

FUTURA

Connettere, trasformare, innovare

Elettronica open al lavoro con robot, computer e smartphone, in laboratorio e sul campo

Daniele Grosso



MINISTERO DELL'ISTRUZIONE DELL'UNIVERSITA' E DELLA RICERCA





Hello!

Daniele Grosso

Titoli: Laurea in Fisica, Specializzazione in Fisica Medica,
Dottorato in Fisica, Abilitazione per l'insegnamento della Fisica

Ricerca: in Fisica Medica, Fisica Ambientale, Astrofisica, Robotica, Didattica

Didattica: supporto in corsi per Fisica, Chimica, Ingegneria Elettrica
Scuole, corsi, campi estivi di Robotica, Physical Computing, Coding



Dipartimento di Fisica
Università degli Studi di Genova

grosso@fisica.unige.it



Physical Computing Robotica e Coding

Dipartimento di Fisica di Genova e Associazione per l'Insegnamento della Fisica

A partire dal 2010, linea per le attività di Laboratorio del PLS

- **Scuola Estiva** (Nazionale) per gli Insegnanti - la **III settimana di luglio**, 18|24 posti
- **Scuola Invernale** (Regionale) per gli Insegnanti – 1 pomeriggio a settimana, 8 incontri

Corsi al DIFI (Dipartimento di Fisica, UNIGE), in Laboratorio - **collaborazione con AIF**

Si affrontano tematiche come:

Physical Computing con arduino, Robotica, Coding, Modelli e Simulazioni, Smartphone in lab
per avvicinare i ragazzi alle discipline scientifiche

ROBOTICA e PLS

*Sfruttiamo il fatto che la Robotica è intrinsecamente **interdisciplinare** per **stimolare l'interesse verso le materie scientifiche***



“

per interni



4-99 anni

per il banco



6|8-99 anni



8-99 anni, **ok all'esterno**

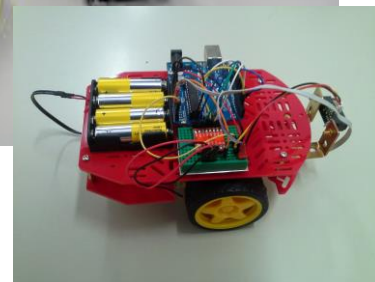
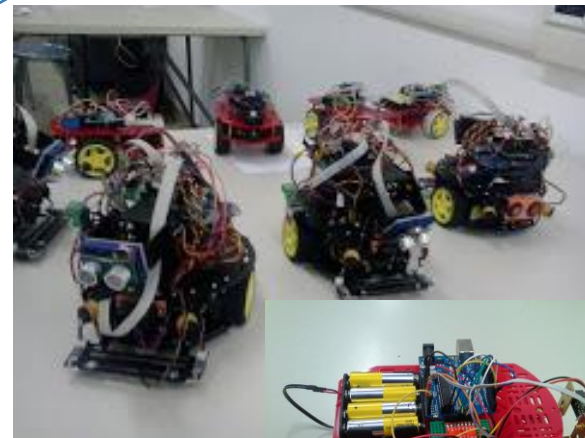
sono range indicativi !

Robotica

Thymio – **Ozobot EVO** – **mbot ranger** – **zumo** – **sphero**

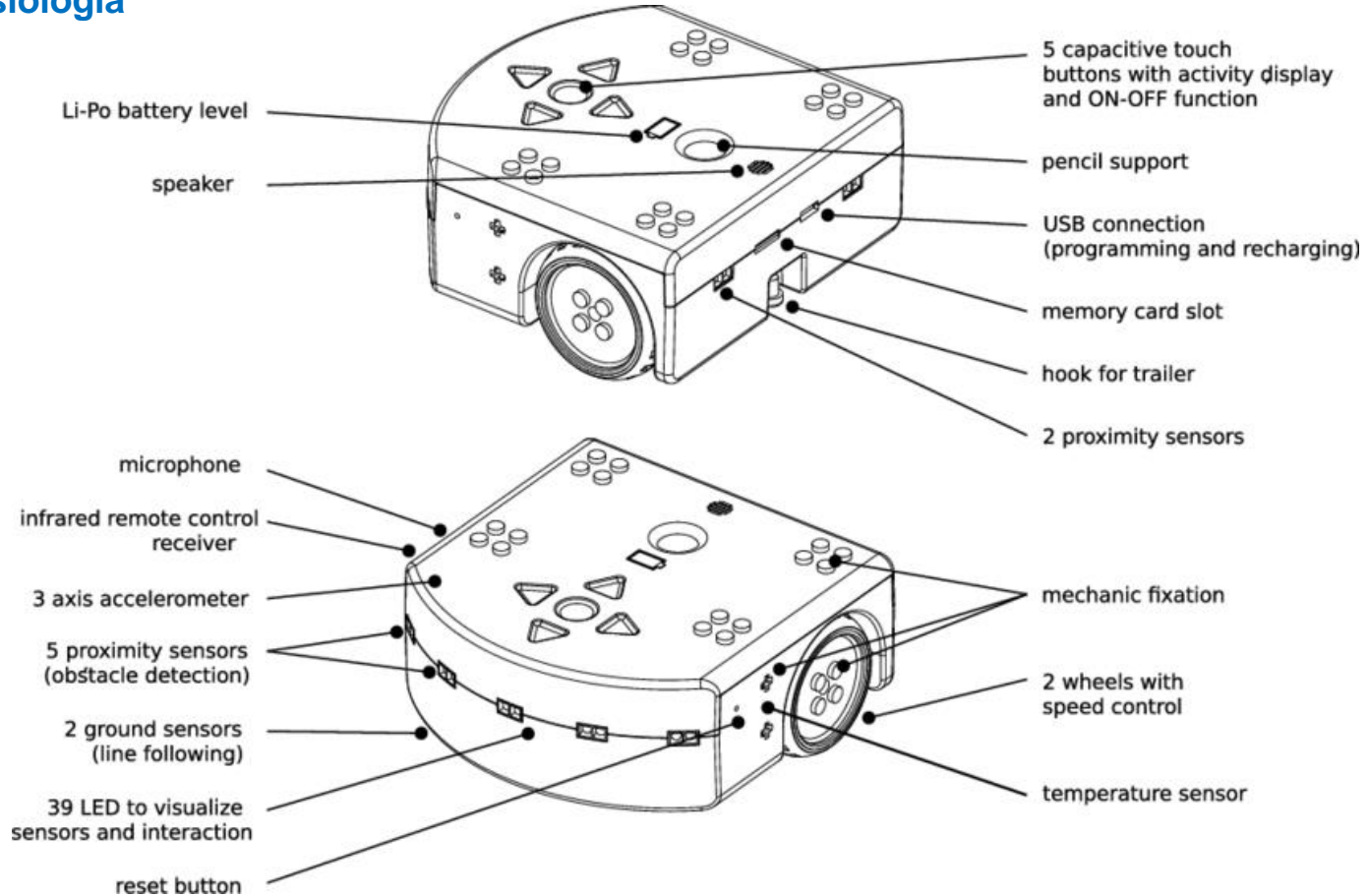
Piattaforme autocostruite:

MOMOLAB (Modular Mobile LABoratory) - **LEMU** (Light Edition Mobile Unit) - **CONTUBOT**- ...



Thymio

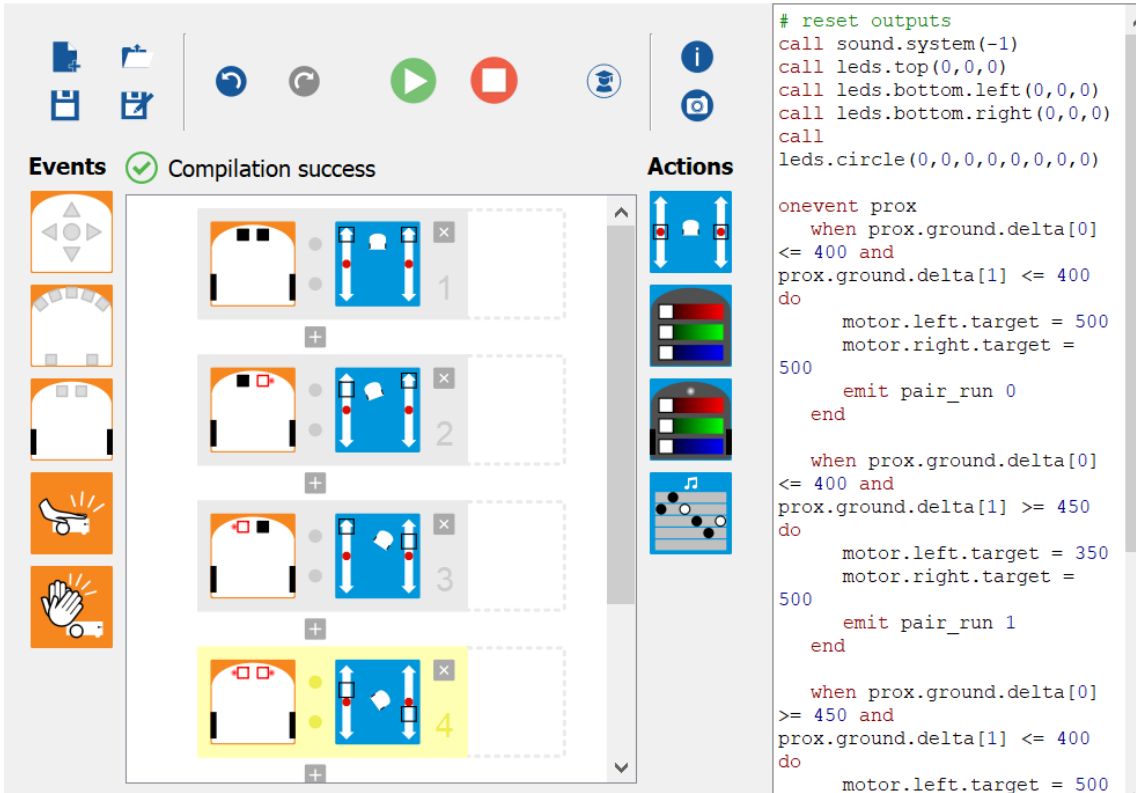
anatomia e fisiologia



Thymio – LINE FOLLOWER

Programmazione event driven in VPL

Untitled [modified] - Thymio Visual Programming Language - ver. 1.6.1



```
# reset outputs
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call
leds.circle(0,0,0,0,0,0,0)

onevent prox
  when prox.ground.delta[0]
  <= 400 and
  prox.ground.delta[1] <= 400
  do
    motor.left.target = 500
    motor.right.target =
  500
    emit pair_run 0
  end

  when prox.ground.delta[0]
  <= 400 and
  prox.ground.delta[1] >= 450
  do
    motor.left.target = 350
    motor.right.target =
  500
    emit pair_run 1
  end

  when prox.ground.delta[0]
  >= 450 and
  prox.ground.delta[1] <= 400
  do
    motor.left.target = 500
```



ambiente visual su android, non è necessario saper leggere => ok da 5/6 anni

Thymio – SUMO

Programmazione event driven in VPL

Untitled [modified] - Thymio Visual Programming Language - ver. 1.6.1

Events ✓ Compilation success

Actions

1

2

3

4

```
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call
leds.circle(0,0,0,0,0,0,0,0)

onevent prox
  when prox.ground.delta[0]
  >= 450 and
  prox.ground.delta[1] >= 450
```

Untitled [modified] - Thymio Visual Programming Language - ver. 1.6.1

Events ✓ Compilation success

Actions

4

5

6

7

```
# reset outputs
call sound.system(-1)
call leds.top(0,0,0)
call leds.bottom.left(0,0,0)
call leds.bottom.right(0,0,0)
call
leds.circle(0,0,0,0,0,0,0,0)

onevent prox
  when prox.ground.delta[0]
  >= 450 and
  prox.ground.delta[1] >= 450
do
  motor.left.target = 500
  motor.right.target =
500
  emit pair_run 0
end

  when prox.ground.delta[0]
  <= 400 and
  prox.ground.delta[1] <= 400
do
  motor.left.target =
-200
  motor.right.target =
-350
  emit pair_run 1
end

  when prox.ground.delta[0]
  >= 450 and
  prox.ground.delta[1] <= 400
do
```

Consemplici modifiche si può trasformare l'iseguitore di linea in un lottatore di SUMO

Leggere i dati provenienti da un sensore e prendere decisioni in base ai valori letti – MBOT RANGER

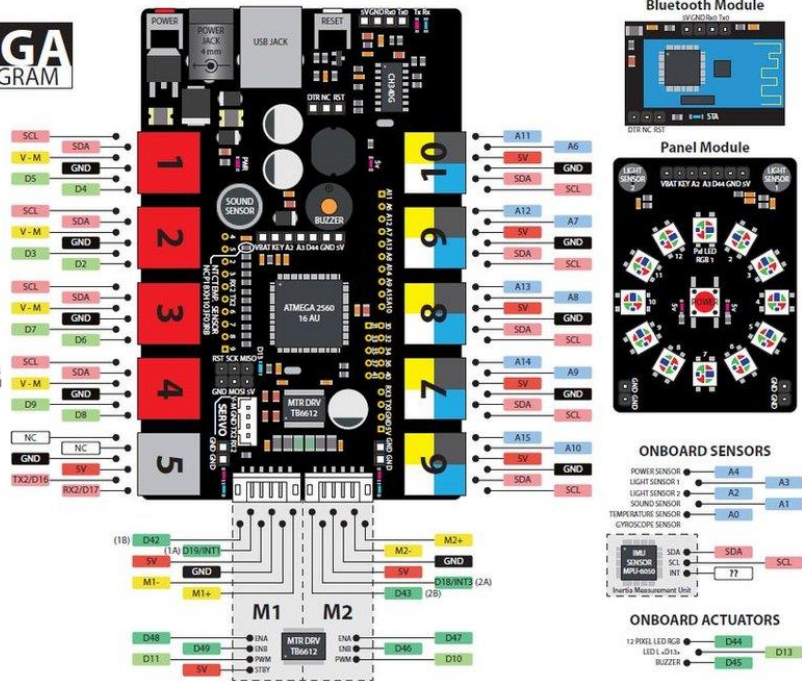
ME AURIGA PINOUT DIAGRAM

PINOUT LEGEND

- 5V Power 5V
- VBAT Power VBat
- Motor Power
- GND Ground
- Digital IN-OUT / Analog IN
- Digital IN-OUT / Analog OUT
- Digital IN-OUT
- Serial communication

CONNECTOR LEGEND

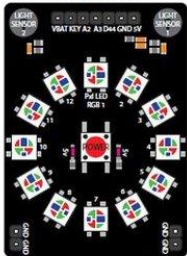
- Red means the output voltage is 6-12V, and it is usually connected to the motor driver module of 6-12V voltage.
- Single-digital port.
- Double-digital port.
- Serial port of hardware.
- Analog signal port.
- PC port.



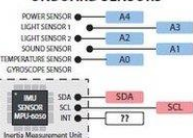
Bluetooth Module



Panel Module



ONBOARD SENSORS



ONBOARD ACTUATORS

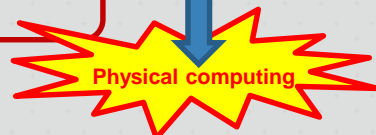


Una notevole semplificazione rispetto alla gestione «low level» del sensore !!

mBot Program

```

forever
  set dis to ultrasonic sensor Port3 distance
  set temp to line follower Port2
  if temp = 0 then
    run forward at speed 80
  if temp = 1 then
    turn left at speed 80
  if temp = 2 then
    turn right at speed 80
  if temp = 3 then
    run backward at speed -80
  if dis < 3 then
    set led led on board all red 20 green 0 blue 0
    take_action
  else
    set led led on board all red 0 green 20 blue 0
  
```



```

define take_action
  run forward at speed 0
  
```

Quale è il collegamento con la disciplina? (ad esempio con la didattica della Fisica)

La cinematica del dual drive

per la stabilità ed il controllo di direzione servono almeno 2 ruote indipendenti ed un punto di appoggio...

$$\Delta\theta_{(k)} = \frac{\Delta U_{R(k)} - \Delta U_{L(k)}}{L}$$



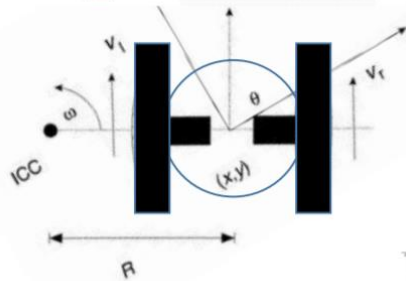
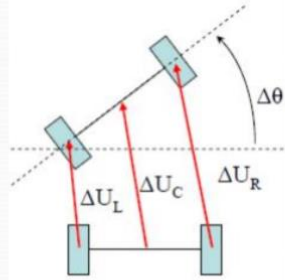
Integrazione...

$$\theta_{(k)} = \theta_{(k-1)} + \Delta\theta_{(k)}$$

$$\theta_{(k)} = \theta_{(k-1)} + \Delta\theta_{(k)}$$

$$x_{(k)} = x_{(k-1)} + \Delta U_{C(k)} \cos\theta_{(k)}$$

$$y_{(k)} = y_{(k-1)} + \Delta U_{C(k)} \sin\theta_{(k)}$$

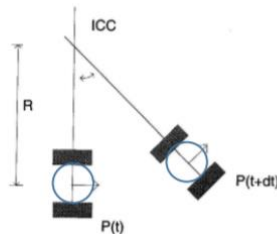


$$ICC = [x - R \sin(\theta), y + R \cos(\theta)]$$

$$\omega (R + l/2) = V_r$$

$$\omega (R - l/2) = V_l$$

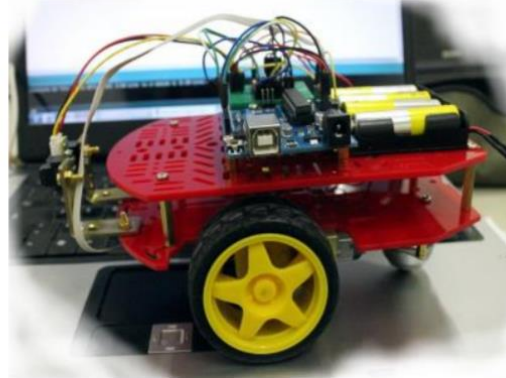
$$R = \frac{l}{2} \frac{V_l + V_r}{V_r - V_l}; \quad \omega = \frac{V_r - V_l}{l}$$



$t + \delta t$



$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix}$$



livello avanzato

Quale è il collegamento con la disciplina? (ad esempio con la didattica della Fisica)

Odometria e determinazione della traiettoria
errori casuali e sistematici, correzione con data fusion

Odometria:

- valutare lo spostamento
- sommare lo spostamento attuale al precedente per ottenere la **traiettoria**

Problema:

- l'accumolo degli errori **casuali**
- la **presenza di errori sistematici**

Tecnica di test:

- programmare il **movimento lungo il perimetro di un quadrato**
- effettuare misure per la **calibrazione**

Compensazione degli errori mediante la data fusion:

La traiettoria viene corretta con informazioni provenienti da fonti indipendenti
bussola, gps, triangolazione, mappe ...

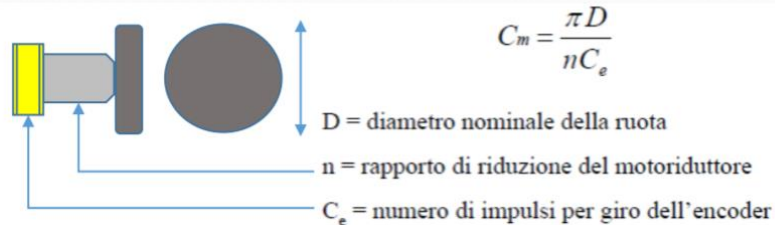
livello base

livello intermedio

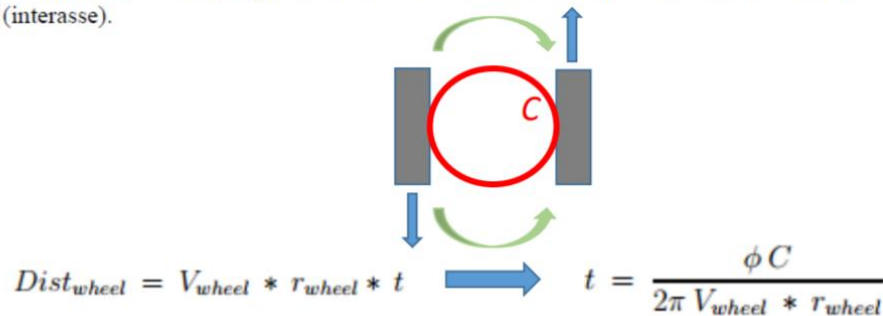
livello avanzato

Per misure di distanza percorsa e per stabilizzare la traiettoria occorre:

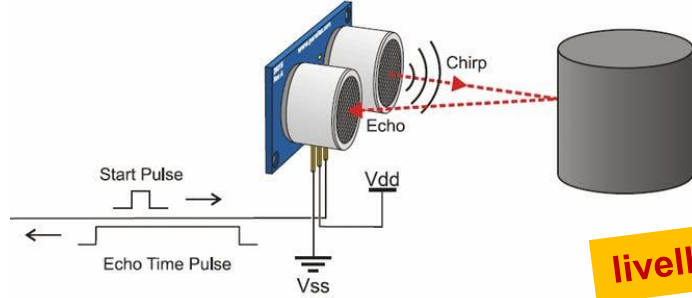
- convertire la rotazione delle ruote in un movimento lineare
- compensare la diversa velocità di rotazione dei motori



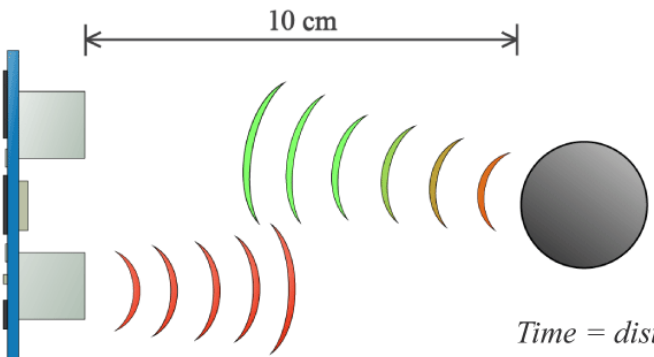
Per studiare i movimenti del robot, si studiano i movimenti del punto C, cioè del centro dell'asse che congiunge le ruote. Con b indichiamo la distanza fra le ruote (interasse).



Funzionamento di un sensore ad ultrasuoni



livello base



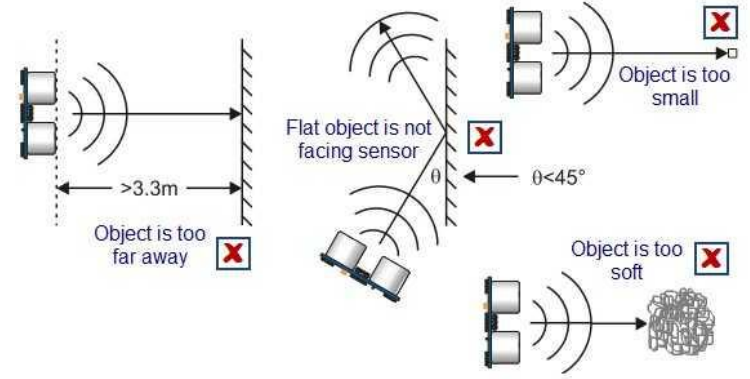
speed of sound:
 $v = 340 \text{ m/s}$
 $v = 0,034 \text{ cm}/\mu\text{s}$

Time = distance / speed:
 $t = s / v = 10 / 0,034 = 294 \mu\text{s}$

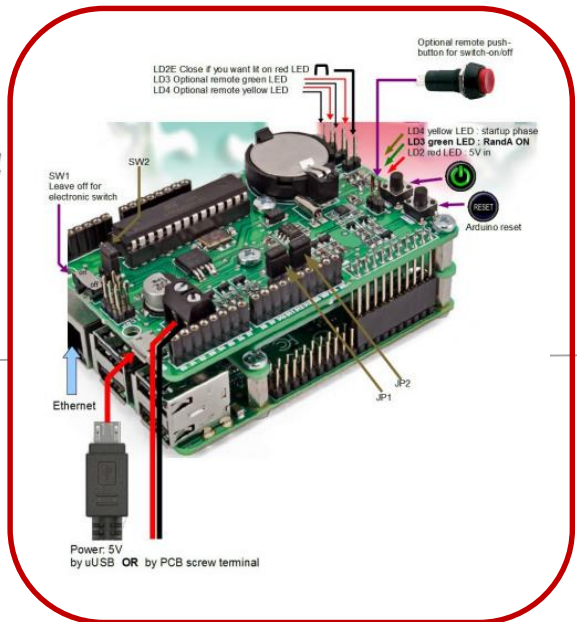
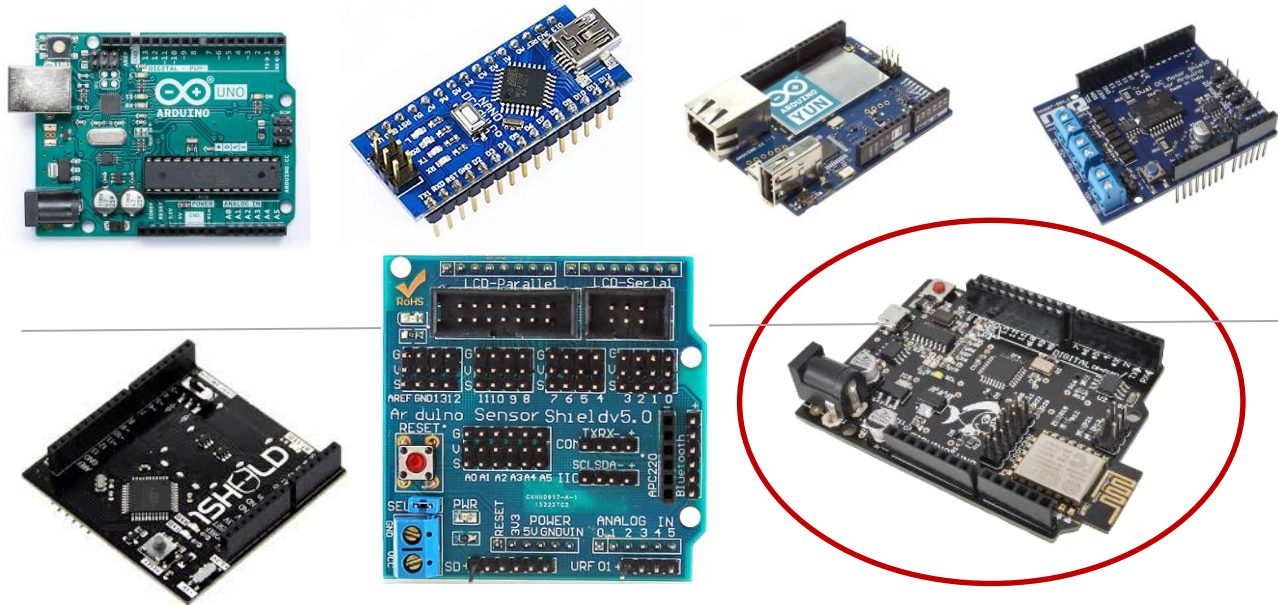
Distance:
 $s = t \cdot 0,034 / 2$

livello base/intermedio

PROBLEMI



Osservati **sperimentalmente** da bambini e i ragazzi analizzando il comportamento dei robot che hanno costruito



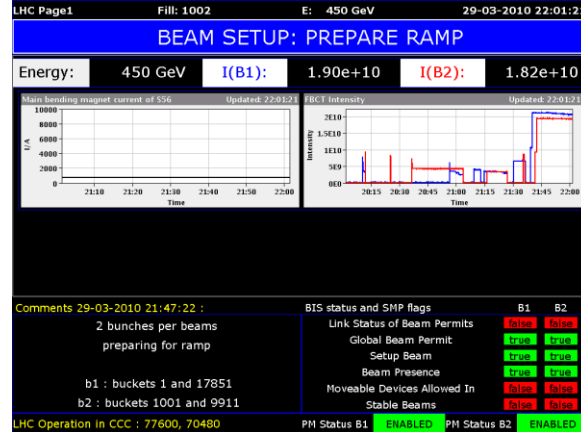
Physical Computing con arduino, raspberry

Cosa succede «dietro le quinte»? – APPROFONDIAMO: sensori, acquisizione, ADC, DAC, elaborazione ...

- Arduino uno, arduino nano, **fishino** uno, **fishino mega** + shield commerciali/autocostruite
- **Raspberry**
- **microbit**

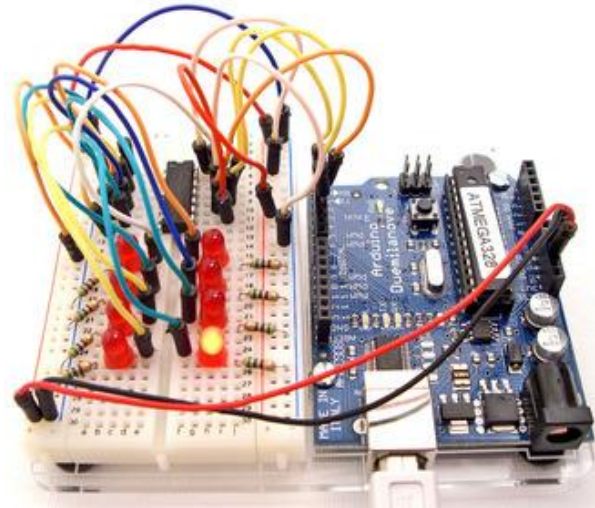
Physical computing

Physical computing, in the broadest sense, means building interactive [physical systems](#) by the use of [software](#) and [hardware](#) that can sense and respond to the [analog](#) world
wikipedia

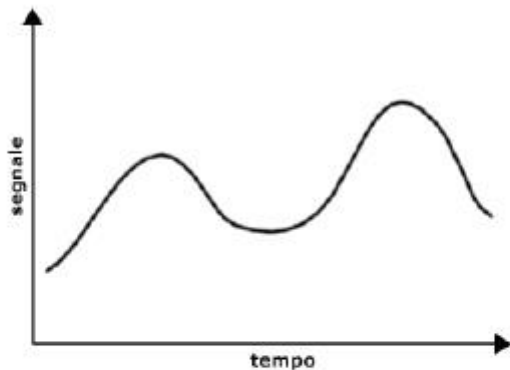


Physical computing

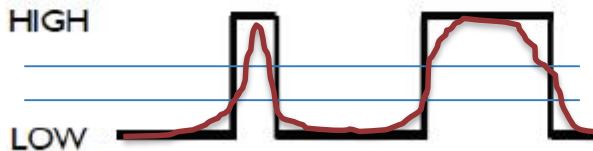
Physical computing, in the broadest sense, means building interactive [physical systems](#) by the use of [software](#) and [hardware](#) that can sense and respond to the [analog](#) world
wikipedia



Segnali digitali ed analogici



Un **segnale analogico** può assumere qualsiasi valore (entro un dato range)
Si può pensare che siano “analogiche” tutte le grandezze fisiche misurabili nell’ambiente, in realtà molte grandezze fisiche sono discrete e l’atto del misurare implica una «discretizzazione» a causa della risoluzione limitata dello strumento con cui si effettua la misura ...

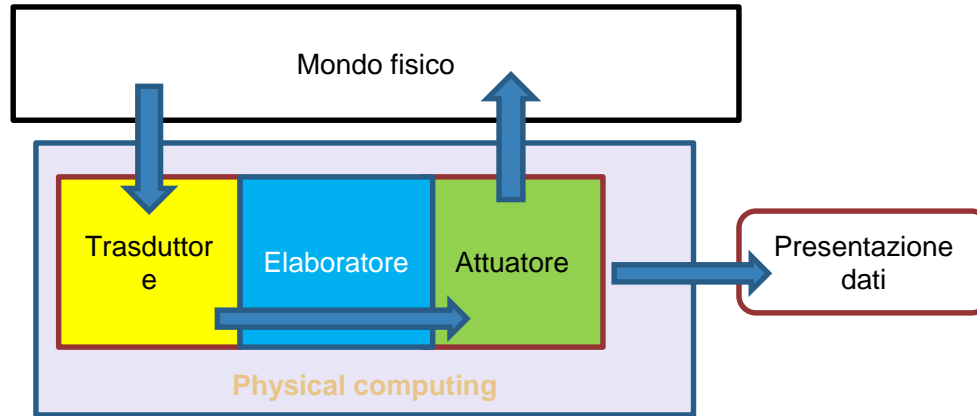


Un **segnale digitale** minimale può assumere due soli stati: High/Low, 1/0, alto/basso corrispondenti a **due livelli di tensione** (ad esempio, 5-0V).
Si può pensare che siano “digitali” tutte le informazione scambiate tra «componenti logici»...

TTL $s < 0,8 \text{ V}$ e quella superiore a $s > 2,2 \text{ V}$ (altrimenti è indeterminato)

CMOS (5V): $s < 1,3 \text{ V} \Rightarrow 0$ e $s > 3,7 \text{ V} \Rightarrow 1$

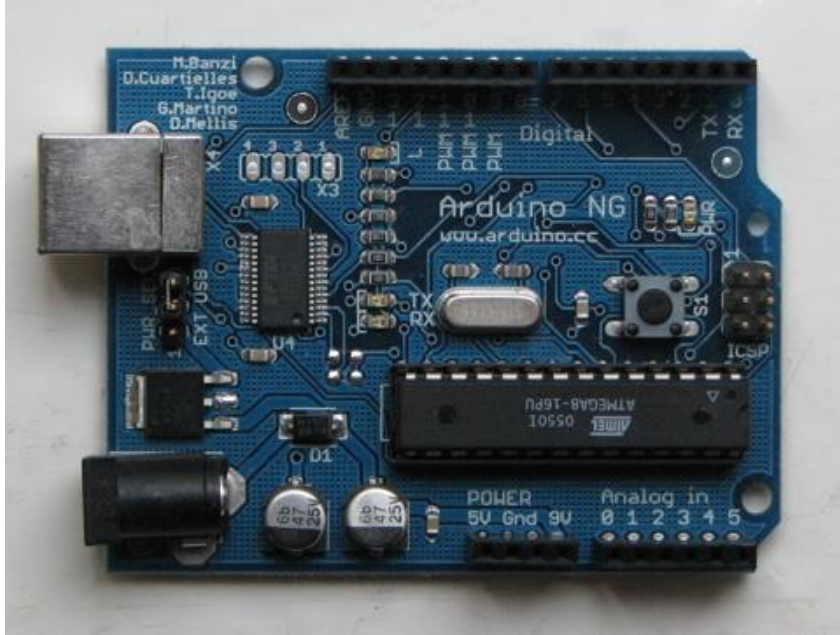
Il dispositivo ideale per il physical computing



Trasduttore/i che non interferisce con i fenomeni/altera la misura, sensibilità, accuratezza, riproducibilità, rate campionamento, immunità al rumore, selettività etc illimitati,
Attuatore/i con caratteristiche ideali analoghe al trasduttore, potenza illimitata in uscita etc
Elaboratore con memoria illimitata, capacità di calcolo illimitata etc etc

La realtà è differente, **ogni miglioramento aumenta i costi**, tuttavia,
a patto di accettare compromessi ragionevoli,
sono disponibili in commercio strumenti a costi accessibili
(quindi di interesse chi si occupa di didattica)

arduino



Arduino è una piattaforma hardware per il physical computing sviluppata all'Interaction Design Institute di Ivrea, un istituto fondato dalla Olivetti e da Telecom Italia

Arduino è basato su

- una scheda di I/O e
- un ambiente di sviluppo che usa una libreria Wiring per semplificare la scrittura di programmi

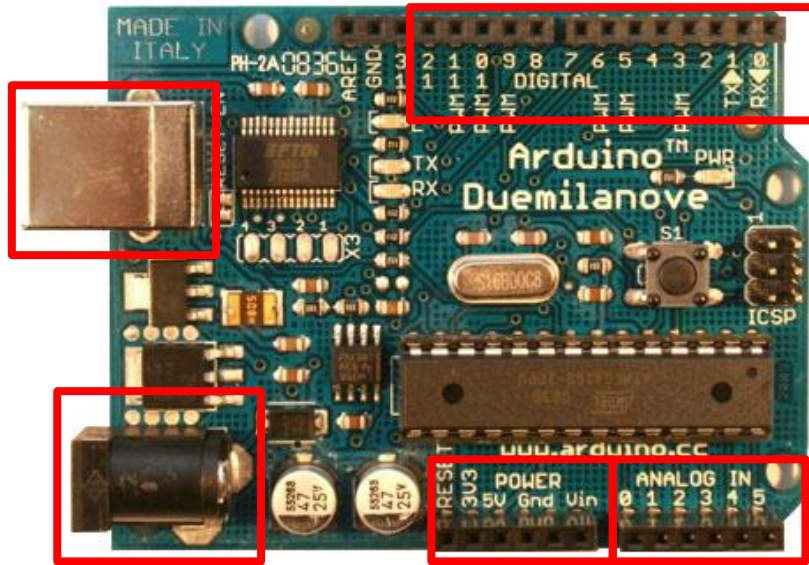
Tool di sviluppo

- ARDUINO: <http://arduino.cc/>
- PROCESSING: <http://processing.org/>

Versioni differenti di arduino

- Serial Arduino, con [porta seriale DB9](#), e [microcontroller](#) ATmega8
- Arduino Extreme con [USB](#) e chip ATmega8
- Arduino Mini, versione in miniatura con ATmega168 a [montaggio superficiale](#)
- Arduino Nano, più piccola della Mini, controller ATmega168 [SMD](#) e alimentaz. USB
- LilyPad Arduino per applicazioni indossabili con ATmega168 [SMD](#)
- Arduino NG, con e ATmega8
- ...
- Arduino BT, con [Bluetooth](#) e ATmega168
- Arduino Duemilanove con Atmega168 (o Atmega328)
e alimentata in [corrente continua](#) tramite USB, con switching automatico
- Arduino UNO
- Arduino Mega con **ATmega2560** a montaggio superficiale e memoria aggiuntiva
- Seeduo Mega con **ATmega1280** a montaggio superficiale e memoria aggiuntiva

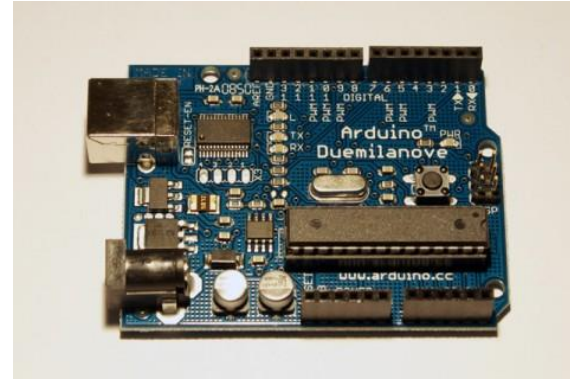
ARDUINO 2009



- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out – TX/RX (dark green)
- 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)

Caratteristiche tecniche

- *ATmega168*
- **Digital I/O Pins: 14**
(of which 6 provide PWM output)
- **Analog Input Pins: 6** (DIP) or 8
(SMD)
- DC Current per I/O Pin: **40 mA**
- Flash Memory
- **16 KB** SRAM
- **1 KB** EEPROM



ARDUINO

Caratteristiche tecniche – PIN digitali

Pin Digitali

I Pin Digitali dell'Arduino possono essere utilizzati come input o output attraverso i comandi [pinMode\(\)](#), [digitalRead\(\)](#), e [digitalWrite\(\)](#).

Ogni Pin ha una resistenza di pull-up interna che può essere accesa o spenta usando [digitalWrite\(\)](#) (con un valore di HIGH o LOW) quando il Pin é configurato come input.

Il carico massimo di corrente per Pin é di 40 mA.

Serial: 0 (RX) and 1 (TX). usati per ricevere (RX) o inviare (TX) dati seriali (TTL)

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

See the [attachInterrupt\(\)](#) function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

BT Reset: 7. (Arduino BT-only)

Connected to the reset line of the bluetooth module.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).

These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

LED: 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

ARDUINO

Caratteristiche tecniche – pin analogici

Pin Analogici

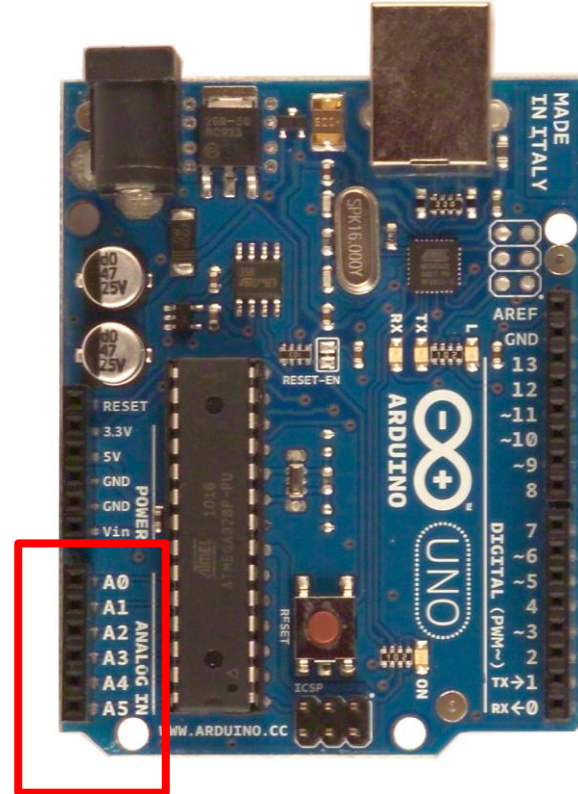
I Pin Analogici supportano conversioni da analogico a digitale con una **risoluzione di 10 bit** (ADC) usando la funzione [analogRead](#)

I Pin Analogici possono essere anche usati come Pin Digitali: analog input 0 come digital pin 14 fino a analog input 5 come digital pin 19

I Pin Analogici 6 and 7 (presenti sull'Arduino Mini e BT) non possono essere usati come Pin Digitali.

I²C: 4 (SDA) and 5 (SCL).

Supporto del protocollo I²C (TWI) per mezzo della [libreria Wire](#) (la documentazione sul sito di Wiring).



ARDUINO

Caratteristiche tecniche - alimentazione

Pin di Alimentazione

- **VIN** (alcune volte appare come "9V"). Il voltaggio della scheda quando l'Arduino usa un'alimentazione esterna (diversa quindi dall'alimentazione standard dell'USB di 5V) **si può alimentare la scheda attraverso questo Pin** se la scheda é alimentata attraverso il jsk si può ricevere l'alimentazione da questo Pin.
- **5V**. L'alimentazione standard usata per alimentare la scheda e i sensori/attuatori attaccati ad essa. questo può avvenire anche attraverso VIN con il regolatore di tensione a bordo della scheda oppure mediante un alimentatore esterno a5V
- **3V3**. (Solo 2009)
Pin di alimentazione legato al Chip FTDI, ma accessibile
- **GND**. Masse

Altri Pin

AREF. [Reference](#)

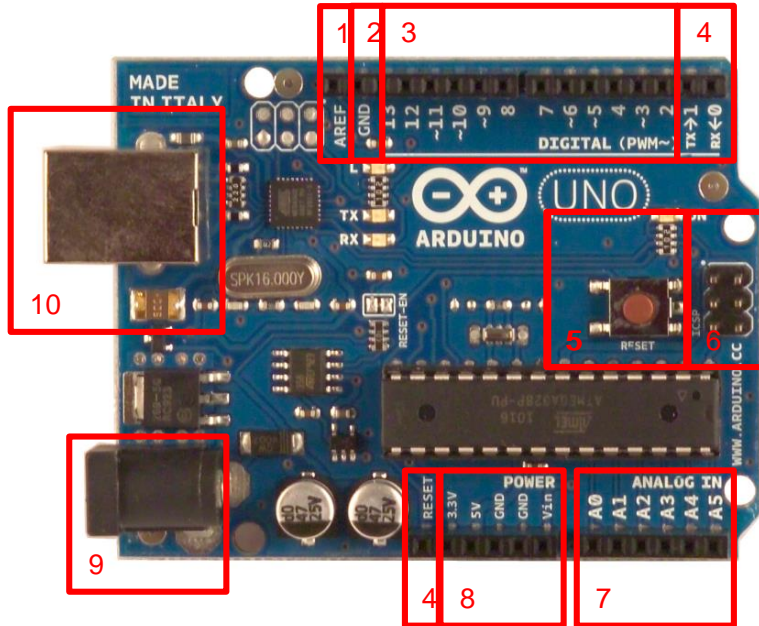
voltage for the analog inputs. Used with [analogReference\(\)](#)

Reset. (Solo 2009)

Bring this line LOW to reset the microcontroller.

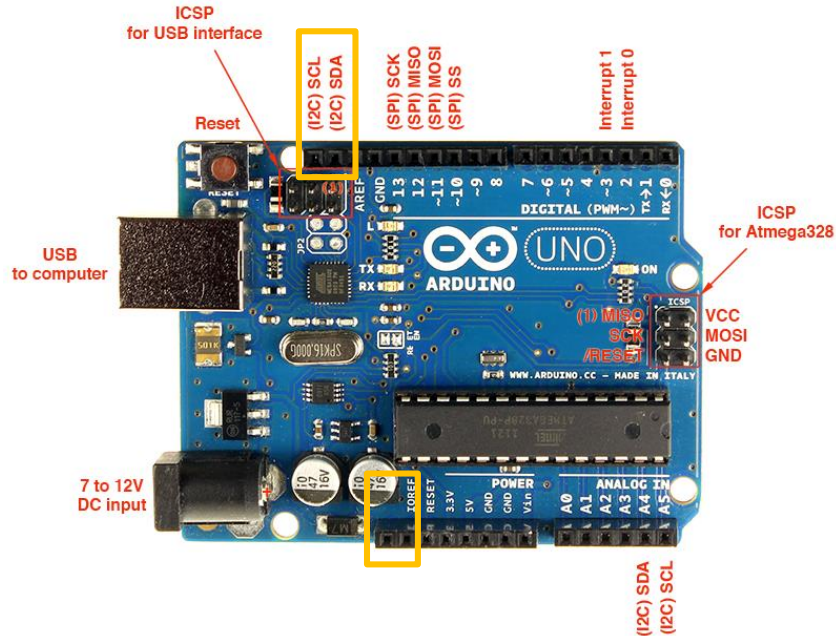
Typically used to add a reset button to shields which block the one on the board

ARDUINO UNO



1. Analog Reference pin
2. Digital Ground
3. Digital Pins 2-13
4. Digital Pins 0-1/Serial In/Out – TX/RX
5. Reset Button – S1
6. In-circuit Serial Programmer
7. Analog In Pins 0-5
8. Power and Ground Pins
9. External Power Supply In (9-12VDC) – X1 Toggles External Power and USB Power (place jumper on two pins closest to desired supply) – SV1
10. 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)

ARDUINO UNO REV3

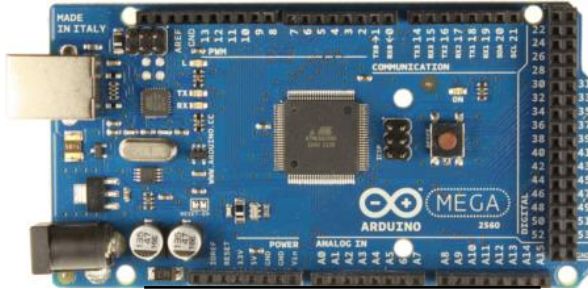


1. Analog Reference pin
2. Digital Ground
3. Digital Pins 2-13
4. Digital Pins 0-1/Serial In/Out – TX/RX
5. Reset Button – S1
6. In-circuit Serial Programmer
7. Analog In Pins 0-5
8. Power and Ground Pins
9. External Power Supply In (9-12VDC) – X1
Toggles External Power and USB Power
(place jumper on two pins closest to desired supply) – SV1
10. 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)

Le caratteristiche introdotte nella versione R3 sono:

- **ATmega16U2** invece del 8U2 come **convertitore USB-seriale**
- **sono stati aggiunti i pin SDA e SCL per la comunicazione TWI vicino al pin AREF e vicino al pin RESET** altri sono stati collocati altri due nuovi pin, **il pin IOREF** che consentirà agli shield di adattarsi alla tensione fornita dalla scheda, ed un pin riservato per scopi futuri
- **Circuito di RESET migliorato**

Arduino mega

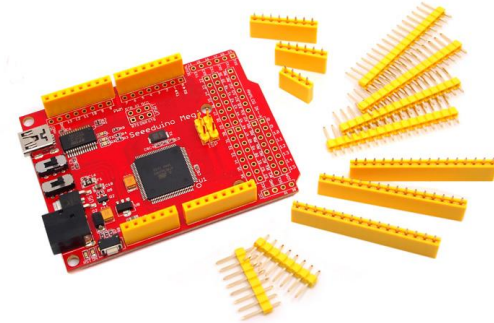


Microcontrollore	ATmega2560
Tensione di funzionamento	5 V
Tensione di ingresso (raccomandato)	7-12V
Tensione di ingresso (limiti)	6-20V
I/O digitali	54 (di cui 14 funzionano anche come uscite PWM)
Ingressi analogici	16
Corrente DC per ogni pin I/O	40 mA
Corrente DC per pin 3.3V	50 mA
Memoria flash	256 KB di cui 8 KB utilizzati dal bootloader
SRAM	8 KB
EEPROM	4 KB
Velocità di clock	16 MHz

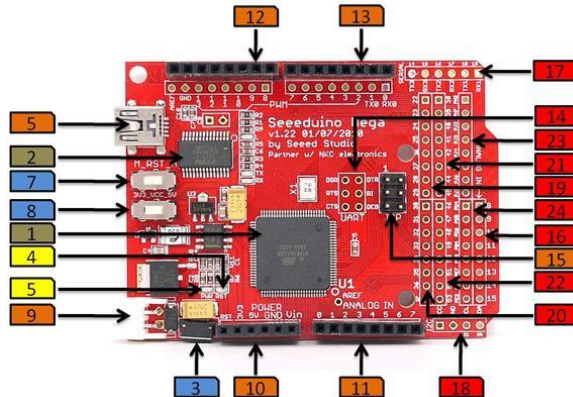


dispone di un **host Usb** per il collegamento con dispositivi Android e l'utilizzo del kit di sviluppo ADK di Google

Seeduino mega

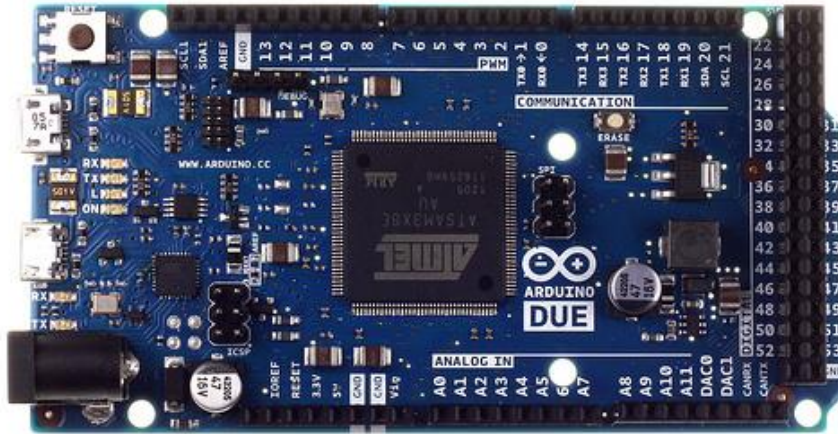


derived from [Arduino Mega](#) , based on ATmega1280
Seeduino Mega v3.0 is based on ATmega2560
instead of ATmega1280 which has larger Flash
Memory 256KB



- 1.Label [5]** The Seeduino Mega uses a 4 Position mini-Type B Jack while the rest of the Arduino family uses standard 4 Position Type B Jack. The Arduino uses circular Power Jack.
- 2.Label[9]** The Seeduino Mega uses a 2 Pin JST connector for the power supply while the rest of the Arduino family uses coaxial power plugs.
- 3.Label[3]** For the Reset button the Seeduino Mega uses a an edge mounted push button while other Arduino based platforms uses a four legged tack switch .
- 4.Yellow Labels** are indicator LEDs.
- 5.Blue Labels** are mode selector and reset switches.
- 6.Brown Labels** are items that are common to the Arduino family; this makes the Seeduino Mega compatible with most Arduino shields.
- 7.Red Labels** are the things that add the MEGA in the Seeduino, the set includes additional ADC input pins and a bunch of bidirectional IO pins.

Arduino due



- Arduino board based on a 32-bit ARM core microcontroller
- based on the Atmel SAM3X8E ARM Cortex-M3 CPU ([datasheet](#)).
- 54 digital input/output pins
- 12 can be used as PWM outputs,
- 12 analog inputs
- 4 UARTs (hardware serial ports),
- 84 MHz clock, an USB OTG capable connection,
- 2 DAC (digital to analog)
- 2 TWI, a power jack, an SPI header, a
- JTAG header, a reset button and an erase button

Arduino Due board runs at 3.3V

The maximum voltage that the I/O pins can tolerate is 3.3V

Providing higher voltages, like 5V to an I/O pin could damage the board

Seeduino stalker



7300-STALKERV2



7300-BLUETBEE



7300-UARTSBEE

Può essere impiegata come hub per la raccolta dati

- microcontrollore ATmega328
- Dispone di RTC (Real Time Clock), slot per micro SD card,
- 20 I/O
- protocollo di comunicazione I2C e UART
- connettore per inserire moduli wireless come Xbee, Bluetooth e RF

WiFiBee

WiFi MAC module (MRF24WB0MA based)
For adding WiFi to your Stalker. Connects to the microcontroller via SPI. TCP/IP stack needs to be added separately.

BluetoothBee

Bluetooth SPP Module (BlueCore4 based)
For adding a Bluetooth Serial Port to your Stalker. Connects to the microcontroller via UART.

XBee / XBee ZB

Proprietary RF / ZigBee Module
For adding a radio interface meant for wireless sensor mesh networks to your Stalker. Connects to the microcontroller via UART. Accepts AT commands.

GPSBee

Global Positioning System Module
For adding location sensing to your Stalker. Connects to the microcontroller via UART. Outputs NMEA Messages.

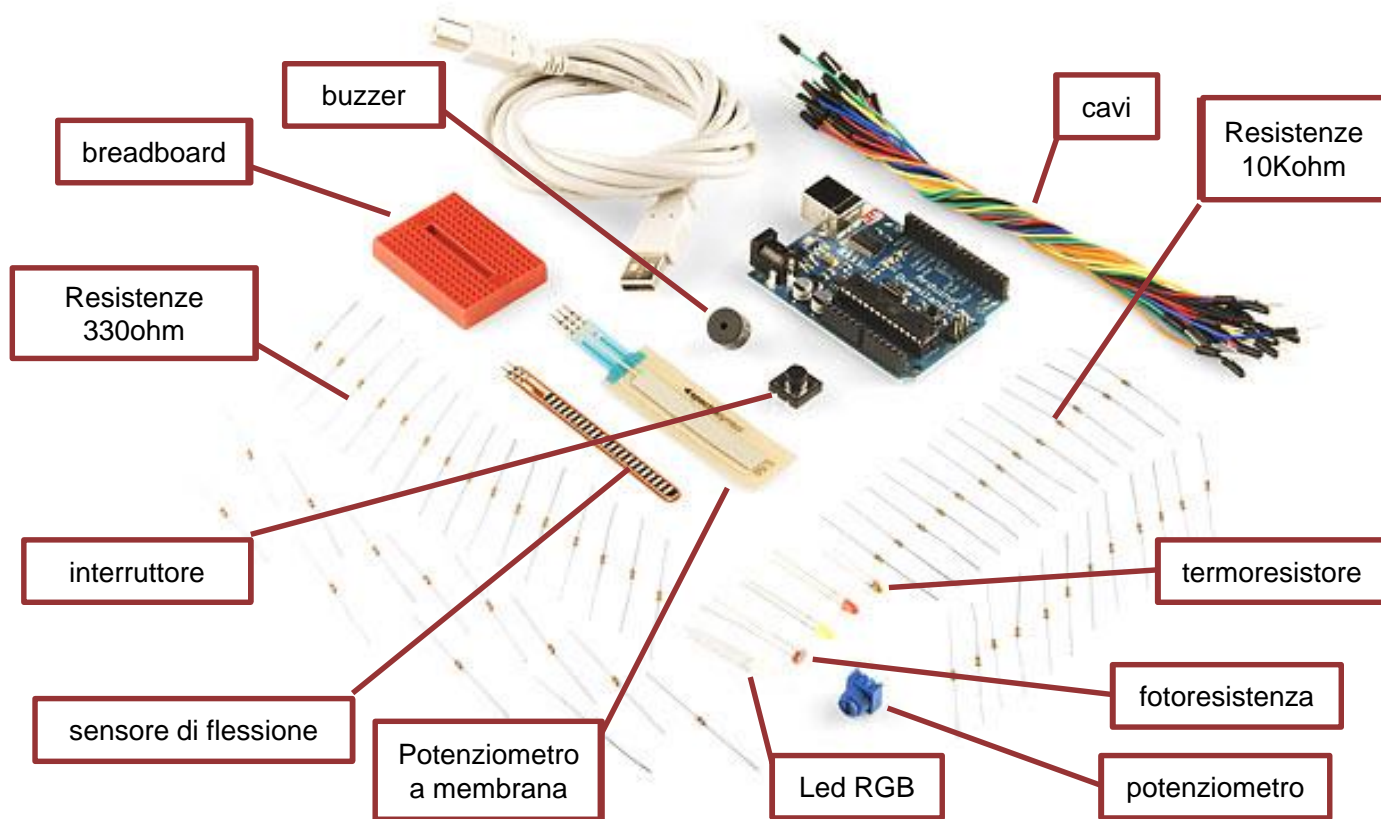
RFBee

FSK RF Module (CC1101/ATmega168 based) For adding simple RF capabilities to your Stalker. Onboard microcontroller (Arduino based) runs firmware for RF communication. Connects to the Stalker's microcontroller via UART. Accepts AT Commands.

UartSBee

USB-UART Bridge (FT232RL based)
For programming using Arduino and serial communication with PC. Can be used with other Bee modules to connect them to PC directly.

ARDUINO STARTER KIT



ARDUINO SD shield



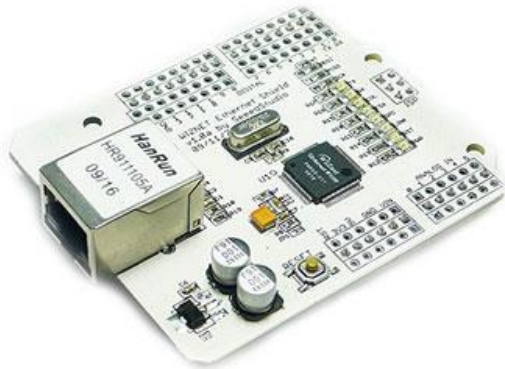
- Permette ai dispositivi come Arduino e Seeduino di leggere e scrivere le SD card con apposite [librerie disponibili gratuitamente](#).



SD shield è particolarmente interessante per la realizzazione di semplici data logger

ARDUINO

Ethernet shield (compatibile seeduino)



- Dispositivo basato sul chip ethernet Wiznet) W5100 ([datasheet](#))
- Permette di connettere una scheda Arduino o Seeeduino ad una LAN utilizzando la libreria [Ethernet library](#). *SeedWiznet* supporta fino a quattro connessioni socket simultanee.
- I pin digitali **10, 11, 12 e 13** di Arduino o Seeeduino vengono impiegati per comunicare con il chip W5100, quindi **quando si utilizza *SeedWiznet*, questi pin non possono essere utilizzati come I/O.**
- Il pulsante di reset sulla scheda resetta sia il chip W5100 che la scheda Arduino o Seeeduino.

Questa scheda è particolarmente interessante per la realizzazione di servizi web
(ad es pubblicazione dei dati raccolti)

ARDUINO ETHERNET SHIELD



Dispositivo basato sul chip ethernet
Wiznet W5100 ([datasheet](#))



permette di connettere una scheda Arduino ad una LAN
utilizzando la libreria [Ethernet library](#)



dispone di connettore per micro SD card



supporta fino a quattro connessioni socket simultanee



I pin digitali 10, 11, 12 e 13 di Arduino vengono impiegati
per comunicare con il chip W5100, quindi quando si
utilizza Arduino Ethernet Shield, questi pin non possono
essere utilizzati come I/O



Il pulsante di reset sulla scheda resetta sia il chip W5100
che la scheda Arduino



Compatibile anche con la scheda Arduino MEGA



Power Over Ethernet utilizzabile

ARDUINO SHIELD ARDUINO Wi-Fi

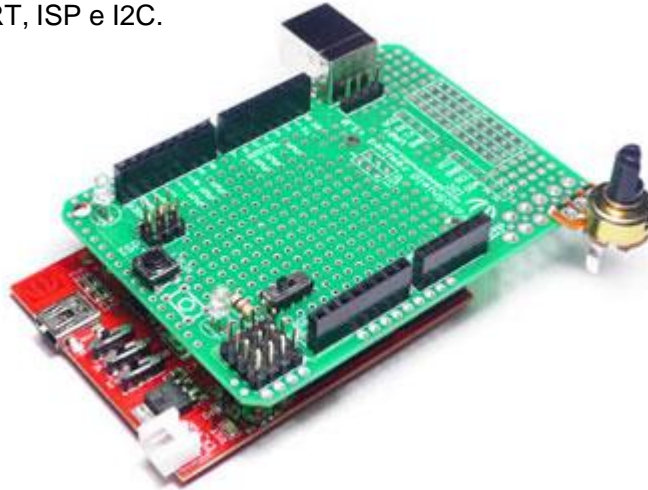


- Modulo transceiver Wi-Fi a 2,4 GHz standard 802.11 IEEE della Microchip
- antenna integrata su PCB e supporto integrato per l'hardware AES, TKIP e (WEP, WPA , WPA2)
- copertura di qualche decina di metri
- Consente l'uso di microSD card
- Alimentazione 12 Vdc fornita direttamente dalla scheda Arduino



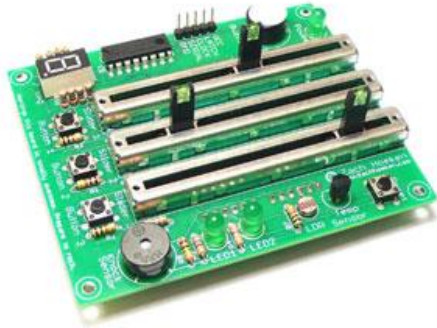
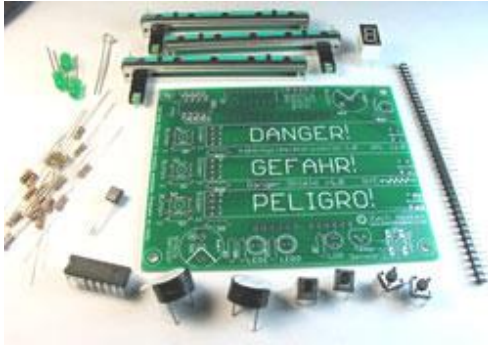
ARDUINO PROTO SHIELD

Piastra sperimentale (58,50 x 82,70 mm) per piccole applicazioni, realizzata appositamente per le schede Arduino o Seeduino, permette di avere un numero maggiore di piazzole su cui montare i componenti. Alcuni piazzole sono predisposte per montare un connettore USB tipo B, un mini potenziometro da stampato, pulsanti, LED, ecc. Dispone di piazzole riservate al montaggio di connettori per UART, ISP e I2C.



- 2 Pz. Connettore maschio 40 pin
- 1 Pz. Connettore femmina 40 pin
- 3 Pz. - Connettore femmina 8 pin
- 1 Pz. - Connettore femmina 6 pin
- 1 Pz. Connettore doppio maschio 40 pin per Arduino/Seeduino
- 1 Pz. Connettore femmina 3 pin x 2 file
- 1 Pz. Connettore maschio 3 pin x 2 file
- 4 Pz. Mini pulsante da c.s.
- 1 Pz. Connettore USB femmina tipo B
- 1 Pz. - Potenziometro 10 kohm montaggio verticale
- 2 Pz. Doppio mini deviatore da c.s.
- 1 Pz. - LED bicolore 3 mm
- 2 Pz. - LED rosso 3 mm
- 2 Pz. - LED verde 3 mm
- 4 Pz. - Resistenza 1kohm
- 2 Pz. - Resistenza 10 kohm

ARDUINO DANGER SHIELD



● Montata sopra una scheda Arduino o Seeduino, **permette di testare i vari ingressi/uscite**

● **Viene fornita in kit (va montata)**

Il KIT contiene:

● **3 Slider lineari** con LED integrati, 4 pulsanti,

● **3 LED indicatori**, 1 **Buzzer**,

● **1 Sensore di temperatura**,

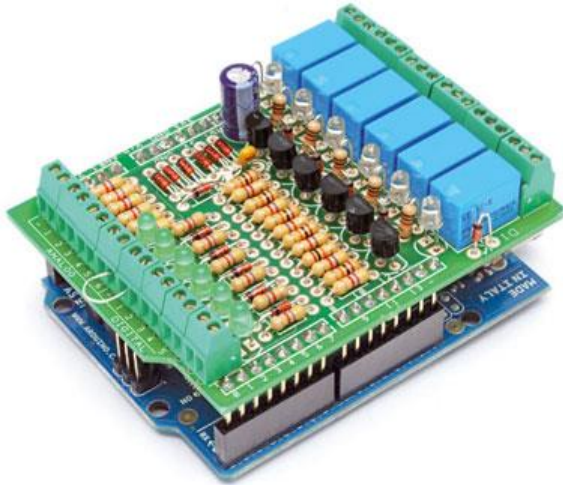
● **1 Fotoresistenza**,

● **1 knock sensor**,

● **1 Display a 7 segmenti** e

● **1 integrato 74HC595N (shift register).**

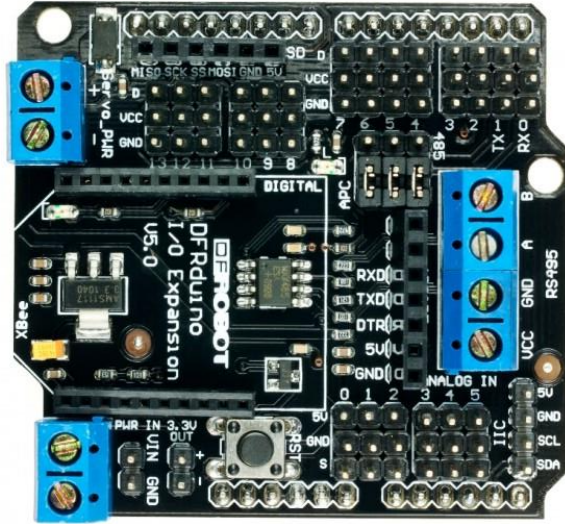
SHIELD ARDUINO - IN/OUT



- 6 relay
- 6 digital input
- 6 analog input
- The digital inputs and relay outputs are equipped with an LED that indicates the status
- The lines of I/O are connected to the Arduino through corresponding pin-strip pitch 2.54 mm
- NB The mini relay works at 12V

Futura elettronica <http://www.futurashop.it>
assemblaggio: http://www.futurashop.it/Allegato_PDF_IT/7100-FT919K.pdf
esempio: <http://www.futurashop.it/image/Download/7100-FT919K.zip>

DFROBOT IO Expansion Shield per Arduino (V5)



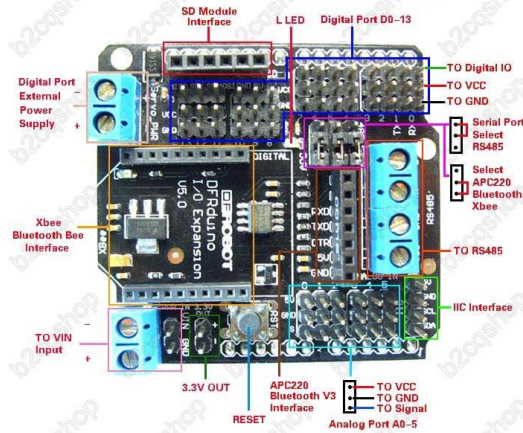
Robot domestici: <http://www.robot-domestici.it>
Robot Italy: <http://www.robot-italy.com/>
Emmeshop: <http://www.emmeshop.it/>
Dfrobot: www.dfrobot.com

Specification

- Support RS485
- Support Xbee (Xbee pro)
- Support Bluetooth
- Support APC220
- Support SD card read/write

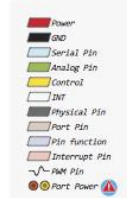


Sensor IO Expansion V5 Explained



Info: <http://www.funnyrobotics.com/2011/04/arduino-with-bluetooth-wireless.html>

FISHINO
www.fishino.it/en
PINOUT



⚠ The power sum for each pin's group should not exceed 120mA.

Internet Of Things? Fishino!

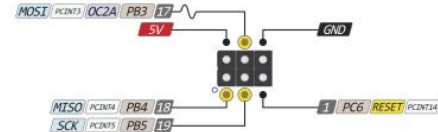
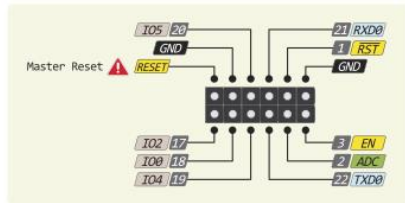
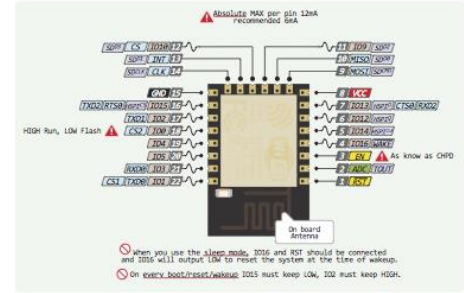
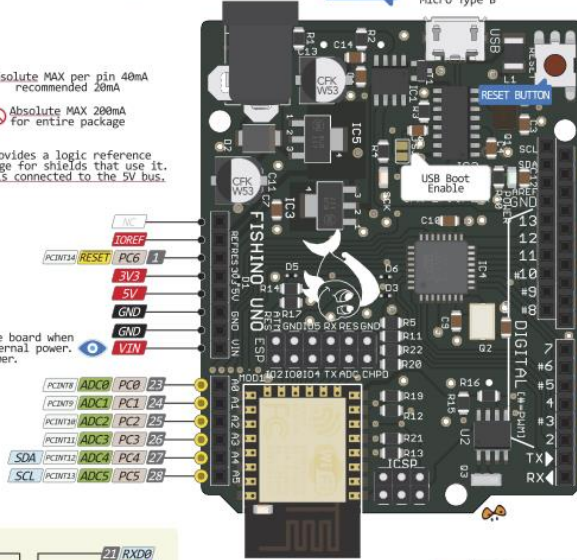
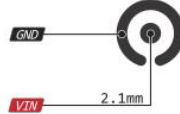
Open your projects to the web with Fishino!

⚠ Absolute MAX per pin 40mA recommended 20mA
⚠ Absolute MAX 200mA for entire package

⚠ **IOREF** provides a logic reference voltage for shields that use it. It is connected to the 5V bus.

The input voltage to the board when it is running from external power. Not USB bus power.

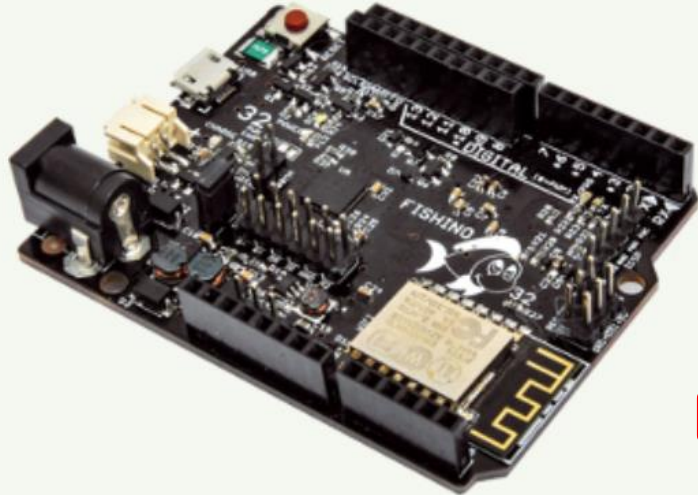
👁 7-12V Depending on current draw



FISHINO

Fishino32

The 32 bit Fishino board



- » High performance PIC32MX controller
- » high performance, with 120 MHz clock, 512 KBytes flash and 128 KBytes RAM
- » WiFi module on board
- » MicroSD card reader on board
- » RTC module on board
- » Stereo audio codec on board
- » High performance switching power supply
- » Additional pinheader connector for breadboard compatibility
- » Can be powered by a single cell LiPo battery, with on board charger
- » Can be powered off by software and awoken by external events

Esempio «hallo world» – programma arduino per led lampeggiante

```
int dt1=1000;  
int dt2=dt1;
```

// the setup function runs once when you press reset or power the board

```
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}
```

// the loop function runs over and over again forever

```
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(dt1); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(dt2); // wait for a second  
}
```

Esempio – programma arduino per la lettura di un dato con ADC

```
const int analogInPin = A0; // Analog input pin that the potentiometer is
attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to
```

```
int sensorValue = 0; // value read from the pot
int outputValue = 0; // value output to the PWM (analog out)
```

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

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print(sensorValue);
  Serial.print("\t");
  Serial.println(outputValue);

  delay(5);
}
```

Esempio – tecnica alla base di un timing «non bloccante»

```
const int ledPin = LED_BUILTIN;// the number of the LED pin
int ledState = LOW;          // ledState used to set the LED
unsigned long previousMillis = 0;    // will store last time LED was updated
const long interval = 1000;        // interval at which to blink (milliseconds)
```

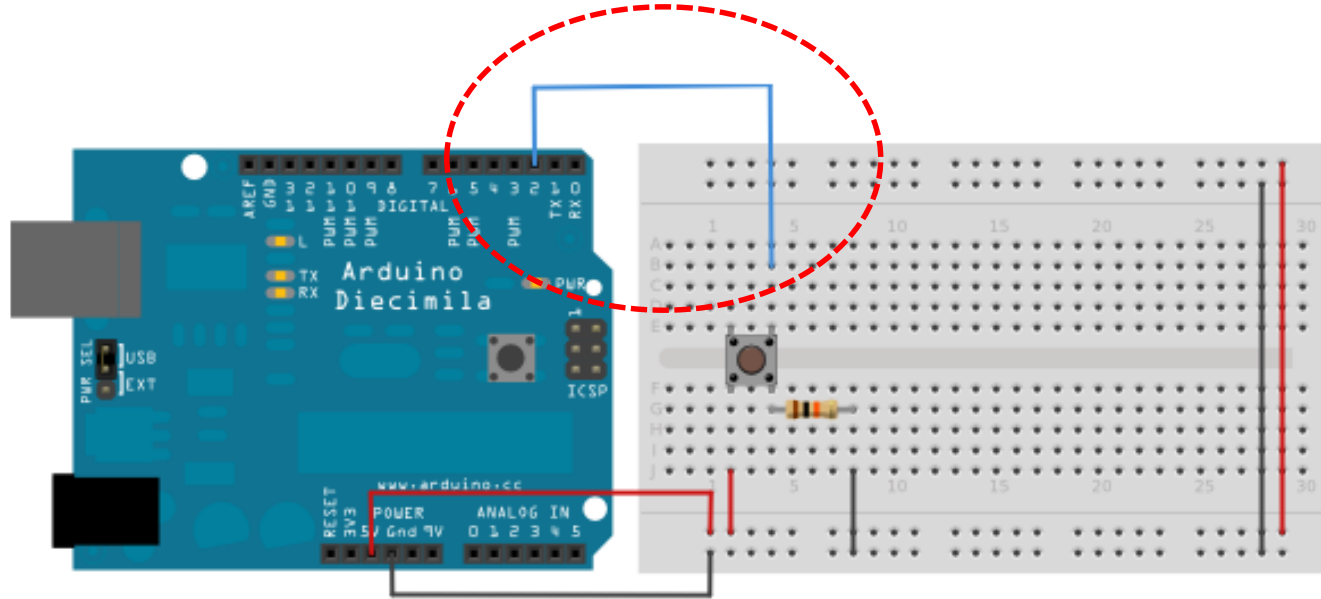
```
void setup() {
  pinMode(ledPin, OUTPUT);
}
```

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }
    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```



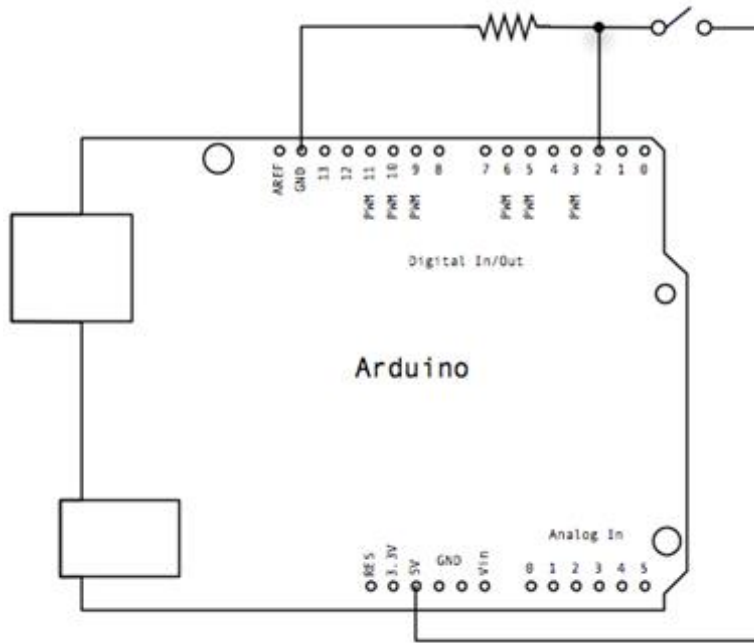
ARDUINO

Rilevare lo stato di un interruttore



ARDUINO

Rilevare lo stato di un interruttore



```
const int buttonPin = 2; // the number of the
pushbutton pin
const int ledPin = 13; // the number of the LED pin
```

```
// variables will change:
int buttonState = 0; // variable for reading the
pushbutton status
```

```
void setup() {
// initialize the LED pin as an output:
pinMode(ledPin, OUTPUT);
// initialize the pushbutton pin as an input:
pinMode(buttonPin, INPUT);
}
```

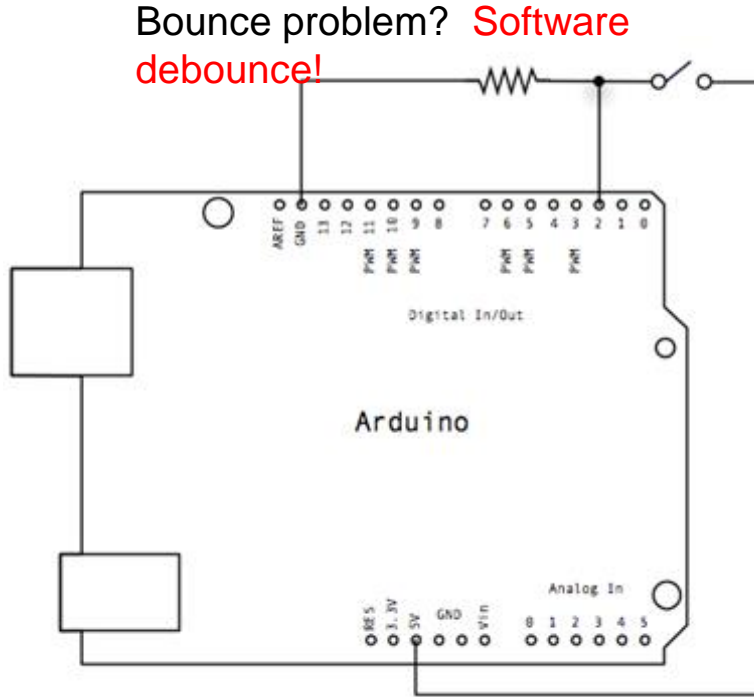
```
void loop(){
// read the state of the pushbutton value:
buttonState = digitalRead(buttonPin);
```

```
// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
if (buttonState == HIGH) {
// turn LED on:
digitalWrite(ledPin, HIGH);
}
```

```
else {
// turn LED off:
digitalWrite(ledPin, LOW);
}
}
```

ARDUINO

Rilevare lo stato di un interruttore



```
void setup() {  
  pinMode(buttonPin, INPUT);  
  pinMode(ledPin, OUTPUT);  
}  
void loop() {  
  // read the state of the switch into a local variable:  
  int reading = digitalRead(buttonPin);  
  
  // check to see if you just pressed the button  
  // (i.e. the input went from LOW to HIGH), and you've waited  
  // long enough since the last press to ignore any noise:  
  
  // If the switch changed, due to noise or pressing:  
  if (reading != lastButtonState) {  
    // reset the debouncing timer  
    lastDebounceTime = millis();  
  }  
  if ((millis() - lastDebounceTime) > debounceDelay) {  
    // whatever the reading is at, it's been there for longer  
    // than the debounce delay, so take it as the actual current state:  
    buttonState = reading;  
  }  
  // set the LED using the state of the button:  
  digitalWrite(ledPin, buttonState);  
  // save the reading. Next time through the loop,  
  // it'll be the lastButtonState:  
  lastButtonState = reading;  
}
```

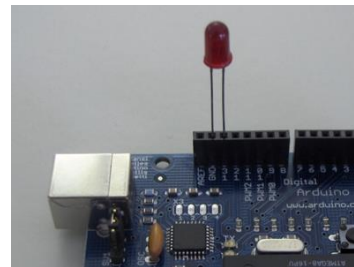
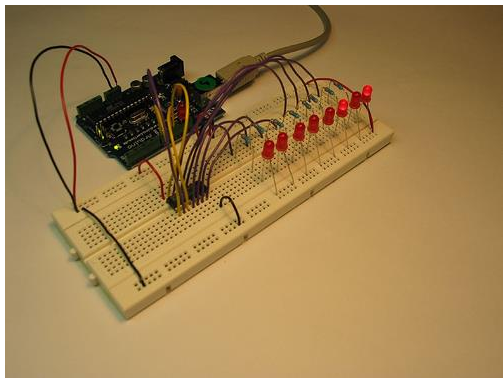
ARDUINO

Controllare un led

Viene fatto lampeggiare un LED

Utilizziamo il pin 13 perche' ha una resistenza di pull up in serie (per questo è sufficiente un LED ed un programma)

Con opportune **resistenze è possibile pilotare più LED** con un programma
ad es con piu' DigitalWrite(...)



```
int ledPin = 13;           // LED connected to digital pin 13

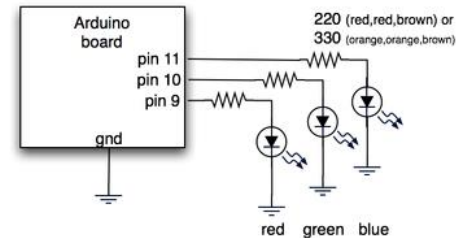
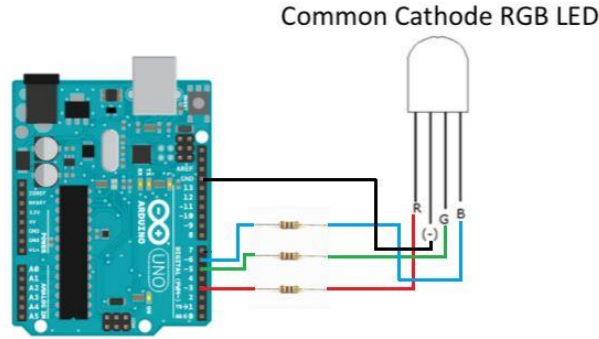
void setup()
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                // waits for a second
}
```


ARDUINO

Controllare un led RGB

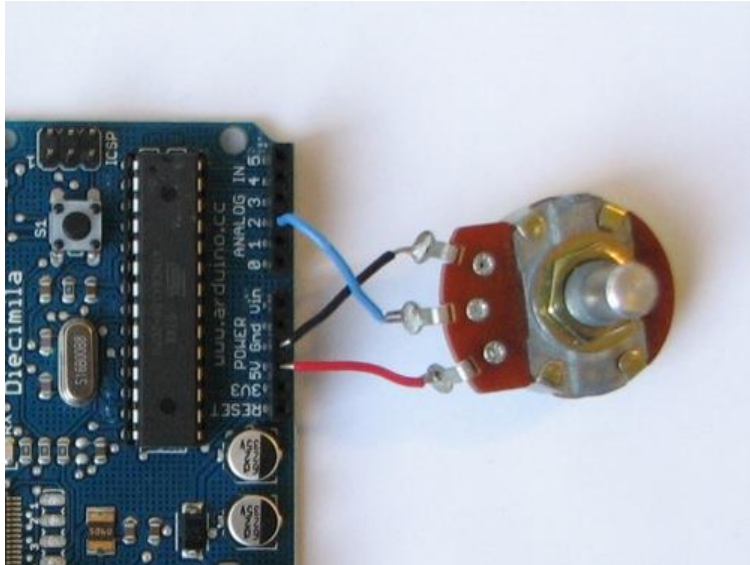
```
int sensorPin = 3;
int bluePin = 6;
int greenPin = 5;
int redPin = 3;
int sensorValue = 0;
void setup (){
  pinMode (redPin, OUTPUT);
  pinMode (greenPin, OUTPUT);
  pinMode (bluePin, OUTPUT);
  Serial.begin (9600);
}
void loop (){
  sensorValue = analogRead (sensorPin);
  int ledFadeValue = map(sensorValue,0, 1023, 0, 255);
  int ledFadeValue2 = map(sensorValue,0, 1023, 0, 255 );
  int ledFadeValue3 = map(sensorValue,0, 1023, 255, 0);
  analogWrite (3, ledFadeValue);
  analogWrite (5, ledFadeValue2);
  analogWrite (6, ledFadeValue3);
  delay (20);
}
```



ARDUINO

Rilevare una resistenza variabile

Il circuito è molto semplice:
un potenziometro collegato come segue:
PIN1 → 5V
PIN2 → ANALOG0
PIN3 → Ground



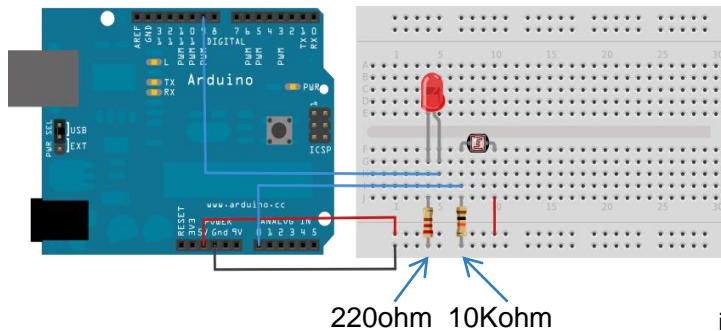
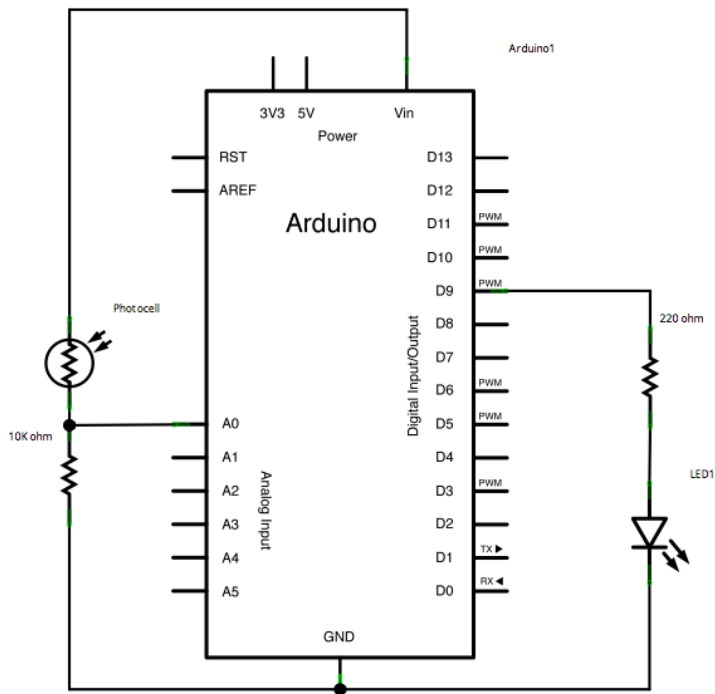
```
int potPin = 2; // select the input pin for the potentiometer
int ledPin = 13; // select the pin for the LED
int val = 0; // variable to store the value coming from the sensor
```

```
void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT
}
```

```
void loop() {
  val = analogRead(potPin); // read the value from the sensor
  digitalWrite(ledPin, HIGH); // turn the ledPin on
  delay(val); // stop the program for some time
  digitalWrite(ledPin, LOW); // turn the ledPin off
  delay(val); // stop the program for some time
}
```

ARDUINO

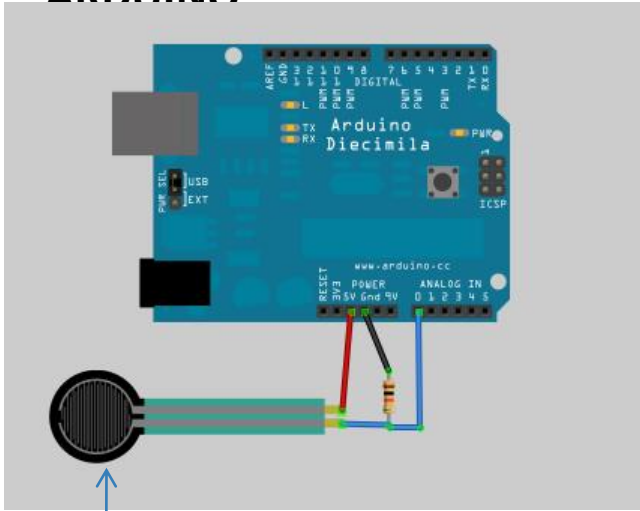
Rilevare una misura di luminosità



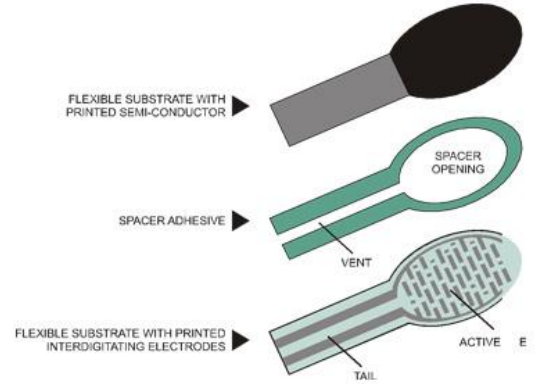
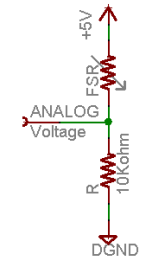
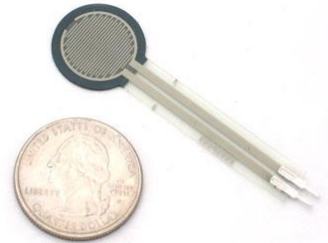
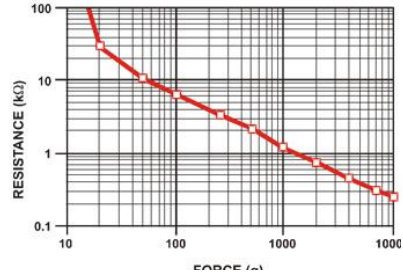
```
int sensorMin = 1023; //
minimum sensor value
int sensorMax = 0; //
maximum sensor value
// calibrate during the first five
seconds
while (millis() < 5000) {
  sensorValue =
  analogRead(sensorPin);
  // record the maximum sensor
  value
  if (sensorValue > sensorMax) {
    sensorMax = sensorValue;
  }
  // record the minimum sensor
  value
  if (sensorValue < sensorMin) {
    sensorMin = sensorValue;
  }
}
```

È possibile usare la funzione **map** per la **calibrazione**:
`sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);`

ARDUINO

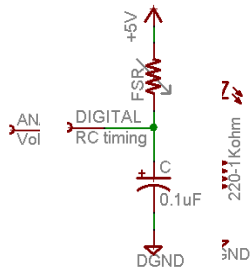
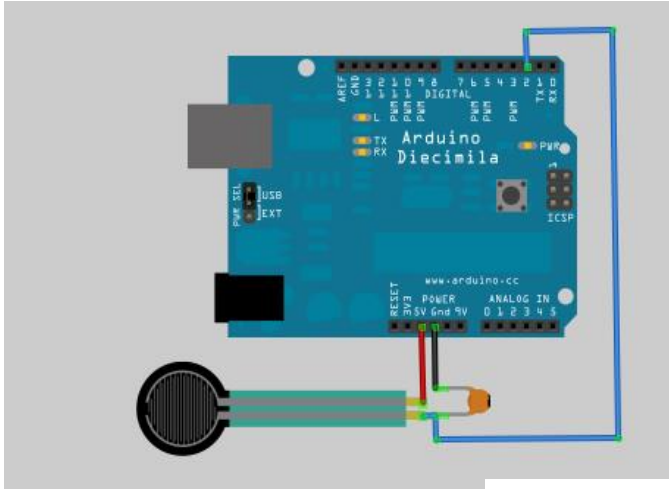


FSR = Force Sensitive Resistor



ARDUINO

Controllo di un LED mediante FSR e PWM



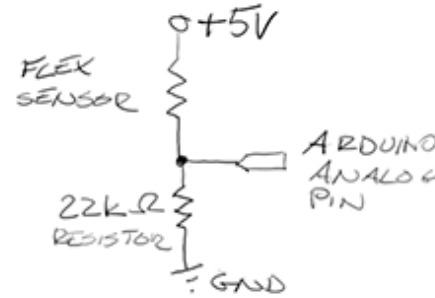
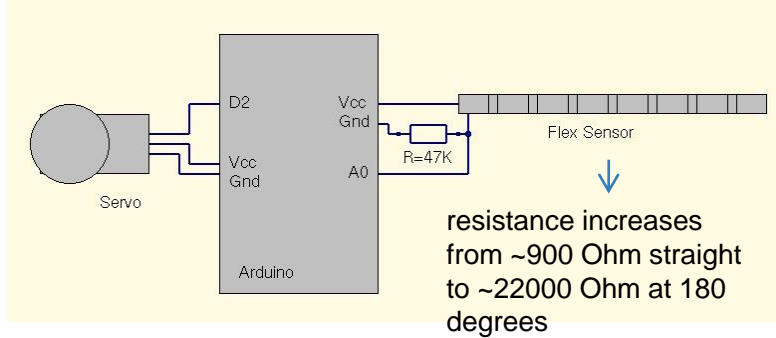
```
int fsrAnalogPin = 0; // FSR is connected to analog 0
int LEDpin = 11; // connect Red LED to pin 11
(PWM pin)
int fsrReading; // the analog reading from the FSR
resistor divider
int LEDbrightness;
```

```
void setup(void) {
  Serial.begin(9600); // We'll send debugging
information via the Serial monitor
  pinMode(LEDpin, OUTPUT);
}
void loop(void) {
  fsrReading = analogRead(fsrAnalogPin);
  Serial.print("Analog reading = ");
  Serial.println(fsrReading);
```

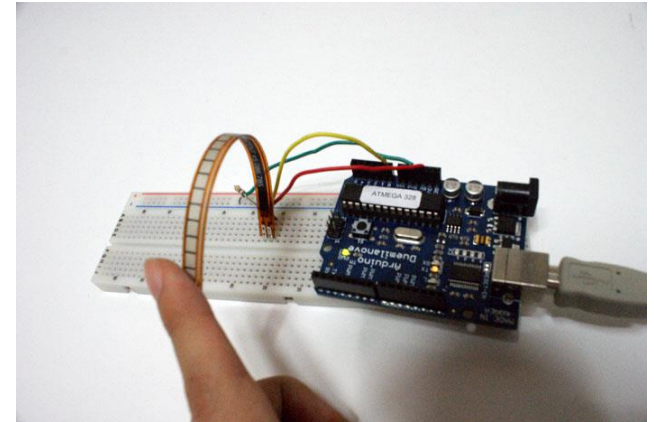
```
// we'll need to change the range from the analog
reading (0-1023) down to the range
// used by analogWrite (0-255) with map!
LEDbrightness = map(fsrReading, 0, 1023, 0, 255);
// LED gets brighter the harder you press
analogWrite(LEDpin, LEDbrightness);
delay(100);
}
```

ARDUINO

Misura di flessione

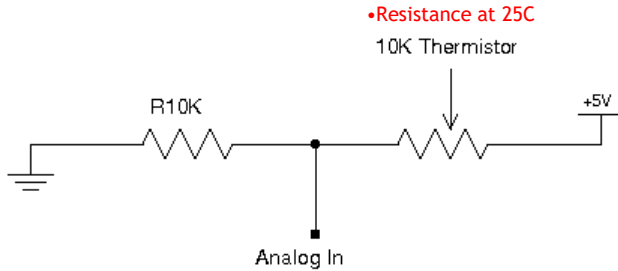


```
int CdS_PIN = 0; // select the input pin for the CdS Cell
int ledPIN = 13; // select the output pin for the LED
int CdS_VAL = 0; // variable to store the value coming from the CdS Cell
void setup() {
  Serial.begin(9600); // set the baud rate for the serial window
  pinMode(ledPIN, OUTPUT); // declare the ledPin as an OUTPUT
  // ANALOG IN is an input by default
}
void loop() {
  CdS_VAL = analogRead(CdS_PIN); // read the value from the CdS Cell
  CdS_VAL = (CdS_VAL/2 + 16); // shift the CdS_VAL to be a useful flashrate
  // (barely visible to 511ms)
  digitalWrite(ledPIN, HIGH); // set the ledPIN HIGH, which turns on the LED
  delay(CdS_VAL); // do nothing for (CdS_VAL) milliseconds
  digitalWrite(ledPIN, LOW); // set the ledPIN LOW, which turns off the LED
  delay(CdS_VAL); // do nothing for (CdS_VAL) milliseconds
  Serial.println(CdS_VAL); // print CdS_VAL in the serial arduino serial window
}
```



ARDUINO

Misurare la temperatura con resistenze NTC



I termistori non hanno una risposta lineare:

$$R=R0 \cdot e^{\{B/T-B/T0\}}$$

R0 (4.7K) è la resistenza ad una temperatura nota T0, in Kelvin.

B (beta) è un parametro del termistore che può essere calcolato a partire da una misura oppure trovato nei [datasheet](#)

Quando conviene utilizzare un sensore di temperatura come LM35

```
int sensor=0; //analog pin
float temp;
float adc; //value of analog pin
float adcVolts; //value of analog pin in volts
//thermistor parameter
float beta=3950.0; //beta
float r0=4700.0; //resistance at t0 temperature
float t0=25.0 + 273.15; //Kelvin
float r1=10000.0;
float r2=4400.0;
float r; //r1 parallel r2
float rT; //resistance of thermistor
float k; //constant part of exponential equation
float vs; //effettive bias voltage
void setup(){
  Serial.begin(9600);
  r=( r1 * r2 ) / ( r1 + r2); //r1 parallel r2
  k = r0 * exp(-beta / t0); //constant part of exponential
  equation
  vs= ( r1 * 5.0 ) / ( r1 + r2); //effective bias voltage
}
void loop(){
  adc=analogRead(sensor);
  adcVolts = adc * 5.0 / 1023; // convert the 10 bit ADC
  value to a voltage
  rT = adcVolts / ((vs - adcVolts)/r); // resistance of
  thermistor
  temp = beta / log(rT / k) -273.15; //Celsius
  Serial.print(temp,DEC);
  Serial.println("°C");
}
```


ARDUINO

Misurazione temperatura con TMP36

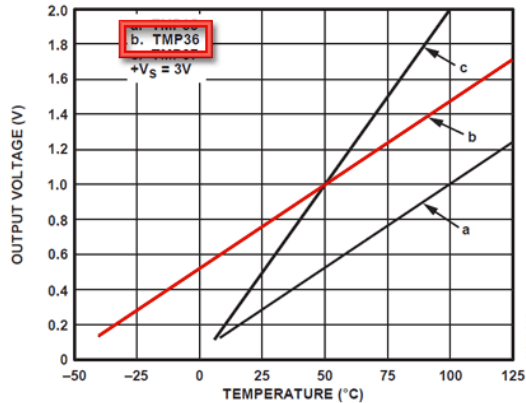
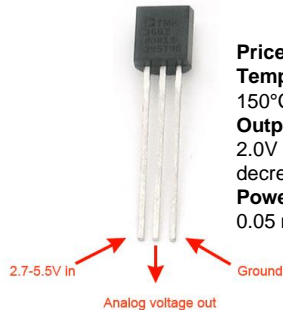


Figure 6. Output Voltage vs. Temperature



Price: [\\$2.00 at the Adafruit shop](#)

Temperature range: -40°C to 150°C / -40°F to 302°F

Output range: 0.1V (-40°C) to 2.0V (150°C) but accuracy decreases after 125°C

Power supply: 2.7V to 5.5V only, 0.05 mA current draw

```
//TMP36 Pin Variables
int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected
to
//the resolution is 10 mV / degree centigrade with a
//500 mV offset to allow for negative temperatures
/*
 * setup() - this function runs once when you turn your Arduino on
 * We initialize the serial connection with the computer
 */
void setup()
{
  Serial.begin(9600); //Start the serial connection with the computer
  //to view the result open the serial monitor
}
void loop() // run over and over again
{
  //getting the voltage reading from the temperature sensor
  int reading = analogRead(sensorPin);
  // converting that reading to voltage, for 3.3v arduino use 3.3
  float voltage = reading * 5.0 / 1024;
  // print out the voltage
  Serial.print(voltage); Serial.println(" volts");

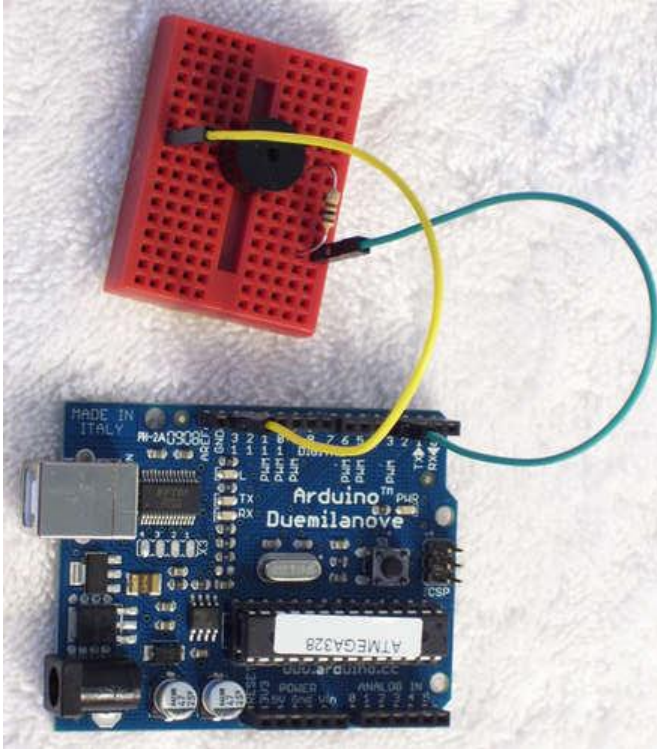
  // now print out the temperature
  float temperatureC = (voltage - 0.5) * 100; //converting from 10 mv per
  degree wit 500 mV offset
  //to degrees ((voltage - 500mV) times 100)
  Serial.print(temperatureC); Serial.println(" degress C");

  // now convert to Fahrenheight
  float temperatureF = (temperatureC * 9 / 5) + 32;
  Serial.print(temperatureF); Serial.println(" degress F");

  delay(1000); //waiting a second
}
```

ARDUINO

Controllare un buzzer



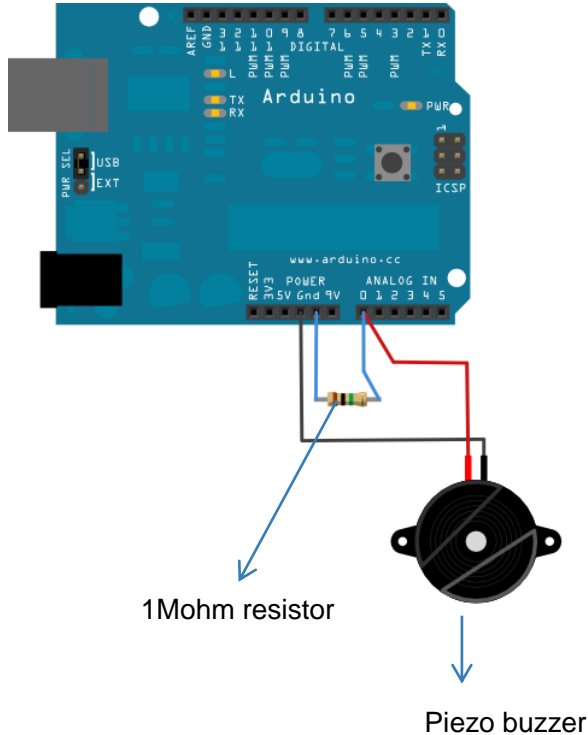
```
void setup()
{
  pinMode(11, OUTPUT); // pin as
  output
}
void loop()
{
  analogWrite(11,128);
  delay(500);
  digitalWrite(11, LOW);
  delay(500);
}
```

The sketch above will beep the piezo on and off, and be somewhat annoying. Perfect.

However with the *analogWrite()* function it is impossible to use the full frequency range of the piezo buzzer.

With a value of 254 for the duty cycle, the frequency generated is around 1500 Hz:

ARDUINO Knock sensor



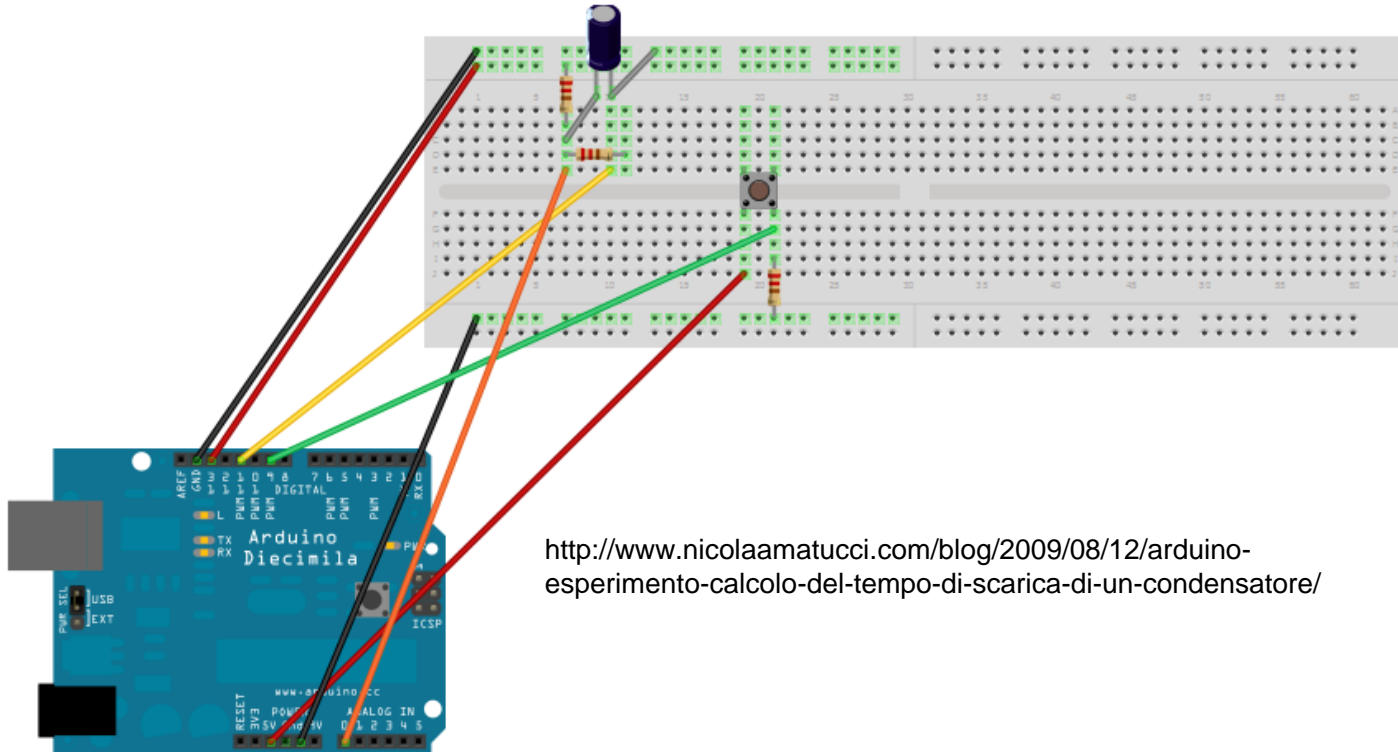
```
const int ledPin = 13; // led connected to digital pin 13
const int knockSensor = A0; // the piezo is connected to analog
pin 0
const int threshold = 100; // threshold value to decide when the
detected sound is a knock or not
// these variables will change:
int sensorReading = 0; // variable to store the value read
from the sensor pin
int ledState = LOW; // variable used to store the last LED
status, to toggle the light
```

```
void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as a
OUTPUT
  Serial.begin(9600); // use the serial port
}
void loop() {
  // read the sensor and store it in the variable sensorReading:
  sensorReading = analogRead(knockSensor);

  // if the sensor reading is greater than the threshold:
  if (sensorReading >= threshold) {
    // toggle the status of the ledPin:
    ledState = !ledState;
    // update the LED pin itself:
    digitalWrite(ledPin, ledState);
    // send the string "Knock!" back to the computer, followed by
newline
    Serial.println("Knock!");
  }
  delay(100); // delay to avoid overloading the serial port buffer
```

ARDUINO

Carica di un condensatore



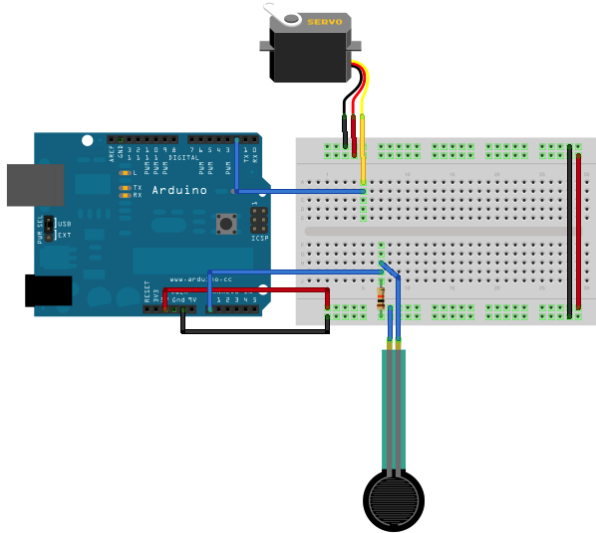
<http://www.nicolaamatucci.com/blog/2009/08/12/arduino-esperimento-calcolo-del-tempo-di-scarica-di-un-condensatore/>

ARDUINO

Controllo di un servomotore (pulse)



Un servomotore (o servo) è un motore che può essere mosso di un angolo variabile tramite un solo segnale di controllo.



```
int servoPin = 2; // Control pin for servo motor
int minPulse = 500; // Minimum servo position
int maxPulse = 2500; // Maximum servo position
int pulse = 0; // Amount to pulse the servo
```

```
long lastPulse = 0; // the time in milliseconds of the last pulse
int refreshTime = 20; // the time needed in between pulses
```

```
int analogValue = 0; // the value returned from the analog sensor
int analogPin = 0; // the analog pin that the sensor's on
```

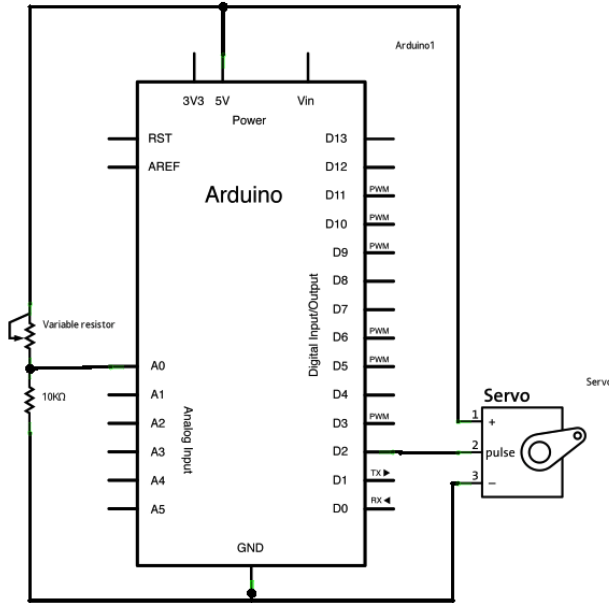
```
void setup() {
  pinMode(servoPin, OUTPUT); // Set servo pin as an output pin
  pulse = minPulse; // Set the motor position value to the
  minimum
  Serial.begin(9600);
}
```

```
void loop() {
  analogValue = analogRead(analogPin); // read the analog input
  pulse = map(analogValue,0,1023,minPulse,maxPulse); // convert
  the analog value
  // to a range between minPulse
  // and maxPulse.
```

```
// pulse the servo again if the refresh time (20 ms) have passed:
if (millis() - lastPulse >= refreshTime) {
  digitalWrite(servoPin, HIGH); // Turn the motor on
  delayMicroseconds(pulse); // Length of the pulse sets the motor
  position
  digitalWrite(servoPin, LOW); // Turn the motor off
  lastPulse = millis(); // save the time of the last pulse
}
}
```

ARDUINO

Controllo di un servomotore (servo.h)



```
#include <Servo.h> // include the servo library
```

```
Servo servoMotor; // creates an instance of the servo object to control a servo
```

```
int analogPin = 0; // the analog pin that the sensor is on  
int analogValue = 0; // the value returned from the analog sensor
```

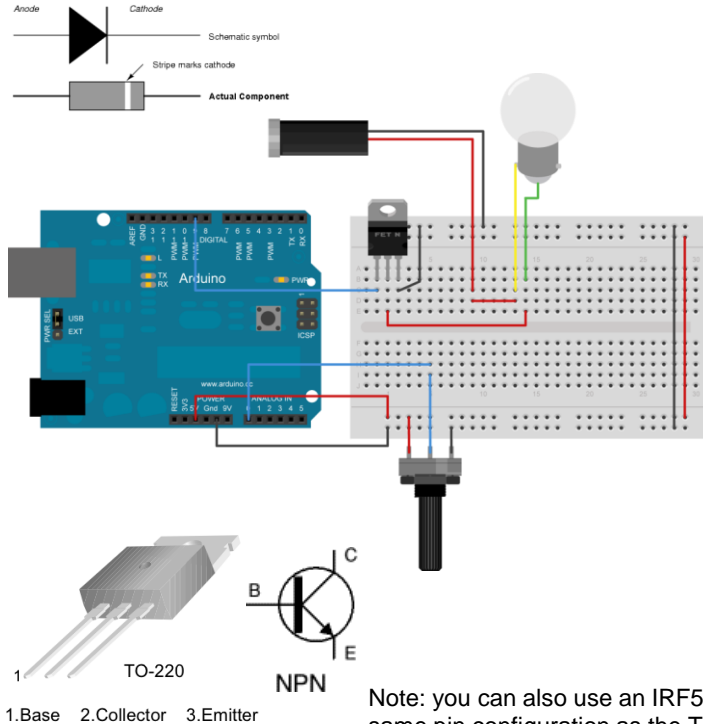
```
int servoPin = 2; // Control pin for servo motor. As of Arduino 0017, can be any pin
```

```
void setup() {  
  servoMotor.attach(servoPin); // attaches the servo on pin 2 to the servo object  
}
```

```
void loop()  
{  
  analogValue = analogRead(analogPin); // read the analog input (value between 0 and 1023)  
  analogValue = map(analogValue, 0, 1023, 0, 179); // map the analog value (0 - 1023) to the angle of the servo (0 - 179)  
  servoMotor.write(analogValue); // write the new mapped analog value to set the position of the servo  
  delay(15); // waits for the servo to get there  
}
```

CARDUINO

Controllo di un carico resistivo



```
const int transistorPin = 9;  
// connected to the base of the transistor
```

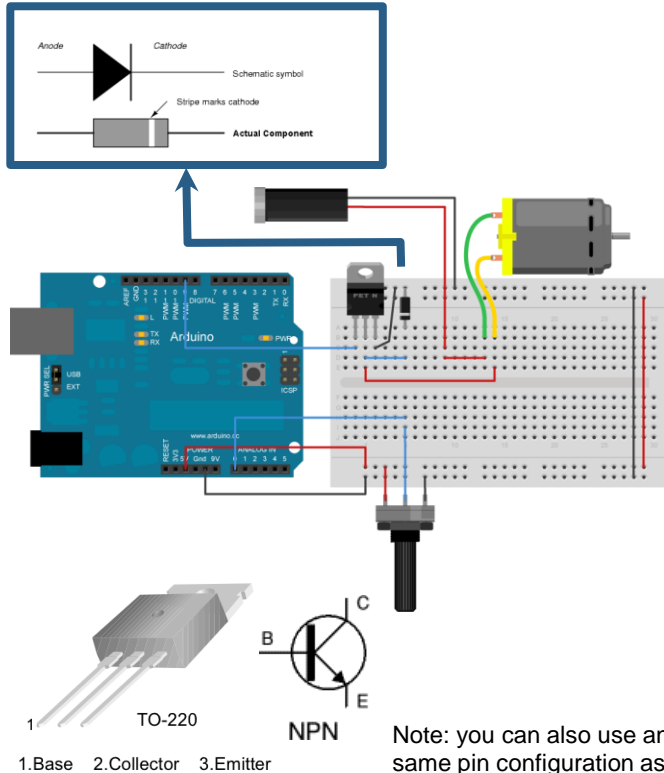
```
void setup() {  
  // set the transistor pin as output:  
  pinMode(transistorPin, OUTPUT);  
}
```

```
void loop() {  
  digitalWrite(transistorPin, HIGH);  
  delay(1000);  
  digitalWrite(transistorPin, LOW);  
  delay(1000);  
}
```

Note: you can also use an IRF510 or IRF520 MOSFET transistor for this. They have the same pin configuration as the TIP120, and perform similarly. They can handle more amperage and voltage, but are more sensitive to static electricity damage

ARDUINO

Controllo di un carico induttivo



```
const int potPin = 0;           // Analog in 0 connected to the
                                // potentiometer
const int transistorPin = 9;    // connected to the base of the
                                // transistor
int potValue = 0;              // value returned from the
                                // potentiometer
```

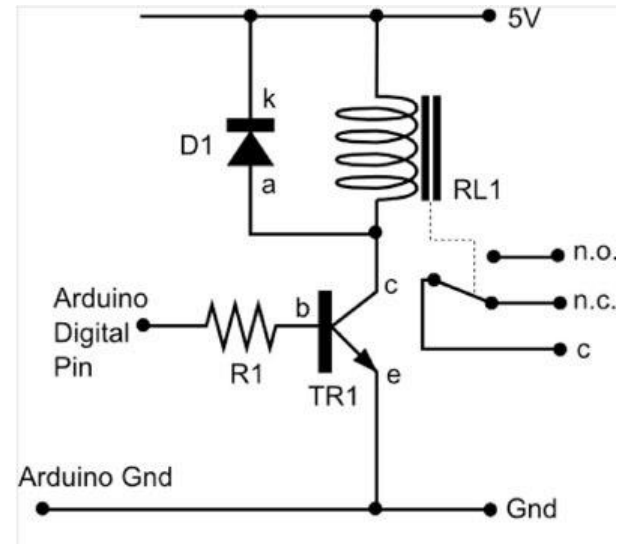
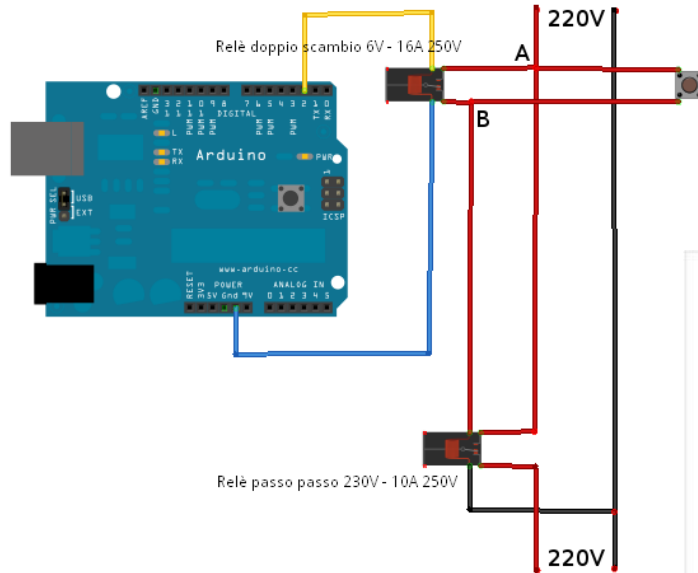
```
void setup() {
  // set the transistor pin as output:
  pinMode(transistorPin, OUTPUT);
}
```

```
void loop() {
  // read the potentiometer, convert it to 0 - 255:
  potValue = analogRead(potPin) / 4;
  // use that to control the transistor:
  analogWrite(9, potValue);
}
```

Note: you can also use an IRF510 or IRF520 MOSFET transistor for this. They have the same pin configuration as the TIP120, and perform similarly. They can handle more amperage and voltage, but are more sensitive to static electricity damage

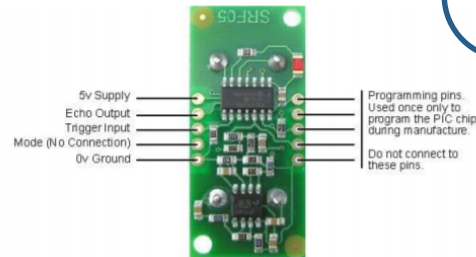
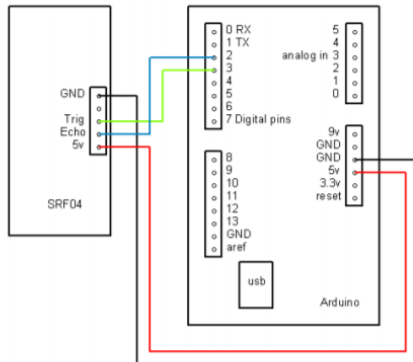
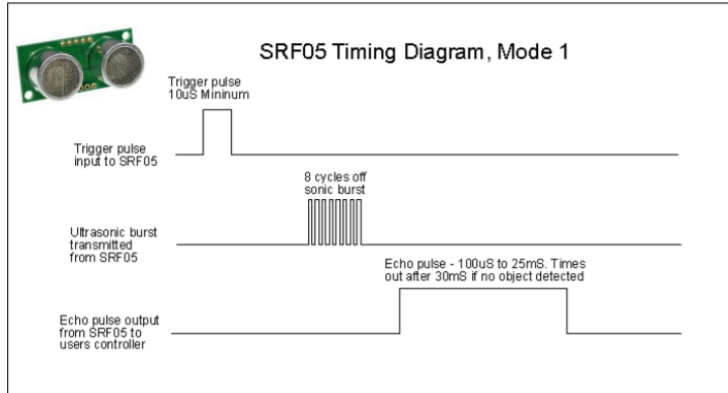
ARDUINO

Controllare un relè



Misurare distanze con un sensore ad ultrasuoni

Pinger ultrasuoni



Connections for 2-pin Trigger/Echo Mode (SRF04 compatible)

```
#define ECHOPIN 2
```

```
#define TRIGPIN 3
```

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

```
void loop() {
```

```
  // Set the trigger pin to low for 2uS  
  digitalWrite(TRIGPIN, LOW);  
  delayMicroseconds(2);  
  // Send a 10uS high to trigger ranging  
  digitalWrite(TRIGPIN, HIGH);  
  delayMicroseconds(10);  
  // Send pin low again  
  digitalWrite(TRIGPIN, LOW);  
  // Read in times pulse  
  int distance = pulseIn(ECHOPIN, HIGH);  
  // Calculate distance from time of pulse  
  // cs=340 m/s  
  distance = distance/58;  
  Serial.println(distance);  
  delay(50); // Wait 50mS before next ranging  
}
```


Temperatura con sonda DS18B20



[Robot-domestici](#)

[Robot Italy](#)

[Datasheet](#)

[Libreria onewire](#)

```
#include <OneWire.h>
#include <DallasTemperature.h>
// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 2
// Setup a oneWire instance to communicate with any OneWire devices (not
just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
// Pass our oneWire reference to DallasTemperature.
DallasTemperature sensors(&oneWire);
void setup(void) {
  // start serial port
  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");

  // Start up the library
  // IC Default 9 bit. If you have troubles consider upping it 12.
  // Ups the delay giving the IC more time to process the temperature
  measurement
  sensors.begin();
}
void loop(void) {
  // call sensors.requestTemperatures() to issue a global temperature
  // request to all devices on the bus
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperatures
  Serial.println("DONE");

  Serial.print("Temperature for Device 1 is: ");
  // Why "byIndex"? You can have more than one IC on the same bus
  // 0 refers to the first IC on the wire
  Serial.print(sensors.getTempCByIndex(0));
}
```

Schema di traccia per esperienze di laboratorio nella scuola secondaria

Traccia

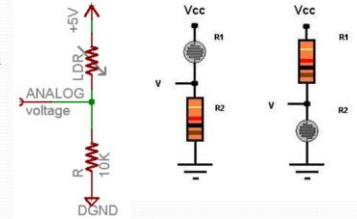
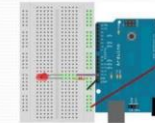
1. Realizzare un programma per arduino che legga i dati sul canale analogico o
2. connettere uno **slider** (potenziometro) ad arduino ed utilizzare il programma per acquisire i dati
3. Realizzare un grafico di $R_s=R_{slider}(t)$ con processing
4. Realizzare un circuito su breadboard con una fotoresistenza in un partitore di tensione e connettere la fotoresistenza ad arduino al canale analogico 1
5. Modificare il programma per leggere i canali 0 ed 1 trasmettere i valori letti a processing
6. Realizzare un grafico di $R_f=R_{fotoresistenza}(t)$ (oscurare il trasduttore...)
7. Aggiungere un led la cui frequenza di oscillazione è regolata dallo slider (lineare)
8. orientare la fotoresistenza in modo che riceva la luce emessa dal led e analizzare di $R_f=R_{fotoresistenza}(t)$ al variare della frequenza

Materiali

- arduino
- Fotoresistenza, resistenza da 5-10 kohm
- potenziometro slider
- Processing
- LED, resistenza da 330 ohm

Metodi

- per utilizzare la fotoresistenza è necessario realizzare un **partitore di tensione** (occorre limitare la corrente)
- Per non danneggiare il LED occorre limitare la corrente



$$V = V_{CC} * \frac{R_2}{R_1 + R_2}$$

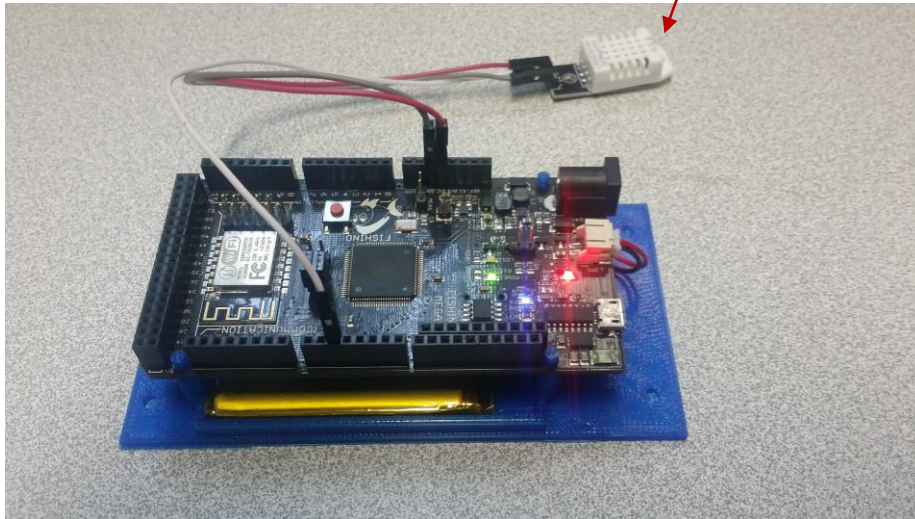
NB consultare gli esempi per il codice arduino e processing

Spunti di riflessione, proposte

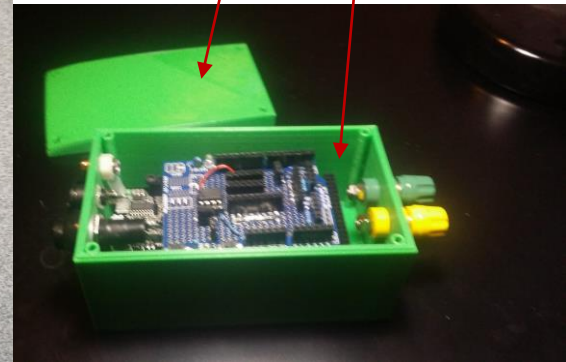
- verificare il circuito con diversi trasduttori resistivi
- Come realizzare un grafico per rappresentare l'andamento temporale di una grandezza fisica $s=s(t)$? quali problemi emergono?
- Come realizzare un grafico XY che rappresenti il legame tra 2 grandezze fisiche? quali problemi emergono?
- Cosa succede quando si rimuove la fotoresistenza? (valori indefiniti e resistenze pullup e/o pulldown)

Progettare e realizzare strumenti di misura – stazione meteo wireless

sensore per la rilevazione di umidità e temperatura



chassis realizzato mediante stampa 3D



Processing

Processing is an open source programming language and environment for people who want to create images, animations, and interactions.

- free to download and open source
- Interactive programs using 2D, 3D or PDF output
- OpenGL integration for accelerated 3D
- For GNU/Linux, Mac OS X, and Windows
- Projects run online or as double-clickable applications
- Over 100 libraries extend the software
sound, video, computer vision, and more...
- Well documented

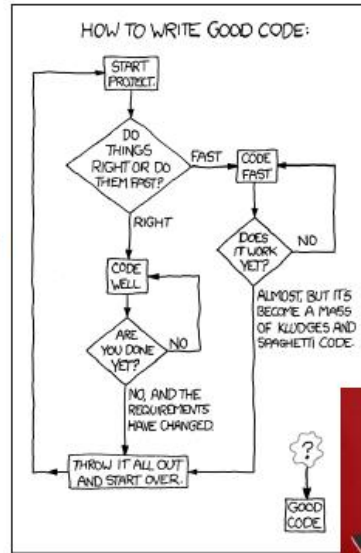
(*processing.org*)

Per info e download: <http://processing.org/>

Per l'installazione e' sufficiente decomprimere il programma in una directory

Codice processing (live.....),

- **Basic**
Sequenza di comandi
- **Intermediate**
Programmazione procedurale
`void setup() { ... }`
`void draw() { ... }`
- **Complex**
Programmazione Orientata agli Oggetti
`void setup() { ... }`
`void draw() { ... }`
`class Puppet {`
 `color colore;`
 `Puppet(color c_) {`
 `colore = c_;`
 `}`



Esempio if then else

```
size(200, 200);
background(0);

for(int i=10; i<width; i+=10) {
  // If 'i' divides by 20 with no remainder draw the first
  line
  // else draw the second line
  if(i%20 == 0) {
    stroke(153);
    line(i, 40, i, height/2);
  } else {
    stroke(102);
    line(i, 20, i, 180);
  }
}
```

Esempio if then else

```
size(200, 200);
background(0);

for(int i=2; i<width-2; i+=2) {
  // If 'i' divides by 20 with no remainder
  // draw the first line else draw the second line
  if(i%20 == 0) {
    stroke(255);
    line(i, 40, i, height/2);
  } else if (i%10 == 0) {
    stroke(153);
    line(i, 20, i, 180);
  } else {
    stroke(102);
    line(i, height/2, i, height-40);
  }
}
```


cicli

```
float box_size = 11;
float box_space = 12;
int margin = 7;

size(200, 200);
background(0);
noStroke();
// Draw gray boxes
for (int i = margin; i < height-margin; i += box_space){
  if(box_size > 0){
    for(int j = margin; j < width-margin; j+= box_space){
      fill(255-box_size*10);
      rect(j, i, box_size, box_size);
    }
    box_size = box_size - 0.6;
  }
}
```

Operatori logici

```
size(200, 200);  
background(126);
```

```
boolean op = false;
```

```
for(int i=5; i<=195; i+=5) {  
  // Logical AND  
  stroke(0);  
  if((i > 35) && (i < 100)) {  
    line(5, i, 95, i);  
    op = false;  
  }  
}
```

```
// Logical OR  
stroke(76);  
if((i <= 35) || (i >= 100)) {  
  line(105, i, 195, i);  
  op = true;  
}
```

```
// Testing if a boolean value is "true"  
// The expression "if(op)" is equivalent to "if(op == true)"  
if(op) {  
  stroke(0);  
  point(width/2, i);  
}
```

```
// Testing if a boolean value is "false"  
// The expression "if(!op)" is equivalent to "if(op == false)"  
if(!op) {  
  stroke(255);  
  point(width/4, i);  
}  
}
```

array

```
size(200, 200);
```

```
float[] coswave = new float[width];
```

```
for (int i = 0; i < width; i++) {  
    float amount = map(i, 0, width, 0, PI);  
    coswave[i] = abs(cos(amount));  
}  
for (int i = 0; i < width; i++) {  
    stroke(coswave[i]*255);  
    line(i, 0, i, height/3);  
}  
for (int i = 0; i < width; i++) {  
    stroke(coswave[i]*255 / 4);  
    line(i, height/3, i, height/3*2);  
}  
for (int i = 0; i < width; i++) {  
    stroke(255 - coswave[i]*255);  
    line(i, height/3*2, i, height);  
}
```

Esempio

simpleRead (lato arduino)

- Imposta il pin su input
- Imposta la velocità trasmissione dei dati
- Trasmette **1 byte** attraverso la seriale

```
int switchPin = 4;           // Switch connected to pin 4

void setup() {
  pinMode(switchPin, INPUT); // Set pin 0 as an input
  Serial.begin(9600);
}

void loop() {
  if (digitalRead(switchPin) == HIGH) {
    Serial.print(1, BYTE);    // send 1 to Processing
  } else {                   // If the switch is not ON,
    Serial.print(0, BYTE);    // send 0 to Processing
  }
  delay(100);                // Wait 100 milliseconds
}
```

Esempio simpleRead

- Attivare la libreria serial
- Creare un oggetto serial
- Selezionare porta USB da una lista di disponibili
- Impostare la velocità
- Estrarre un dato dal buffer dei trasmessi sulla porta

```
import processing.serial.*;

Serial myPort; // Create object from Serial class
int val;      // Data received from the serial port

void setup()                                     setup
{
  size(200, 200);
  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);
}

void draw()                                     draw
{
  if ( myPort.available() > 0) { // If data is available,
    val = myPort.read();        // read it and store it in val
  }
  background(255);              // Set background to white
  if (val == 0) {                // If the serial value is 0,
    fill(0);                     // set fill to black
  }
  else {                          // If the serial value is not 0,
    fill(204);                   // set fill to light gray
  }
  rect(50, 50, 100, 100);
}
```

Esempio

simpleWrite (lato arduino)

- Imposta il pin su output
- Imposta la velocità trasmissione dei dati
- **Controlla se il dato è disponibile** per la lettura
- **legge 1 byte trasmesso** attraverso la seriale

```
char val; // Data received from the serial port
int ledPin = 4; // Set the pin to digital I/O 4
```

```
void setup() {
  pinMode(ledPin, OUTPUT); // Set pin as OUTPUT
  Serial.begin(9600); // Start serial communication at 9600 }

```

```
void loop() {
  if (Serial.available()) { // If data is available to read,
    val = Serial.read(); // read it and store it in val
  }
  if (val == 'H') { // If H was received
    digitalWrite(ledPin, HIGH); // turn the LED on
  } else {
    digitalWrite(ledPin, LOW); // Otherwise turn it OFF
  }
  delay(100); // Wait 100 milliseconds for next reading
}

```

Esempio simpleWrite

- Attivare la libreria serial
- Dichiarare un oggetto serial
- Selezionare porta USB da una lista di disponibili
- Creare un oggetto serial
Impostare la velocità
- Cambiare colore (riempimento)
- **Trasmettere un dato**
- Definire una funzione
- Leggere le coordinate del mouse

```
import processing.serial.*;

Serial myPort; // Create object from Serial class
int val;      // Data received from the serial port
void setup()
{
  size(200, 200);
  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);
}
void draw() {
  background(255);
  if (mouseOverRect() == true) { // If mouse is over square,
    fill(204);                    // change color and
    myPort.write('H');           // send an H to indicate
  }
  else {                          // If mouse is not over square,
    fill(0);                      // change color and
    myPort.write('L');           // send an L otherwise
  }
  rect(50, 50, 100, 100);      // Draw a square
}

boolean mouseOverRect() { // Test if mouse is over square
  return ((mouseX >= 50) && (mouseX <= 150) && (mouseY >= 50) && (mouseY <= 150));
}
```


Esempio serialDuplex

- Attivare la libreria serial
- Dichiarare un oggetto serial
- Visualizza un
elenco delle porte disponibili
Selezionare porta USB
da una lista di disponibili
- Creare un oggetto serial
- Ricevere un dato
- Trasmettere un dato

```
import processing.serial.*;

Serial myPort; // The serial port
int whichKey = -1; // Variable to hold keystroke values
int inByte = -1; // Incoming serial data

void setup() {
  size(400, 300);
  // create a font with the third font available to the system:
  PFont myFont = createFont(PFont.list()[2], 14);
  textFont(myFont);

  // List all the available serial ports:
  println(Serial.list());

  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);
}

void draw() {
  background(0);
  text("Last Received: " + inByte, 10, 130);
  text("Last Sent: " + whichKey, 10, 100);
}

void serialEvent(Serial myPort) {
  inByte = myPort.read();
}

void keyPressed() {
  // Send the keystroke out:
  myPort.write(key);
  whichKey = key;
}
```

Esempio saveFile

- Disattiva il riempimento
- Definisce una forma tramite i suoi vertici
- Aggiunge un elemento in coda ad un array
- Creare un file leggibile da altri programmi
- Salva i dati come stringhe di testo

```
int[] x = new int[0];
int[] y = new int[0];
void setup() {
  size(200, 200);
}
void draw() {
  background(204);
  stroke(0);
  noFill();
  beginShape();
  for (int i = 0; i < x.length; i++) {
    vertex(x[i], y[i]);
  }
  endShape();
  // Show the next segment to be added
  if (x.length >= 1) {
    stroke(255);
    line(mouseX, mouseY, x[x.length-1], y[x.length-1]);
  }
}
void mousePressed() { // Click to add a line segment
  x = append(x, mouseX);
  y = append(y, mouseY);
}
void keyPressed() { // Press a key to save the data
  String[] lines = new String[x.length];
  for (int i = 0; i < x.length; i++) {
    lines[i] = x[i] + "\t" + y[i];
  }
  saveStrings("lines.txt", lines);
  exit(); // Stop the program
}
```

Esempio loadFile

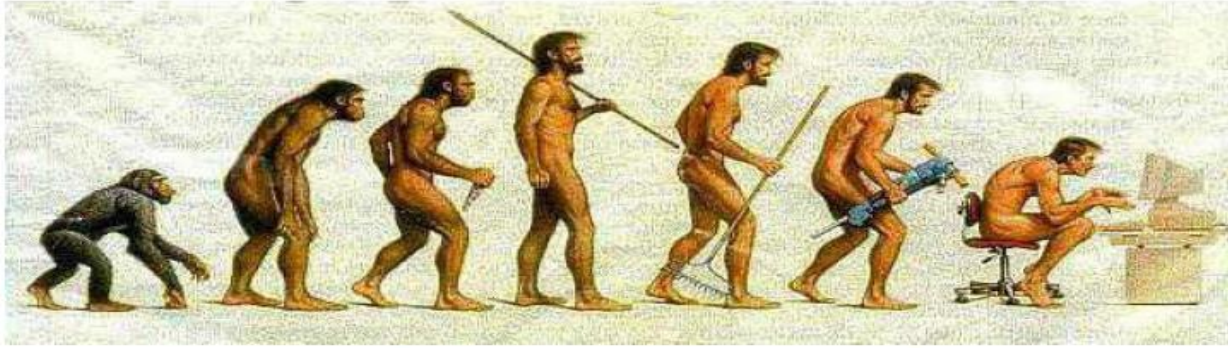
- Dichiarare un array di stringhe
- Imposta il numero di frame/s
- Leggere un file
- Estrarre i singoli dati usando **tab come separatore**

```
String[] lines;
int index = 0;

void setup() {
  size(200, 200);
  background(0);
  stroke(255);
  frameRate(12);
  lines = loadStrings("positions.txt");
}

void draw() {
  if (index < lines.length) {
    String[] pieces = split(lines[index], '\t');
    if (pieces.length == 2) {
      int x = int(pieces[0]) * 2;
      int y = int(pieces[1]) * 2;
      point(x, y);
    }
    index = index + 1;
  }
}
```

**L'evoluzione del software ha portato a Processing semplice,
flessibile e potente, multiplatforma...**



Forse qualcosa è andato storto... ?

... poteva andare anche peggio !!!

ARRGH! MY MAP OF LISTS OF MAPS
TO STRINGS IS TOO HARD TO
ITERATE THROUGH! I'LL JUST ASSIGN
EVERYTHING A NUMBER AND USE
A *!*!*@ ARRAY



Light Edition (arduino based) Mobile Unit



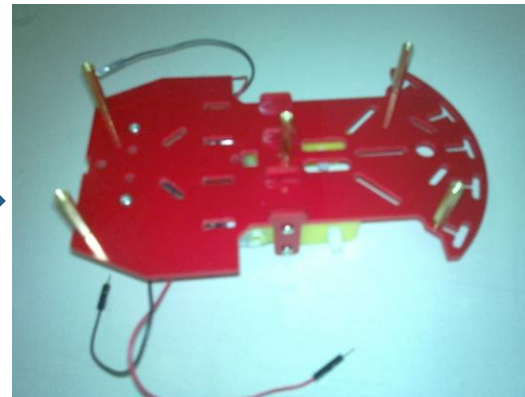
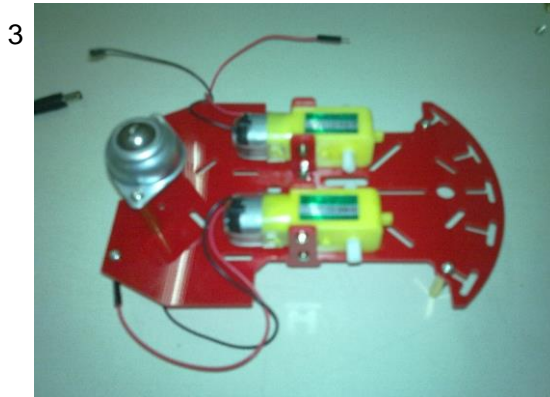
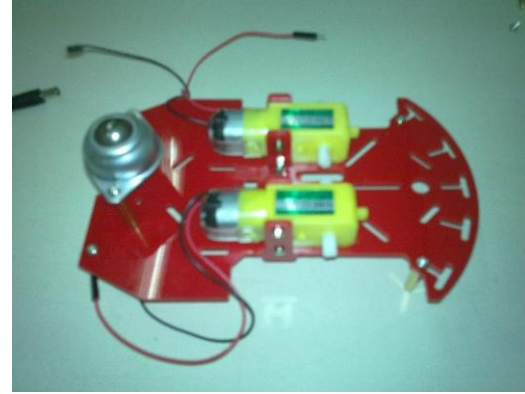
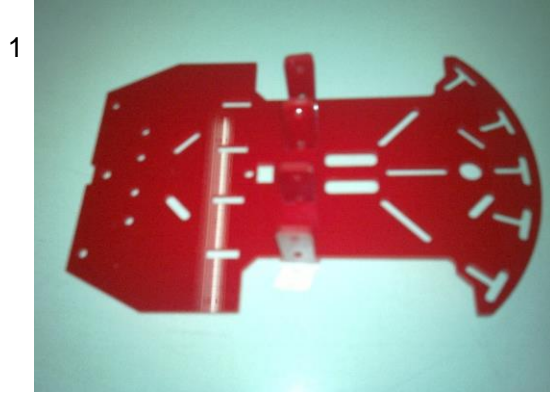
LEMU è basato sul [robot beginner kit](#) offerto da [robot-domestici](#)

Il kit comprende:

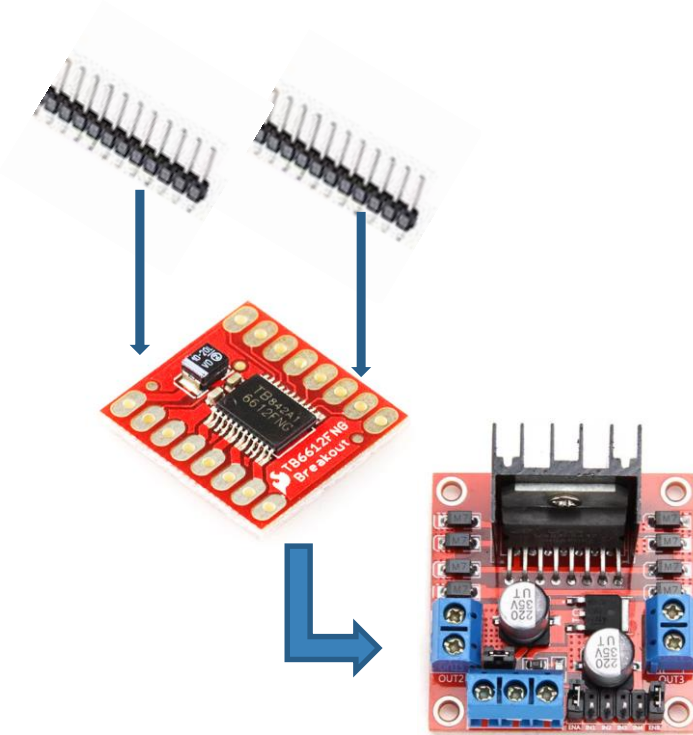
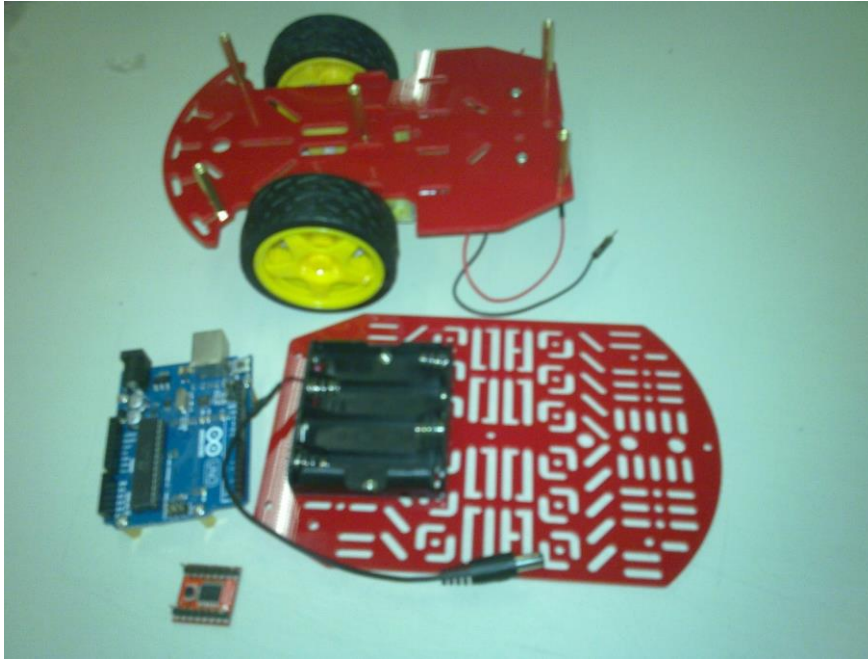
- 1 x Magician Chassis
- 1 x SHARP - [GP2Y0A21YK](#) (sensore IR analogico, range 60cm)
- 1 x Motor Driver [1A Dual TB6612FNG](#)
- 2 x Strip 10 pin
- 1 x Infrared Sensor
- Jumper Wire - 3-Pin JST
- 10 x Jumper Wires M/F
- 10 x Jumper Wires F/F

Ordinate anche una mini breadboard, vi servirà (e una scorta di Jumper Wires M/M) serviranno anche 1 o 2 squadrette per fare da supporto al sensore IR

Assemblaggio chassis/motori (1/3)

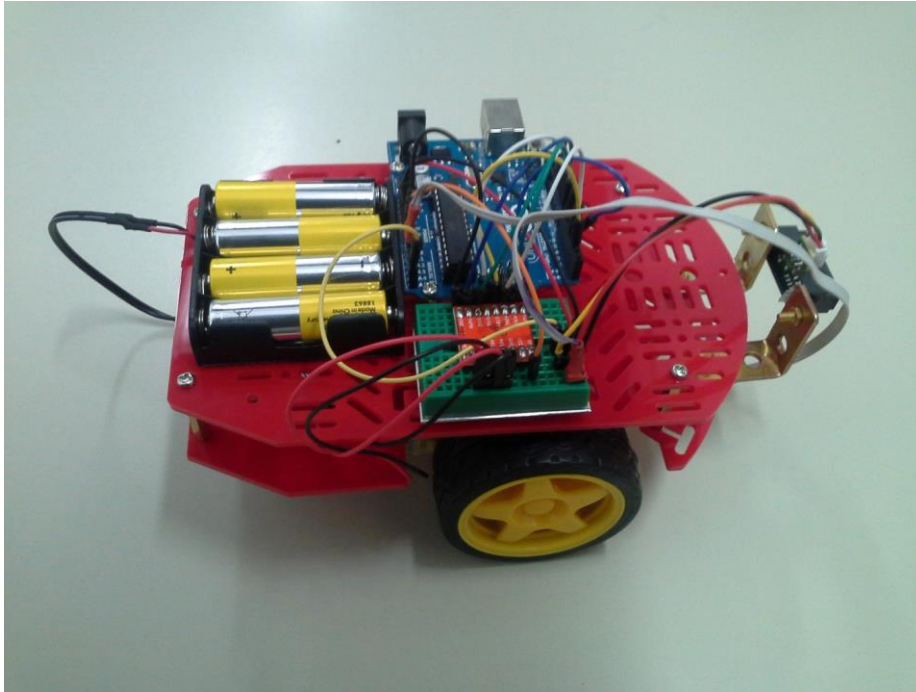


Assemblaggio 2/3 driver motori, pacco batterie, arduino



Saldare gli strip maschi in modo tale che le indicazioni dei pin sul driver motori siano leggibili quando si installa il driver su una minibreadboard

Assemblaggio 3/3 sensore infrarosso, cablaggi



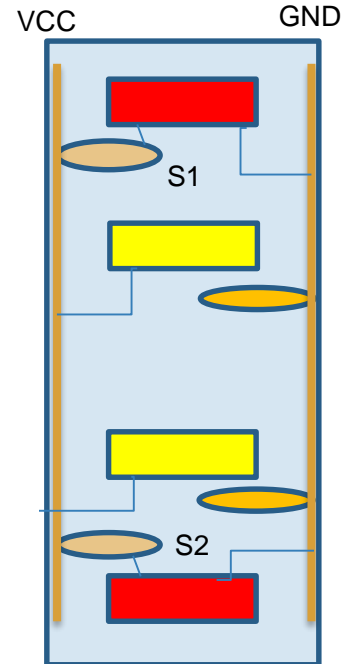
Realizziamo un encoder per LEMU

Questa parte è opzionale ma fortemente consigliata perché tramite l'encoder si potranno stimare:

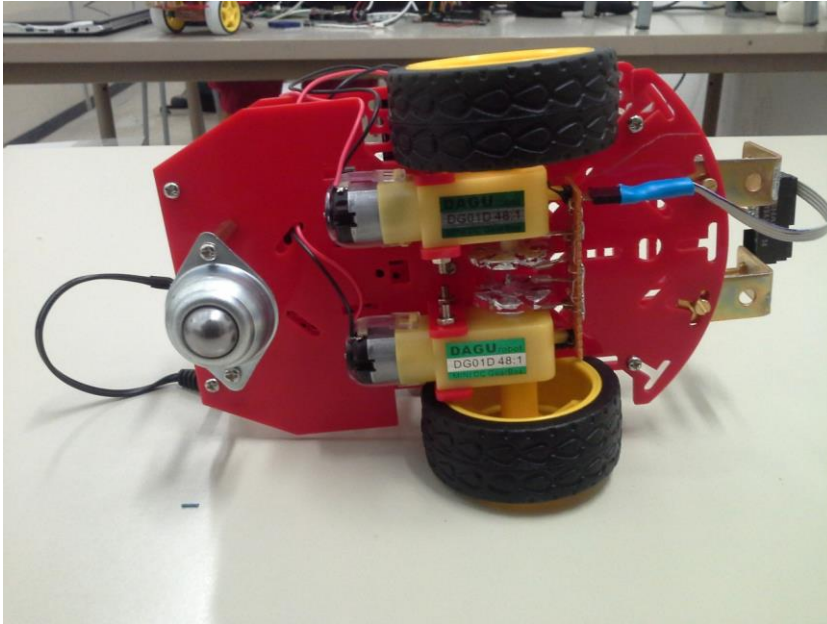
- spazio percorso
- velocità angolare delle ruote
- velocità e accelerazione del LEMU
- e si potrà controllare la traiettoria

Materiale occorrente:

- 2 coppie di sensori IR RX/TX
- Una millefori (che andrà tagliata ad hoc)
- 2 resistenze da 330ohm
- 2 resistenze da 10Kohm



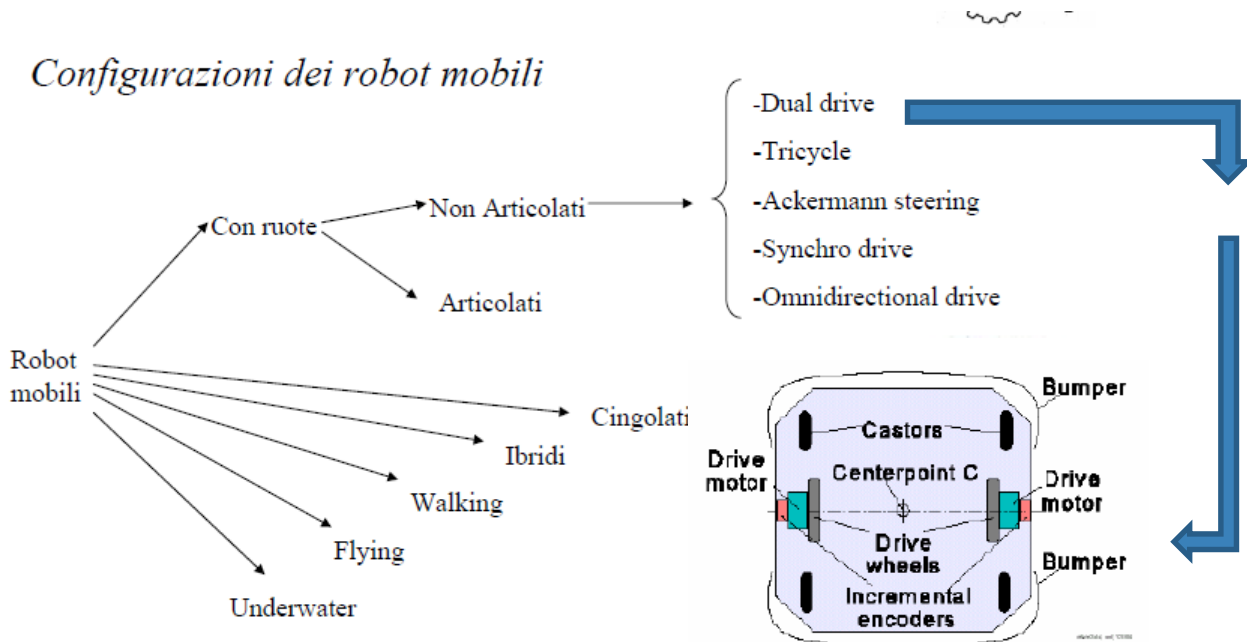
Fissare l'encoder allo chassis



È possibile utilizzare colla a caldo per il fissaggio

Attenzione
è necessario rivestire di alluminio le ruote dentate di plastica comprese nel kit perché **sono trasparenti all'infrarosso** (utilizzare una colla ad alta tenuta)

Meccanica dual drive

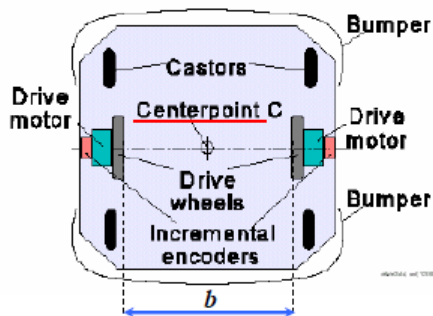


- Per la stabilità servono almeno 3 punti di appoggio: **2 ruote indipendenti e ball caster**
- Il **controllo proporzionale** (oppure on-off) avviene mediante **encoder**

Dual drive: spostamento lineare

Per misure di distanza percorsa e per stabilizzare la traiettoria occorre:

- convertire la rotazione delle ruote in un movimento lineare
- compensare la diversa velocità di rotazione dei motori



$$C_m = \frac{\pi D}{nC_e}$$

D = diametro nominale della ruota

n = rapporto di riduzione del motoriduttore

C_e = numero di impulsi per giro dell'encoder

Per studiare i movimenti del robot, si studiano i movimenti del punto C , cioè del centro dell'asse che congiunge le ruote. Con b indichiamo la distanza fra le ruote (interasse).

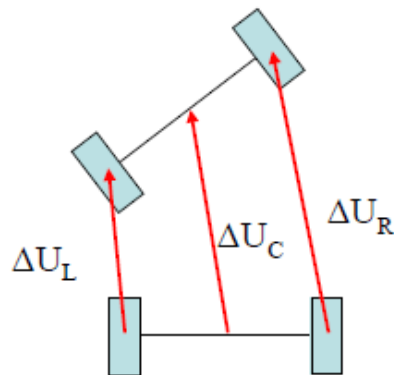
Dual drive: spostamento

Dato il numero di impulsi registrati dall'encoder
è possibile determinare il tratto percorso (...misure!)

$$\Delta U_{L,R(k)} = C_m N_{L,R(k)}$$

Il punto medio subirà uno spostamento dato dalla media degli spostamenti delle due ruote:

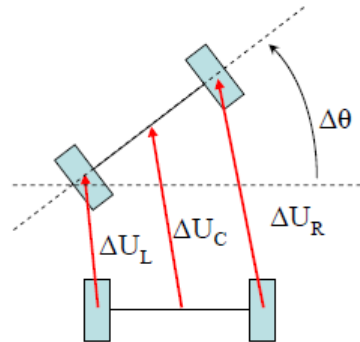
$$\Delta U_{C(k)} = \frac{\Delta U_{L(k)} + \Delta U_{R(k)}}{2}$$



Dual drive: orientamento

La variazione di orientamento e quindi l'orientamento del robot (rispetto all'istante iniziale in cui è stato attivato) si possono ottenere dalle relazioni:

$$\Delta\theta_{(k)} = \frac{\Delta U_{R(k)} - \Delta U_{L(k)}}{L}$$



Integrazione...

$$\theta_{(k)} = \theta_{(k-1)} + \Delta\theta_{(k)}$$

$$\theta_{(k)} = \theta_{(k-1)} + \Delta\theta_{(k)}$$

$$x_{(k)} = x_{(k-1)} + \Delta U_{C(k)} \cos\theta_{(k)}$$

$$y_{(k)} = y_{(k-1)} + \Delta U_{C(k)} \sin\theta_{(k)}$$

Odometria e determinazione della traiettoria errori casuali e sistematici, correzione con data fusion

Odometria:

- valutare lo spostamento
- sommare lo spostamento attuale al precedente per ottenere la **traiettoria**

Problema:

- l'accumolo degli errori **casuali**
- la **presenza di errori sistematici**

Tecnica di test:

- programmare il **movimento lungo il perimetro di un quadrato**
- effettuare misure per la **calibrazione**

Compensazione degli errori mediante la **data fusion**:

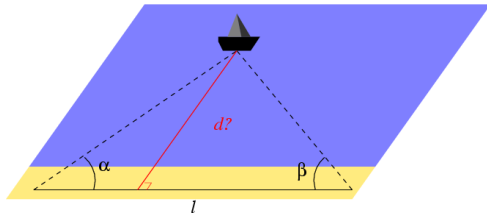
La traiettoria viene corretta con informazioni provenienti da fonti indipendenti; **bussola, gps, triangolazione, mappe ...**

Triangolazione di landmark

Landmark: punti di riferimento di posizione nota

Mediante **triangolazione** (o **multilaterazione**)

si stima la posizione del robot relativamente ai landmark

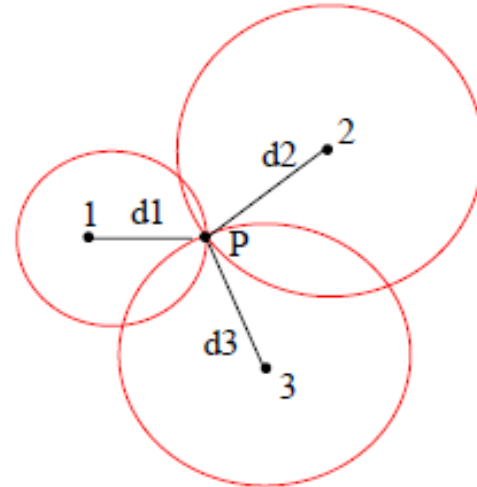


$$d = x \cdot \tan \alpha$$

$$d = y \cdot \tan \beta$$

$$\begin{cases} x + y = l \\ y \cdot \tan \beta = x \cdot \tan \alpha \end{cases}$$

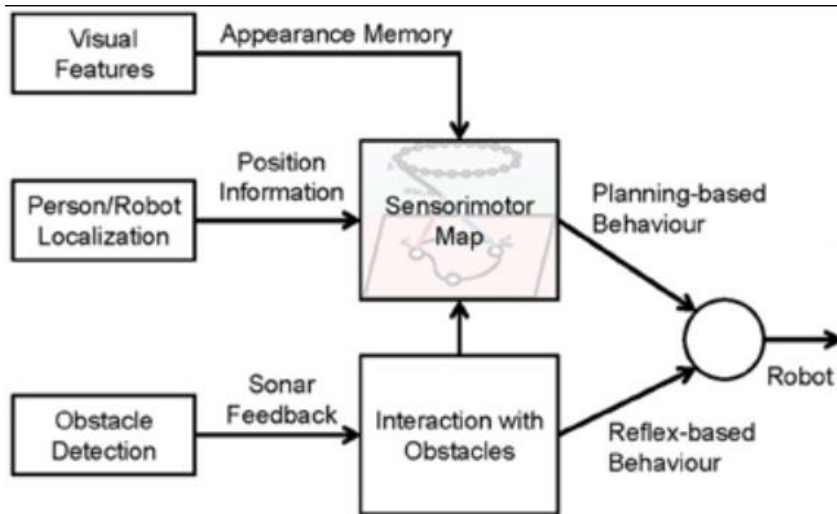
$$d = \frac{l \cdot \tan \alpha \cdot \tan \beta}{\tan \alpha + \tan \beta} \rightarrow d = \frac{l \cdot \sin \alpha \cdot \sin \beta}{\sin(\alpha + \beta)}$$



Navigazione e mappe

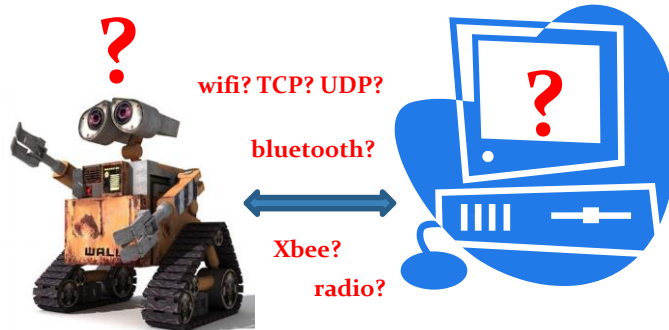
In alternativa e/o a complemento dei landmark si usano le mappe dell'ambiente (note a priori o costruite dinamicamente)

Si determina la propria posizione relativamente alla mappa e si aggiorna la mappa mentre si cerca di raggiungere un target ...



Trasmissione BT

- Stabilire la connessione PC-Xduino via BT
- Utilizzare **la libreria software seriale** per trasmettere le informazioni da seeduino a PC
trasmettere un carattere da PC a seeduino per controllare il robot
- Stabilire un **protocollo asimmetrico EFFICIENTE !!**
- Controllo **heartbeat**: in **assenza di comandi**, dopo un certo tempo, può essere disattivato il robot (per sicurezza) oppure entrare in **modalità autonoma ...**)



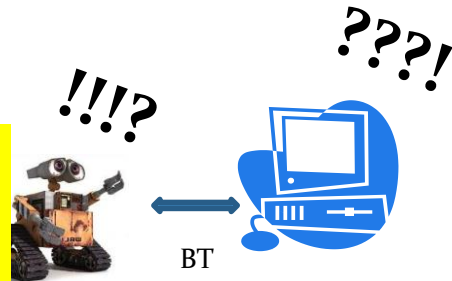
Telemetria e controllo remoto

- Il robot **acquisisce dati** sull'ambiente attraverso i sensori
- Le informazioni sono trasmesse attraverso la **connessione BT**
- Una apposita funzione del programma di controllo remoto del robot (ad es realizzato con **processing**),
 - rimane **in ascolto sul canale seriale BT**
 - **legge i dati dei sensori** (immessi in una **stringa di testo tab delimited** (oppure in una **sequenza di byte di lunghezza prefissata**)
 - **estrae i dati di ciascun sensore** e aggiorna un vettore di stato (che rappresenta lo stato interno del robot)
- Una funzione del programma **decide** quali comandi inviare al robot (è sufficiente un carattere per 256 stati!) ⁽¹⁾
- Una funzione **invia i comandi al robot** attraverso BT

- definire un protocollo comune efficiente non è facile
- imparare a parlare non è facile
- insegnare una lingua non è facile



... ci vuole **tanta PAZIENZA!**



Lato robot (arduino|seeduino...)

```
#include <SoftwareSerial.h> //Software Serial Port
#define RxD 4
#define TxD 5
#define DEBUG_ENABLED 1
SoftwareSerial blueToothSerial(RxD,TxD);
char incoming;
char InternalStatus[50];
int dataBT=38400;
void setup() {
  Serial.begin(9600);
  pinMode(RxD, INPUT);
  pinMode(TxD, OUTPUT);
  setupBlueToothConnection();
}
void setupBlueToothConnection() {
  blueToothSerial.begin(38400); //Set BluetoothBee BaudRate to default baud rate 38400
  delay(1000);
  sendBlueToothCommand("\r\n+STWMOD=0\r\n");
  sendBlueToothCommand("\r\n+STNA=wally\r\n"); // CAMBIA IL NOME DEL TUO ROBOT
  sendBlueToothCommand("\r\n+STAUTO=0\r\n");
  sendBlueToothCommand("\r\n+STOAUT=1\r\n");
  sendBlueToothCommand("\r\n +STPIN=0000\r\n");
  delay(2000); // This delay is required.
  sendBlueToothCommand("\r\n+INQ=1\r\n");
  delay(2000); // This delay is required.
} //Checks if the response "OK" is received
void CheckOK() {
  char a,b;
  while(1) {
    if(blueToothSerial.available()) {
      a = blueToothSerial.read();
      if('O' == a) {
        // Wait for next character K. available() is required in some cases, as K is not immediately
        available.
        while(blueToothSerial.available()) {
          b = blueToothSerial.read();
          break;
        }
        if('K' == b) {
          break;
        }
      }
    }
  } while( (a = blueToothSerial.read()) != -1) {
    //Wait until all other response chars are received
  }
}
```

```
void sendBlueToothCommand(char command[]) {
  blueToothSerial.print(command);
  //CheckOK();
}
void loop() {
  updateStatus();
  blueToothSerial.println(InternalStatus);
  // get character
  incoming=blueToothSerial.read();
  if (incoming!=-1) {
    Serial.println(incoming); // dump everything
  } // decide what to do with it
  switch (incoming) {
    case '1':
      Serial.println("[1]"); // insert here your code
      break;
    case '2':
      Serial.println("[2]"); // insert here your code
      break;
    case '3':
      Serial.println("[3]"); // insert here your code
      break;
  }
  delay(100); // may be removed
}
void updateStatus(void) {
  char status[16];
  int A0=analogRead(A0);
  int A1=analogRead(A1);
  int A2=analogRead(A2);
  int A3=analogRead(A3);
  int A4=analogRead(A4);
  int A5=analogRead(A5);
  sprintf(InternalStatus, "%d\t%d\t%d\t%d\t%d\t", A0, A1, A2, A3, A4, A5);
}
```

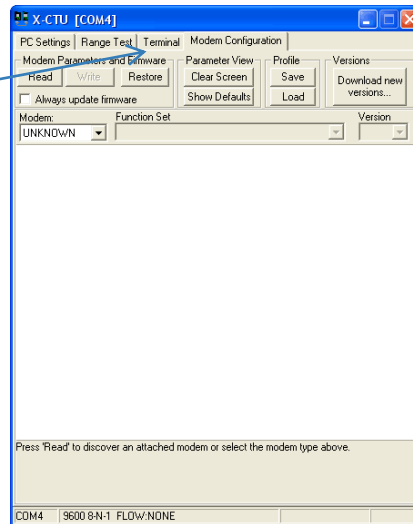
Lato processing

```
import processing.serial.*;
Serial myPort; // Create object from Serial class
int val; // Data received from the serial port
void setup() {
  size(200, 200);
  // List all the available serial ports
  println(Serial.list());
  String portName = Serial.list()[12];
  myPort = new Serial(this, portName, 38400);
}
void draw() {
  while (myPort.available() > 0) {
    String inBuffer = myPort.readString();
    if (inBuffer != null) {
      println("BT: "+inBuffer); // dump everything
      int [] nums;
      String [] splits = split(inBuffer, "\t"); // split
      nums = new int[splits.length]; // create a target array
      for(int i = 0; i < splits.length; i++){
        nums[i] = int(splits[i]); // insert into numbers
        print("<"); // begin
        for(int i = 0; i < splits.length; i++){
          print(nums[i]+" "); // dump one by one
        }
        println(">"); // end
      }
    }
  }
}
```

Gestire e monitorare la connessione

Utilizzare:

- terminal nell'ide di arduino per **monitorare la connessione USB**
- un emulatore terminale, ad esempio X-CTU, uno strumento molto utile! (http://ftp1.digi.com/support/utilities/40002637_c.exe) per **monitorare la connessione BT**



Riferimenti utili:

- <http://www.westernwillow.com/cms/blog/franco/creating-bluetooth-serial-port-ubuntu>
- <http://askubuntu.com/questions/2977/how-to-assign-a-serial-port-to-my-bluetooth-phone>
- <http://scottada.ms/article/adding-so-called-virtual-bluetooth-serial-port-ubuntu-1110-using-atheros-ar3011-based-usb>
- http://www.thinkwiki.org/wiki/How_to_setup_Bluetooth
- <http://www.broadcom.com/support/bluetooth/update.php>
- <http://www.ladyada.net/make/xbee/configure.html>

Perdita di connessione?

Il robot commuta su

autonomo

search and connect mode

stop

con thymio, ozobot o mbot ranger è tutto più semplice!

This screenshot shows the Arduino IDE interface. The 'Actions' panel on the right has the 'Robots' category selected, which is circled in red. A red arrow points from this category to the 'Thymio Blockly Interface' window below. Another red circle highlights the 'Clock' icon in the 'Actions' panel. The main workspace shows a script with various function calls for controlling an LED and a motor.

This screenshot shows the Makeblock IDE interface. On the left, there is a 3D model of a robot board with a motor attached. On the right, a script is shown with the following blocks: 'when clicked', 'forever' loop containing 'set motor M1 speed 50', 'wait 1 secs', 'set motor M1 speed 0', 'wait 1 secs', 'set motor M1 speed -50', 'wait 1 secs', and 'set motor M1 speed 0'. The 'Robots' category is selected in the 'Scripts' panel.

This screenshot shows the 'Thymio Blockly Interface'. It features two event-driven scripts. The first script is triggered by 'on forward button touched' and contains blocks for 'start driving forward with speed 300' and 'set top LED to green'. The second script is triggered by 'on front middle sensor detecting proximity' and contains blocks for 'stop motors' and 'set top LED to red'.

This screenshot shows the Makeblock IDE interface. A script is shown with the following blocks: 'when clicked', 'forever' loop containing four 'if not read digital pin' conditions (pins 2, 3, 4, 5) each followed by 'play tone pin 9 on note C6', 'D6', 'E6', and 'F6' with a beat of 'Eighth'. The 'Robots' category is selected in the 'Scripts' panel. A 'Complete programming' button is visible at the bottom.



Grazie!

domande ?

email: grosso@fisica.unige.it



Credits

- Prof.ssa **Miranda Pilo** (DIFI)
- Prof. **Flavio Gatti** (DIFI, INFN)

... la mia Famiglia che mi supporta e sopporta ...

- **Sara, Victor, Fiamma**