ARI Sezione di Ivrea Serate tecniche 2012

ARDUINO il corso
Lezione 4
I sensori

IZ1MHN e IW1ALX

ARI Sezione di Ivrea Serate tecniche 2012

Prima di iniziare… (che novità!)

- Domande sulla lezione scorsa? (e chi si ricorda!)
- Come è andato il compito a casa? (avete fatto esercizi, vero???)

I sensori

- Movimento
- Luce
- Temperatura
- Distanza
- Rotazione
- Posizione (GPS)
- Accelerazione

I Sensori

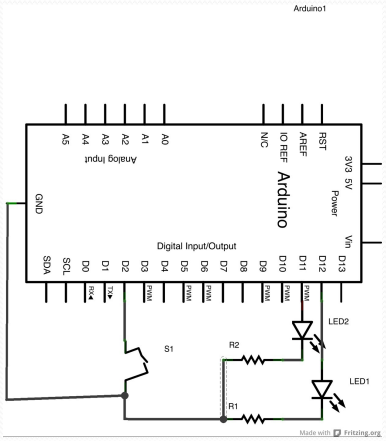
- Consentono ad Arduino di esplorare il mondo esterno
- Possono presentarsi come shield o come singolo componente
- Per molti sensori esistono delle comodissime librerie!
- Possono dialogare in
  - Analogico (una tensione. Attenzione NON sempre sono lineari!)
  - Digitale (on/off)
  - PWM (si misura la durata dell'impulso con la funzione `pulseIn`)
  - Seriale o protocolli sincroni (es 1 Wire, I2C, SPI…)

Movimento

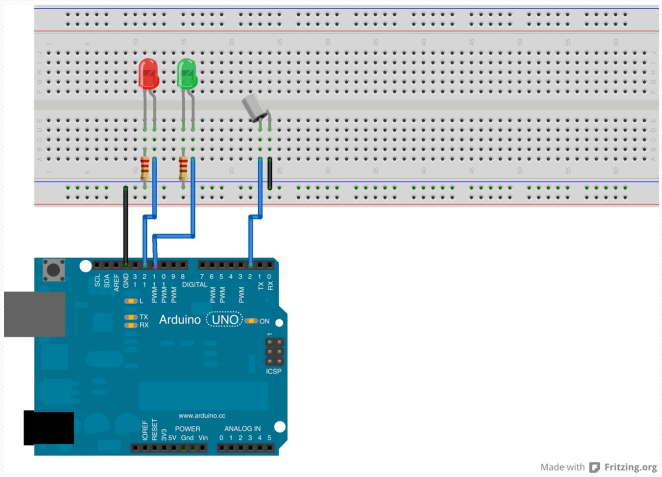
- Si usano i “tilt sensor” eredi non inquinanti dei vecchi interrutori a mercurio.
- Sono “digitali” come output: acceso o spento!

Movimento: schema

R1 = R2 = 220 Ohm



Movimento: circuito



Movimento: sketch 1/2

```
const int tiltSensorPin = 2;
const int LedUno = 11;
const int LedDue = 12;

void setup() {
  pinMode (tiltSensorPin, INPUT);
  digitalWrite (tiltSensorPin, HIGH);
  //vi ricordate... vero?

  pinMode (LedUno, OUTPUT);
  pinMode (LedDue, OUTPUT);
}
```

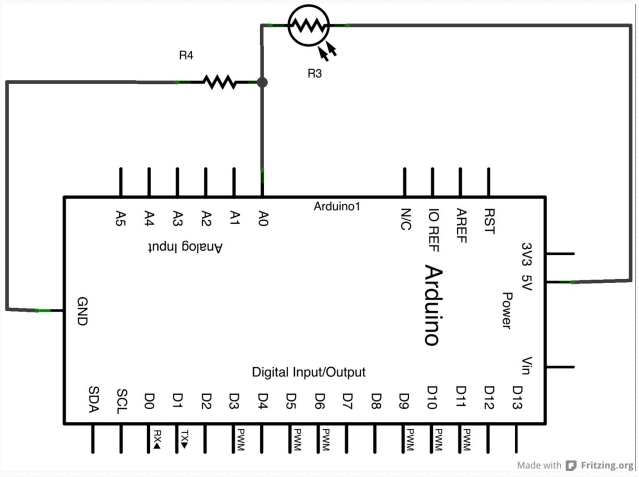
Movimento: sketch 2/2

```
void loop()
{
  if (digitalRead(tiltSensorPin))
  {
    digitalWrite(LedUno, HIGH);
    digitalWrite(LedDue, LOW);
  }
  else{
    digitalWrite(LedUno, LOW);
    digitalWrite(LedDue, HIGH);
  }
}
```

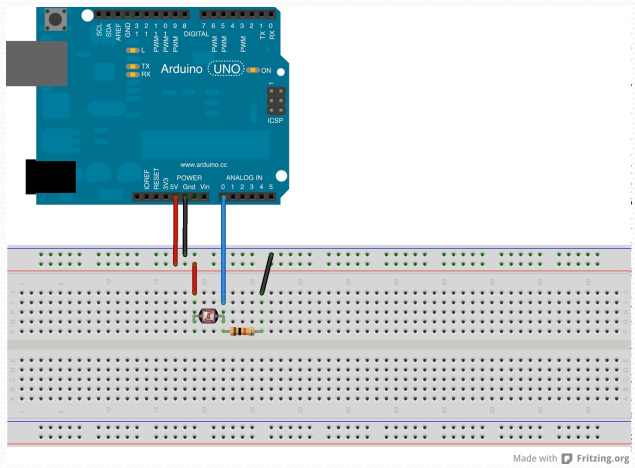

Luce

- Si può operare in due modi diversi:
  - Fotoresistenza o LDR (Light Dependant Resistor)
  - Fototransistor (noi useremo un BPW96C)
- Sono “analogici”

Luce con fotoresistenza: schema



Luce con fotoresistenza: circuito

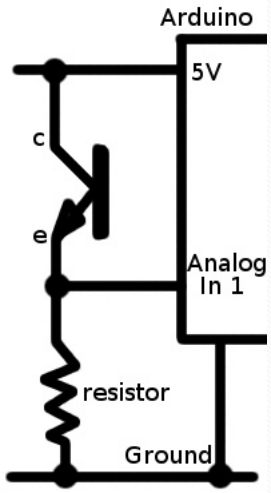


Luce con fotoresistenza: sketch

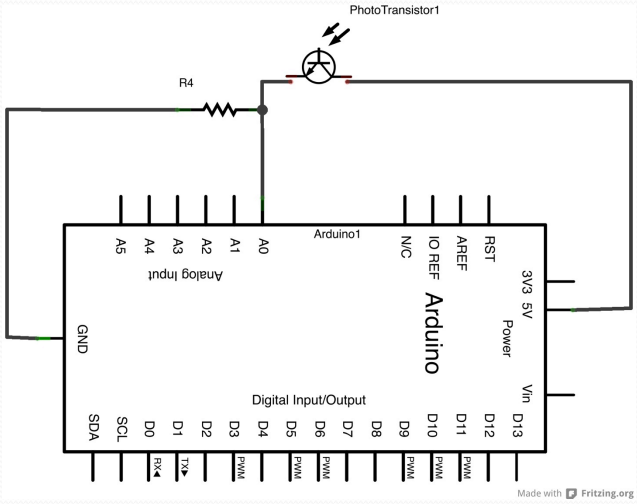
```
// accendo e spengo un LED con secondo la luminosità

const int ledPin = 13;
const int sensorPin = 0;
void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  int rate = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);
  delay(rate);
  digitalWrite(ledPin, LOW);
  delay(rate);
}
```


Luce con fototransistor

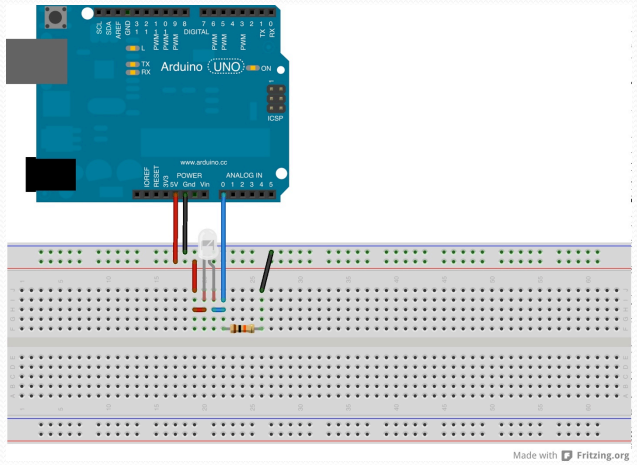
- Rimane tutto invariato!
- BPW96C



Luce con fototransistor: shema



Luce con fototransistor: circuito



Luce con fototransistor: sketch

```
const int ledPin = 13; // uso il solito led 13
const int sensorPin = 0; // uso analog 0 per il sensore

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600); // solo per aggiungere qualcosa
}

void loop()
{
  int rate = analogRead(sensorPin);
  Serial.print(rate);
  Serial.print(" ");
  digitalWrite(ledPin, HIGH);
  delay(rate);
  digitalWrite(ledPin, LOW);
  delay(rate);
}
```

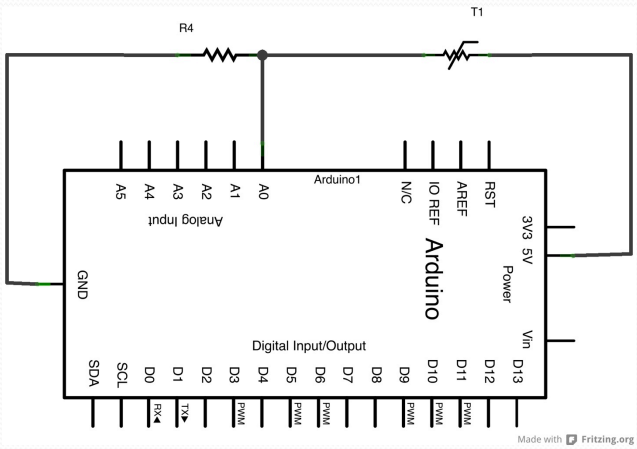

Temperatura

- Termistori NTC (negative temperature coefficient) o (meno usati) PTC (positive temperature coefficient)
- LM35, LM335, TMP36…
- DTSC (Digital Temperature Sensor Chip): DS18B20
- Termocoppie

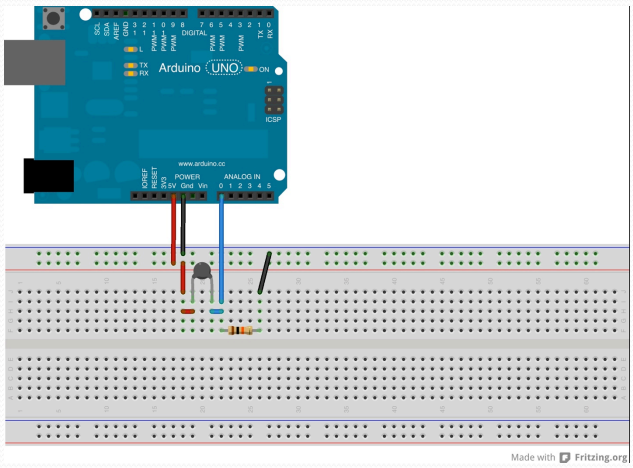
Termistori

- Un termistore è una restenza il cui valore varia con il variare della temperatura.
- Tutti le resistenze subiscono questa variazione, ma I termistori sono costruiti per amplificarla.
- Rispetto alle altre opzioni presentano alcuni vantaggi:
  - Costo!!!
  - Sono indipendenti dalla tensione.
  - Non richiedono amplificatori.
  - Accurati in rapporto al prezzo!
  - Robusti!
  - Possono essere facilmente resi impermeabili!

Termistori: schema



Termistori: circuito



Termistori: un pò di…

- Dallo schema… si vede che usiamo in pratica il solito partitore
- Supponiamo che la resistenza fissa sia da 10K e battezziamo il termistore R (che fantasia, vero?!)
  - Vo = R / (R + 10K) * Vcc
  - ADC = Vi * 1023/ Vcc
  - Ma Vo = Vi, quindi ADC = R / (R + 10K) * Vcc * 1023 / Vcc
  - Quindi: ADC = R / (R + 10K) * 1023 (e Vcc è sparita!)
  - Alla fine: R = 10K / ((1023 / ADC)-1)

In bella scrittura!

$$Vo = \frac{R}{R+10K} * Vcc$$

$$ADC = Vi * \frac{1023}{Vcc}$$

$$Vo = Vi$$

$$ADC = \frac{R}{R+10K} * Vcc * \frac{1023}{Vcc}$$

$$ADC = \frac{R}{R+10K} * 1023$$

$$R = \frac{10K}{\frac{1023}{ADC} - 1}$$

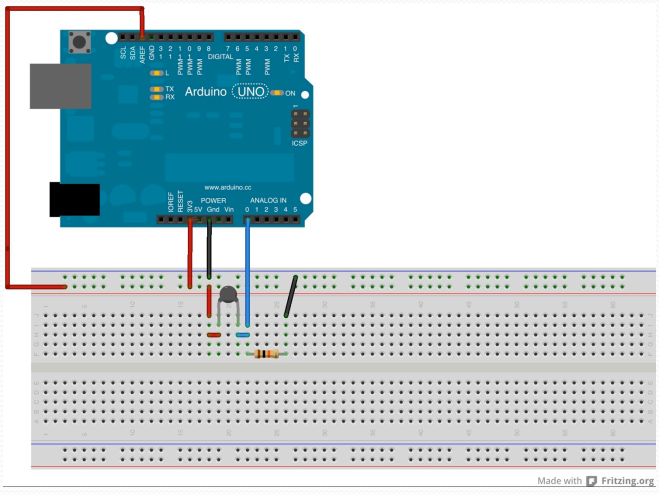
Termistore: sketch V1

```
#define Partitore 10000
#define PinTermistore A0
 void setup() {
  Serial.begin(9600);
}

void loop() {
  float Lettura;
  float Termistore;
  Lettura = analogRead(PinTermistore);
  Serial.print("Valore Lettura ");
  Serial.println(Lettura);
  Lettura = (1023 / Lettura)  - 1;
  Termistore = Partitore / Lettura;
  Serial.print("Resistenza del Termistore ");
  Serial.println(Termistore);
  delay(1000);
}
```

Aumentiamo la precisione

- Usiamo due barbatrucchi:
  - Usiamo il pin 3.3V come tensione di riferimento
    - La 5V arriva paro paro dalla porta USB del PC…
    - Connettiamo il pin 3.3V con lil pin AREF
    - Vcc non entrava nell'equazione, quindi …
    - Dobbiamo ricordarci il comando
      - `analogReference (ESTERNAL)`
  - Facciamo una media di letture invece di una lettura one shot!
    - Eliminiamo fluttuazioni ecc ecc

Termistore V2: circuito



Termistore V2: sketch 1/3

```
#define Partitore 10000
#define PinTermistore A0
#define NumeroCampioni 5

int Campioni[NumeroCampioni];

void setup() {
  Serial.begin(9600);
  analogReference(EXTERNAL);
}
void loop() {
  int i;
  float Media;
  float Termistore;
```


Termistore V2: sketch 2/3

```
  for (i=0; i< NumeroCampioni; i++)
  {
    Campioni[i] = analogRead(PinTermistore);
    delay(10);
  }
  Media = 0;
  for (i=0; i< NumeroCampioni; i++)
  {
    Media += Campioni[i];
  }
  Media /= NumeroCampioni;
```

Termistore V2: sketch 3/3

```
  Serial.print("Valore Medio Lettura ");
  Serial.println(Media);

  Media = (1023 / Media)  - 1;
  Termistore = Partitore / Media;

  Serial.print("Resistenza del Termistore ");
  Serial.println(Termistore);
  delay(1000);
}
```


Termistori… e mo?

- Ok adesso abbiamo misurato una resistenza.
- Ma noi vorremmo una tensione
- Due strade:
  - Convertiamo usando la tabella delle specifiche (magari caricata in EEPROM)
  - Usiamo un pizzico di fisica e ci accontentiamo di alcune “tolleranze”

Termistori e fisica

- Usiamo l'equazione di Steinhart-Hart

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3$$

- Nella versione semplificata diventa

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right)$$

- Dove (i valori sono in K)
  - To = temperatura ambiente
  - B = coefficiente del termistore
  - Ro = resistenza a temperatura ambiente
  - R = resistenza misurata

Termistori e fisica: lo sketch 1/4

```
#define PinTermistore A0
#define ResistenzaNominale 10000     // R alla T Nominale
#define TemperaturaNominale 25       // Temperatura x la R
nominale
#define NumeroCampioni 5
#define CoefficienteB 3950           // Coefficiente B
#define ResistenzaSerire 10000       // in Ohm

int Campioni[NumeroCampioni];

void setup() {
  Serial.begin(9600);
  analogReference(EXTERNAL);
}
```

Termistori e fisica: lo sketch 2/4

```
void loop() {
  uint8_t i;    //risparmiamo qualcosa
  float Media;

  for (i=0; i< NumeroCampioni; i++) {
   Campioni[i] = analogRead(PinTermistore);
   delay(10);
  }

  Media = 0;
  for (i=0; i< NumeroCampioni; i++) {
     Media += Campioni[i];
  }
  Media /= NumeroCampioni;
```


Termistori e fisica: lo sketch 3/4

```
Serial.print("Lettura Media ");
 Serial.println(Media);

 Media = 1023 / Media - 1;
 Media = ResistenzaSerire / Media;
 Serial.print("Resistenza Termistore ");
 Serial.println(Media);
```

Termistori e fisica: lo sketch 4/4

```
 float steinhart;
  steinhart = Media / ResistenzaNominale;      // (R/Ro)
  steinhart = log(steinhart);                  // ln(R/Ro)
  steinhart /= CoefficienteB;      // 1/B * ln(R/Ro)
  steinhart += 1.0 / (TemperaturaNominale + 273.15);
                                   // + (1/To)
  steinhart = 1.0 / steinhart;     // Inverto
  steinhart -= 273.15;             // converto da K a C

  Serial.print("Temperatura ");
  Serial.print(steinhart);
  Serial.println(" *C");
  delay(1000);
}
```

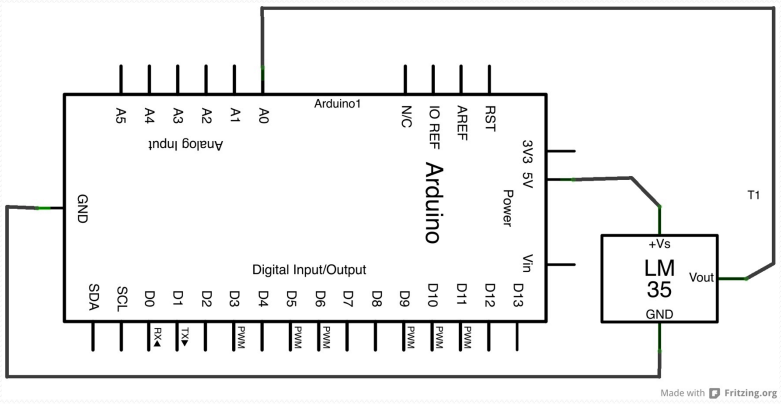

Termistori fisica e precisione

- Più o meno si può dire che:
  - Resistenza e termistore con tolleranza 1%, errore 0.5 °C
  - Resistenza e termistore con tolleranza 5%, errore 1 °C

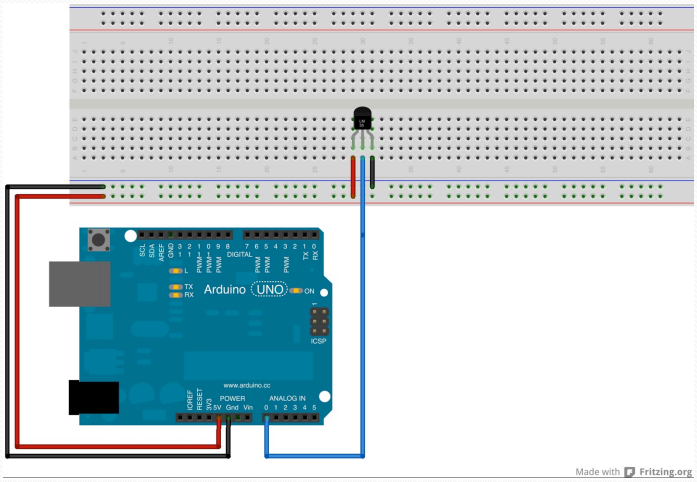
Temperatura e LM35

- È un sensore che produce una tensione analogica proporziaonale alla temperatura.
- La variazione è di 1 mV ogni 0.1 °C
- La precisione è di circa 0.5 °C

LM35: schema



LM35: circuito

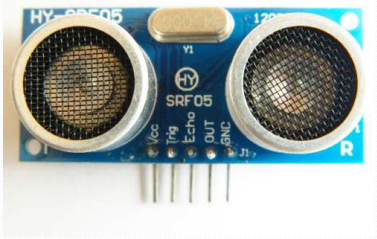


LM35: sketch

```
const int PinLM35 = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int value = analogRead(PinLM35);
  Serial.print(value);
  Serial.print(" > ");
  float millivolts = (value / 1024.0) * 5000;
  float celsius = millivolts / 10;
  Serial.print(celsius);
  Serial.print(" gradi Celsius, ");
  delay(1000);
}
```

Distanza

- si possono usare due tipi di sensori
  - Ultrasuoni
  - Infrarosso (sotto i 2 mt, ma più preciso)
- Sono sensori “PWM”



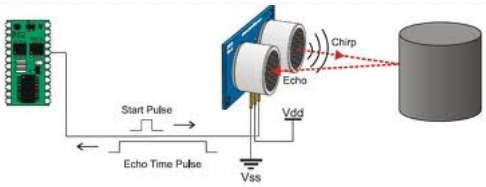
PING

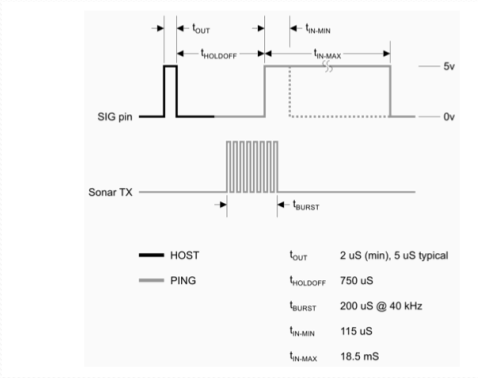
- I sensori a ultrasuoni sono anche soprannominati PING
- In pratica si invia un impulso e si misura il tempo che ci mette per andare e tornare.
- Si usa la funzione `pulseIn` per leggere la durata del PWM restituito dal sensore
- Due famiglie:
  - 3 pin: trigger e echo comuni
  - 4 (o 5) pin: trigger e echo separati

PING 3 Pin: es Parallax Ping)))

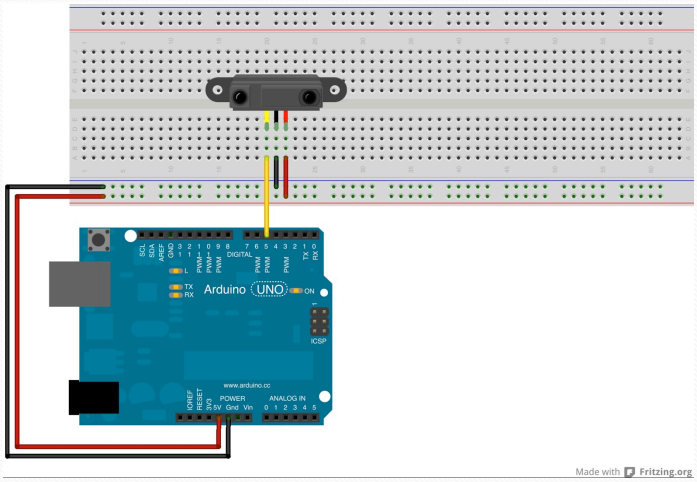
- Features
  - Supply Voltage – 5 VDC
  - Supply Current – 30 mA typ; 35 mA max
  - Range–2cmto3m(0.8into3.3yrds)
  - Input Trigger – positive TTL pulse, 2 uS min, 5 µs typ.
  - Echo Pulse – positive TTL pulse, 115 uS to 18.5 ms
  - Echo Hold-off – 750 µs from fall of Trigger pulse
  - Burst Frequency – 40 kHz for 200 µs
  - Burst Indicator LED shows sensor activity
  - Delay before next measurement – 200 µs
  - Size–22mmHx46mmWx16mmD(0.84inx1.8inx0.6in)

PING 3 Pin





PING 3 Pin: cricuito



PING 3 Pin: note

- Il trigger è un impulso da almeno 2 microsecondi
- Mandare prima il pin a LOW per essere sicuri il trigger sia "pulito"
- Riportare il pin basso prima di aspettare la risposta

```
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);   //pulisce
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);  //triggera
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
pinMode(pingPin, INPUT);      //aspetta la risposta
```

PING 3 Pin: note

- La risposta è un PIN che rimane alto tanti microsecondi quanto è il tempo per andare e tornare dal bersaglio
- La velocità del suono la approssimiamo a 340 m/s che sono 29 microsecondi al centimetro.
- Il tempo va dimezzato, perchè misuriamo andata e ritorno!

```
pinMode(pingPin, INPUT);
duration = pulseIn(pingPin, HIGH);
cm = duration / 29 / 2;
```


PING 3 PIN: sketch 1/2

```
const int pingPin = 5;

void setup() {
  Serial.begin(9600);
}

long microsecondi2cm(long microsecondi)
{
return microsecondi / 29 / 2;
}
```

PING 3 PIN: sketch 2/2

```
void loop()
{
  long durata, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  durata = pulseIn(pingPin, HIGH);
  cm = microsecondi2cm(duration);
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);
}
```


PING 4 PIN: es SRF04 e SRF05

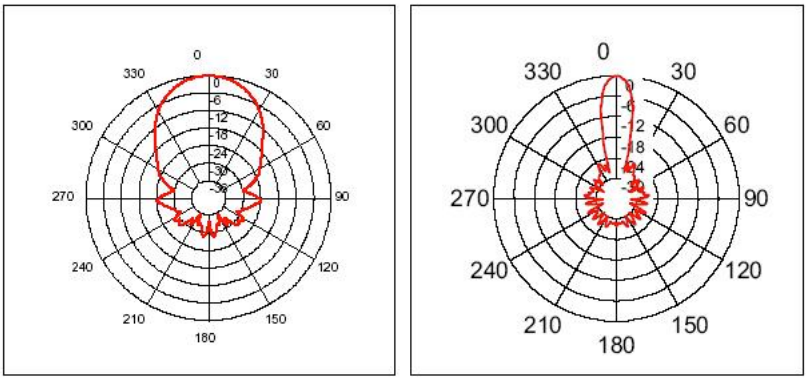
Sensor	Communication	Range		Angle*	Echoes**	Ranging Time	Notes
		Minimum	Maximum				
SRF01	Serial	0 cm	6 m	45°	One	70 ms	A
SRF02	I2C / Serial	15 cm	6 m	45°	One	70 ms	A
SRF04	Digital	3 cm	3 m	45°	One	100 µs - 36 ms	
SRF05	Digital	3 cm	4 m	45°	One	100 µs - 36 ms	
SRF06	Analog	2 cm	5 m	55°	One	70ms - 100 ms	
SRF08	I2C	3 cm	6 m	45°	17	65 ms	B C
SRF10	I2C	3 cm	6 m	60°	One	65 ms	A B
SRF235	I2C	10 cm	1 m	15°	One	10 ms	A D
SRF485	RS485 Serial	30 cm	5 m	45°	One	70 ms	A
SRF485WPR	RS485 Serial	60 cm	5 m	30°	One	70 ms	A

*: Approximate angle of the sensor cone at 1/2 sensor range (see diagram above).
**: The number of echoes recorded by the sensor. These are the recorded echoes from the most recent reading, and are overwritten with each new ranging.
A: These sensors are smaller than the typical (SRF 04 / 05 / 08) size.
B: Range time can be adjusted down by adjusting the gain.
C: This sensor also includes a photocell on the front for light detection.
D: Operates at a higher 235kHz frequency. See the SRF235 Technical Specifications page for how this affects sensor performance.

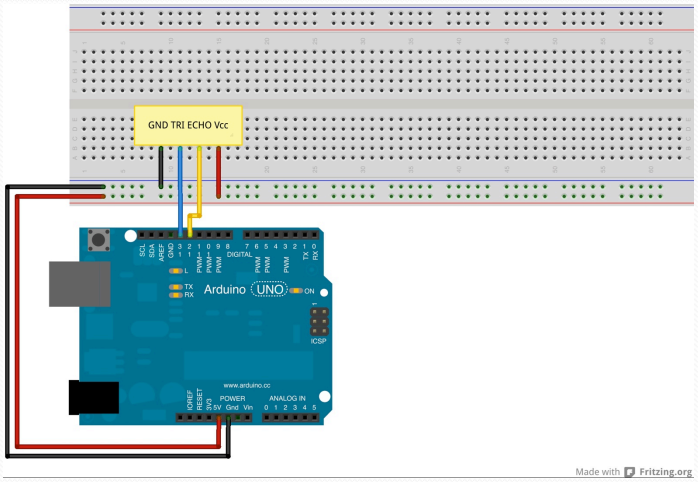
Revision History:

- 2011-04-29: Added new SRF modules to the comparison table
- 2007-11-05: Page created.

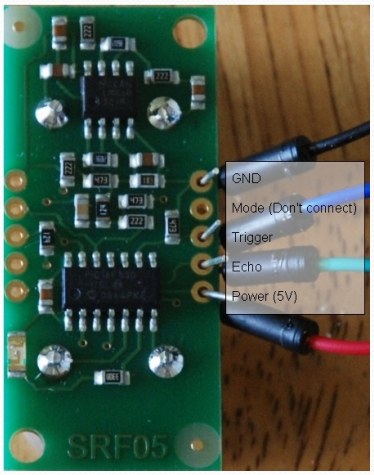
PING SRF04 e SRF05



PING 4 Pin: circuito



PING 4 Pin: collegamenti



PING 4 Pin: sketch 1/2

```
#define trigPin 13
#define echoPin 12

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

PING 4 Pin: sketch 2/2

```
void loop() {
  long durata, distanza;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  durata = pulseIn(echoPin, HIGH);
  distanza = (durata/58);
  Serial.print(distanza);
  Serial.println(" cm");
  delay(500);
}
```

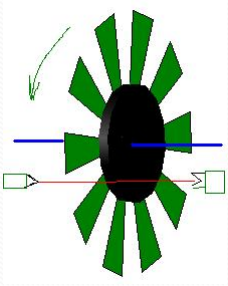

Encoder rotativo

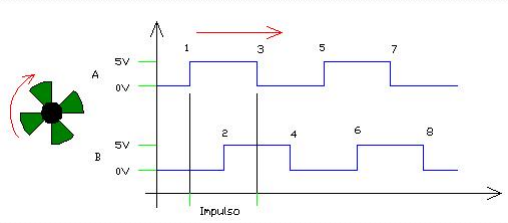
- Un Encoder rotativo è un dispositivo che produce due segnali quando viene ruotato. I due segnali:
  - I due segnali cambiano da alto a basso un determinato numero di volte per giro (passo).
  - I due segnali sono leggermente sfasati (90°) tra loro.
- Se si traccia il momento in cui i segnali cambiano di stato tra alto e basso, si può determinare sia la rotazione sia la direzione

Come funziona un Encoder

- In modo molto semplice…
  - All'interno c'è un disco calettato su un albero.
  - Il disco ha aree opache e aree trasparenti.
  - Parallelo all'albero c'è una barriera a fotoqualcosa (diodi, resistenze…).
  - Ogni volta che un'area opaca ruotando passa sopra alla barriera la interrompe (e questo basta per avere la rotazione, ma non per la direzione).
  - Se mettiamo due barriere, sfalsate tra loro, abbiamo anche la direzione di rotazione.

Come funziona un Encoder

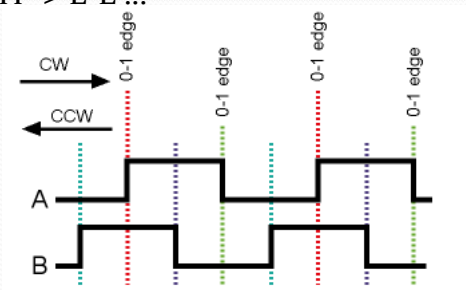




http://recemi.xoom.it/recemi/Encoder.htm

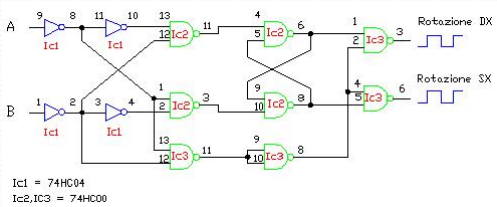
Come funziona un Encoder

- Se guardiamo lo schema sotto…
  - Se giriamo in senso orario avremo sulle tracce A e B rispettivamente L-L -> L-H -> H-H -> H-L…
  - Se giriamo in senso antiorario invece vedremo (sempre su A e B) H-L-> H-H -> L-H -> L-L …



Come funziona un Encoder

- A noi verrebbe comodo avere due segnali separati dove leggere la rotazione verso DX e quella verso SX.
- Un pò di logiche…



Tipi di Encoder Rotativi

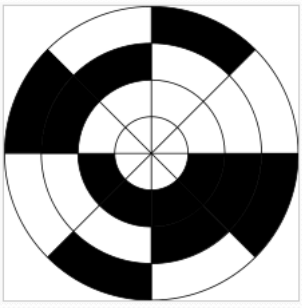
- Gli encoder si dividono in due famiglie: relativi o assoluti.
  - Relativi: esprimono solamente la rotazione e la direzione
  - Assoluti: esprimono anche la posizione in riferimento ad uno zero.
    - Il "disco" non ha più semplicemente dei settori, ma viene suddiviso anche in cilindri.
    - La combinazione di tracce e cilindri da la risoluzione dell'encoder
    - possono lavorare con due codifiche: binaria o Gray

Tipi di encoder

- Codifica Binaria Standard

Standard Binary Encoding

Sector	Contact 1	Contact 2	Contact 3	Angle
0	off	off	off	0° to 45°
1	off	off	ON	45° to 90°
2	off	ON	off	90° to 135°
3	off	ON	ON	135° to 180°
4	ON	off	off	180° to 225°
5	ON	off	ON	225° to 270°
6	ON	ON	off	270° to 315°
7	ON	ON	ON	315° to 360°

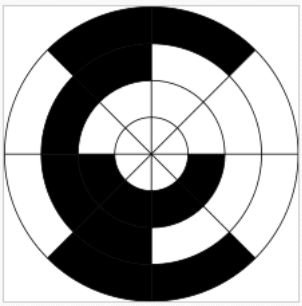


Tipi di encoder

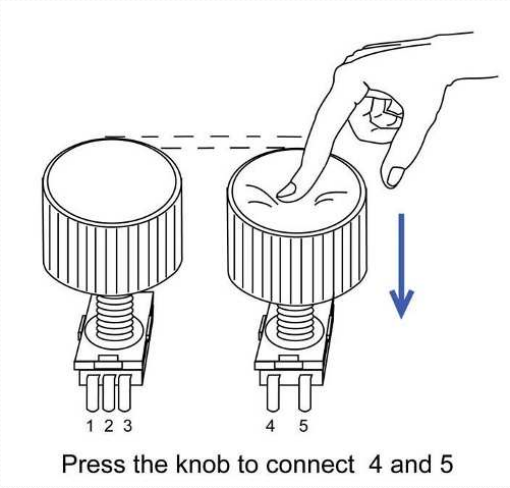
- Codifica Gray

Gray Coding

Sector	Contact 1	Contact 2	Contact 3	Angle
0	off	off	off	0° to 45°
1	off	off	ON	45° to 90°
2	off	ON	ON	90° to 135°
3	off	ON	off	135° to 180°
4	ON	ON	off	180° to 225°
5	ON	ON	ON	225° to 270°
6	ON	off	ON	270° to 315°
7	ON	off	off	315° to 360°

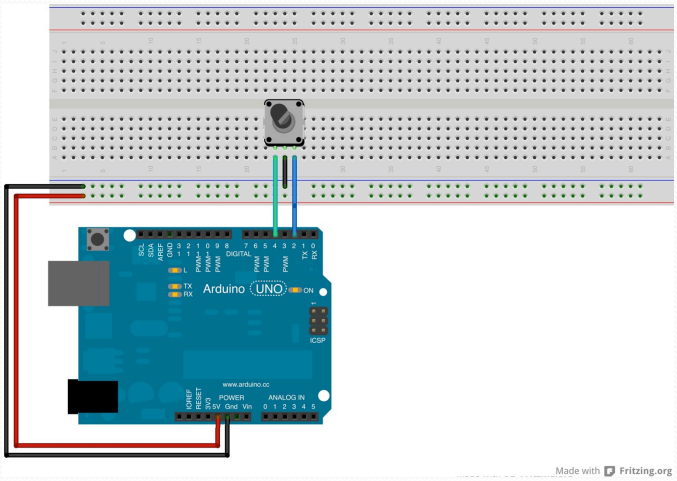


Il nostro Encoder

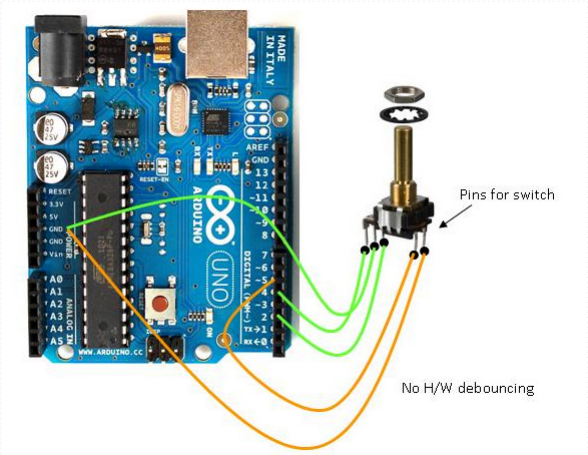


Press the knob to connect 4 and 5

Encoder: il circuito



Encoder: i collegamenti



Encoder: lo sketch

```
int val;
int encoder0PinA = 2;
int encoder0PinB = 4;
int encoder0Pos = 0;
int encoder0PinALast = LOW;
int n = LOW;

void setup() {
  pinMode (encoder0PinA,INPUT_PULLUP);
  pinMode (encoder0PinB,INPUT_PULLUP);
  Serial.begin (9600);
}
```

ARI Sezione di Ivrea Serate tecniche 2012

Encoder: lo sketch

```
void loop() {
  n = digitalRead(encoder0PinA);
  if ((encoder0PinALast == LOW) && (n == HIGH)) {
    if (digitalRead(encoder0PinB) == LOW) {
      encoder0Pos--;
    } else {
      encoder0Pos++;
    }
    Serial.print (encoder0Pos);
    Serial.print (" | ");
  }
  encoder0PinALast = n;
}
```

Meantime… Interrupt

- Un interrupt è un segnale asincrono che indica alla CPU che una periferica ha bisogno di "cure".
- Evita di "bloccare" un programma in attesa di dati.
- Su Arduino il numero di interrupt che si possono sfruttare dipendono dalle scheda:
  - Arduino UNO: 2 interrupt
  - Arduino Mega: 6 interrupt
  - Arduino DUE: tutti i pin!!!

Interrupt con UNO r3

- Al pin 2 corrisponde interrupt 0
- Al pin 3 corrisponde interrupt 1
- Per segnalare ad Arduino che vogliamo usare un intterrupt si usa la funzione `attachInterrupt`
- la funzione `attachInterrupt` chiede tre parametri:
  - Interrupt (0 1),
  - routineRichiamata,
  - Evento

Interrupt con UNO r3

- Il parametro Evento ci dice in quale situazione si scatena l'interrupt.
- Può assumere 4 valori:
  - LOW triggera quando il pin è basso,
  - CHANGE triggera i cambiamenti di stato,
  - RISING triggera solo la transizione basso -> alto
  - FALLING: triggera solo la transizione alto -> basso

Interrupt con MEGA

- Al pin 2 corrisponde interrupt 0
- Al pin 3 corrisponde interrupt 1
- Al pin 21 corrisponde interrupt 2
- Al pin 20 corrisponde interrupt 3
- Al pin 19 corrisponde interrupt 4
- Al pin 18 corrisponde interrupt 5

Tanto per confondere i russi ;-)

Interrupt con DUE

- Cambiano i parametri della attachInterrupt che diventano:
  - pin,
  - routineRichiamata,
  - Evento
- Viene introdotto l'Evento HIGH che triggera tutte le volte che il Pin è alto

"Sganciare" un interrupt

- Se fosse necessario "Spegnere" un interrupt, si può usare la funzione `detachInterrupt`.
- Accetta come parametro il numero di Interrupt o nel caso della DUE il Pin

Interrupt: un esempio

```
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;
}
```


Encoder e Interrupt: lo sketch

Come abbiamo visto, la funzione attachInterrupt chiede tre parametri: interrupt, routineRichiamata, evento

Nel nostro esempio:

- pin=3 quindi interrupt=1
- funzione doEncoder
- evento: al CHANGE

Encoder e Interrupt: lo sketch 1

```
#define encoder0PinA  3
#define encoder0PinB  4

int led = 13;

volatile unsigned int encoder0Pos = 0;
int encoder0PinALast = LOW;
int n=LOW;
```


Encoder e Interrupt: lo sketch 2

```
void setup() {

  pinMode(led, OUTPUT);

  pinMode(encoder0PinA, INPUT_PULLUP);
  pinMode(encoder0PinB, INPUT_PULLUP);

  attachInterrupt(1, doEncoder, CHANGE);
  Serial.begin (9600);
  Serial.println("start");

}
```

Encoder e Interrupt: lo sketch 3

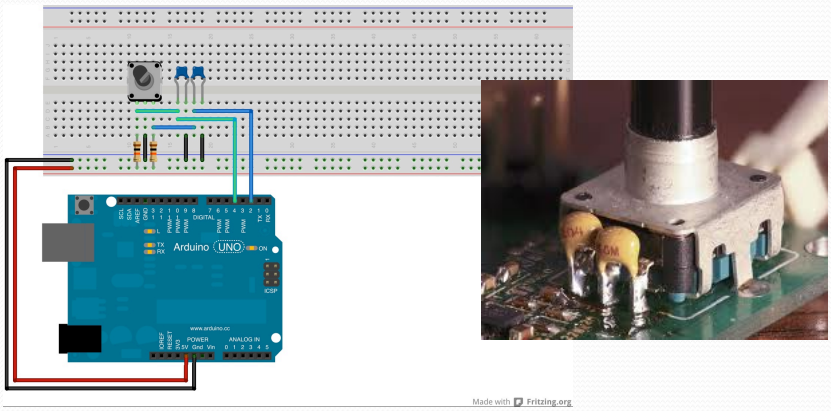
```
void loop(){
  digitalWrite (led, HIGH);
  delay (500);
  digitalWrite (led, LOW);
  delay (500);

}
```


Encoder e Interrupt: lo sketch 4

```
void doEncoder() {
   n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoder0PinB) == LOW) {
       encoder0Pos--;
     } else {
       encoder0Pos++;
     }
     Serial.print (encoder0Pos);
     Serial.print (" | ");
   }
   encoder0PinALast = n;
}
```

Encoder e debounce

- Anche gli encoder usano un debounce
- Si realizza in HW con due condensatori e due resistenze



Encoder e menù

- Uno degli usi di un Encoder è la realizzazione di un menù di funzioni, usando l'Encoder per selezionare la voce e un pulsante per la selezione (sovente incorporato nell'Encoder stesso)

GPS (Global Positioning System)

- Nasce formalmente nel 1973 dalla convergenza di tre sistemi:
  - Transit (chiamato anche NAVSAT (Navy Navigation Satellite System) (US Navy, circa 1960);
  - MOSAIC (MObile System for Accurate ICBM Control) (US Air Force, 1963);
  - SECOR (Sequential Collation of Range) (US Army 1964).
- Il suo "inventore" è Roger L. Easton.

GPS: la storia

- Il primo satellite fu lanciato nel 1989 e il ventiquattresimo nel 1994.
- Fu reso disponibile all'uso "civile" ne 1991 da Regan papà dopo l'incidende del volo 007 della Korean Air Lines, abbattuto dai russi perchè entrato in area protetta.
- Il suo nome era SPS (Standard Positioning System) in contrapposizione al PPS (Precision Positioning System) riservato alle sole forze armate NATO.
- Il segnale civile veniva degradato con la Selective Availability (SA) consentendo precisioni dell'ordine di 100–150 m.
- Nel 2000 Bill Clinton ha rimosso la SA.

GPS: struttura

- È fatto da tre "pezzi"
  - Space Segment: i satelliti (passati da 24 a 31 e diventeranno 32);
  - Control Segment: una stazione primaria e una secondaria, quattro antenne terrestri e 6 stazioni di controllo;
  - User Segmet: i ricevitori GPS.
- Space e Control sono di proprietà e gestione della US Air Force, in particolare della 50th Space Wing (50 SW)
- Europa, Cina e India stanno sviluppando sistemi indipendenti.

GPS: funzionamento

- I satelliti GPS trasmettono su 2 canali:
  - L1, disponibile al servizio SPS per uso civile, 1575,42MHz con precisione metrica;
  - L2 per il servizio PPS, 1227,6MHz, con precisione centimetrica;
- I sistemi PPS sfruttano la doppia frequenza per eliminare l'errore introdotto dalla rifrazione atmosferica.
- I sistemi civili hanno 2 ulteriori limiti:
  - Massima altezza: 18 Km
  - Velocità massima: 515 m/s
  - I due limiti non devono essere concorrenti.

GPS: funzionamento

- Si basa su un metodo di posizionamento sferico detto trilaterazione, che sfrutta la misura del tempo impiegato da un segnale radio a percorrere la distanza satellite-ricevitore.
- In pratica ogni satellite trasmette un segnale orario e viene calcolata la differenza di orario tra il ricevitore e il satellite
- Il problema nasce dalla scarsa precisione dell'orologio contenuto nei ricevitori, che va sincronizzato con quello dei satelliti.

ARI Sezione di Ivrea Serate tecniche 2012

GPS: funzionamento

- Per ovviare a questa scarsa precisione è necessario ricevere il segnale da almeno quattro satelliti.
- Una volta noto il "diametro" delle sfere, dato dalla distanza tra il satellite e il ricevitore, determinato in base al Δt trasmissione/ricezione, si intersecano le tre sfere con una quarta sfera (la superficie terrestre) determinando un punto su quest'ultima.

GPS e Arduino

- Ormai il costo di un ricevitore GPS è sceso parecchio.
- I componenti disponibili possono lavorare tipicamente con due protocolli: seriale o I2C.

ARI Sezione di Ivrea Serate tecniche 2012

Il nostro GPS

IC features:

- Built-in 15X15X4mm ceramic patch antenna on the top of module Ultra-High Sensitivity: **-165dBm** (w/o patch antenna), up to 45dB C/N of SVs in open sky reception.
- High Update Rate: up to 10Hz
- 12 multi-tone active interference canceller
- High accuracy 1-PPS timing support for Timing Applications (10ns jitter)

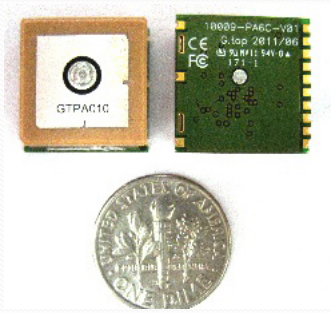
Il nostro GPS

IC features:

- AGPS Support for Fast TTFF (EPO™ Enable 7 days/14 days/ 30 days)
- EASY™: Self-Generated Orbit Prediction for instant positioning fix
- AlwaysLocate™ Intelligent Algorithm (Advance Power Periodic Mode) for power saving
- Logger function Embedded
- Consumption current(@**3.3V**):
  - Acquisition: 25mA Typical
  - Tracking: 20mA Typical

Il nostro GPS

- Modulo PA6C della GlobalTop
- www.gtop-tech.com/en/product/MT3339_GPS_Module_03.html
- Il chip è un MT3339 della MediaTeck



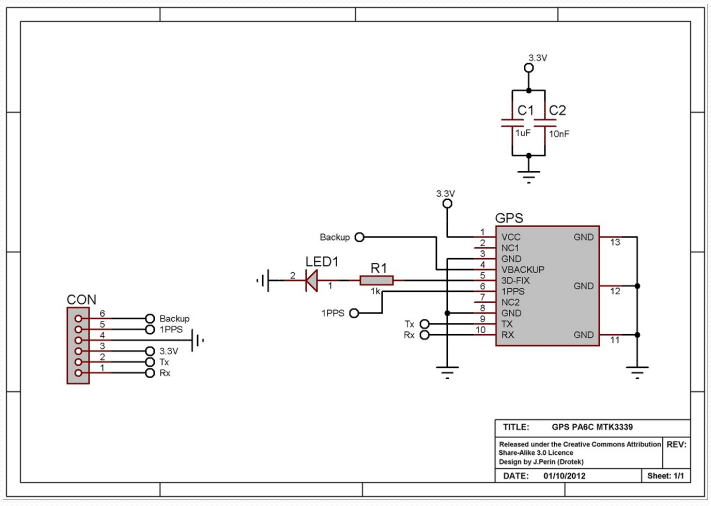
Dimension	16 x 16 x 6.2 mm, SMD type
GPS Solution	MTK MT3339, 66 Channels
Frequency	L1, 1575.42 MHz ; C/A Code
Sensitivity	Acquisition -148 dBm Tracking -165 dBm
Position Accuracy	Without aid: 3.0 m (50% CEP) DGPS (SBAS(WAAS,EGNOS,MSAS)): 2.5 m (50% CEP)
TTFF (Time to First Fix)	Cold Start: <35 Seconds (Typical) Warm Start: <33 Seconds (Typical) Hot Start: <1 Seconds (Typical)
Timing Accuracy (1PPS Output)	10 ns RMS
Protocol	NMEA 0183
DGPS	SBAS [QZSS, WAAS, EGNOS, MSAS,GAGAN]
AGPS	Supported (Offline Mode)
Interface	1 UART
Baud Rate	4800 ~ 115200 bps
Update Rate	1 ~ 10 Hz
Power Supply	VCC: 3.0 V to 4.3 V VBACKUP: 2.0 V to 4.3 V
Current Consumption	Acquisition: 25 mA Tracking: 20 mA
Working Temperature	-40 °C to +85 °C

Il nostro GPS

Board features:

- Dimensions 18 mm x 27 mm
- Ground plane on top and bottom of board
- Power supply: 3.3V
- Blue LED for STATUS
- No need external antenna
- 115200 Baud & 10 Hz firmware uploaded

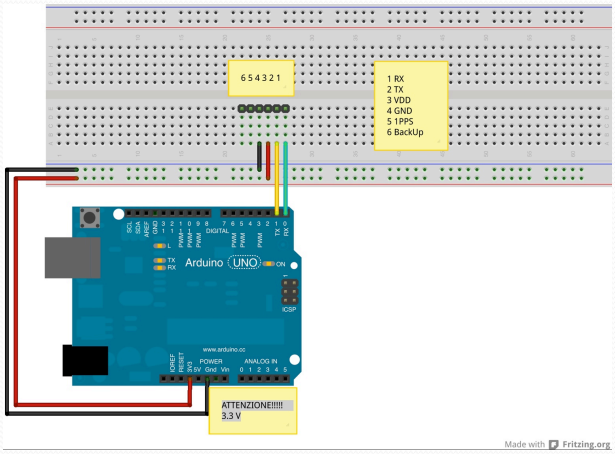
Il nostro GPS



GPS diretto

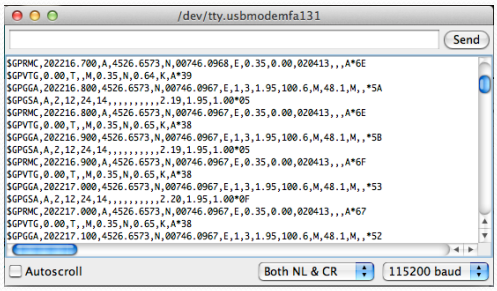
- Sfruttiamo Arduino solo come convertitore per leggere sul Serial Monitor la corretta ricezione, in modo da controllare il corretto funzionamento.
- Lo sketch è praticamente vuoto
- Lo sketch va caricato PRIMA di collegare i pin RX e TX, altrimenti si va in conflitto con il Serial Monitor.
- Il monitor va impostato a 115200 baud.
- ATTENZIONE: usate il 3.3V, altrimenti…

GPS diretto: circuito



GPS diretto: sketch e output

```
// VDD +3.3V
// GND …
// GPS RX Digital 0
// GPS TX Digital 1

void setup() {}
void loop() {}
```

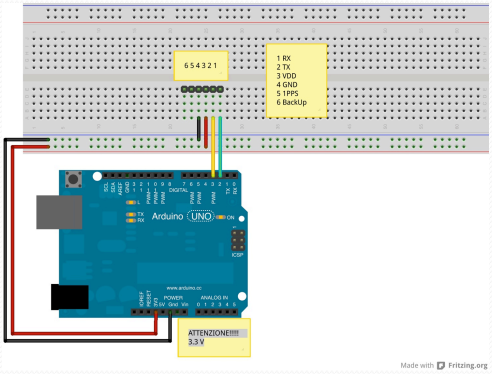


GPS “advanced”

- Usiamo altri 2 pin per dialogare in seriale con Arduino
- Sfruttiamo la libreria Adafruit_GPS
- Si scarica da:
  - https://github.com/adafruit/Adafruit-GPS-Library
- Dopo aver scaricato lo zip va scompattato e copiato nella /libraries (ricordatevi di chiudere e rilanciare il vostro IDE).
- NB: rinominate il file in Adafruit_GPS

GPS “advanced”: circuito e note

- Ricordatevi che il nostro GPS va a 115200!
- Ricordatevi che il nostro GPS è a 3.3V!



Next time

- GPS “advanced”
- Comunicazioni Seriali con Arduino
- Analizzatori Logici
- Menù?

Q & A

Next time: output “fisico”

- Gestione degli attuatori
  - Gestione di un Servo
  - PWM
- Audio

Che ci servirà?

Oltre alla solite cose:

- Un servomotore
- ?