UNITÀ 2: INPUT/OUTPUT DIGITALI E INTERRUPT

OBIETTIVI

Nella presente unità non intendiamo parlare di cosa siano gli input e output digitali. Intendiamo, invece, chiarire alcune idee sul controllo dei dispositivi digitali e spiegare quali siano le resistenze pull-up and pull-down.

Approfondiremo inoltre gli interrupt: Arduino è in grado di sospendere un programma in corso di esecuzione e di eseguire un compito diverso. Una volta che quest'ultimo è stato eseguito, Arduino torna al programma originale; tutto questo può succedere quando i pin di input rilevano segnali particolari. Approfondiamo di seguito

SESSIONE TEORICA

- OUTPUT DIGITALI; LIVELLI ATTIVI HIGH LOGIC
- INPUT DIGITALI; RESISTENZE PULL-UP e PULL-DOWN
- INTERRUPT

SESSIONE PRATICA

- ESEMPIO 1: LED a illuminazione
- ESEMPIO 2: Monitoraggio degli input
- ESEMPIO 3: Monitoraggio degli input con resistenze pull-up
- ESEMPIO 4: Monitoraggio degli input senza interrupt
- ESEMPIO 5: Monitoraggio degli input con interrupt
- ESEMPIO 6: Controllo di due interrupt

MATERIALI

-Laptop o computer

-Scheda Arduino IDE, inclusi i materiali supplementari già installati e configurati

-Scheda di controllo Arduino UNO

-Cavo USB

SOMMARIO

SESSIONE TEORICA

1. OUTPUT DIGITALI E LIVELLI LOGICI

Non è difficile capire quale sia il contenuto di questa sezione. Sottintendiamo che tu abbia già familiarità con le funzioni relative alla gestione delle uscite digitali. Approfondiamo in ogni caso:

- **pinMode(n,OUTPUT)**: permette di configurare qualsiasi pin D0:D13 come output digitale.
- **digitalWrite(n,LOW)**: invia un livello logico di "0" da 0 V attraverso il pin *n* indicato.
- **digitalWrite(n,HIGH)**: invia un livello logico di "1" da +5 V attraverso il pin *n* indicato.

Ora immaginiamo di voler controllare l'illuminazione di un LED collegato al pin digitale D4. Sarebbe necessario digitare una sequenza di funzioni come questa:

```
pinMode(4,OUTPUT);          //Output pin D4

digitalWrite(4,HIGH);       //inviare livello "1" attraverso il PIN D4
```

Perché si parla di impostare un livello su "1" (+5 V) ogni volta che vogliamo accendere qualcosa? Utilizziamo il livello "0" (0 V) solo per spegnere qualcosa? Non possiamo farlo in senso opposto? Beh, possiamo! I livelli "1" e "0" sono ugualmente rappresentativi. Date un'occhiata ai diagrammi nella figura seguente. Che tipo di livello logico dovremmo inviare attraverso il segnale D4 (pin 7) per accendere il LED nel circuito a sinistra? E per il LED a destra?

Figura 1

Tenendo presente che l'anodo deve sempre essere positivo rispetto al catodo, per accendere un LED dobbiamo inviare un "1" (+5 V) attraverso D4 (pin 7) all'anodo nel circuito a sinistra. Questo perché il catodo è collegato a GND (0 V) attraverso la resistenza di assorbimento. Tuttavia, bisogna inviare uno "0" (0 V) attraverso D4 al catodo nel circuito a destra mentre l'anodo è collegato a +5 V attraverso la resistenza.

In questo caso si parla di un LED, ma lo stesso vale per altri dispositivi come bobine di relè, motori, buzzer, ecc. Esaminando i diagrammi nella figura seguente si può notare che la direzione di rotazione del motore cambia quando lo si accende con il livello "1" anziché con il livello "0".

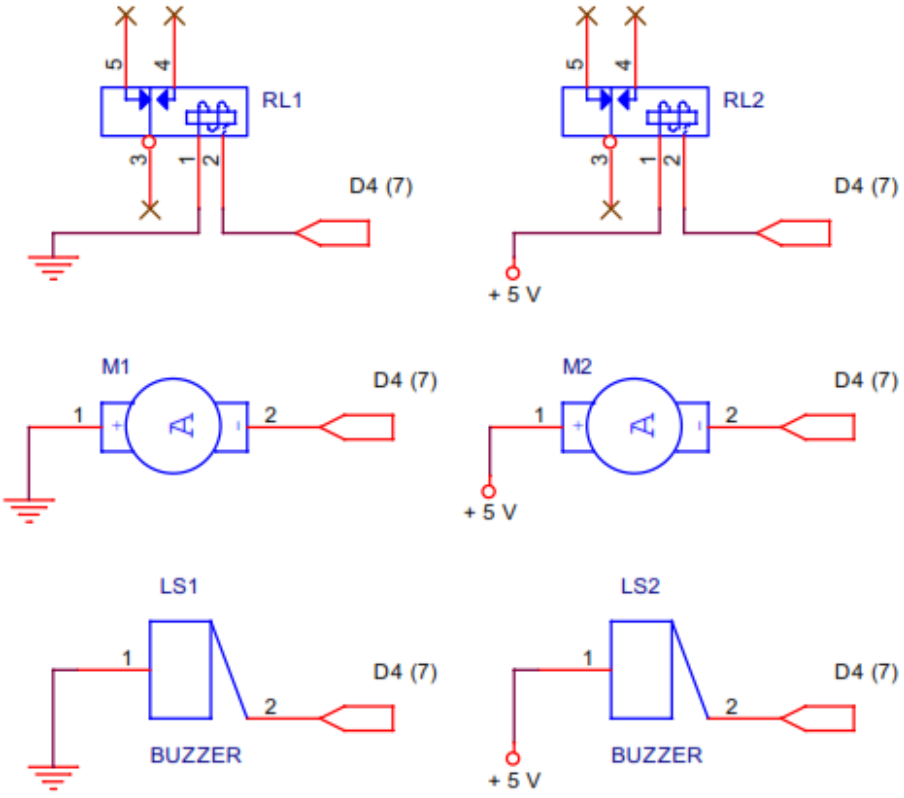


Figura 2

Riassumendo: ci sono periferiche di output che si accendono quando si applica il livello "1" e vengono chiamate "segnali attivi alti". Ci sono altre periferiche di output che si accendono quando si applica il livello "0". Si possono incontrare entrambe, ciò dipende da dal modo in cui la periferica viene assemblata e connessa.

2. INPUT DIGITALI; RESISTENZE PULL-UP AND PULL-DOWN

Puoi pensare di sapere tutto sugli input digitali, ma esamineremo a breve un problema che è possibile incontrare. Per iniziare è necessario ricordare che le periferiche digitali di input più semplici ed economiche sono i pulsanti e gli interruttori; se ne trovano di tutte le forme e dimensioni; alcuni sono progettati per uso industriale all'interno di macchine utensili, pannelli di controllo ecc.

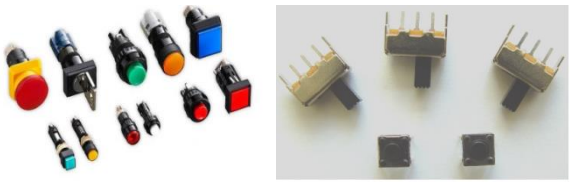


Figura 3

Altri sono molto più semplici (e meno costosi). Sono ampiamente utilizzati in ambiente domestico, in ambito formativo, per imprimere particolari sulle carte stampate, nei piccoli elettrodomestici, ecc... Utilizzeremo alcuni piccoli pulsanti come quelli che puoi vedere nella parte inferiore della foto. In ogni caso, indipendentemente da come siano utilizzati, il modo in cui lavorano è abbastanza semplice. Quando vengono innescati, un foglio di metallo si apre e blocca un contatto elettrico tra due o più terminali nel dispositivo. Si dice che un interrupt o un commutatore è "embedded". Quando lo si accende, l'interruttore rimane dov'è fino a quando non lo si sposta nuovamente. Gli interruttori luminosi hanno un funzionamento simile, ma i pulsanti funzionano diversamente: aprono e chiudono uno o più circuiti solo quando accesi, dopodiché, quando rilasciati, ritornano alla loro posizione di riposo - come il campanello, ad esempio. Nelle seguenti immagini osserviamo il diagramma di un commutatore scorrevole e di un pulsante, con i rispettivi simboli elettrici.

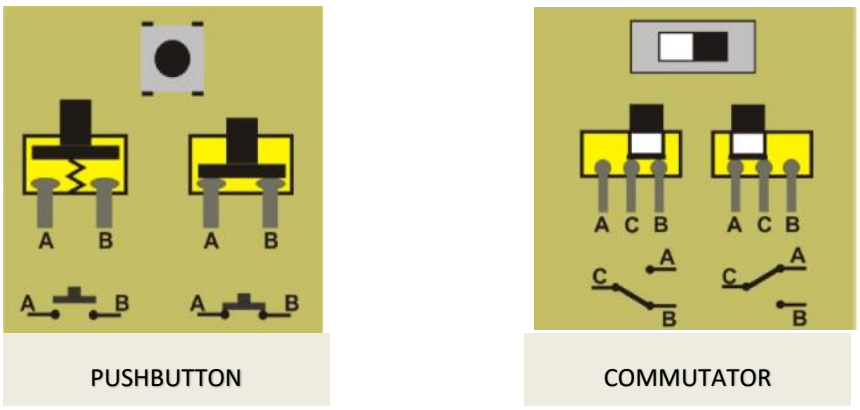


Figura 4

Il commutatore nella figura ha tre pin; uno di questi è in comune con gli altri due - C. Quando lo facciamo scorrere verso destra il foglio chiude il circuito tra i pin C-B. Il circuito C-A rimane aperto, ma non collegato. Se lo spostiamo verso sinistra, il circuito C-A si chiude e il circuito C-B rimane aperto.

pulsante ha due pin, A e B. Assumiamo che in posizione di riposo di solito è aperto e i pin non sono collegati. Quando lo premiamo, si chiude il circuito e entrambi i pin sono collegati. Quando lo rilasciamo una molla spinge il foglio di metallo nella posizione di riposo. Esistono pulsanti che sono solitamente chiusi e aperti solo quando premuti.

Di seguito i due linguaggi Arduino per controllare gli input digitali:

- **pinMode(n,INPUT)**: permette di configurare qualsiasi pin digitale D0:D13 come input. Se si preme RESET ogni volta che si avvia il sistema tutti i pin sono riconosciuti come input predefiniti.
- **digitalRead(n)**: legge I livelli logici di qualsiasi numero di pin.

Nei seguenti due schemi un pulsante è stato collegato a D2 (pin 5) che è apparentemente configurato come ingresso.

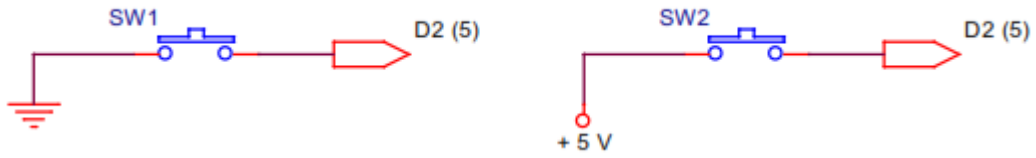


Figura 5

Ogni volta che si preme il pulsante nel circuito a sinistra, il terminale D2 si collega con GND (0 V) ed è quindi a livello "0". In quello sulla destra, viceversa, quando si preme il pulsante si collega con +5 V e quindi è a livello "1".

Quesito: cosa succede ai terminali se non premiamo nessuno dei due pulsanti nei due circuiti? Potremmo pensare "l'opposto di quando li si preme". In altre parole, se non si preme il pulsante nel circuito a sinistra, il terminale D2 rimane a livello "1" e quello a destra a livello "0".

Sbagliato! Se il pulsante non è premuto in nessuno dei due circuiti, qual è la differenza per quanto riguarda il terminale D2? Assolutamente nessuna. Il D2 rimane disconnesso in entrambi i casi. Diciamo che è "galleggiante". Quale livello prevale quindi, livello "1" o livello "0"? Non esiste una risposta definitiva; a volte prevale "1", altre volte "0". Comprensibilmente questa non è una situazione ideale.

Per quanto riguarda l'aspetto elettrico, la cosa migliore da fare è aggiungere una resistenza al circuito. Se è collegata al positivo + 5 V si chiama resistenza o "**PULL-UP**"; Se è collegata a GND o 0 V, si chiama "**PULL-DOWN**". Diamo un'occhiata ai seguenti due circuiti. (Figura 6).

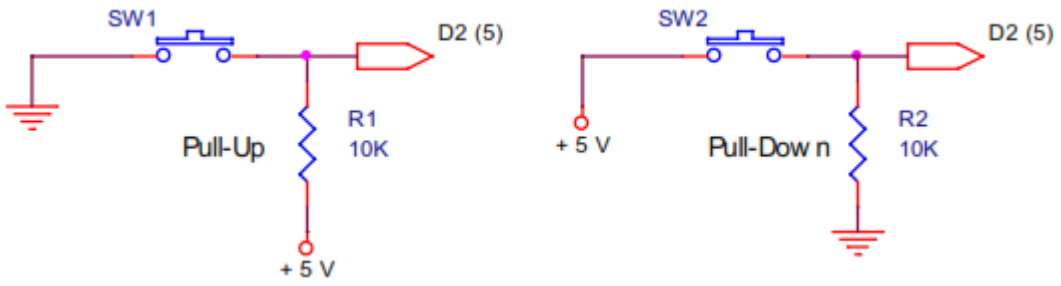


Figura 6

In quello a sinistra, la resistenza pull-up R1 mantiene il segnale D2 (pin 5) a livello "1" (+5 V) quando il pulsante SW1 non è premuto, ossia è aperto. Quando viene premuto il terminale D2, si passa al livello "0". Nel circuito a destra, il la resistenza pull-down R2 tiene abbassato il terminale al livello "0" (GND) quando il pulsante SW2 non è premuto. Quando viene premuto il terminale D2, si passa al livello "1".

Quale valore assegnare dunque alle resistenze? Non è così importante; ogni volta che si preme un pulsante, c'è un flusso di elettroni tra GND e + 5 V, o, in altre parole, la corrente (I) entra nel circuito. Dobbiamo cercare di minimizzare questo flusso di corrente e il suo consumo. Ad esempio, con 100 Ω attraverso R1 o R2, il consumo sarà il seguente:

$$I = \frac{V}{R} = \frac{5}{100} = 0.05\ A = 50\ mA$$

Con 10 KΩ attraverso R1 o R2, il consumo sarà il seguente:

$$I = \frac{V}{R} = \frac{5}{10000} = 0.0005\ A = 0,5\ mA$$

E' in ogni caso importante non esagerare. Potremmo essere tentati di utilizzare una resistenza più potente per ridurre al minimo il consumo, ma quello che accade con un valore troppo elevato è che il terminale D2 rimanga isolato dal +5 V (nel caso del pull-up) o dal GND (nel caso del pull-down) e non resti in posizione. Se ciò accade si torna al punto di avvio, quindi è meglio non utilizzare alcuna resistenza. I valori medi sono compresi tra 4700 e 10 KΩ.

Ricordiamo nuovamente che il livello attivo - quando si preme il pulsante - non deve necessariamente essere "1", ma potrebbe anche essere "0". Tutto dipende da come si collega quel pulsante.

Come ci aiuta Arduino? Ci fornisce già la resistenza pull-up e consente di non doverne installare una esterna. Nella figura 7 la resistenza è stata integrata nel controller Arduino.

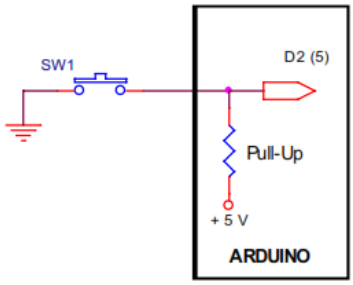


Figura 7

È possibile configurare questa opzione utilizzando la funzione seguente:

- **pinMode (n, INPUT_PULLUP)**: n rappresenta il pin di input che si intende collegare alla resistenza pull-up corrispondente.

È possibile applicare tale funzione a qualsiasi pin D0:D13 che verrà configurato come input. Non si applica invece agli output, così come non si ha la possibilità di configurare un input come resistenza pull-down.

Potremmo pensare che non sia importante risparmiare su una resistenza di input, poiché implica costi irrisori. Da un punto di vista aziendale, invece, se un progetto necessita di una serie di resistenze pull-up, ricordiamo che stiamo risparmiando anche sui seguenti aspetti:

- spazio, perché non li terremo in magazzino;
- dimensione delle schede stampate. Sono vendute al decimetro quadrato e più componenti ci sono, maggiore è la dimensione della scheda;
- tempo sul disegno dei binari per le schede stampate: più componenti utilizziamo, più tempo necessitiamo per progettarli;
- tempo di assemblaggio del circuito: più componenti utilizziamo, più tempo impiegheremo per assemblarli.

3. INTERRUPT

Ora passiamo ad un altro argomento: gli interrupt. Per comprenderne le caratteristiche immaginiamo una macchina utensile che stia realizzando dei particolari. Improvvisamente l'allarme si attiva e la macchina va in blocco per uno dei motivi seguenti:

- Un particolare è posizionato nel modo errato, da cui risulterà un prodotto difettoso;
- Il motore della cinghia del convogliatore è rotto e i particolari si accumulano;
- Un sensore rileva che un componente della macchina si sta surriscaldando;
- Un lavoratore inserisce la propria mano nella macchina quando non dovrebbe.

Che cosa dovrebbe fare il controller per interrompere questo malfunzionamento?

A) Niente; continuare ad eseguire il programma come se nulla fosse accaduto;
B) Arrestare la macchina utensile e bloccare l'esecuzione del programma;
C) smettere di eseguire il programma precedente ed eseguirne un altro che agisca sulle cause che attivano l'allarme svolgendo le attività necessarie: rimuovere la parte difettosa, fermare la produzione di particolari, attivare un sistema di raffreddamento, emettere un segnale di avviso, ecc ...

La risposta è piuttosto intuitiva. Un interrupt non prevede necessariamente un blocco immediato, bensì prevede l'interruzione di un compito che si sta eseguendo in quel momento per eseguirne un altro.

Come attiviamo un interrupt? Ci sono numerosi eventi che possono innescarne uno, tutto dipende dal modello del controller. Il modo più comune per le periferiche esterne è quello di inviare segnali attraverso pin di controllo. Arduino UNO ha due fonti o pin di interrupt diversi: D2 / INT0 e D3 / INT1.

In realtà si tratta di pin di input/output digitali come tutti gli altri D0:D13, ma per funzionare come ingressi interrupt INT0 e INT1 devono essere configurati nel modo giusto e dotati di una speciale autorizzazione.

Cosa succede quando il controller riceve un interrupt? La figura mostra come funziona il processo:

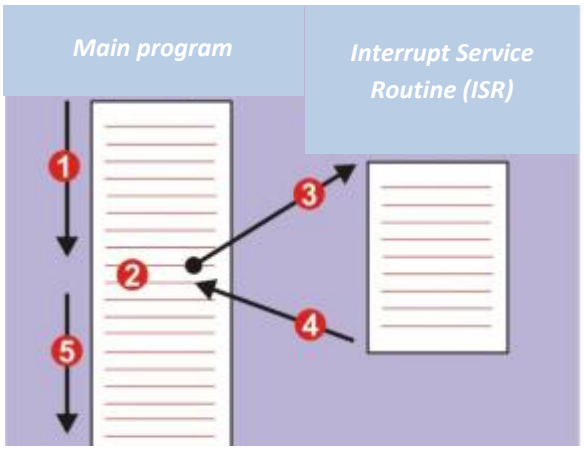


Figura 8

1. Il controller sta eseguendo il programma principale;

2. A un certo punto una periferica richiede e provoca un interrupt;

3. Il controller sospende il programma principale e continua ad eseguire un programma per gestire l'evento, noto anche come ISR (Interrupt Service Routine);

4. terminata l'esecuzione dell'interrupt il controller torna al programma principale;

5. L'esecuzione viene ripresa da dove è stata interrotta.

Per gestire e utilizzare gli interrupt Arduino UNO ha a disposizione quattro funzioni:

- **Funzione attachInterrupt()**

configura come un interrupt dovrebbe funzionare

Sintassi:

attachInterrupt(pin, ISR, mode);

pin: rappresenta il pin che verrà configurato come un input di interrupt. Nel caso di Arduino UNO potrebbe essere INT0 (D2) o INT1 (D3).

ISR: rappresenta l'azione di routine o la funzione che deve essere eseguita ogni volta che si verifica l'interrupt.

mode: quando l'interrupt dovrebbe essere attivato:

LOW: quando il pin invia un livello "0";

CHANGE: quando viene rilevato un cambiamento di stato nel pin;

RISING: quando il pin rileva un incremento ("0" -> "1");

FALLING: quando il pin rileva un incremento ("1"-> "0").

- **Funzione detachInterrupt()**

termina un interrupt.

Sintassi:

detachInterrupt(pin);

pin: rappresenta il pin di interrupt che deve essere spento. Nel caso di Arduino UNO questo potrebbe essere INT0 (D2) o INT1 (D3).

- **Funzione interrupts()**

avvia gli interrupt. Si può considerare come una sorta di autorizzazione generale per gli interrupt che sono stati precedentemente configurati tramite attachInterrupt ().

Sintassi:

interrupts();

- **Funzione noInterrupts()**

disattiva tutti gli interrupt. Si può considerare come divieto generale, che impedisce a tutti gli interrupt di agire.

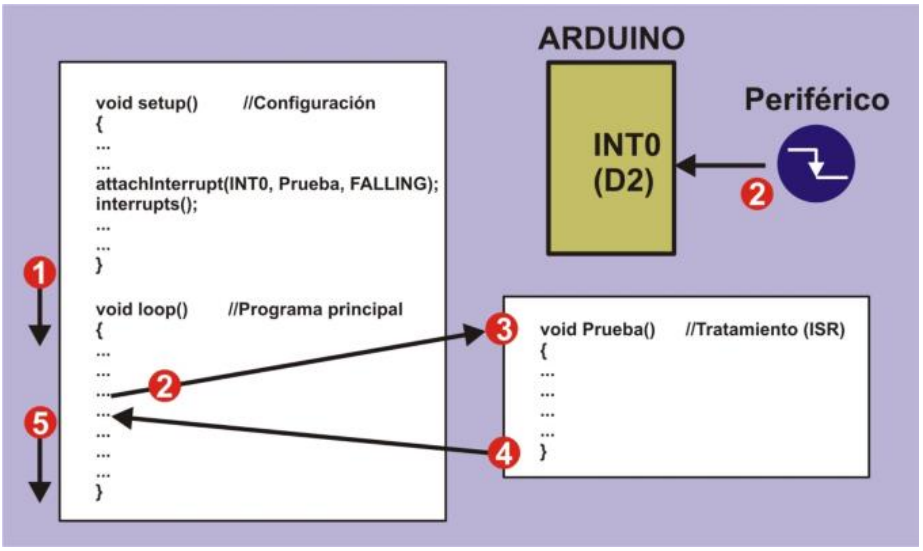


Figura 9

Sintassi:

noInterrupts();

La seguente figura ci dà un'idea di come funzioni un programma che utilizza gli interrupt.

Ogni qualvolta venga rilevato un decremento, il pin D2 (INT0) configura l'interrupt attraverso la funzione **setup**() utilizzando il comando **attachInterrupt** (). A questo punto, verrà eseguita la funzione **Test**() che corrisponde all'ISR. Quasi contemporaneamente **interrupts**() rilascia l'autorizzazione generale. Ecco cosa succede:

1. Inizia ad eseguire il programma principale **loop** ();

2. A un certo punto una periferica emette una richiesta di interrupt dovuta a un decremento attraverso il pin INT0 (D2). Il controller smette di eseguire il programma;

3. Il controller esegue la funzione **Test ()** (Interrupt Service Routine);

4. Una volta eseguita la funzione **Test ()**, il controller torna al programma principale;

5. Il controller riprende l'esecuzione del programma da dove è stato interrotto.

Limiti

A causa della natura stessa di Arduino, il suo sistema di interrupt è limitato. Un esempio è che le funzioni molto conosciute delay() e millis() non funzionano se sono già nella interrupt service routine. Questo accade perché tali funzioni usano il sistema di interrupt interno e Arduino non può rispondere ad un interrupt da un altro sistema.

Non è possibile trasferire i parametri a un interrupt service routine e l'ISR non può, a sua volta, restituire dati. Se ne avessimo necessità, dovremmo utilizzare variabili globali per trasferire i dati da utilizzare dal programma principale e dalla funzione interrupt service routine.

Un interrupt è un evento che può accadere come non accadere, così come una situazione di allarme può verificarsi o meno e una periferica può inviare il segnale di interruzione o meno. È una buona idea eseguire la funzione di interrupt service routine nel più breve tempo possibile. Durante l'esecuzione della funzione ISR, il controller sospende l'esecuzione del programma principale.

Vantaggi

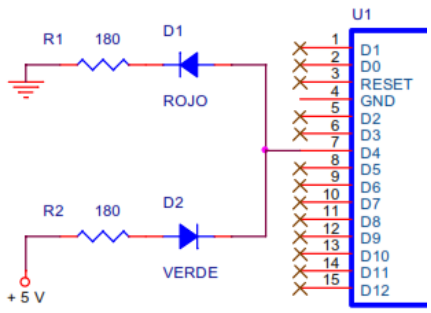
In generale, un programma che utilizza gli interrupt è molto più efficiente, ma è necessario stare attenti a come si usano. Immaginiamo di svolgere un progetto in cui è necessario eseguire un certo compito ogni volta che la periferica invia un segnale. Abbiamo due possibilità: la prima è tramite la funzione digitalRead (), che legge e attende che il segnale sia attivato (mentre è in attesa non è possibile fare altro).

Il programma principale svolge le sue funzioni, quando il segnale lo raggiunge il controllore "risponde" all'interrupt e esegue l'attività corrispondente. Non è perciò previsto tempo di inattività, come se stessimo svolgendo più compiti contemporaneamente.

SESSIONE PRATICA

4. ESEMPIO: LED A ILLUMINAZIONE

Ciò che presentiamo di seguito è come eseguire l'installazione elettrica per far sì che un LED o altre periferiche possano essere accese dopo aver ricevuto un livello logico "0" o un livello logico "1". Osserviamo il diagramma in figura.

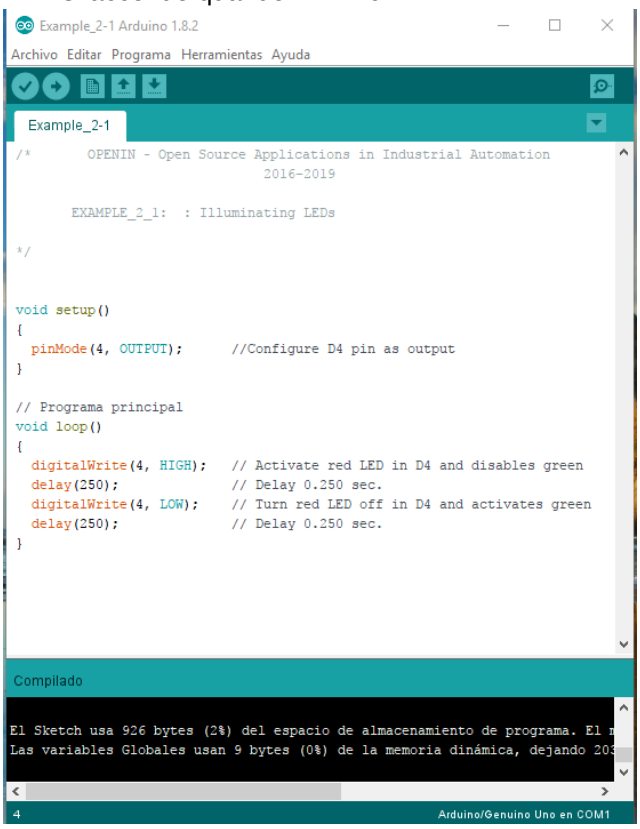


Lo stesso pin di output D4 agisce su entrambi i LED, ma è l'anodo del LED rosso con il catodo che va a GND ad essere collegato. Pertanto questo LED sì accenderà quando D4 = "1". D'altra parte, D4 è collegato al catodo del LED verde con l'anodo che va a +5 V; Questo LED si accende quando D4 = "0"

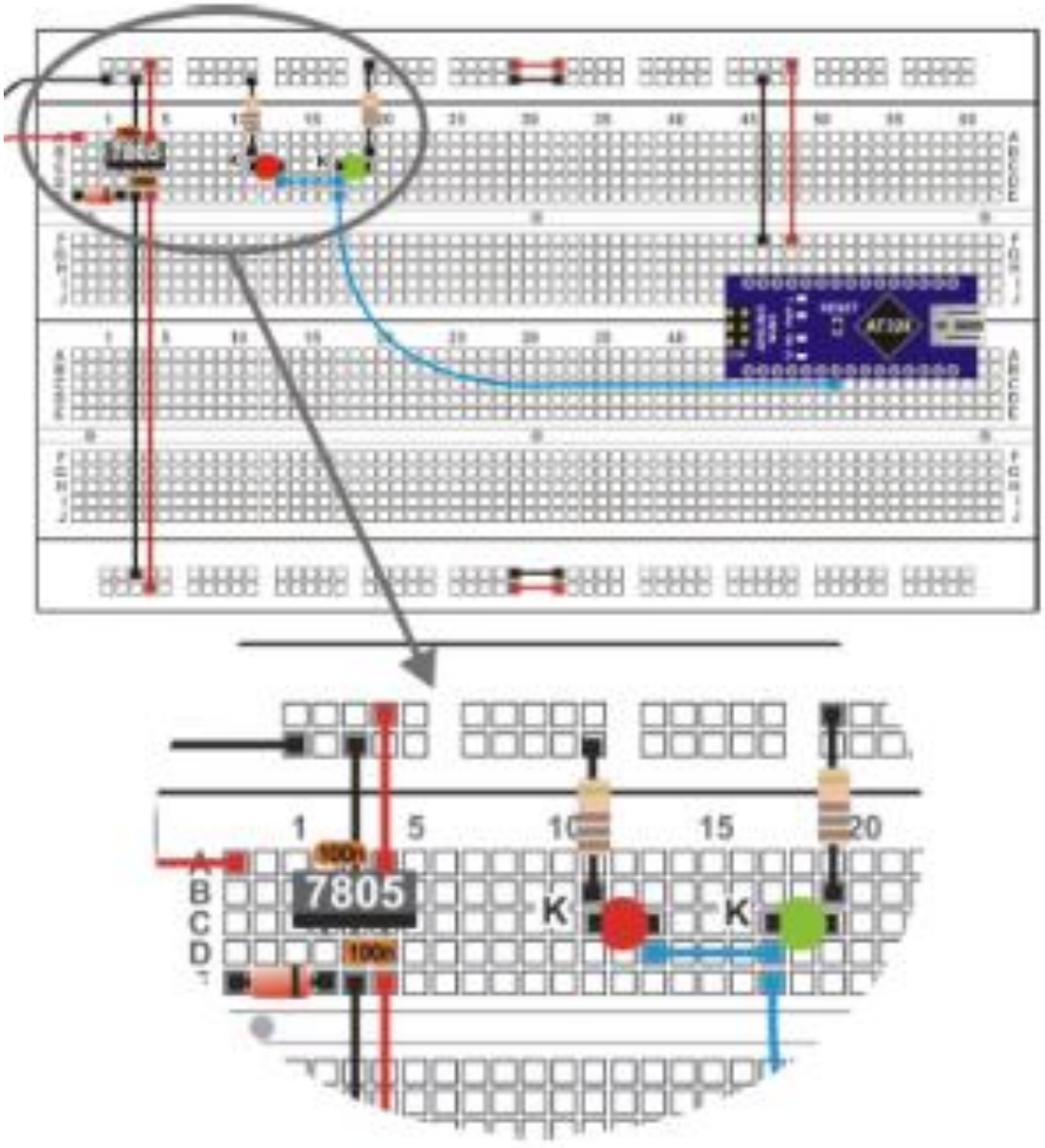
Il programma è molto semplice. Il pin D4 è configurato come output nella funzione setup ().

Il programma principale è un ciclo continuo. Il pin D4 è al livello "1". Il LED rosso si accende mentre il verde si spegne.

Scandito a 0,25 secondi, il perno D4 è impostato a livello "0". Il LED rosso si spegne e il LED verde si accende. Dopo 0,25 secondi, il ciclo viene ripetuto.



Ecco un'idea di come appare l'assemblaggio tecnico sulla scheda. Prestare attenzione alla disposizione e al posizionamento dei componenti.



Adesso tocca a te

Il programma è molto semplice, ma è possibile cambiare il valore della resistenza di assorbimento LED verde LED R2. Il suo valore è di 180 Ω, quindi possiamo portarlo a 10 KΩ.

Cosa si nota?

-

Perchè?

-

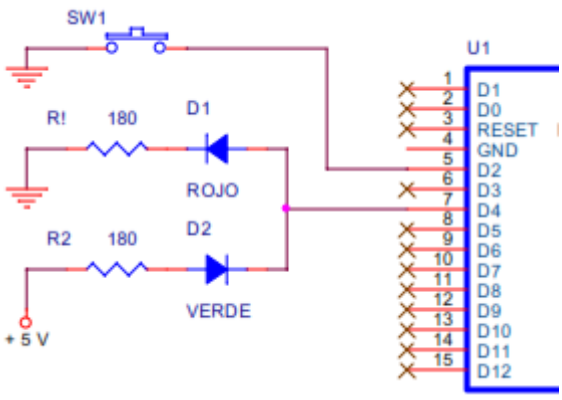
Proviamo a calcolare quanta corrente stia attraversando il LED verde in questo momento:

$$I = \frac{V}{R} = \frac{V - V_{AK}}{R} = \frac{5 - 1.5}{10000} = 0.00035\ A = 0,35\ mA$$

Notiamo ora che la corrente (I) che sta attraversando il LED è 0.35 mA, mentre il produttore suggerisce che dovrebbe essere di circa 20 mA. Abbiamo quindi sperimentato un altro modo per regolare la luminosità di un LED luminoso. Portiamo nuovamente la resistenza R2 al suo valore originale di 180 Ω.

5. ESEMPIO 2: MONITORAGGIO DEGLI INPUT

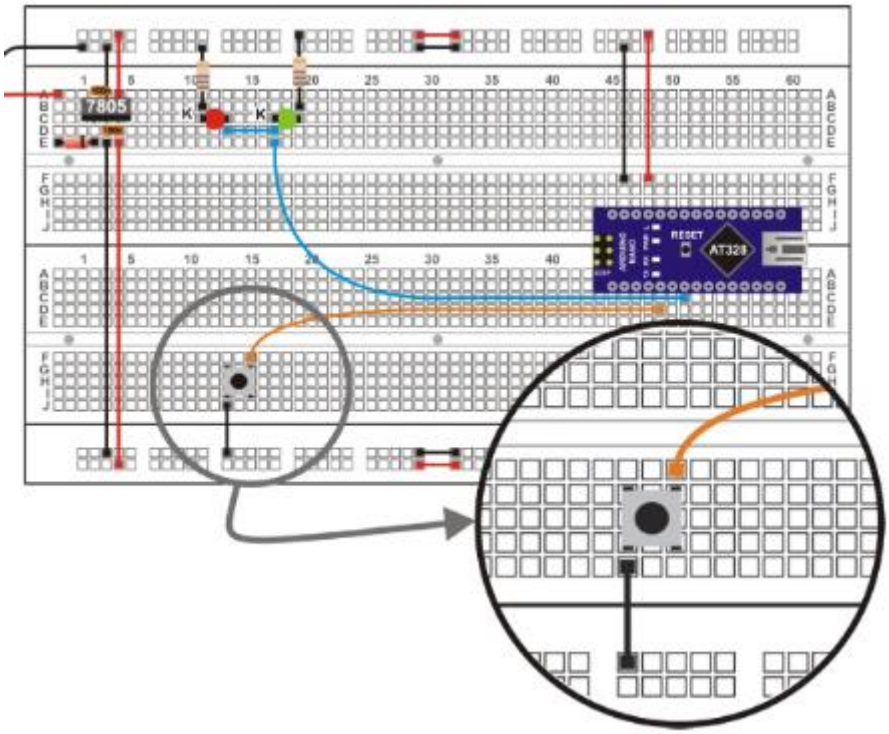
Il diagramma di seguito è esattamente lo stesso dell'esempio precedente tranne un pulsante SW1 collegato a D2.



Quello che dobbiamo fare è leggere il livello logico del pin d'ingresso D2 e applicarlo ai LED rosso e verde. Se l'ingresso è su "1" il rosso si accende e se è su "0" il verde si accende. Entrambi i LED sono controllati dal pin di uscita D4.

Suggeriamo di utilizzare delle pinze piatte per raddrizzare i pin del pulsante come mostrato nella figura e inserire correttamente il pulsante sulla scheda del modulo. I pin dovrebbero essere completamente perpendicolari al corpo del pulsante.

La figura seguente mostra come appare l'assemblaggio. Si dovrebbe gradualmente familiarizzare con il circuito sulla scheda del modulo.



Adesso tocca a te

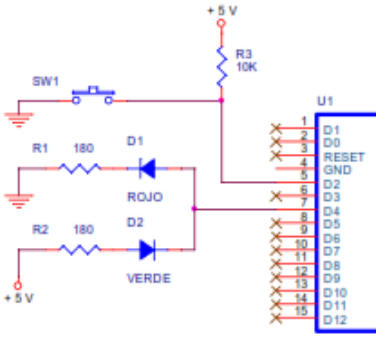
Procediamo ora a registrare l'esercitazione su Arduino UNO e assicuriamoci che funzioni correttamente.

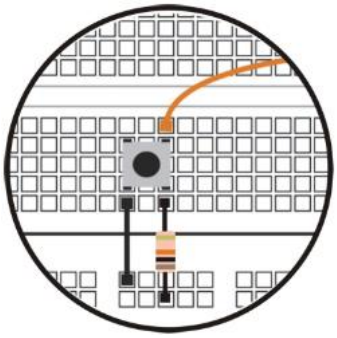
Quando si preme il pulsante si invia un livello "0" attraverso il pin D2 e il LED verde deve accendersi. Ma se non lo premiamo, cosa accade? Inoltre: se si tocca il filo del pulsante o il filo a LED o addirittura i pin del controller Arduino UNO con le dita, cosa succede?

Sembra che i LED si accendano in maniera casuale. Questo perché quando non si preme SW1, l'input D2 è "galleggiante" e non ha un livello logico definito. Osserviamo nuovamente l'esercizio 2.2 nella sezione teorica.

Modifichiamo leggermente il circuito elettrico aggiungendo una resistenza da 10 KΩ come indicato nel diagramma teorico. Si tratta di una resistenza pull-up. Questa resistenza assicura che il pin D2 rimanga ancorato al livello "1" mentre il pulsante non è stato premuto.

A questo punto vedremo che lo stesso esempio_2-2 funziona correttamente senza modificarlo in alcun modo e che i LED si accendano normalmente.





6. ESEMPIO 3: MONITORAGGIO DEGLI INPUT CON RESISTENZE PULL-UP

Questo esempio ci fornisce una soluzione definitiva per le resistenze pull-up. Il pin D2 è configurato come ingresso con una resistenza pull-up, in modo da non doverne aggiungere una esterna. Ora il circuito elettrico appare esattamente come quello del primo diagramma dell'esempio precedente. L'unica differenza è la programmazione.

```
void setup()
{
  pinMode(4, OUTPUT);      //Configure D4 pin as output
  pinMode(2, INPUT_PULLUP);
}
```

Il pin D2 è configurato in modo che funzioni come un input con una resistenza pull-up utilizzando la funzione pinMode (2, INPUT_PULLUP) che è in setup ().

Prima di registrare il programma estrarre la resistenza pull-up da 10 K installata nell'esempio precedente. Non ci può essere più di una resistenza e in questo caso Arduino la fornisce già.

Ora registrare il programma e assicurarsi che tutto funzioni correttamente. I LED collegati all'uscita D4 controllano lo stato logico del pulsante collegato al pin D2. Il rosso si accende quando il pulsante non è stato premuto (livello "1" a causa della resistenza pull-up) e quello verde quando è premuto (livello "0").

Riassumendo i primi tre esempi possiamo trarre le seguenti conclusioni:

• Qualsiasi periferica output può essere attivata utilizzando il livello "1" o il livello "0"; Tutto dipende da come è connesso.

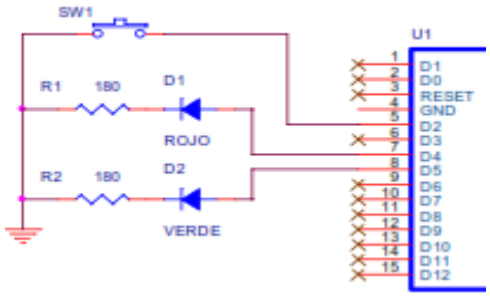
• Allo stesso modo, le periferiche digitali possono generare un livello "1" o un livello "0" quando vengono premute. Dipende da come sono connesse.

• Il livello "0" è ugualmente valido come il livello "1". Sono entrambi rappresentativi.

• Alcune periferiche di input possono includere o non includere una resistenza pull-up. Se non lo fanno, dovremo agire di conseguenza e installarne una esterna oppure usare l'Arduino UNO.

7. ESEMPIO 4: INPUT DI MONITORAGGIO SENZA INTERRUPT

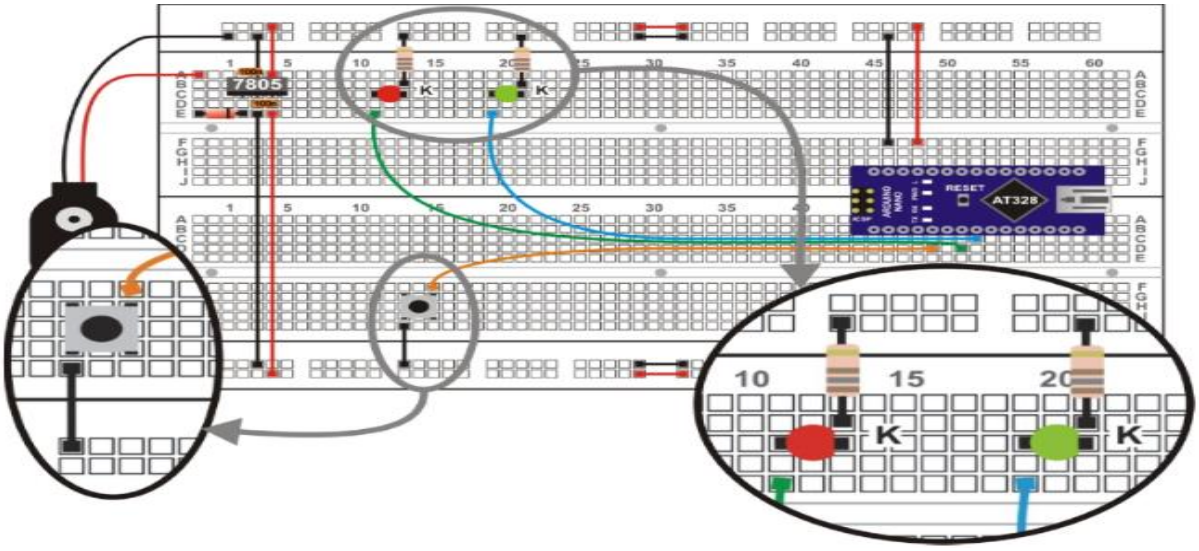
Un LED rosso è stato collegato al pin di output D4 e monitorerà costantemente lo stato del pin di input collegato a D2. L'idea è che l'altro LED, quello verde, collegato all'uscita D5, lampeggi ogni cinque secondi.



La figura mostra lo schema elettrico. Il LED rosso è controllato da D4; monitora lo stato del pulsante collegato a D2. Il LED verde è controllato da D5 e lampeggia ogni cinque secondi. Si noti che entrambi i LED si accendono a livello "1".

Di seguito l'aspetto del circuito sulla scheda del modulo.

Una volta che hai eseguito l'installazione e registrato il programma, assicurati che funzioni correttamente.



Il LED rosso che controlla lo stato del pulsante sembra essere "ritardato". E a volte, se si preme rapidamente il pulsante, il LED rosso "non si avverte nemmeno". Perché?

Mentre il controller sta eseguendo una delle funzioni delay(500), non può eseguire contemporaneamente la funzione digitalWrite (4, digitalRead (2)). In altre parole, non è in grado di tenere il passo con quello che sta accadendo sul pulsante di input D2 e quindi non può monitorare lo stato di D4.

Infatti, il controller monitora ogni volta che un ciclo di accensione e spegnimento venga terminato dall'uscita D5; questo accade ogni secondo. Proviamo a risolvere il problema nell'esempio seguente usando un interrupt.

8. ESEMPIO 5: MONITORAGGIO CON INPUT E INTERRUPT

L'esempio esegue esattamente lo stesso programma dell'ultimo esempio presentato, ma questa volta nel modo corretto. Il LED collegato all'uscita D5 si accende per mezzo secondo e si spegne per mezzo secondo. Allo stesso tempo l'uscita D4 controlla lo stato logico dell'ingresso D2.

Utilizziamo lo stesso schema circuitale e lo stesso circuito di pratica. Cambiamo il programma che dispone di tre sezioni diverse. Osserviamo la seguente figura:

Setup():

I pin D4 e D5 sono configurati come output. D2 è configurato come input con una resistenza pull-up. L'interrupt è configurato utilizzando la funzione attachInterrupt (). Quest'ultimo è associato al pin INT0 (D2); si accende quando viene rilevato un cambiamento di stato nel pin e ogni volta che ciò accade, viene eseguita la funzione Treatment_0 (). La funzione interrupt () sospende e autorizza gli interrupt, che in questo caso sono INT0.

loop():

è il programma principale. Provoca l'intermittenza nell'output D5. Il LED si accende per mezzo secondo e si spegne per mezzo secondo.

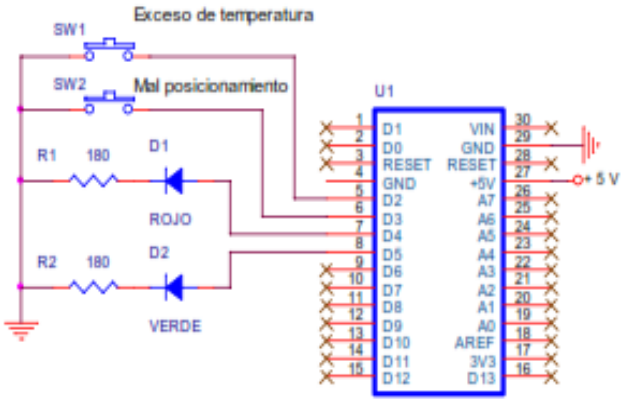
Treatment_0():

Questo è il programma che tratta la interrupt service routine o ISR. Ogni volta che viene rilevato un cambiamento di stato nel pin INT0 (D2), il controller interrompe ciò che sta facendo e esegue la funzione Treatment_0 (). Questa funzione legge lo stato dell'input D2 e controlla l'output D4. Una volta terminato torna al programma principale e riprende.

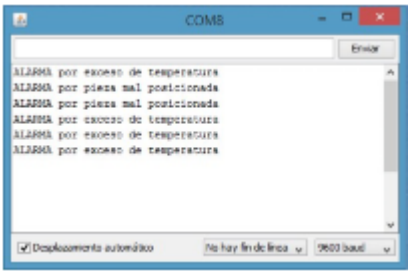
Ora, quando registriamo il programma, saremo in grado di verificare che il controller monitora istantaneamente il LED rosso registrando lo stato di input. Nonostante ciò, il LED verde continua a lampeggiare come se nulla fosse accaduto. Ricordiamo che il programma treatment- quello che controlla lo stato di input - viene eseguito in pochi microsecondi e non ha alcun effetto apprezzabile sul programma principale.

9. ESEMPIO 6: CONTROLLO DI DUE INTERRUPT

Terminiamo l'unità 2 con un altro esempio che utilizza i due interrupt disponibili in Arduino UNO: INT0 (D2) e INT1 (D3). Immaginiamo una macchina utensile che sia responsabile del controllo dei due output, D4 e D5, che seguono una sequenza predefinita.



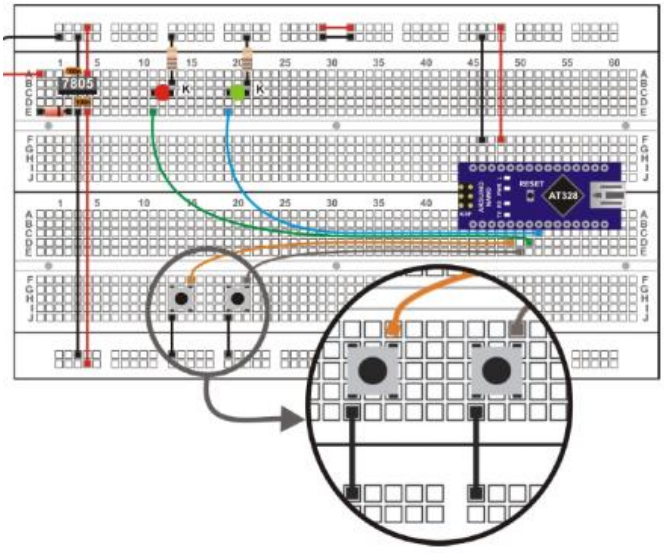
Due sensori rilevano due possibili situazioni di allarme: surriscaldamento e un particolare mal posizionato. I sensori, coordinati dai pulsanti SW1 e SW2, sono collegati ai pin D2 e D3; innescano rispettivamente gli interrupt INT0 e INT1. Diamo un'occhiata allo schema elettrico della figura.



Il programma attiva e disattiva in sequenza i LED rosso e verde. Quando uno degli interrupt viene rilevato, il programma di trattamento appropriato trasmette un messaggio di avviso seriale come quello che vediamo nell'immagine a fianco.

Entrambi gli interrupt sono configurati per essere attivi. Ciò avviene ogni volta che si preme il pulsante appropriato. Ricordiamo che i pin D2 (INT0) e D3 (INT1) sono configurati come input con resistenze pull-up. Quando i pulsanti non sono abbassati, i perni sono accesi "1".

Qui possiamo vedere l'assemblaggio. È stato aggiunto il pulsante SW2 collegato al D3 (INT1).



Dopo aver registrato il programma verificare che funzioni correttamente. Aprire il monitor seriale. Il programma principale si limita ad accendere e spegnere i LED rosso e verde senza che i pulsanti siano premuti. Volendo è possibile modificare tale impostazione.

Se si preme uno dei pulsanti di controllo, viene visualizzato il messaggio appropriato nella finestra del monitor seriale.

Abbiamo visto che di tanto in tanto quando viene attivato uno dei due interrupt, il sistema si chiude. È necessario riavviarlo premendo RESET. Ciò accade a causa dell'effetto "knock on" dei due pulsanti. Ecco ciò che accade nello specifico: anche se si preme solo una volta, può essere letto come diversi interrupt consecutivi e Arduino non è in grado di gestirli correttamente. Abbiamo anche visto in laboratorio che se l'interrupt o i segnali di allarme provengono da un generatore che non ha effetto "knock on", il sistema funziona correttamente e non si spegne. Abbiamo attivato fino a 10 interrupt al secondo per cinque minuti e il sistema è sempre stato in piedi.

È possibile sperimentare altre periferiche oltre ai pulsanti. Mettiamoci al lavoro!

RIFERIMENTI

PUBBLICAZIONI

[1]. **EXPLORING ARDUINO**, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

SITI WEB

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com