

IOT Robots

2016

ISS Robots

Controllori, Attuatori, Sensori

MICROCOMPUTER

- Raspberry Pi

MICROCONTROLLERS

- Arduino
- ESP8266

- Motor driver
- Batterie

Sensori

- Ultrasonic Sensor
- Infrared line sensor
- Infrared distance sensor
- Camera

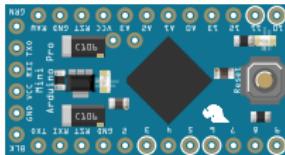
Attuatori

- DC Motor
- Servo
- Stepper Motor

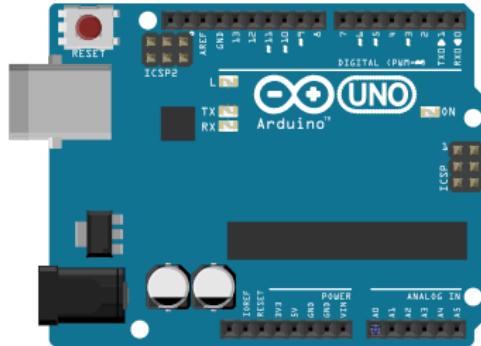
Arduino

ARDUINO

- PROMINI



- UNO (REV 3)



ATMEGA 328

RAM 2K

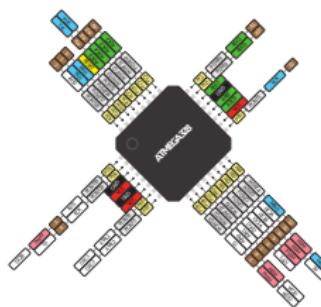
Flash 32K

Timers

UART

I2C

SPI



ESP8266

NodeMCU, Adafruit HUZZAH, Wemos D1

WiFi SoC ESP8266

CPU 80MHz 160MHz

RAM 160K

BOOT ROM 64K

Flash 4M

Timers

UART

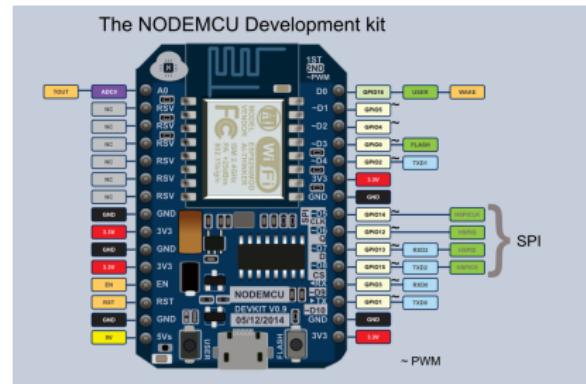
I2C

SPI

Wi-Fi Direct (P2P), soft-AP

802.11 b/g/n

TCP/IP Integrated protocol stack



Come programmare?

- ESP-open-sdk (C)
- NodeMCU (Lua)
- Arduino (C++)
- Sming (C++)

Programmazione dei microcontrollori

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a toolbar with icons for file operations. The sketch window contains the following code:

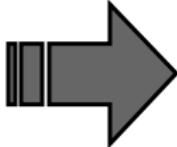
```
File Edit Sketch Tools Help
Blink S
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

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

Done compiling.

Binary sketch size: 3,828 bytes (of a 26,672 byte maximum)
```



The screenshot shows the AVR-GCC compiler output. It displays assembly code for the ATmega328P microcontroller, followed by a hex dump of the generated binary file. The assembly code includes instructions like `ldr r2, [r0]`, `ldr r3, [r0]`, `add r3, r2`, and `str r3, [r0]`. The hex dump shows the binary representation of the code, starting with `100CB00009017F0128173907A9F0E0917E01F0910A`.

```
100CB00009017F0128173907A9F0E0917E01F0910A
100CB0007F01E45CFE4F808120917E0130917F01A5
100CB0002F5F3F4F2FF73332730937F0120937E01B7
100CE000908008958FFEF0F0F0895CF93DF03C09120
100CF0007C01D0917E017D012196CF73D2780017E01B0
100D000090917F01C817D097991082E00E09450039A
100D10008F3F2FF920761F0E0917C001F0017D0110
100D2000E45CFE4F808300937D01C0937C01DFC0D4
100D3000DF01C9E018080583E00C0947050F031F0343
100D4000CF93DF031F02CD87DEB780C1698380917B
100D500000901882359F041E050E08E016F5F7F4E9
100D600083E00E948A03181615903F081E090E0A7
100D7000F0019383828380E090E00F900F0C919120
100D800017910F916895FC0120812E5F208342E452
100D900050E066E97110800E0094260E01818153
100DA0009881913A50F4813200F04BC047E080E00C
100D00062E071E0800E0E94250445C0913209F034
100DC00040C0803230F467E07082E082E0100E09438
100D000690430C08023209F034C082180930901E0C
100DE0008891020190010301A0910481B09105104D
100DF0008839440105810531F58091090108F0AA
100E000012C087E797E790930108800980082B0E02
100E100088E190E00F80F894A095800360000FBE2D
100E200020923600010C088E10FB6C89480036000B2
100E30001902600000FBEA805109201881002000851
100E400002C080E00080581E008095102310110926F7
100E5000300188EE03E0A0E0B0E0800332010003FF
100E60003301A0933401B09335018EE091E0909358
100E70002F0180932E018895CF92DF9E2F02FF927F
100E80000F931F93CF93DF936C017A01E01E60E72
100E9000717E00E010E0C15DF0651F06991D60184
100EA000ED091FC910190F081E02DC0618995080FAC
100EB000191FF1FC2801DF91CF911F910F1FF90C2
100EC000EF90DF900CF000805EE0FF1F0509049103
08E0000E02D0094F894FFCF10
100ED0000000000180000000000000000000000000000E007B
0C0EE8003C072E0652063B069B0675062
00000001FF
Blink.cpp_hex
[1-Blink.cpp_hex]
232,1
```

INTERPRETER/

FIRMWARE/

OPERATION/SYSTEM/

Programmazione dei microcontrollori

Esempio

```
1 #include "arduino.h"
2
3 void setup() {
4     DDRD = DDRD | B00100000;
5 }
6 void loop() {
7     PORTD = PORTD | B00100000;
8     delay(1000);
9     PORTD = PORTD & B11011111;
10    delay(1000);
11 }
12 }
```

```
1 #include "arduino.h"
2
3 void setup() {
4     DDRB |= _BV(DDB5);
5 }
6
7 void loop() {
8     PORTB |= _BV(PORTB5);
9     _delay_ms(1000);
10    PORTB &= ~_BV(PORTB5);
11    _delay_ms(1000);
12 }
```

Wiring

Arduino, WiringPi, Sming basati su Wiring

- Open Source
- Open Hardware
- Linguaggio o framework accessibile
- IDE semplice
- ...



Wiring

Struttura del programma

File Edit Sketch Tools Help

Blink

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

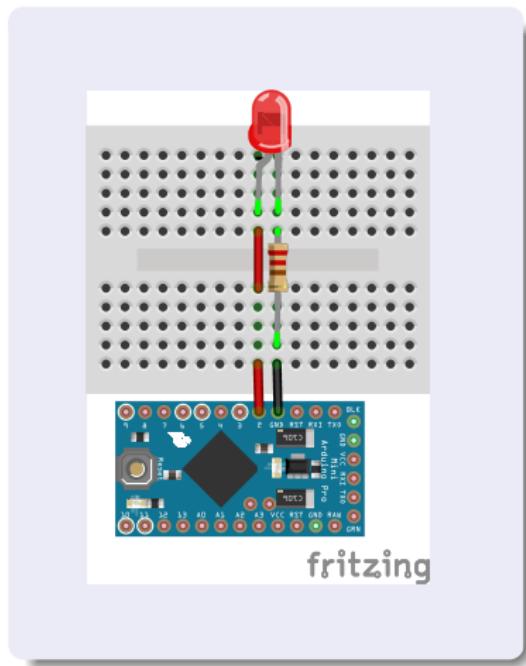
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}

Done compiling.

Binary sketch size: 3,828 bytes (of a 28,672 byte maximum)
```

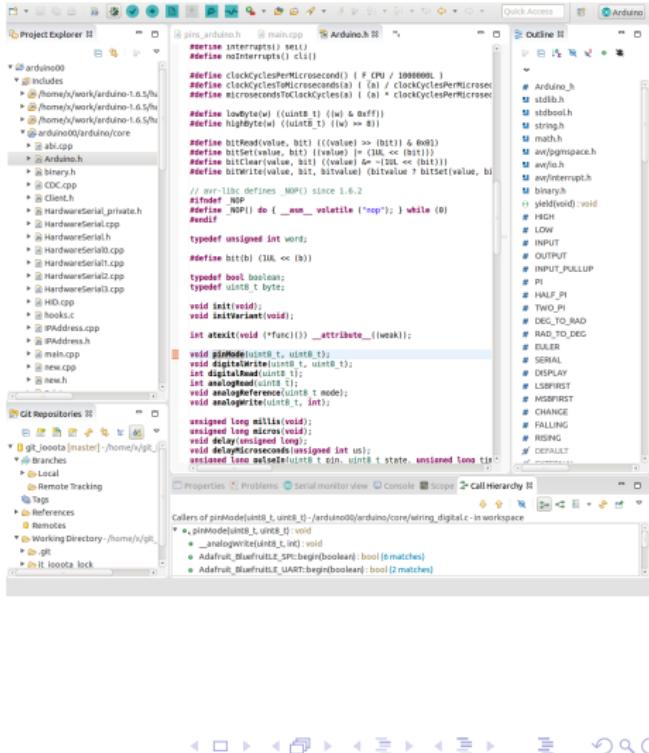


IDE per sviluppo

Eclipse plugin AVR-eclipse
completely free and
open Arduino IDE
alternative

PlatformIO

cross platform code
builder and library
manager



Arduino Button

polling

File Edit Sketch Tools Help

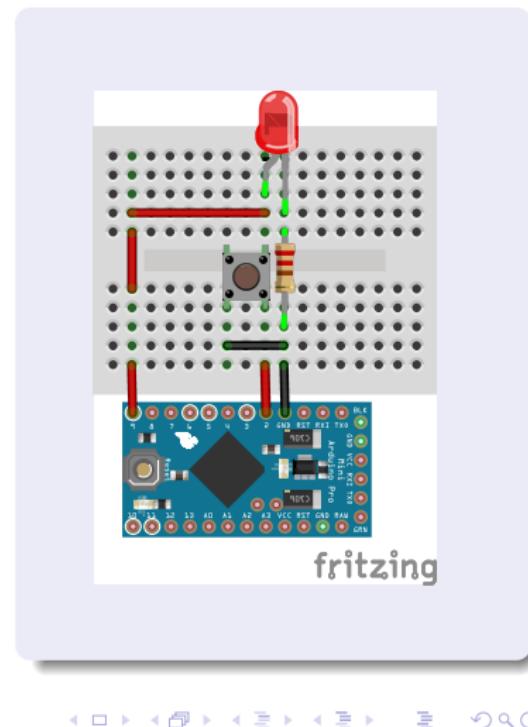
sketch_may15b \$

```
void setup(){
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop(){
  int sensorVal = digitalRead(2);
  Serial.println(sensorVal);
  if (sensorVal == HIGH) {
    digitalWrite(13, LOW);
  }
  else {
    digitalWrite(13, HIGH);
  }
}
```

Done compiling.

Binary sketch size: 4,476 bytes (of a 28,672 byte maximum)



Arduino Button

interrupt

File Edit Sketch Tools Help

sketch_may15a.ino

```
int pin = 13;
volatile int state = LOW;

void setup(){
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, CHANGE);
}

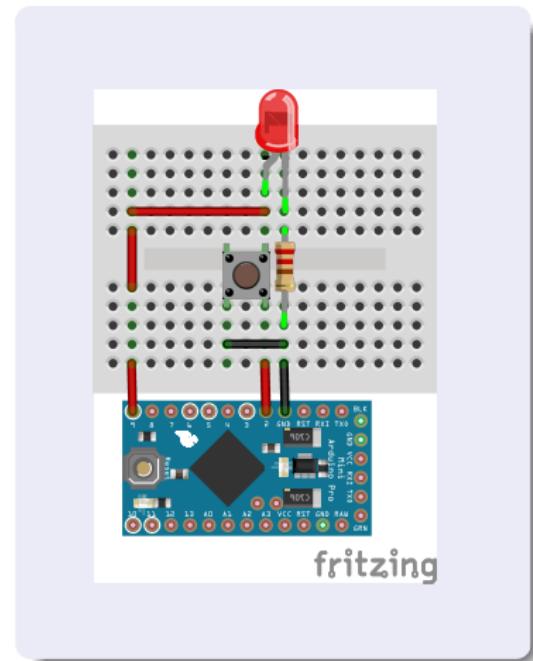
void loop(){
  digitalWrite(pin, state);
}

void blink(){
  state = !state;
}

Done compiling.

Binary sketch size: 4,446 bytes (of a 28,672 byte maximum)
```

1 SparkFun Pro Micro 5V/16MHz on /dev/ttyACM0



Raspberry Pi

Single Board Computer

- CPU ARM11
700Mhz
- 512 RAM
- 2 USB
- Ethernet port
- HDMI
- 3.5 mm audio
- GPIO
 - ▶ **Digital I/O**
 - ▶ PWM pin
 - ▶ Serial
 - ▶ I2C

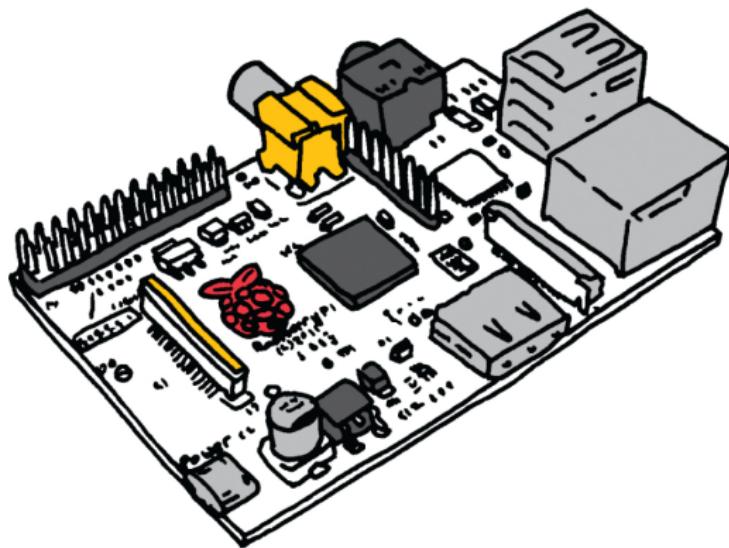


Figura : RASPBERRY PI MODEL B

Raspberry Pi

Software

Sistemi operativi

- Raspbian(Debian)
- Pidora (Fedora Remix)
- Arch Linux
- RISC OS
- E molti altri sistemi

E' possibile installare Android, anche se non e' supportato ufficialmente.

Linguaggi di programmazione

- Python
- Java
- Bash
- C / C++
- Ruby
- Perl
- ...

Raspbian

Versione custom per il corso

Software Aggiuntivo

- Server DHCP
- Tool e librerie per la progettazione
 - ▶ wiringPI
 - ▶ RaspiCam
 - ▶ Java
 - ▶ pi4j
 - ▶ vim
- Server VNC accessibile al 192.168.137.2:1
- Trasferimento del file tramite protocollo SMB
- Configurazione wifi facilitata
- [https://137.204.107.21/www-files/
2016-02-26-raspbian-jessie-iss.img.zip](https://137.204.107.21/www-files/2016-02-26-raspbian-jessie-iss.img.zip)

RASPBERRY PI PINOUT

Raspberry Pi (Rev1)

3V3	1	2	SV
GPIO0	3	4	5V
GPIO1	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi (Rev 2)

3V3	1	2	SV
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi B+, 2, 3 & Zero

3V3	1	2	SV
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Key
+
Ground
UART
I2C
SPI
GPIO
Pin Number



Figura : RASPBERRY PI PINOUT

Programmazione GPIO

E' possibile gestire GPIO tramite un filesystem virtuale.

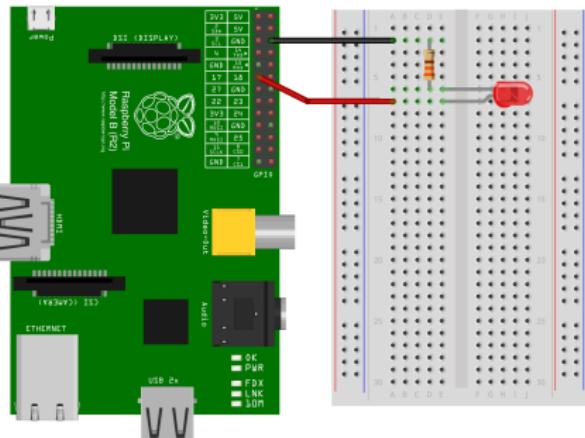
- “/sys/class/gpio“ è un’interfaccia di controllo in user space
 - ▶ “export“ permette di esportare il controllo di GPIO in user space. (*echo 19 > export*)
 - ▶ “unexport“ rimuove la risorsa da user space (*echo 19 > unexport*)
- /sys/class/gpio/gpioN/ è un’interfaccia di controllo di un singolo GPIO
 - ▶ “direction” imposta GPIO come Input o Output (*echo in > direction*)
 - ▶ “value” legge il valore di GPIO (*cat value*)
 - ▶ ...

Raspberry Pi GPIO Output

```
#!/bin/bash
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction

while :
do
    echo 0 > /sys/class/gpio/gpio17/value
    sleep 0.2
    echo 1 > /sys/class/gpio/gpio17/value
    sleep 0.2
done
```

```
#!/bin/bash
gpio mode 0 out
gpio write 0 1
sleep 1
gpio write 0 0
```

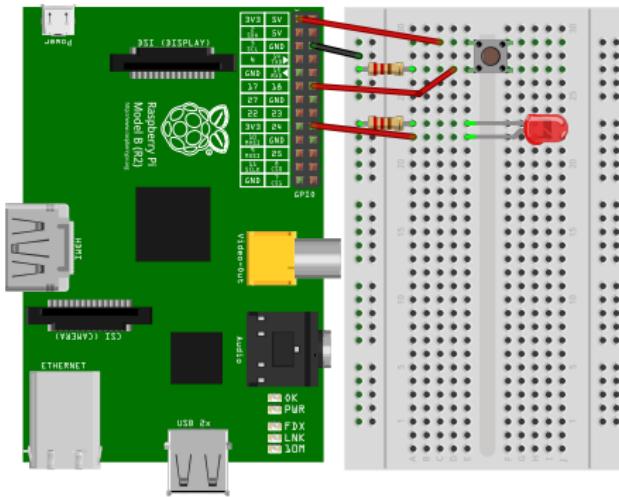


fritzing

Raspberry Pi

```
#!/bin/bash
echo 24 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio24/direction
echo 18 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio18/direction

while :
do
    cat /sys/class/gpio/gpio24/value > \
    /sys/class/gpio/gpio24/value
    sleep 0.2
done
```



fritzing

Differenza tra Arduino e Raspberry Pi

	Arduino	Raspberry Pi
Processor	ATMega 328	ARM11
Clock Speed	16 MHz	700 MHz
Flash	32 KB	SD
RAM	2 KB	512 MB
I/O Current Max	40 mA	10 mA
Min Power	42 mA	700 mA
Register Width	8-bit	32-bit
Operating System	None	Linux and Other

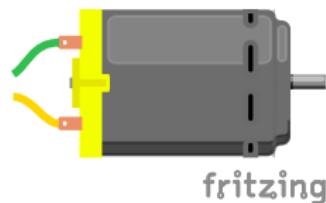
Tabella : Arduino Uno vs Raspberry

Microcontroller vs Microprocessor

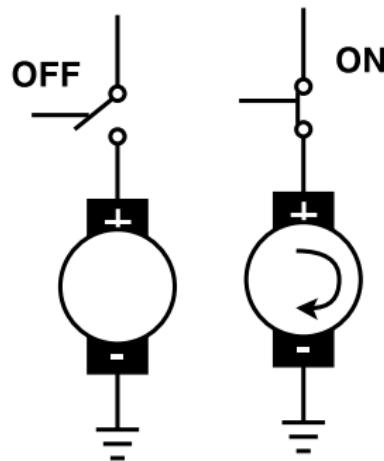
DC Motor

Controllo ON-OFF di motori DC

- DC Motors



Lego RCX DC Motors



Quando l'interruttore è aperto il motore è fermo, quando è chiuso gira alla massima velocità

Attuatori

HBridge

- H Bridge (Motor Driver)

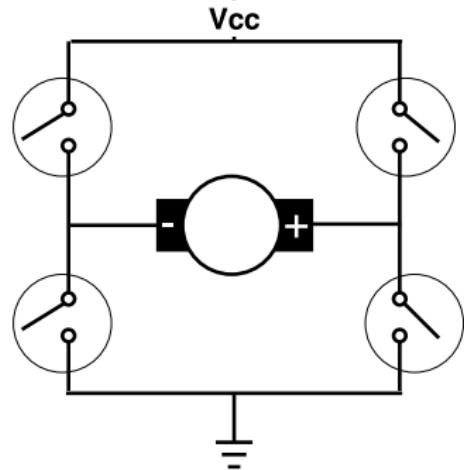
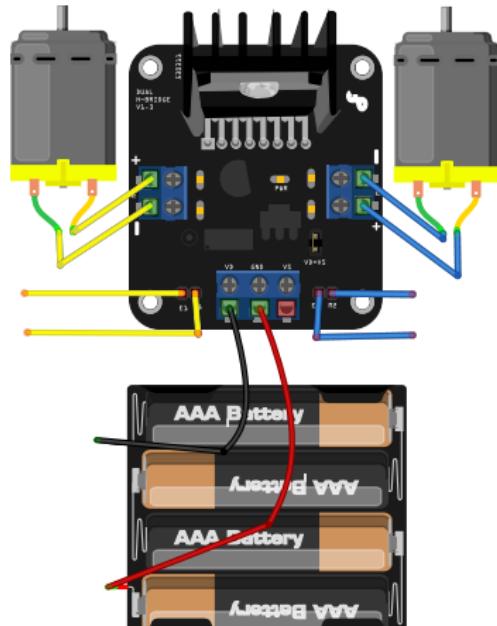


Figura : schema semplificato di un ponte H



Controllo di velocità

PWM Pulse Width Modulation

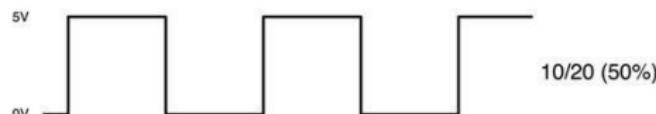
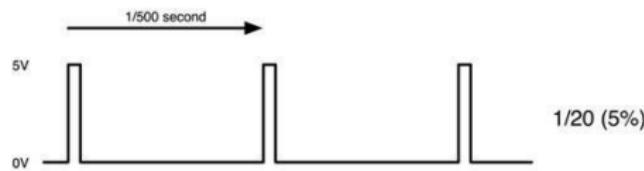


Figura : modulazione di larghezza di impulso

Arduino UNO 5 pin predisposti per PWM
Raspberry Pi 1/2 pin supportano Hardware PWM.

Direct PWM control

Input		Output
FIN	RIN	
PWM	L	Forward
L	PWM	Reverse
H	H	Brake
L	L	Idle

Servomotore

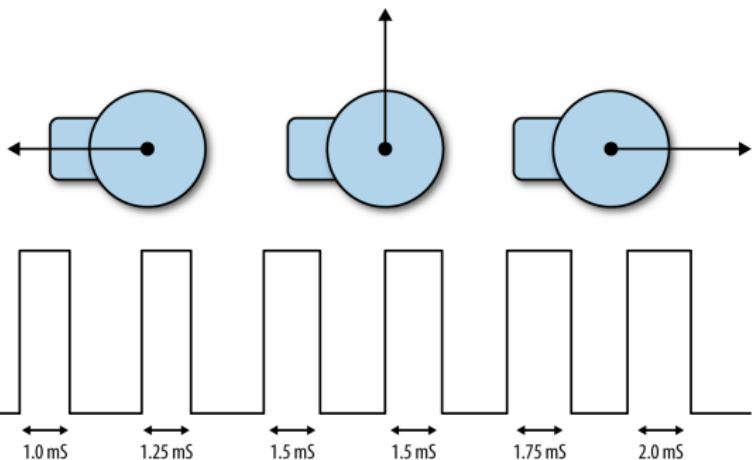


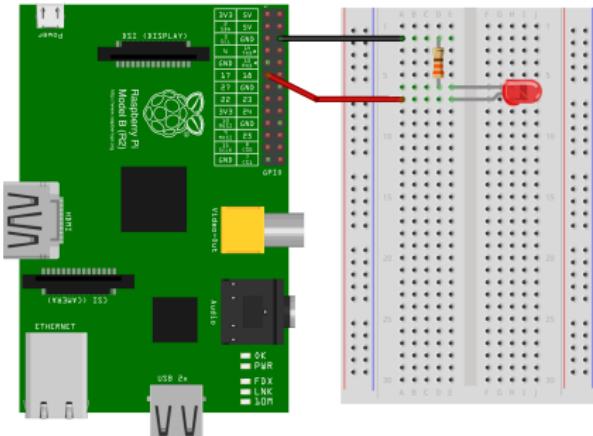
Figura : controllo del servomotore con PWM

I servomotori sono attuatori muniti di un sistema di retroazione che permette di controllarne la posizione angolare attraverso il segnale di ingresso.



Software PWM

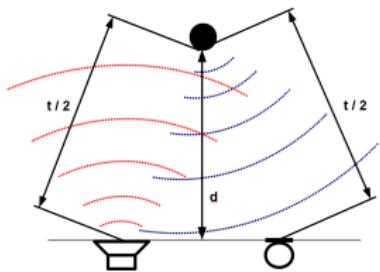
Esempio



```
35 com.pi4j.wiringpi.Gpio.wiringPiSetup();  
36  
37 // create soft-pwm pins (min=0 ; max=100)  
38 SoftPwm.softPwmCreate(1, 0, 100);  
39  
40 // continuous loop  
41 while (true) {  
42     // fade LED to fully ON  
43     for (int i = 0; i <= 100; i++) {  
44         SoftPwm.softPwmWrite(1, i);  
45         Thread.sleep(100);  
46     }  
47  
48     // fade LED to fully OFF  
49     for (int i = 100; i >= 0; i--) {  
50         SoftPwm.softPwmWrite(1, i);  
51         Thread.sleep(100);  
52     }  
53 }
```

Ultrasonic Sensor

Principio di funzionamento

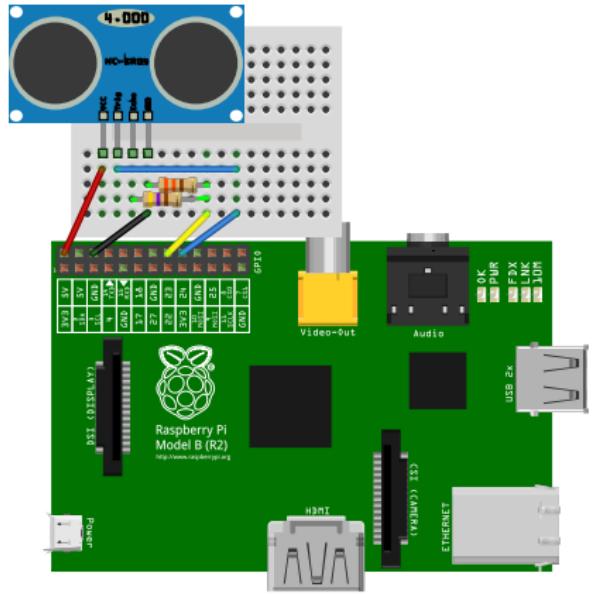


Working Voltage 5V
Output Signal 0-5V
Sentry Angle max 15 degree
Working Current 15 mA
Working frequency 40Hz

$$\text{distance} = \frac{\text{speed of sound} \times \text{time taken}}{2}$$

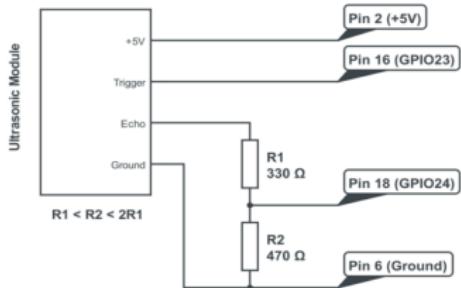
Ultrasonic sensor

Utilizzo con Raspberry Pi



fritzing

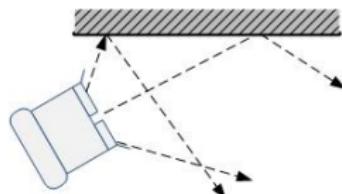
Partitore di tensione



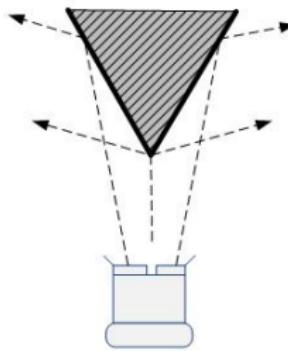
```
21 //Send trig pulse
22 digitalWrite(TRIG, HIGH);
23 delayMicroseconds(20);
24 digitalWrite(TRIG, LOW);
25 //Wait for echo start
26 while(digitalRead(ECHO) == LOW);
27 //Wait for echo end
28 long startTime = micros();
29 while(digitalRead(ECHO) == HIGH);
30 long travelTime = micros() - startTime;
31 //Get distance in cm
32 int distance = travelTime / 58;
33 return distance;
```

Ultrasonic sensor

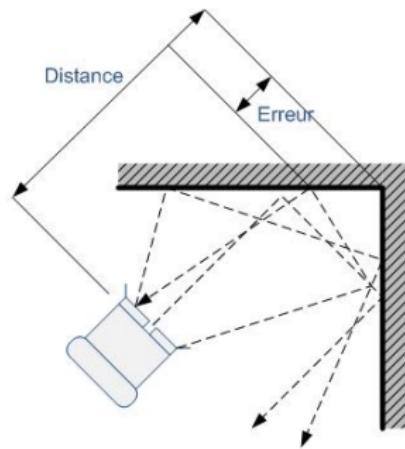
Problemi e limiti



Cas n°4

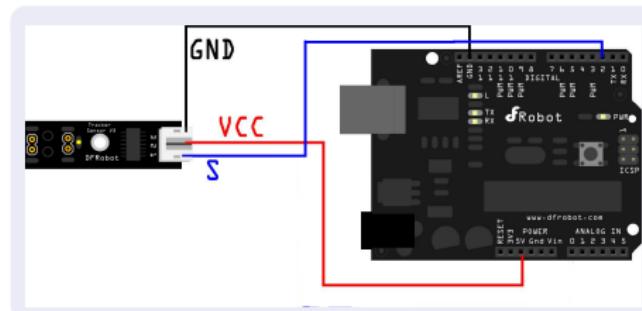


Cas n°5



Cas n°6

Line Sensor



Power supply +5V

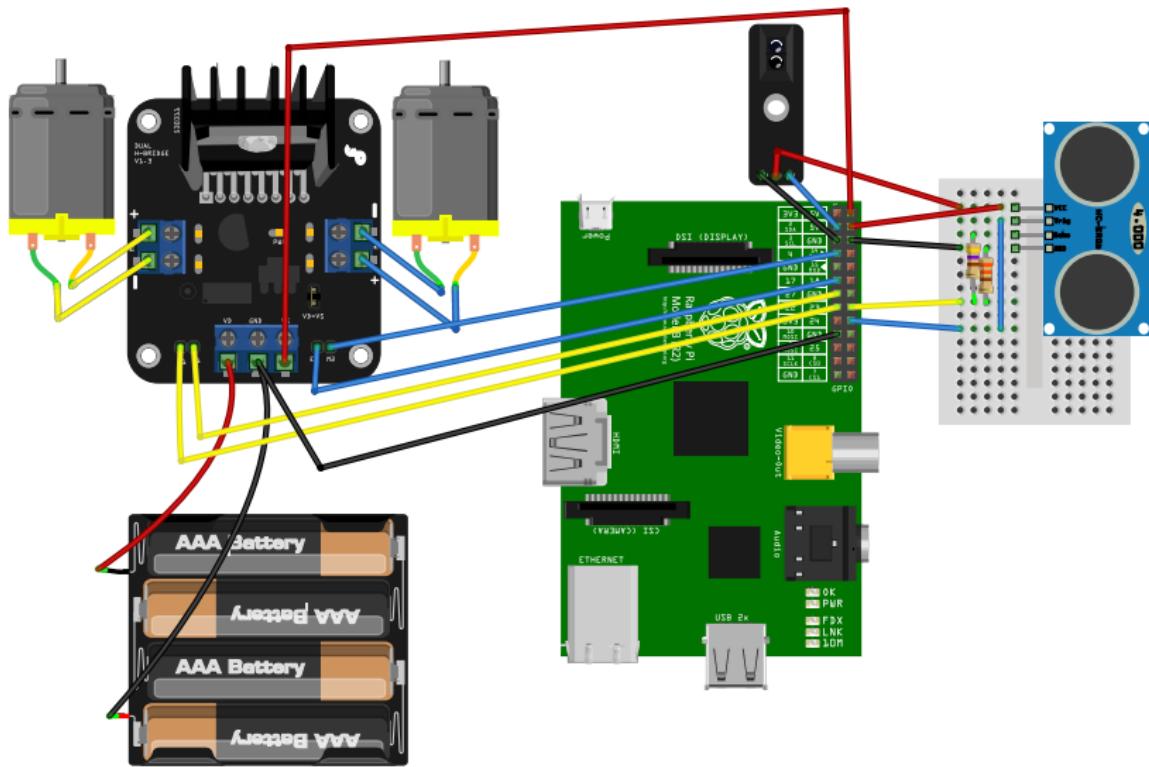
Operating current 10mA

3-wire interface

- 1 signal
- 2 power
- 3 power supply negative

```
//Arduino Sample Code
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.println(digitalRead(2)); // print the data from the sensor
  delay(500);
}
```

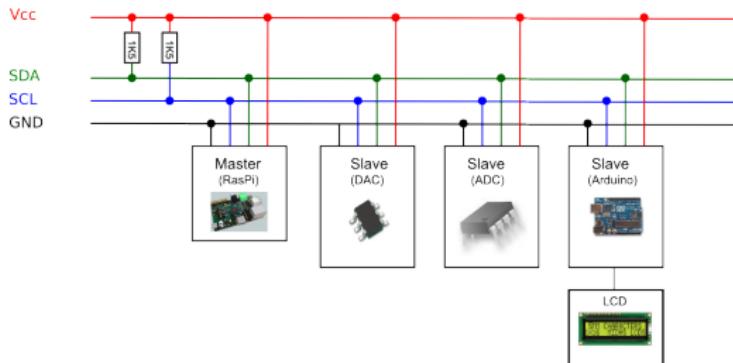
Robot



I2C Bus

Inter Integrated Circuit

- Due linee seriali di comunicazione
 - ▶ SDA (Serial DAta) per i dati
 - ▶ SCL SCL (Serial CLock) per il clock
- Ogni dispositivo è identificato dall'indirizzo univoco, e può trasmettere o ricevere
- Master - dispositivo che emette il segnale di clock
- Slave - nodo che si sincronizza sul segnale di clock



Bussola digitale

HMC5883I

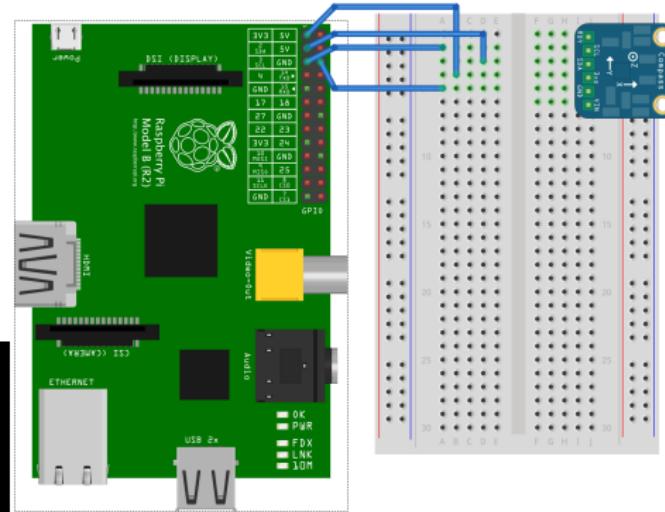
- Leggere i dati grezzi

```
int r = i2cDevice.read(address, data, 0, 6);  
  
x=((data[0]&0xff)«8) + (data[1]&0xff);  
  
y=((data[2]&0xff)«8) + (data[3]&0xff);  
  
z=((data[4]&0xff)«8) + (data[5]&0xff);
```

- Applicare le correzioni
- Calcolare la direzione

```
Math.toDegrees(Math.atan2(compassx, compassy))
```

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1  
 0 1 2 3 4 5 6 7 8 9 a b c d e f  
00: -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- 1e --  
20: -- -- -- -- -- -- -- -- -- --  
30: -- -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- -- -- -- --  
50: -- -- -- 53 -- -- -- -- -- --  
60: -- -- -- -- -- 68 -- -- -- --  
70: -- -- -- -- -- -- 77 -- -- --  
pi@raspberrypi ~ $
```



Raspberry Pi Camera



<http://www.raspberrypi.org/documentation/usage/camera/README.md>

Still resolution 5 Megapixel

Video modes 1080p30, 720p60 and 640x480p60 90

Picture formats JPEG (accelerated) , JPEG + RAW ,
GIF , BMP , PNG , YUV420 , RGB888

Video formats raw h.264 (accelerated)

```
#acquisire immagine e salvare come image.jpg  
raspistill -o image.jpg  
#salvare immagine come png (-e png)  
raspistill -e png -o image.png  
#cambiare la risoluzione  
raspistill -w 640 -h 360 -o image.jpg  
#acquisire video di 5 secondi  
raspivid -t 5000 -o video.h264  
#video con bitrate 3.5MBits/s  
raspivid -t 5000 -o video.h264 -b 3500000  
#cambiare la risoluzione del video  
raspivid -w 640 -h 360 -o video.h264
```

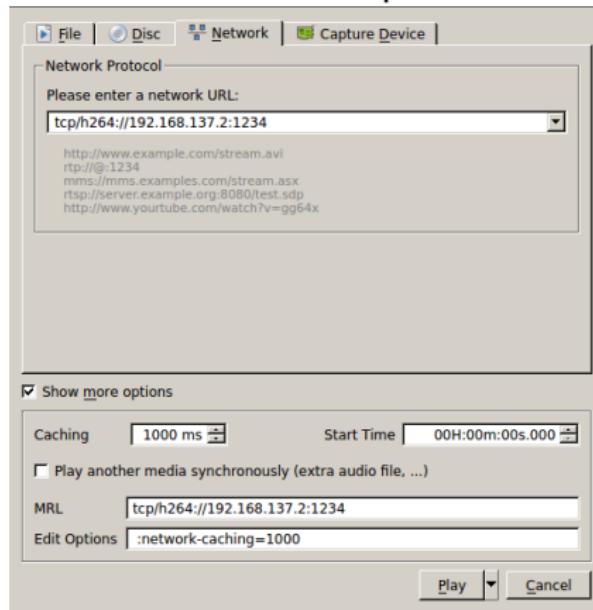
Raspberry Camera

Streaming video

- Raspberry Pi command

```
nc.traditional -l -p 1234 -c "exec raspivid -t 600000 -n -w 640 -h 360 -o -"
```

- Client side VLC setup



Link utili

Streaming video

Arduino Eclipse IDE Eclipse with WinAVR and the Eclipse plugin AVR-eclipse
<http://eclipse.baeyens.it/>

Platform IO open source ecosystem for IoT development
<http://platformio.org/platformio-ide>

NodeMCU A lua based firmware for wifi-soc esp8266
<https://github.com/nodemcu/nodemcu-firmware>

Sming framework for ESP8266 native development with C++
<https://github.com/SmingHub/Sming>

Wiring
<http://wiring.org.co/>

Wiring Pi GPIO Interface library for the Raspberry Pi
<http://wiringpi.com/>

The Untold History of Arduino
<https://arduinohistory.github.io/>

Raspberry pi custom distribution
<http://www.raspberrypi.org/documentation/usage/camera/README.md>