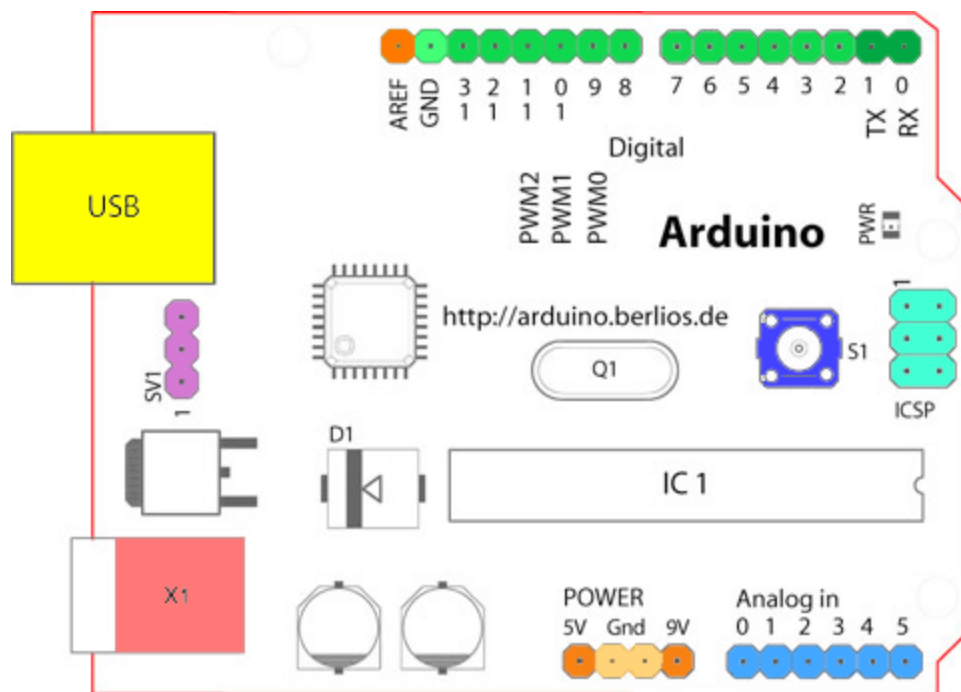


Arduino

Arduino è una piattaforma hardware per il [physical computing](#) sviluppata all'Interaction Design Institute, un istituto fondato dalla [Olivetti](#) e da [Telecom Italia\[1\]](#) con sede a [lvrea](#). Arduino può essere utilizzato per lo sviluppo di oggetti interattivi [stand-alone](#) ma può anche interagire, tramite collegamento, con software residenti su computer, come [Adobe Flash](#), [Processing](#), [Max/MSP](#), [Pure Data](#), [SuperCollider](#), [Vvvv](#).

La piattaforma hardware Arduino è distribuita agli [hobbisti](#) in versione generalmente pre-assemblata, acquistabile in internet o in negozi specializzati, ma le informazioni sul progetto hardware sono rese disponibili a tutti quale [hardware open source](#) nei termini della licenza [Creative Commons Attribution-ShareAlike 2.5](#). In questo modo, chiunque lo desideri può legalmente auto-costruirsi un clone di Arduino o derivarne una versione modificata, scaricando gratuitamente lo [schema elettrico](#) e l'elenco dei componenti elettronici necessari. Il codice sorgente sono disponibili, e concessi in uso, secondo i termini legali della licenza [GPLv2](#). Il progetto ha preso avvio in [Italia](#) a [lvrea](#), nel 2005, con lo scopo di rendere disponibile, a progetti di [Interaction design](#) realizzati da studenti, un [device](#) per il controllo che fosse più economico rispetto ai sistemi di [prototipazione](#) disponibili all'epoca.

I progettisti sono riusciti nell'intento di creare una piattaforma di semplice utilizzo ma che, al tempo stesso, permettesse una significativa riduzione dei costi rispetto a molti prodotti disponibili sul mercato. A ottobre 2008 erano già stati venduti più di 50.000 esemplari di Arduino in tutto il mondo.



Starting clockwise from the top center:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (digitalRead and digitalWrite) if you are also using serial communication (e.g. Serial.begin).
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

Input/Output

Per comunicare con altri dispositivi / componenti, Arduino è dotato di una serie di PIN, 13 Digitali e 6 Analogici. Questi gli permettono di inviare e ricevere segnali elettrici digitali (0 -1 logico) o analogici (0 - 5V).

Ogni PIN può essere settato in modalità INPUT o OUTPUT a seconda che si voglia inviare o ricevere un segnale

A/D converter

I PIN analogici sfruttano un convertitore analogico/digitale (A/D) a 6 canali. Il convertitore ha una risoluzione di 10 bit, restituendo valori interi da 0 a 1023 rappresentanti il segnale in ingresso, nel caso in cui il PIN sia settato in modalità INPUT.

Nel caso sia in modalità OUTPUT, emette un segnale di tensione compresa tra 0 e 5V a seconda del valore passato in ingresso alla funzione utilizzata per controllare il PIN.

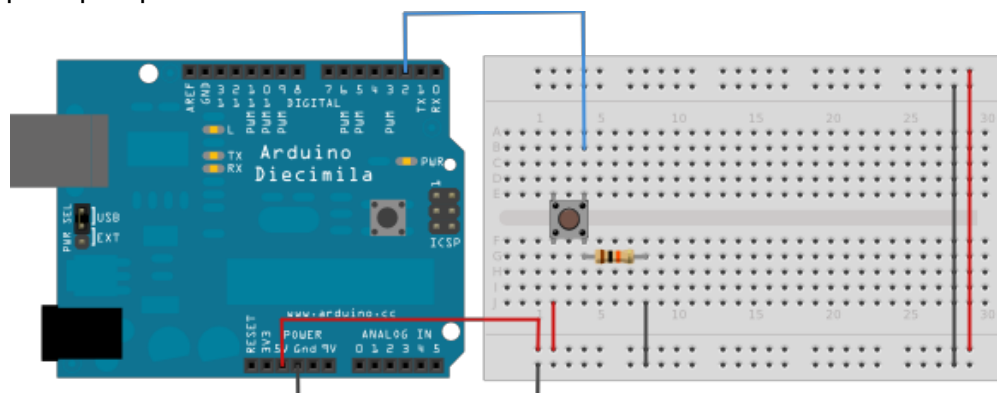
Pushbutton

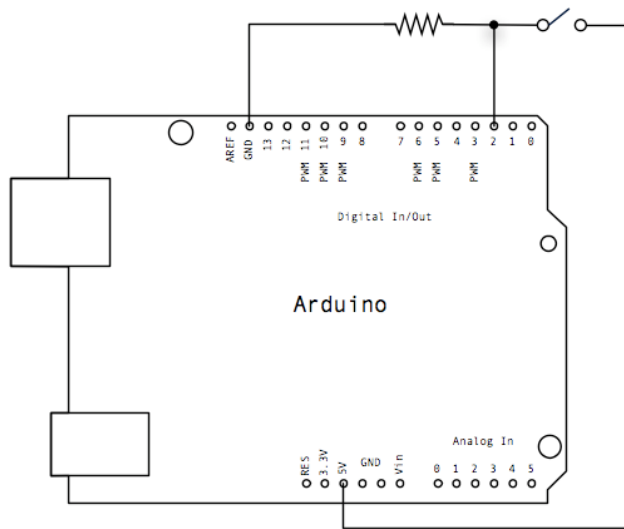
Connect three wires to the Arduino board. The first two, red and black, connect to the two long vertical rows on the side of the breadboard to provide access to the 5 volt supply and ground. The third wire goes from digital pin 2 to one leg of the pushbutton. That same leg of the button connects through a pull-down resistor (here 10 KOHms) to ground. The other leg of the button connects to the 5 volt supply.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.

You can also wire this circuit the opposite way, with a pullup resistor keeping the input HIGH, and going LOW when the button is pressed. If so, the behavior of the sketch will be reversed, with the LED normally on and turning off when you press the button.

If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, it will randomly return either HIGH or LOW. That's why you need a pull-up or pull-down resistor in the circuit.





Leggere un Potenzimetro (analog input)

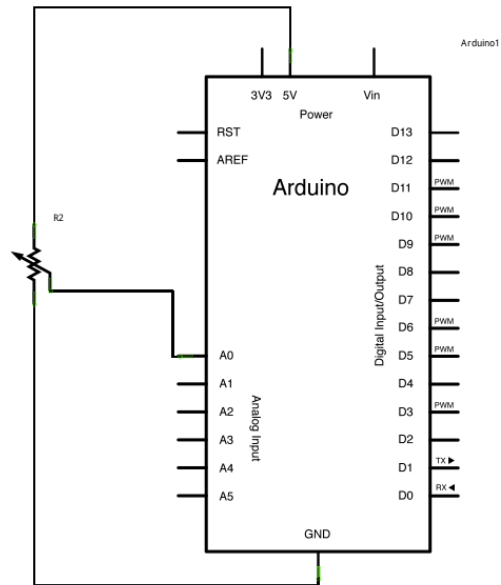
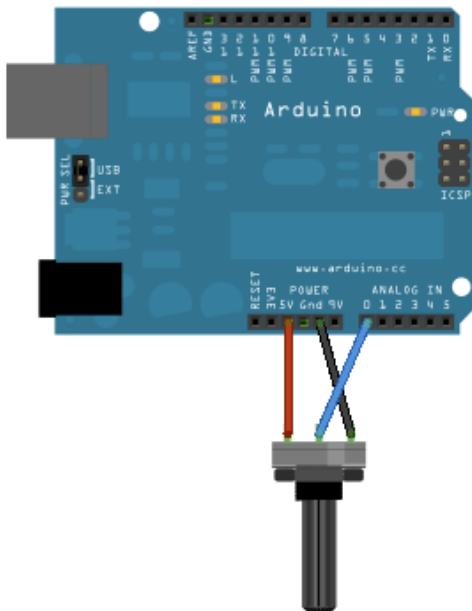
Un potenziometro è semplicemente una manopola che possiede una resistenza variabile, che possiamo leggere su Arduino tramite uno degli ingressi analogici.

In questo esempio, questo valore, controlla la frequenza alla quale il LED lampeggia.

Collegeremo tre cavi all'Arduino

- Il primo va dalla massa a uno dei pin esterni del potenziometro
- Il secondo va dai 5V all'altro pin esterno del potenziometro
- Il terzo va dall'input analogico 2 al pin intermedio del potenziometro

Girando la manopola del potenziometro, cambiamo la resistenza che si oppone al passaggio della corrente ai PIN ai lati del potenziometro, deviandone parte su quello centrale. Questo cambiamento, viene letto da Arduino, che converte il valore in ingresso in un numero intero. Quando la manopola è totalmente girata in un senso, ci saranno 0 Volt che arrivano al PIN intermedio, e leggeremo 0. Quando la manopola è totalmente girata nel senso opposto, avremo 5 volts in ingresso sul PIN, e avremo come valore acquisito "1023" (2^{10} bit di risoluzione del convertitore). In tutti i casi intermedi, la funzione `analogRead()` ritornerà un numero tra 0 e 1023 che è proporzionale al voltaggio applicato al pin intermedio.



```
/*
  ReadAnalogVoltage
  Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial
  monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and
  ground.
```

This example code is in the public domain.
 */

```
// the setup routine runs once when you press reset:
```

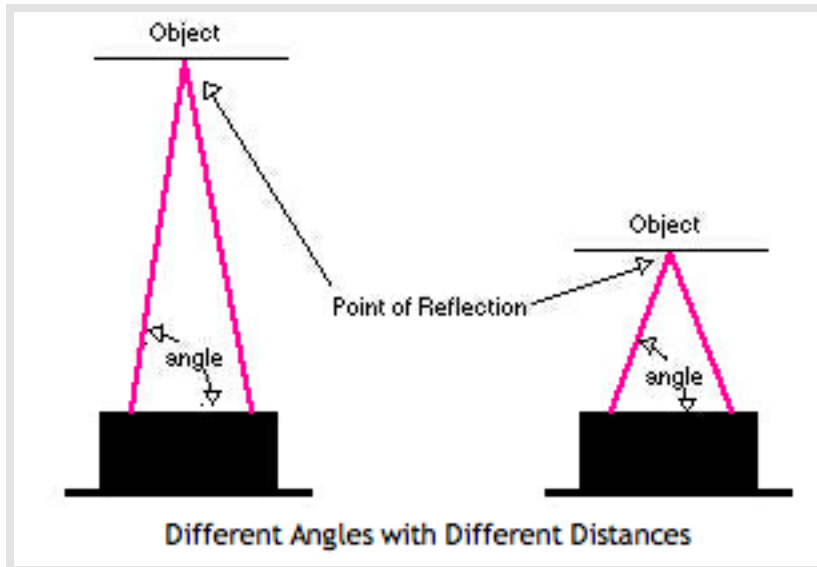
```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
```

```
// the loop routine runs over and over again forever:
```

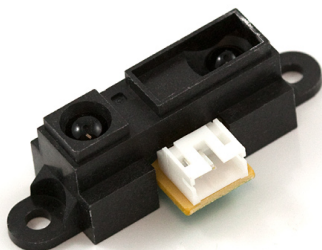
```
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

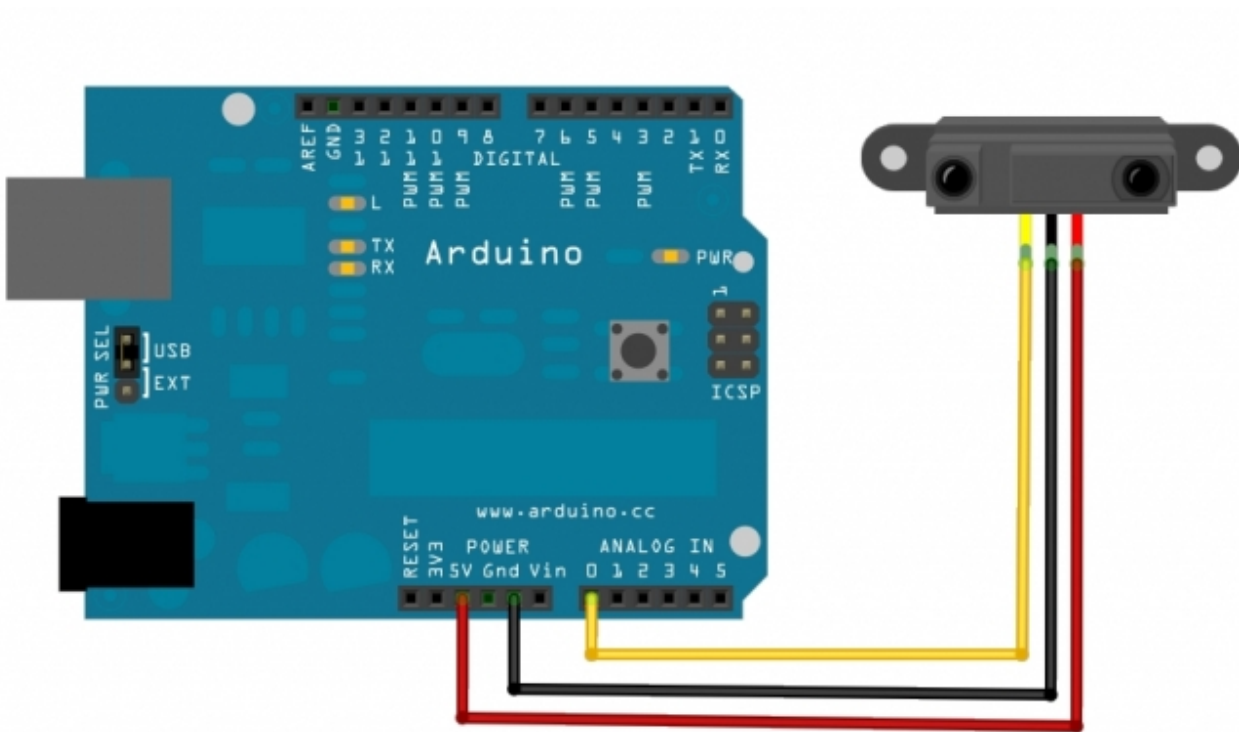
Sensore IR Sharp GP2Y0A21YK:

Questa tipologia di sensore funziona sul principio della riflessione, ossia un led a raggi infrarossi emette una luce (impercettibile all'occhio umano e non dannosa) ed un ricevitore cattura il fascio dopo che questo si è riflesso contro l'ostacolo:



La distanza è calcolata grazie all'angolo con cui il fascio di luce si infrange contro la parte ricevente del sensore, questo angolo viene trasformato dall'elettronica a bordo del sensore in una variazione in volt da 2.55 a 1.95 ossia a distanza inferiore corrisponde maggiore voltaggio erogato.





Protocollo MIDI:

Intro

Il protocollo MIDI è un insieme di specifiche che hanno permesso di realizzare un sistema standard di comunicazione tra i vari apparecchi elettronici musicali.

Tramite MIDI si possono inviare note, messaggi di controllo e di sincronizzazione del tempo. Senza entrare nel dettaglio di tutte le sue funzionalità, esamineremo solo gli aspetti che ci interessano di questo sistema, cioè la possibilità di inviare dei messaggi di controllo (Control Change) che rappresentano la variazione di un controllo fisico (una manopola del nostro controller) e permettono allo strumento (Ableton) di interpretare il dato e mapparli su un parametro di un synth effetto (frequenza di taglio di un filtro).

Ogni pacchetto di informazioni che rappresenta una variazione da inviare allo strumento, è chiamato "messaggio midi".

Messaggi MIDI

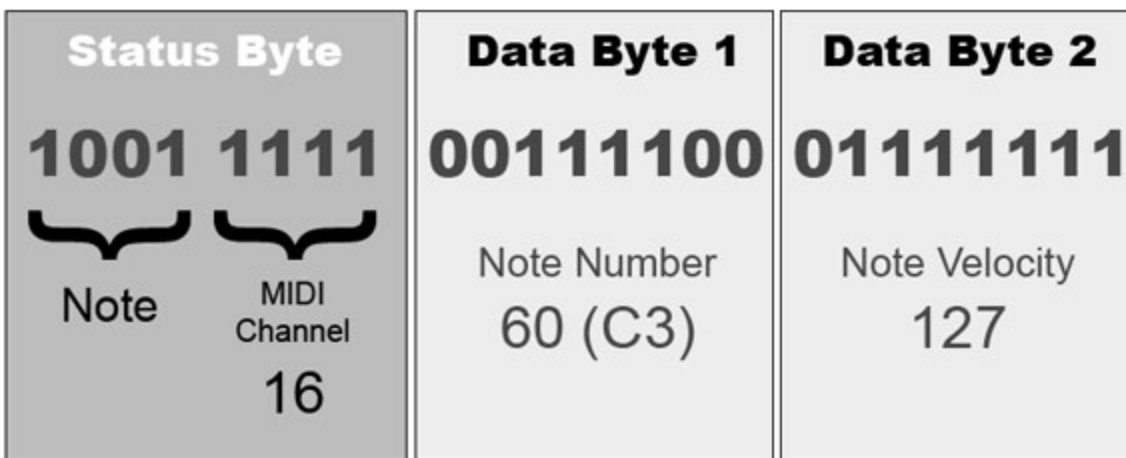
Ogni messaggio è formato da 3 byte:

- 1 byte di parametro di comando (es: "controllo cambiato") e numero di canale
- 2 di parametri (per esempio, il numero del controllo cambiato, e il suo valore effettivo)

I comandi e i valori sono rappresentati tramite numeri esadecimali:

Comando	Esadecimale	Parametro 1	Parametro 2
Note Off	0x80	key (nota)	velocity
Note On	0x90	key	velocity
Aftertouch	0xA0	key	touch
Continuos Controller	0xB0	numero controller	valore controller
Patch Change	0xC0	numero strumento	numero patch
Channel Pressure	0xD0	pressure	-
Pitch Bend	0xE0	lsb	msb
non musical commands	0xF0		

Per esempio, un messaggio noteOn che indica che è stata premuta la nota C3 a massima velocity, e deve essere inviato al dispositivo collegato sul canale MIDI 16 è formato così



[ESEMPIO MIDI CC MESSAGE]

Il controller che costruiremo, invierà dei segnali CC di 2 tipi:

- Potenziometri: da 0 a 127 a seconda di quanto è girata la manopola#

- Bottoni: 127 quando premuto , 0 quando rilasciato

Invio dei messaggi MIDI:

Per inviare i messaggi MIDI al PC, dovremo quindi costruire una struttura dati che rispetti quella forma, e inviarli a una delle velocità supportate dal protocollo MIDI (es 115200 bps) , sulla porta seriale.

```
typedef union {  
    struct {  
        uint8_t command;  
        uint8_t channel;  
        uint8_t data2;  
        uint8_t data3;  
    }  
    msg;  
    uint8_t raw[4];  
}  
t_midiMsg;
```

Scrivendo direttamente questa struttura sulla porta seriale, il PC riceverà il corrispondente messaggio midi, per esempio:

```
void setup() {  
    Serial.begin(115200);  
}
```

```
void loop() {
```

```
#define MIDI_COMMAND_NOTE_OFF    0x80  
#define MIDI_COMMAND_NOTE_ON    0x90  
#define MIDI_COMMAND_NOTE_CC    0xB0
```

```
t_midiMsg midiMsg;
```

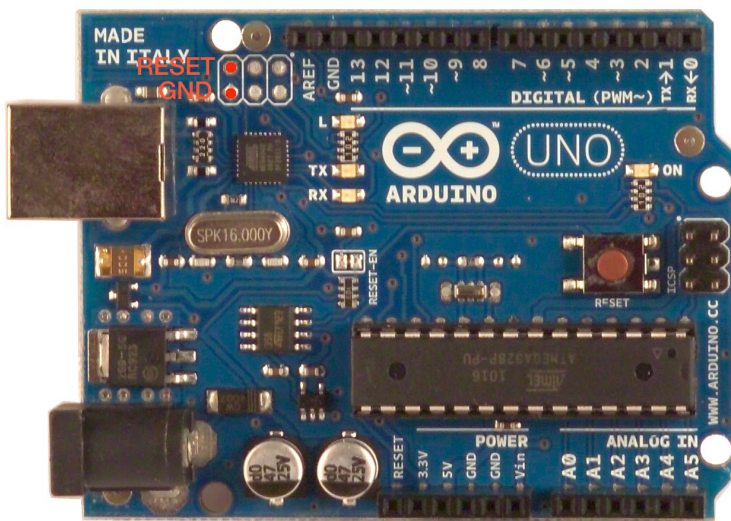
```
    midiMsg.msg.command = MIDI_COMMAND_NOTE_CC; //CC command  
    midiMsg.msg.channel = 1; // on channel 1  
    midiMsg.msg.data2 = 2;  
    Serial.write(midiMsg.raw, sizeof(midiMsg));  
}
```

Questo piccolo sketch, invierà un MIDI CC sul canale 1 , e con valore “2” , potete verificare che stia funzionando tramite Midi-Ox / Midi Monitor o un programma simile

Sovrascrittura del firmware

Per permettere al Arduino di comunicare tramite MIDI, ed essere riconosciuto come un generico dispositivo MIDI nativo (senza quindi bisogno di driver aggiuntivi / convertitori seriale - midi) , dovremo sovrascrivere il suo firmware, possibilità che è stata aggiunta in seguito al passaggio di Arduino al chip **ATmega16U2**

Le versioni più recenti di Arduino, permettono di programmare il firmware direttamente dalla porta USB, collegando temporaneamente 2 pin:



per le versioni precedenti fare riferimento a:

<http://arduino.cc/en/Hacking/DFUProgramming8U2>

Il firmware che utilizzeremo , è HIDuino : <https://github.com/ddiakopoulos/hiduino>

- 1) Assicurativi di aver caricato lo sketch MIDI che volete utilizzare
- 2) Scaricare il programmatore da <http://www.atmel.com/tools/flip.aspx>
- 3) Scaricare il firmware corrispondente alla propria versione di arduino da <https://github.com/ddiakopoulos/hiduino/tree/master/Compiled%20Firmwares>
- 4) Aprire il programmatore, e per sicurezza fare una copia backup del firmware attuale
- 5) Dal programmatore, fare “erase” della memoria

6) Caricare il nuovo firmware (File => Load Hex File)

6) Scrivere il firmware (Device => Program)

A questo punto il vostro Arduino entrerà in “modalità MIDI” e non sarà quindi più possibile scrivere nuovi sketch tramite il suo IDE. Per ripristinare questa possibilità, ricaricare il firmware originale backupato, con la stessa procedura descritta sopra, oppure riscaricare l’originale dal sito di Arduino.

Scollegando e ri-collegando il cavo USB al vostro PC, vedrete che verrà riconosciuto dall’OS come un “Generico dispositivo MIDI” , e sarà quindi utilizzabile da qualunque DAW

Ir Midi Controller: Esempio completo

Per mettere in pratica quanto appena visto, realizzeremo un controller dotato di un sensore IR, che invia un MIDI CC proporzionale alla distanza della mano che gli muoveremo sopra.

Per rendere più immediato il feedback, inseriremo anche un LED RGB che si colorerà progressivamente da blu, a verde, poi arancione, e infine a rosso, sempre in maniera proporzionale alla distanza della nostra mano.

Per rendere più agevole il suo utilizzo, invieremo anche un altro messaggio su un differente numero di CC; per permetterci per esempio, di attivare o disattivare l'effetto che stiamo comandando.

```
// Init the Pins used for PWM
const int redPin = 9;
const int greenPin = 10;
const int bluePin = 11;
const int IRpin = A0;           // analog pin for reading the IR sensor
int prevVal = 100;
const int zeroThresold = 15;

#define MIDI_COMMAND_NOTE_OFF    0x80
#define MIDI_COMMAND_NOTE_ON    0x90
#define MIDI_COMMAND_NOTE_CC    0xB0

/* The format of the MIDI message to send via serial */
typedef union {
    struct {
        uint8_t command;
        uint8_t channel;
        uint8_t data2;
        uint8_t data3;
    }
    msg;
    uint8_t raw[4];
}
t_midiMsg;

void setup() {
    pinMode(redPin, OUTPUT);
```

```

pinMode(greenPin, OUTPUT);
pinMode(bluePin, OUTPUT);

Serial.begin(115200);
delay(200);
}

void loop() {
  float volts = analogRead(IRpin); //Read IR sensor value
  volts = constrain(map(volts,0,620,1,127),0,127);
  t_midiMsg midiMsg;

  midiMsg.msg.command = MIDI_COMMAND_NOTE_CC; //CC command
  midiMsg.msg.channel = 1; // on channel 1
  midiMsg.msg.data2 = 2; // CC #2

  if(volts > zeroThresold){
    midiMsg.msg.data3 = volts; /* Actual CC value */
    /* Send CC value */
    Serial.write(midiMsg.raw, sizeof(midiMsg)); //Send Midi packet
  }else if(volts < zeroThresold && prevVal > 15){
    /* Send 0 for every value below threshold*/
    midiMsg.msg.data3 = 0;
    Serial.write(midiMsg.raw, sizeof(midiMsg)); //Send Midi packet
  }
  prevVal = volts; //Store previous value to compare on next iteration

  //Lights the led according to midi value
  meter(volts);

  //Wait to let IR sensor re-shot
  delay(40);
}

/**
 * Funzione per accendere un LED RGB con un colore proporzionale al valore MIDI in
 * ingresso, compreso tra 0 e 127
 */
void meter(int level){
  // 0 - 127
  int r,g,b;
  if(level < 15){

```

}

--

<i>number base equivalences</i>								
dec	hex	bin		dec	hex	bin		
=====								
=====								
0	0	0		8	8	1000		
1	1	1		9	9	1001		
2	2	10		10	A	1010		
3	3	11		11	B	1011		
4	4	100		12	C	1100		
5	5	101		13	D	1101		
6	6	110		14	E	1110		
7	7	111		15	F	1111		

BIT

cifra binaria, (in inglese "binary digit") ovvero uno dei due simboli del [sistema numerico binario](#), classicamente chiamati zero (0) e uno (1).

Byte:

Dal [1964](#) il byte è tipicamente formato da 8 bit[\[4\]](#) ed è pertanto in grado di assumere $2^8 = 256$ possibili valori (da 0 a 255). Gli [informatici](#) di [lingua francese](#) utilizzano il termine octet (ovvero ottetto), sebbene il termine venga utilizzato in inglese per denotare una generica sequenza di otto bit.

Componenti:

http://www.robot-italy.com/it/arduino-uno-r3.html	€ 22,33
http://www.robot-italy.com/it/bourns-potentiometer-slide-50k-with-green-led.html	€ 4,53
http://www.robot-italy.com/it/alps-500kohm-potentiometer-9mm.html	€ 1,83
http://www.robot-italy.com/it/itead-ultrasonic-ranging-module-hc-sr04.html	€ 3,66
http://www.robot-italy.com/it/arcade-push-button-33mm-convex-green.html	€ 1,20
http://www.sparkfun.com/datasheets/Sensors/Infrared/GP2D120XJ00F_SS.pdf	

Contatti:

Teo

kingofska [at] gmail [dot] com

Links:

<http://it.wikipedia.org/wiki/Arduino>

<http://arduino.cc/en/Hacking/DFUProgramming8U2>

http://it.wikipedia.org/wiki/Legge_di_Ohm

http://it.wikipedia.org/wiki/Partitore_di_tensione

Sensore IR :

http://www.sparkfun.com/datasheets/Sensors/Infrared/GP2D120XJ00F_SS.pdf

<http://www.youtube.com/watch?v=HWQhY1StF7o>

<http://www.youtube.com/watch?v=D-Ea5Gslg8I>

<http://www.mauroalfieri.it/elettronica/tutorial-sharp2d120x-e-arduino.html>

Contatti:

angrybit[at]artathack[dot]me