Corso Arduino 2014

28 Maggio 2014

Riccardo Binetti
punkerbino@gmail.com

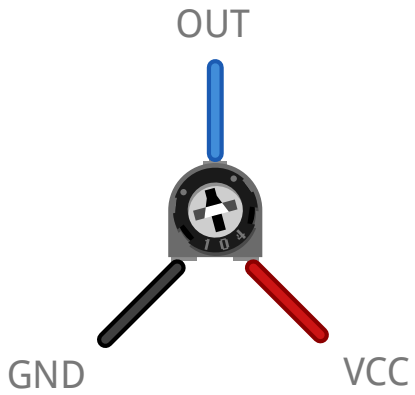
Lezione 2 - Il calore dell'analogico

Segnali analogici

L'altra volta abbiamo lavorato con segnali digitali. Un ingresso digitale può assumere solo 2 valori, HIGH o LOW. Se anche l'ingresso non è esattamente 0v o 5v viene letto uno di questi due valori.
Gli ingressi analogici permettono di leggere segnali che variano in modo continuo tra due estremi (che solitamente su Arduino sono 0 e 5 volt). La lettura non è veramente "continua" ma l'intervallo viene diviso in tanti (in questo caso 1024) piccoli "gradini".

Trimmer

Come possiamo generare un segnale analogico che varia? Ad esempio, con un trimmer (o potenziometro)



Collegato in questo modo, facendolo ruotare da un estremo all'altro OUT varierà continuamente tra GND e VCC.

Leggere i segnali analogici

Come leggiamo ora il segnale che esce dal potenziometro?
`analogRead(int pin);`

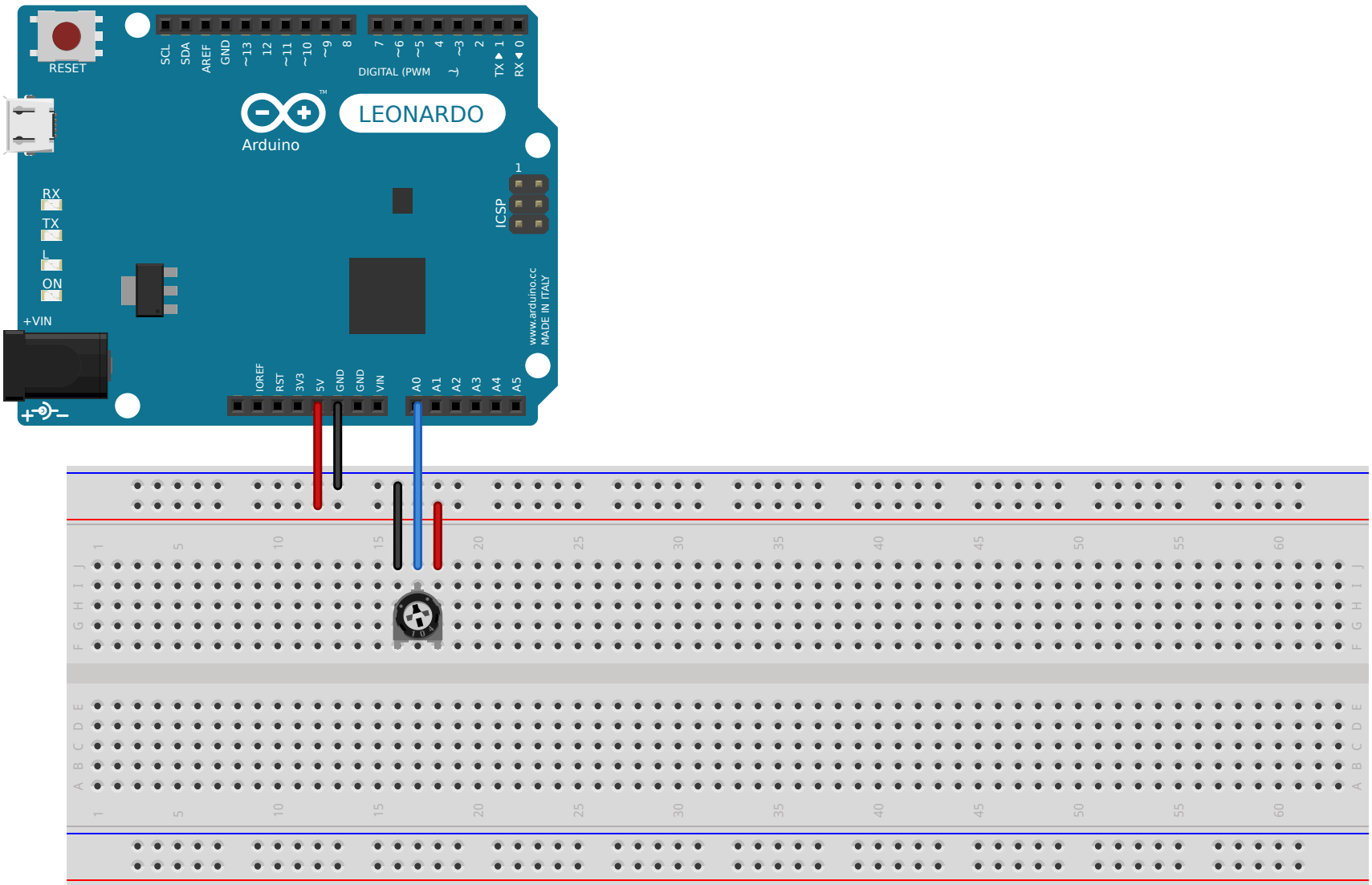
- Legge il `pin analogico con il numero pin`
- Restituisce un numero tra 0 e 1023 (va salvato in una variabile: `int val = analogRead(pinPot);`)
- Il pin va indicato con il numero corrispondente ai pin analogici, senza la A
- I pin analogici vanno da A0 ad A5 su Arduino Leonardo

Tocca a voi

Realizzate un circuito che faccia blinkare il LED ad un intervallo compreso tra 100 e 1000 ms

Go!

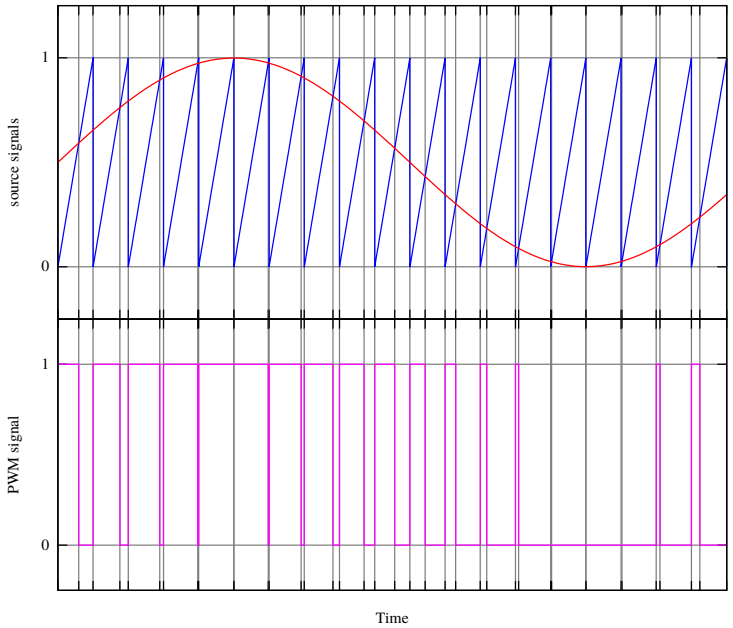
Circuito potenziometro



Soluzione

```
int ledPin = 13;
int buttonPin = 6;
int potPin = 4;
int interval = 1000;
void setup(){
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
}
void loop(){
  interval = 100 + analogRead(potPin) * 1000.0/1024.0;
  digitalWrite(ledPin, HIGH);
  delay(interval);
  digitalWrite(ledPin, LOW);
  delay(interval);
}
```

PWM



- La PWM (Pulse Width Modulation) serve a fare emettere un'approssimazione di un voltaggio analogico, usando un voltaggio digitale
- La funzione che fa questo su Arduino è analogWrite

analogWrite

- Sintassi: `analogWrite(pin, value)`
- `pin` è il numero del pin che si vuole usare, si posssono usare solo i pin con il simbolo ~ sulla board
- `value` è un numero tra 0 e 255
- Quando value è 0, l'uscita sarà sempre LOW, quando value è 255, l'uscita sarà sempre HIGH
- La frequenza dell'onda quadra è di solito 490Hz, su Leonardo i pin 3 e 11 vanno a 980Hz

Ciclo for

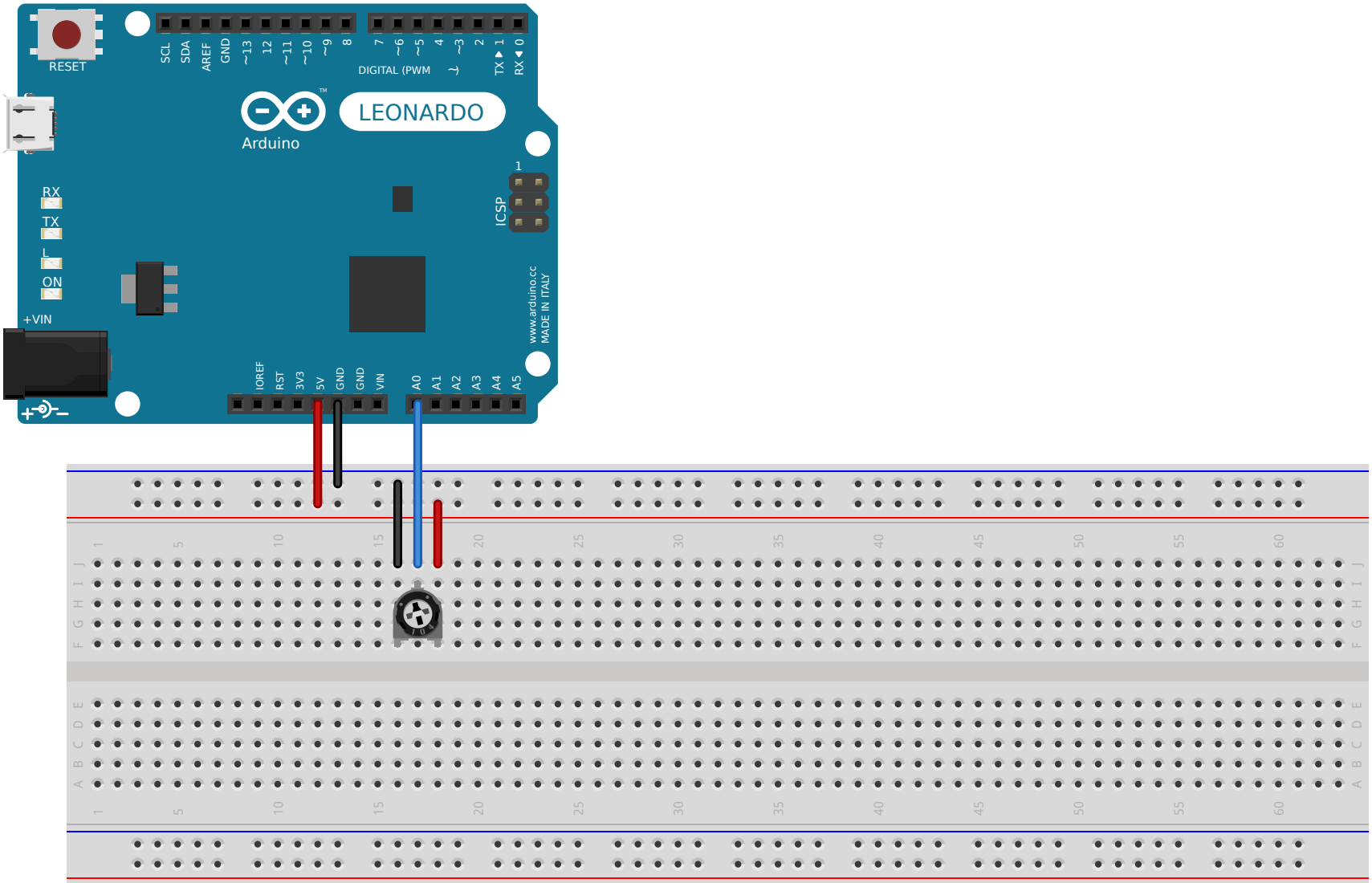
- Sintassi: `for (int i=0; i<255; i++){//codice che usa la variabile i}`
- `int i=0` definisce una variabile chiamata i e gli assegna il valore 0
- `i < 255` è la condizione del ciclo, finché è vera il ciclo for continua
- `i++` è la condizione da eseguire ogni volta che finisce il ciclo e lo si rinizia

Fading LED

Realizzate un programma che fa accendere e spegnere gradualmente un LED (usate il LED interno).
Bonus: controllate la velocità di fading con il potenziometro

Go!

Circuito Fading LED



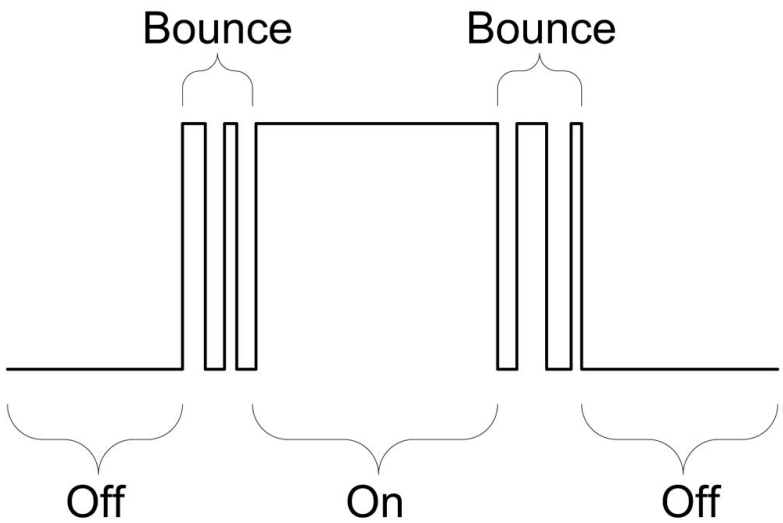
Soluzione

```
int ledPin = 13;
void setup()  {
  pinMode(ledPin, OUTPUT);
}
void loop()  {
  for(int fadeValue = 0 ;fadeValue <= 255; fadeValue +=5)
  analogWrite(ledPin, fadeValue);
  delay(30);
}
  for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5)
    analogWrite(ledPin, fadeValue);
    delay(30);
  }
}
```

Problemi con i bottoni

L'altra volta avevamo fatto accendere un LED mentre si teneva premuto il pulsante. Farlo accendere e spegnere premendo il pulsante non è facile come sembra.
Perchè?

Bouncing

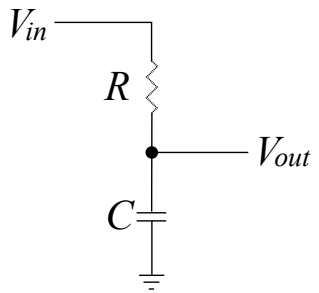


Quando il bottone viene premuto, in realtà gli switch meccanici fanno un po' di falsi contatti prima di stabilizzarsi.

Debouncing

Come si può risolvere questo problema?

- Via software (bloccante): quando rileviamo lo stato che vogliamo, inseriamo un delay successivo per evitare i contatti dopo (5ms dovrebbero bastare)
- Via software (non bloccante): ad ogni giro del loop, se il bottone è nello stato "attivo" incrementiamo un contatore, se è nello stato "a riposo" lo mettiamo a zero. Impostiamo una soglia a quel contatore oltre cui il bottone è considerato premuto
- Via hardware: con un filtro passa basso (con R=100kΩ e C = 47nF, τ = R×C = 4.7ms)



Timing

- Oltre alla funzione `delay`, ci sono altre funzioni per la gestione del tempo utili su Arduino
- `delayMicroseconds(nMicros)`: come la delay ma il delay è in microsecondi
  - Ha una risoluzione di circa 3 microsecondi
- `millis()`: ritorna il valore di millisecondi passati da quando la scheda è stata accesa
  - Il numero va in overflow dopo circa 50 giorni
- `micros()`: ritorna il valore di microsecondi passati da quando la scheda è stata accesa
  - Il numero va in overflow dopo circa 70 minuti

In particolare la funzione millis(), può essere utile a fare programmi come il blinking LED che però non "sprecano tempo" con la `delay()`
(Provate a pensare come)

Polling vs Interrupt

Tutti i programmi che abbiamo scritto finora rilevavano gli input tramite polling. In pratica, il programma ad ogni passaggio chiedeva al pin "In che stato sei?" e agiva di conseguenza. Alcuni input digitali permettono di agire in modo diverso, tramite interrupt. In pratica, quando c'è un determinato cambiamento di stato su un pin, viene eseguita una funzione. Gli interrupt sono una feature abbastanza avanzata e non sempre sono il modo migliore per gestire gli input digitali. In ogni caso...

attachInterrupt

- attachInterrupt(interrupt, ISR, mode)
- interrupt: il numero dell'interrupt che si vuole abilitare. Il numero di interrupt non corrisponde al numero di pin, usate questa tabella

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	

- ISR: il nome della funzione da eseguire. La funzione non deve ricevere nessun parametro e non deve ritornare niente.
- mode: il cambiamento di stato su cui deve essere attivato l'interrupt
  - CHANGE: l'interrupt viene attivato ogni volta che lo stato passa da HIGH a LOW o viceversa
  - RISING: l'interrupt viene attivato ogni volta che lo stato passa da LOW a HIGH
  - FALLING: l'interrupt viene attivato ogni volta che lo stato passa da HIGH a LOW
  - LOW: l'interrupt viene attivato ogni volta che lo stato è LOW

Cose a cui stare attenti

- L'ISR ha priorità sulla funzione loop
  - Quindi se mettete come mode LOW, verrà continuamente eseguita l'ISR e non il loop
- Durante l'esecuzione dell'ISR, il contatore millis() non si incrementa
- Le variabili che devono essere modificate all'interno dell'ISR devono avere il modificatore `volatile`

Esempio Interrupt

```
int pin = 13;
volatile int state = LOW;
void setup() {
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, CHANGE);
}
void loop() {
  digitalWrite(pin, state);
}
void blink() {
  state = !state;
}
```

Seriale

Ultimo argomento di oggi: la seriale. Ci servirà nella prossima per comunicare con il pc.

- Serial.begin(9600): da inserire nella setup, inizia la comunicazione seriale a baud (velocità) 9600bps
- Serial.println("Hello POuL!"): scrive la string "Hello POuL!" sulla seriale e va a capo
- Serial.print("Hello POuL!"): uguale ma non va a capo alla fine
- Su Arduino UNO la seriale viene scritta sia sulla seriale virtuale (quella collegata al PC) che su quella hardware (collegata ai pin 0 e 1 della board)
- Su Arduino Leonardo, la classe Serial scrive solo sul pc. Per scrivere sulla porta hardware bisogna usare Serial1.
- Oltre a stringhe costanti, ovviamente, si possono stampare variabili (utile per debugging)

Monitor seriale

Per leggere quello che viene scritto dalla seriale, potete usare il monitor seriale dell'Arduino IDE.

- Strumenti $\rightarrow$ Porta Seriale e selezionate la porta seriale giusta
- Strumenti $\rightarrow$ Monitor Seriale e selezionate in basso a destra il baud giusto (default 9600)

Esempio seriale

Scrivete un programma che stampa ogni due secondi “Hello ” seguito da un numero che incrementa ogni volta che viene stampato e va a capo.
Bonus: fate due stampe, una ogni 2 secondi e una ogni 3, usando la millis()

Soluzione

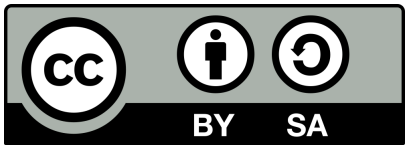
```
int val = 0;
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Hello ");
  Serial.println(val);
  val++;
  delay(1000);
}
```

Domande?

Se vi vengono in mente più tardi, fatele sul gruppo

Per oggi è tutto

Ci vediamo settimana prossima



`http://www.poul.org`