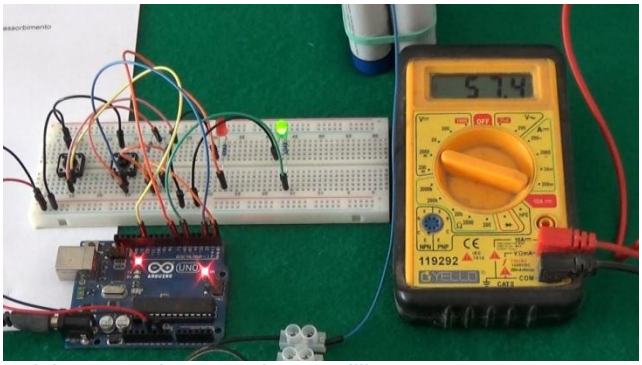
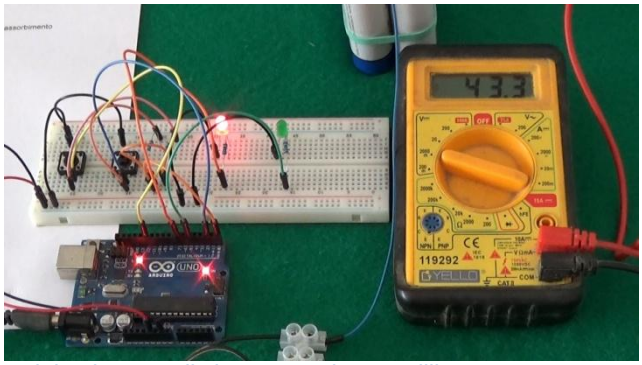
45 – sleep e wake da interrupt esterno - sleep and wake from external interrupt (notes at the end of this section)



Arduino operativo: assorbe 57 milliampere



Arduino in stato di sleep: assorbe 43 milliampere

In questo esercizio si “esplorano” due funzioni apparentemente secondarie di Arduino: la ***sleep*** e l’***interrupt***.

La ***sleep*** e’ una istruzione o meglio una funzione che consente di “addormentare” il microcontrollore e quindi di risparmiare energia. La disattivazione del microcontrollore interrompe, di fatto, il programma in esecuzione ma lascia comunque operative le varie porte e quindi, anche se Arduino e’ “dormiente”, sulle porte di output dichiarate “HIGH” continuera’ ad essere erogata una tensione da 5 volt cosi’ come continueranno ad essere alimentati i sensori e gli attuatori collegati alle porte di alimentazione messe a disposizione da arduino (5v e 3v3). Esistono diversi gradi di “sleep”. In questo esercizio abbiamo utilizzato lo *SLEEP_MODE_PWR_DOWN,* il “sonno profondo” che consente il maggior risparmio di energia.
Il problema di questo “sonno profondo” e’ che Arduino puo’ essere risvegliato solo attraverso pochi sistemi e, tra questi, un interrupt esterno sulla porta 2 oppure 3. Puo’ sembrare una funzione inutile, ma se si vuole costruire sistema in grado di operare, magari per dei mesi, utilizzando la sola alimentazione di una batteria, questa funzione, unita ad altri accorgimenti, puo’ divenire indispensabile.

L’***interrupt*** e’ invece un segnale lanciato da un sensore ed immediatamente recepito ed elaborato da Arduino. E’ un segnale asincrono rispetto al programma in esecuzione (le istruzioni di verifica dello stato del sensore non sono quindi presenti ne’ nel ciclo di loop e ne’ nelle relative routine) che, una volta rilevato, diventa immediatamente prioritario. In pratica quando Arduino riceve un interrupt interrompe il programma che sta eseguendo, lancia una routine che viene eseguita solo in caso di interrupt e, al termine, riprende l’esecuzione del programma dal punto nel quale era stato interrotto. E’ un po’ come una telefonata: quando la riceviamo interrompiamo il lavoro che stiamo facendo, prestiamo attenzione a cio’ che ci viene detto al telefono e, al termine, riprendiamo il lavoro interrotto. L’interrupt puo’ essere lanciato da un evento interno oppure da un evento esterno e quindi, ad esempio, dalla pressione di un pulsante (come in questo esercizio) oppure dall’allarme lanciato da un orologio digitale oppure ancora da un sensore al quale Arduino deve dare immediata attenzione. Le porte 2 e 3 di Arduino sono le uniche abilitate a ricevere i segnali di interrupt ed e’ su queste porte che devono essere collegati i sensori che, all’occorrenza, chiedono attenzione.

Una interessante proprieta’ di Arduino e’ che l’interrupt puo essere recepito ed elaborato anche quando il sistema e’ in stato di “sleep” ed e’ proprio questa sua caratteristica che viene utilizzata in questo esercizio.

Dal punto di vista operativo, la pressione del pulsante collegato alla porta 12 attiva la funzione di sleep, spegne il led verde, fa lampeggiare tre volte il led rosso e “addormenta” Arduino mantenendo acceso il led rosso (il led rosso segnala che Arduino e’ dormiente). A questo punto,

se lo si desidera, si puo' verificare sull'amperometro collegato al polo positivo della batteria, il consumo del sistema in stato di "sonno". In questo esercizio il consumo in stato di sonno (e con il led rosso acceso) e' di 43 milliampere (circa 0,39 watt).

La pressione del pulsante collegato alla porta 2 provoca un interrupt che "sveglia" Arduino. In pratica, subito dopo la pressione, si spegne il led rosso, lampeggia tre volte il led verde ed Arduino si sveglia, proseguendo nella sua normale operativita' e mantenendo acceso il led verde (il led verde segnala che Arduino sta operando normalmente). Ora e' possibile verificare nuovamente sull'amperometro il consumo di energia e calcolare quindi il risparmio indotto dalla funzione sleep. In questo esercizio Arduino, sveglio e con un led acceso, assorbe 57 milliampere (circa 0,51 watt).

Osservando le escursioni dell'amperometro durante il lampeggio del led rosso (o verde) si evince che un led assorbe circa 12 milliampere. Eliminando il led rosso ed il led verde (e quindi considerando il solo assorbimento di Arduino), e possibile affermare che il sistema, in stato di sleep e privo di attuatori o sensori che assorbono energia, riduce il consumo di circa il 30%.

Il filmato di questo esercizio e' disponibile qui (click).

Nota: Questo esercizio e questa nota sono parte di una serie che vede protagonisti arduino ed alcuni dei componenti ad esso collegabili. Per la maggior parte degli esercizi e' anche disponibile un filmato su youtube.

- Esercizi facenti parte della raccolta
- Filmati presenti su youtube
- Informazioni su arduino e sui componenti collegabili (PDF scaricato nell'area di download)
- Breve manuale di programmazione (PDF scaricato nell'area di download)

Per eventuali chiarimenti o suggerimenti sul contenuto di questa scheda scrivere a giocarduino@libero.it

Here some notes about this project, translated by google translator 

In this project we explore two seemingly secondary functions: ***sleep*** and ***interrupts***.

Sleep is an instruction, or rather a function that allows you to "numb" the microcontroller and thus save energy. Disabling the microcontroller interrupts the running program but still leaves operating the various components, and so even if Arduino is sleeping the output pins declared "HIGH" will continue to provide a voltage of 5 volts and power pins (5V and 3.3V) will continue to feed the connected sensors and actuators.

There are different degrees of "sleep". In this example we used the SLEEP_MODE_PWR_DOWN, the "deep sleep" that enables the greatest energy savings. When Arduino is in "deep sleep" can be awakened only by an external interrupt on pin 2 or 3.

"Sleep" can seem a useless feature, but if you want to build system that operate, even for months, using only the power of a battery, this feature, combined with other measures, it can become indispensable.

Interrupts is a signal given by a sensor and immediately understood and processed by Arduino. It is an asynchronous signal with respect to the running program. The verification instructions for pin 2 and 3 are not present in the loop section but are in an specific routine that, activated by signal on pin 2 or 3, immediately becomes a priority. In practice, when Arduino receives an "interrupts", interrupts the running program, launches a routine that is performed only in case of "interrupts" and, at the end, resumes execution of the program from the point where it was interrupted. It's a bit

as a phone call: when we receive it, we interrupt the work we are doing, we pay attention to what we are told on the phone and at the end, we resume the interrupted job.

Interrupts may be launched by an internal or external event and then, for example, by the pressure of a button (as in this project) or by an alarm launched from a digital clock or again by a sensor to which Arduino must give immediate attention. Pins 2 and 3 are the only authorized to receive interrupts signals and is on these pins that must be connected sensors which, if any, demand attention.

An interesting property is that Arduino can receive and process “interrupts” also when is in an a sleep state, and this characteristic is used in this exercise.

From an operational standpoint, pressing the button connected to the port 12, activates the sleep function, the green LED turns off, flashes three times the red LED and Arduino goes to sleep, keeping lit the red light (the red LED indicates that Arduino is asleep). Now, if you wish, you can check, on the ammeter connected to the battery positive, the system consumption in "sleep"state. In this example, the consumption in the sleep state (red LED on) is 43 milliamperes (about 0.39 watts).

The button connected to pin 2 causes an interrupt that "wakes" Arduino. In practice, if pressed, turns off the red LED, the green LED flashes three times and Arduino wakes up and continues its normal operability, keeping green LED active (green LED indicates that Arduino is operating normally). Now is possible check again energy consumption on the ammeter and then calculate the savings induced by "sleep" function. In this example, Arduino, awake and with a led on, absorbs 57 milliamperes (about 0.51 watts).

Note: This project and this note is part of a series that sees, as main characters, Arduino and some of connectable components. For most projects there is also a video on youtube.

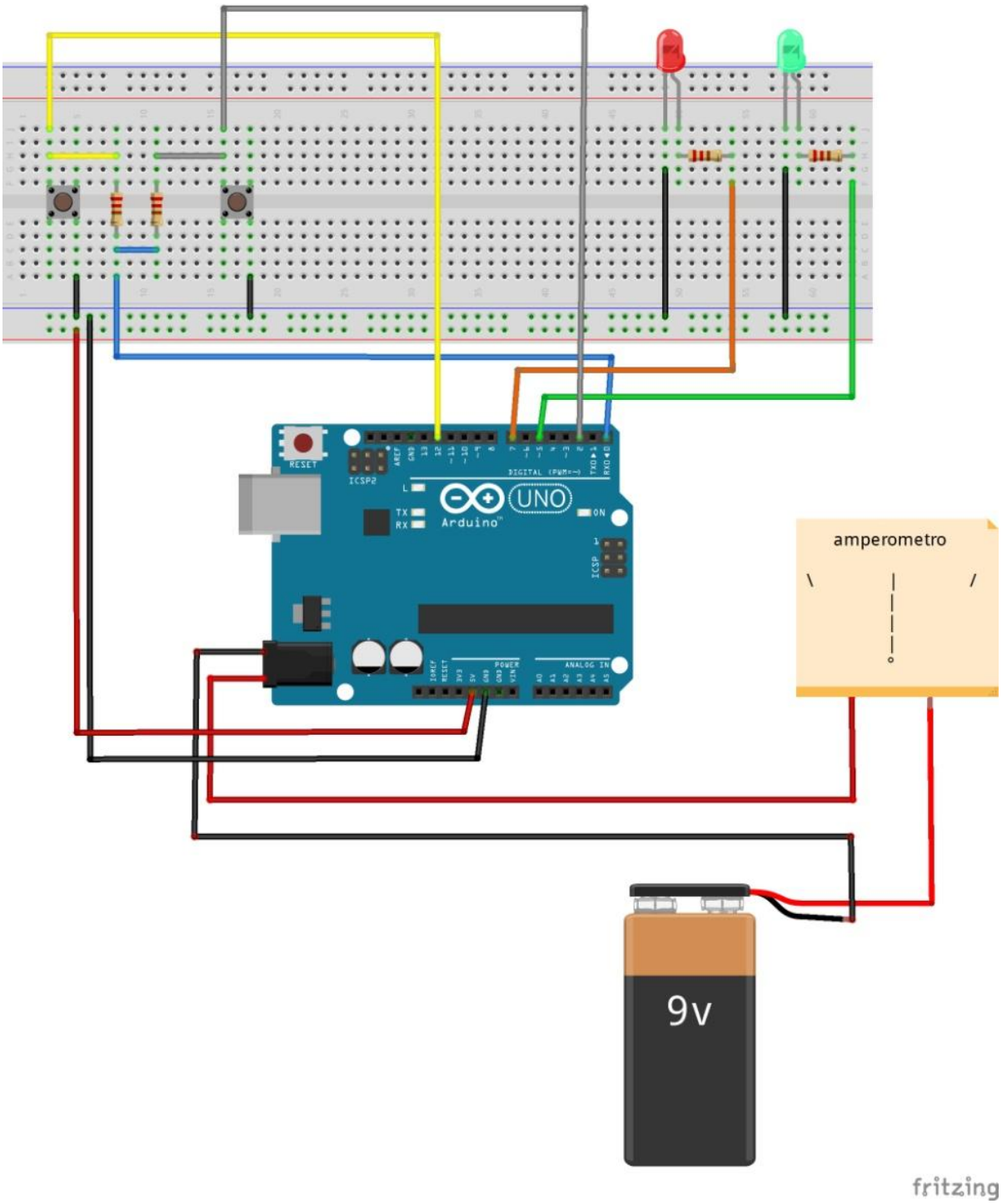
- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download area
- Quick programming guide (almost english; pdf will be downloaded in your download area)

For any questions or suggestions about this note (and on its english translation), please write to giocarduino@libero.it (simple words and short sentences, please)

Materiali

- un led verde
- un led rosso
- due pulsanti
- quattro resistenze da 220 ohm da collegare ai led ed ai pulsanti
- un amperometro, se si desidera misurare la differenza di assorbimento
- un po’ di cavetteria

Schema



Nota: il collegamento tra la porta 0 (RX) e le resistenze da 220 ohm a loro volta collegate ai pulsanti ed ai pin 2 e 12 serve a mantenere HIGH lo stato delle porte 2 e 12 (sia con Arduino dormiente che con Arduino operativo). La pressione del pulsante collega le porte direttamente a terra (GND), provocandone il tempraneo cambiamento di stato, da HIGH a LOW

Note: link between pin 0 (RX) and 220ohm resistance in turn connected to buttons and to pins 2 and 12 maintains both pins in HIGH status. Pressing button, connects pin directly to ground (GND), and causes a temporary change of state from HIGH to LOW

Programma

```
/*  Attenzione: facendo il copia/incolla dal PDF all'IDE si perde la formattazione del testo.
 *  Per rendere piu' facilmente leggibile il programma e' opportuno formattarlo subito dopo il
 *  trasferimento nell'IDE, premendo CTRL+T.
 *
 *  Questo programma nasce da un'idea reperita in rete
 *
 *  E' una demo di come si possa attivare la funzione sleep e disattivarla
 *  attraverso un segnale (un interrupt) esterno
 *
 *  Arduino entra in stato di sleep premendo un pulsante collegato alla porta 12
 *  e si risveglia mediante un interrupt provocato dalla pressione di un pulsante
 *  collegato alla porta 2
 *
 *  I led indicano lo stato del sistema:
 *  - led rosso: il sistema e' dormiente
 *  - led verde: il sistema  e' sveglio e sta lavorando
 *
 *
 * ------------------------------------------------------------------------------------------
 *  Attention: cut and paste from PDF to IDE loses formatting. To restore it press CTRL + T
 *
 *  Sleep and wake up on external interrupt demo -
 *  - sleep mode by pushing button on pin 12
 *  - wake by an interrupt generated pushing button on pin 2
 *
 *  system status reported by red and green leds:
 *  - red:   system sleeping
 *  - green: system working
 * ------------------------------------------------------------------------------------------
 */
#include <avr/interrupt.h>
#include <avr/sleep.h>

int portasonno    = 12;     // identificativo della porta 12, normalmente alimentata.
// Provoca lo stato di sleep nel momento in cui un pulsante la collega alla terra (gnd)
// sleep pin (sleep generated by temporary ground connection)
//
int portarisveglio = 2;     // identificativo della porta 2, normalmente alimentata.
// Provoca il risveglio di arduino nel momento in cui un pulsante genera un interrupt collegandola
// a terra (gnd)
// wake pin (wake on interrupt generated by temporary ground connection)
//
int statoporta12    = 0;    // variable per la memorizzazione dello stato della porta 12 - pin 12
status
//
int ledrosso        = 7;    // porta cui e' collegato il led rosso  - red led pin
int ledverde        = 5;    // porta cui e' collegato il led verde  - green led pin
int i               = 0;    // utilizzato nei cicli di for - used in "for" cycle
//
// ***************************************************************************************
// *** routine gestione "addormentamento" e "risveglio" (sleep and wake routine) ****
// ***************************************************************************************
void dormiora()
{
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);   // tipo di "addormentamento" desiderato
  // (SLEEP_MODE_PWR_DOWN e' il piu' profondo, quello che provoca il maggior risparmio di energia)
  //
  //  Serial.println ("mi addormento");
  digitalWrite (ledverde, LOW);
  for (i = 0; i < 3; i++)  // segnala l'"addormentamento" facendo lampeggiare il led rosso
  // sleeping now signal, by blinking red pin
  //
 {
    digitalWrite (ledrosso, HIGH);
    delay (500);
    digitalWrite (ledrosso, LOW);
    delay (500);
  }
  digitalWrite (ledrosso, HIGH); // il led rosso segnala sistema dormiente
  // red light for sleeping system
  //
  //  Serial.println ("dormo");
  delay (500);
  sleep_enable();  // abilita l'attivazione della modalita' sleep
  // Enable sleep bit in the mcucr register
  //
```

Arduino: sleep, wake ed interrupts

```
  attachInterrupt(0, svegliatiora, LOW);  // abilita la ricezione di un interrupt di tipo 0.
  // L'interrupt di tipo 0 e' l'interrupt generato da uno stato LOW sulla porta 2. Nel momento in
  // cui viene ricevuto un interrupt viene automaticamente lanciata la routine "svegliatiora")
  // Use interrupt #0 (pin 2) and run "svegliatiora" function on interrupt
  //
  sleep_mode();  //attivazione della funzione sleep. Da ora il  sistema e' in sleep e puo' esere
  // svegliato solo da un interrupt sulla porta 2- Here the device is  actually put to sleep!!
  //
  // ********************************************************************************************
  // questo e' il punto in cui il sistema riprende a lavorare subito dopo il risveglio e subito
  // dopo aver eseguito la routine "svegliatiora"
  // system start work here, after wake and "svegliatiora" routine
  //*********************************************************************************************
  //
  digitalWrite (ledrosso, LOW);
//  Serial.println ("mi sto svegliando");
  for (i = 0; i < 3; i++) // segnala il risveglio facendo lampeggiare il led verde - Wake signal, by
blinking green pin
 {
    digitalWrite (ledverde, HIGH);
    delay (500);
    digitalWrite (ledverde, LOW);
    delay (500);
  }
  digitalWrite (ledverde, HIGH); // il led verde segnala sistema attivo green light for working
system
  delay (500);
}
// ********************************************************************************************
// ********  routine lanciata automaticamente al momento della ricezione dell'interrupt ********
// **********                  here the interrupt is handled after wakeup             ***********
// ********************************************************************************************
void svegliatiora()
{
  sleep_disable();                      // disabilita la funzione sleep
  detachInterrupt(0);                   // disabilita l'interrupt
  // le due precedenti istruzioni devono essere le prime ad essere eseguite al momento del
  // risveglio, in modo da evitare eventuali ulteriori interrupt derivanti dalla pressione
  // prolungata del pulsante - These above two instruction must be execute at first after wake
}
//
//
void setup()
{
//  Serial.begin (9600);
  pinMode(portasonno, INPUT);
  pinMode(portarisveglio, INPUT);
  pinMode(ledrosso, OUTPUT);
  pinMode(ledverde, OUTPUT);
  digitalWrite (ledverde, HIGH);
}
//
//
void loop()
{
//  Serial.println ("Sono sveglio");
//  delay (1000);
  statoporta12 = digitalRead(portasonno);  // rileva lo stato della porta 12- read sleep pin
//  Serial.print ("porta 12 = ");
//  Serial.println (statoporta12);
  if (statoporta12 == LOW) // se la porta 12 e' collegata a terra (se c'e un segnale LOW)- if LOW
signal on pin 12
  {
    dormiora();  // mette il sistema in stato di sleep - call sleep routine
  }
}
```