Arduino Lab

Software in contatto con l'ambiente

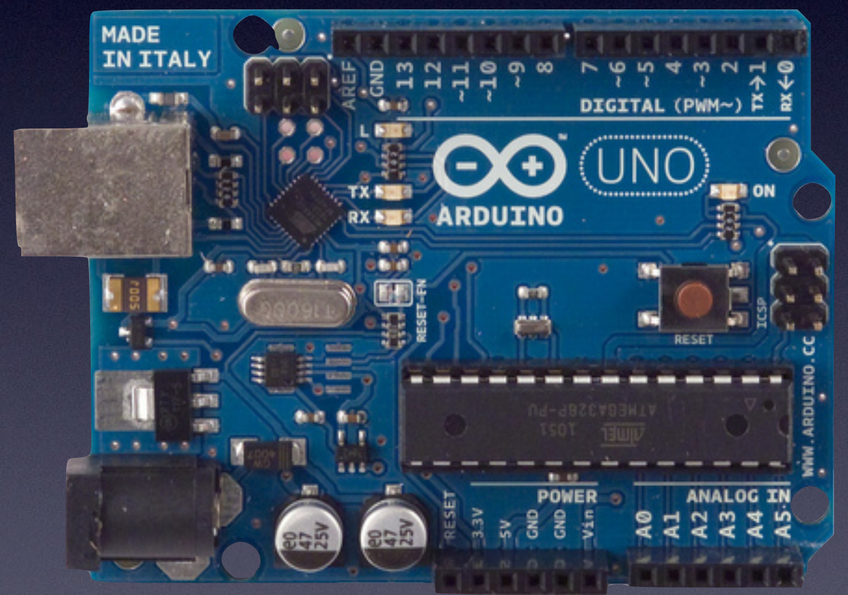
Mario Restuccia - Tancredi Orlando

Arduino

Arduino è una piattaforma di prototipazione open source
Disponibile in vari modelli

Hardware (Arduino UNO):

- Board con microcontroller (ATmega328)
- 14 pin I/O digitale
- 6 pin di input analogico
- Connessione USB usata come alimentazione, porta seriale e per caricare gli *sketch*
- Made in Italy
- Documentazione su come costruire la scheda disponibile sotto CC BY-SA 2.5



Arduino UNO, il modello da noi usato

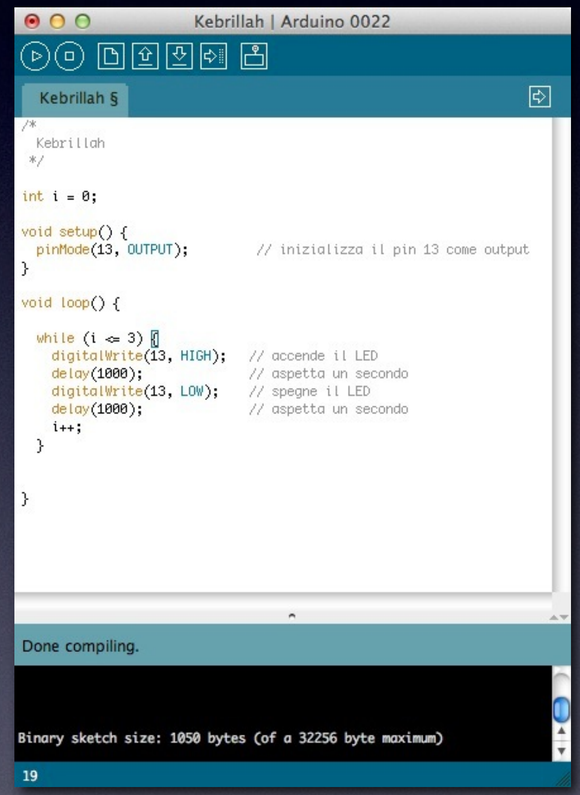
http://arduino.cc

Arduino

Arduino è un ambiente di sviluppo open source

Software:

- Basato su Processing e altro software open source
- Usa il linguaggio Wiring, basato su C++
- Multipiattaforma
- Utilizzato per scrivere il codice e caricarlo sulla board
- Disponibile sotto licenza GPL v.2



Arduino 0022

http://arduino.cc

Kebrillah

Esempio di accensione di un LED

```
1. void setup() {
2.   pinMode(13, OUTPUT);
3. }
```

```
1. Metodo “setup”, eseguito una volta sola
2. Inizializza il pin 13 come output
3. Fine del metodo “setup”
```

```
5. void loop() {
6.     digitalWrite(13, HIGH);
7.     delay(2000);
8.     digitalWrite(13, LOW);
9.     delay(2000);
10.}
```

```
5. Metodo “loop”, eseguito in continuazione
6. Alimenta il pin 13, accendendo il LED
7. Aspetta 2 secondi
8. Spegne il LED
9. Aspetta 2 secondi
10.Fine del metodo “loop”
```

Pin 13
(resistenza 1KΩ)

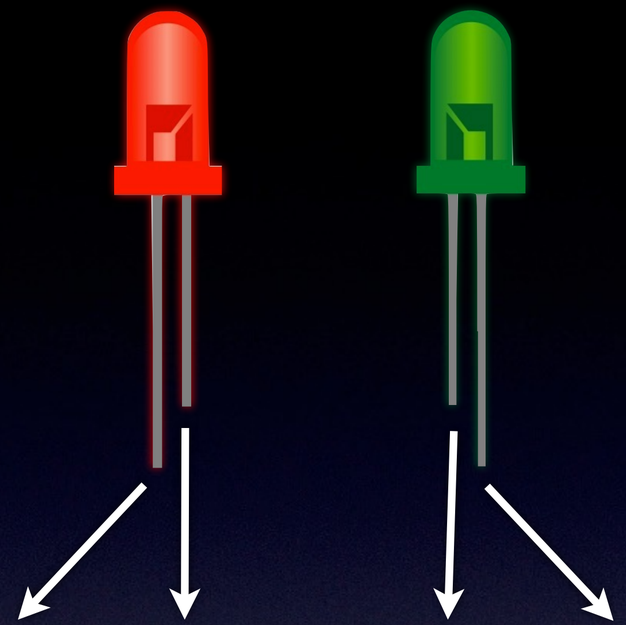
Ground pin

SeriaLED

Accende dei LED e comunica via seriale

```
1. void setup() {
2.   pinMode (13, OUTPUT);
3.   pinMode (4, OUTPUT);
4.   Serial.begin (9600);
5. }

7. void loop() {
8.   Serial.println("verde");
9.   digitalWrite(13, HIGH);
10.  delay(1000);
11.  digitalWrite(13, LOW);
12.
13.  Serial.println("rosso");
14.  digitalWrite(4, HIGH);
15.  delay(1000);
16.  digitalWrite(4, LOW);
17.
18.  Serial.println("Hello MeLUG!");
19.  delay(5000);
20.}
```

```
1. Metodo setup
2. Imposta il pin 13 (LED verde) come output
3. Imposta il pin 4 (LED rosso) come output
4. Numero di bit al secondo per seriale
5. Fine metodo setup

7. Metodo loop
8. Invia via seriale il messaggio “verde”
9. Accende il LED connesso al pin 13
10.Aspetta 1 secondo
11.Spegne il LED

13.Invia il messaggio “rosso”
14.Accende il LED connesso al pin 4
15.Aspetta 1 secondo
16.Spegne il LED

18.Invia un messaggio di saluto via seriale
19.Aspetta 5 secondi
20.Fine metodo loop
```



Pin 4
(resistenza 1KΩ)

Ground pin

Pin 13
(resistenza 1KΩ)

Arduino

Porta seriale USB

Computer

Sounduino

Genera frequenze audio inviate via seriale

Codice Arduino

```
1. const int BUZZ =   13;
2. const int BAUD = 9600;
3. int note = 0;

5. void setup()
6. {
7.   pinMode(BUZZ, OUTPUT);
8.   Serial.begin(BAUD);
9. }

11.void loop() {
12.  while (Serial.available() > 0) {
13.    int rx = Serial.read();
14.    if (rx != ';') {
15.      note = note + rx;
16.    } else {
17.      if (note != 0) {
18.        tone(BUZZ, note);
19.        note = 0;
20.      } else {
21.        noTone(BUZZ);
22.      }
23.    }
24.  }
25.}
```

1. Assegna alla costante BUZZ il numero 13
2. Baud rate (per la comunicazione seriale

7. Inizializza il pin 13 come output
8. Inizializza la comunicazione seriale

12.Ascolta i dati in arrivo via seriale
13.Assegna alla variabile rx i dati via seriale
14.Se ciò che riceve è diverso da ";"
15.Incrementa la variabile note del valore rx
16.Altrimenti
17.Se la variabile note è diversa da zero
18.La riproduce
19.E reimposta a zero la variabile note
20.Altrimenti
21.Esegue una pausa senza emettere suoni

Codice Perl

```
1.  #!/usr/bin/perl
2.  use strict;
3.  use warnings;
4.  use Device::SerialPort;
5.  use Time::HiRes qw(usleep); # For sleep in ms

7.  if ($#ARGV + 1 != 2) {
8.    print "Usage: $0 port filename\n";
9.    print "Example: $0 /dev/ttyASM0 money.txt\n";
10.   exit 1;
11. }

13. my $file = $ARGV[0];
14. my $dev  = $ARGV[1];

16. if (!-e $file || !-e $dev) {
17.   print "File or brain not found.\n";
18.   exit 1;
19. }

21. my $arduino = DeviceSerialPort->new($dev);
22. $arduino->baudrate(9600);
23. $arduino->databits(8);
24. $arduino->parity("none");
25. $arduino->stopbits(1);
```


```
1.  #!/usr/bin/perl
2.  use strict;
3.  use warnings;
4.  use Device::SerialPort;
5.  use Time::HiRes qw(usleep); # For sleep in ms

7.  if ($#ARGV + 1 != 2) {
8.    print "Usage: $0 port filename\n";
9.    print "Example: $0 /dev/ttyASM0 money.txt\n";
10.   exit 1;
11. }

13. my $file = $ARGV[0];
14. my $dev  = $ARGV[1];

16. if (!-e $file || !-e $dev) {
17.   print "File or brain not found.\n";
18.   exit 1;
19. }

21. my $arduino = DeviceSerialPort->new($dev);
22. $arduino->baudrate(9600);
23. $arduino->databits(8);
24. $arduino->parity("none");
25. $arduino->stopbits(1);
```

```
27. my %frequencies; eval { %frequencies = do "frequencies.pl"; };
28. open NOTES, "$file";

30. my ($tempo, @tone, $note, $duration);

32. while (<NOTES>) {
33.   chomp;       # No newline
34.   s/#.*//;     # No comments
35.   s/^\s+//;    # No leading white
36.   s/\s+$//;    # No trailing white
37.   next unless length;
38.   if ($_ =~ m/^TEMPO/) {
39.         $tempo = split(/\s+/, $_, -1);
40.         print "Tempo is $tempo.";
41.   } else {
42.         @tone = split(/\s+/, $_);
43.   }
44.   $note = $frequencies{$tone[0]};
45.   $duration = $tone[1]*$tempo;
46.   print "Playing $tone[0] (\@$note Hz) for $tone[1] units ($duration ms).";
47.   while ($note > 255) {
48.         $arduino->write(chr(255));
49.         $note -= 255;
50.   }
51.   $arduino->write(chr($note));
52.   $arduino->write(";");
53.   usleep($duration);
54. }

55. close NOTES;
```

```
27. my %frequencies; eval { %frequenc"; };
28. open NOTES, "$file";

30. my ($tempo, @tone, $note, $duration);

32. while (<NOTES>) {
33.   chomp;       # No newline
34.   s/#.*//;     # No comments
35.   s/^\s+//;    # No leading white
36.   s/\s+$//;    # No trailing white
37.   next unless length;
38.   if ($_ =~ m/^TEMPO/) {
39.         $tempo = split(/\s+/, $_, -1);
40.         print "Tempo is $tempo.";
41.   } else {
42.         @tone = split(/\s+/, $_);
43.   }
44.   $note = $frequencies{$tone[0]};
45.   $duration = $tone[1]*$tempo;
46.   print "Playing $tone[0ne[1]
47.   while ($note > 255) {
48.         $arduino->write(chr(255));
49.         $note -= 255;
50.   }
51.   $arduino->write(chr($note));
52.   $arduino->write(";");
53.   usleep($duration);
54. }

55. close NOTES;
```

Demo

Displaytemp

Monitora la temperatura e la mostra sul display LCD

Codice Arduino

```
1. #include <LiquidCrystal.h>

3. const int inPin = 0;
4. const int hotLed = 8;
5. const int coldLed = 13;
6. const int hot = 25;
7. const int cold = 15;
8. LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

10.void setup() {
11.  pinMode(hotLed, OUTPUT);
12.  pinMode(coldLed, OUTPUT);
13.  Serial.begin(9600);
14.  lcd.begin(16, 2);
15.  lcd.print("Salve a tutti");
16.}

18.void loop() {
19.  lcd.setCursor(0, 0);
20.  int value = analogRead(inPin);
21.  long celsius = (value * 500L) /1024;
```

```
1. Assegna alla costante BUZZ il numero 13
2. Baud rate (per la comunicazione seriale

7. Inizializza il pin 13 come output
8. Inizializza la comunicazione seriale

12.Ascolta i dati in arrivo via seriale
13.Assegna alla variabile rx i dati via seriale
14.Se ciò che riceve è diverso da ";"
15.Incrementa la variabile note del valore rx
16.Altrimenti
17.Se la variabile note è diversa da zero
18.La riproduce
19.E reimposta a zero la variabile note
20.Altrimenti
21.Esegue una pausa senza emettere suoni
```

```
1. Serial.print("Temperatura: " +celsius);
2.   lcd.print("Temp: " +celsius);
3.   if(celsius > hot) {
4.     digitalWrite(hotLed, HIGH);
5.     Serial.println("pin caldo acceso");
6.     Serial.print("C'è caldo!");
7.     lcd.setCursor(0, 1);
8.     lcd.print("C'è caldo!");
9.   }
10.  else {
11.    if(celsius < cold) {
12.      digitalWrite(coldLed, HIGH);
13.      Serial.println("pin freddo acceso");
14.      Serial.print("C'è freddo!");
15.      lcd.setCursor(0, 1);
16.      lcd.print("C'è freddo!");
17.    }
18.    else {
19.      digitalWrite(coldLed, LOW);
20.    }
21.    digitalWrite(hotLed, LOW);
22.  }
23.  delay(30000);
24.}
```

```
1. Assegna alla costante BUZZ il numero 13
2. Baud rate (per la comunicazione seriale



7. Inizializza il pin 13 come output
8. Inizializza la comunicazione seriale



12.Ascolta i dati in arrivo via seriale
13.Assegna alla variabile rx i dati via seriale
14.Se ciò che riceve è diverso da ";"
15.Incrementa la variabile note del valore rx
16.Altrimenti
17.Se la variabile note è diversa da zero
18.La riproduce
19.E reimposta a zero la variabile note
20.Altrimenti
21.Esegue una pausa senza emettere suoni
22.1
23.2
24.3
```

Demo

TwitterArduino

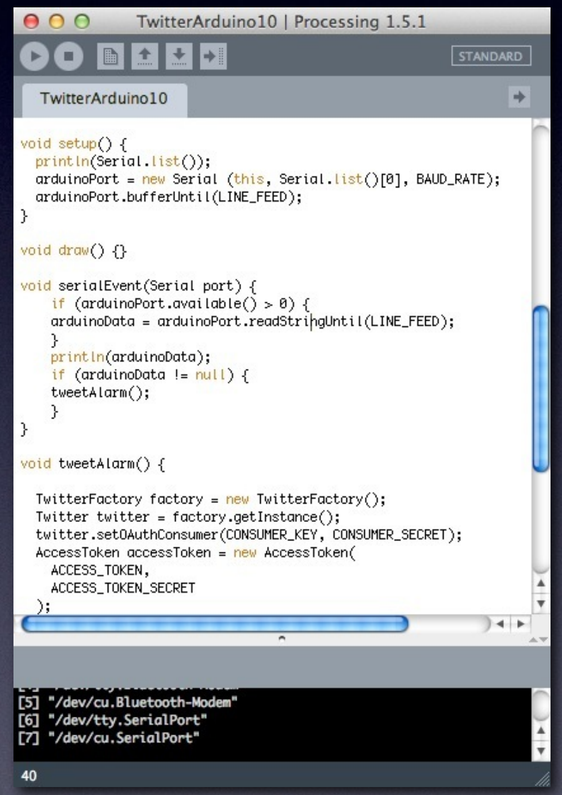
Twitta i dati ricevuti via seriale da Arduino

Processing

Processing è un ambiente di sviluppo open source

IDE:

- Linguaggio di programmazione basato su Java
- L'IDE di Arduino è derivato da Processing
- Perfetto per lavorare con Arduino
- Multipiattaforma
- Disponibile sotto licenza GPL



Processing 1.5.1

http://processing.org

Codice Processing

Versione semplificata senza UI

```
1. import processing.serial.*;

3. final int LF = 10;
4. Serial port;
5. String data;
6. final int BAUD = 9600;
7. final String C_KEY = "●●●";
8. final String C_SECRET = "●●●";
9. final String A_TOKEN = "●●●";
10.final String A_TOKEN_SECRET = "●●●";

11.void setup() {
12.  println(Serial.list());
13.  port = new Serial (this, Serial.list()[0], BAUD);
14.  port.bufferUntil(LF);
15.}

17.void draw() {}

19.void serialEvent(Serial port) {
20.    if (port.available() > 0) {
21.    data = port.readStringUntil(LF);
22.    }
23.    println(data);
```

```
1. Assegna alla costante BUZZ il numero 13
2. Baud rate (per la comunicazione seriale




7. Inizializza il pin 13 come output
8. Inizializza la comunicazione seriale
```

```
12.Ascolta i dati in arrivo via seriale
13.Assegna alla variabile rx i dati via seria
14.Se ciò che riceve è diverso da ";"
15.Incrementa la variabile note del valore rx
16.Altrimenti
17.Se la variabile note è diversa da zero
18.La riproduce
19.E reimposta a zero la variabile note
20.Altrimenti
21.Esegue una pausa senza emettere suoni
```

```
24.    if (data != null) {
25.    tweetAlarm();
26.    }
27.}

29.void tweetAlarm() {
30.  TwitterFactory factory = new TwitterFactory();
31.  Twitter twitter = factory.getInstance();
32.  twitter.setOAuthConsumer(C_KEY, C_SECRET);
33.  AccessToken accessToken = new AccessToken(
34.    A_TOKEN,
35.    A_TOKEN_SECRET
36.  );
37.  twitter.setOAuthAccessToken(accessToken);
38.  try {
39.    Status status = twitter.updateStatus(
40.      data
41.    );
42.    println(
43.      "Stato '" + status.getText() + "'."
44.    );
45.  }
46.  catch (TwitterException e) {
47.    e.printStackTrace();
48.  }
49.}
```

```
1. Assegna alla costante BUZZ il numero 13
2. Baud rate (per la comunicazione seriale



7. Inizializza il pin 13 come output
8. Inizializza la comunicazione seriale
```

```
12.Ascolta i dati in arrivo via seriale
13.Assegna alla variabile rx i dati via seriale
14.Se ciò che riceve è diverso da ";"
15.Incrementa la variabile note del valore rx
16.Altrimenti
17.Se la variabile note è diversa da zero
18.La riproduce
19.E reimposta a zero la variabile note
20.Altrimenti
21.Esegue una pausa senza emettere suoni
```

Interfaccia Grafica

Dati ricevuti via seriale

Account twitter **@arduinofurioso**

Pulsante per inviare il tweet

@arduinofurioso

Demo

Grazie per l'attenzione

tancredi.orlando@gmail.com
mario.restuccia@gmail.com

Tancredi Orlando
Mario Restuccia

@turlando
@linkxvi