ARDUINO DA ZERO: INTERRUTTORE CREPUSCOLARE

di Davide Fiorino

Arduino è apprezzato in tutto il mondo per la sua semplicità di utilizzo, che lo rende un ottimo prodotto per la realizzazione di prototipi elettronici per scopi hobbistici, didattici e non solo. Se siete in procinto di apprendere Arduino, questa serie di articoli sarà un'ottima guida per familiarizzare e permettervi quindi di costruire quello che avete in mente.

Descrizione del progetto

Non vogliamo annoiare riempendovi di nozioni teoriche su Arduino e sul suo funzionamento, piuttosto, l'approccio che adotteremo in questa guida è quello di partire da progetti pratici e spiegarvi i concetti solo quando sarà necessario. Il primo progetto che realizzeremo è appunto un interruttore crepuscolare. Lo scopo è di creare un sistema che possa accendere o spegnere un dispositivo (per comodità utilizzeremo un led) in base al livello di luce ambientale. Programmeremo quindi il nostro Arduino per leggere il dato sulla luce da un sensore crepuscolare (una fotoresistenza), e al momento opportuno accendere o spegnere un led. Un sistema del genere può avere varie applicazioni, oltre a quello di accendere la luce di casa quando fa buio: si potrebbe, ad esempio, controllare la luminosità di un display o far accendere automaticamente i fari di un'automobile.

L'IDE di Arduino

Per iniziare il nostro progetto abbiamo bisogno, oltre alla scheda, dell'IDE installato su un computer. L'IDE (Integrated Development Environment) di Arduino è un'applicazione Open Source che ci permette di scrivere e caricare facilmente i programmi sulla scheda.

È possibile scaricarla dal sito ufficiale (www.arduino.cc) ed è compatibile con Windows, Mac OS X e Linux. Avviando l'IDE ci troviamo davanti un editor di testo dove è possibile scrivere il codice del programma. Il linguaggio che si utilizza è, praticamente, il C/C++ ed è arricchito da librerie che permettono di interfacciarsi con vari componenti esterni. Spesso i nomi delle funzioni delle librerie di Arduino danno subito l'idea di ciò che fanno, come Setup() e Loop() che vedremo in seguito. I file scritti usando Arduino sono chiamati sketches e hanno estensione ".ino". Come potete vedere nella Figura 1, nella parte alta della finestra si trova una toolbar che include i pulsanti per verificare il codice, per caricarlo sulla scheda e per visualizzare il monitor seriale. Anche se è la prima volta che utilizzate l'IDE vedrete che vi sembrerà subito familiare. Date anche un'occhiata alle preferenze; è possibile modificare la directory dove salvare gli sketch o la lingua.

Componenti necessari

I componenti elettronici che serviranno per questo primo progetto sono davvero semplici. Oltre alla scheda Arduino Uno (vanno bene anche altri modelli) avremo bisogno anche di una fotoresistena, di un diodo led, di una resistenza, di una breadboard e di qualche cavetto per effettuare i contatti. La fotoresistenza, come suggerisce il nome stesso, è un componente elettronico la cui resistenza è inversamente proporzionale alla quantità di luce che lo colpisce. Quindi, il valore in ohm della resistenza diminuirà con l'aumentare

Figura 1: L'interfaccia della IDE di Arduino

della luce che lo investe. La breadboard è una basetta che ci permette di creare tantissimi contatti, sebbene per questo progetto non sia strettamente necessario, vi consigliamo di utilizzarla. Per effettuare correttamente il collegamento elettrico ci servirà anche una resistenza dello stesso ordine di grandezza del valore massimo in ohm della fotoresistenza.

Scrivere il codice

Il codice che permette il funzionamento di questo sistema è molto semplice e si può scrivere in una ventina di righe. Dando un'occhiata al listato 1 si può notare come il linguaggio sia praticamente come il "C", con l'aggiunta di funzioni specifiche. Due funzioni fondamentali nei programmi di Arduino sono la setup() e la loop() che vanno sempre implementate.

Funzioni setup() e loop()

E adesso una breve descrizione sulle due funzioni presenti nel corpo del programma. La funzione setup() viene richiamata quando si collega l'alimentazione alla scheda (via USB o tramite alimentatore) e quando si preme il testo Reset. È usata per inizializzare variabili, pin, mode, importare librerie, ecc.
E' importante ricordare che essa è invocata soltanto una volta nel ciclo del programma.
La funzione loop() invece fa proprio quello che suggerisce il nome stesso: viene eseguita in continuazione dopo che è avvenuto il setup. È proprio all'interno di questa funzione che va inserito il corpo del programma.

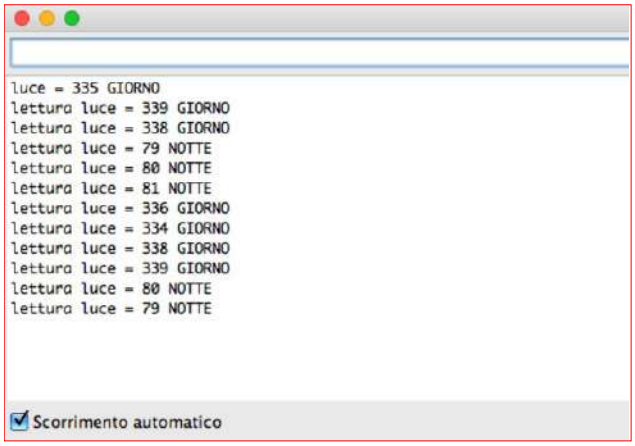

Figura 2: Il monitor seriale di Arduino

Algoritmo

Lo scopo del software è quello leggere il valore della luce ambientale tramite la fotoresistenza e, secondo questo valore, decidere se accendere o spegnere il led. La lettura del dato sulla luce ed il controllo di questo valore, effettuato con una struttura (if…else), sarà eseguito ripetutamente e verrà quindi digitato all'interno della funzione loop().
Ma ogni quando tempo si deve effettuare la lettura? Ovviamente siamo noi a deciderlo, e ciò si realizza utilizzando la funzione delay(), alla quale va passato il valore, in millisecondi, del tempo di attesa, prima di eseguire un nuovo ciclo.

Variabili e costanti

Come potete vedere nelle prime cinque righe del Listato 1, sono dichiarate una serie di variabili e costanti.Le prime quattro, dichiarate con le parole "const unsigned int", sono delle costanti di tipo "numeri interi positivi".
Quando non serve mutare il valore di un dato durante il corso del programma o quando non necessitano numeri negativi è inutile usare un "int". Bisogna considerare che Arduino ha una memoria limitata, e quindi è consigliabile scrivere il codice cercando di "non fare sprechi" di risorse, usando i tipi di dati più appropriati ed evitando le ridondanze.
La variabile ANALOG_IN_PIN contiene il nome del pin analogico al quale è collegato la fotoresistenza (Arduino ha dei pin analogici e digitali). Allo stesso modo LED_PIN contiene il numero del pin digitale al quale sarà collegato il led (abbiamo scelto il pin 13 perché è lo stesso al quale è collegato il led incorporato di Arduino Uno). TIME_INTERVAL contiene il valore di tempo, in secondi, ogni quando sarà eseguito il ciclo (la conversione in millisecondi è eseguita nella funzione delay()). La costante SOGLIA conserva il valore al di sopra del quale sarà acceso il led. Questo valore deve essere impostato la prima volta che si esegue il programma perché varia a seconda del valore della resistenza che inseriamo nel circuito. Infine,

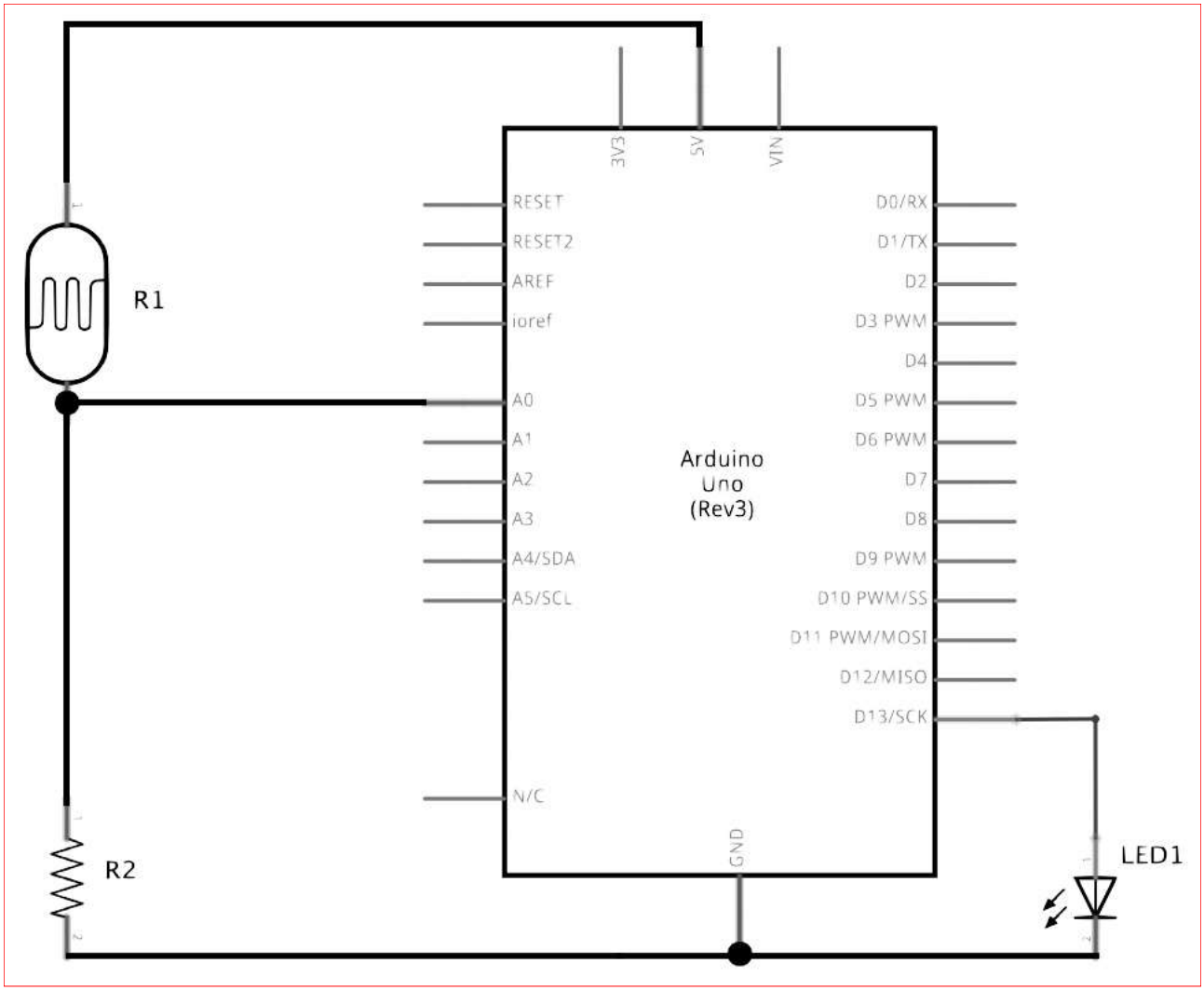

Figura 3: Lo schema elettrico del progetto

nella variabile sensorValue sarà memorizzato il valore della lettura della luce a ogni ciclo di programma.

Impostiamo i pin

Alla riga 6 troviamo la funzione setup() all'interno della quale si impostano i pin e si eseguono altre operazioni. La funzione Serial.begin() apre la comunicazione seriale con Arduino, e imposta la velocità a 9600 baud. La comunicazione tramite il monitor seriale (in Figura 2) è un modo per trasmettere informazioni tra il computer e la scheda. In questo programma ci serve per leggere il dato sulla luminosità dell'ambiente (sensorValue) e regolare, di conseguenza, il valore di soglia al quale accendere il led (SOGLIA). Quest'operazione, come abbiamo detto, andrà fatta solo la prima volta, ma sarà sempre possibile aprire il monitor seriale qualora ci serva conoscere il valore ottenuto dalla fotoresistenza. La funzione pinMode() invece serve per impostare i pin e richiede due parametri: il nome del pin e la modalità nella quale lo vogliamo programmare (input o output). Il pin A0 andrà chiaramente messo in Input perché esso "legge" un dato dall'esterno e il pin 13, al quale è collegato il led, in Output.

Il programma vero e proprio

All'interno della funzione loop() (righe 11 – 23) è scritto il programma vero e proprio, che sarà ripetuto, nel nostro listato, ogni 2 secondi. S'inizia leggendo il valore in ingresso dal pin A0, con la funzione analogRead(), e si memorizza nella variabile sensorValue. Questa è proprio la lettura della luce che prendiamo dalla fotoresistenza. Alle righe 13 e 14, tramite la funzione Serial.print(), visualizziamo sul monitor seriale il valore di questo dato.
Alla riga 15 effettuiamo il controllo con la

Next Generation Intelligent LCDs

demmel products

Intelligent Products for Smart Solutions

www.iLCD.info

RISPARMIA TEMPO.
RISPARMIA DENARO.
RISPARMIA MANODOPERA.

Riduci significativamente i costi di sviluppo

Time-to-market ultra-veloce per le tue applicazioni

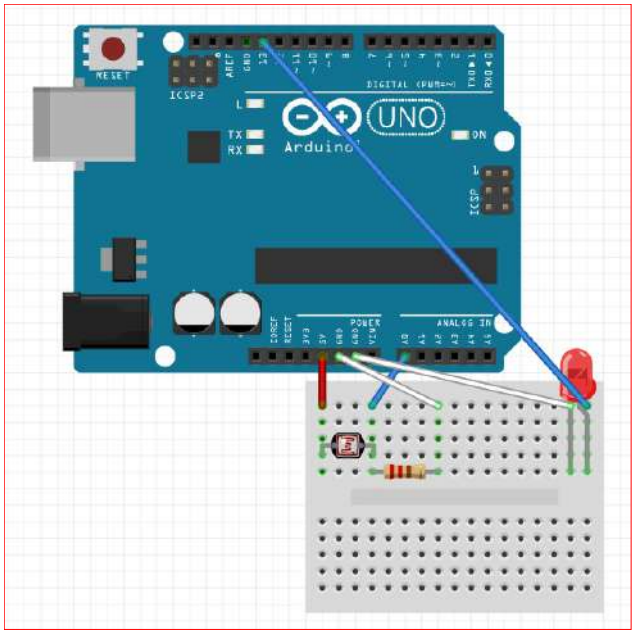

Figura 4: Rendering del progetto con Fritzing

struttura di controllo if. Se il valore della lettura è maggiore del valore di soglia, sarà eseguito il codice all'interno della parentesi graffa. Se la condizione è vera sarà inviata, sul monitor seriale, la parola "Giorno" e il led sarà spento tramite la funzione digitalWrite() che imposta lo stato del pin 13 al valore LOW (no corrente).
Alla riga 18 invece, se la lettura è inferiore del valore di soglia, verrà visualizzata la parola "Notte" e lo stato del pin 13 sarà impostato su HIGH, che corrisponde al passaggio di corrente e quindi all'accensione del led. Alla fine (riga 22) troviamo la funzione delay() che, come abbiamo detto, imposta una pausa. All'interno delle parentesi va inserito il tempo in millisecondi, che è ottenuto moltiplicando TIME_INTERVAL (che per praticità abbiamo scritto in secondi) per 1000.

Collegamenti elettrici

Una volta scritto e compreso il codice, passiamo alla parte "manuale", cioè facciamo i collegamenti elettrici. Si tratta di uno schema molto semplice, e anche se siete dei neofiti, non dovreste avere problemi di sorta. Come potete vedere dallo schema elettrico (Figura 3) i componenti esterni collegati ad Arduino sono soltanto tre: la fotoresistenza (R1), la resistenza (R2) e il led. R1 ed R2 vanno messe in serie, formando un partitore: un capo di R1 va collegato al pin 5V di Arduino ed R2 va collegata, da un lato a R1 (la fotoresisitenza) e dall'altro alla massa (pin GND). All'altro capo della fotoresistenza va collegato il pin A0 di Arduino, dal quale leggeremo il valore della luce. Il led invece andrà semplicemente collegato al pin 13 (terminale lungo) e alla massa (GND) che sulla scheda sono adiacenti. Soltanto il pin 13 ha un resistore interno di protezione, quindi se desiderate collegare il led a un altro pin, ricordatevi di inserire anche una

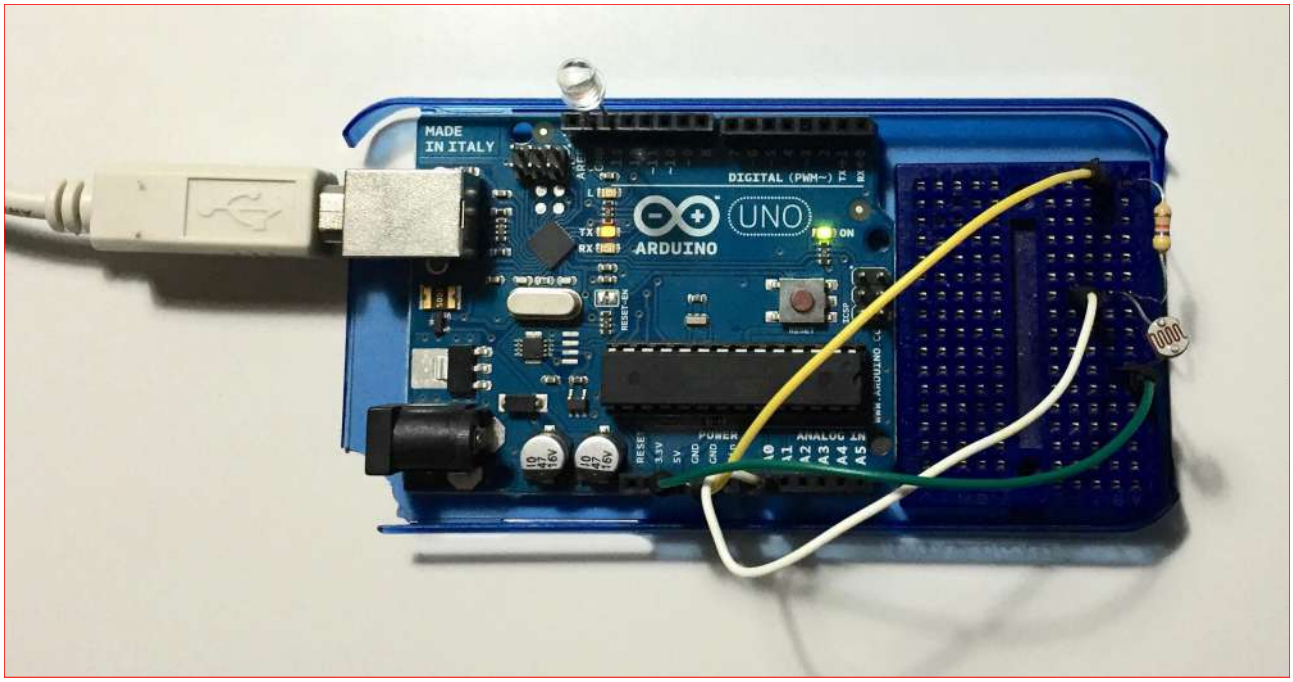

Figura 5: I vari componenti collegati alla scheda Arduino Uno

resistenza. Come abbiamo detto all'inizio è consigliabile usare una breadboard per fare questi collegamenti, come mostrato chiaramente nel rendering (Figura 4). Nel nostro esempio, il progetto è stato realizzato utilizzando una scheda Arduino Uno, ma vanno bene anche gli altri modelli. Alla fine il tutto dovrebbe essere simile a quanto mostrato nella Figura 5.

Test del progetto

Adesso che abbiamo scritto il programma sull'IDE e abbiamo effettuato i collegamenti elettrici sulla nostra scheda è giunta l'ora di provare il funzionamento. Colleghiamo Arduino al computer mediante cavo USB e apriamo l'IDE con il nostro sketch. Prima di caricare il programma sulla memoria della scheda è opportuna fare ancora un paio di cosette. Dal menù strumenti (su Mac), sotto "Scheda", assicuriamoci di aver selezionato il nostro modello di Arduino e nella voce "Porta" selezioniamo la porta alla quale abbiamo collegato la scheda. Prima di fare l'upload dello sketch clicchiamo sul tasto Verifica nella barra in alto a destra; in questo modo il codice sarà compilato e saranno segnalati eventuali problemi. Ovviamente il codice riportato nel listato 1 è stato testato e non contiene errori. A questo punto, con l'Arduino collegato, clicchiamo sul pulsante "Carica" (il secondo da sinistra) e il programma sarà caricato sulla scheda. Effettuato l'upload, il codice sarà immediatamente eseguito da Arduino. Apriamo adesso il monitor seriale (ultimo tasto a destra), assicuriamoci che la velocità sia impostata su 9600 baud e che sia stato selezionata la voce "Scorrimento automatico". In questa finestra dovremmo vedere le risposte delle letture della fotoresistenza, accompagnati dalla scritta "Giorno" o "Notte". Facciamo qualche prova, accendendo e spegnendo la luce in modo tale da individuare un corretto valore di soglia (non è detto che 200 vada bene). Aggiorniamo il valore, sostituendolo a 200 della riga 4 (listato 1). Come si può notare la comunicazione seriale è stata molto utile per avere un riscontro immediato sui valori in input. Possiamo anche modificare il periodo di aggiornamento, cambiando il valore di TIME_INTERVAL (riga 3), rendendo il sistema più reattivo; ad esempio con un tempo di 0.5 secondi l'accensione del led sarà quasi immediata.

Conclusione

In questa prima puntata del nostro piccolo corso su Arduino abbiamo già realizzato un progetto completo che utilizza un sensore che cattura informazioni dall'esterno e che

genera, di conseguenza, un'azione. Questo sistema può essere, volendo, perfezionato pilotando, invece del led, un relè che potrebbe pilotare la luce di casa e, in genere, dei carichi maggiori divenendo, in tal modo, un progetto di domotica. Insomma, con Arduino quello che potete fare dipende soltanto dalla vostra fantasia.

Listato 1

```
1    const unsigned int ANALOG_IN_PIN = A0;
2    const unsigned int LED_PIN = 13;
3    const unsigned int TIME_INTERVAL = 2; //in secondi
4    const unsigned int SOGLIA = 200; //valore luce di soglia
5    int sensorValue = 0;

6    void setup() {
7       Serial.begin(9600);
8       pinMode(ANALOG_IN_PIN, INPUT);
9       pinMode(LED_PIN, OUTPUT);
10   }

11   void loop() {
12      sensorValue = analogRead(ANALOG_IN_PIN);
13      Serial.print("lettura luce = " );
14      Serial.print(sensorValue);
15      if (sensorValue > SOGLIA) {
16         Serial.println(" GIORNO");
17         digitalWrite(LED_PIN,LOW);
18      } else if (sensorValue < SOGLIA) {
19        Serial.println(" NOTTE");
20        digitalWrite(LED_PIN, HIGH);
21      }
22      delay(TIME_INTERVAL*1000);
23   }
```