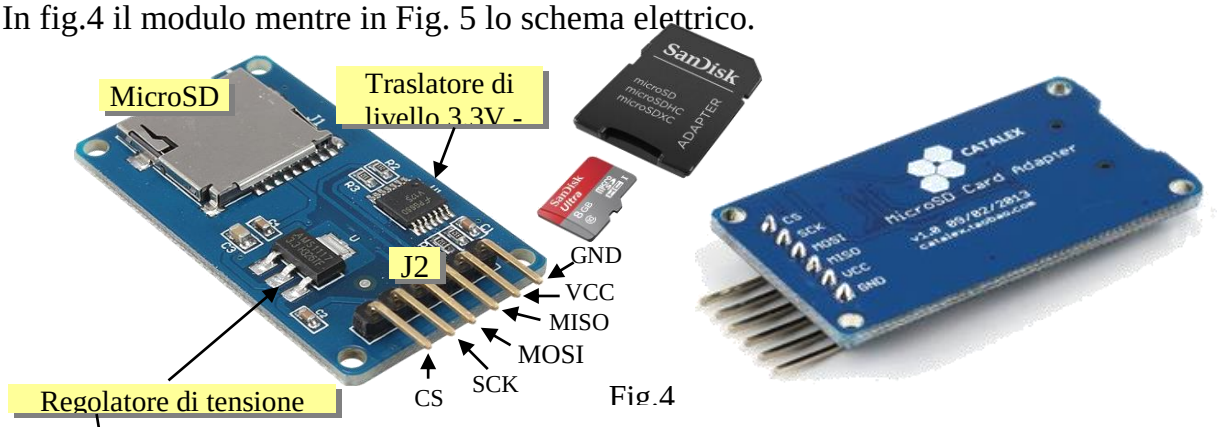


Fig.4

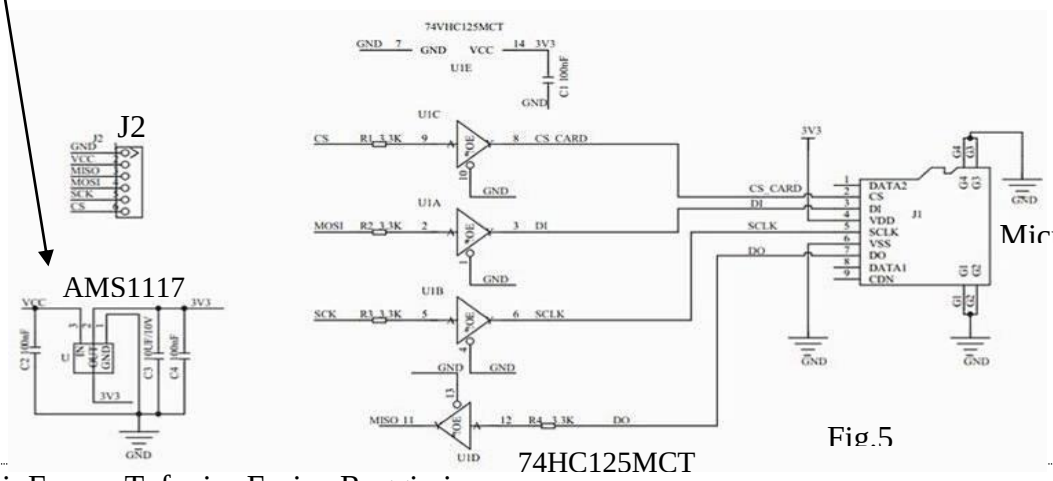


Fig.5

Gli elementi principali sono:

1) Regolatore di tensione 3,3V (AMS1117) (Input=Vcc - Output=3,3V)
2) Traslatore di livello 3,3V -5V (74HC125MCT)
3) Alloggiamento per la MicroSD
4) Connettore **J2** per la connessione alla scheda Arduino

Pin Modulo	Pin Arduino	Descrizione
CS:	10	Chip Select del modulo
SCK:	13	Clock per il sincronismo della comunicazione seriale
MOSI:	11	Master Output Slave Input (linea dati Arduino-->MicroSD)
MISO:	12	Master Input Slave Output (linea dati MicroSD-->Arduino)
VCC GND:		Alimentazione

Descrizione SD e MicroSD

La scheda SD è un dispositivo di memoria di massa estremamente versatile che trova spazio all'interno di molti dispositivi di utilizzo quotidiano. L'idea alla base delle schede SD è quella di creare dei dispositivi di archiviazione di massa di ridotte dimensioni e che, allo stesso tempo, siano in grado di fornire capacià di memorizzazione elevate. La SD Association ha stabilito dei requisiti minimi che le schede Secure Digital devono rispettare se vogliono rientrare in una delle sei classi esistenti:

- Classe 2 (velocità minima di scrittura/lettura di 2 megabyte al secondo);
- Classe 4 (velocità minima di scrittura/lettura di 4 megabyte al secondo);
- Classe 6 (velocità minima di scrittura/lettura di 6 megabyte al secondo);
- Classe 10 (velocità minima di scrittura/lettura di 10 megabyte al secondo);
- UHS Speed Class 1 (acronimo UHS-1, con velocità minima di scrittura/lettura di 10 megabyte al secondo);
- UHS Speed Class 3 (acronimo UHS-3, con velocità minima di scrittura/lettura di 30 megabyte al secondo).

Una scheda SD è un supporto di memoria di massa a stato solido. Ciò vuol dire che, come le RAM o i dischi rigidi SSD, non ha parti in movimento al suo interno, favorendone così resistenza e aspettativa di vita. le schede SD fanno uso di memorie NAND Flash, che archivia dati caricando elettricamente (o scaricando) le varie celle di memoria presenti al suo interno. Particolarmente veloci, le NAND Flash appartengono alla famiglia delle memorie EEPROM (acronimo di Electronically Erasable Programmable Read Only Memory, "memoria di sola lettura programmabile e cancellabile elettronicamente").

Protocollo I^2C