

ARDUINO

86

I pulsanti possono avere forme differenti



simbolo elettrico



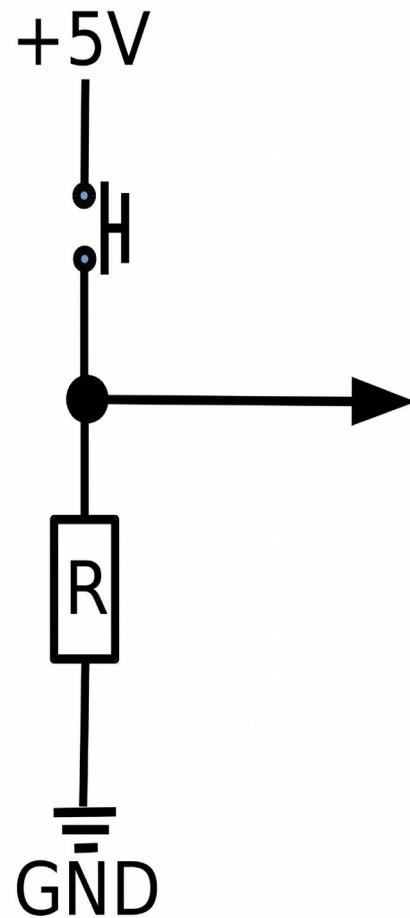
pulsante NO



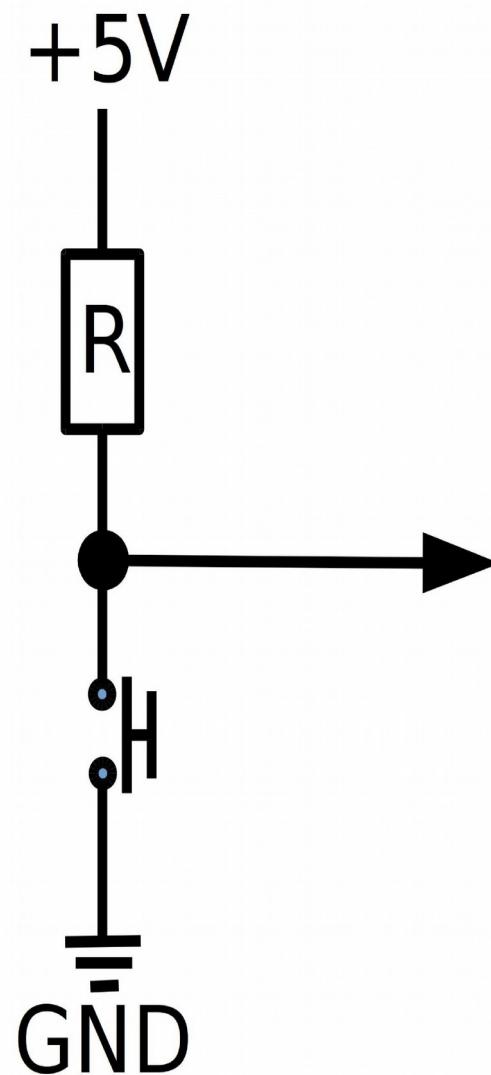
pulsante NC

ARDUINO

Pulsante in pull down



Pulsante in pull up



PROGRAMMAZIONE

Leggere un ingresso digitale

88

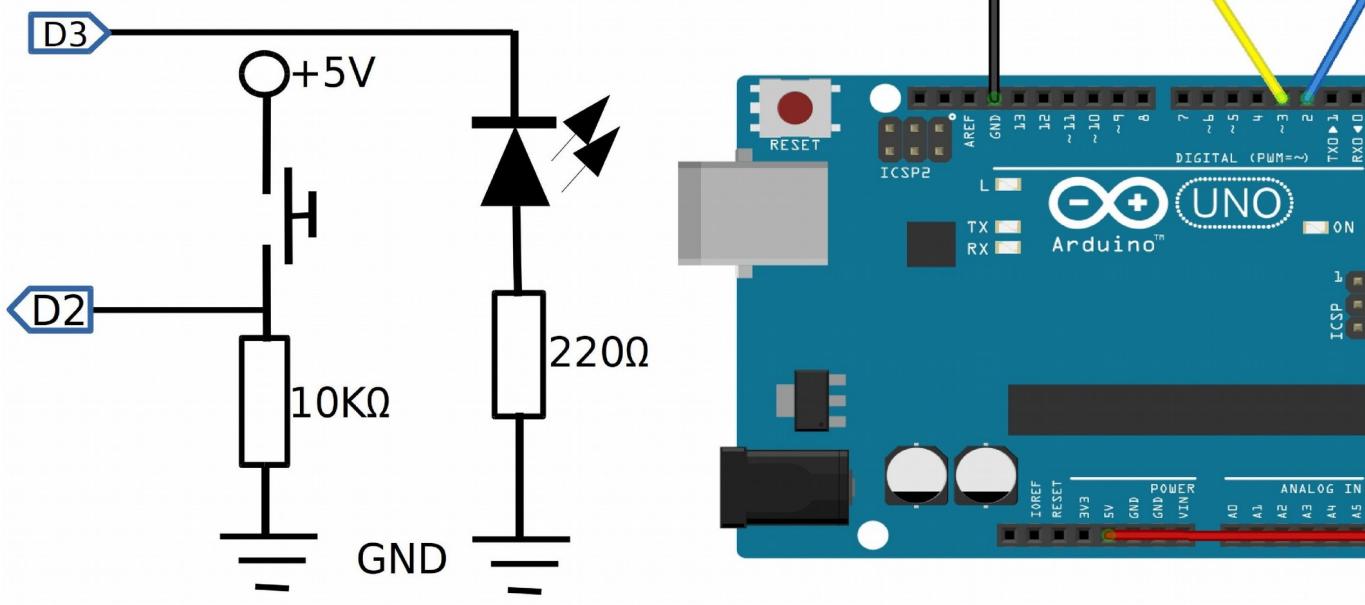
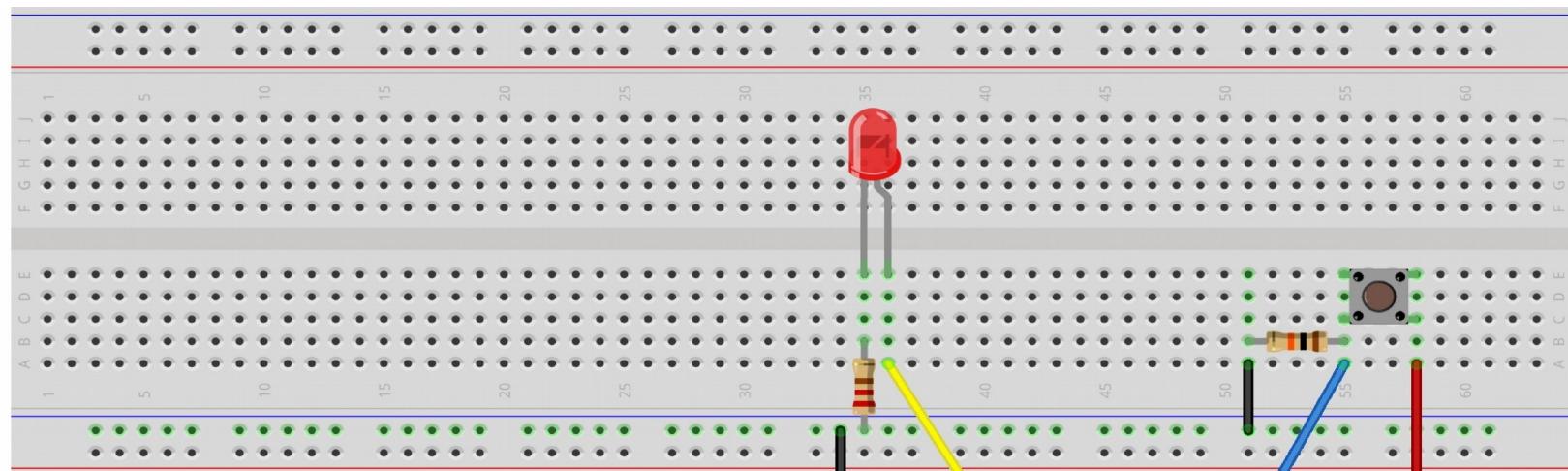
pinMode (connessione, **INPUT**);

pinMode (connessione, **INPUT_PULLUP**);

digitalRead (connessione);

PROGRAMMAZIONE

89



fritzing

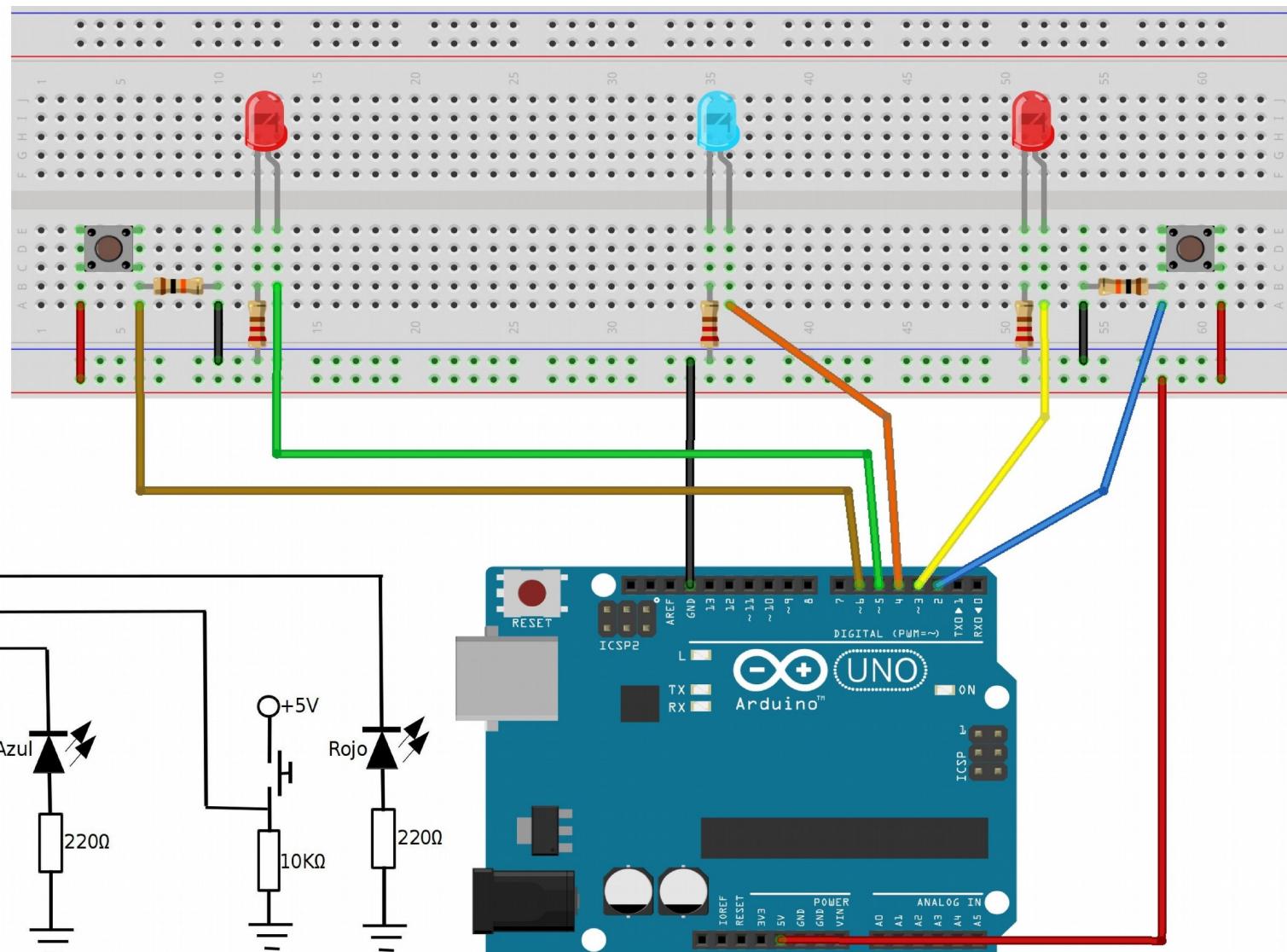
PROGRAMMAZIONE

90

```
1 #define PULSANTE 2
2 #define LED 3
3
4 void setup() {
5     pinMode ( PULSANTE, INPUT );    // configuro la connessione PULSANTE come ingresso
6     pinMode ( LED, OUTPUT );      // configuro la connessione LED come uscita
7 }
8
9 void loop() {
10    if ( digitalRead == HIGH ) {   // se l'ingresso del pulsante è alto
11        digitalWrite ( LED, HIGH ); // accendo il LED
12    }
13    else { // se no
14        digitalWrite ( LED, LOW ); // spengo il LED
15    }
16 }
```

PROGRAMMAZIONE

94



fritzing

PROGRAMMAZIONE

95

```
1 #define pulsante_1 2
2 #define LED_1 3
3 #define LED_BLU 4
4 #define LED_2 5
5 #define pulsante_2 6
15 void setup() {
16     pinMode ( pulsante_1, INPUT);
17     pinMode ( LED_1, OUTPUT );
18     pinMode ( LED_BLU, OUTPUT );
19     pinMode ( LED_2, OUTPUT );
20     pinMode ( pulsante_2, INPUT );
21
22 }
23
```

PROGRAMMAZIONE

96

```
24 void loop(){
25   if (inizio==0){ // se è a zero è una nuova sfida
26     inizio_gioco(); // chiamo la funzione per il gioco di luci
27   }
28 }
29
```

```
1 #define pulsante_1 2
2 #define LED_1 3
3 #define LED_BLU 4
4 #define LED_2 5
5 #define pulsante_2 6
6 byte inizio = 0; // quando inizia il gioco la variabile va a 1
```

PROGRAMMAZIONE

97

```
45 void inizio_gioco(){ // questa funzione fa il gioco di luci dell'inizio
46
47     inizio=1; // metto a 1 la variabile per non rifare il gioco di luci fino alla fine
48     for( int i=0; i<10; i++){ // ripeto 10 volte il gioco di luci
49         digitalWrite ( LED_1, HIGH );
50         delay (150);
51         digitalWrite ( LED_1, LOW );
52         digitalWrite ( LED_BLU, HIGH );
53         delay (150);
54         digitalWrite ( LED_BLU, LOW );
55         digitalWrite ( LED_2, HIGH );
56         delay (150);
57         digitalWrite ( LED_2, LOW );
58         digitalWrite ( LED_BLU, HIGH );
59         delay (150);
60         digitalWrite ( LED_BLU, LOW );
61     }
62     delay ( 1000 );
63     digitalWrite ( LED_1, HIGH );
64     digitalWrite ( LED_2, HIGH );
65     delay ( 3000 );
66     digitalWrite ( LED_1, LOW );
67     digitalWrite ( LED_2, LOW );
68
69 }
70 }
```

PROGRAMMAZIONE

I numeri casuali

98

valore = **random** (minimo, massimo);

valore = **random** (500, 5000);

randomSeed(numero);

randomSeed (analogRead(0));

randomSeed (millis());

PROGRAMMAZIONE

I numeri casuali

99

```
71 void tempo(){ // questa funzione trova il tempo di attesa  
72  
73   randomSeed(analogRead(0)); // creo il numero generatore dei numeri casuali  
74   valore=random(500,5000); //memorizzo il numero casuale da 500 a 5000  
75 }  
76
```

PROGRAMMAZIONE

Come fare un'attesa senza `delay()`

100

Esiste una funzione che si chiama “`millis()`” che legge un contatore che si trova dentro il micro controllore e che si somma di uno tutti i milli secondi che passano. Possiamo leggere il valore del contatore , sommare il valore del tempo dell'attesa e aspettare che il valore del contatore sia uguale alla nostra somma e quando sarà uguale o maggiore vorrà dire che è passato il tempo giusto.

Se leggiamo il contatore quando si trova a 15530 e abbiamo un valore di attesa di 1500 sommando i due valori otterremo un valore di 17030.

Ora dobbiamo continuare a leggere il contatore che si somma di uno tutti i milli secondi e quando sarà uguale o maggiore della nostra somma, 17030, significa che sono passati 1500 milli secondi; il tempo della nostra attesa.

PROGRAMMAZIONE

Come fare un'attesa senza **delay()**

100

Per memorizzare il numero del contatore “`millis()`” ci serve una variabile di tipo “`unsigned long`” perché il valore del contatore potrebbe essere molto grande.

`attesa = millis();`

Metto nella variabile “`attesa`” il numero del contatore per conoscere il suo tempo dopo sommo il valore della mia attesa.

`attesa = attesa + valore;`

Sommo il valore della variabile “`valore`” nella variabile “`attesa`”.

Quando il valore di `millis()` è uguale alla somma significa che è passato il tempo della mia attesa.

PROGRAMMAZIONE

Il controllo di flusso “ while ”

101

```
While ( a ==  
0 ) {  
Istruzioni ...  
}
```

Al contrario del “if” che esegue una sola volta le istruzioni nelle sue parentesi se è vera la condizione, il “**while**” esegue sempre le istruzioni nelle parentesi finché sarà vera la sua condizione.

Il “while” è specifico per fermare un programma in attesa che una condizione si ponga vera o falsa per esempio per aspettare che un motore arrivi a fine corsa o che un riscaldatore raggiunga la temperatura e molto altro.

PROGRAMMAZIONE

101

```
77 void attesa_tempo() { // questa funzione controlla se un giocatore hara nell'attesa
78
79   attesa = millis(); // memorizzo il numero del contatore dei millisecondi
80   attesa = attesa + valore; // sommo al tempo letto del contatore il mio tempo di attesa
81
82   while ( attesa > millis() ){ // se la somma è minore del valore del contatore
83     if ( digitalRead (pulsante_1) == HIGH) { // guardo se il pulsante 1 è premuto
84       punti_2(); // se è premuto metto un punto al giocatore 2
85       rifare = 1; // porto a 1 la fariabile di controllo per rifare la partita
86     }
87     if ( digitalRead (pulsante_2) == HIGH) { // guardo se il pulsante 2 è premuto
88       punti_1(); //se è premuto metto un punto al giocatore 1
89       rifare = 1; // porto a 1 la fariabile di controllo per rifare la partita
90     }
91   }
92 }
```

PROGRAMMAZIONE

103

```
94 void punti_1(){ // questa funzione aggiunge un punto al giocatore 1
95
96     punteggio_1 = punteggio_1 +1; // sommo 1 ai punti del giocatore 1
97
98     for ( int i=0; i<4; i++){      // faccio lampeggiare 4 volte il led del giocatore 1
99         digitalWrite ( LED_1, HIGH );
100        delay(200);
101        digitalWrite ( LED_1, LOW );
102        delay ( 200 );
103    }
104    digitalWrite (LED_BLU, LOW); // spengo il led blu
105 }
106 }
```

PROGRAMMAZIONE

103

```
107 void punti_2(){ // questa funzione aggiunge un punto al giocatore 2
108
109 punteggio_2 = punteggio_2 +1; // sommo 1 ai punti del giocatore 2
110
111 for ( int i=0; i<4; i++){      // faccio lampeggiare 4 volte il led del giocatore 2
112     digitalWrite ( LED_2, HIGH );
113     delay(200);
114     digitalWrite ( LED_2, LOW );
115     delay ( 200 );
116 }
117 digitalWrite (LED_BLU, LOW); // spengo il led blu
118 }
119
```

PROGRAMMAZIONE

104

```
24 void loop(){
25   if (inizio==0){ // se è a zero è una nuova sfida
26     inizio_gioco(); // chiamo la funzione per il gioco di luci
27   }
28   partite = partite +1; // sommo 1 per tenere il conto delle partite
29   tempo(); // chiamo la funzione per il numero casuale dell'attesa
30   attesa_tempo(); // chiamo la variabile che controlla che nessuno bari nell'attesa
31 }
32
```

PROGRAMMAZIONE

105

```
120 void gara(){ // questa funzione controlla chi preme per primo il pulsante
121
122     while ( gara_finita == 0 ){ // il "while" si ferma quando cambia di valore la variabile
123         if ( digitalRead ( pulsante_1) == HIGH ){ // se il giocatore 1 preme per primo
124             gara_finita = 1; // metto a 1 la variabile di controllo per fermare il "while"
125             punti_1(); // aggiungo un punto al giocatore 1
126         }
127
128         if (gara_finita == 0 ){ // se il giocatore 1 non ha premuto controllo il secondo
129
130             if ( digitalRead ( pulsante_2) == HIGH ){ // si el jugador 2 presiona por primero
131                 gara_finita = 1; // metto a 1 la variabile di controllo per fermare il "while"
132                 punti_2(); // aggiungo un punto al giocatore 2
133             }
134         }
135     }
136     gara_finita=0;
137 }
```

PROGRAMMAZIONE

104

```
24 void loop(){
25   if (inizio==0){ // se è a zero è una nuova sfida
26     inizio_gioco(); // chiamo la funzione per il gioco di luci
27   }
28   partite = partite +1; // sommo 1 per tenere il conto delle partite
29   tempo(); // chiamo la funzione per il numero casuale dell'attesa
30   attesa_tempo(); // chiamo la variabile che controlla che nessuno bari nell'attesa
31 }
32
```

PROGRAMMAZIONE

107

```
24 void loop(){  
25     if (inizio==0){ // se è a zero è una nuova sfida  
26         inizio_gioco(); // chiamo la funzione per il gioco di luci  
27     }  
28     partite = partite +1; // sommo 1 per tenere il conto delle partite  
29     tempo(); // chiamo la funzione per il numero casuale dell'attesa  
30     attesa_tempo(); // chiamo la variabile che controlla che nessuno bari nell'attesa  
31     if (rifare==0){ // se "rifare" è = a 0 è perchè nessuno ha barato  
32         digitalWrite (LED_BLU,HIGH); // accendo il led blu per iniziare la sfida  
33         gara();  
34         if ( partite > 10){ // se il numero delle partite è 11 il gioco finisce  
35             fine(); // vado a vedere chi ha vinto  
36             punteggio_1 = 0; // metto a zero il punteggio del giocatore 1 per una nuova gara  
37             punteggio_2 = 0; // metto a zero il punteggio del giocatore 2 per una nuova gara  
38             inizio = 0; // metto a zero la variabile per iniziare un nuovo gioco  
39         }  
40     }  
41 }  
42 rifare=0; //metto a zero la variabile per iniziare una nuova partita  
43 }
```

PROGRAMMAZIONE

108

```
138 void fine(){
139     if ( punteggio_1>punteggio_2){    // se il punteggio del giocatore 1 è maggiore
140         for(int i=0; i<4; i++){        // faccio lampeggiare il led blu e il led rosso
141             digitalWrite(LED_BLU,HIGH); // del giocatore 1 per 4 volte
142             digitalWrite(LED_1,HIGH);
143             delay(1000);
144             digitalWrite(LED_BLU,LOW);
145             digitalWrite(LED_1,LOW);
146             delay(500);
147         }
148     }
149
150     else {                          // se non è maggiore vince il giocatore 2
151         for(int i=0; i<4; i++){        // faccio lampeggiare il led blu e il led rosso
152             digitalWrite(LED_BLU,HIGH); // del giocatore 2 per 4 volte
153             digitalWrite(LED_2,HIGH);
154             delay(1000);
155             digitalWrite(LED_BLU,LOW);
156             digitalWrite(LED_2,LOW);
157             delay(500);
158         }
159     }
160     partite = 0; // metto a zero il numero delle partite giocate
161     delay ( 5000 ); // aspetto 5 secondi prima di ricominciare un nuovo gioco
162 }
163
```

PROGRAMMAZIONE

109

```
6 byte inizio = 0; // quando inizia il gioco la variabile va a 1
7 int valore=0; // in "valore" metto ul numero casuale del tempo di attesa
8 unsigned long attesa = 0; // memorizzo il valore del contatore dei millisecondi per calcolare l'attesa
9 byte rifare = 0; // se un giocatore bara la variabile va a 1
10 byte punteggio_1 = 0; // memorizzo il punteggio del giocatore 1
11 byte punteggio_2 = 0; // memorizzo il punteggio del giocatore 2
12 byte gara_finita = 0; // va a 1 quando finisce una gara
13 byte partite = 0; // memorizzo il numero delle partite
14
```

ARDUINO

IL DISPLAY LCD

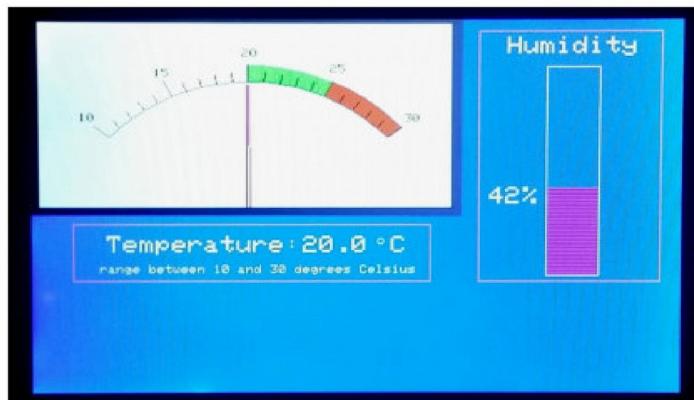
110



Display lcd 2 linee 16 caratteri



Display lcd 4 linee 20 caratteri



Display lcd grafico

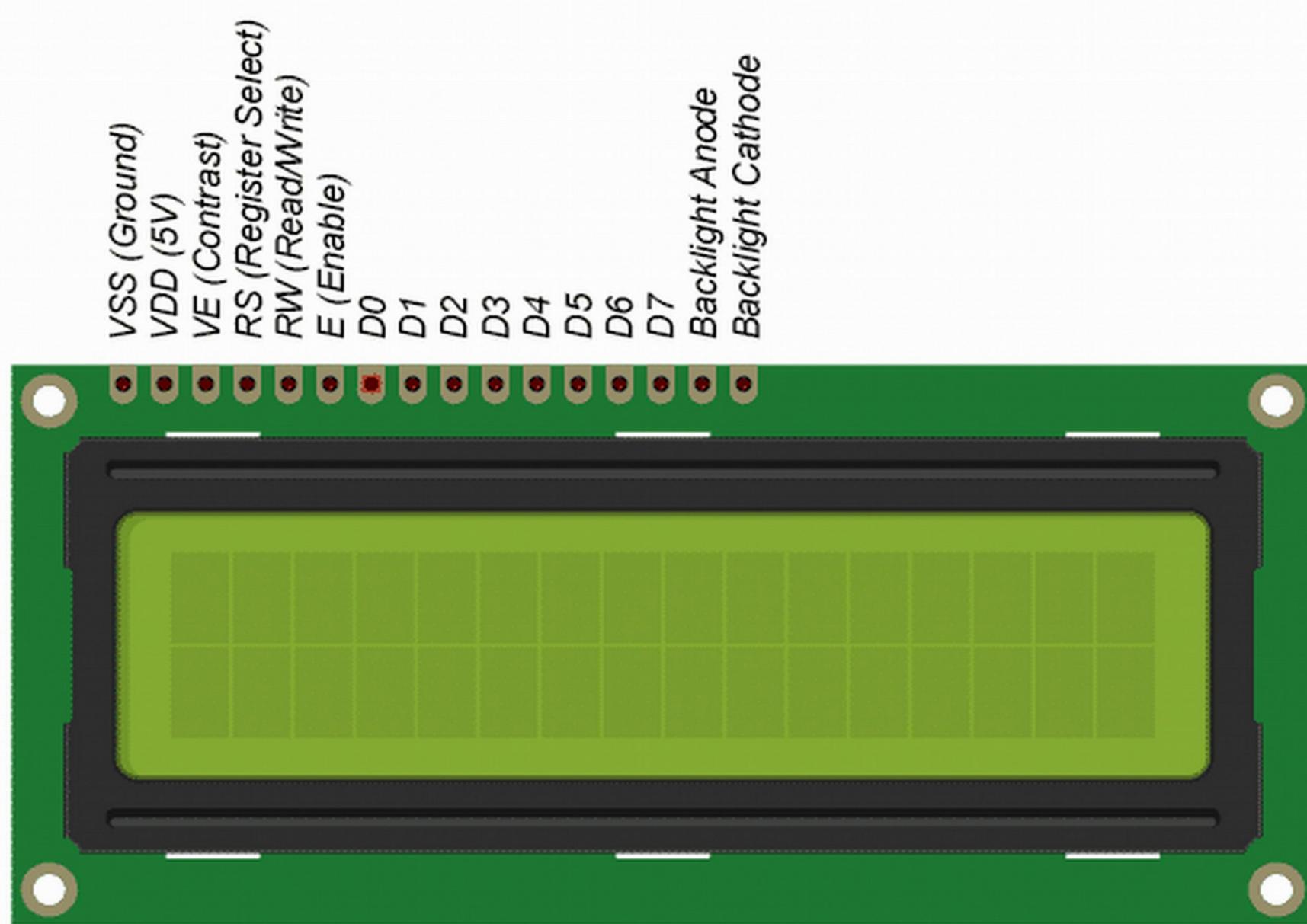


Display oled grafico

ARDUINO

IL DISPLAY LCD

111



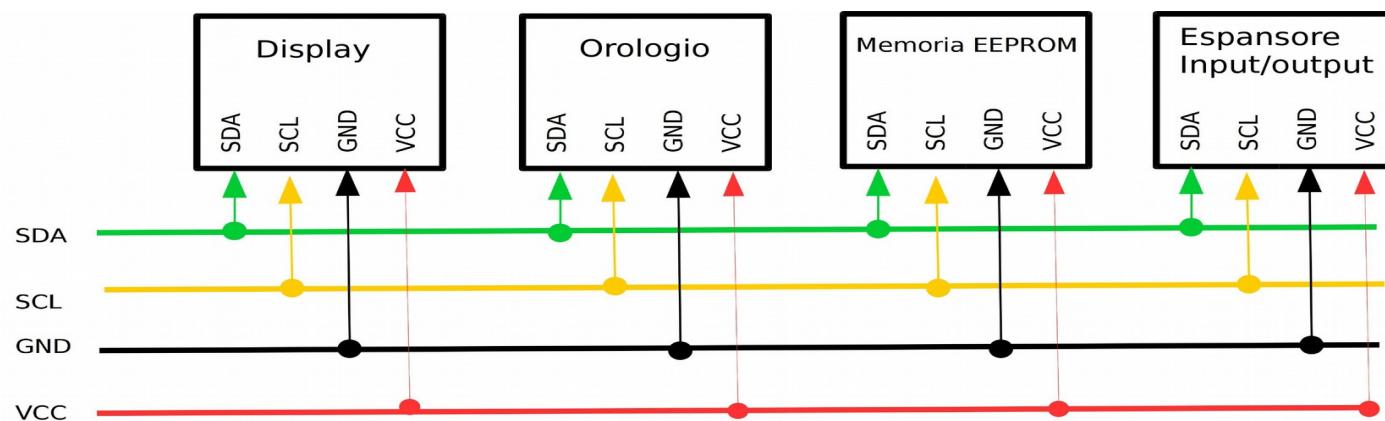
ARDUINO

IL BUS I2C

112



Tutte le schede hanno un numero di indirizzo diverso, non possono avere lo stesso. L'indirizzo si scrive con un numero esadecimale.

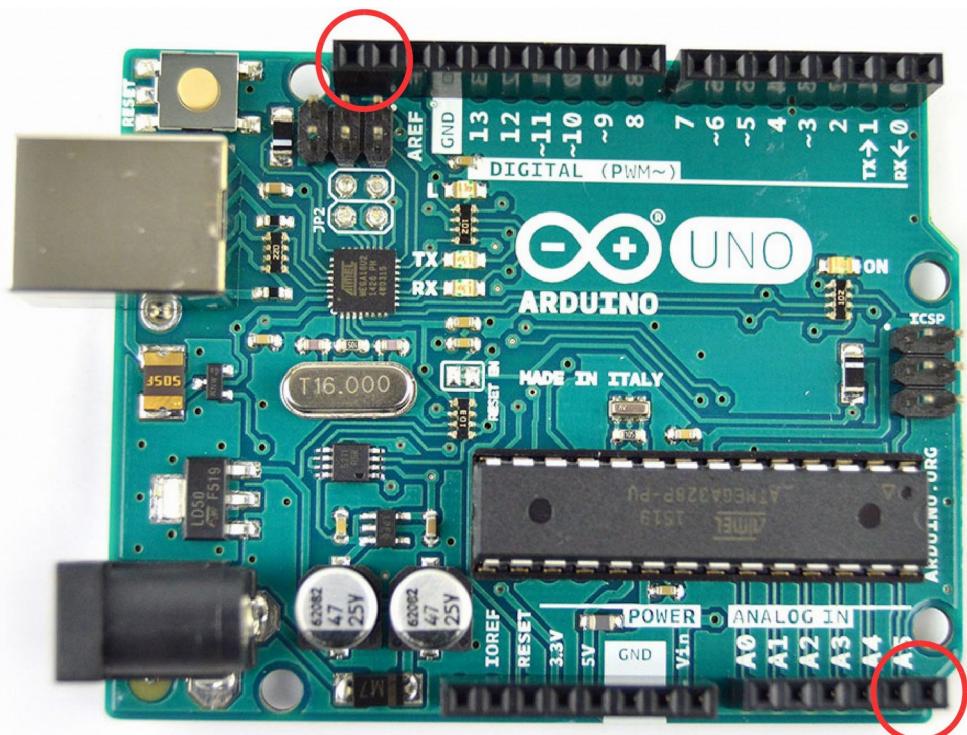


ARDUINO

IL BUS I2C

113

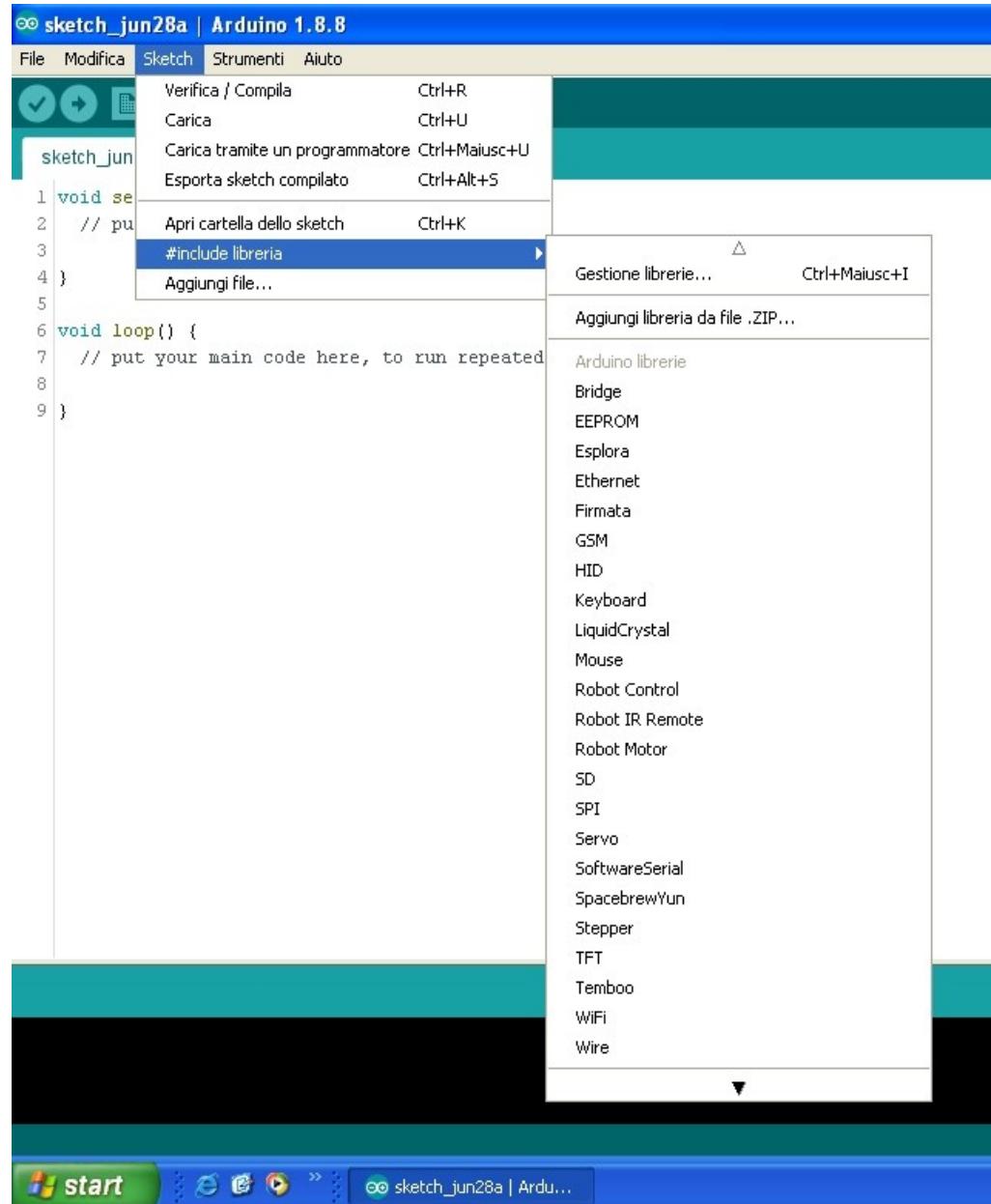
Esadecimale	Decimale	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	0100
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111



PROGRAMMAZIONE

LE LIBRERIE

114



PROGRAMMAZIONE

115

Gestore librerie

Tipo: Tutti Argomento: Tutti Filtra la tua ricerca...

Arduino Cloud Provider Examples by Arduino
Examples of how to connect various Arduino boards to cloud providers
[More info](#)

Arduino Low Power by Arduino
Power save primitives features for SAMD and nRF52 32bit boards With this library you can manage the low power states of newer Arduino boards
[More info](#)

Arduino SigFox for MKRFox1200 by Arduino
Helper library for MKRFox1200 board and ATAB8520E Sigfox module This library allows some high level operations on Sigfox module, to ease integration with existing projects
[More info](#)

Versione 1.0.4 [Installa](#)

Arduino Uno WiFi Dev Ed Library by Arduino
This library allows users to use network features like rest and mqtt. Includes some tools for the ESP8266. Use this library only with Arduino Uno WiFi Developer Edition.
[More info](#)

Chiudi

PROGRAMMAZIONE

116

sketch_jun28a §

```
1 #include <Wire.h>
2
3 void setup() {
4     // put your setup code here, to run once:
5
6 }
7
8 void loop() {
9     // put your main code here, to run repeatedly:
10
11 }
```

PROGRAMMAZIONE

117

Gestor de Librerías

Tipo Todos Tema Todos liquidCrystal I2C

LCDMenubar by Niels Reitkampen version 2.1.0 INSTALLED
Easy creation of a tree based menu with screensaver and multi layers. Examples for the basic function and different output types [serial monitor, liquidcrystal, i2c, graphic displays (u8glib / u8g2lib...)]
[More info](#)

LiquidCrystal I2C by Frank de Brabander
A library for I2C LCD displays. The library allows to control I2C displays with functions extremely similar to LiquidCrystal library.
THIS LIBRARY MIGHT NOT BE COMPATIBLE WITH EXISTING SKETCHES.
[More info](#)

Versión 1.1.2 [Instalar](#)

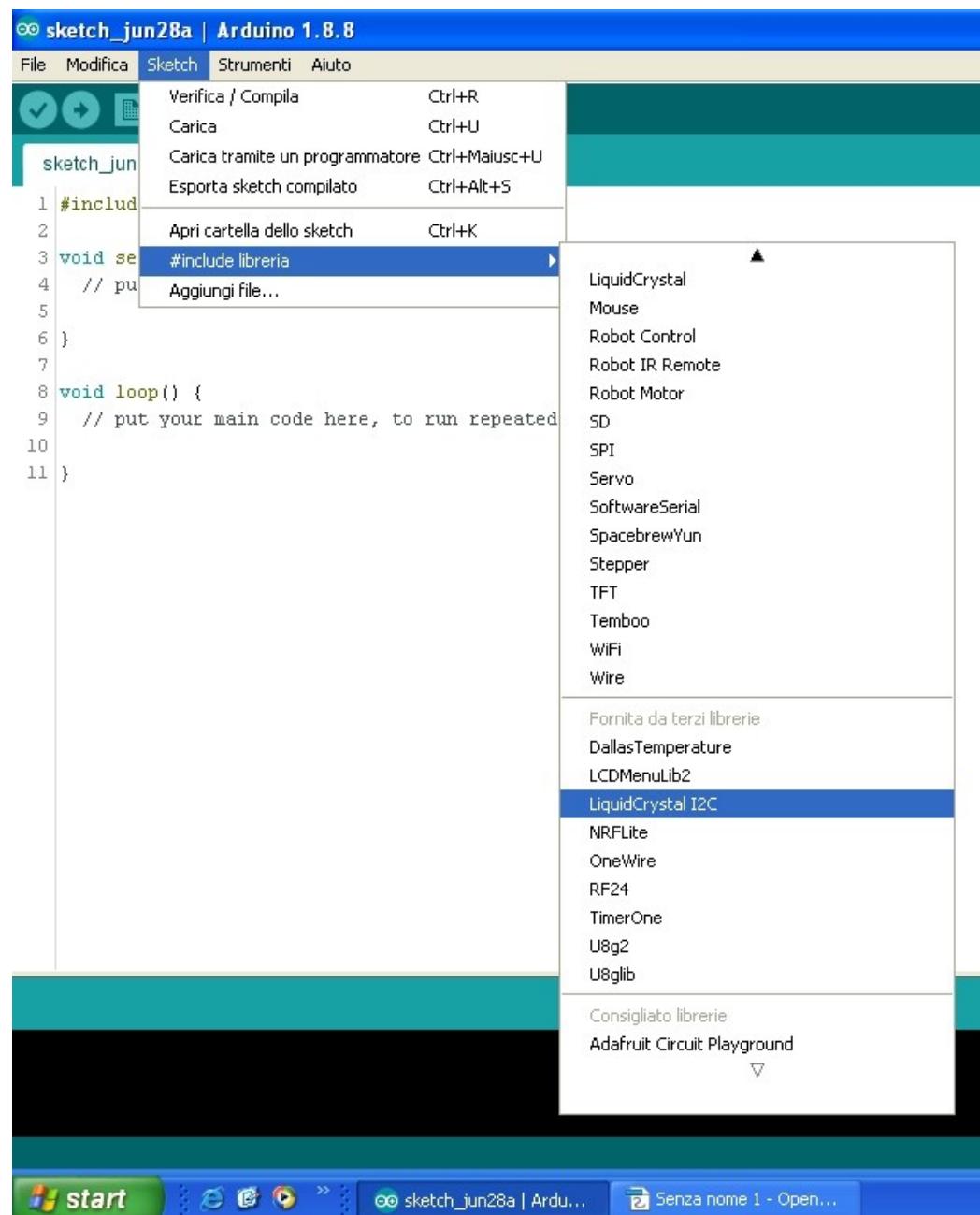
LiquidCrystal_I2C_Hangul by Junwha Hong, Dohun Kim, HyungHo Kim
A library for printing Hangul on I2C LCD displays. The library allows to control I2C displays with functions extremely similar to LiquidCrystal library. This Library allows to print hangul on LCDs.
[More info](#)

LiquidCrystal_PCF8574 by Matthias Hertel
A library for driving LiquidCrystal displays (LCD) by using the I2C bus and an PCF8574 I2C adapter. This library is derived from the original Arduino LiquidCrystal library and uses the original Wire library for communication.
[More info](#)

Cerrar

PROGRAMMAZIONE

117



PROGRAMMAZIONE

118

sketch_jun28a §

```
1 #include <Wire.h>
2
3 #include <LiquidCrystal_I2C.h>
4
5 void setup() {
6     // put your setup code here, to run once:
7
8 }
9
10 void loop() {
11     // put your main code here, to run repeatedly:
12
13 }
```

PROGRAMMAZIONE

118

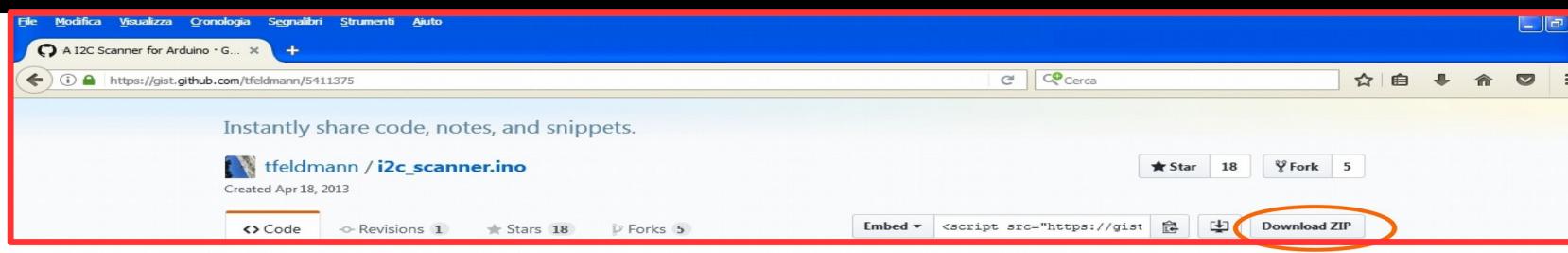
Per conoscere l'indirizzo di una scheda I2C ci serve uno scanner I2C.

Lo troviamo in Internet
nel sito GitHub.

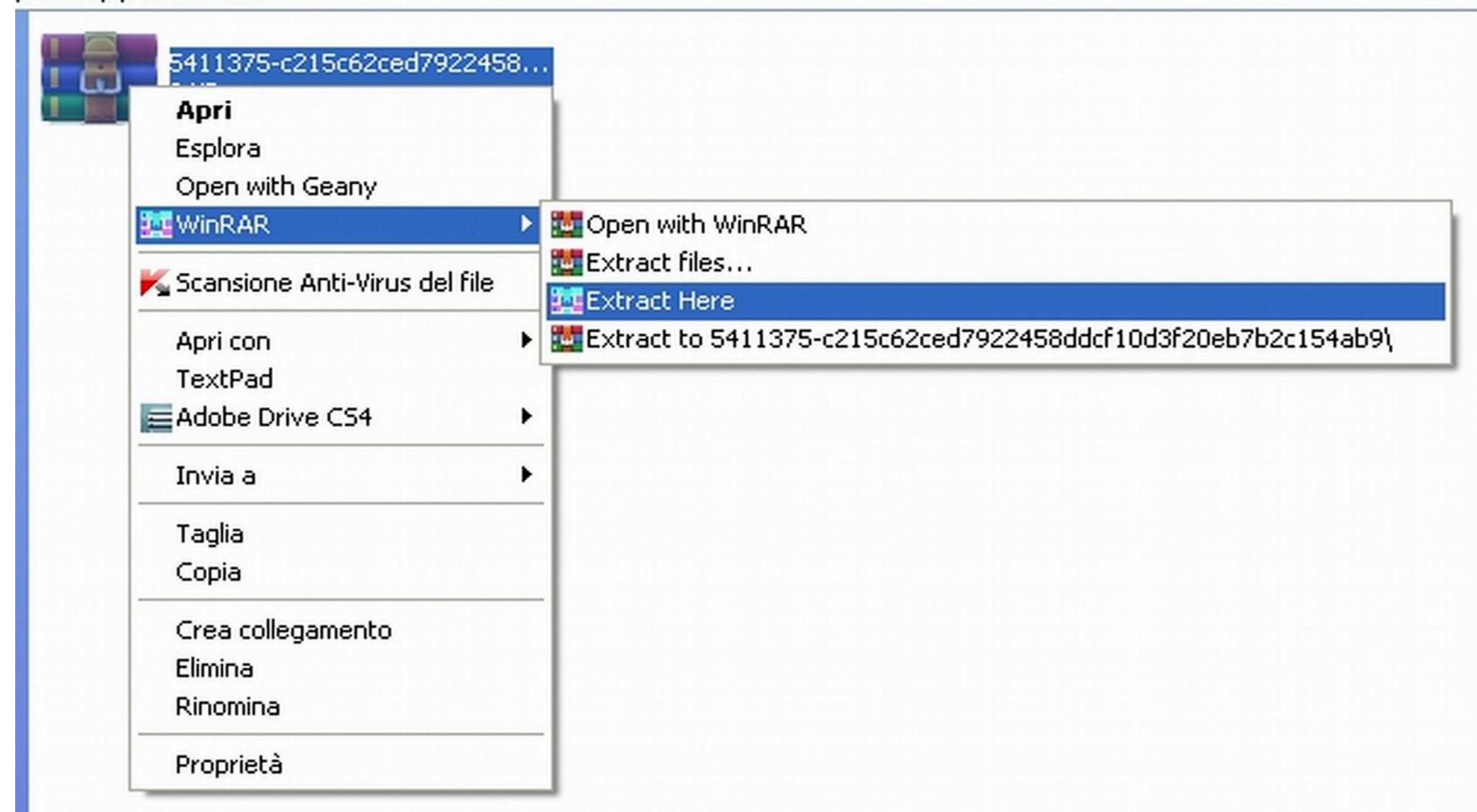
<https://gist.github.com/tfeldmann/5411375>

PROGRAMMAZIONE

119



\Desktop\scanner I2C



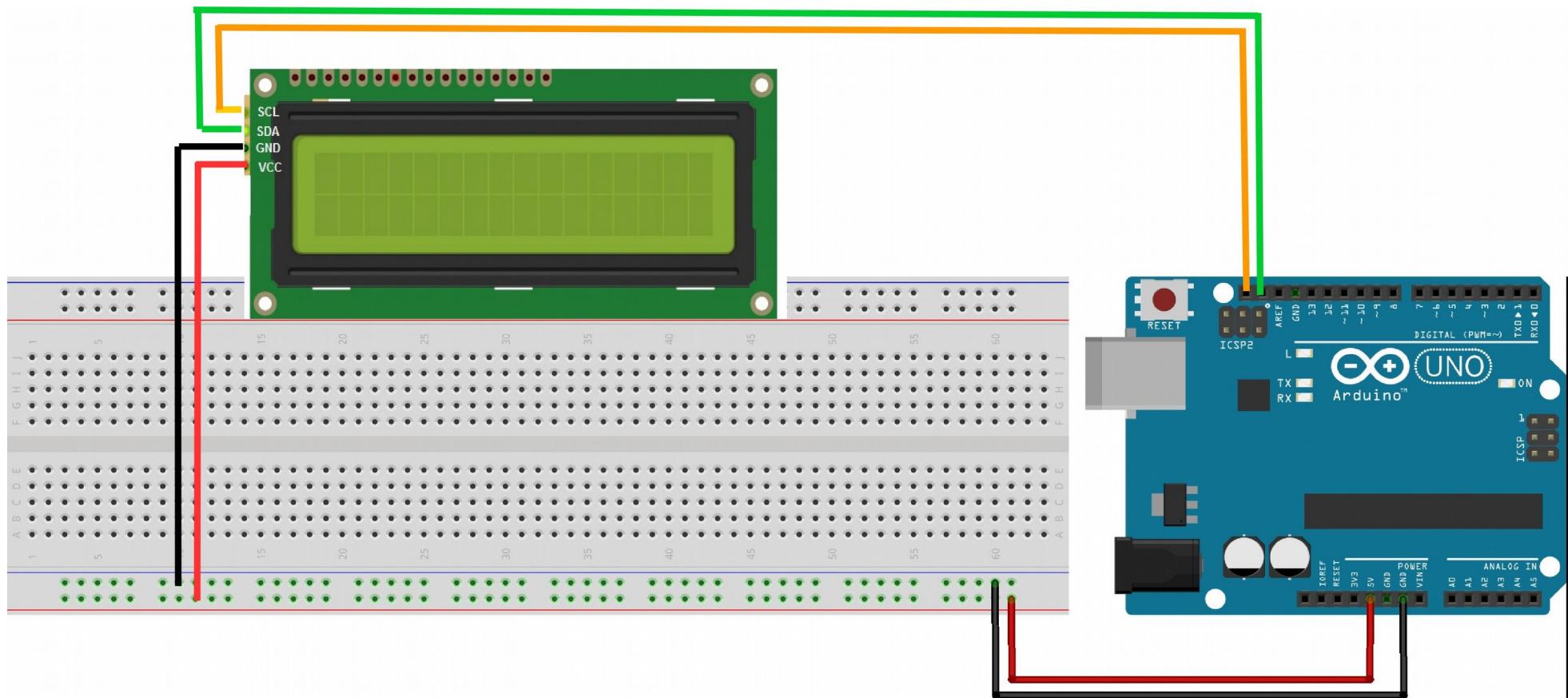
PROGRAMMAZIONE

119



PROGRAMMAZIONE

120



fritzing

PROGRAMMAZIONE

121

```
#include <Wire.h>  
  
#include <LiquidCrystal_I2C.h>  
  
LiquidCrystal_I2C lcd(0x3F, 16, 2);
```

In alto nel
programma

```
lcd.init();  
lcd.backlight();
```

Nel
setup()

PROGRAMMAZIONE

121

sketch_jun28a§

```
1 #include <Wire.h>
2
3 #include <LiquidCrystal_I2C.h>
4
5 LiquidCrystal_I2C lcd( 0x3F, 16, 2 ); // inserisco i dati del display
6
7 void setup() {
8
9 lcd.init(); // inizializzo il display
10 lcd.backlight(); // accendo il LED del display
11 }
12
13 void loop() {
14 // put your main code here, to run repeatedly:
15
16 }
```

PROGRAMMAZIONE

122

SCRIVERE SUL DISPLAY

```
lcd.setCursor(posizione, linea);
```

```
lcd.print("Hello World");
```

```
lcd.print(variabile);
```

PROGRAMMAZIONE

123

```
1 #include <Wire.h>
2
3 #include <LiquidCrystal_I2C.h>
4
5 LiquidCrystal_I2C lcd( 0x3F, 16, 2 ); // inserisco i dati del display
6
7 void setup() {
8
9 lcd.init(); // inizializzo il display
10 lcd.backlight(); // accendo il LED del display
11 }
12
13 void loop() {
14     lcd.setCursor ( 1, 0 );
15     lcd.print ( "Hello, World" );
16     lcd.setCursor ( 0, 1 );
17     lcd.print ( "16x2 LCD Screen" );
18 }
19 }
```

PROGRAMMAZIONE

Kmh 00 t 00.00
h 00:00:00

124

```
4
5 LiquidCrystal_I2C lcd(0x3F,16,2); // metto i dati del display
6
7 byte velocita=0; // memorizzo la velocità in kmh
8 float gradi=0; // memorizzo la lettura della temperatura
9 byte ore=0; // memorizzo le ore di viaggio
10 byte minuti=0; // memorizzo i minuti di viaggio
11 byte secondi =0; // memorizzo i secondi di viaggio
12
13 void setup()
14 {
15   lcd.init(); // inizializzo il display
16   lcd.backlight(); // accendo il led del display
17
18 }
```

PROGRAMMAZIONE

125

```
21 void loop()
22 {
23     lcd.setCursor(0, 0);          // metto il cursore in alto a sinistra
24     lcd.print("Kmh ");           // scrivo Kmh più uno spazio
25
26     if (velocita<10) lcd.print("0"); // se la velocità è minore di 10 scrivo uno 0
27     lcd.print(velocita);          // scrivo il valore della velocità
28     lcd.setCursor(9,0);          // metto il cursore in posizione 10 nella prima linea
29     lcd.print("t ");              // scrivo "t" più uno spazio
30     if (gradi<10) lcd.print("0");// se gradi è minore di 10 scrivo uno 0
31     lcd.print(gradi);            // scrivo i gradi
32     lcd.setCursor(3,1);          // metto il cursore in posizione 3 basso
33     lcd.print("h ");              // scrivo "h" più uno spazio
34     if (ore<10) lcd.print("0");// se il valore delle ore è minore di 10 scrivo uno 0
```

PROGRAMMAZIONE

126

La funzione per calcolare il tempo del viaggio

Mettiamo il valore di “`millis()`” nella variabile diviso di 1000
in modo di avere un numero in secondi.

In un'ora abbiamo 3600 secondi
In un minuto abbiamo 60 secondi

Ci serve sapere ore, minuti e secondi.

Se il valore di “tempo” è uguale a 12015 secondi:

$$12000 / 3600 = 3.3375$$

Quando metto questo numero nella variabile “ore” di tipo “int”
nella variabile si memorizza il valore 3.

PROGRAMMAZIONE

Se ora metto nella variabile “tmp” il valore della variabile “tempo” meno la variabile “ore” moltiplicata per 3600 ottengo il numero dei secondi che formano i minuti.

127

`tmp = tempo - (ore * 3600);`

$$3 * 3600 = 10800$$

$$12015 - 10800 = 1215$$

`tmp= tempo / 60;`

$$1215 / 60 = 20.25$$

`Minuti = tmp;`

`secondi = tempo - (minuti * 60);`

$$20 * 60 = 1200$$

$1215 - 1200 = 15$ che sono i secondi.

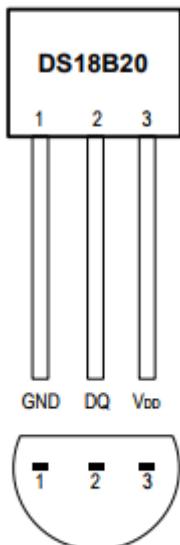
PROGRAMMAZIONE

128

```
48 void tempo_di_viaggio(){      // calcolo il tempo di viaggio
49   tempo = millis() / 1000; // memorizzo il tempo in secondi
50   tmp = tempo / 3600; // divido i secondi per 3,600 per trovare le ore
51   ore = tmp; // memorizzo la parte intera della divisione
52   tempo = tempo - (ore * 3600); // moltiplico la parte intera per 3,600 e la sottraggo al totale dei secondi per trovare i minuti
53   tmp = tempo / 60; // divido i secondi restanti per 60 per calcolare i minuti
54   minuti = tmp; // memorizzo la parte intera del calcolo dei minuti
55   secondi = tempo - (minuti * 60); // moltiplico i minuti per 60 e la sottraggo dal totale per trovare i secondi
56
57 }
58
```

PROGRAMMAZIONE

129



BOTTOM VIEW

TO-92
(DS18B20)

Gestor de Librerías

Tipo Todos Tema Todos Onewire

MAX31850 DallasTemp by Adafruit
A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!) A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!)
[More info](#)

MAX31850 OneWire by Adafruit
A version of the OneWire Arduino library with MAX31850 support A version of the OneWire Arduino library with MAX31850 support
[More info](#)

OneWire by Jim Studt, Tom Pollard, Robin James, Glenn Trewitt, Jason Dangel, Guillermo Lovato, Paul Stoffregen, Scott Roberts, Bertrik Sikken, Mark Tillotson, Ken Butcher, Roger Clark, Love Nystrom Versión 2.3.3 INSTALLED
Access 1-wire temperature sensors, memory and other chips.
[More info](#)

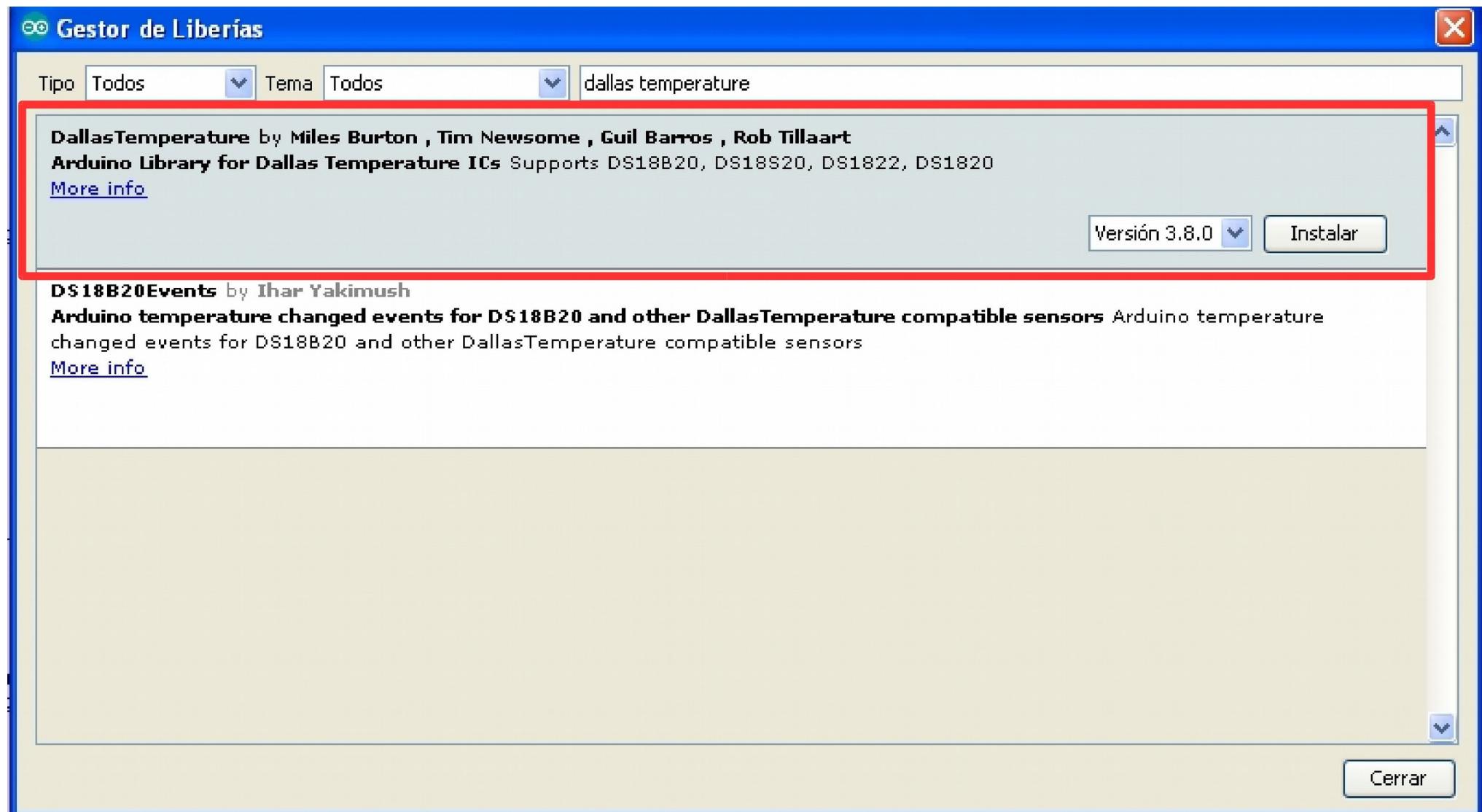
Seleccione vers... Instalar

OneWireHub by Ingmar Splitt, orgua, MarkusLange, Shagrat2
OneWire slave device emulator with support for up to 32 simultaneous 1wire devices. supported sensors: BAE910, DS1822, DS18B20, DS18S20, DS1990, DS2401, DS2405, DS2408, DS2411, DS2413, DS2423, DS2431, DS2432, DS2433, DS2438, DS2450, DS2501, DS2502, DS2503, DS2505, DS2506, DS2890

Cerrar

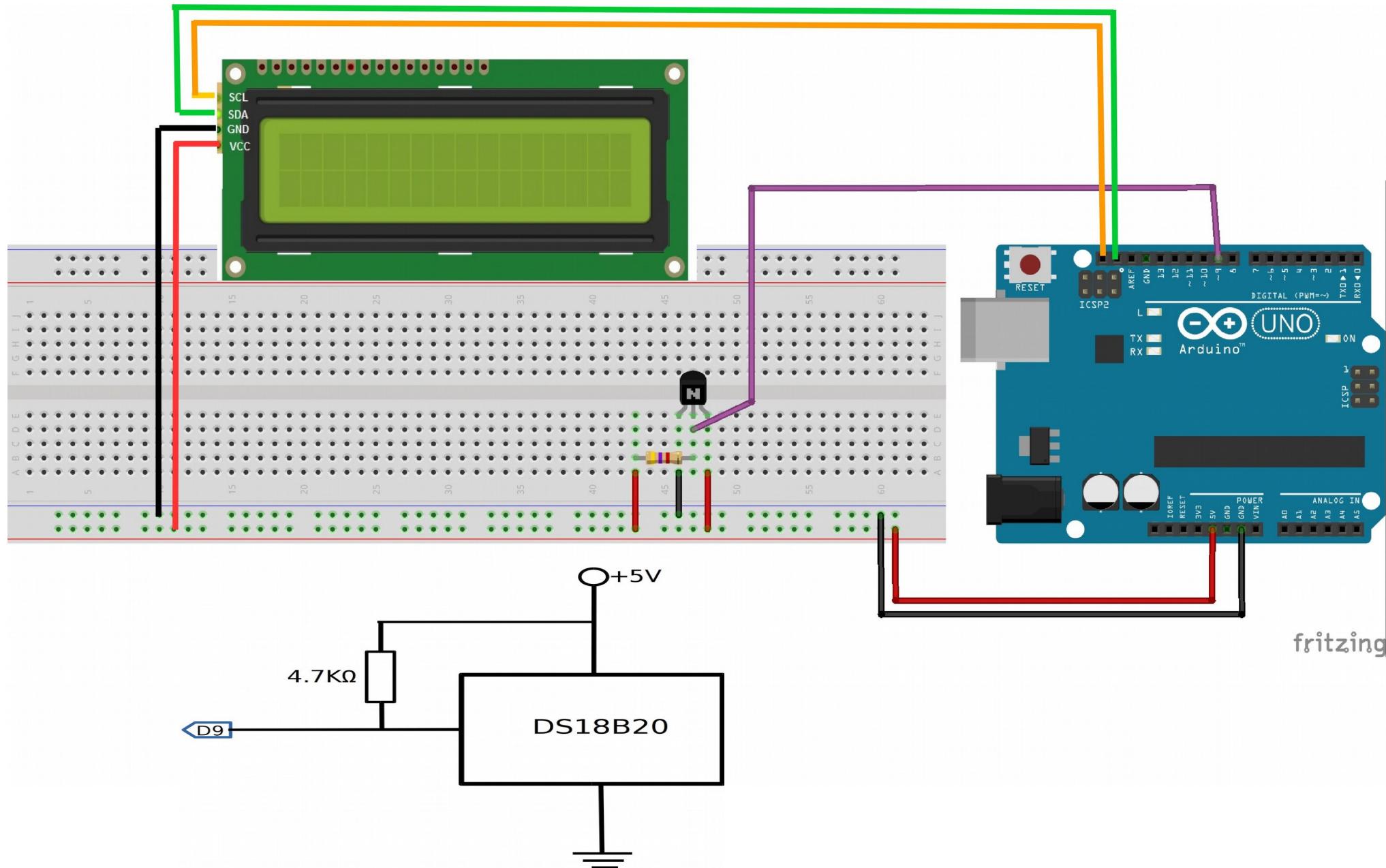
PROGRAMMAZIONE

130



PROGRAMMAZIONE

130



PROGRAMMAZIONE

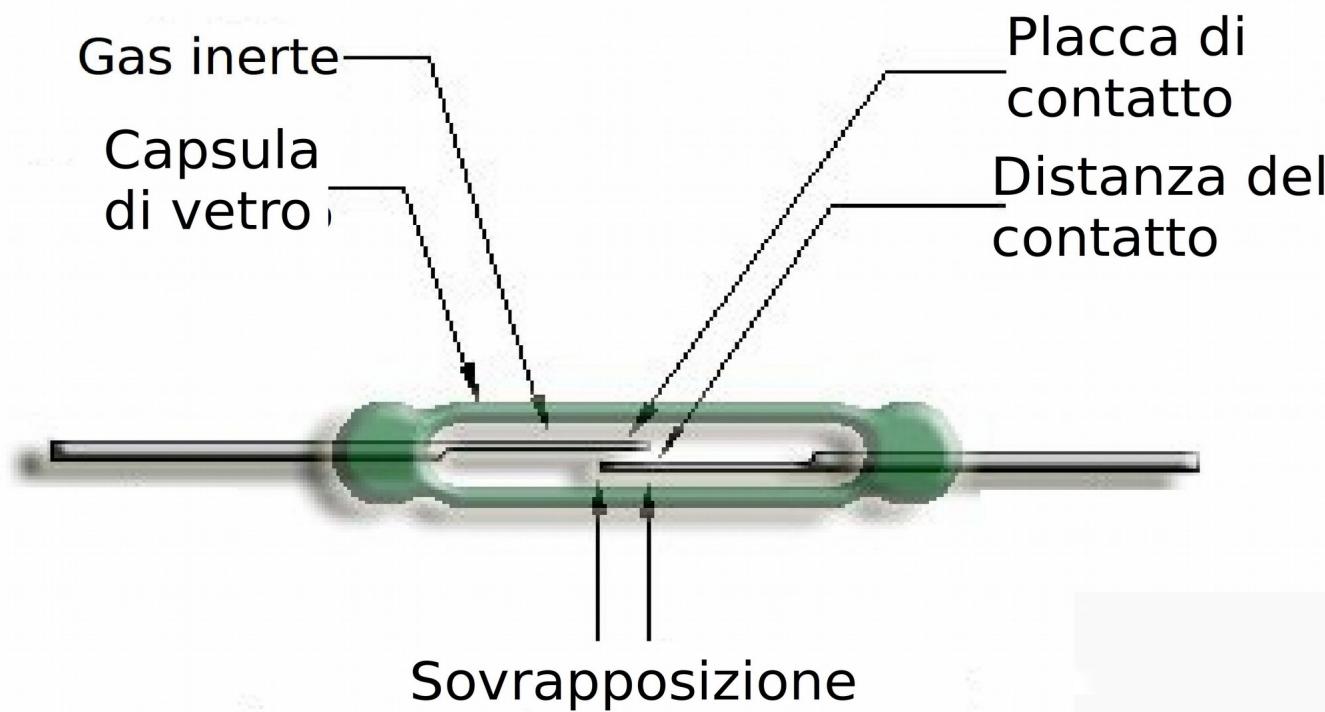
131

```
1 #include <OneWire.h>
2 #include <DallasTemperature.h>
3 #define DQ 9 // è la connessione del DQ del DS18B20
4 OneWire oneWire(DQ); // comunica a la libreria "oneWire" la connessione del DS18B20
5 DallasTemperature sensorDS18B20(&oneWire); // comunica a la libreria "DallasTemperature" la connessione del DS18B20
6
7
8 #include <Wire.h>
9 #include <LiquidCrystal_I2C.h>

66 void temperatura(){ // leggo la temperatura
67     sensorDS18B20.requestTemperatures(); // richiedo la lettura della temperatura alla libreria
68     gradi = sensorDS18B20.getTempCByIndex(0); // memorizzo in gradi la lettura della libreria
69 }
70
```

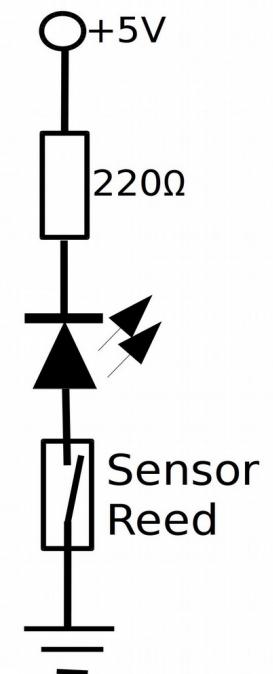
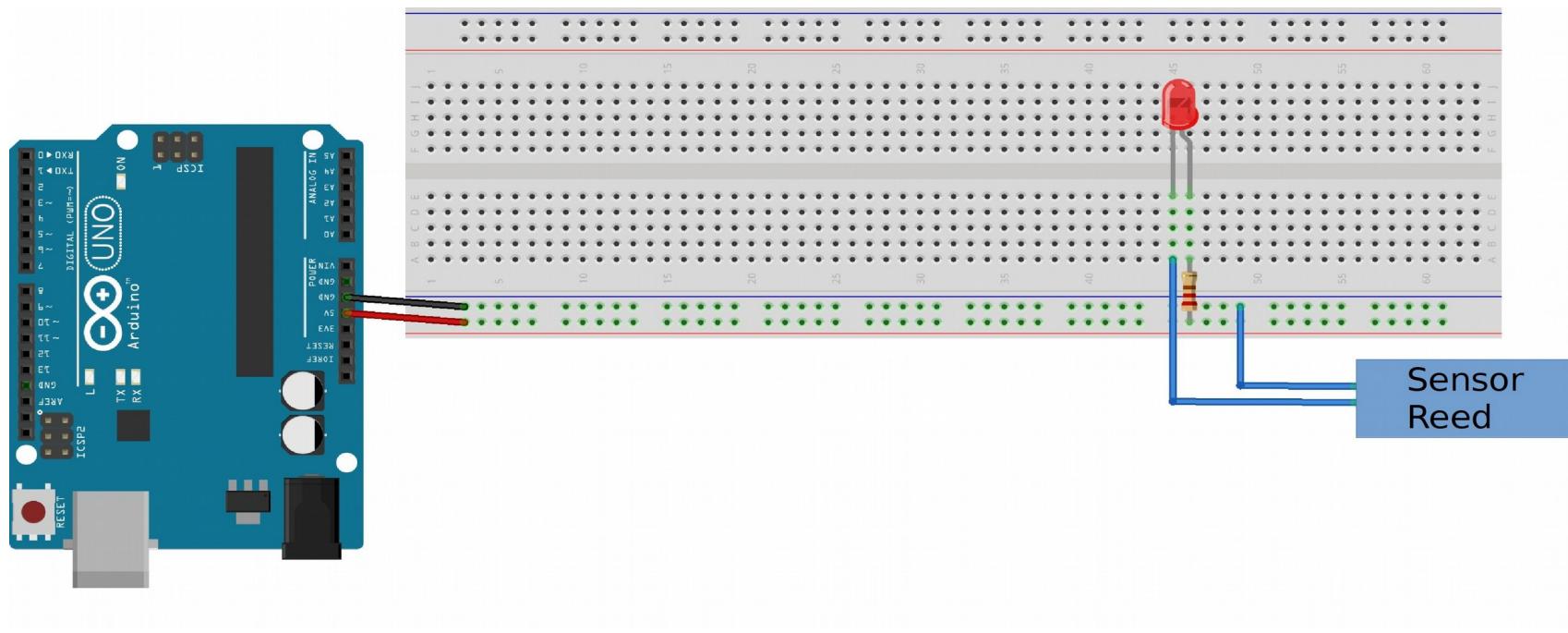
PROGRAMMAZIONE

133



PROGRAMMAZIONE

133



PROGRAMACIÓN

L'interrupt

134

`attachInterrupt(numero dell'Interrupt, nome della funzione, modo);`

Come numero del Interrupt scriviamo 0 se il sensore è collegato all'ingresso digitale 2 o scriviamo 1 se è collegato al 3.

LOW chiama la funzione quando l'ingresso ha un valore basso
CHANGE chiama la funzione quando l'ingresso cambia di stato
RISING chiama la funzione quando l'ingresso passa da basso a alto
FALLING chiama la funzione quando l'ingresso passa da alto a basso

`detachInterrupt(numero del interrupt);`

`interrupts(numero del interrupt);`

PROGRAMMAZIONE

Il calcolo della velocità.

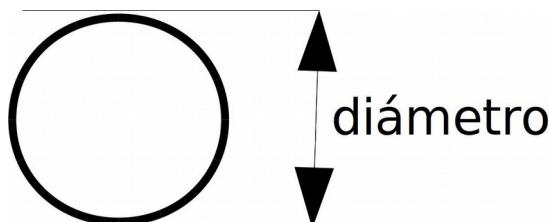
135

Il calcolo per sapere i chilometri all'ora è :

Metri al secondo * 3.6

Se con la bicicletta stiamo facendo 5 metri al secondo
in un'ora facciamo 18 chilometri.

Per sapere quanta strada stiamo percorrendo ci serve conoscere
il diametro in centimetri della ruota della bicicletta
per trovare la circonferenza.



$$65 * 3.14 = 204.1$$

Il diametro lo moltiplichiamo per 3.14 che è
il valore del π pi.

Se abbiamo una ruota che ha un diametro di
65 centimetri abbiamo una circonferenza di
204.1 centimetri.

Però ci serve la circonferenza in metri quindi dividiamo per 100
perché in un metro ci sono 100 centimetri.

PROGRAMACIÓN

136

Se la ruota fa un giro al secondo stiamo percorrendo:

$$2.041 * 3.6 = 7.3476 \text{ chilometri all'ora.}$$

Per sapere quanto tempo impiega la ruota per fare un giro usiamo la funzione `millis()`. Quando l'ingresso passa a basso è perché il magnete si trova di fronte al sensore e leggiamo un tempo e quando ritorna bassa ne leggiamo un altro. Se sottraiamo il tempo nuovo al vecchio sappiamo quanto tempo ha impiegato la ruota per fare un giro.

Se ottengo un risultato di 750 ms (milli secondi) devo fare il calcolo con la formula:

Metri al secondo * 3.6

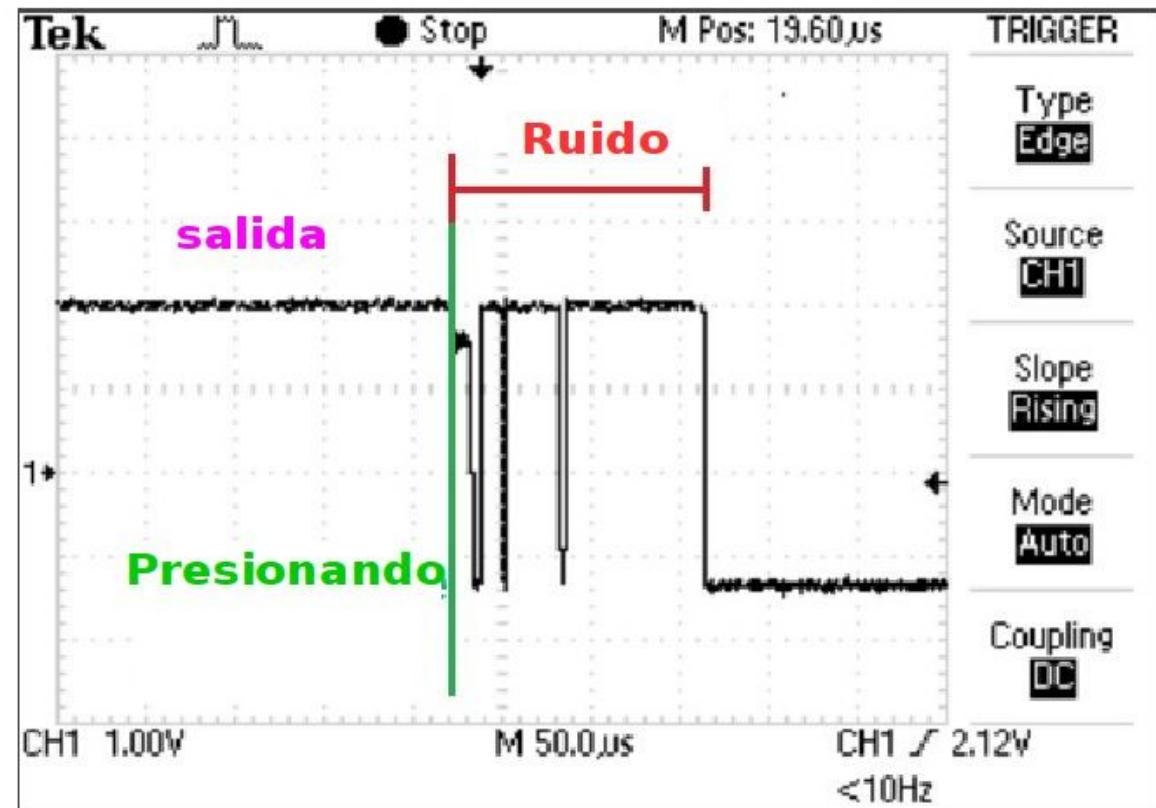
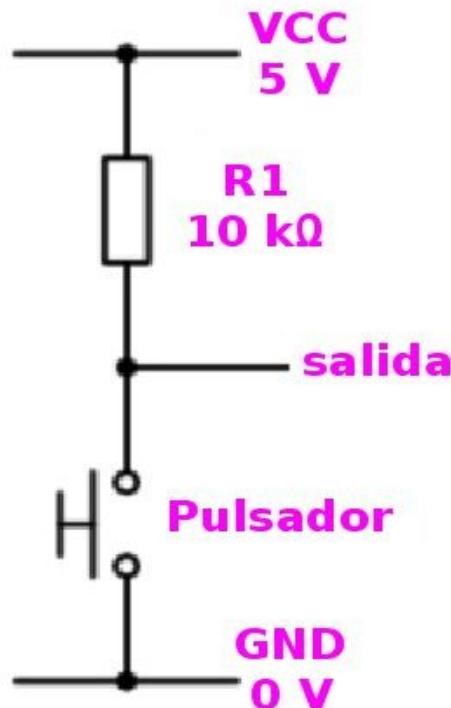
Però in questa formula i secondi sono una costante e sono i metri che cambiano, nel nostro caso, i metri sono una costante perché sono la circonferenza della ruota quello che cambia è il tempo del giro che può essere più o meno rapido e oltretutto il tempo è in milli secondi. La formula quindi può essere:

```
velocita=(circonferenza * 3600)/tempo_del_giro;
```

PROGRAMMAZIONE

Il rumore del pulsante

137



PROGRAMMAZIONE

137

Scriviamo la funzione che chiamerà l' Interrupt che si chiama:

void Kmh(){

```
84
85 void Kmh(){ // calcolo la velocità
86
87 x_nuovo=millis(); // in x nuovo memorizzo il valore di millis()
88 if (x_nuovo>x_vecchio+76){ // se il valore nuovo è minore del valore vecchio più 76 milli secondi è un rumore
89 // del sensore e non è valida la lettura, se è maggiore la lettura è valida
90 tempo_del_giro=x_nuovo-x_vecchio; // il tempo del giro della ruota è la differenza del tempo nuovo-il vecchio
91 x_vecchio=x_nuovo; // memorizzo il valore nuovo nel vecchio per prendere un nuovo valore
92 }
93
94 velocita=(circonferenza*3600)/tempo_del_giro;// calcolo la velocità
95 c=1; // metto a 1 la variabile di controllo
96
97 }
98
99
100 tempo_di_viaggio(); // chiamo la funzione "tempo di viaggio"
101 temperatura(); // chiamo la funzione per leggere la temperatura
102 if (c==1) cl=millis(); // controllo se la ruota è ferma memorizzando il valore di millis()
103 if (c==0){
104     if((cl+2000)<millis()) velocita=0; // se dopo due secondi non si esegue la funzione kmh è perché
105 // la ruota è ferma e metto a 0 la velocità
106 }
107 c=0; // metto a 0 la variabile di controllo
108 }
```

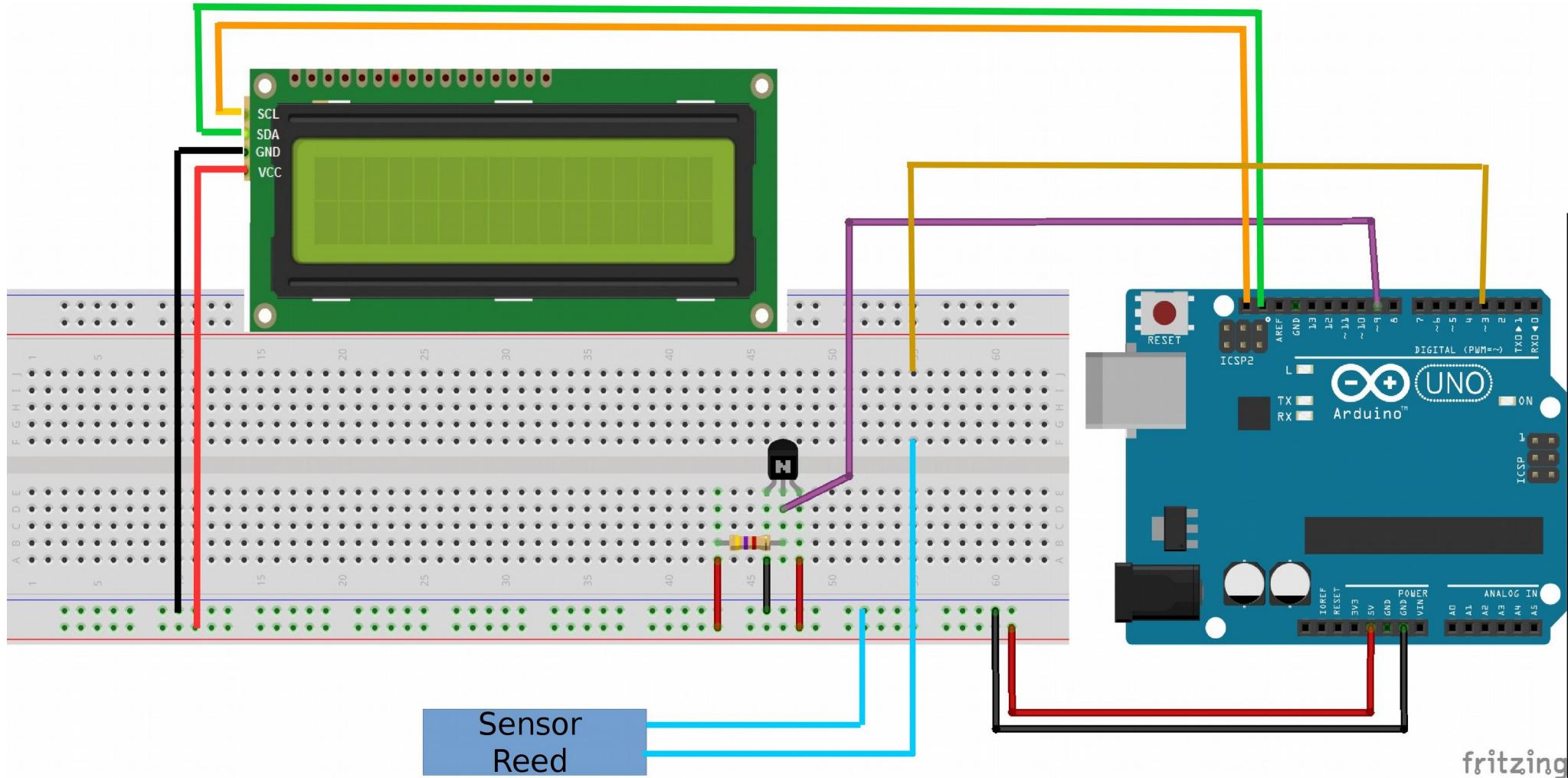
PROGRAMMAZIONE

139

```
21 unsigned long x_nuovo=0; // memorizza il valore di millis() quando passa il magnete davanti al sensore
22 unsigned long x_vecchio=0; // memorizza il tempo della lettura precedente
23 int tempo_del_giro=0; // memorizza il tempo di un giro della ruota
24 float circonferenza=1.97;// diametro in metri della ruota
25 byte c=0; // è la variabile di controllo se si ferma la ruota
26 unsigned long cl=0; // memorizza il valore di millis() per vedere se la ruota è ferma
27
28 #define DISPLAY 5 // è il collegamento del led del display
29 #define foto_resistenza 0 // è il collegamento della fotoresistenza
30
31 void setup()
32 {
33     lcd.init(); // inizializzo il display
34     lcd.backlight(); // accendo il led del display
35     sensorDS18B20.begin(); //inizializzo la libreria del sensore di temperatura
36     pinMode(sensore,INPUT_PULLUP); // imposto come ingresso il pin del sensore reed
37     attachInterrupt(1, Kmh, LOW); // attivo l'interrupt che chiama la funzione Kmh() quando l'ingresso 3 va basso
38
39 }
```

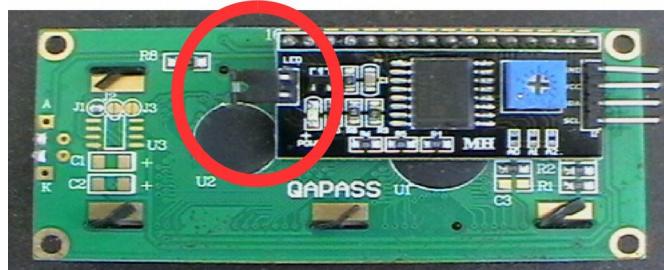
PROGRAMMAZIONE

139

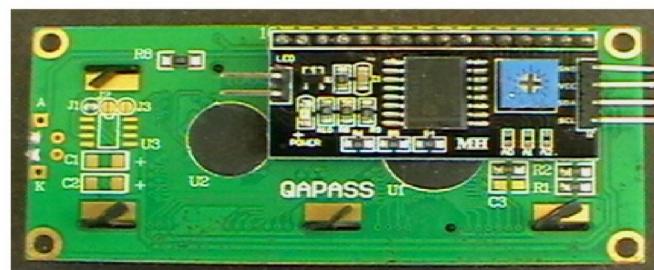


PROGRAMMAZIONE

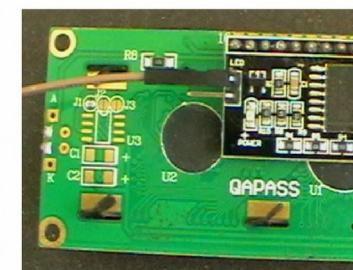
140



1



2

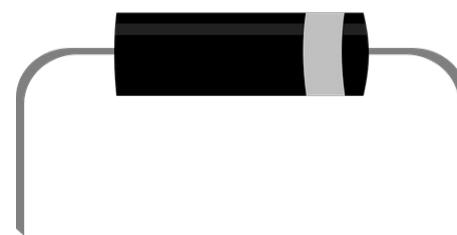
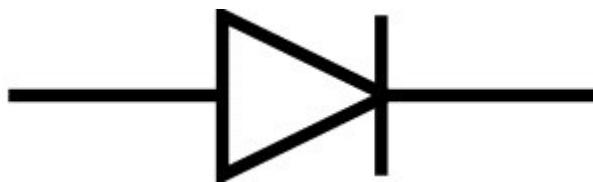


3

ELETTRONICA

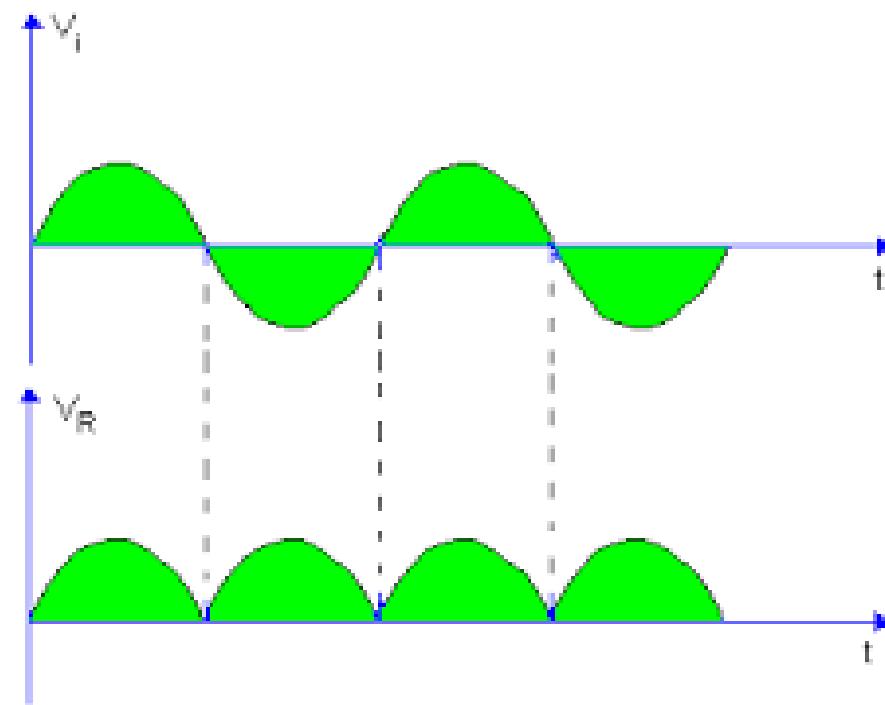
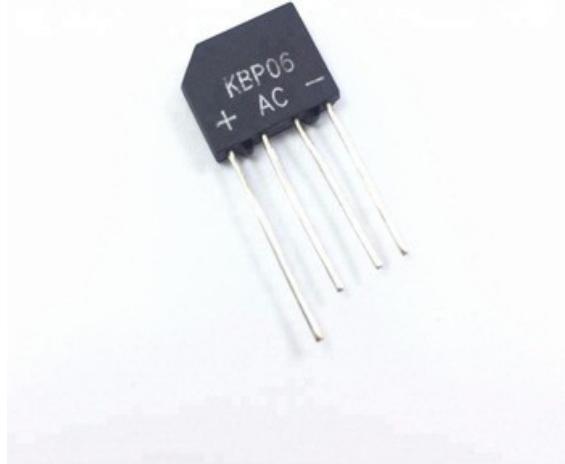
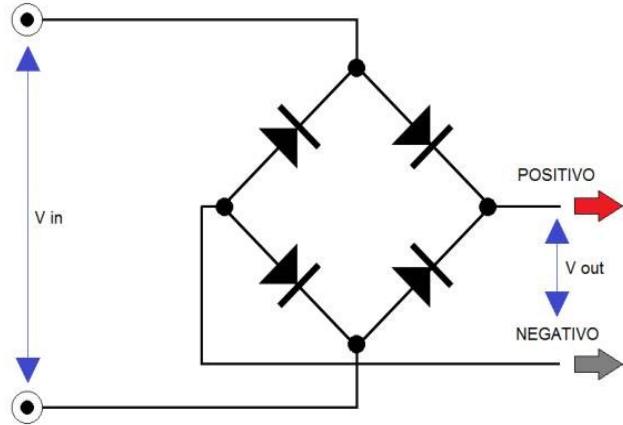
I DIODI

141



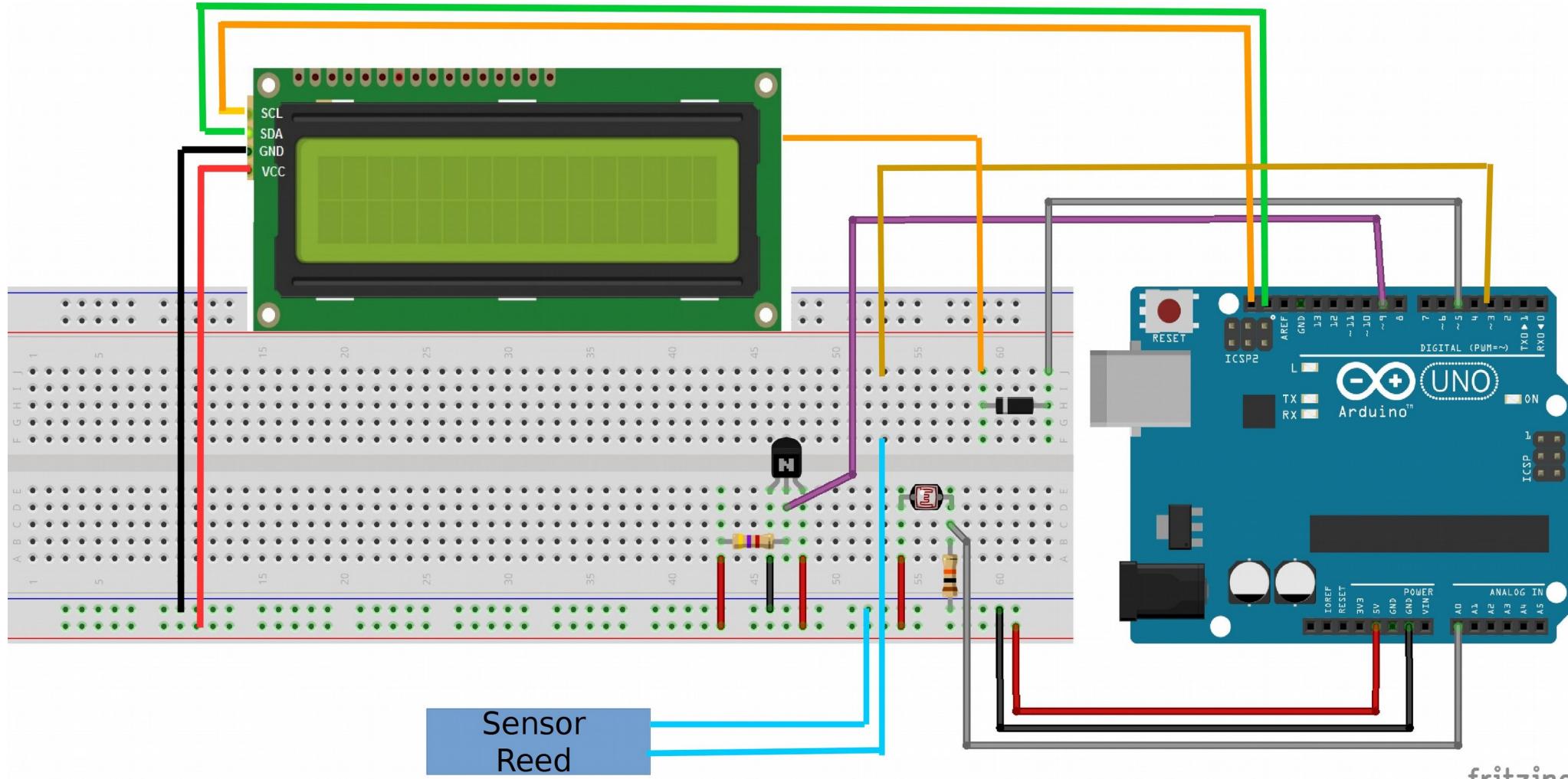
ELETTRONICA

142



PROGRAMMAZIONE

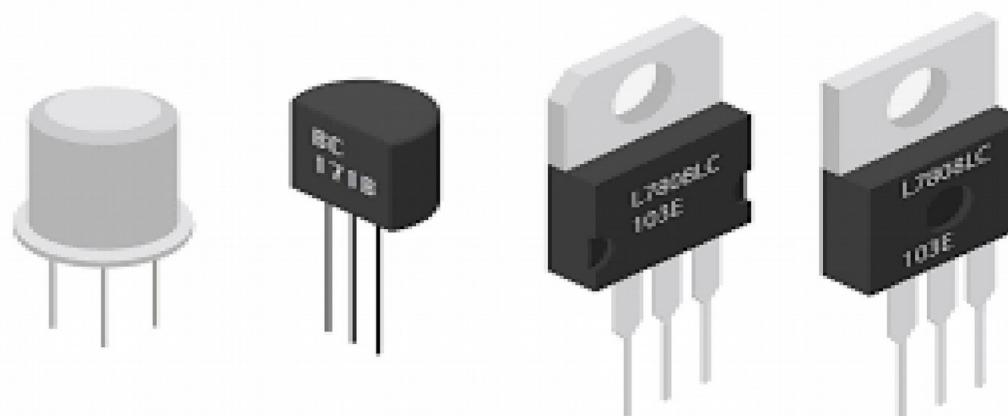
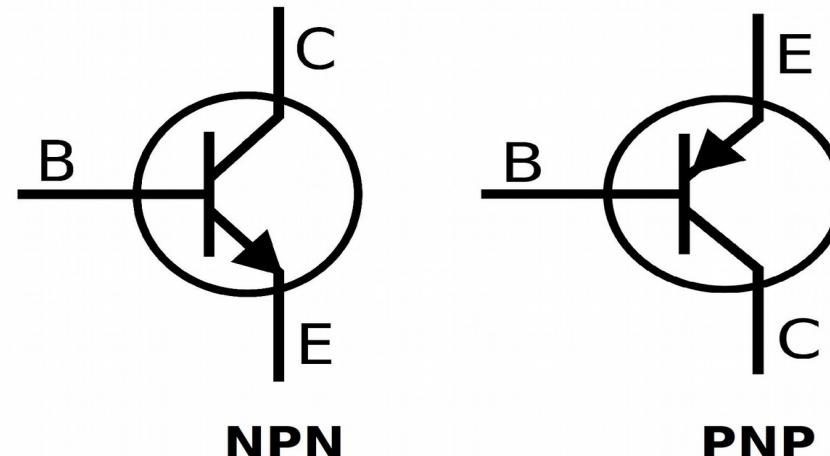
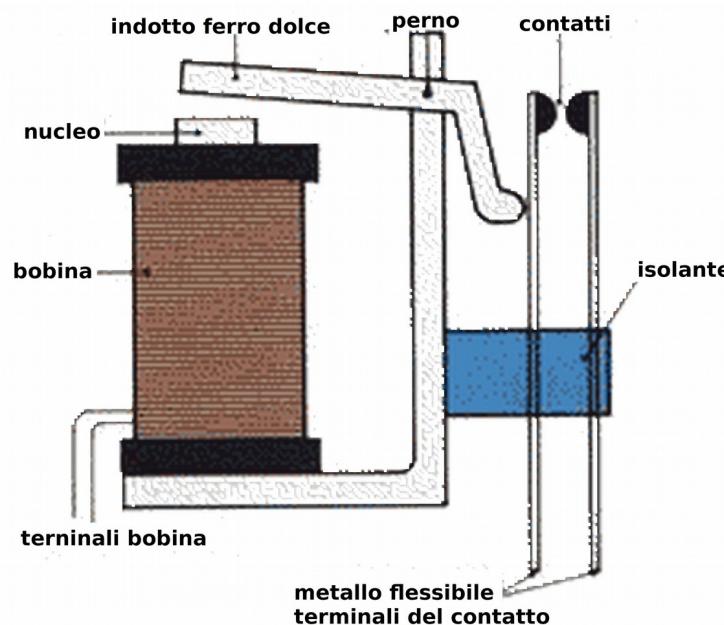
143



```
71 if ((analogRead(foto_resistenza)/4) < 60) analogWrite(DISPLAY,60); // metto un limite alla luce del display  
72 if ((analogRead(foto_resistenza)/4) > 60) analogWrite(DISPLAY, (analogRead(foto_resistenza)/4));  
73 }  
74 }
```

ELETTRONICA

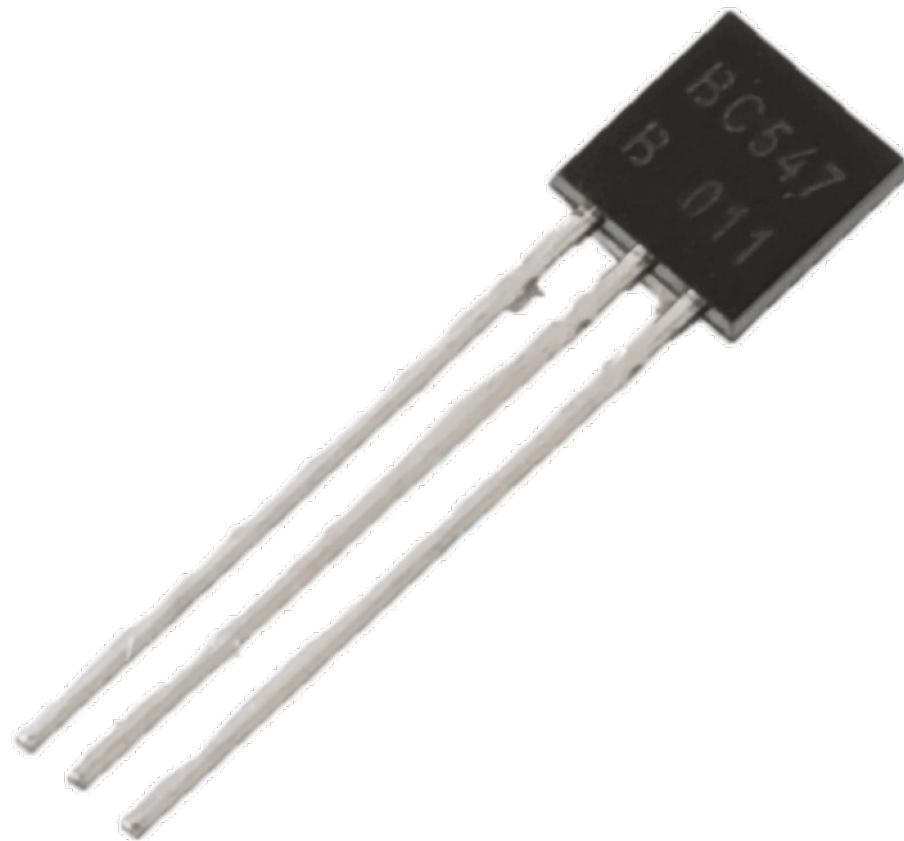
146



ELETTRONICA

147

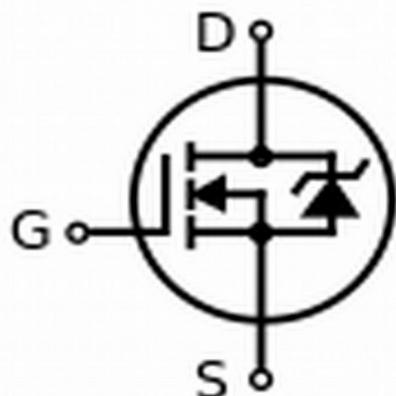
VCB = 45 V Max
VCE = 30 V Max
VEB = 6 V Max
IC = 100 mA
Ptot = 300mW
Hfe = 100 - 200
Ft = 50 MHz



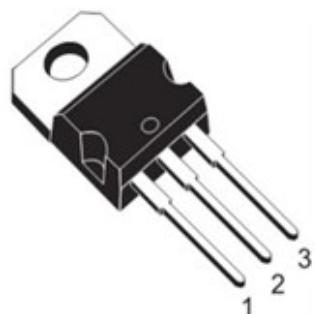
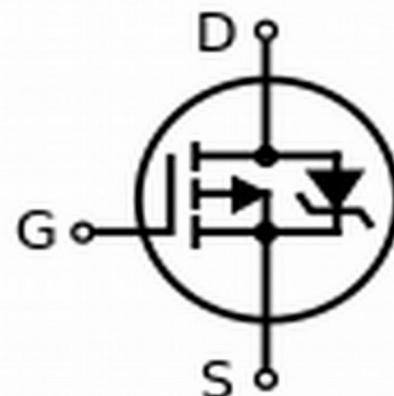
ELETTRONICA

148

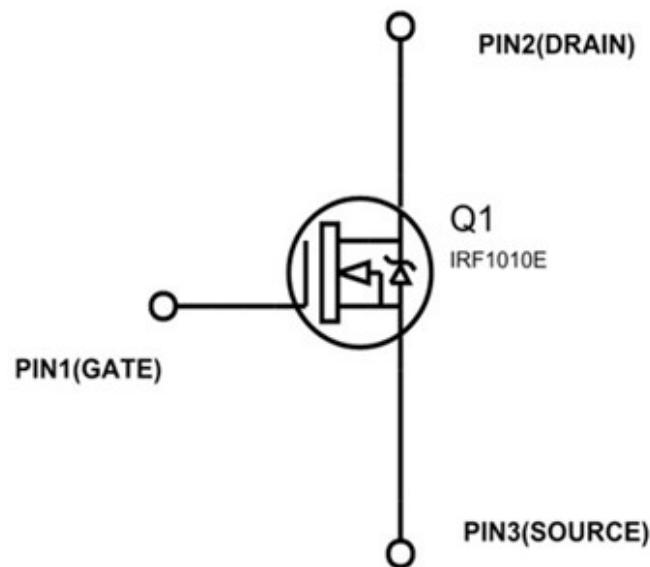
N



P

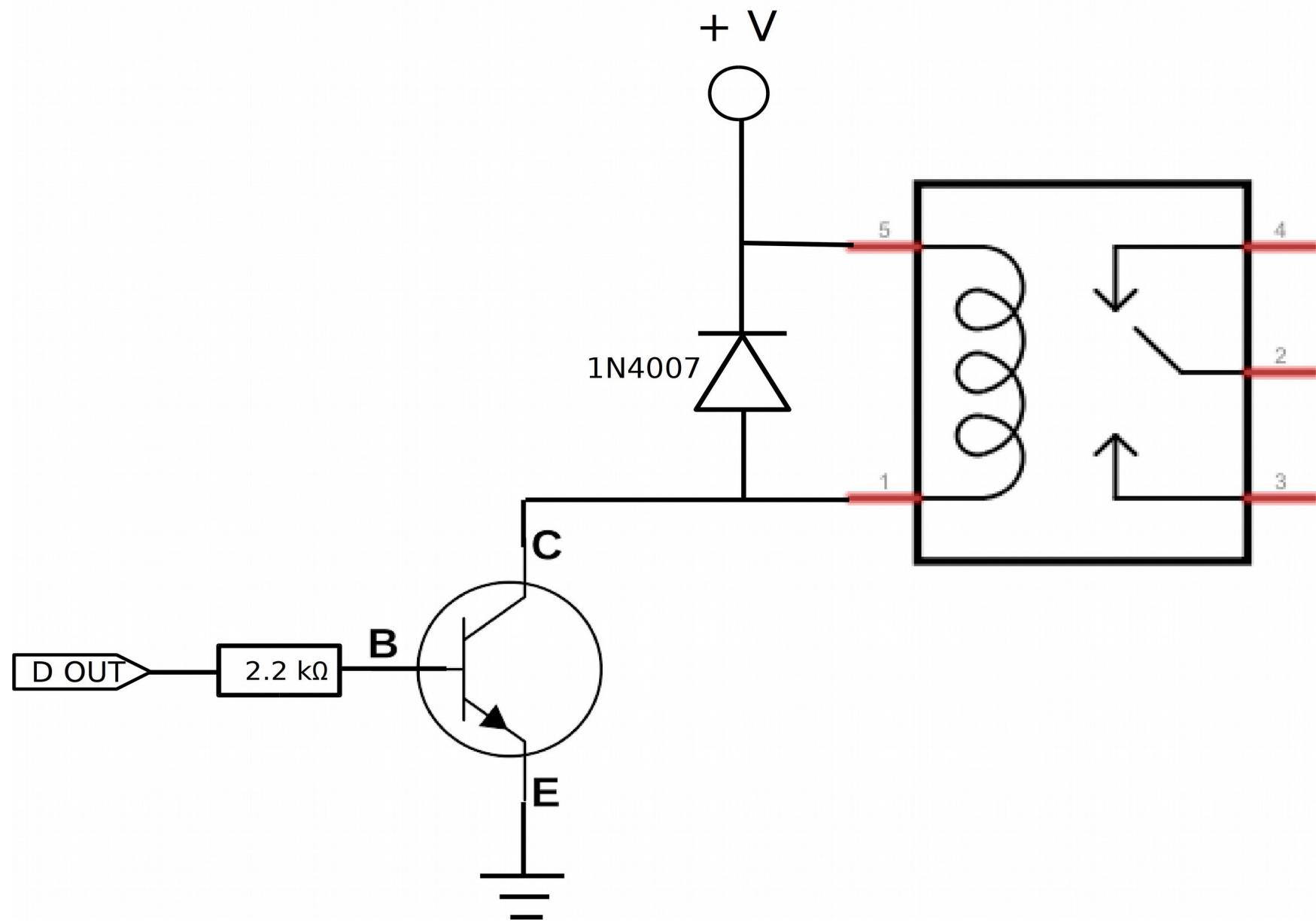


TO-220



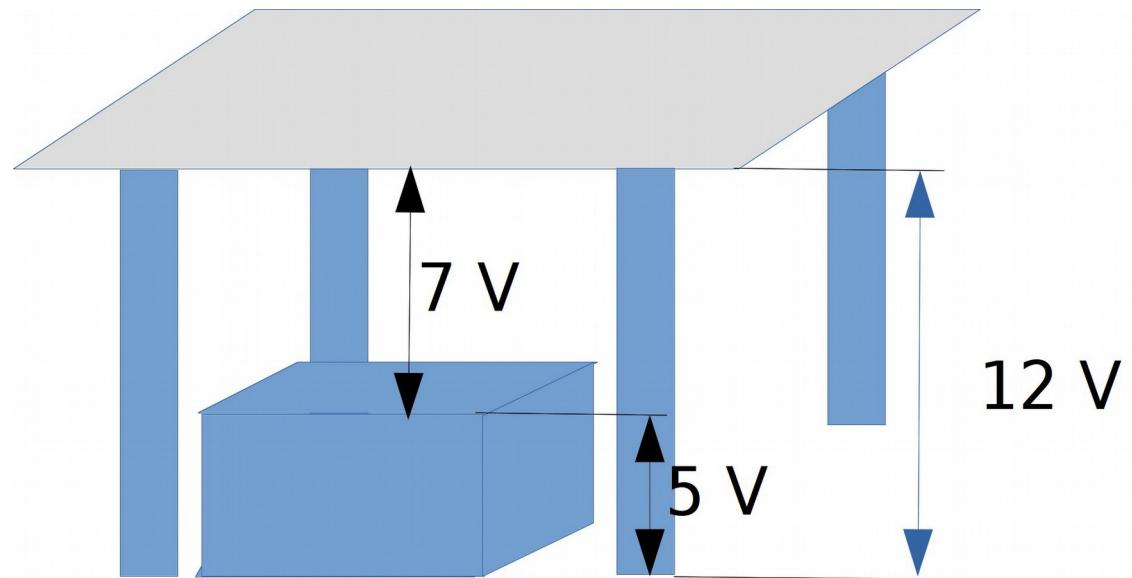
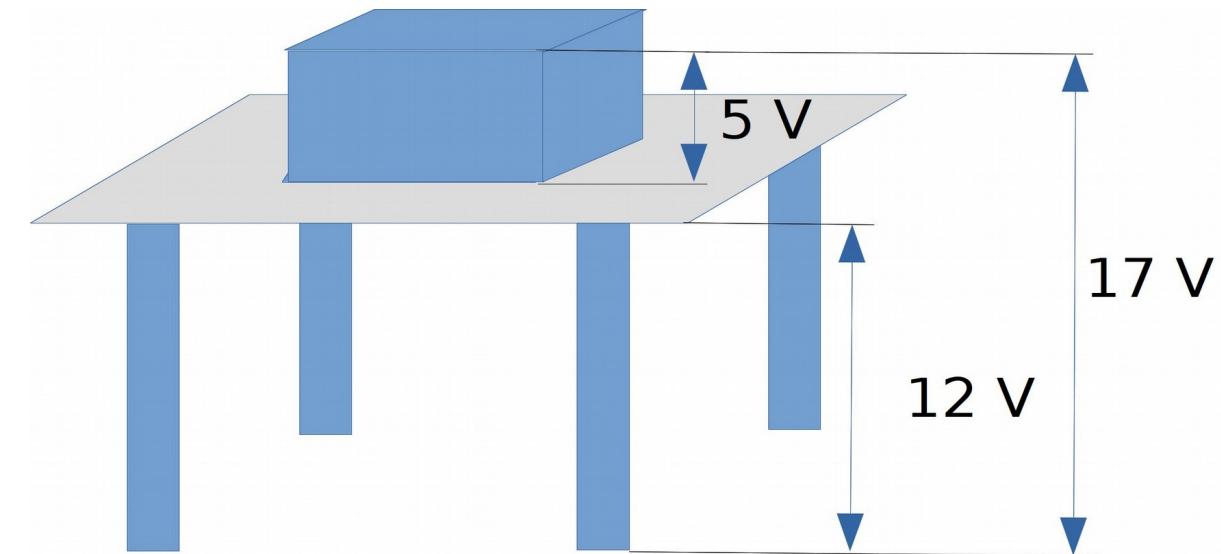
ELETTRONICA

149



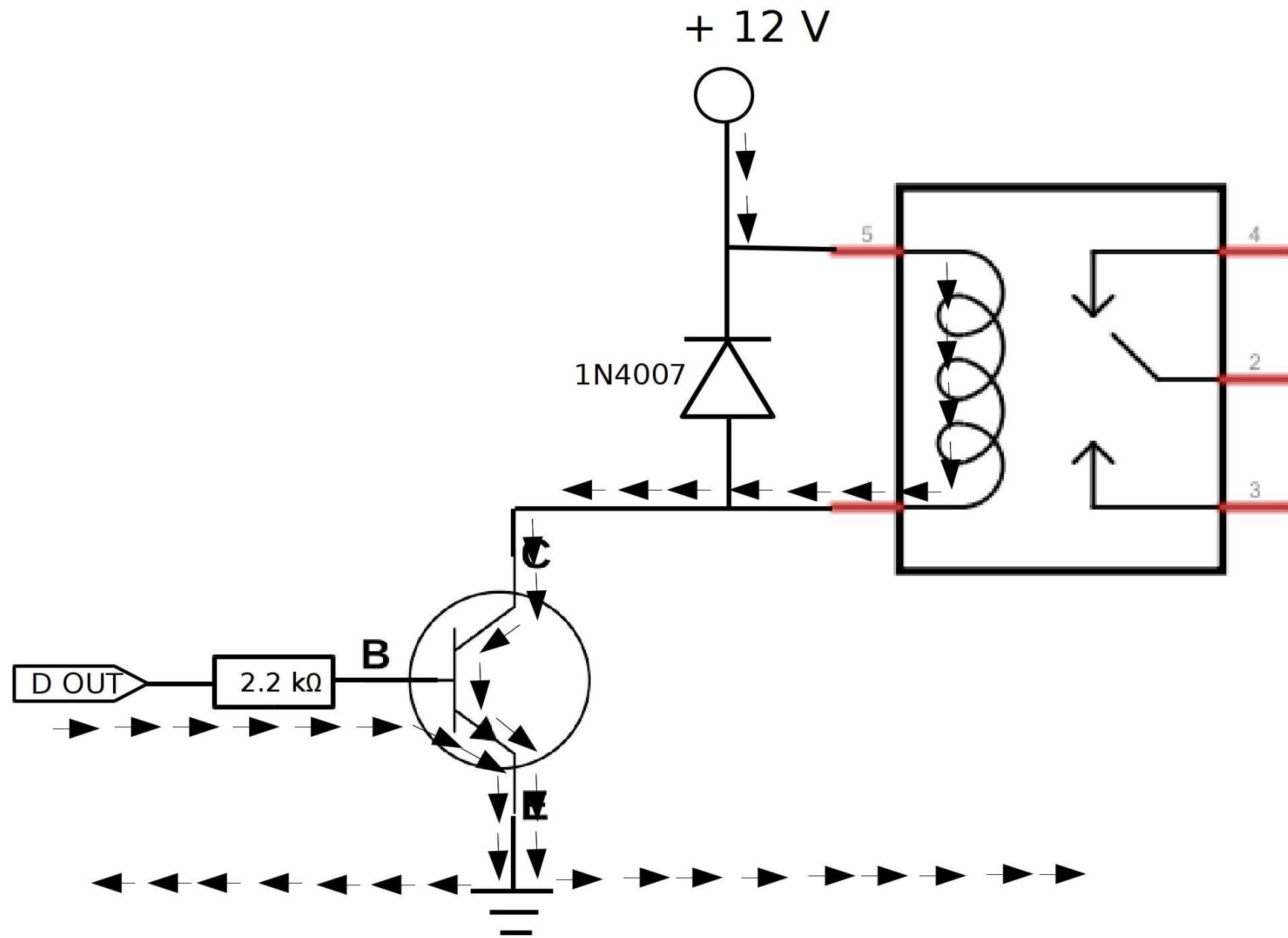
ELETTRONICA

150



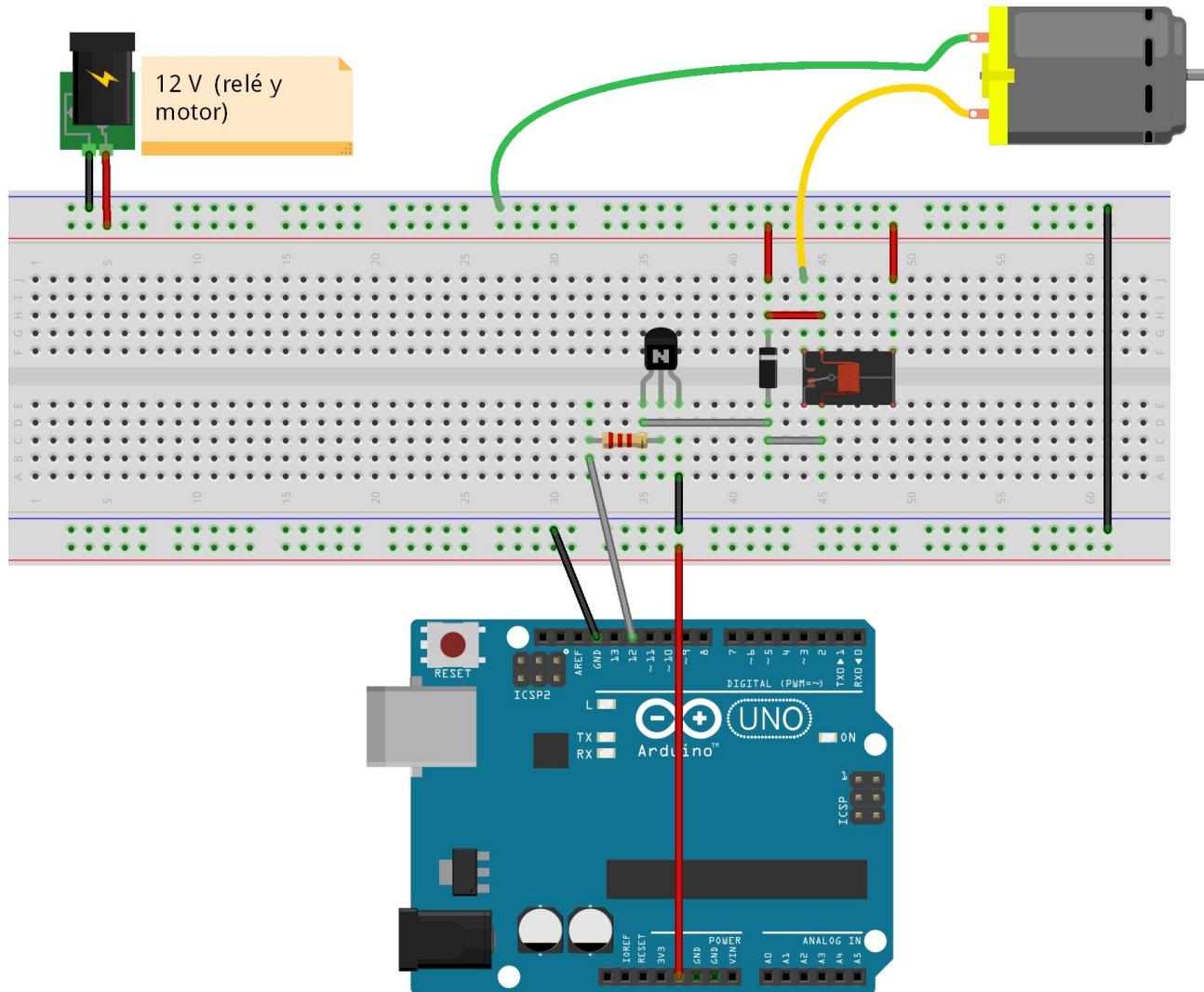
ELETTRONICA

151



ELETTRONICA

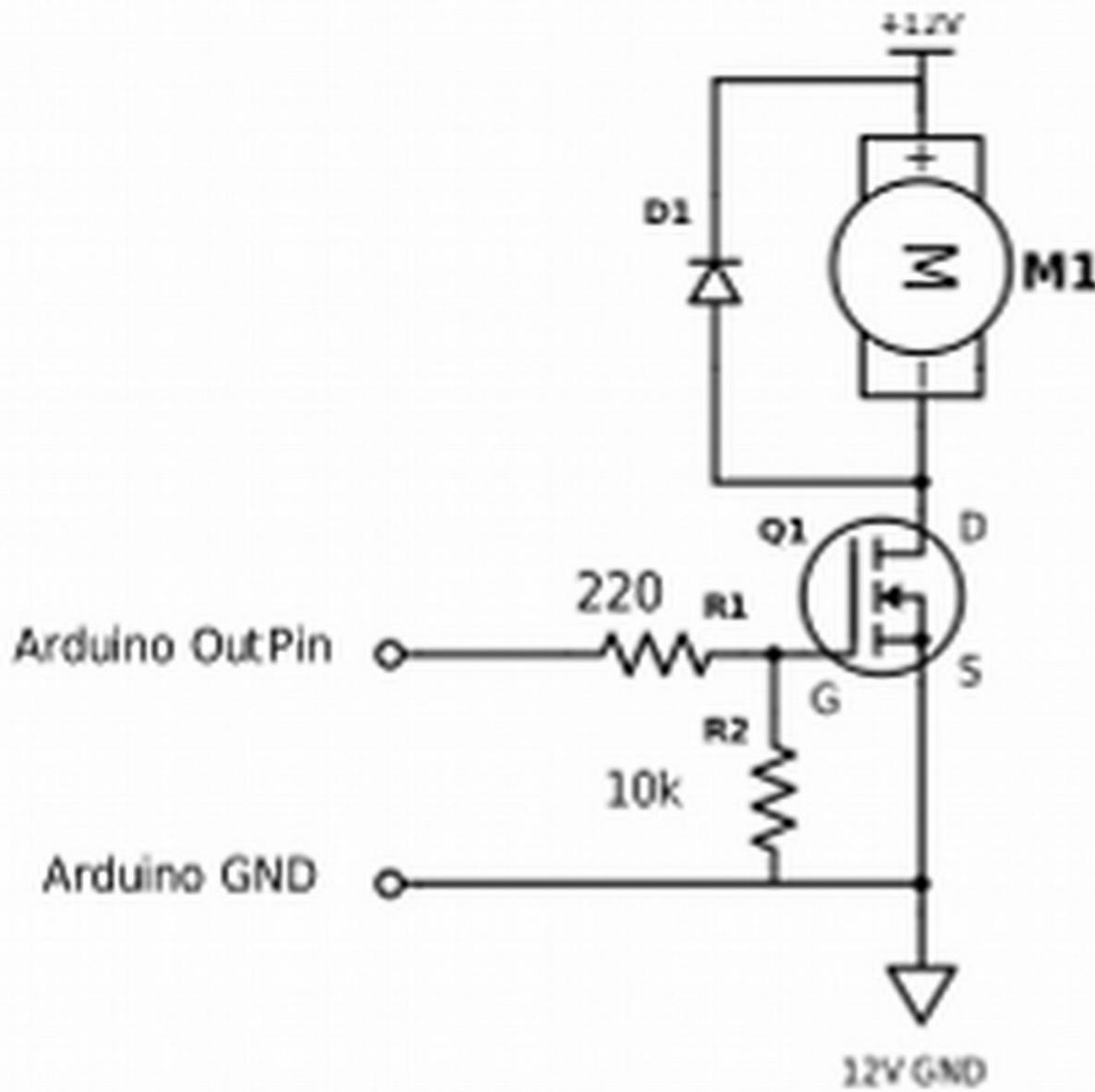
152



fritzing

ELETTRONICA

153



ARDUINO

154

```
1 #define MOTORE 11
2 #define POTENZIOMETRO 0
3
4 int valore = 0; // in "valore" metto la lettura del potenziometro
5
6 void setup(){
7   pinMode ( MOTORE, OUTPUT );
8 }
9
10
11 void loop(){
12   valore = analogRead ( POTENZIOMETRO ); // in "valore" metto la lettura del potenziometro
13   valore = valore / 4; // divido la variabile
14   analogWrite ( MOTORE, valore ); // scrivo la velocità del motore
15 }
16
```

ARDUINO

154

