Laboratorio Sistemi Automatici

art. ELETTROTECNICA
prof. Michele Giandomenico

Indice

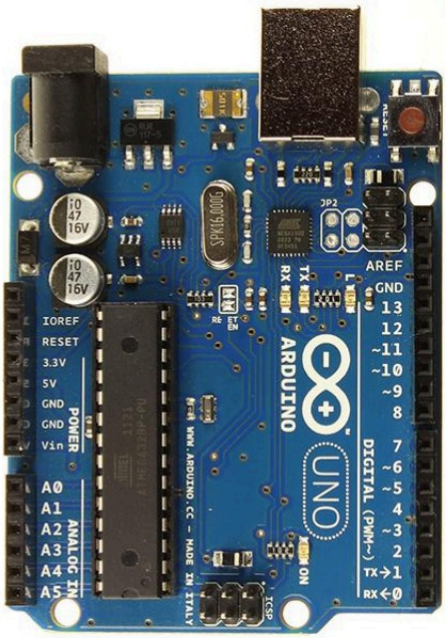
1. Introduzione

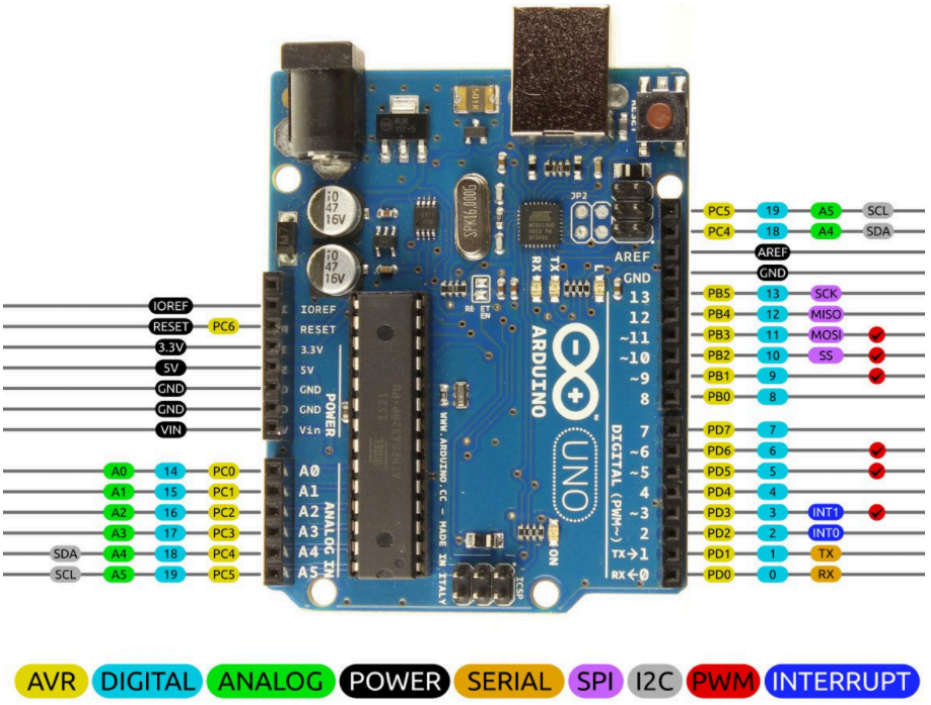
Per esplorare gli ambiti dell'**Automazione** e della **Domotica Civile ed Industriale**, nei nostri laboratori viene utilizzato il **Microcontrollore Arduino**.

Il Microcontrollore **è uno dei dispositivi più diffusi grazie ai bassi costi ed alle elevatissime prestazioni**.

video "Visita guidata ad Arduino Park"

- Microcontroller: ATmega328
- Tensione di lavoro: 5V
- Tensione di ingresso (raccomandata): 7-12V
- Tensione di ingresso (limiti): 6-20V
- Pin digitalio I/O: 14 (di cui 6 forniscono un'uscita PWM)
- Pin analogici: 6
- Corrente Continua per i pin I/O: 40 mA
- Corrente continua per l'uscita a 3.3V: 50 mA
- Flash Memory: 32 KB (ATmega328) di cui 0.5 KB usata per bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocità del clock: 16 MHz





Pinout (Piedinatura) Arduino

Che cos'è un Microcontrollore?
Il Microcontrollore è un dispositivo elettronico in grado di **interfacciarsi con il mondo esterno mediante pulsanti, sensori, ecc.** ed all'interno del quale è viene inserito un **programma software** che si occupa di monitorare ciò che avviene agli ingressi **e di eseguire i comandi previsti dal programma, attivando le rispettive uscite.**

Differenza tra Microcontrollore e Microprocessore
Il **Microprocessore** è il 'cervello' dei Computer ma per funzionare **ha bisogno di altri dispositivi di supporto** (memoria Ram, unità di calcolo, periferiche di ingresso/uscita, ecc), mentre, **il Microcontrollore, ha già a bordo tutto ciò che serve per svolgere le sue funzioni**.

Linguaggio di programmazione ed ambiente di programmazione
Il linguaggio di programmazione utilizzato per Arduino si chiama **Wiring** (derivato dal C e dal C++) ed i programmi realizzati si dicono **Sketch.**
L'ambiente integrato di sviluppo del software si definisce **IDE** (**I**ntegrated **D**evelopment **E**nvironment) e si presenta come un editor di testo.

2. Nozioni indispensabili per adoperare Arduino

Per comprenderne il funzionamento è necessario sapere, innanzitutto, cosa si intende per segnale digitale e segnale analogico:

- **Segnale digitale,** un segnale si dice digitale quando **può assumere solo due condizioni certe** (ad es. on-off, presenza-assenza, alto-basso, ecc..);
- **Segnale Analogico,** un segnale si dice Analogico quando **può assumere tutti i valori compresi in un certo intervallo** (in inglese **range**)

Pin (punti di collegamento) del Microcontrollore
I pin, **in fase di programmazione**, possono essere impostati **sia come ingressi che come uscite**. Se un pin si imposta **come ingresso** il Microcontrollore porrà quel pin ad **alta impedenza** (vuole dire che applicando i 5V verrà assorbita una corrente piccolissima). Se invece, un pin si **imposta come uscita,** il Microcontrollore porrà quel pin a **bassa impedenza** per consentire la massima erogazione di corrente. **IMPORTANTE:** dalle uscite del Microcontrollore la **massima corrente erogabile è di 20 mA (0,02A) per usi continuativi** (alimentare un componente per un lungo periodo), mentre, **per brevi periodi il massimo consentito è di 40 mA**

Ingressi
Arduino possiede ingressi **sia digitali che analogici**:

- Agli **ingressi digitali** (sul Microcontrollore contraddistinti con le cifre da 0 a 13) viene riconosciuto lo **stato logico 0** (detto anche stato logico basso, in inglese **LOW**) quando vi **saranno applicati 0 Volt.** Viene riconosciuto lo **stato logico 1** (detto anche stato logico alto, in inglese **HIGH**) quando al pin **verranno applicati 5 Volt;**
- Agli **ingressi analogici** (sul Microcontrollore contraddistinti con le sigle A0, A1….A5) si potranno applicare tutti i valori compresi **tra 0V e 5V**.

Uscite
Arduino possiede **sia uscite digitali che analogiche:**

- Se ad una uscita digitale **si assegna via software lo stato logico 0** (LOW) su quel pin **avremo 0 Volt**, se invece si assegna lo **stato logico 1** (HIGH), su quel pin **avremo 5 Volt;**
- **Alle uscite analogiche** (sul Microcontrollore sono contraddistinte dal segno **~** detto tilde) potremo far assumere, via software, **tutti i valori compresi tra 0 Volt e 5 Volt**. Questa modalità si dice pilotaggio **PWM** **P**ulse **W**idth **M**odulation, in italiano '**modulazione della larghezza dell'impulso**' ed è diventata la tecnica più diffusa per la **gestione della velocità nei motori elettrici e per la modulazione dell'intensità luminosa**.

Tipi di memoria

- **SRAM:** RAM statica **volatile**, usata dal microcontrollore **per creare e gestire le variabili** (ed i dati in esse contenuti) del programma dell'utente
- **Flash:** memoria **non volatile** facilmente riscrivibile usata **per memorizzare il programma dell'utente**

Indice

3. Interfacciamento dei pin di INPUT e di OUTPUT con i componenti esterni

Per rappresentare i pin di Arduino negli schemi elettrici si utilizzano i seguenti simboli

a)

b)

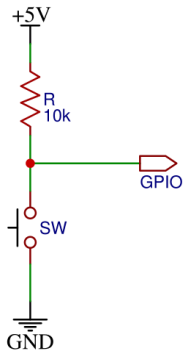
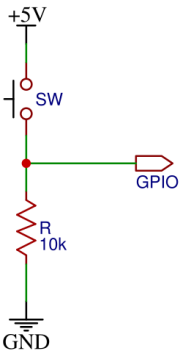
Per rappresentare un **pin di uscita** il conduttore parte **dalla punta del simbolo** (a).
Per rappresentare un **pin di ingresso** il conduttore raggiunge il pin **dal lato verticale del simbolo** (b)**.**

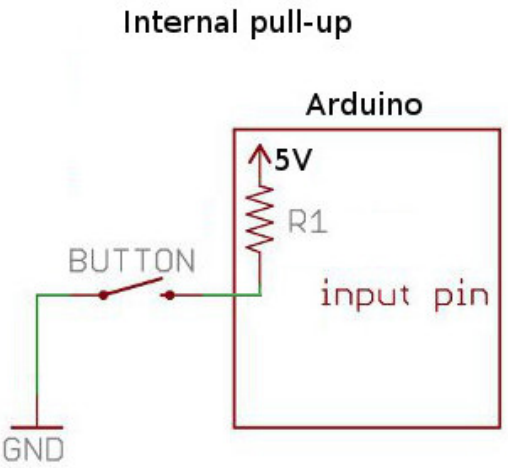
Configurazioni di interfacciamento dei pin di INPUT

Quando si adoperano **i pin di un circuito integrato** questi **non possono essere lasciati flottanti** (liberi di assumere stati logici 0 o 1 a causa di eventuali interferenze elettromagnetiche).
Per ovviare a questo problema è necessario '**forzare'** il pin ad assumere un **valore di tensione certo.**

NOTA: ricordiamo che il nostro scopo è quello di comunicare al Microcontrollore la variazione dello stato logico dell'ingresso (INPUT) affinché venga eseguito il comando previsto dal software che abbiamo caricato.
Tale variazione può essere sia il passaggio dallo stato logico 0 (LOW) allo stato logico 1 (HIGH) che il passaggio dallo stato logico 1 (HIGH) allo stato logico 0 (LOW)

Per ottenere ciò vi sono diverse tecniche ma che **prevedono tutte l'uso di una resistenza** (reale o interna) di **elevato valore (almeno 10 KΩ) per fare in modo che sul pin di input ci sia un basso assorbimento di corrente.**

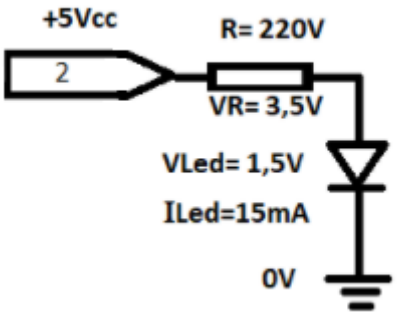
PULL-UP in questo caso il pin viene **forzato** allo *stato logico alto* ***1*** (**HIGH=5V**), la pressione del pulsante comporta il passaggio dallo *stato logico* ***1*** (**HIGH=5V**) allo *stato logico* ***0*** (**LOW=0V**)	**PULL-UP**  **PULL-DOWN**  	**PULL-DOWN** in questo caso il pin viene **forzato** ad assumere lo *stato* ***logico basso 0*** del Ground (**LOW=0V**), la pressione del pulsante comporta il passaggio dallo *stato logico* ***0*** (**LOW=0V**) allo *stato logico* ***1*** (**HIGH=5V**)



Esiste una terza soluzione, **circuitalmente più semplice**, che sfrutta la **funzione software** INPUT_PULLUP per attivare direttamente la **resistenza interna di PULL-UP** sul pin del Microcontrollore.
Nel dichiarare la modalità di pin, nel **void setup()**, si procederà nel seguente modo pinMode (button, INPUT_PULLUP);
Basterà a questo punto **collegare il pulsante tra il pin ed il ground** e tenere conto nella programmazione che, alla pressione del pulsante, la variazione sarà **da HIGH a LOW**. Se vogliamo quindi far succedere qualcosa alla pressione del pulsante 'button' dovremo scrivere

```
if (button==LOW) { //logica negativa
ciò_che_deve_succedere;
}
```

Collegamento di un carico ad un pin (OUTPUT)
Nel collegare un carico ad una uscita del Microcontrollore bisogna **porre massima attenzione alla corrente che questo assorbirà durante il suo funzionamento** ricordando che per funzionamenti continuativi **la massima corrente erogabile è di 20 mA**.



Nel circuito di esempio in figura viene alimentato un diodo Led e ciò è possibile perché la sua corrente nominale è pari a **15 mA e quindi inferiore ai 20 mA consentiti**. Sarà però necessario inserire in serie ad esso un resistore (R) per creare una caduta di tensione di 3,5V in modo che ai capi del Led venga applicata la sua tensione nominale VLed= 1,5V.
Per calcolare il valore del resistore si ricorre alla legge di Ohm.
Dato che conosciamo il valore della c.d.t (VR= 3,5V) che si vuole ottenere e la corrente assorbita dal led (15 mA) che circolerà durante il funzionamento

R= VR / ILed = 3,5 / 0,015 = 230Ω

nel circuito si userà un resistore da 220Ω perché è il valore più prossimo disponibile in commercio.

Indice

4. Procedura per stilare un programma (Sketch)
Variabili
La prima operazione è quella di **dichiarare le variabili che rappresentano gli elementi che verranno manipolati in fase di esecuzione del programma**.
Per facilitare la lettura e l'interpretazione del programma **è consigliato usare definizioni chiare** che riconducono al compito che quella data variabile ha nel programma.

I nomi delle variabili **non devono contenere degli spazi** e vanno precedute dal termine che ne **identifica la tipologia** (**int** numero intero, **char** carattere, **float** numero con virgola, ecc).

Tipo numerico	Valori numerici	Dimensione in bit
byte	Da 0 a 255	8bit
int	Da 32767 a -32768	16bit
long	Da 2147483647 a -2147483648	32bit
float	Numeri con la Virgola	32bit

Chiusura riga di comando e Commenti

- La riga di comando si conclude con **il punto e virgola ;**
- Per inserire un commento nel programma è necessario farlo precedere da un doppio slash **//**

Esempio:

```
int pin_led=2; //con questa riga di comando stiamo dichiarando la variabile pin_led
               // e le stiamo assegnando il numero intero  2
```

NOTA IMPORTANTE

Nella programmazione bisogna **porre grande attenzione all'uso del simbolo dell'uguale**.

- **un solo uguale assegna**: se scriviamo **a=1** il programma assegnerà ad **a** il valore **1**
- **due uguale confrontano**: se scriviamo **a==1** il programma confronta se **a** vale **1**

Strutture Principali e Blocchi Funzionali

Il void setup() è una **struttura principale** all'interno della quale vanno inserite le righe di comando necessarie ai settaggi di avvio. In questa parte del programma il Microcontrollore entra **esclusivamente una sola volta all'avvio**. I settaggi più comuni riguardano la dichiarazione della modalità di pin (*pinMode)* con la quale si dichiara se il pin è un ingresso (**INPUT**) o una uscita (**OUTPUT**) e l'inizializzazione della comunicazione seriale **Serial.begin** (9600); .

I blocchi di programma iniziano con una parentesi graffa aperta **{** e si concludono con una parentesi graffa chiusa **}**

Es.

```
void setup() {
pinMode (pin_led, OUTPUT);
Serial.begin (9600); //inizializzazione comunicazione seriale con baud rate 9600
}//fine void setup
```

Il void loop() è la **struttura principale** nella quale il Microcontrollore **entra ciclicamente e continuativamente** per leggere le righe di comando, dalla prima fino all'ultima, che rappresentano il programma vero e proprio.

```
void loop() {
        // in questo spazio va inserito il programma vero e proprio
}//fine void loop
```

I **Blocchi Funzionali** vengono utilizzati per organizzare le azioni eseguite all'interno del programma.
Per creare un Blocco Funzionale bisogna assegnare un nome al blocco e, opzionalmente, impostare una serie di parametri che la funzione riceverà quando viene richiamata (si dice anche invocata).

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

void loop() {
lampeggia();//viene invocato continuamente
            //il Blocco Funzione lampeggia()
}

void lampeggia(){
  digitalWrite(pin_led,HIGH);
  delay(1000);
  digitalWrite(pin_led,LOW);
  delay(1000);
}
```

nel seguente esempio nel Blocco Funzione lampeggia() si entra una sola volta

```
int pin_led=2;
int test=1;

void setup() {
pinMode(pin_led,OUTPUT);

}

void loop() {
   if (test==1){
   lampeggia();
   test=0;
   }
}

void lampeggia(){
  digitalWrite(pin_led,HIGH);
  delay(1000);
  digitalWrite(pin_led,LOW);
  delay(1000);
}
```

In questo esempio come inviare dei parametri invocando il Blocco Funzione

```
int pin_led=2;
int test=1;

void setup() {
pinMode(pin_led,OUTPUT);

}

void loop() {
   if (test==1){
   lampeggia(1000,500);//in questo modo possono
   test=0;              //essere inviati dei parametri
   }                    //da passare alle variabili dichiarate
}                      //direttamente nel Blocco Funzione

void lampeggia(int pausa_acceso,int pausa_spento){
  digitalWrite(pin_led,HIGH);
  delay(pausa_acceso);//1000
  digitalWrite(pin_led,LOW);
  delay(pausa_spento);//500
}
```

Se il blocco funzionale non dà 'in uscita' un risultato si usa il void, mentre, se darà ad esempio una stringa, si inserirà String nomefunzione ().

Funzioni

(In questo paragrafo aggiungeremo quelle che useremo man mano)

pinMode con questa funzione viene definito se il pin è un ingresso (INPUT) o una uscita (OUTPUT).
Es. pinMode (pin_led, OUTPUT);

digitalWrite con questa funzione si ordina al microcontrollore di far assumere ad un pin il **valore digitale** HIGH (5V) oppure LOW (0V)
Es. digitalWrite (pin_led, HIGH);

analogWrite con questa funzione si ordina al microcontrollore di far assumere ad un pin il **valore di tensione analogico desiderato nel pilotaggio PWM**
Es. analogWrite (pin_led, 127);// Al pin avremo circa 2,5V

digitalRead con questa funzione si ordina al microcontrollore di leggere il **valore digitale** di un determinato pin
Es. digitalRead (3);

delay con questa funzione si ordina al microcontrollore di fermare il programma per il tempo indicato tra le parentesi tonde espresso in millisecondi.
Es. delay (3000); // il programma si ferma per 3 secondi

map questa funzione consente di ridefinire un range di valori in un altro.
Es. x= map (valore_sensore, 0, 1023, 0, 255);// il range 0÷1023 della
// variabile x viene assegnata alla variabile valore_sensore rimappandola
//con il range 0÷255. Può essere utilizzata anche per invertire il range.

constrain se la variabile ha delle oscillazioni negative o va oltre il valore massimo previsto e si vogliono ulteriormente definire i limiti raggiungibili si usa la funzione y= constrain (x,0,100);

Strutture di Controllo

(In questo paragrafo aggiungeremo quelle che useremo man mano)

Grazie alle Strutture di Controllo **possiamo realizzare la logica di funzionamento** del programma sviluppato.

Dichiarazione Condizionale if
Grazie a questa struttura di controllo, **possiamo ordinare al Microcontrollore di eseguire delle azioni al presentarsi di condizioni prestabilite.**
In italiano il termine **if** si traduce con la congiunzione condizionale **se.**

```
if (questa_variabile==assume_questo_valore) {
        digitalWrite (pin_led, HIGH);      // esegui ciò che viene richiesto
      }
```

il microcontrollore entrerà in questa struttura **fino a quando la condizione sarà vera**.
La struttura è 'delimitata' dalle due parentesi graffe.
Prestare attenzione al fatto che sono stati usati due segni di uguale **==** perchè, in questo caso, **si sta facendo il confronto** se ciò che è a sinistra dei due uguale corrisponda come valore a ciò che è scritto alla loro destra, ossia, **se quella condizione è vera.**

Struttura di controllo else
Il termine **else** tradotto in italiano si traduce con l'avverbio **altrimenti.**
Nella programmazione **si usa insieme ad altre strutture di controllo** perchè, visto il suo significato, si **occupa di eseguire dei comandi quando la condizione richiesta nella struttura di controllo principale è falsa**

```
if (questa_variabile==assume_questo_valore) {
 digitalWrite (pin_led, HIGH);      // esegui ciò che viene richiesto
} else {    // altrimenti, se quello richiesto dall'if non è vero, esegui il seguente comando
 digitalWrite (pin_led, LOW);
 }
```

Ciclo for
La caratteristica del ciclo **for** è che, fino a quando non è soddisfatta la condizione prevista, il microcontrollore continuerà a ripeterlo senza proseguire nella scansione delle successive righe di codice

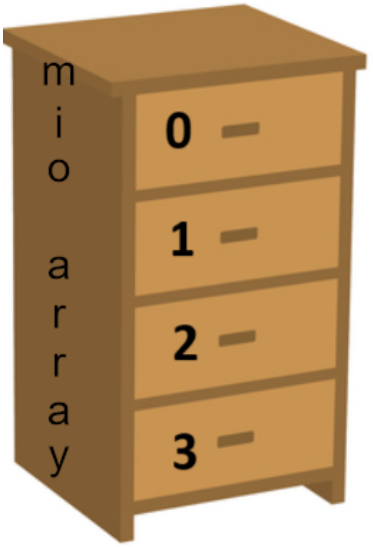
```
for (variabie=0, variabile<100, variabile++){
 esegui_questo_codice;
}
```

Gli Array

Con una variabile possiamo indicare un solo dato, **con l'array possiamo indicare tanti dati dello stesso tipo con un solo nome collettivo detto ' identificatore dell'array'**.

Gli elementi si distinguono uno dall'altro attraverso l'indice che che ne identifica la posizione all'interno dell'array.

Gli array vengono indicizzati partendo dal numero zero e quindi **il primo valore dell'array avrà indice 0** (valore dell'indice va da 0 a N-1).



Per fare un esempio pratico, potremmo associare l'Array ad una cassettiera che nei cassetti contiene elementi dello stesso tipo, per richiamare un elemento bisogna far riferimento alla cassettiera (mio_array) ed al numero di cassetto [indice] che contiene il dato desiderato, ricordando sempre che il primo cassetto ha indice 0. Se volessimo il contenuto del terzo cassetto avremmo mio_array [2].

```
//posso dichiarare il nome dell'array ed il numero degli
//elementi che lo compongono ed assegnarli in fase
//di esecuzione del programma
int mio_array[8];

//posso dichiarare il nome dell'array e gli elementi che
//lo compongono
int mio_array[] = {1, 2, 3, 4, 5};

//per richiamare un elemento bisogna ricordarsi che
//l'indice parte da 0 e non da 1
Serial.println(mio_array[0]);//stampa il numero 1
Serial.println(mio_array[1]);//stampa il numero 2
Serial.println(mio_array[2]);//stampa il numero 3
Serial.println(mio_array[3]);//stampa il numero 4
Serial.println(mio_array[4]);//stampa il numero 5
```

lo stesso risultato si può ottenre con il ciclo for

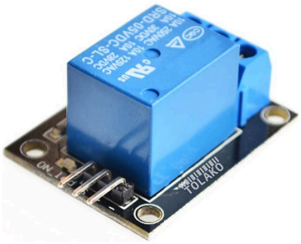
```
for(int indice = 0; indice <= 4; indice++){
Serial.println(mio_array[indice]);
delay(500);
}
```

Indice

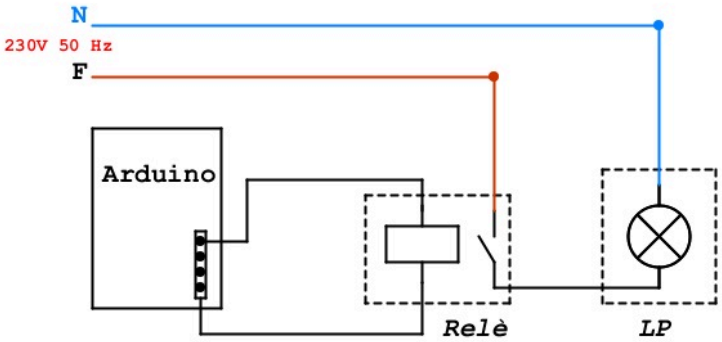
5. Interfacciamento di Potenza

se il carico collegato ai pin di Arduino assorbe correnti superiori a quelle erogabili dal microcontrollore (20 mA), sarà necessario interporre tra il pin ed il carico un componente che faccia da 'tramite' e che separi i due circuiti a potenze diverse.

I componenti più utilizzati a tale scopo sono i **Relè** ed i **Transistors**.



Schema semplificativo comando LP con relè



La soluzione mediante relè consiste **nell'alimentare con Arduino (circuito a bassa potenza) la bobina del relè che comanderà indirettamente il 'contatto pulito*' collegato al circuito a potenza superiore.**

Il limite di questo metodo è che **l'unica cosa che possiamo fare è alimentare o non alimentare l'utilizzatore**.

Va anche detto che i relè risultano essere **più 'ingombranti'** ed avendo degli organi meccanici **sono soggetti ad usura**.

***per 'contatto pulito', tecnicamente, si intende un contatto non sotto tensione**

La soluzione mediante il **Transistor ad Emettitore Comune,** usato come interruttore statico, oltre ad **alimentare l'utilizzatore** (**condizione di saturazione***) o **non alimentare l'utilizzatore** (**condizione di interdizione****), consente la **modulazione** del valore di tensione applicato all'utilizzatore stesso (pilotaggio **PWM**).

Ciò ha reso questa tecnologia la più diffusa per il controllo della luminosità e del controllo della velocità dei motori.

Arduino ci mette a disposizione delle uscite analogiche (pin **PWM**) che consentono di pilotare il Transistor via software.

L'abilità tecnica necessaria per poterli utilizzare è quella di saper dimensionare il circuito di polarizzazione del transistor in funzione della corrente assorbita dal carico.

*** condizione di saturazione:** superata la tensione **Vbe b**ase-**e**mettitore di 'innesco' che fa circolare la corrente di base **Ib** gli elettroni dalla zona dell'emettitore vengono 'richiamati/attratti' dalla tensione **c**ollettore **e**mettitore **Vce** saturando (non c'è più spazio per tutti) la zona della base, ciò causerà la circolazione della corrente **Ice** tra collettore ed emettitore. Dato che **il carico verrà collegato in serie alla maglia collettore-emettitore** potremo pilotarlo indirettamente attraverso il valore della tensione di base **Vbe e la conseguente corrente Ib**.

**** condizione di interdizione:** quando la tensione di base **Vbe** si mantiene **sotto i suoi valori di 'innesco' (Ib=0)**, di conseguenza, nella maglia **colletttore-emettitore** nella quale è inserito il carico, non circolerà nessuna corrente (**Ic=0)**.

Chiarimenti sulle condizioni di saturazione o interdizione del transistor BJT al minuto 3 del video a questo link

Esempio di calcolo

Vediamo concretamente come calcolare la resistenza di base nel caso in cui vogliamo alimentare un piccolo motore DC la cui tensione di alimentazione è 5V e la resistenza interna è di 20 Ohm con un transistor 2N2222 usato come interruttore.

$V_{CE(sat)}$*	Collector-Emitter Saturation Voltage	I_C = 150 mA I_C = 500 mA	I_B = 15 mA I_B = 50 mA			0.3 1	V V
$V_{BE(sat)}$*	Base-Emitter Saturation Voltage	I_C = 150 mA I_C = 500 mA	I_B = 15 mA I_B = 50 mA	0.6		1.2 2	V V
h_{FE}*	DC Current Gain	I_C = 0.1 mA I_C = 1 mA I_C = 10 mA I_C = 150 mA I_C = 500 mA I_C = 150 mA I_C = 10 mA T_{amb} = -55 °C	V_{CE} = 10 V V_{CE} = 10 V V_{CE} = 10 V V_{CE} = 10 V V_{CE} = 10 V V_{CE} = 1 V V_{CE} = 10 V	35 50 75 100 40 50 35		300	

Tabella: transistor 2N2222, estratto datasheets

I dati sul transistor che ci interessano e che sono disponibili sui datasheets del componente sono i seguenti:

Vbesat = 0.6 V (tensione di saturazione)

hfe = 100 (amplificazione)

Ic(max) = 600 mA (corrente di Collettore)

Il dato del motore necessario per i nostri calcoli è la sua corrente assorbita che, nel nostro circuito, corrisponde alla corrente che dovrà circolare nel collettore Ic

supponiamo Ic = 250 mA= 0,25A

Quindi:

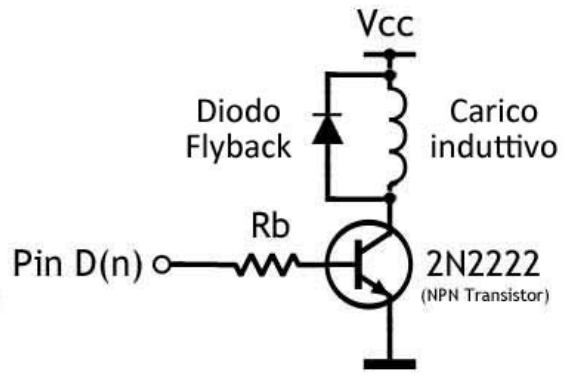
dato che hfe=$\frac{Ic}{Ib}$=> Ib=$\frac{Ic}{hfe}$= $\frac{0{,}25A}{100}$ = 0,0025 A

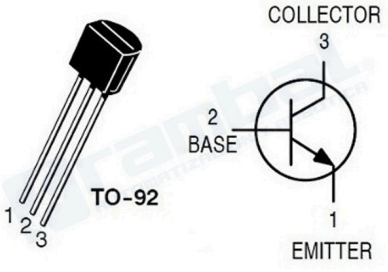
Calcolata la corrente di base **Ib** 0,0025 A, che farà circolare la **Ic** di 0,25A necessaria al motore, siamo pronti per il calcolo della Resistenza di base **Rb** che dipenderà ovviamente anche dal valore della tensione di alimentazione **Vb** che nel nostro caso è quella sul pin di Arduino di **5V**. Nell'applicare la legge di Ohm, ai 5V va sottratta la caduta di tensione sulla giunzione Base-Emettitore che corrisponde alla Vbesat (0,6V)

Rb = (5 V - 0.6 V) / 0,0025A = 1760 Ohm

Quindi, per la polarizzazione del nostro 2N222, dovremo utilizzare una resistenza di base Rb da 1,8 Kohm che è la più prossima disponibile in commercio.

Nota tecnica
I motori sono carichi induttivi che richiedono alcuni accorgimenti ulteriori per evitare che correnti di scarica possano circolare in modo incontrollato nel transistor e causare la rottura. Per evitare queste correnti **è necessario posizionare un diodo, detto *diodo di flyback* o *diodo snubber* (in italiano *diodo soppressore*), ai capi del carico induttivo con il catodo collegato al terminale connesso all'alimentazione e l'anodo collegato al terminale connesso al collettore del transistor**. Lo stesso accorgimento va realizzato anche per le bobine dei relè.



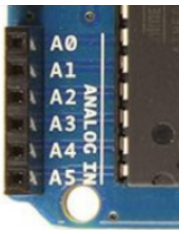


Funzionamento diodo di Flyback:

In **condizioni normali** la tensione di alimentazione del circuito **polarizza inversamente il diodo di Flyback** e, pertanto, la corrente circolerà tutta nel carico induttivo.
La sovratensione causata dall'avvolgimento (induttanza), legata dalla **mancanza improvvisa dell'alimentazione**, avrà una **polarità opposta** rispetto a quella di alimentazione e, pertanto, **polarizzerà il diodo di Flyback direttamente** e la conseguente **corrente si smorzerà nella maglia composta dalla bobina e dal diodo** senza interessare gli altri componenti che **altrimenti andrebbero distrutti**.

Indice

6. Ingressi Analogici



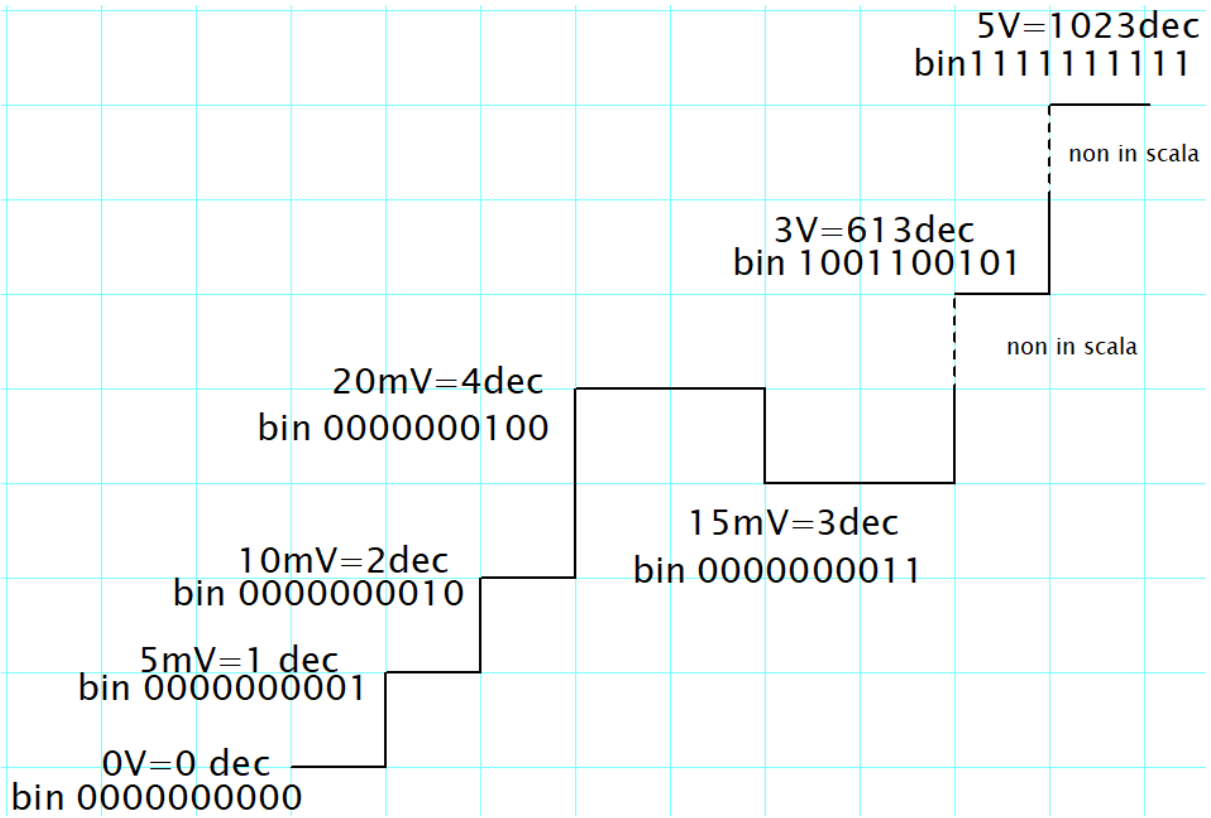
Sulla scheda sono disponibili **6** pin Analogici che vanno da **A0÷A5**. Questi pin hanno una **risoluzione** di **10 bit**. Per capire cosa si intende, quando si dice che un pin ha una risoluzione di 10 bit, dobbiamo ricordare che con 10 bit, nella codifica binaria, si possono scrivere tutti i numeri decimali che vanno da **0÷1023**.

da BIN **0000000000**=**0** DEC **fino a** BIN **1111111111**=**1023** DEC
10 bit **10 bit**

Quando un sensore deve inviare al Microcontrollore una grandezza variabile, come ad esempio una temperatura, il livello di un liquido, una pressione, ecc., lo fa **inviando un valore di tensione proporzionale al valore della grandezza che sta trasducendo** (*per **trasduzione** si intende il processo di rilevamento della grandezza fisica da parte del sensore e la sua conversione in un segnale elettrico 'manipolabile'*)
I valori di tensione applicabili ai pin analogici di Arduino vanno da **0V÷5V**.
Cosa succede quando la tensione viene applicata ad un pin analogico di Arduino?
La tensione viene convertita da segnale Analogico in un segnale Digitale (conversione A/D) e ciò, in pratica, significa che quello che era un valore di tensione viene trasformato in un numero binario (Campionamento).
Nello specifico, dato che il valore più alto di tensione applicabile ai pin analogici di Arduino è 5V che corrispondono a una sequenza di 10 bit che valgono tutti 1111111111 (dec 1023), per determinare ***il minimo incremento/decremento del segnale in ingresso** (la tensione) **che fa incrementare/decrementare il segnale in uscita** (in questo caso il numero binario) basta fare questa semplice divisione per calcolare quella che viene detta ***Risoluzione del convertitore Analogico/Digitale**

Risoluzione=5V/1023= 0,0048= 4,8 mV ≃ 5mV

quindi, **ad ogni gradino di incremento/decremento di 5 mV, vi sarà un incremento/decremento del numero binario**



in fase di programmazione il numero binario può essere manipolato sia in formato decimale DEC (metodo più adatto a noi che siamo abituati a ragionare in base 10), in binario BIN ed esadecimale HEX.

Indice

7. Pilotaggio PWM

Pulse Width Modulation (Modulazione della **Larghezza** dell'Impulso)

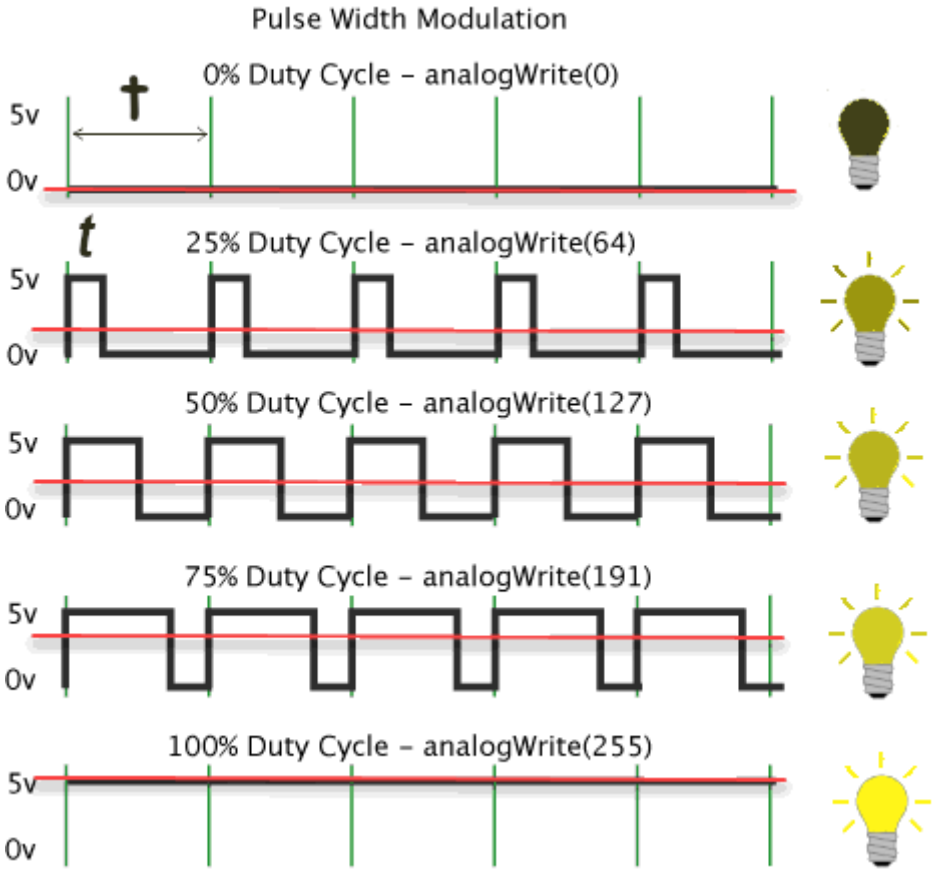
A bordo di Arduino alcuni pin sono contraddistinti con il segno della tilde ~ perché si tratta di pin 'speciali' che offrono la possibilità di essere usati come *uscite digitali* **PWM**.

Il termine inglese "Pulse Width Modulation" fa riferimento ad una tecnica di regolazione della potenza elettrica consistente nel modulare l'ampiezza/larghezza, ovvero la durata temporale, di una serie di impulsi.

Questa tecnica **non agisce quindi sul valore della tensione**, che sarà sempre quello nominale per quel dato carico, **ma sul tempo per il quale la tensione viene applicata all'utilizzatore**.

Il pilotaggio PWM ha sostituito quella che prevedeva invece l'uso della variazione potenziometrica del valore della tensione di alimentazione per il controllo di velocità dei motori o del livello di illuminamento***. Questa tecnologia prevede l'applicazione di tale segnale alla maglia base/emettitore di Transistor che effettuerà l'Interfacciamento di Potenza con il carico da pilotare**.

***** Con il metodo tradizionale, se volessimo gestire la velocità di un motore, realizzeremmo una variazione potenziometrica agendo sulla tensione che lo alimenta. Questo sistema comporta due inconvenienti: il primo e che per ridurre la potenza che arriva al motore occorre inserire in serie una resistenza che determini la necessaria caduta di tensione, con relativo spreco di potenza e produzione di calore indesiderato, il secondo è che, a tensioni molto basse, può capitare che il motore non riesca a produrre la coppia sufficiente e quindi non si avvii.



Su questi pin avremo quindi la possibilità di decidere, via software, per quanto tempo (**t**) in un determinato periodo (**T**) il segnale sarà stato logico alto (presenza dei 5V) mediante il comando **analogWrite**. Il rapporto tra **t/T** viene detto **Duty Cycle** (**ciclo di lavoro utile**) e **rappresenta** appunto **il tempo per il quale il segnale è Alto rispetto al periodo del segnale stesso**, moltiplicandolo per 100 si otterrà il Duty Cycle percentuale.
La linea rossa sul grafico rappresenta la Potenza media erogata dal pin in funzione del Duty Cicle percentuale..

$$\text{Duty Cycle\%} = \frac{t}{T} 100$$

I pin **PWM** di Arduino **hanno una risoluzione a 8 bit**, ciò significa che dato che con 8 cifre in codifica binaria possiamo scrivere tutti i numeri che vanno da **0 DEC** (BIN 00000000) a **255 DEC** (BIN 11111111) nel software potremo inserire tutti i valori compresi tra questi due estremi e potremo quindi gestire sul pin tutti i valori che vanno da **0V a 5V*** mediante il comando **analogWrite**. Di seguito alcuni esempi per gestire la luminosità di un led:

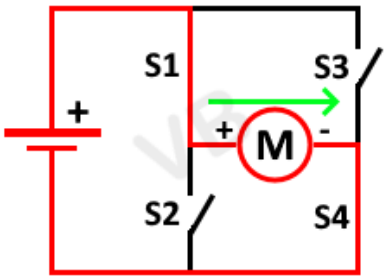
- analogWrite (pin_led, 0) sul pin avremo 0V

- analogWrite (pin_led, 51) sul pin avremo 1V
- analogWrite (pin_led, 102) sul pin avremo 2V
- analogWrite (pin_led, 127) sul pin avremo circa 2,5V
- analogWrite (pin_led, 153) sul pin avremo 3V
- analogWrite (pin_led, 255) sul pin avremo 5V

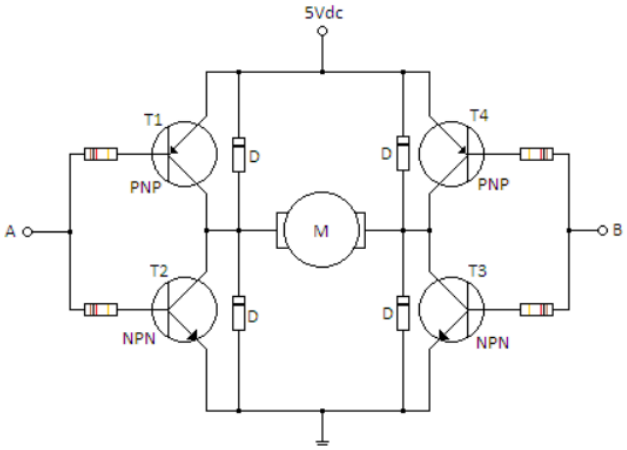
***** dato che 255/5 fa 51, per individuare il numero da mettere nel comando, sarà sufficiente moltiplicare il valore di tensione che si vuole ottenere al pin per 51 ed arrotondando eventualmente a numero intero.

Indice

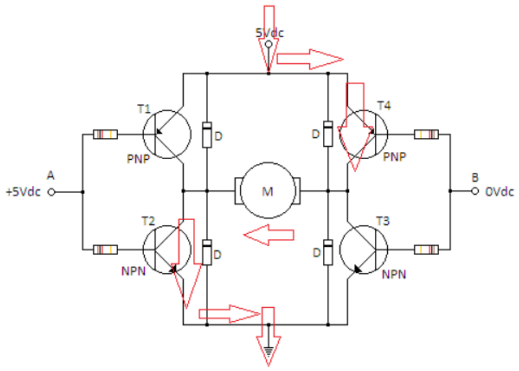
8. Ponte H L298N

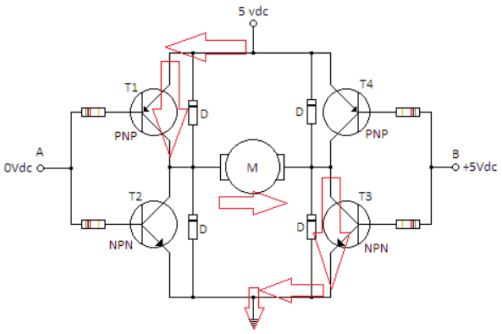


Per invertire il senso di rotazione di un motore a corrente continua è **necessario scambiare di posizione la polarità di alimentazione**. Nella configurazione a fianco vediamo una possibile soluzione che ci consente di invertire la polarità semplicemente chiudendo rispettivamente **S1** e **S4** oppure **S3** e **S2**. Si definisce ponte H (H Bridge) perchè è come se gli interruttori fossero posizionati sulle astine verticali di una H ed il motore su quella orizzontale.
I limiti di una soluzione del genere è che il pilotaggio dovrà essere **manuale**, **ingombrante** e la **velocità del motore non potrà essere gestita.**



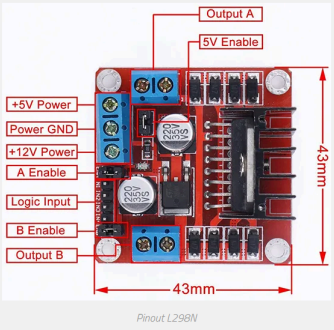
Fortunatamente, ci viene incontro l'elettronica, grazie alla quale, **con l'uso di 4 transistor usati come interruttori statici** posizionati al posto degli interruttori, è possibile effettuare il cambio di polarità semplicemente **applicando 5Vcc su un lato del ponte H e mantenendo a 0V l'altro e viceversa**. Ciò è possibile perché, su ogni lato verticale, avremo un Transistor **PNP** ed uno **NPN**, il primo dei quali condurrà quando la tensione è **0V** e l'altro quando la tensione è **5V**.





Di fatto, non si fa altro che replicare ciò che si è visto con gli interruttori ma in un **dispositivo miniaturizzato (integrato)**, **pilotabile mediante i pin del microcontrollore**

e quindi ‘programmabile’ e, cosa importantissima, potremo **gestire anche la velocità del motore mediante la tecnologia PWM**.

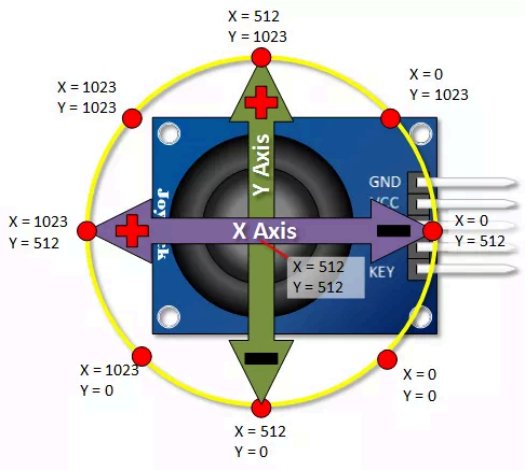


La scheda L298N ci mette a disposizione 2 ponti H e quindi potremo gestire il senso di rotazione di 2 motori DC. **Attenzione: la scheda va alimentata tra il pin +12V e il pin GND**. Il pin +5V è invece un pin dal quale possiamo prelevare tale tensione per altri usi. **Output A** e **Output B** sono i morsetti ai quali collegare i motori e sui quali avverrà il cambio di polarità.
Per pilotare l’Output A sono disponibili i due pin **In1** e **In2** mentre per pilotare l’Output B i pin **In3** e **In4**.

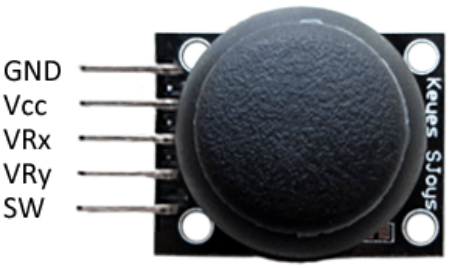
Riepilogando, se ad esempio applichiamo mediante due pin di arduino 5V ad In1 e manteniamo In2 a 0V sull'Output A avremo **+** e **-**, mentre, se manteniamo a 0V In1 ed applichiamo 5V su In2 sull’Output A avremo **-** e **+**.

Indice

9. Il Joystick analogico



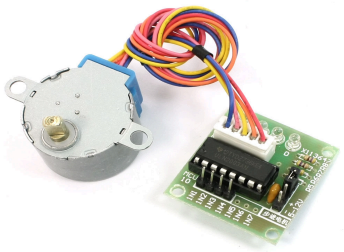
Il Joystick analogico è un dispositivo che **trasforma il movimento della sua levetta in un segnale elettrico mediante l'utilizzo di due potenziometri rotativi la cui tensione in uscita corrisponderà alla posizione assunta dalla levetta stessa**.
Per poter 'manipolare' tali segnali si ricorre agli ingressi Analogici di Arduino che hanno una risoluzione a **10 bit**, pertanto, con il movimento del Joystick avremo valori decimali che varieranno da **0÷1023** tenendo però presente che, nella posizione di riposo, ci troveremo a metà della corsa della levetta (~512)



Il pinout (piedinatura) prevede l'alimentazione **VCC**, il ground **GND**, le due uscite di tensione per gli assi **VRx** (orizzontale) e **VRy** (verticale) ed il pin dello switch **SW** che si attiva schiacciando il pomello del Joystick che collega tale pin al GND, pertanto, per poter usare tale pin, è necessario, in fase di programmazione, utilizzare la configurazione con attivazione della resistenza di Pull-up interna di Arduino.

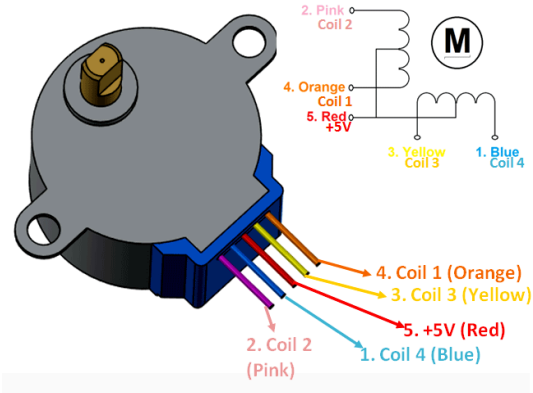
Indice

10. Motore Passo Passo



Il motore passo **appartiene alla categoria** dei motori **brushless** (senza spazzole) Il **motore passo-passo** o **Stepper** è un motore elettrico **sincrono** (**Campo magnetico statorico** e **Campo magnetico rotorico** hanno la **stessa velocità**) che può suddividere la propria rotazione in un grande numero di passi (step). Il controllo della posizione del rotore avviene mediante controllo elettronico e ciò ne consente l'uso **nei Sistemi ad anello aperto** (*non è necessario un segnale di feedback che comunichi al Sistema la posizione assunta dal rotore*).
È considerato la scelta ideale per tutte quelle applicazioni che richiedono precisione nello spostamento angolare e nella velocità di rotazione (robotica, stampanti, automazioni ecc.)

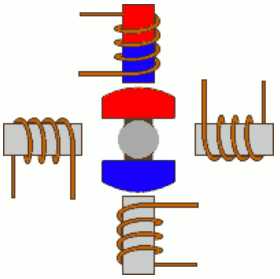
Il motore passo passo in dotazione al nostro laboratorio è del tipo a 5 fili ed è fornito della scheda di pilotaggio (**Driver**) che ci consente di pilotarlo facilmente e di **alimentarlo mediante un circuito esterno**. Ciò è necessario dato che l'**assorbimento di corrente elettrica** del motore (**55mA**) **è superiore** a quella erogabile dal microcontrollore (**20mA max**)



Il rotore del motore compie un giro completo in **64 passi** ad una velocità di circa **30 giri al minuto**. Se dividiamo i 360° di un giro completo per i 64 passi otteniamo che **ad ogni passo** il rotore si sposta di circa **5,5°** gradi. **Il numero di passi del motore ne definisce la 'risoluzione'** (grado di precisione raggiungibile con un passo), **più grande è la risoluzione e maggiore sarà la sua precisione**.

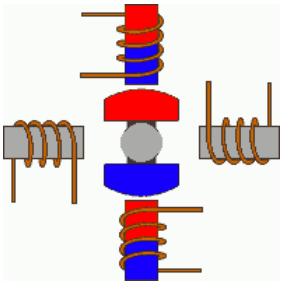
Pilotaggio ad una fase

La rotazione del rotore si ottiene alimentando in sequenza una sola fase



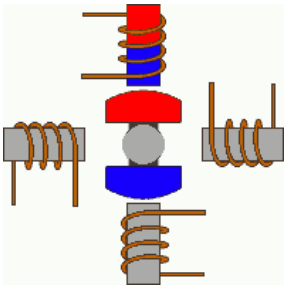
Pilotaggio a due fasi

La rotazione del rotore si ottiene alimentando in sequenza due fasi contemporaneamente (maggiore coppia motrice)



Pilotaggio a mezzo passo

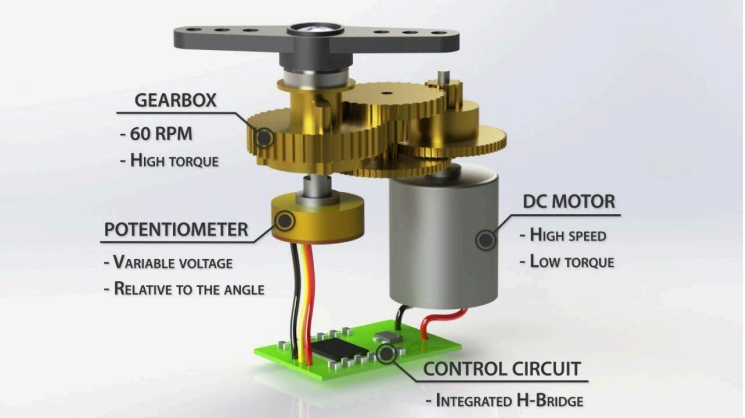
Alternando il pilotaggio ad una fase con quello a doppia fase (di due fasi contigue) si ottiene il pilotaggio a ½ passo



Indice

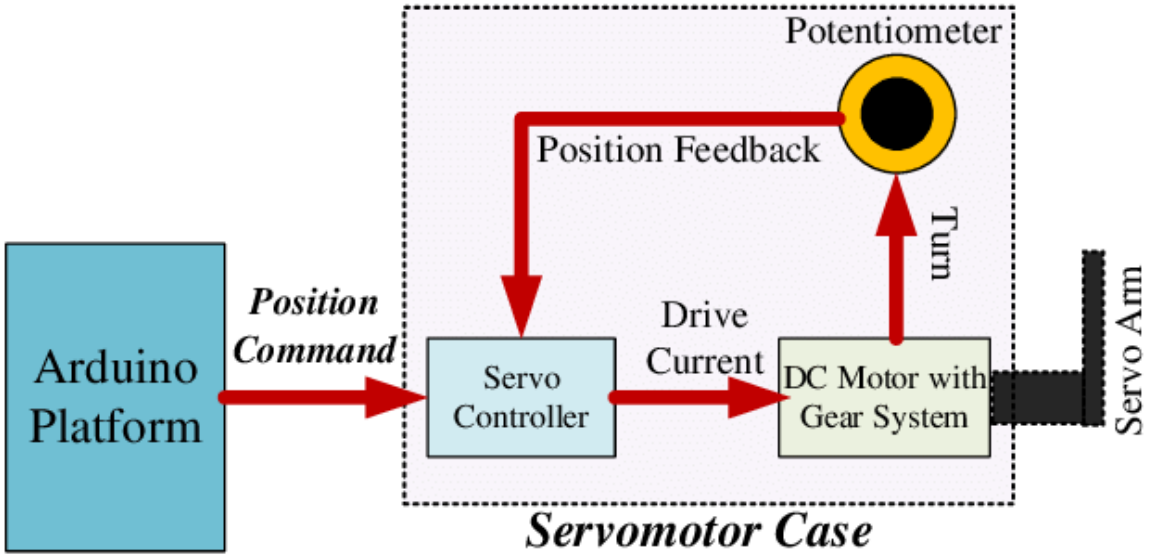
11. Motore Servo

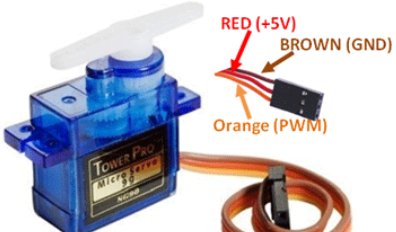
I Servomotori sono molto utilizzati per gli azionamenti nelle **automazioni industriali** e nella **robotica**. Nel modello in dotazione al nostro laboratorio l'albero è in grado di ruotare da 0° a 180° mantenendo stabilmente la posizione raggiunta. Per ottenere la **rotazione del perno è utilizzato un motore a corrente continua** e un **meccanismo di demoltiplica** (il sistema ad ingranaggi) che consente di **ridurre la velocità e di aumentare la coppia in fase di rotazione**.



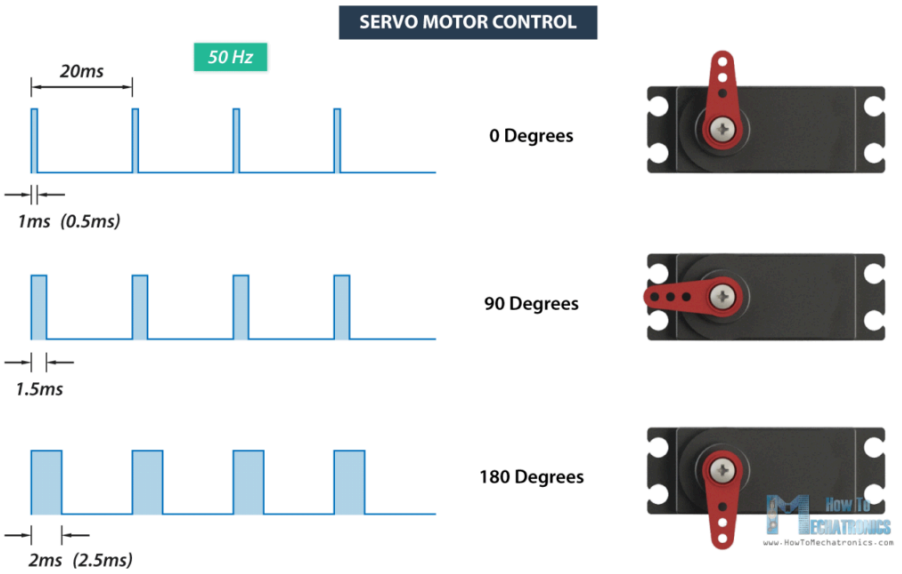
La rotazione del motore è effettuata tramite un **circuito di controllo interno** in grado di rilevare l'**angolo di rotazione** raggiunto dal perno tramite un **potenziometro resistivo**, che funge **da sensore di posizione**, e bloccare il motore **nella posizione desiderata** corrispondente al segnale PWM inviato dal pin del microcontrollore.

Il Motore Servo può essere usato nei Sistemi ad Anello Aperto (Open Loop) perchè è dotato di un Sistema ad Anello Chiuso interno che verifica autonomamente il segnale di Retroazione (feedback) e lo confronta con il segnale di posizione inviato dal Microcontrollore.





Per quanto riguarda il collegamento al microcontrollore questo si riduce all'alimentazione a 5V (rosso), al ground (nero) ed un terzo filo da collegare ad un pin PWM per il pilotaggio (arancio).

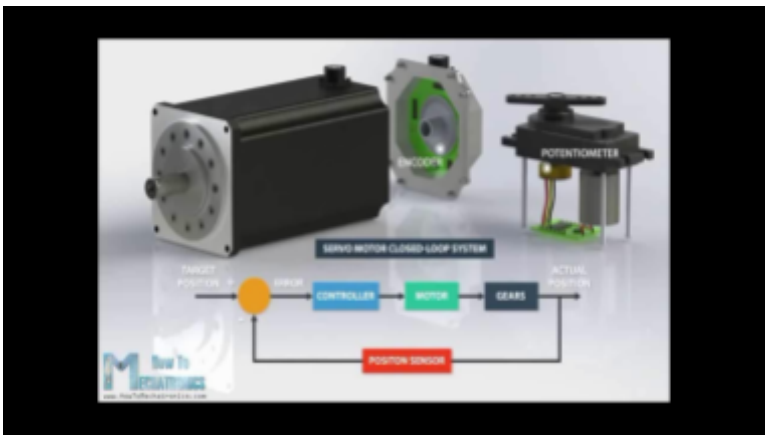


In funzione del valore di tensione inviato sotto forma di segnale **PWM** il motore **servo assumerà una determinata posizione** ma, grazie ad una apposita libreria denominata <**Servo.h**>, potremo effettuare il pilotaggio **inserendo** in fase di programmazione **direttamente l'angolo di rotazione** che si vuole ottenere senza curarci di stabilire a quale valore di tensione PWM corrisponda.

```
#include <Servo.h> //inserisco la libreria nel programma
Servo nomeservo; //assegno un nome al servo

void setup() {
nomeservo.attach(3);  //dichiaro a quale pin PWM collego il pin arancio del servo
nomeservo.write(0);  //settiamo a 0 gradi il rotore del servo
}

void loop() {
nomeservo.write(35);  //ad esempio settiamo a 35 gradi il rotore del servo
}
```



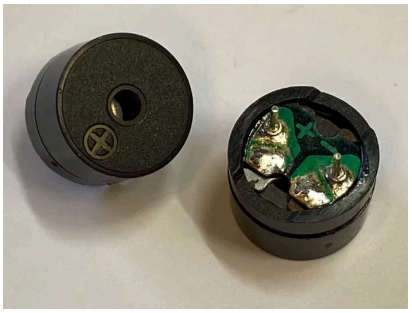
video sul principio di funzionamento

Indice

12. Buzzer

I buzzer possono essere di due tipi:

attivo



passivo

Un **buzzer attivo** usa un oscillatore interno che permette di emettere un **tono a frequenza fissa** se viene alimentato con una tensione continua (5V).

Il **buzzer passivo non possiede uno oscillatore interno** ed è quindi indispensabile un circuito in grado di generare un'onda quadra che mette in oscillazione la membrana interna del buzzer, **questi attuatori potranno così emettere toni a diversa frequenza**. Per generare questo tipo di segnale si ricorre ai pin PWM (Pulse Width Modulation) contraddistinti dal segno della tilde **~**

Per riprodurre un suono con Arduino si ricorre alla libreria interna **tone** utilizzando la seguente sintassi

tone(pin_buzzer, frequenza nota,durata);// la durata va espressa in millisecondi

Se non si inserisce la durata, per interrompere il suono, si usa la funzione

noTone(pin_buzzer);

Frequenze Note

Nota	Frequenza Hz
Do	261
Do#	277
Re	293
Re#	311
Mi	329
Fa	349
Fa#	370
Sol	392
Sol#	415
La	440
La#	466
Si	493

Indice

19. Sensori e Trasduttori

Il **Sensore** è il componente che **rileva la variazione del parametro fisico.**
il **Trasduttore** è il circuito che **trasduce*** la quantità della variazione **convertendola in un segnale elettrico 'manipolabile'.** I circuiti che si occupano di tale conversione si dicono **Circuiti di Amplificazione e Condizionamento.**
Solitamente, sono entrambi i dispositivi a costituire quello che comunemente chiamiamo Sensore.

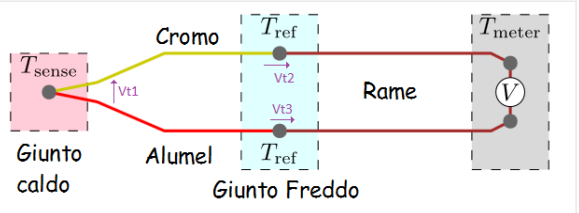
letteralmente significa **trasmettere energia*

	Trasduttore attivo	Trasduttore passivo
Differenze	genera l'uscita sotto forma di tensione o corrente, senza alcuna fonte di energia esterna che lo alimenti	I parametri interni come capacità, resistenza e induttanza cambiano a causa del segnale di ingresso
Fonte di energia aggiuntiva	Non Richiede fonte di energia aggiuntiva	Richiede fonte di energia aggiuntiva
Complessità	Semplice	Complesso
Esempi	Termocoppia, Cella Fotovoltaica	Termistore, Potenziometro
***Risoluzione**	Bassa	Alta
****Amplificazione e Condizionamento**	Richiesta	Non sempre richiesta

* minima variazione del segnale di ingresso che causa una variazione del segnale di uscita
** circuiti che rendono il segnale elettrico tale da poter essere manipolato dai circuiti di controllo

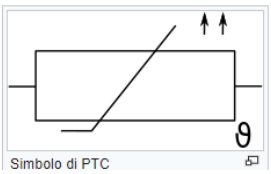
I sensori possono essere suddivisi in:

- **Sensori attivi** , quando non necessitano di altre fonti di energia perché sono loro stessi a generare la tensione di uscita
- **Sensori passivi**, quando necessitano di circuiti di alimentazione aggiuntivi



La termocoppia è essa stessa capace di generare la tensione di uscita

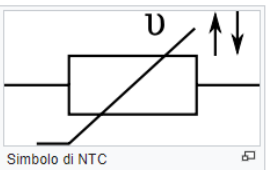
Termocoppia trasduttore attivo



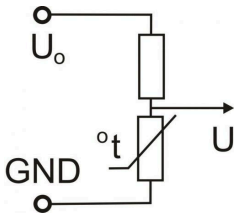
Simbolo di PTC

I termistori si possono classificare in:

- **NTC** (*Negative Temperature Coefficient*): la resistenza decresce con l'aumentare della temperatura;
- **PTC** (*Positive Temperature Coefficient*): la resistenza cresce con l'aumentare della temperatura.



Simbolo di NTC



Il termistore ha bisogno di essere inserito in un circuito alimentato per poterne sfruttare le caratteristiche

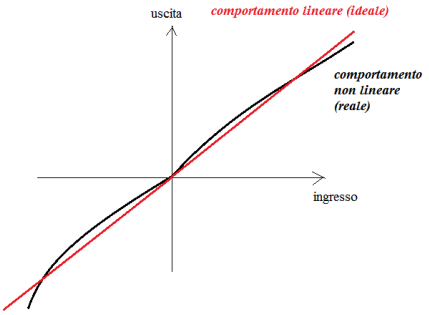
Termistore trasduttore passivo

Parametri dei Trasduttori

Campo di misura (in inglese **range)**
E' l'intervallo di valori in cui il trasduttore lavora secondo i parametri stabiliti.

Risoluzione
La Risoluzione rappresenta la **minima variazione del segnale di ingresso che causa una variazione del segnale di uscita.**

Linearità
La linearità è il parametro che misura l'errore **fra la retta ideale** di funzionamento e la **caratteristica reale** del trasduttore stesso

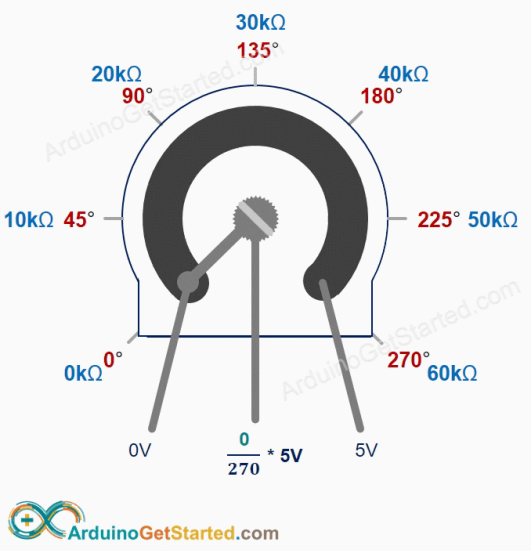


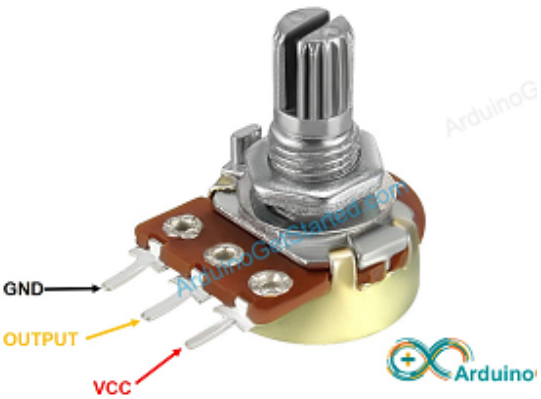
Velocità di risposta
E' il tempo impiegato dal trasduttore ad adeguare il segnale d'uscita rispetto ad una variazione della grandezza d'ingresso.

Indice

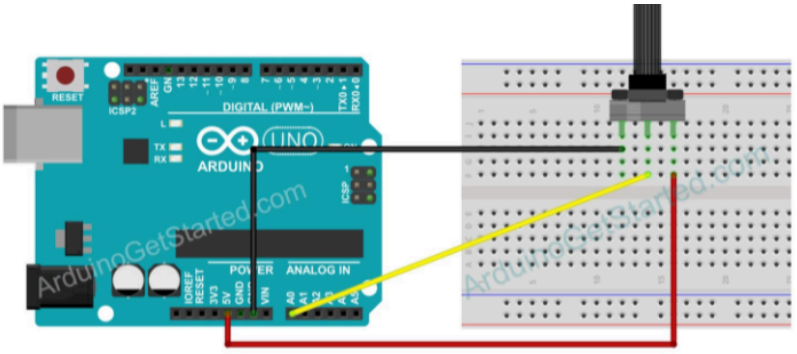
20. Potenziometro Rotativo

Il potenziometro rotativo può essere utilizzato come un **sensore di posizione angolare** dato che, applicando una tensione tra i suoi piedini esterni, **tra il piedino centrale ed il ground avremo un valore di tensione corrispondente alla posizione assunta dal contatto strisciante (cursore)** messo in movimento mediante il perno.





Per la gestione del segnale in uscita sarà necessario **collegare** il pin centrale ad un **ingresso analogico** di Arduino dato che tale segnale sarà appunto analogico potendo **assumere tutti i valori compresi tra 0V÷5V**

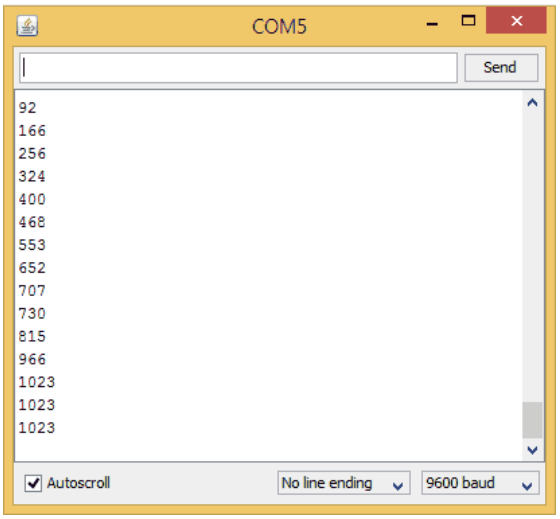


questo lo Scketch per la gestione del segnale analogico prodotto dal potenziometro

```
int pin_potenziometro _out= A0;
int valore_pin_out = 0;

void setup(){
        Serial.begin(9600); // inizializzazione porta seriale
}
void loop(){
valore_pin_out = analogRead(pin_potenziometro _out); // leggo valore potenz.
Serial.println(valore_pin_out); // scrivo il valore sulla seriale    delay(500);
}
```

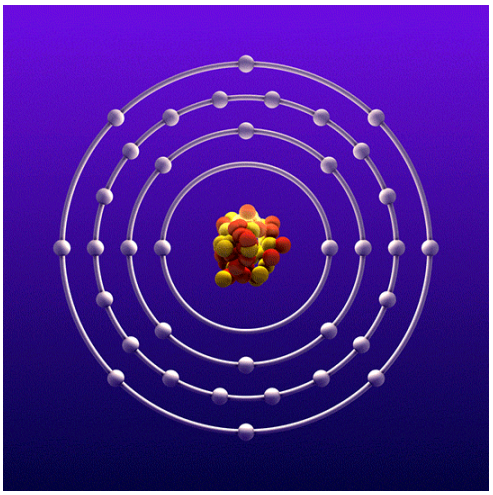
Sul monitor seriale potremo leggere tutti i valori che vanno da **0÷1023**, ricordiamo che i **pin analogici** di Arduino hanno una risoluzione a **10 bit** che corrispondono ai valori di tensione da **0V÷5V**.
A questo punto siamo in grado di **associare qualunque azione ad ogni cifra disponibile** che corrisponde a sua volta ad una **determinata posizione** del cursore.



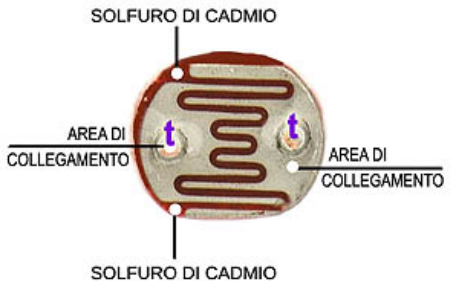
Indice

21. Fotoresistenza

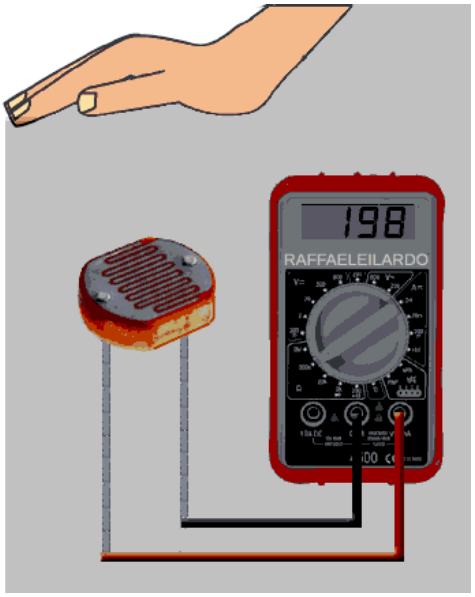
Per comprendere il funzionamento di una fotoresistenza occorre descrivere brevemente il comportamento dei materiali **semiconduttori** con riferimento alle cosiddette **"bande di energia**". In particolare, la **banda di valenza** e la **banda di conduzione** sono quelle maggiormente legate alle proprietà elettriche dei materiali.



Finché gli elettroni occupano la banda di valenza, la conducibilità elettrica del materiale è pressoché nulla.

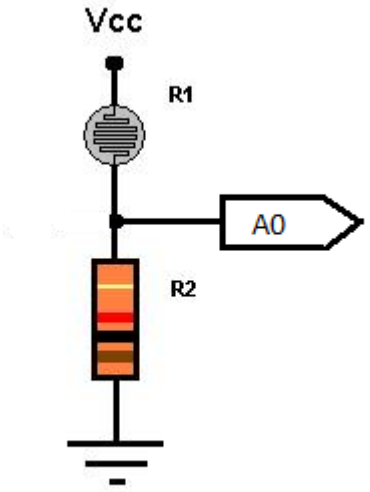


Affinché si verifichi un passaggio di corrente, **occorre che si rendano disponibili delle cariche libere di muoversi**. Tali cariche si liberano, ad esempio, per effetto di radiazioni luminose, ovvero **quando i Fotoni provocano la rottura dei legami elettrone-lacuna**, fornendo così agli elettroni **quella quantità di energia necessaria a farli transitare nella banda di conduzione**.



Gli elettroni, resi liberi dall'energia di spostarsi all'interno del materiale, permettono il passaggio di una corrente elettrica, alla quale contribuisce il movimento in senso opposto delle "lacune", cioè dei posti lasciati vuoti dagli elettroni che si sono spostati ad atomi vicini.

Il fenomeno fisico descritto prende il nome di **effetto fotoconduttivo** e si manifesta con una conducibilità elettrica che aumenta con l'aumentare della radiazione luminosa che investe il materiale.

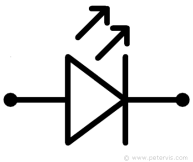


Trattandosi di un sensore passivo, per sfruttare queste sue caratteristiche, **è necessario inserirla in un circuito alimentato** ponendola in serie ad un'altra resistenza realizzando di fatto un **partitore di tensione**.
Collegando il punto centrale del partitore ad un ingresso analogico del Microcontrollore avremo una tensione che varierà in funzione della luce e che potremo 'manipolare' in fase di programmazione.

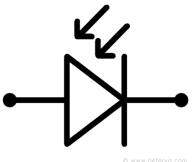
Indice

22. Fotodiodo

Per capire a fondo il funzionamento dei diodi è necessario conoscere le caratteristiche dei materiali semiconduttori e della giunzione PN.

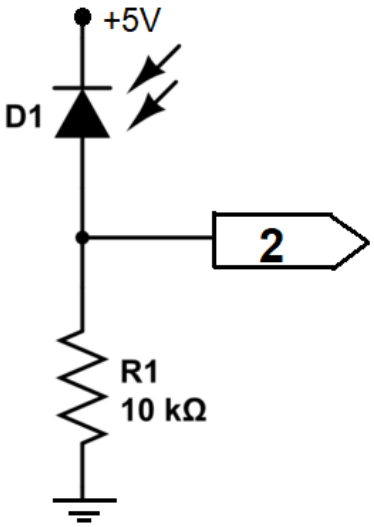


Il **LED** (**L**ight **E**mitter **D**iode) è un diodo in grado di emettere luce quando polarizzato direttamente

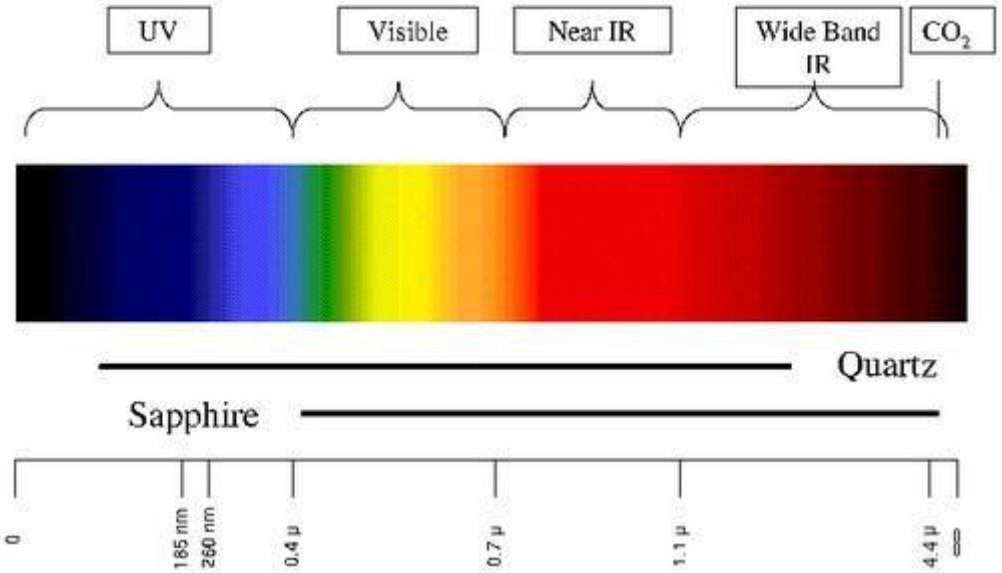


Il **Fotodiodo, invece, è un diodo che va in conduzione quando colpito da radiazione luminosa.**

La cosa importante da tenere presente nel collegamento del fotodiodo è che, quando la radiazione luminosa (i fotoni) colpiscono il fotodiodo, questi forniranno energia agli elettroni che si muoveranno dal catodo verso l'anodo (verso reale della corrente) e quindi, nel collegarlo nel circuito, sarà necessario collegarlo con polarizzazione inversa.



Il fotodiodo in dotazione al nostro laboratorio è un fotodiodo particolare perchè è sensibile alla radiazione infrarossa compresa tra i **400** ed i **700 nm** (nanometri) che corrisponde a quella **emessa dalle fiamme** pertanto, può essere usato come sensore nei sistemi antincendio.

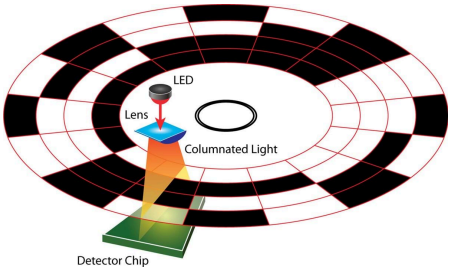


Indice

23. Encoders

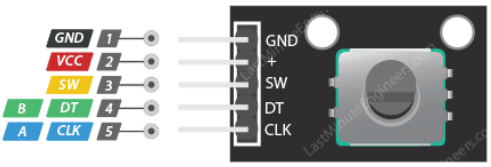
L'Encoder è un apparato che **converte la posizione angolare del suo asse rotante in un segnale elettrico digitale.**

Gli encoder possono essere sia meccanici che ottici e possono essere di due tipi:
- **Incrementali** quando i segnali d'uscita sono **proporzionali** in modo **incrementale** (**crescente** o **decrescente**) allo spostamento effettuato.
- **Assoluti** quando **ad ogni posizione dell'albero** corrisponde un valore ben definito.



ad esempio, nella posizione in cui è adesso il disco, il sensore ottico darebbe **come output il numero binario 1001 corrispondente a quella unica posizione.**

Uno degli ' Encoder in dotazione al nostro laboratorio è il KY-040 ed appartiene alla tipologia meccanico incrementale.

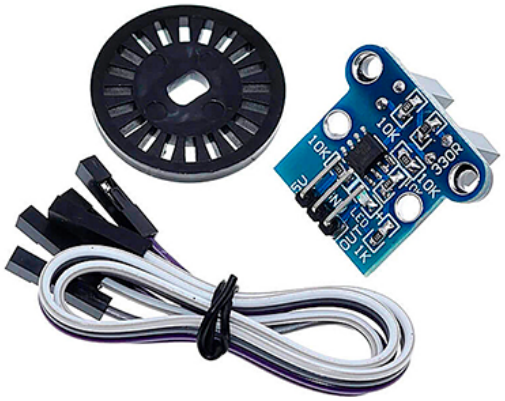


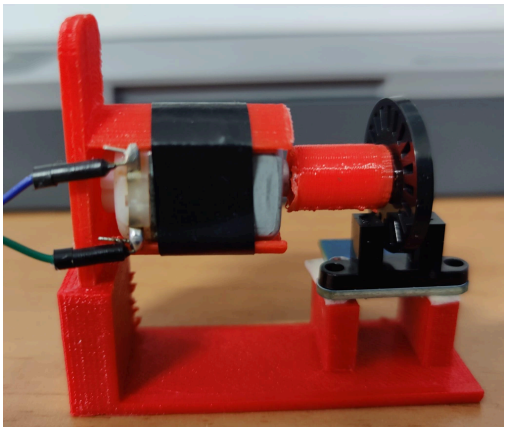
pinout KY-040

Funzionamento KY-040
Questo encoder ha un numero fisso di posizioni per giro che è pari a 30. Queste posizioni sono facilmente percepibili come piccoli "clic" che si sentono facendolo ruotare. Su un lato dell'interruttore ci sono tre pin. Normalmente vengono indicati come A, B e C.
Nel caso del KY-040, sono orientati come mostrato. All'interno dell'encoder ci sono due interruttori. La rotazione del disco collega (parte piena) o disconnette (parte vuota) I pin A e B con il pin C.

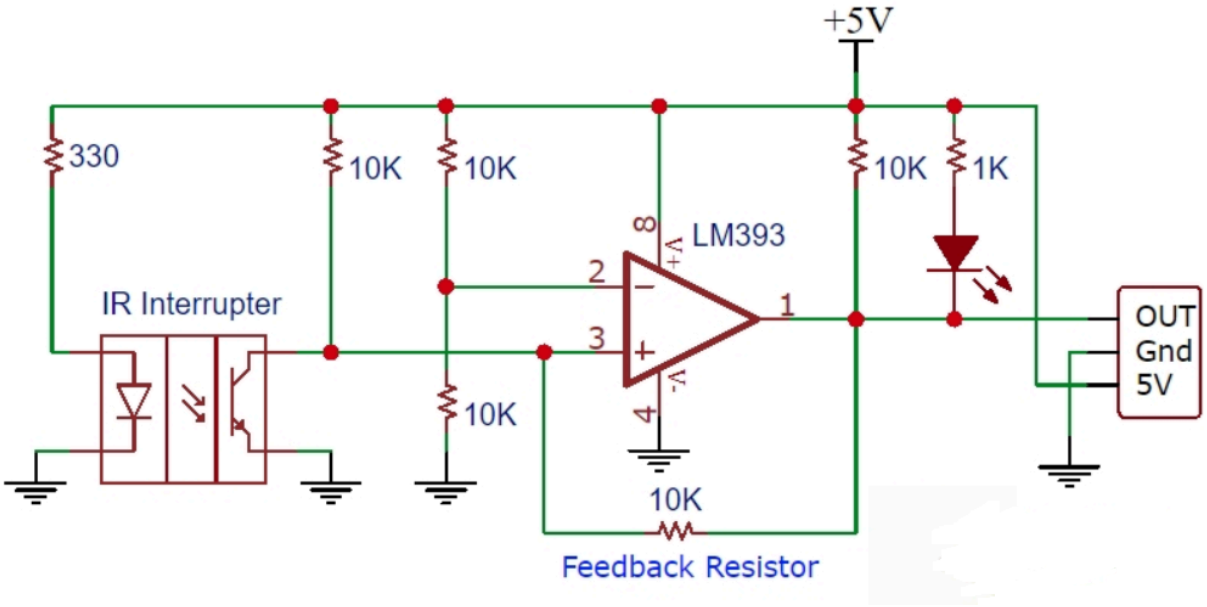
Di seguito la forma d'onda generata dalla rotazione oraria (clockwise) e antioraria (counter clockwise). Nel caso della rotazione oraria sarà il segnale A a manifestarsi prima di B, mentre, nella rotazione antioraria, si manifesterà per primo il segnale B e dopo il segnale A

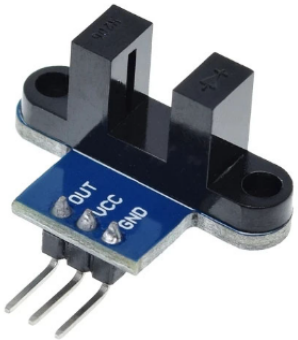
Inviando tali segnali al microcontrollore saremo in grado di stabilire **sia il verso che la velocità di rotazione** dell'alberino.
Grazie a queste sue caratteristiche trova largo uso come sensore **nel controllo dei motori elettrici che necessitano di fornire il segnale di retroazione** (feedback) nei **Sistemi ad Anello Chiuso**.





L’ altro Encoder disponibile in laboratorio, già accoppiato ad un motore c.c., è l’**HC-020K**. Si tratta di un Encoder Ottico costituito da una ruota con **20 fessure** e da una forcella che contiene un fotoaccoppiatore (diodo LED e Fotodiodo).





Riguardo ai collegamenti, oltre all’alimentazione, avremo il pin **OUT** che dovrà essere collegato ad un pin che possa funzionare da **INTERRUPT** (pin 2 o 3 di Arduino) e bisognerà quindi usare l’apposita libreria PinChangeInterrupt.h per la programmazione.

[video funzionamento encoder rotativo]

Calcolo della Velocità di rotazione
Dobbiamo innanzitutto determinare la frequenza (impulsi al secondo). Per farlo, stabiliremo un tempo per il 'campionamento', ad esempio 100ms.
Contati gli impulsi in 100 ms, tempo che corrisponde ad 1/10 di 1000 ms (1s) se li moltiplichiamo per 10 otteniamo il numero di impulsi al secondo che altro non sono che la frequenza in Hz. Ottenuta la frequenza si procede a determinare gli **RPM** (**R**otation **P**er **M**inute) con la formula seguente:

$$\mathrm{RPM} = \frac{f \cdot 60}{PPR}$$

Dove **f** è la frequenza (impulsi al secondo), **60** sono i secondi in un minuto e **PPR** (**P**ulse **P**er **R**otation) gli impulsi ad ogni rotazione.

Dato che il disco del nostro Encoder ha **20 fessure** avremo che ad ogni rotazione avremo altrettanti impulsi (**PRR** = 20), una volta calcolata la frequenza attraverso il software implementato, la formula finale diventa:

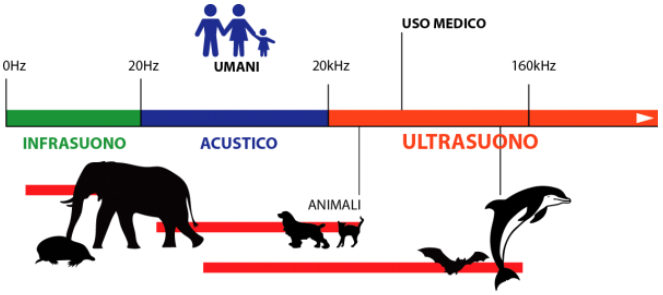
$$\mathrm{RPM} = \frac{\mathrm{frequenza} \cdot 60}{20}$$

che può essere ulteriormente semplificata dato che 60 e 20 saranno dei numeri fissi basterà moltiplicare la frequenza x 3

Indice

24. Sensore ad Ultrasuoni HC-SR4

Gli **ultrasuoni** sono onde sonore con frequenze **superiori a quelle udibili dall'orecchio umano**. Sono frequenze che superano i **20 kHz** e che trovano impiego per lo più in campo medico ed industriale.



Il sensore HC-SR04 viene utilizzato come misuratore di **distanza/livello** e **sfrutta la proprietà dei corpi di riflettere le onde sonore.**

Il sensore presenta 4 pin:
Vcc - viene collegato alla tensione di alimentazione di 5V.
Trig - è il pin "Trigger" che deve essere attivato per inviare il segnale ad ultrasuoni.
Echo - è il pin che produce un impulso che si interrompe quando viene ricevuto il segnale riflesso dall'oggetto.
GND - viene collegato agli 0V.

Funzionamento:
Un impulso della durata di 10 µS (microsecondi) viene applicato al pin Trigger.
Si genera un treno impulsi ultrasonici a 40 KHz che si allontanano dal sensore viaggiando nell'aria circostante.
Il segnale sul pin Echo diventa alto ed inizia la registrazione del tempo di ritorno in attesa dell'onda riflessa.
Se l'impulso non viene riflesso, il segnale su Echo torna basso dopo 38 ms.
Se invece il segnale viene riflesso, colpendo il pin Echo, interrompe la registrazione e tale tempo corrisponderà al tempo impiegato per percorrere il percorso di andata e ritorno. Dato che interessa conoscere la **distanza tra il sensore e l'ostacolo** sarà quindi necessario **dividere per 2 la distanza ottenuta** secondo la seguente formula:
distanza = 0.0343 * durata / 2
Come si ottiene il coefficiente **0.0343**?
Ricordiamo che la **velocità del suono nell'aria è di circa 343 m/s** e dato che la funzione pulseIn() introdotta nello sketch, che ci permette di ottenere la durata dell'impulso ALTO sul pin Echo, è espressa in **microsecondi** e che si vuole

esprimere la distanza in centimetri, è necessario trasformare la velocità del suono da **m/s** a **cm/µs**

$$343 \ \frac{m}{s} = \frac{34300}{1000000} \ \frac{cm}{\mu s} = \ 0{,}0343 \ \frac{cm}{\mu s}$$

l'**HC-SR04** è in grado di misurare distanze comprese tra i **2 e i 400 cm**.

```
//porta bassa l'uscita del trigger
digitalWrite( triggerPort, LOW );
//invia un impulso di 10microsec su trigger
digitalWrite( triggerPort, HIGH );
delayMicroseconds( 10 );
digitalWrite( triggerPort, LOW );
long durata = pulseIn( echoPort, HIGH );
long distanza = 0.034 * durata / 2;
Serial.print("distanza: ");
```

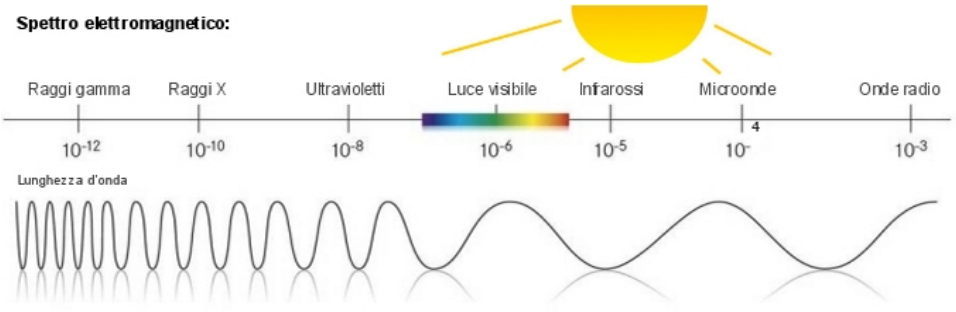
Questa la porzione di codice che 'restituisce' il valore della distanza in centimetri

---

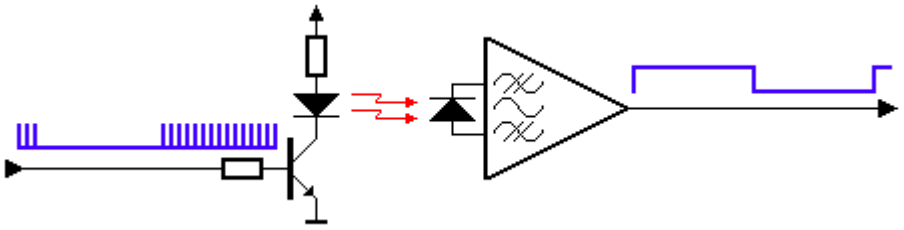
Indice

25. Sensore ad InfraRossi

La radiazione infrarossa ricade tra quelle **non visibili dall'occhio umano**.

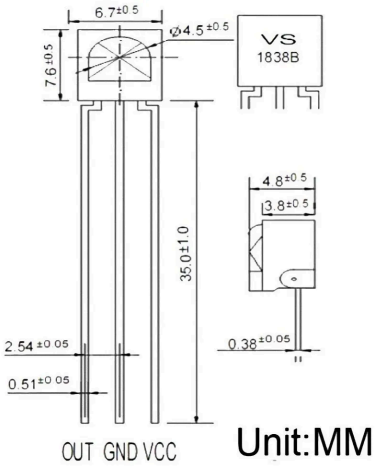


Per realizzare una **comunicazione** tra apparati elettronici digitali, mediante questo tipo di tecnologia sono necessari un **trasmettitore** ed un **ricevitore**.



Trasmettitore (LED ad infrarossi) *Ricevitore (Fotodiodo)*

Il codice binario, che rappresenta il 'messaggio' da trasferire dall'apparato trasmittente (ad es. un telecomando), viene trasformato in una **sequenza di segnali luminosi** che viaggiano nello spazio (**non vi devono essere ostacoli** tra il trasmettitore ed il ricevitore), **raggiunto il ricevitore** questi segnali luminosi **vengono riconvertiti** nuovamente in un **codice binario 'manipolabile'** dall'apparato elettronico ricevente.



Nella figura a fianco è riportato il sensore **TL1838** in dotazione al nostro laboratorio. Il suo collegamento si riduce all'alimentazione **5Vcc** (pin 3), al **Ground** (pin 2) ed il pin **OUT** andrà collegato al pin di Arduino che si deciderà di usare e che dovrà essere dichiarato tra le variabili secondo la sintassi prevista dalla libreria <IRremote.h>.
Una volta impostato lo Sketch per leggere i codici provenienti da un telecomando ad InfraRossi, questi potranno essere opportunamente usati per gestire qualsiasi pin di Arduino a distanza.

La portata di questo tipo di tecnologia è di **circa 10mt.**

Indice

26. Sensore di prossimità ad Infrarossi SG035-SZ

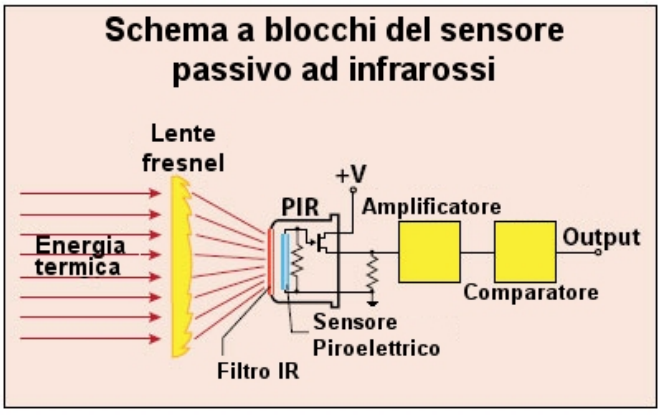
Il sensore è munito di due led uno dei quali ha la funzione di **trasmettere** il segnale ad infrarossi e l'altro di **riceverlo**. Il principio di funzionamento si basa **sulla proprietà dei corpi di riflettere il raggi infrarossi**.
L'uscita **OUT** è a **logica negativa**, ciò significa che se il segnale **non è riflesso** tale pin è **stato logico alto (5V)** mentre, **quando è rilevato un ostacolo**, il pin va **stato logico basso (0V)**. Il Range di sensibilità, da 2 a 30 cm, può essere regolato mediante un apposito trimmer. L'angolo di rilevamento è di 35°.

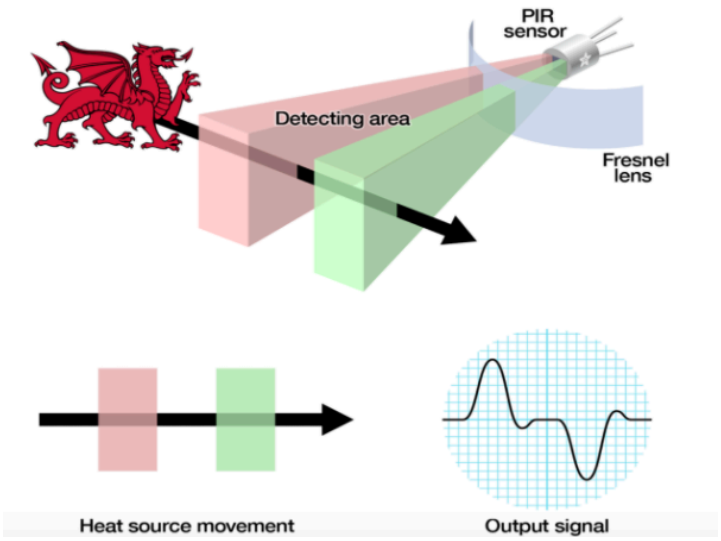
Indice

27. Sensore PIR (Passive InfraRed) HC-SR501

Il sensore ad infrarossi passivo è un dispositivo **in grado di rilevare i raggi infrarossi irradiati dai corpi** e per questo viene utilizzato come dispositivo in grado di rilevare i movimenti.

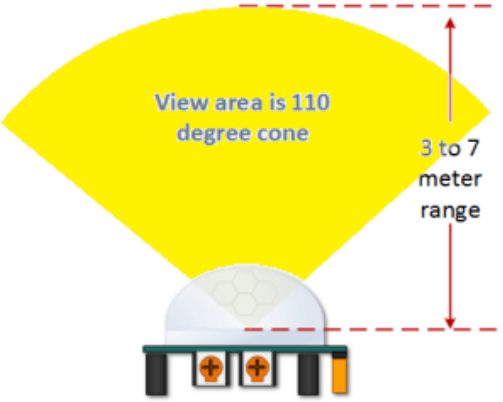
L'energia termica, attraverso la cosiddetta '**Lente Fresnel**', **viene fatta convogliare sulla parte sensibile del sensore Piroelettrico**. Il segnale prodotto ha la necessità di essere amplificato per poter innescare il cambio di stato logico dell'uscita.

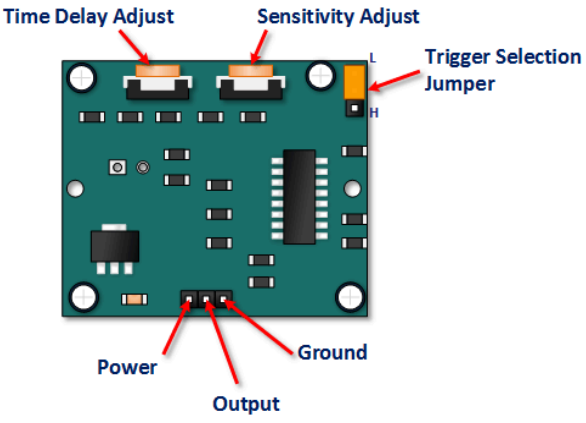




la parte frontale del sensore **ha due aree di rilevamento**, ed affinché il sensore determini che ci sia stato un movimento davanti ad esso, è **necessario che il corpo che emette i raggi infrarossi le attraversi entrambe.** Ciò per evitare false segnalazioni, ad esempio, per una variazione di temperatura ambientale.

Il dispositivo copre un angolo frontale di 110° gradi e può essere regolato per distanze da 3 a 7 metri mediante l'apposito trimmer (potenziometro).





Nella figura a fianco notiamo il potenziometro **Time Delay Adjust** mediante il quale possiamo regolare il tempo per il quale il pin **Output** rimarrà stato logico alto dopo il rilevamento del movimento (da 3" a 5'). Alla sua destra il potenziometro **Sensitivity Adjust** mediante il quale possiamo regolare la distanza di intervento (da 3 a 7 metri)

E' presente un Jumper che permette di impostare due modalità di funzionamento:

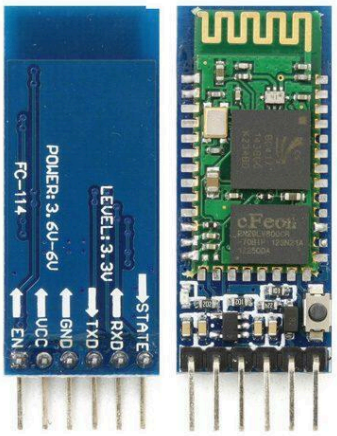
H : In questa posizione il sensore continuerà a mantenere il livello del segnale in uscita HIGH fintanto che il movimento continuerà ad essere percepito escludendo l'impostazione del Time Delay Adjust.

L : In questa posizione il sensore continuerà a mantenere il livello del segnale in uscita HIGH per il tempo definito attraverso il potenziometro Time Delay Adjust.

La tensione di alimentazione del modulo va dai 5 ai 10V.

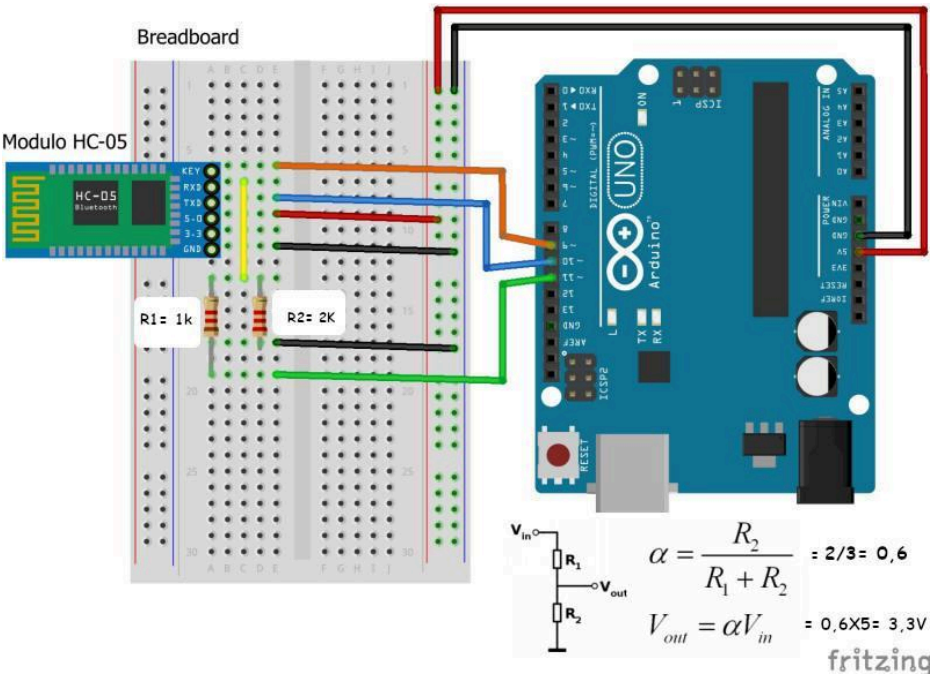
Indice

31.
Antenna Bluetooth HC-05



Il modulo **Bluetooth HC-05** è uno dei moduli più popolari utilizzati per le comunicazioni **BLE** (Tecnologia Bluetooth Low Energy).
Il modulo ha una portata di circa **10 mt** (Classe 2), si **imposta** facilmente tramite comandi **AT** ed è programmabile sia come **Master** che come **Slave**. Questo dispositivo viene usato quando si vuole far comunicare un microprocessore o un microcontrollore con il mondo esterno e l'interfaccia può essere uno SmartPhone, un Personal Computer, o qualunque altro dispositivo fornito di una connessione Bluetooth.
i comandi **AT** (**AT**tention) sono un tecnica di comunicazione seriale che **consentono di interagire con i moduli Bluetooth per conoscerne o modificarne le impostazioni**.

Collegamenti
Il modulo andrà alimentato tra i pin VCC (5V) ed il GND (Ground 0V), il pin **TXD** (Transmitter Data) del modulo andrà collegato al pin di Arduino che è stato dichiarato come **RX** (Receiver) mentre, quello di Arduino che è stato dichiarato come **TX** (Transmitter), andrà collegato al pin **RXD** (Receiver Data) del modulo ma facendo **attenzione** al fatto che il **modulo lavora con valori di tensione di 3,3V**, sarà quindi necessario farlo attraverso un **partitore di tensione** che riduca i 5V in uscita dal pin di Arduino nella corretta tensione di funzionamento del modulo.

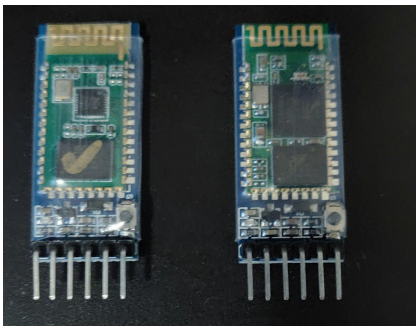


Programmazione

Per quanto riguarda la programmazione di Arduino, si rende necessario utilizzare la libreria SoftwareSerial.h e definire i pin che verranno usati come TX ed RX.

```
#include <SoftwareSerial.h>
    //definisco pin RX e TX da Arduino verso modulo BT
    #define BT_TX_PIN 12 //qui va collegato il pin RX del bluethoot attraverso
                         //partitore di tensione 1k/2k perchè tensione max 3,5V
    #define BT_RX_PIN 11 //qui va collegato il pin TX del bluethoot
    //istanzio oggetto SoftwareSerial (il nostro futuro bluetooth)
    SoftwareSerial bluetooth(BT_RX_PIN, BT_TX_PIN);

```



Attenzione: il nostro laboratorio dispone di due tipologie di antenne HC-05, per utilizzare i comandi AT nel modello di destra (Old) è necessario impostare il baud rate della seriale di Arduino a 9600 e quello dell'antenna a 38400, mentre, per il modello di sinistra (New), dovranno essere entrambia settate a 38400.

```
Serial.begin(9600);  //Inizializziamo l'interfaccia seriale al baud rate dell'AT-mode
bluetooth.begin(38400);  //Inizializziamo l'interfaccia del modulo bluetooth sempre
                         //al baud rate riferito alla modalità AT
//Serial.begin(38400);   //Per il modello NEW è necessario anche il Serial di arduiino a 38400
//bluetooth.begin(38400);//Inizializziamo l'interfaccia del modulo bluetooth sempre
                         //al baud rate riferito alla modalità AT
```

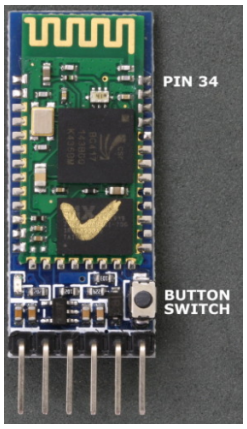
Attivazione modalità Command Mode

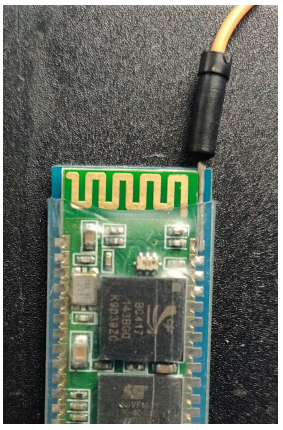
Per attivare la modalità **Command Mode,** che consente di inviare i comandi **AT**, è necessario, una volta collegato il modulo ad Arduino, tenere premuto lo switch

presente sul dispositivo prima di alimentare quest'ultimo, una volta alimentato, **il lampeggiare con una frequenza di 2" del led posto sul modulo, indicherà che ci troviamo nella modalità Command Mode** e potremo quindi rilasciare lo switch.

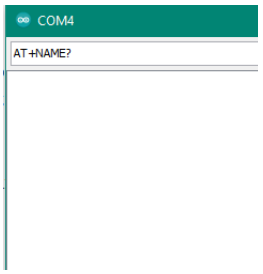
ATTENZIONE: per l'antenna NEW, prima di poter interagire con l'antenna, è necessario porre il pin 34 allo stato logico alto (5V), ciò viene facilitato dalla presenza della guaina in gomma che riveste l'antenna, infatti, si potrà inserire un jumper che verrà mantenuto in posizione dalla guaina stessa, mentre **l'altro capo del jumper andrà collegato al pin 5V di Arduino**.

Per il modello Old l'altro capo del jumper andrà collegato al pin 3,5V di Arduino.

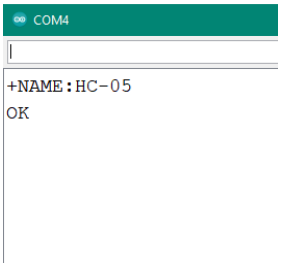




Per interagire con il modulo BT si ricorre al monitor seriale dell'IDE di Arduino



comando inviato



risposta del modulo HC-05

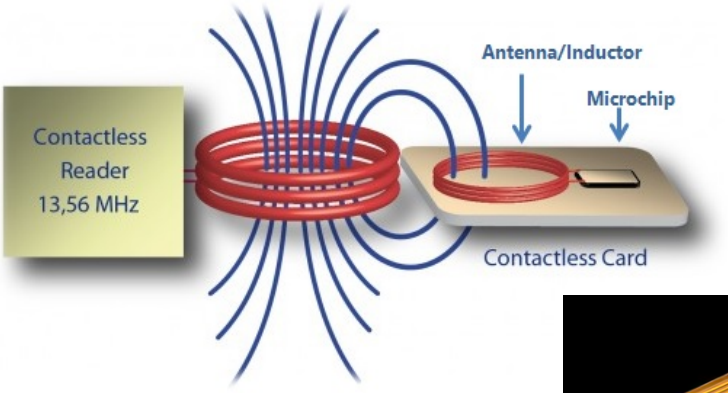
AT	Testa la connessione	AT+PSWD?	Password attuale
AT+ADDR?	Mac Address	AT+PSWD=	Assegnare password al disp.
AT+NAME?	Nome del dispositivo	AT+ROLE?	Ruolo disp. 1=Mater/0=Slave
AT+NAME=	Assegnare nome al disp.	AT+UART?	Richiesta parametri UART (Baud Rate, bit stop, bit parità)

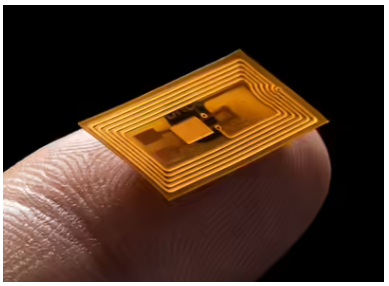
Comandi AT più comuni (da scrivere in maiuscolo)

Indice

33. Tecnologia RFID/NFC Radio **F**requency **I**dentifier**/N**ear **F**ield **C**ommunication

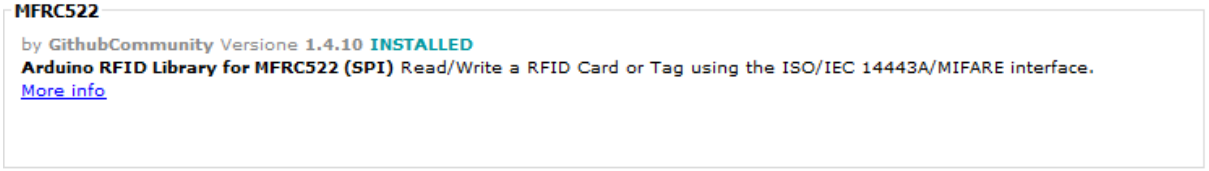
La tecnologia **NFC** (letteralmente Campo di Comunicazione Ravvicinato) è diventata estremamente diffusa grazie alle sue caratteristiche che la rendono **praticamente impossibile da hackerare** dato che, lo scambio di dati, avviene esclusivamente se i due dispositivi, **lettore e Tag**, si trovano ad una **distanza inferiore ai 10 cm**.
Il principio di funzionamento si basa sul fenomeno dell'Induzione Elettromagnetica. Il dispositivo che deve leggere il Tag sarà l'unico ad essere alimentato perchè dovrà creare un campo magnetico variabile che andrà a concatenarsi con l'antenna (avvolgimento/bobina) **posta sul Tag**. Ciò causerà nel Tag la formazione di una **Forza Elettromotrice Indotta** che alimenterà il Microchip che emetterà la **Radiofrequenza** contenente i dati **UID** (**U**nique **ID**) da trasmettere al lettore.

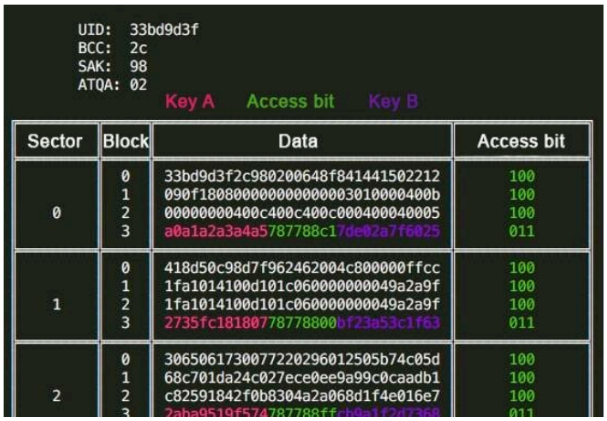




il fatto che **il Tag non necessiti di alimentazione elettrica** è stato un altro elemento che ha causato la diffusione di questa tecnologia perché **si presta ad essere inserito in supporti estremamente ridotti** quali cards, adesivi, etichette antitaccheggio, ecc.

La scheda RC522 usa il protocollo di comunicazione SPI ed una tensione di alimentazione di 3,3V. Per il suo funzionamento, oltre all'alimentazione ed il ground, occupa 5 pin del Microcontrollore (non viene utilizzato solo il pin IRQ).
Consente la lettura e la scrittura di dati su Tag del tipo MIFARE.
Per realizzare il software con l'IDE di Arduino è necessario installare la libreria





L'**UID** del tag **MIFARE classic** da **1Kb** è memorizzato nel **settore 0** del **blocco 0** ed occupa 4 byte (32 bit) ed è codificato in esadecimale, nell'esempio è **33 bd 9d 3f** .
La memoria è costituita da 16 settori (0➗15) ognuno costituito da 4 blocchi.

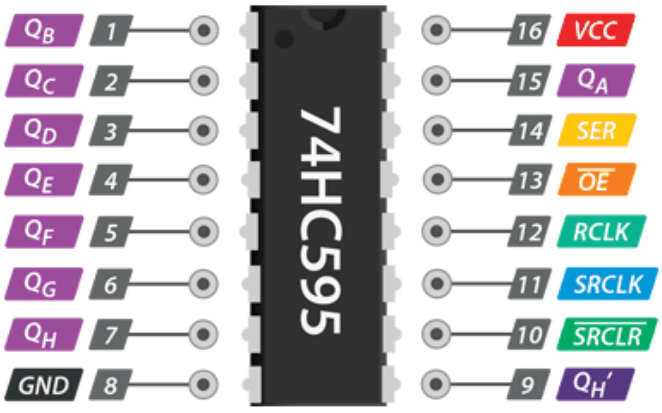
Con appositi programmi ed Arduino è possibile scrivere per memorizzare dati all'interno dei blocchi utilizzando la codifica ASCII in esadecimale.

Indice

34. Shift Register (Registro a Scorrimento)

Lo Shift Register è un integrato in grado di ricevere in ingresso, ad ogni segnale di clock, dei bit in serie e di renderli disponibili in parallelo in uscita quando il pin Latch passa stato logico alto.

animazione



Pinout dell'integrato

SER (Serial Input) il pin viene utilizzato per inviare i dati nel registro a scorrimento un bit alla volta

OE (Output Enable) è un pin attivo-basso: quando portato HIGH, i pins di uscita sono disabilitati (impostati sullo stato di alta impedenza). Quando è LOW, i pin di uscita funzionano normalmente

RCLK (Register Clock / Latch) è un pin molto importante. Quando questo pin viene portato ALTO, il contenuto dello Shift Register viene copiato nello Storage/Latch Register, che alla fine appare in uscita. Quindi, il pin può essere visto come l'ultimo passaggio prima di vedere i nostri risultati in uscita

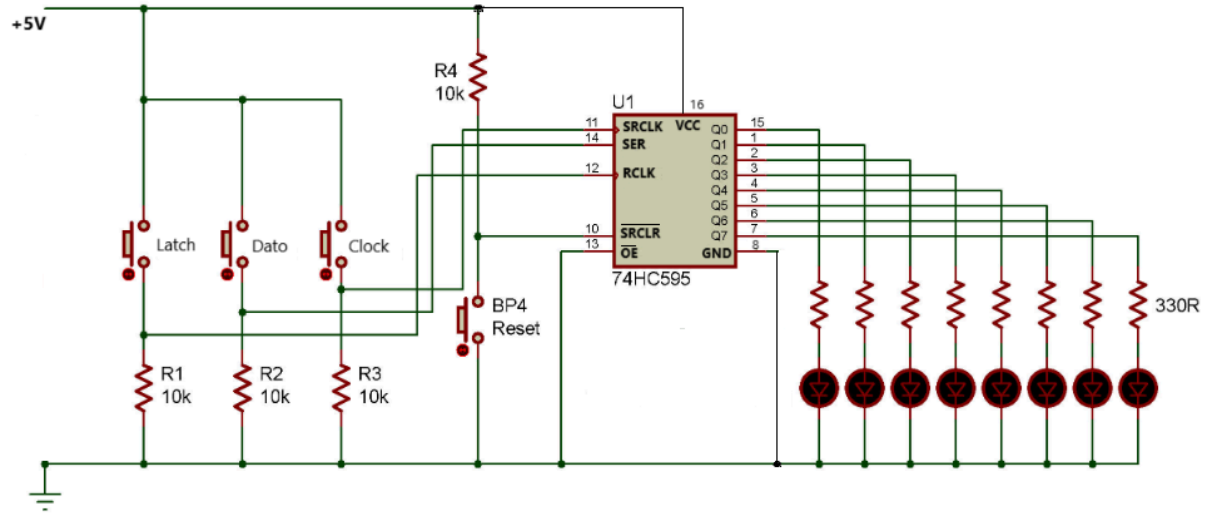
SRCLK (Shift Register Clock) è il clock per il registro a scorrimento ed è attivato dal fronte positivo. Ciò significa che i bit vengono inseriti sul fronte di salita del clock

SRCLR (Shift Register Clear) il pin ci permette di azzerare l'intero Shift Register, azzerando tutti i bit. Poiché questo è un pin attivo-basso, dobbiamo portare il pin SRCLR LOW per eseguire il reset

QA–QH (Output Enable) pins di output.

QH' pin per collegamento a cascata di un ulteriore Shift Register

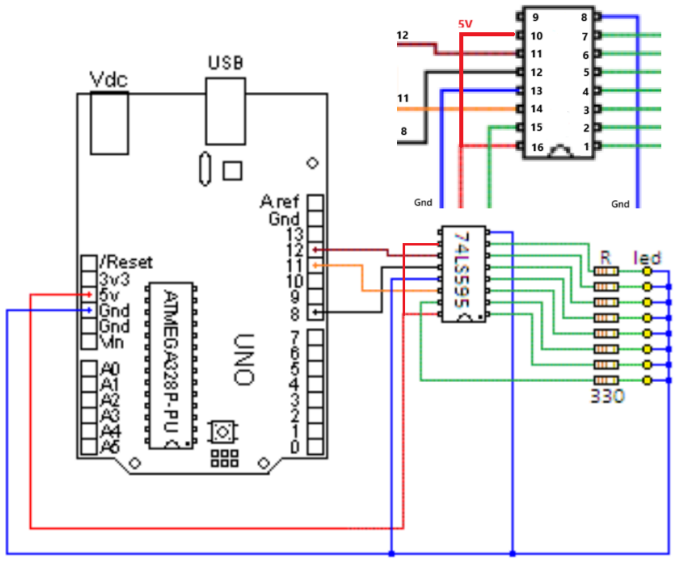
Per capirne bene il funzionamento è il caso di realizzare prima il circuito per pilotare l'integrato manualmente, mediante dei pulsanti, per poi passare al pilotaggio mediante il Microcontrollore.



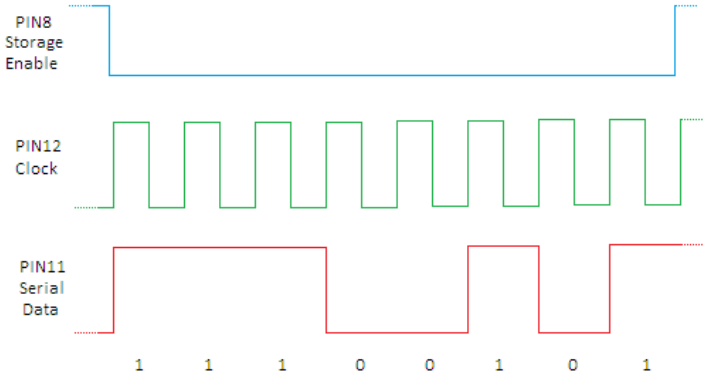
Circuito per pilotaggio manuale

Pilotaggio manuale

- Se si preme il tasto **Clock** mentre il tasto **Dato è aperto** (non premuto)si inserirà nel registro un **bit 0** ;
- se si preme il tasto **Clock** mentre il tasto **Dato è chiuso** (premuto) si inserirà nel registro un **bit 1 ;**.
- **Per avere i bit,** che sono stati inseriti in serie, **parallelizzati sulle uscite, bisogna premere il tasto Latch.** Con **Latch**, in elettronica, si intende un circuito in grado di ricevere segnali in forma binaria;
- l'uscita con il **bit 1** accenderà il led (5V) mentre l'uscita con il **bit 0** rimarrà spenta (0V).



Circuito per pilotaggio mediante Microcontrollore



Pilotaggio mediante Microcontrollore

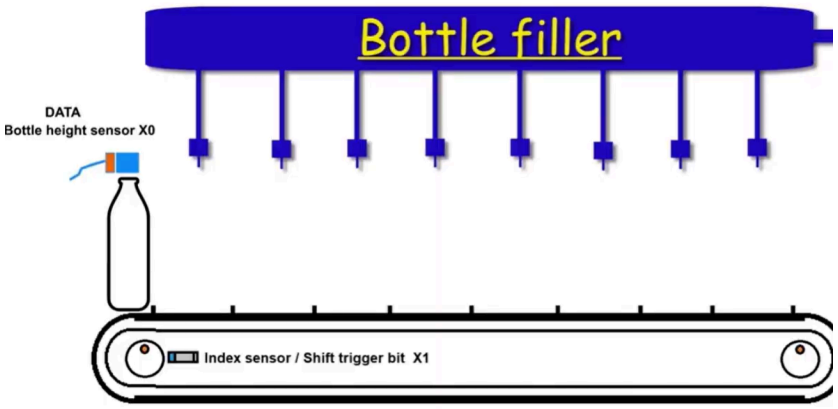
- si crea la variabile di tipo **byte** che conterrà il **dato** da trasferire allo Shift Register;
- per scrivere un dato nello Shift Register, è necessario porre a zero il pin collegato all' RCLK (ricordiamo che quando questo pin è alto i dati vengono immessi sulle uscite dello Shift Register), per il nostro pilotaggio il pin lo chiameremo **pin_latch** e sarà il pin **8** ;
- la scrittura del dato avviene con la funzione **shiftOut ()** inserendo nelle parentesi tonde i seguenti argomenti
  shiftOut (pin_dato, pin_clock, LSBFIRST, dato);
  per il **pin_dato** avremo il pin **11** e per il **pin_clock** il pin **12**;
- portando ora a stato logico alto il **pin latch** RCLK il dato verrà trasferito alle uscite dello Shift Register.

```
//Il pin8 viene usato per abilitare la scrittura sullo shift register
int pin_latch = 8;
//il pin11 viene usato per inviare il dato
int pin_dato = 11;
//Il pin12 viene usato per generare il clock
```

```
int pin_clock = 12;
//variabile che contiene il dato da scrivere sullo shift register
byte dato = 0;
void setup(){
  //Setto i pin come uscite
  pinMode(pin_latch, OUTPUT);
  pinMode(pin_dato, OUTPUT);
  pinMode(pin_clock, OUTPUT);
  //Per il serial monitor (debug)
  Serial.begin(9600);
}

void loop(){
  //setto tutte le 8 uscite dello shift register a 1
  dato=0b11111111;
  //se avessimo voluto avere a 0 il pin QA avremmo inviato
  //dato=0b01111111;
  //o se avessimo voluto avere a 0 il pin QH avremmo inviato
  //dato=0b11111110;
  //invio il valore al serial monitor
  Serial.println(dato,BIN);//visualizzazione codifica binaria
  //abilita la scrittura
  digitalWrite(pin_latch, LOW);
  //invia il dato
  shiftOut(pin_dato, pin_clock, LSBFIRST, dato);
  //Disabilita scrittura
  digitalWrite(pin_latch, HIGH);
  //ritardo
  delay(500);
}
```

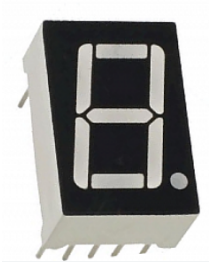
La modalità sopra descritta viene utilizzata solitamente per aumentare il numero di uscite disponibili sul microcontrollore ma, nel campo delle automazioni, lo Shift Register, associato ad appositi sensori, viene spesso usato per identificare elementi con caratteristiche fisiche diverse (non omogenei) per creare un registro da utilizzare nella gestione delle differenze rilevate.

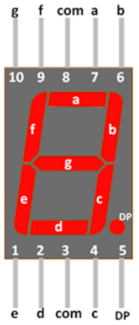


animazione

Display a Segmenti

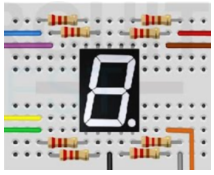
Altro utilizzo molto importante degli **Shift Register** è per il pilotaggio dei **Display a Segmenti.** Questo dispositivo è composto da **7 segmenti** denominati con le lettere che vanno dalla **'a'** alla **'g'**, da un pin per l'accensione del punto **'DP'** e da due pin **'com'** per la connessione al **Ground**.

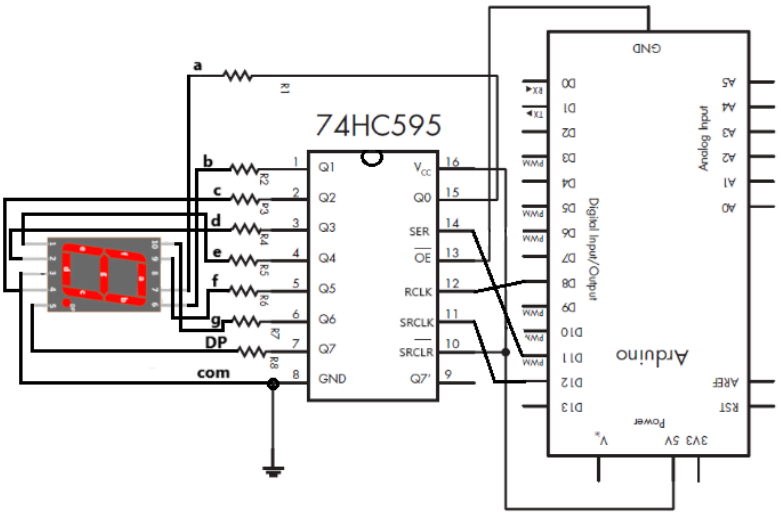




Lo Shift Register è ideale per il suo pilotaggio perché è possibile farlo **con soli 3 pin del Microcontrollore** anche se il display è a più cifre dato che gli Shift Register possono essere collegati in cascata mediante il pin QH'.

[animazione]





Connessioni per pilotaggio Display

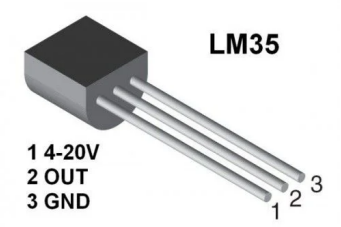
	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	
Segment	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**DP**	**DECIMAL**
0	1	1	1	1	1	1	0	0	**252**
1	0	1	1	0	0	0	0	0	**96**
2	1	1	0	1	1	0	1	0	**218**
3	1	1	1	1	0	0	1	0	**242**
4	0	1	1	0	0	1	1	0	**102**
5	1	0	1	1	0	1	1	0	**182**
6	1	0	1	1	1	1	1	0	**190**
7	1	1	1	0	0	0	0	0	**224**
8	1	1	1	1	1	1	1	0	**254**
9	1	1	1	1	0	1	1	0	**246**
A	1	1	1	0	1	1	1	0	**238**
B	0	0	1	1	1	1	1	0	**62**
C	1	0	0	1	1	1	0	0	**156**
D	0	1	1	1	1	0	1	0	**122**
E	1	0	0	1	1	1	1	0	**158**
F	1	0	0	0	1	1	1	0	**142**

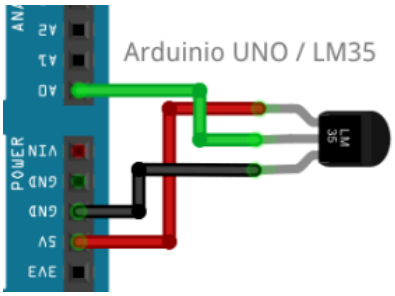
Nella tabella a fianco vi sono riportati i numeri decimali la cui codifica binaria fa ‘accendere’ i segmenti corretti per ottenere la cifra o la lettera voluta sul Display.
Inviando quindi nel Registro a Scorrimento tali numeri, questi vengono trasformati direttamente dal Microcontrollore in un treno di bit e portando stato logico alto il pin Latch se ne otterrà la visualizzazione sul display.
Nel comando shiftOut() bisognerà far caricare il treno di bit partendo dall’ LSB (Least Significant Bit).

Indice

28. Sensore di temperatura LM35

Il sensore **LM35** permette di acquisire temperature che vanno da **2°C a 150°C** e può essere alimentato con tensioni da **4÷20V.**
La **risoluzione** di questo sensore è di **10mV/°C** (in uscita avremo un incremento/decremento di tensione di **10mV** ad ogni incremento/decremento di **1°C** di temperatura). I collegamenti si riducono all'alimentazione mediante i pin 1 e 3 ed al collegamento del pin 2 ad un ingresso analogico di Arduino.





Considerando che i pin analogici di Arduino consentono di campionare segnali a 10 bit (valori da 0 a 1023) con una risoluzione di circa 5mV (5V/1023=4,8mV) e che quella del sensore è di 10 mV/°C, si ottiene che avremo un sistema di misura con una sensibilità di circa mezzo grado (5/10=0,5).
Vediamo come convertire il numero decimale, corrispondente al segnale analogico proveniente dal sensore, che a sua volta corrisponde al valore di tensione applicato al pin, nella temperatura misurata:
Sappiamo che ogni unità letta corrisponde a 4,8875mV, per cui, se moltiplichiamo il numero letto per tale valore otteniamo i millivolt applicati al pin. Ad esempio, se leggiamo il segnale pari a 43 e lo moltiplichiamo per 4,8875 otteniamo 43x4,8875= 210mV.
Dato che il sensore fornisce 10mV/°C dividendo i millivolt per 10 otteniamo i gradi 210/10= 21°C. Dato che il risultato della prima moltiplicazione va sempre diviso per 10 possiamo semplicemente moltiplicare il valore del segnale analogico direttamente per 0,48875 (4,8875/10).

```
int pin_sensore_lm35=A0;
int valore_pin_sensore_lm35=0;
float temperatura=0.0;

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

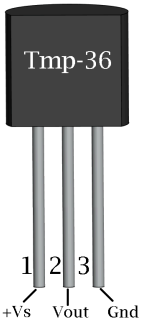
```
valore_pin_sensore_lm35=analogRead(valore_pin_sensore_lm35);
temperatura= valore_pin_sensore_lm35 * 0.48875

Serial.print("Temperatura= ");
Serial.println(temperatura);

}
```

Indice

29. Sensore di temperatura TMP36



Il **TMP36** permette di acquisire **temperature** comprese nell'**intervallo** tra **-40°C e +125°C** restituendo in uscita valori di tensione lineari tra circa **0.1Vdc e 1.7Vdc**. **Una variazione di grado** produce una variazione della tensione di uscita pari a **10mV**.
Alla temperatura di **0°C** il sensore fornisce una tensione di **500mV**.

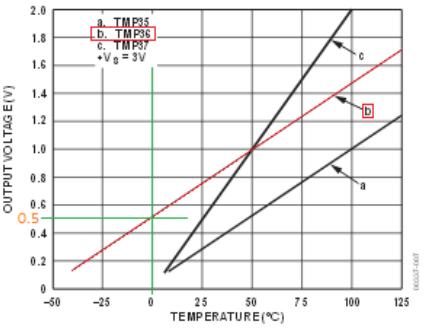


Figure6. Output Voltagevs. Temperature

la formula da inserire nello Sketch che converte il valore acquisito in gradi centigradi è la seguente
°C = ((valoreADC * PrecisioneADC) – TensioneZeroGradi) / stepGradoTensione

dove

°C = valore della temperatura in gradi centigradi

valoreADC = valore della conversione analogico digitale restituito da analogRead

PrecisioneADC = questo valore è ottenuto dividendo la tensione di riferimento dell'ADC (default 5Vdc) e il numero massimo restituito dalla conversione (1024). (5Vdc /1024 = 0.00488)

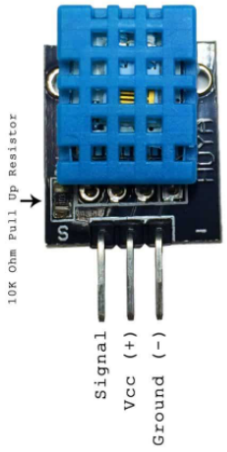
TensioneZeroGradi = indica la tensione di uscita dal sensore quando rileva una temperatura di 0°C

stepGradoTensione = indica la variazione di tensione per ogni variazione di grado (0.01 = 10 mV)

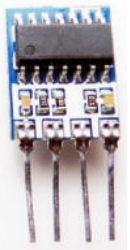
```
void loop()
{
  //leggo dalla porta A0
  val_Adc = analogRead(0);
  //converto il segnale acquisito in un valore
  //espresso in gradi centigradi
  temp = ((val_Adc * 0.00488) - 0.5) / 0.01;
  //invio il dato sulla seriale
  Serial.println(temp);
  //ritardo di mezzo secondo
  delay(500);
}
```

Indice

30. Sensore di Umidità e Temperatura DHT11 con uscita digitale



Il sensore **DHT11** è un sensore di temperatura e umidità con uscita dei dati in formato digitale.



Il sensore utilizza una tecnica digitale esclusiva e contiene al suo interno un processore 8-bit single-chip ed una camera di calibrazione che lo rendono particolarmente preciso.

Le sue piccole dimensioni ed il suo basso consumo, unite alla lunga distanza di trasmissione (20 m), permettono al sensore **DHT11** di essere adatto per molti tipi di applicazioni

Per poterlo utilizzare è necessario ricorrere ad una apposita libreria che consente anche l'uso di più sensori contemporaneamente.

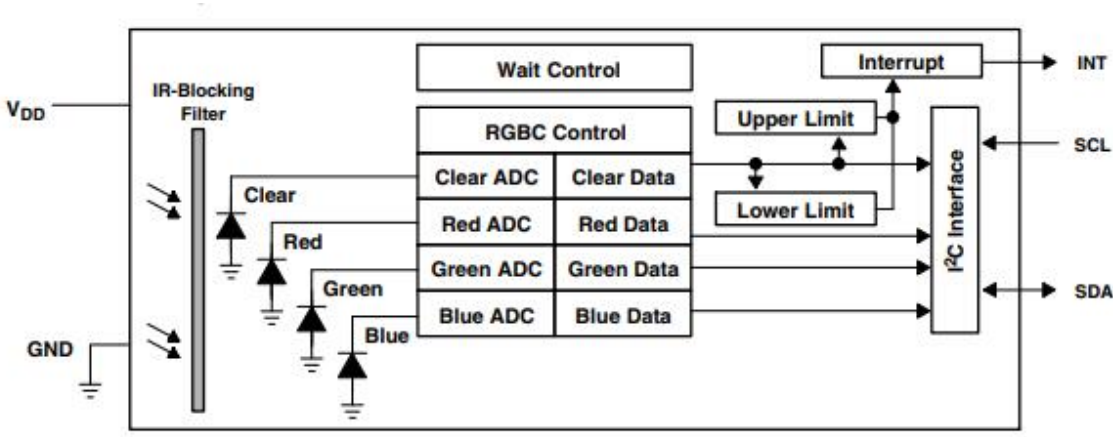
Campo di misura umidità 20-90%, temperatura 0-50°C, **Alimentazione** 3-5,5V DC

Indice

15. Sensore di colore TCS34725

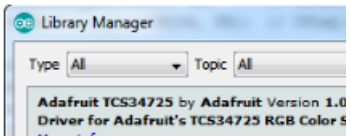
Il TCS34725 è un convertitore di colore in grado di rilevare i livelli di Rosso, Verde e Blu contenuti in un dato colore e li trasforma in un segnale digitale grazie a dei **Fotodiodi RGB**.
Ricordiamo che la differenza tra Led e Fotodiodo è che il primo, quando viene alimentato, produce energia luminosa mentre, il Fotodiodo, va in conduzione quando viene colpito da energia luminosa.
Il sensore di colore TCS34725 trova applicazione in gadget per la salute/fitness, nei processi industriali e apparecchiature diagnostiche mediche.

E' inoltre in grado di rilevare la luce ambientale (ALS) e per questo viene utilizzato per abilitare la luminosità automatica del display per una visualizzazione ottimale in prodotti come smartphone, notebook e TV.



Schema a blocchi sensore TCS34725

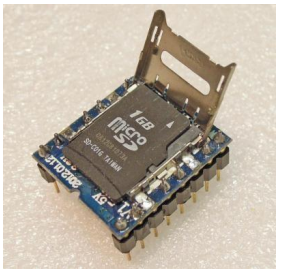
Il sensore sfrutta il Protocollo I2C e va alimentato a 3,3Vdc. Per sfruttare al meglio le sue potenzialità si ricorre alla libreria Adafruit TCS34725

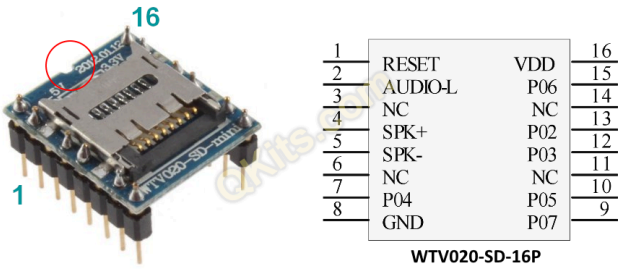


Indice

32. Modulo audio WTV020-SD

Questa scheda permette di riprodurre dei file audio precedentemente salvati su una scheda mini SD. La massima capacità della scheda utilizzabile è di 2 GB, la scheda deve essere formattata in formato FAT32, altri tipi di file system o capacità maggiori non sono riconosciuti dal modulo.





pinout

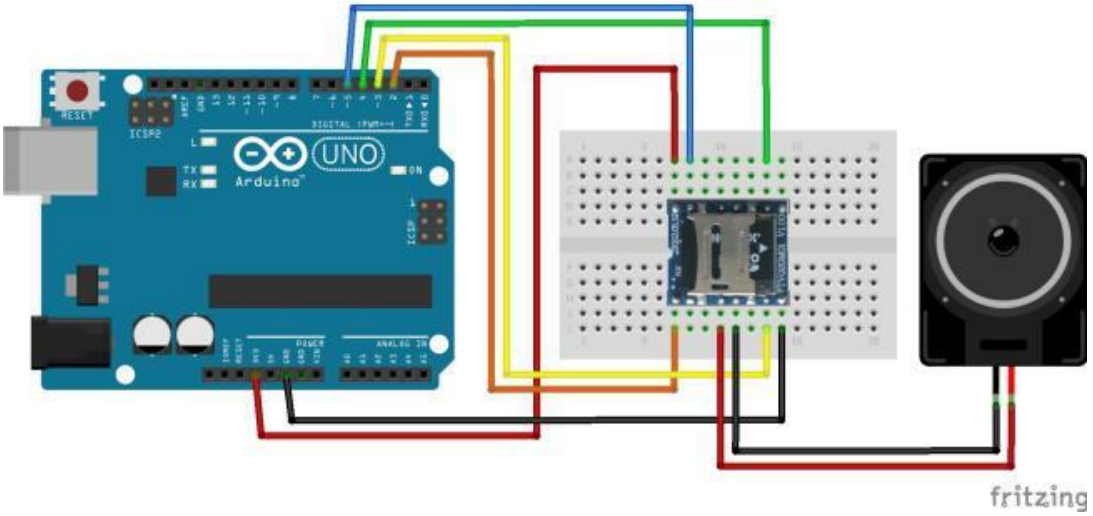
Creazione dei file audio

Per poter riprodurre dei brani è necessario realizzare dei file audio compatibili con il modulo e caricarli nella scheda microSD. Il formato che offre la migliore riproduzione è il formato *.ad4.

IMPORTANTE! I files, una volta generati, devono avere il nome nel formato XXXX.ad4, dove le X stanno ad indicare delle cifre in numerazione decimale, da 0000 a 0511. Per cui sulla microSD dovremo avere quindi qualcosa del genere:

0000.ad4 0001.ad4 0002.ad4, ...0511.ad4

Collegamento ad Arduino



Pin Arduino	Pin Wtv020	Funzione	Descrizione
2	1	Reset.	Pin reset
3	7	P04/CLOCK	CLOCK linea seriale
4	10	P05/Data	DATA linea seriale
5	15	P06 /Busy	BUSY
-	4	SPK+	Polo positivo altoparlante
	5	SPK-	Polo negativo altoparlante
GND	8	massa	Negativo alimentazione
3.3V	16	+3,3 Volt	Positivo alimentazione

Per utilizzare la libreria occorrerà utilizzare il comando #include

```
#include <Wtv020sd16p.h>
```

poi occorre assegnare il numero dei pin utilizzati per le varie funzioni

```
int resetPin = 2;  // Reset pin 2.
int clockPin = 3;  // Clock pin 3.
int dataPin = 4;  // Data pin 4.
int busyPin = 5;  // Busy pin 5.
```

A questo punto si potrà creare l'istanza

```
Wtv020sd16p wtv020sd16p(resetPin, clockPin, dataPin, busyPin);
```

Sono disponibili i seguenti comandi:

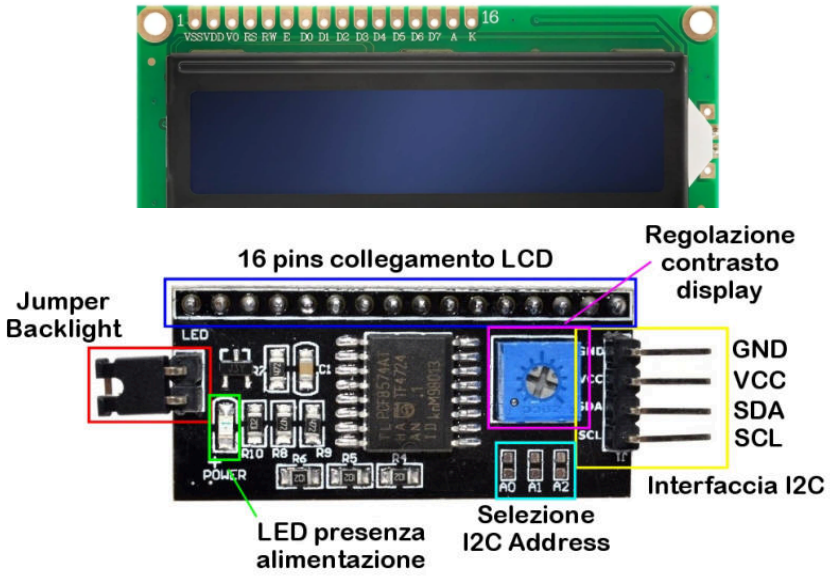
- wtv020sd16p.reset(); Esegue il reset del modulo
- wtv020sd16p.setVolume(X); Imposta il volume dell'uscita collegata all'altoparlante, dove X è un valore tra 0 e 7.
- wtv020sd16p.asyncPlayVoice(X); Manda in esecuzione i file relativo, dove X è un valore tra 0 e 511.
- wtv020sd16p.pauseVoice(); Arresta la riproduzione del brano, o riprende il brano dopo una pausa.
- wtv020sd16p.mute(); ammutolisce il suono durante la riproduzione.
- wtv020sd16p.unmute(); riattiva il suono durante la riproduzione
- wtv020sd16p.stopVoice(); Interrompe la riproduzione del file corrente..

es. per far riprodurre il file che abbiamo denominato 0014.ad4 che ha una durata di 4 secondi il comando sarà il seguente

```
wtv020sd16p.asyncPlayVoice(14);
delay(4000);
```

Indice

16. Display LCD



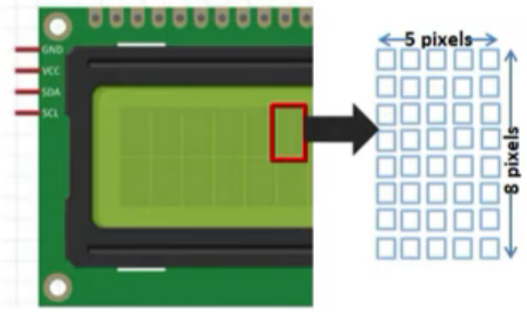
Il display utilizza il protocollo [I²C](#) (I quadro C), grazie al quale, può essere gestito semplicemente con due conduttori **SDA**(**S**erial **DA**ta) **SCL**(**S**erial **CL**ock) oltre ai conduttori di alimentazione **VCC** e **GND.**
Per il suo funzionamento è necessario includere nello Sketch le relative librerie, definire l'indirizzo I2C e la tipologia di display LCD e procedere alle impostazioni di inizializzazione nel void setup()

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);//assegna l'indirizzo 0x27
                                 //display 16 colonne 2 righe
void setup() {
lcd.init();//inizializza Display
lcd.backlight();//accende luce posteriore
lcd.clear();//pulisce display
lcd.begin(16,2);//imposta display 16 colonne 2 righe
}
void loop() {
lcd.setCursor(4,0);//Cursore posizionato su quarta colonna riga 0
lcd.print("ITI Nicotera");//stampa questo testo
lcd.setCursor(2,1);//Cursore posizionato su seconda colonna riga 1
lcd.print(" ELETTROTECNICA ");//stampa questo testo
}
```

Per stampare codici ASCII bisognerà utilizzare la funzione lcd.write invece di lcd.print lcd.write(numerocodiceascii);

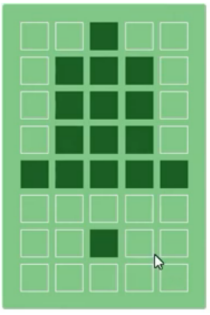
Come creare nuovi caratteri

Ogni '**Cella**' è composta da una **matrice** (tabella ordinata di valori individuabili per colonna e riga) composta da **5 colonne e 8 righe** che consentono di gestire l'accensione o lo spegnimento di ogni singolo pixel della cella.



Per creare dei nuovi caratteri è necessario creare degli Array di tipo Byte e definire quali pixel accendere o mantenere spenti per comporre il nuovo carattere.

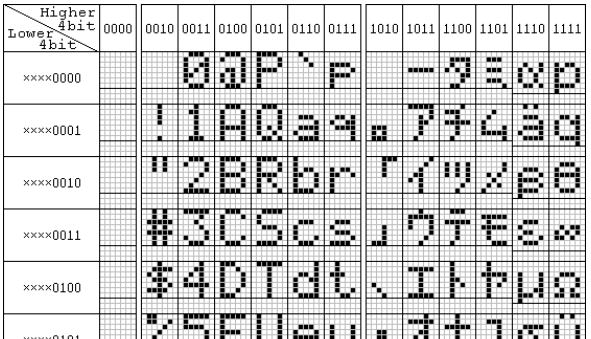
```
byte campana[8] = {
  B00100,
  B01110,
  B01110,
  B01110,
  B11111,
  B00000,
  B00100,
  B00000
};
```



nel void setup() viene creato il carattere con **lcd.createChar(0, campana);** dove il primo parametro è l'**indice** (0-7 dato che si possono creare fino a 8 caratteri), mentre, il secondo, è il nome dell'Array di byte creato in precedenza.
Quando si vorrà stampare il carattere si posiziona il cursore e si stampa il carattere richiamando il numero dell'**indice** assegnato

```
 lcd.setCursor(0, 0);
lcd.write(byte(0));
```

oltre alla possibilità di creare nuovi caratteri esiste anche un set di caratteri



che possono essere richiamati con il loro numero binario, o direttamente decimale, secondo questa tabella.
Se ad esempio, vogliamo stampare il segno della percentuale, dovremo far riferimento al numero binario composto da 0010 (Higher 4 bit) e 0101 (Lower 4 bit) che corrisponde al numero decimale

00100101 => 37

e nel codice di Arduino

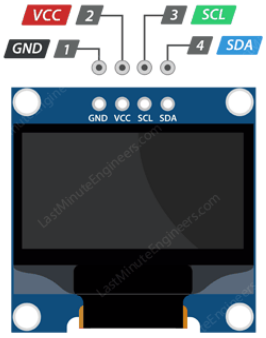
```
lcd.setCursor(0, 0);
lcd.write((char) 37);
oppure
lcd.setCursor(2, 0);
lcd.write((char) B00100101);
```

Da notare che non si effettua un 'print' ma un 'write' perché con write viene stampato il byte e non la stringa.

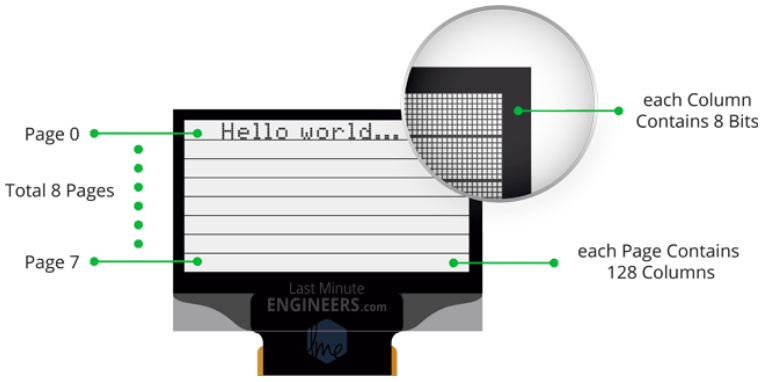
Indice

17. Display OLED

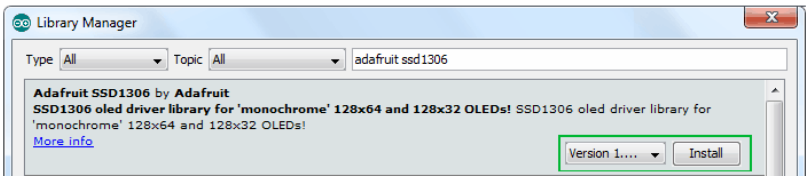
Il chip di controllo di questo display è l'**SSD1306** e per il suo funzionamento si ricorre al protocollo **I²C** (I quadro C), grazie al quale, può essere gestito semplicemente con due conduttori **SDA**(**S**erial **DA**ta) **SCL**(**S**erial **CL**ock) oltre ai conduttori di alimentazione **VCC** e **GND.**

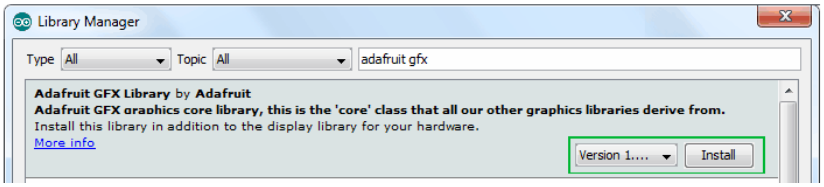


Rispetto al Display LCD offre la possibilità di accendere ogni singolo pixel disponibile su di esso. Il modello disponibile nel nostro laboratorio è 128x64 ed è composto da 8 righe (pages 0÷7), ognuna delle quali contiene 8 pixel (bit), e da 128 colonne.

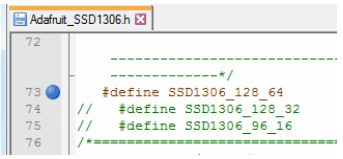


per il suo funzionamento, oltre alla libreria Wire.h (necessaria per il protocollo I2C), sono necessarie le seguenti librerie





nel caso si riceva questo messaggio sarà necessario aprire Adafruit_SSD1306.h ed al rigo 73 effettuare la seguente modifica per abilitare il modello 128x64



le righe di comando iniziali prevedono le seguenti impostazioni

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display larghezza, in pixels
#define SCREEN_HEIGHT 64 // OLED display altezza, in pixels

// dichiara che l' SSD1306 è connesso mediante I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

per il void setup() avremo

```
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {      // indirizzo I2C
    Serial.println(F("SSD1306 indirizzamento fallito"));//eventualmente da verificare
    for(;;);                                           //mediante I2c scanner
  }
  delay(2000);
```

di seguito esempi di visualizzazioni da inserire nel void loop()

Testo semplice


```
// Display Text
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
```

Per posizionare il cursore nel punto desiderato display.setCursor(colonna, rigo);

Testo semplice con display.setTextSize (2)


```
// Changing Font Size
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(0,24);
display.setTextSize(2);
display.println("Hello!");
display.display();
delay(2000);
```

Visualizzazione simbolo ASCII


```
// Display ASCII Characters
display.clearDisplay();
display.setCursor(0,24);
display.setTextSize(2);
display.write(3);
display.display();
delay(2000);
```

in questo caso invece del display.println(“testo”) viene usato display.write(cifra codice ASCII)

Scrolling dell’intero testo


```
// Scroll full screen
display.clearDisplay();
display.setCursor(0,0);
display.setTextSize(1);
display.println("Full");
display.println("screen");
display.println("scrolling!");
display.display();
display.startscrollright(0x00, 0x07);
delay(2000);
display.stopscroll();
delay(1000);
display.startscrollleft(0x00, 0x07);
delay(2000);
display.stopscroll();
delay(1000);
display.startscrolldiagright(0x00, 0x07);
delay(2000);
display.startscrolldiagleft(0x00, 0x07);
delay(2000);
display.stopscroll();
```

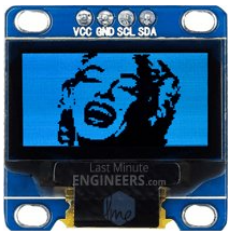
Scrolling di un rigo specifico


```
// Scroll part of the screen
display.setCursor(0,0);
display.setTextSize(1);
display.println("Scroll");
display.println("some part");
display.println("of the screen.");
display.display();
display.startscrollright(0x00, 0x00);
```

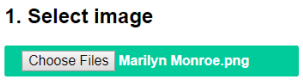
Per lo scrolling di una porzione di testo nella funzione displaystartscrollright(0x00,0x00)

va inserita la pagina che dovrà muoversi. Nell’esempio si muove la pagina numero 0

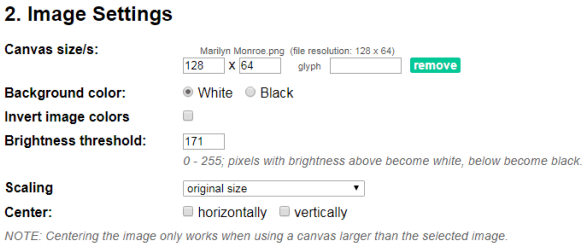
Visualizzazione immagine Bitmap



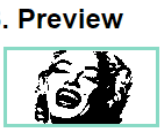
Per poter visualizzare un'immagine è necessario trasformare l’immagine nel codice che accenderà o spegnerà i pixel necessari alla sua rappresentazione sul display. Per farlo ci viene in aiuto l’applicazione online [image2cpp](). Per poter procedere è necessario impostare la risoluzione dell’immagine a quella del display (128x64) con un qualunque editor di immagini.

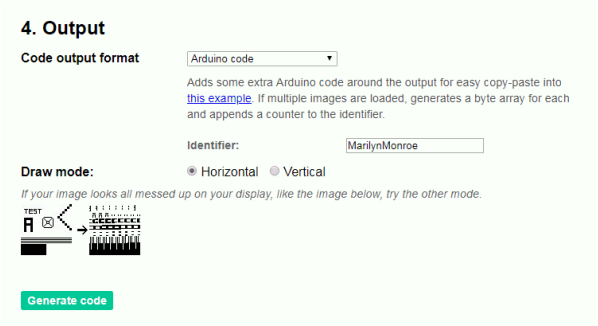


nella pagina dell’applicazione si sceglie il file



nella sezione ‘Image Setting’ si sceglie il livello sotto il quale spegnere i pixel e sopra il quale accenderli ‘Brightness threshold', gli effetti sono visibili nella Preview





a questo punto diamo un nome all’**identifier** e siamo pronti a generare il codice da inserire nel nostro sketch

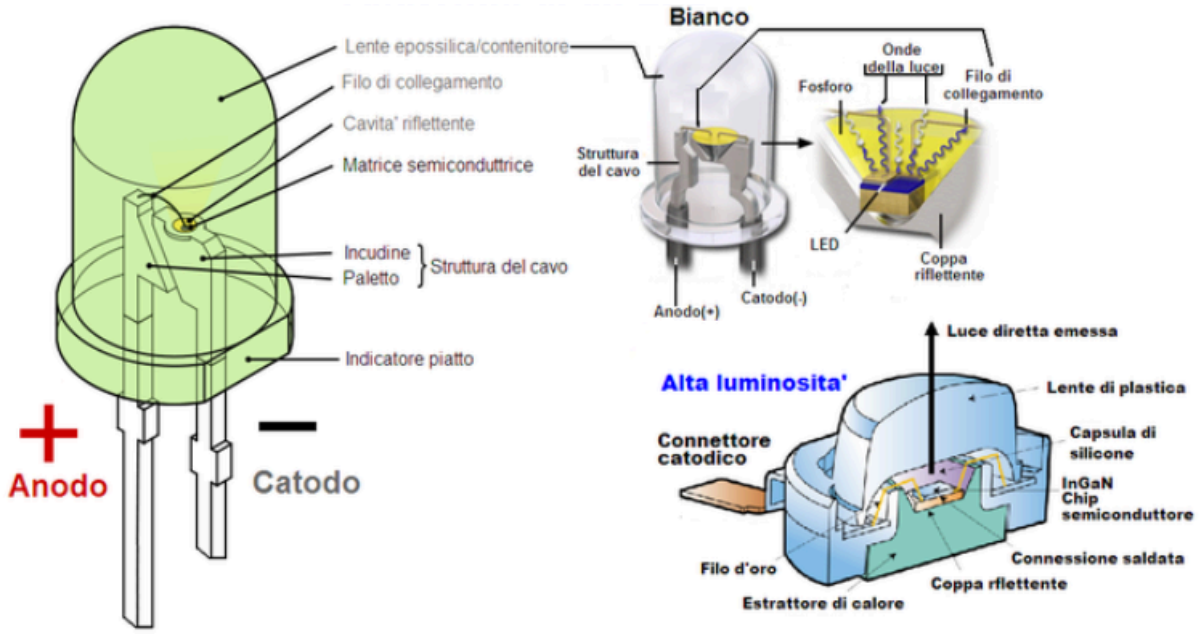
```
// 'Marilyn Monroe', 128x64px
const unsigned char MarilynMonroe [] PROGMEM = {
        0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x1f, 0xff, 0xff, 0xd0, 0x41, 0xff, 0xff, 0xff,
0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x7f, 0xff, 0xff, 0xf8, 0x03, 0xff, 0xff, 0xff,
0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x79, 0xff, 0xff, 0xff, 0xe0, 0x07, 0xff, 0xff, 0xff,
0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x87, 0xff, 0xff, 0xff, 0xf8, 0x03, 0xf7, 0xff, 0xff,
0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x07, 0xff, 0xff, 0xff, 0xf8, 0x01, 0xf1, 0xff, 0xff,
```

a questo link uno scketc di esempio che riproduce il logo di arduino

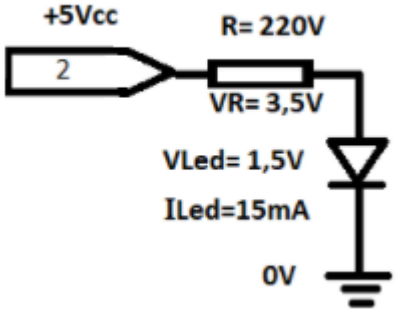
Indice

13. LED (Light Emitting Diode)

Il Led è un componente elettronico costituiti da una giunzione P-N, con arseniuro di gallio o con fosfuro di gallio, che emette luce quando attraversati da una corrente compresa tra 10 mA e i 30 mA a seconda del tipo di LED utilizzato.



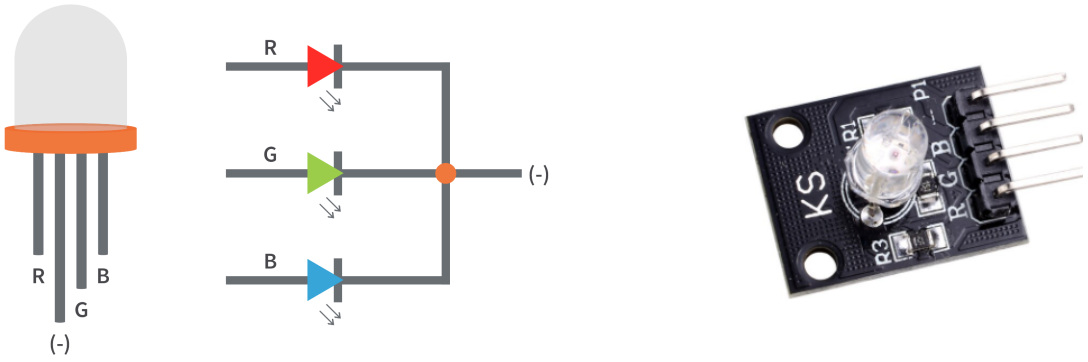
I led hanno un terminale positivo detto Anodo (semiconduttore tipo P) ed uno negativo detto Catodo (semiconduttore tipo N) e, per funzionare, è necessario che sia polarizzato direttamente. Il terminale positivo è il terminale più lungo ma, osservando l'interno del led in controluce, lo si può individuare con certezza: come si vede in figura, l'elettrodo positivo è sottile, a forma di lancia, mentre il negativo ha l'aspetto di un`ascia.



Quando si utilizza un led è necessario disporre sempre una resistenza in serie ad esso allo scopo di limitare la corrente che passa ed evitare che possa distruggersi.

Indice

14. LED RGB



I l**ed RGB** sono dei led costituiti da **tre led di diversi colori** (Red, Green e Blue) **contenuti nello stesso involucro**. Possono essere sia ad anodo comune che a catodo comune. Quelli disponibili nel nostro laboratorio sono del tipo a **catodo comune**.
La particolarità di questo tipo di led è che combinando l'intensità dei tre colori base Rosso (Red), Verde (Green) e Blu (Blue) si possono realizzare tutti gli altri colori.
Per poter '**modulare**' l'intensità dei colori si ricorre al **pilotaggio PWM**
Per gestire i colori è conveniente creare una apposita 'struttura eseguibile' che potremo richiamare durante il programma.
Una volta dichiarati nel void setup() i pin di Arduino associati ai pin del led RGB

```
int pin_led_rosso=6;//pin PWM
int pin_led_verde=3;//pin PWM
int pin_led_blu=9;//pin PWM
```

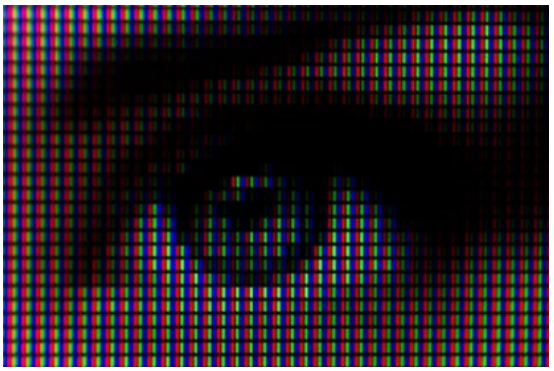
creeremo la seguente struttura

```
void colore_RGB(int valore_red,int valore_green,int valore_blu){
  analogWrite(pin_led_rosso, valore_red);
  analogWrite(pin_led_verde, valore_green);
  analogWrite(pin_led_blu, valore_blu);
}
```

quando si vorrà impostare un colore nello scketch inseriremo il comando

```
colore_RGB(255,0,0); //verde
```

mediante il quale gestiremo l’intensità del Red, Green, Blue mediante l’opportuno numero decimale che corrisponde ad un valore di tensione PWM (risoluzione ad 8 bit).
Questa tecnologia è, ad esempio, presente sui monitor sui quali, grazie a dei micro led rgb, è possibile riprodurre le immagini a colori



Indice

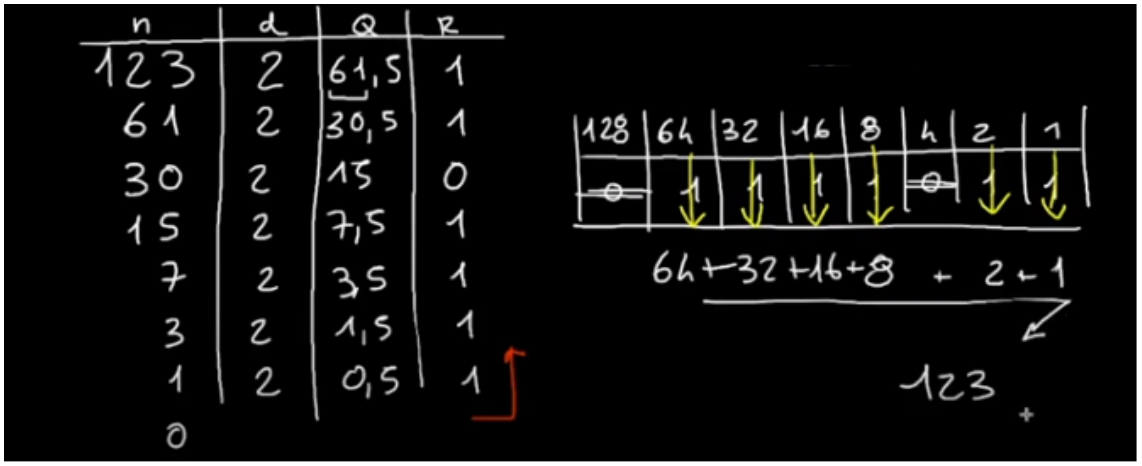
36. Sistemi di Comunicazione tra dispositivi elettronici digitali

Perché si usa la codifica binaria

- Se un apparato è '**elettrico**', per 'esprimersi' (comunicare), potrà usare esclusivamente segnali elettrici (tensione o corrente).
- Un segnale si dice '**digitale**' quando può assumere solo **due** valori logici definiti (on-off, alto-basso, presenza-assenza)

I motivi che hanno spinto i ricercatori a **scegliere la codifica binaria per gestire le 'informazioni' tra apparati elettronici digitali** è dipeso dal fatto che, **usando solo due cifre** lo **0** e l'**1** per rappresentare tutti i numeri, **si prestava a 'tradurre' tali numeri in segnali elettrici** (presenza o assenza di tensione)**.**

Di seguito le modalità per convertire da decimale in binario e viceversa



nella conversione da base 10 a binario, nel comporre il numero binario, si parte dall'ultimo resto verso il primo, in questo caso sarà **1111011.**
Nota: il bit più a sinistra si dice MSB (**M**ost **S**ignificant **B**it: Bit più significativo) mentre **quello più a destra LSB** (**L**east **S**ignificant **B**it: Bit

meno significativo). **La trasmissione seriale invia per primo l'LSB e risale fino all' MSB.**

Perché si usa anche la Codifica Esadecimale

Il sistema esadecimale è un sistema numerico posizionale con base 16. Per la composizione di un numero il sistema esadecimale utilizza le cifre **0, 1, 2, 3, 4, 5, 6, 7. 8, 9, A, B, C, D, E, F**. Si distingue, pertanto, dal sistema decimale per la presenza di sei cifre aggiuntive identificate dalle lettere maiuscole **A**(10), **B**(11), **C**(12), **D**(13), **E**(14), **F**(15). Il sistema esadecimale è conosciuto anche con l'abbreviazione **esa** o **hex**.
esempio:
Per calcolare il numero decimale equivalente del numero esadecimale **4F** riscriviamo l'operazione di calcolo soltanto con valori decimali.

$$4F_{(16)} = 4 \times 16^{1} + 15 \times 16^{0} = 64 + 15 = 79_{(10)}$$

Il motivo per il quale viene usato in informatica dipende dal fatto che consente di rappresentare i numeri con l'utilizzo di un numero inferiore di bit, occupando così un numero inferiore di celle di memoria nel dispositivo

Valore ESADECIMALE: (cifre comprese tra 0 e F) 4F

Valore DECIMALE: (cifre comprese tra 0 e 9) 79

Valore OTTALE: (cifre comprese tra 0 e 7) 117

Valore BINARIO: (usa solo 0 ed 1) 1001111

Decimale	Esadecimale	Binario
0	0	0 0 0 0
1	1	0 0 0 1
2	2	0 0 1 0
3	3	0 0 1 1
4	4	0 1 0 0
5	5	0 1 0 1
6	6	0 1 1 0
7	7	0 1 1 1
8	8	1 0 0 0
9	9	1 0 0 1
10	A	1 0 1 0
11	B	1 0 1 1
12	C	1 1 0 0
13	D	1 1 0 1
14	E	1 1 1 0
15	F	1 1 1 1

Principio di funzionamento

Se abbiamo l'apparato digitale **A** che deve comunicare, ad esempio, un numero all'apparato digitale **B**, supponiamo la cifra 75 (i numeri che siamo abituati ad usare si dicono 'in base 10' perchè sono 10 le cifre usate per rappresentarli tutti, dallo 0 al 9), basterà trasformare questo numero nel numero binario corrispondente e cioè **010010**.
A questo punto l'apparato **A** potrà inviare all'apparato **B** una sequenza di segnali elettrici dove alla cifra binaria **1** si farà corrispondere la presenza di una tensione di **5V** (Logica Positiva) ed alla cifra binaria **0** l'assenza di tensione **0V**. L'apparato **B,** ricevuta la sequenza di segnali elettrici, li convertirà nuovamente in un codice binario che potrà essere 'manipolato' dai suoi circuiti elettronici digitali.

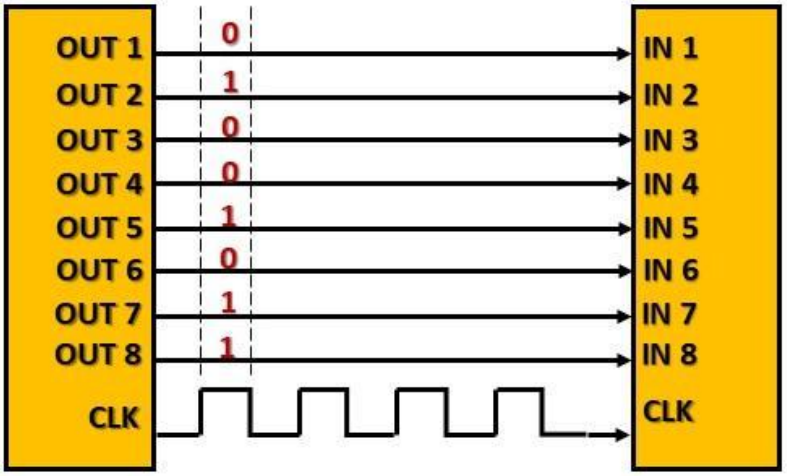
Segnale di Clock

Ora è il caso di soffermarci su una cosa importante: se dobbiamo inviare una sequenza di segnali elettrici lungo dei conduttori come facciamo a 'distanziarli' uno dall'altro in modo che la sequenza rispetti quella voluta? La soluzione è stata quella di aggiungere al circuito di trasmissione un conduttore sul quale viene applicato il cosiddetto ' **Segnale di clock**' che **ha il compito di dare i tempi di trasmissione dei segnali**.

Modalità di trasmissione

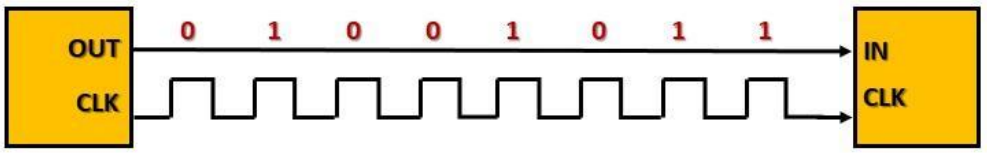
Comunicazione Parallela

La comunicazione parallela si basa su interfacce che sono in grado di trasferire più bit nello stesso momento, utilizzando quello che viene chiamato un **bus di dati**. Questi bus di dati vengono generalmente trasferiti attraverso fasci di 8, 16, 32, … cavi e si parla quindi di bus a 8, 16, 32 bit. Oltre a questi cavi ve n'è uno aggiuntivo chiamato CLK (clock) il cui scopo é quello di scandire il momento in cui i bit devono essere letti.



Comunicazione Seriale

La comunicazione Seriale invece si basa su un flusso di bit trasferito su di un unico cavo, in cui i bit sono trasferiti uno alla volta. Anche in questo caso esiste un secondo canale di comunicazione chiamato CLK (clock) che scandisce il tempo di lettura di ogni singolo bit. Quindi la comunicazione seriale si basa solo e sempre su **2 soli cavi**.



Considerazioni

La comunicazione in **parallelo** è **più efficiente e più veloce**, ma richiede un filo per ogni bit risultando così **più complessa ed onerosa** (costosa).

Di contro, quella **seriale è meno complessa** ma anche **meno veloce**.

La scelta di un tipo o dell'altro dipende quindi dal tipo di applicazione.

Velocità di trasmissione
Il **baud rate** indica la quantità di **bit che vengono trasmessi al secondo**. Il valore più comune di baud rate utilizzato è 9600 bps (bit per secondo), ma sono presenti anche moltissimi altri valori standard (2400, 4800, 19200, 57600, ecc..). **All'aumentare della velocità aumentano ovviamente i rischi di errori di comunicazione.**

Codice ASCII (**A**merican **S**tandard **C**ode **I**nformation **I**nterchange)
La tecnologia di invio diretto dei bit si dice **RTU** (**R**emote **T**erminal **U**nit) mentre, in alternativa, esiste la tecnologia del codice ASCII**.** Con quest'ultima, invece di inviare i bit, si inviano i numeri ed i simboli che sono stati decodificati in una apposita tabella, utili alla comunicazione umana.
Il codice ASCII, per ogni carattere o simbolo fa corrispondere un determinato numero. Si tratta di una codifica a **7 bit** e ciò significa che sono disponibili tutti i numeri **compresi tra 0 e 128** per codificare i caratteri (il numero binario 1111111 corrisponde infatti al 128 decimale). Per fare in modo che potessero essere rappresentati i caratteri ed i simboli di tutte le lingue del mondo, è stato introdotta anche la codifica Unicode che è invece ad **8 Byte** (ogni Byte è composto da 8 bit quindi abbiamo **32 bit)** che consentono di avere **33.554.431** numeri da convertire in caratteri.

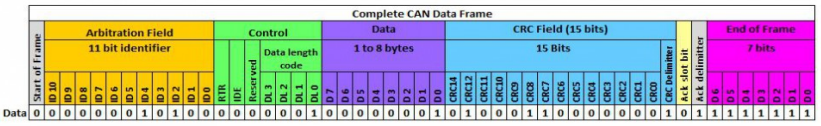
Tipologie di trasmissione tra apparati
Simplex
In un canale di trasmissione Simplex i dati viaggiano al suo interno solo e sempre in un'unica direzione.

Half-duplex
In un canale di trasmissione Half-duplex invece i dati possono passare in entrambi i sensi ma solo alternativamente. Ciò significa che mentre il trasmettitore (Tx) manda dei dati, il ricevitore (Rx) sta ad ascoltare. Un canale half-duplex è bidirezionale in quanto Tx e Rx possono scambiarsi i ruoli ma ricordando comunque che **non lo possono fare contemporaneamente**.

Full-duplex
Il Full-duplex invece è in grado di **trasmettere e ricevere dati in entrambi i sensi e simultaneamente.**

Indice

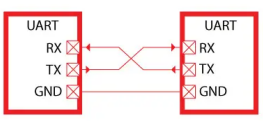
37. Protocolli di Comunicazione

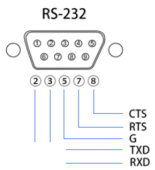


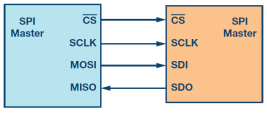
Un **Protocollo di Comunicazione**, **è un insieme di regole che definiscono le modalità di 'confezionamento' (*Frame*)** dei dati da trasmettere. Affinché la comunicazione avvenga, sono necessari i **circuiti fisici** sui quali far viaggiare i segnali, **le parti terminali** di questi circuiti vengono dette **Porte** o **Interfacce** e le linee sulle quali **viaggiano i dati Bus**.

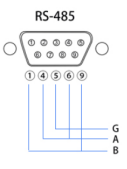
Parametri	UART	RS-232	I2C	SPI	RS-485	CAN-Bus
Topologia	punto punto	punto punto	Multi Master Multi Slave	1 Master Multi Slave	1 Master Multi Slave	Multi Master Multi Slave
Modalità	half-duplex	full-duplex	half-duplex	full-duplex	half-duplex full-duplex 4 fili (RS422)	half-duplex
Segnale	Single-ended	Single-ended	Single-ended	Single-ended	Differenziale	Differenziale
Numero Dispositivi	2	2	128	128	32	128
Distanza	5 mt	15 mt	10 mt	10 mt	1200 mt	1600 mt

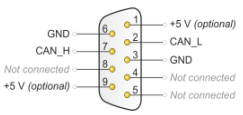
In questa tabella alcune tipologie di Protocolli ed a seguire le rispettive interfacce

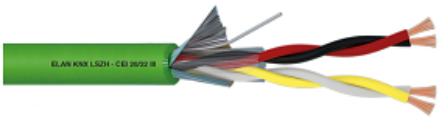








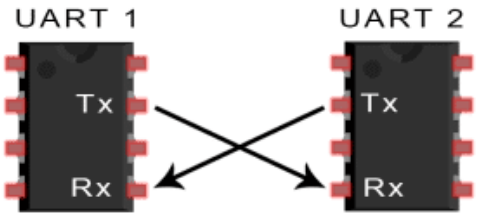




I cavi usati per la trasmissione dati hanno la caratteristica di essere **twistati** (attorcigliati). Il motivo è che, soprattutto nella trasmissione differenziale, **nel caso di interferenze queste creeranno lo stesso disturbo su entrambi i canali** mantenendo così la 'differenza' tra i segnali inalterata. Nei Sistemi di comunicazione che prevedono più coppie di cavi, questi, saranno twistati con 'passi' (distanza delle torsioni) diversi in modo che l'interferenza non si 'accordi' su tutte le coppie allo stesso modo, limitando così l'effetto del disturbo.

38. Porta UART (**U**niversal **A**synchronous **R**eceiver/**T**ransmitter)

La connessione tra due dispositivi che usano questa tecnologia è la seguente



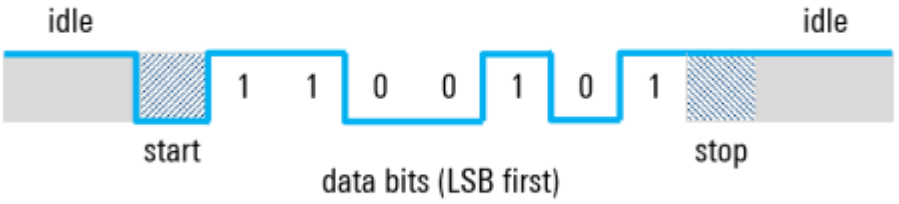
Si noti che il pin TX (Trasmettitore) di un dispositivo deve essere collegato al pin RX (Ricevitore) dell'altro e viceversa.

Si dice comunicazione asincrona quando il trasmettitore e il ricevitore non condividono un segnale di clock comune. Poiché non condividono un clock, entrambi di dispositivi devono trasmettere alla stessa velocità prestabilita (stesso Baud Rate) in modo che la comunicazione sia 'comprensibile'.

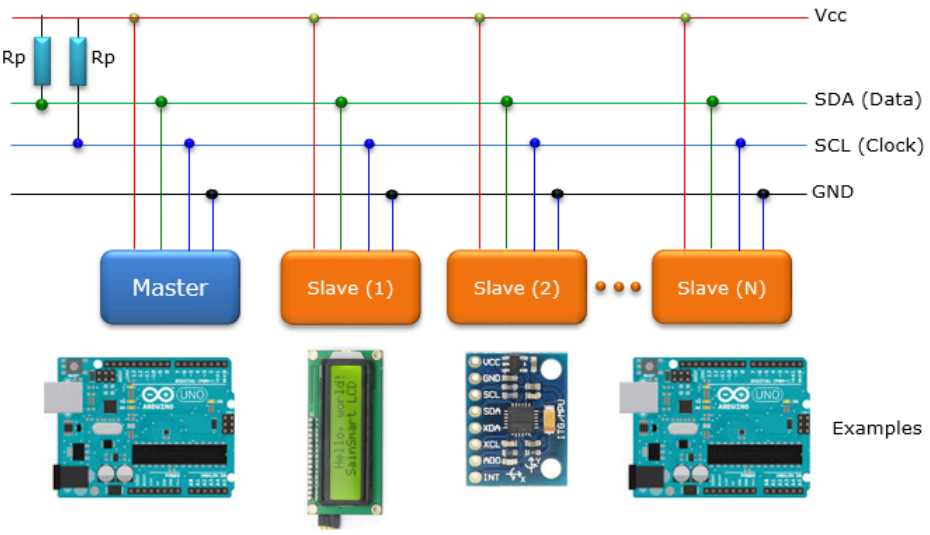
Poiché l'interfaccia UART è asincrona, il trasmettitore deve segnalare che i bit di dati stanno arrivando. Ciò viene realizzato utilizzando il cosiddetto **start bit**. **Il *bit di start* è una transizione dallo stato alto inattivo (Idle) a uno stato basso, immediatamente seguito da bit di dati utente.**
Dopo che i bit di dati sono finiti, lo **stop bit** indica la fine dei dati utente.

Il *bit di stop* è o una transizione di ritorno allo stato alto per un ulteriore tempo di bit.

Esempio:
Se vogliamo inviare la lettera maiuscola “S” in ASCII a 7 bit, la sequenza di bit è 1 0 1 0 0 1 1. Invertiamo prima l’ordine dei bit per metterli nell’ordine del bit meno significativo, cioè 1 1 0 0 1 0 1, prima di inviarli. Dopo l’invio dell’ultimo bit di dati, il bit di stop è utilizzato per terminare il frame e la linea ritorna allo stato di inattività.
’S’ ASCII a 7 bit (0x52) = 1 0 1 0 0 1 1
Ordine LSB = 1 1 0 0 1 0 1



39. Protocollo I²C (si legge I quadro C)
Il bus di comunicazione **I²C** è molto diffuso nel campo della **Domotica** perché può essere facilmente implementato in sistemi che richiedono la comunicazione tra **uno o più master** (padrone) e più dispositivi **slave** (schiavo). Le facili implementazioni derivano dal fatto che sono necessari **solo due fili** per la comunicazione, **SDA** (Serial Data) sul quale viaggiano i dati e **SCL**(Serial Clock) che **sincronizza** la trasmissione. Il protocollo può collegare fino a ben **128** dispositivi. Questo tipo di protocollo consente comunicazioni di tipo **half-duplex**, le **distanze raggiungibili** senza disturbi sono dell’ordine di una **decina di metri**.



Il protocollo è in grado di gestire un numero elevato di dispositivi perchè **ognuno di essi è contraddistinto da uno specifico indirizzo**.

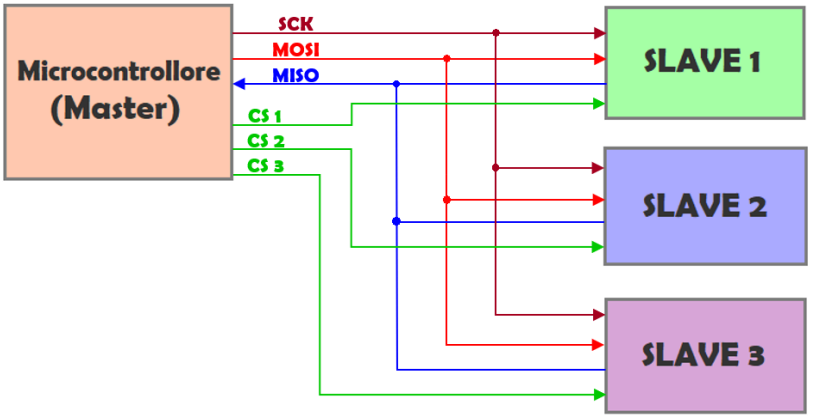
Nel caso non si sia a conoscenza dell'indirizzo di un dispositivo, basta collegarlo ad Arduino e lanciare il seguente sketch I2C_scanner e leggere sul monitor seriale quale sia.

Video protocollo I2C

Indice

40. Protocollo SPI (Serial Peripheral Interface)

Il protocollo **SPI**, in italiano Interfaccia Seriale di Periferica, è uno standard di comunicazione seriale **full-duplex**, sincrono e di tipo Master/Slave. Usa **4 conduttori** per la comunicazione e viene impiegato per lo **scambio dati a breve distanza**.



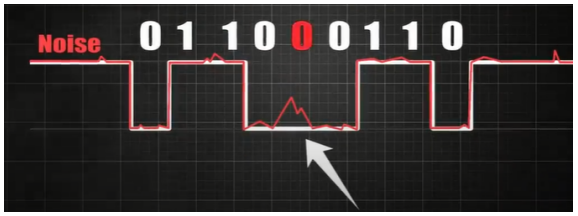
- **SCK (Serial Clock):** è il canale dedicato al **clock** che si occupa di scandire tutte le fasi dello scambio dati
- **MOSI (Master Output Slave Input):** è la linea di ingresso dei dati nello ***Slave*** e di uscita per il ***Master***
- **MISO (Master Input Slave Output):** è la linea di ingresso dei dati per il ***Master*** e di uscita per lo ***Slave***
- **CS (Chip Select) o SS (Slave Select):** se 0 (livello basso) abilita la periferica allo scambio dei dati.

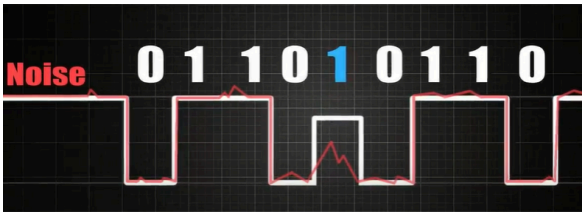
Questo tipo di protocollo prevede **un solo Master**

Indice

41. Protocollo RS-485

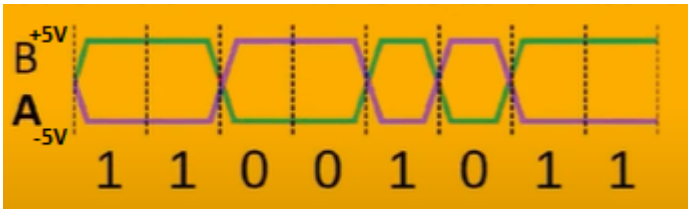
Dato che l'uso di sistemi **Bus Single-ended** (con una solo canale di trasmissione) in ambienti soggetti ad interferenze elettromagnetiche, come industrie e mezzi di trasporto, **può portare a degli errori di trasmissione**, ha spinto i ricercatori a trovare possibili soluzioni.



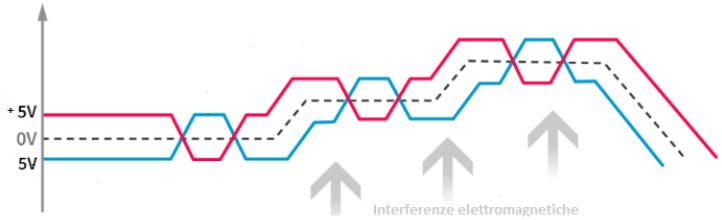


Un sistema per ovviare a questo problema è la Trasmissione Differenziale del **protocollo RS-485**. Questa prevede l'uso di due canali, **A** e **B,** che possono assumere entrambi valori **+5V** e **-5V**.
Gli stati logici dei segnali vengono così interpretati:

- se il canale **B > A** viene riconosciuto lo stato logico **1**;
- se il canale **A > B** viene riconosciuto lo stato logico **0**;

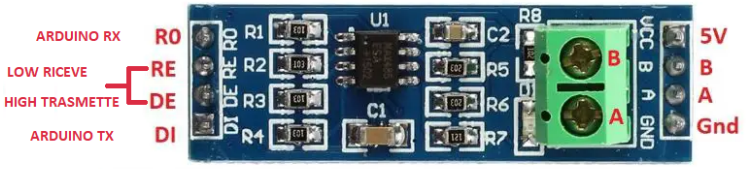


ora, **dato che i conduttori dei due canali sono twistati** (attorcigliati tra di loro), **subiranno entrambi lo stesso livello di disturbo** e ciò farà sì che i valori aumenteranno o diminuiranno della stessa quantità su entrambi i canali **mantenendo così gli stessi livelli differenziali**.

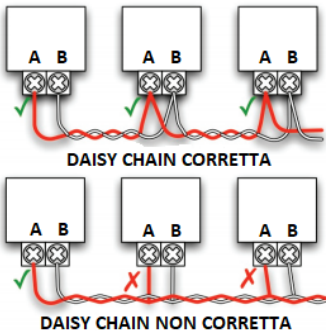


Questo protocollo può comunicare sia in modalità half-duplex che full-duplex ma, in quest'ultimo caso, i canali diventeranno 4. Sul Bus possono essere connessi fino a 32 dispositivi e la distanza coperta è di circa 1200 mt.

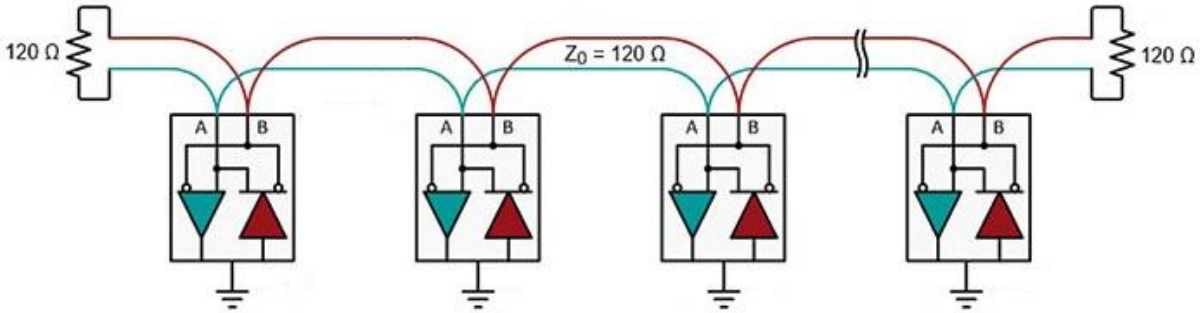
Per sfruttarne le potenzialità con Arduino è necessario utilizzare una scheda che converta l'interfaccia UART del microcontrollore nell'interfaccia RS-485.



La topologia di collegamento corretta del Bus è detta **Daisy Chain** ed è quella che limita maggiormente la riflessione del segnale.



Per **sopprimere totalmente le riflessioni** si usano i **Resistori Terminatori da 120 Ω** che vanno **collegati tra i morsetti A e B del primo e dell'ultimo dispositivo** connesso al Bus.



Indice

42. Protocollo CAN-Bus

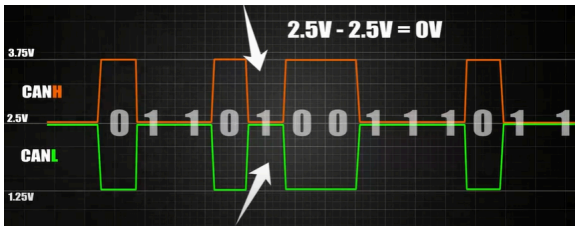
Il sistema **CAN-Bus** (**C**ontroller **A**rea **N**etwork) è uno standard seriale **progettato per funzionare in ambienti fortemente disturbati dalla presenza di onde elettromagnetiche** ed attualmente è utilizzato in

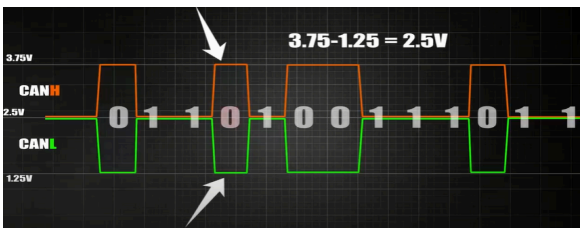
molte applicazioni industriali ed in tutti quegli ambienti dove è richiesto un alto livello di immunità ai disturbi quali:

- Automazione industriale
- Sistemi **SCADA** (**S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition)
- **B**uilding **M**anagement **S**ystems
- Applicazioni ferroviarie
- Aeronautica
- Interfaccia uomo-macchina (**H**uman **M**achine **I**nterface)

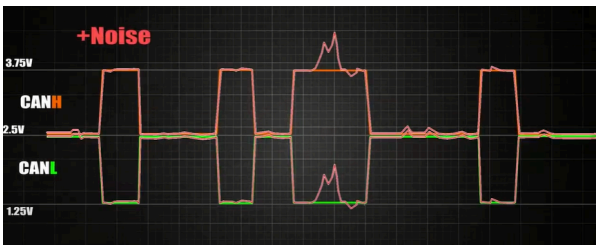
La sua robustezza alle interferenze dipende dalla tecnologia di trasmissione differenziale utilizzata. Fa uso di due canali **H** (**H**igh) e **L** (**L**ow). Il range di tensioni per il canale **H** va da **2.5V** a **3.75V** mentre per il canale **L** va da **2.5V** a **1.25V**.

Si ha la **trasmissione** del bit **1** quando **entrambi i segnali valgono 2.5V** e pertanto la differenza tra i segnali è pari a **0V.**
Si ha la **trasmissione del bit 0 quando il canale High ha il suo valore massimo ed il canale Low ha il suo loro valore minimo e la loro differenza è pari a 3.75-1.25= 2.5V, mentre si avrà la trasmissione del bit 1 quando entrambi i canali valgono 2,5V e la differenza sarà pari a 2,5-2,5=0V.**





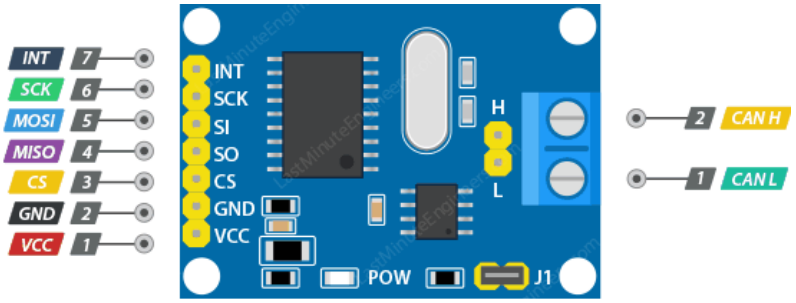
Essendo i **conduttori del Bus twistati**, l'interferenza interesserà **contemporaneamente entrambi i canali** facendo sì che **i valori differenziali si mantengano garantendo così la correttezza della comunicazione.**



La grande diffusione di questo tipo di interfaccia è dipeso dalle seguenti caratteristiche:

- consente il collegamento fino a 128 dispositivi multi Master
- comunicazione half-duplex
- distanze di comunicazione fino a 1600 mt.
- è dotata di sistemi ‘anti-collisione’ e controllo delle comunicazioni
- risulta più ‘leggera’ in termini di hardware rispetto ad altre

per poter implementare questo protocollo con Arduino è necessario utilizzare la scheda MCP2515 che si interfaccia con il Microcontrollore con il protocollo SPI (**S**erial **P**eripheral **I**nterface) ed in uscita con i canali **H** e **L** del protocollo **CAN-Bus**



Scheda MCP2515 per interfacciamento UART/CAN-Bus

INT questo pin genera un Interrupt quando riceve un messaggio valido e lo carica nel buffer

SCK pin del Clock

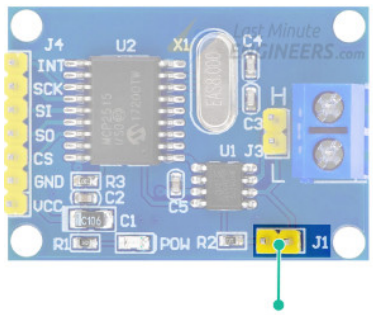
SI pin MOSI Master Output Slave In

SO pin MISO Master In Slave Output

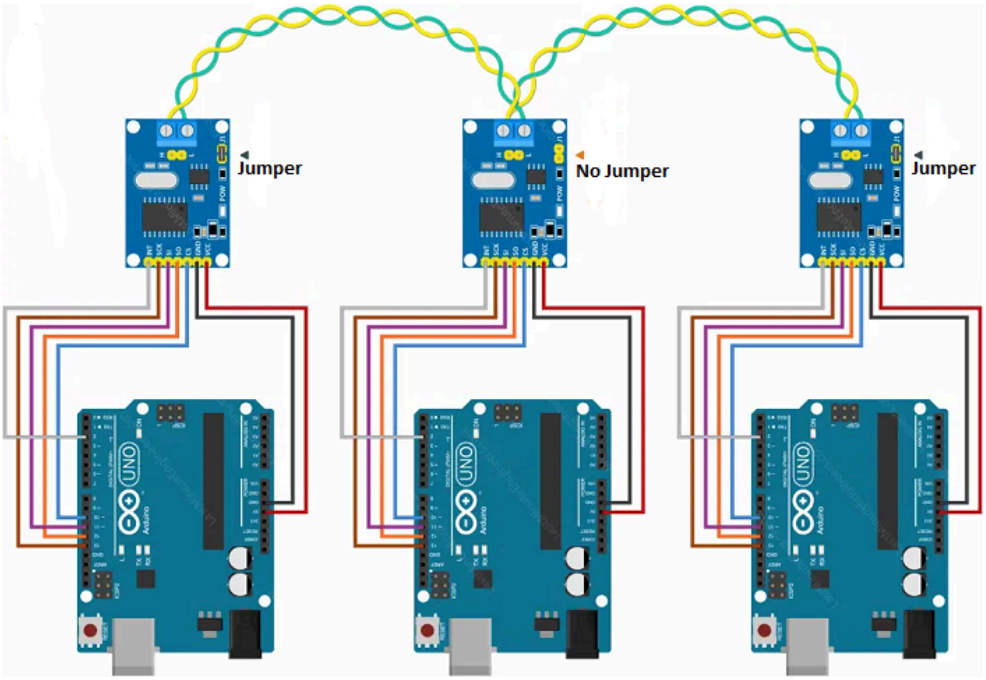
CS se stato logico basso il modulo è in modalità di ‘Transazione’

GND Ground

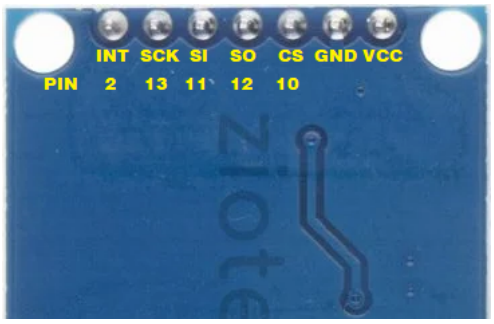
VCC Alimentazione 5V



La modalità di collegamento dei dispositivi connessi è la **Daisy Chain** (entra/esci) **ed il primo e l’ultimo** dispositivo del Bus dovranno **avere il resistore terminatore da 120Ω** che ha la funzione di **eliminare le riflessioni del segnale**. Nel caso della scheda MCP2515 questa ha già a bordo tale resistore che si abilita mediante l’apposito Jumper (J1).



Topologia Daisy Chain con Terminatori



per facilitare le connessioni, senza utilizzare i cavetti dupont, la scheda va inserita nella breadboard e la piedinatura, vista dal retro, è quella a fianco.

Indice

43 Tecnologia Bluetooth Low Energy (BLE)

Il termine Bluetooth indica una **tecnologia di rete point-to-point senza fili.**
Il Bluetooth permette a due device (dispositivi) di comunicare tramite una **frequenza radio sicura, a corto raggio ed a basso consumo (Low Energy).** Le reti realizzate con questa tecnologia si dicono **reti PAN (Personal Area Network).** Nel momento in cui i due dispositivi si connettono (**Paring**) questi si '**sintonizzano**' **su una** delle **79 frequenze** disponibili ed **ogni secondo** effettueranno ben **1600 'salti'** (**Frequency Hopping**) tra queste frequenze **rendendo** di fatto **impossibile che due reti** presenti nello stesso raggio di azione **possano interferire tra di loro**. Questo protocollo di comunicazione trova largo uso nell'**interfacciamento di dispositivi** che devono **operare a corta distanza e nella domotica**.
Nei nostri laboratori utilizziamo l'antenna BLE HC-05

Classi di dispositivi
In base alla **potenza trasmissiva**, cioè la massima distanza raggiungibile dal segnale senza ostacoli entro cui può avvenire il collegamento fra dispositivi BT, abbiamo 4 tipi di classi:

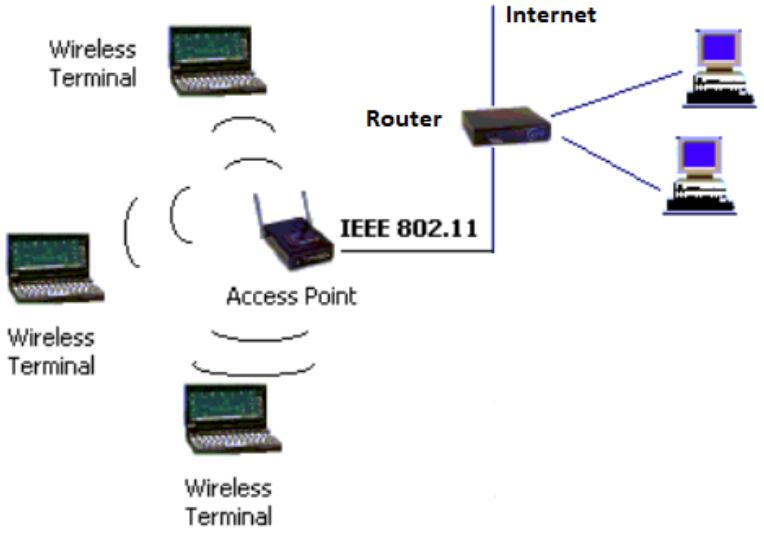
Classe	Potenza ERP		Distanza
	(mW)	(dBm)	(m)
1	100	20	~100
2	2,5	4	~10
3	1	0	~1
4	0,5	-3	~0,5

Indice

Tecnologia WI-FI

Il Wi-Fi, nel campo delle telecomunicazioni, indica una tecnologia che consente ai dispositivi che la compongono di collegarsi tra loro attraverso una

rete locale in modalità wireless (WLAN Wireless Local Area Network) ed alla rete Internet attraverso un router.



WirelessLan

Router (*Instradatore)* è il dispositivo che si occupa di governare il traffico dei dati provenienti da Internet verso i dispositivi all'interno di una rete secondo le tabelle di instradamento assegnando ad ogni dispositivo uno specifico indirizzo IP.

Access Point, in inglese *punto di accesso*, è un dispositivo di rete che, collegato ad una LAN permette all'utente di accedervi in maniera **wireless** (senza fili).

Indice

44. La rete Internet

Le ragioni della nascita del network (rete telematica), che oggi conosciamo con il nome di Internet, sono legate alle ricerche effettuate in America, dal **dipartimento della difesa** durante la guerra fredda, per trovare un Sistema che consentisse di comunicare tra i centri di comando militari anche nel caso in cui una parte dei Sistemi di trasmissione fossero stati distrutti dal nemico (rete Arpanet).

Funzionamento

Internet mette in comunicazione gli apparati ad essa connessa attraverso una serie di computer detti '**nodi di commutazione**'.

I pacchetti dati di una comunicazione, che viaggiano da un host* all'altro, non seguono percorsi di instradamento predefiniti ma quelli più congeniali nel preciso momento di attraversamento, di conseguenza, i pacchetti di una stessa comunicazione **possono seguire percorsi diversi verso lo stesso destinatario.**

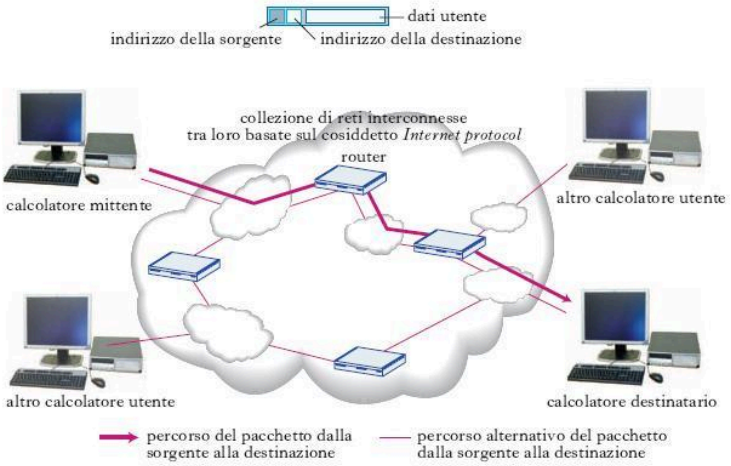
* per host si intendono computer o apparati terminali come smartphone, automobili, elettrodomestici, ecc. in grado di comunicare tra di loro

Protocolli

Per potersi collegare a Internet e usufruire dei relativi servizi, **chi richiede un servizio sulla rete è detto client** , è necessario che un qualsiasi dispositivo elettronico possa "dialogare" con il destinatario ed i nodi interni di rete, ciò è reso possibile dai **protocolli di rete**.

I protocolli più importanti sono il **Transmission Control Protocol** (Protocollo di Controllo di trasmissione dati, **TCP**), l'**User Datagram Protocol** (**UDP**) e l'**Internet Protocol** (Protocollo Internet, **IP**):

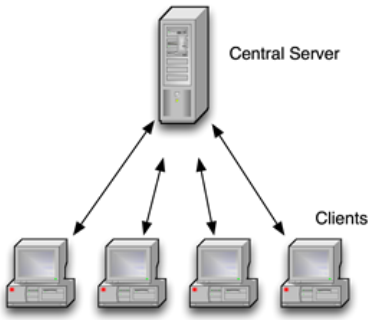
- il **TCP** controlla/verifica che i dati arrivino correttamente
- l'**UDP** si limita ad inviarli senza controllarne la correttezza (si usa dove l'eventuale perdita di pacchetti può essere trascurata ad esempio nel campo del gioco multiplayer)
- il protocollo **IP** consente l'indirizzamento/instradamento tra i nodi interni di commutazione dei pacchetti.



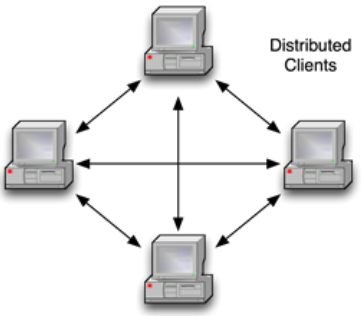
Modelli di rete

I due modelli di rete più diffusi sono:

- **client/server** nel quale i dati risiedono sul server ed il client ne fa richiesta attraverso la rete Internet
- **Peer to Peer** nel quale i dati possono risiedere su tutti i computer che compongono la rete ed ogni computer può fungere sia da client che da server in funzione delle richieste



Client / Server

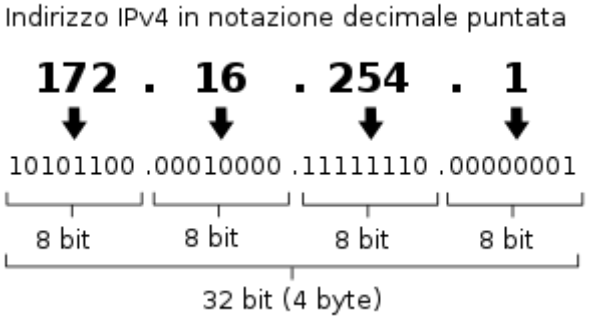


Peer to Peer

Indirizzo IP

L'**indirizzo IP** è un codice numerico che identifica un dispositivo connesso a Internet tramite il sistema **IP**, Internet Protocol. Con questo **indirizzo**, ogni dispositivo è in grado di comunicare con altri nodi della rete ed inviare e ricevere pacchetti di dati. E' determinante, per il funzionamento della rete, che tale indirizzo **sia univoco per ogni dispositivo**.

Standard IPv4



L'indirizzo **IPv4** è costituito da **32 bit** (4 byte) suddiviso in 4 gruppi da 8 bit (1 byte, ottetto), separati ciascuno da un punto. Ciascuno di questi 4 byte è poi convertito in formato decimale di più facile identificazione, che semplifica la lettura e la memorizzazione da parte di noi umani, quindi ogni numero varia tra 0 e 255 (da 000000000 a 11111111 in formato binario) .

Nello standard IPv4 (Internet Protocol versione 4), essendo costituito da **32 bit**, le combinazioni utilizzabili sono 2^{32} **pari a circa 4 miliardi**. Nel 1981, anno dell'introduzione di questo standard, sembravano tanti ma oggi, grazie al dilagare dell'Internet of Things e dell'Industria 4.0 non sono più sufficienti e si è dovuto ricorrere allo standard **IPv6 che è costituito da 128 bit (64 byte), corrispondente, approssimativamente, un numero decimale a 40 cifre...**

IP pubblico e IP privato

L'indirizzo **IP pubblico** è un indirizzo IP **visibile e raggiungibile da tutti** gli host della rete Internet. **Ogni host connesso a Internet deve possedere un proprio indirizzo IP pubblico univoco,** cioè **diverso da quello di qualsiasi altro host** collegato al World Wide Web. Per tutta la durata della connessione, in tutto il mondo, non possono esistere due host con lo stesso indirizzo IP pubblico.
Tutti i dispositivi appartenenti a una **LAN** (Local Area Network) utilizzano un proprio indirizzo **IP privato** (non visibile dall'esterno della rete locale) ma si connettono a Internet utilizzando **un unico indirizzo IP pubblico**, pertanto, una LAN privata potrò avere indirizzi IP uguali ad un'altra LAN privata perchè non potranno esserci conflitti dato che ognuna di esse si 'affaccerà' alla rete Internet attraverso il proprio de unico IP pubblico.
Per comprendere meglio questo concetto facciamo un esempio: immaginiamo **l'indirizzo IP pubblico come l'indirizzo ed il numero civico** di due **diversi hotel** ubicati in **due posti diversi del mondo** e, di conseguenza, con due indirizzi **che non possono essere uguali.**

Possiamo invece considerare gli indirizzi IP privati come se fossero i numeri delle camere dei rispettivi hotel che possono anche essere numeri uguali ma senza che ciò possa creare confusione o fraintendimenti su quale sia la camera assegnata ad un determinato cliente di un determinato hotel (non perdiamo di vista che lo scopo è raggiungere uno specifico host all'interno della rete internet)

IP pubblico In base al tipo di Service Provider Vodafone ecc.), pubblico

dinamico servizio in Italia un host dinamico

e IP pubblico statico fornito dall'ISP (Internet sono Telecom, può avere un IP oppure statico.

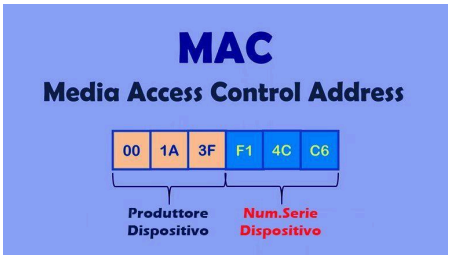
L'**IP pubblico dinamico** può variare ogni volta che ci colleghiamo a Internet.
Viene assegnato "in prestito" ed è utilizzato nella maggior parte dei casi per l'utenza domestica. Quando un dispositivo non è più connesso a Internet l'IP pubblico viene reso disponibile e può essere assegnato a un altro dispositivo.
L'**IP pubblico statico** ha costi più elevati, non cambia mai e rimane assegnato allo stesso dispositivo anche quando non è connesso a Internet. Non cambiando può essere sempre raggiunto da remoto.

DHCP (Dynamic Host Configuration Protocol)
Anche In una rete **LAN**, ogni host ha bisogno di un indirizzo IP, scelto in modo tale che appartenga all'insieme di indirizzi possibili assegnati all'intera sottorete a cui è collegato e che sia **univoco**, cioè è indispensabile che non ci siano altri calcolatori/apparati che stiano già utilizzando quell'indirizzo. Di ciò se ne occupa il **protocollo DHCP** che assegnerà **automaticamente** gli indirizzi IP alle macchine che via via si connetteranno alla rete.
I **Router**,comunque, se necessario, consentono di assegnare un indirizzo **IP statico** ad una determinato apparato associandolo al **Mac Address** dell'apparato stesso, una volta settato, quell'indirizzo non verrà più assegnato ad altri apparati ma riservato a quello specificato nella fase di configurazione.

Server DNS (Domain Name Service)
Il **DNS** è il sistema che regola **la traduzione dei nomi dei siti in indirizzi IP**.
Quando si digita il nome del sito da visitare (ricordare l'IP del sito sarebbe praticamente impossibile), il browser invia una richiesta al server DNS, che cerca l'indirizzo IP associato al nome e lo comunica al browser che, a questo punto, lo utilizzerà per inviare la richiesta di connessione al sito.
Se, ad esempio, vogliamo conoscere l'IP della pagina principale di Google, basta fare un 'ping' verso esso aprendo una finestra di prompt digitando ***ping www.google.com,*** oppure, possiamo fare un tracciamento con il comando ***tracert www.google.com*** che ci consentirà di vedere anche il percorso che i dati di richiesta faranno, tra i vari nodi di commutazione, prima di raggiungere il server.

Come controprova, possiamo copiare ora l'IP trovato e lo inseriamo nella barra degli url del browser per verificare se, effettivamente, il risultato è lo stesso.

Mac Address (Media Access Control Address)
Il Mac Address è un codice a 48 bit (6 byte) **assegnato univocamente ad ogni dispositivo in grado di connettersi alla rete**. Tale codice viene espresso in notazione **esadecimale**. I primi tre ottetti indicano il costruttore mentre gli altri tre identificano l'apparato.



Indice

45. Tecnologia Web Server

Per comprendere il funzionamento delle comunicazioni nella rete Internet è necessario distinguere tra:

- **HTML** (**H**yper**T**ext **M**arkup **L**anguage), è il linguaggio che definisce la struttura di un sito web (ctrl+u per codice pagina, F12 per console);
- **UR**L (**U**niform **R**esource **L**ocator), è l'indirizzo per raggiungere la risorsa (ad esempio un sito web);
- **HTTP** (**H**yper**T**ext **T**ransfer **P**rotocol) è il protocollo che regola il modo in cui questa pagina viene trasferita dal server al client.

L'utente inserisce l'**URL** nella barra degli indirizzi del browser ed invia questa richiesta mediante il protocollo **HTTP** verso il server (HTTPS, se il server utilizza dei sistemi di sicurezza).
Nel server risiedono i file, strutturati con il linguaggio **HTML,** che compongono il sito web oggetto della richiesta del Client che verranno restituiti al browser che ne ha fatto richiesta (**Request/Response protocol)** visualizzando sullo schermo dell'utente la pagina web desiderata.

Per effettuare delle richieste al Server mediante l'**HTTP** esistono diversi metodi, i due più importanti sono senza dubbio il metodo **GET** e **POST**.
La differenza principale tra i due metodi è che, nel caso del **GET,** i dati della richiesta **sono trasmessi direttamente nell'URL e sono quindi visibili**, mentre, nel caso del **POST,** i dati della richiesta **sono contenuti nell'HTTP e quindi non visibili.**
Per capire la differenza facciamo l'esempio di cosa succede salvando l'url del sito. Salvando l'url del metodo **GET**, nel riaprirlo, troveremo la pagina così come l'abbiamo salvata (conveniente ad esempio per un negozio on-line), mentre, nel caso del metodo **POST,** dovremo rifare la procedura di richiesta (ad esempio sito con dati sensibili dell'utente).

Riguardo al linguaggio **HTML** ci limiteremo allo stretto necessario per realizzare delle semplici pagine per gestire i nostri Sistemi, visto anche il limite di memoria disponibile sulla scheda che le ospiterà, ma non possiamo però non parlare del **linguaggio CSS** (**C**ascading **S**tyle **S**heets) che, integrato all'HTML, **consente di formattare a nostro piacimento la pagina Web**.

Volendo fare un paragone con il corpo umano l'**HTML** sarebbe lo scheletro, il **CSS** l'aspetto esterno, il **JavaScript** ed il **PHP** il sistema nervoso perché sono i linguaggi di programmazione che 'animano' le pagine web consentendo di interagire con esse e, nello specifico, **il JavaScript lato Client ed il PHP lato Server**.

La prima informazione che serve al browser, per interpretare i dati della pagina HTML, è la dichiarazione **<!DOCTYPE html>** mentre, i dati, devono essere inseriti all'interno del 'contenitore' delimitato da **<html> </html>** al cui interno vi saranno altre due sezioni delimitate da **<head> </head>** e **<body> </body>**.

```
<!DOCTYPE html>
 <html>
<head>
```

In questa sezione vengono inseriti ulteriori dati che servono al browser per interpretare la pagina, lingua, set di caratteri, (metadati) ecc. Tra questi il tag* <title>nome_pagina </title> *che farà comparire nome_pagina nel tab della pagina stessa*

</head>

<body>

in questa sezione viene inserito ciò che verrà realmente visualizzato dal browser e, ***per definire e delimitare gli elementi che compongono la pagina, si ricorre ai tag*** *(Marcatori), all'interno dei quali,* ***si possono inserire informazioni aggiuntive di formattazione mediante il linguaggio CSS*** *che prevede l'inserimento* ***di attributi*** *costituiti da* ***chiave*** *e* ***valore.***

</body>

</html>

* i metadati indicano le caratteristiche dei dati

Tag HTML più comuni		
Tag	Funzione	esempio
	paragrafo	
 	a capo	
<h1><h2>.....<h6> Headings	dimensione carattere intestazione (h1 grande h6 piccola)	
	spazio vuoto	la & indica 'carattere speciale' non necessita di <> ne di essere chiuso con /
<a>	link definito da attributo href	testo_link
	consente di definire un'area della pagina	
	consente di inserire degli attributi di formattazione solo ad una parte del testo	inserire una parola in grassetto</spa> all'interno di una frase
<img>	inserisce immagine con percorso indicato da attributo scr, testo alternativo da alt, larghezza immagine width e altezza immagine da height	<img src="immagine tramonto.jpg" alt="tramonto" width="500" height="600">
<button>	crea un pulsante cliccabile al quale assegnare delle azioni mediante gli attributi	<button type="button" onclick="alert('Hello world!')">Clccami!</button>

<form>	crea un modulo editabile	
<input>	decide le azioni da compiere con i dati inseriti in modulo creato con il tag <form>	
<!-- -->	consente di inserire un commento nel file HTML	<!-- commento-->
<style>	**importantissimo** perché consente l'inserimento degli attributi del linguaggio **CSS**	usato come tag all'interno dell'<head></head> consente di inserire direttamente in quell'area tutte le formattazioni con l'uso dei selettori, classi o id. usato invece come attributo consente di inserire la formattazione in-line

La sintassi per il CSS è caratterizzata sempre dall'attributo style= seguito da chiave:"valore" concluso con l'uso del punto e virgola ;

Attributi CSS più comuni		
chiave	valore	funzione
	testo non formattato<span style= chiave:"valore";> testo formattato testo non formattato	per applicare il CSS solo ad una parte del testo si usa il tag inserendo al suo interno il tag style seguito da chiave:"valore"
background-color:	"colore_scelto";	colora lo sfondo dell'elemento
color:	"colore_scelto";	colora il testo
margin:	"valore"; "valore valore"; "valore valore valore valore";	definisce la distanza tra l'elemento interessato e gli altri elementi. Se un solo valore questo viene applicato a tutti i lati, se due valori il primo il sopra e sotto ed il secondo destra e sinistra, se quattro valori questi verranno assegnati in senso orario partendo dal sopra
padding:	"valore"; "valore valore"; "valore valore valore valore";	definisce la distanza tra l'elemento ed il suo contenuto. Per il valore valgono le stesse regole della chiave margin
text-align:	"valore";	allinea il testo left, right, center, justify, ecc.
font-style:	"valore";	normal, italic, oblique ecc.
font-size:	"valore";	dimensione del testo

font-weight:	“valore”;	livello del grassetto
height:	“valore”;	definisce l’altezza dell’elemento
width:	“valore”;	definisce la larghezza dell’elemento
border:	“valore tipo colore”;	inserisce un bordo della dimensione, del tipo (solid, dashed, dotted, ecc.) e del colore desiderato
border-radius:	“valore”; “valore valore”; “valore valore valore valore”;	arrotonda i bordi dell’elemento con valore 50% crea un cerchio. Per il valore valgono le stesse regole della chiave margin e padding
box-shadow:	“valore valore valore colore”;	applica una ombra all’elemento primo valore a destra, secondo valore sotto, terzo valore sfumatura

Inserimento CSS

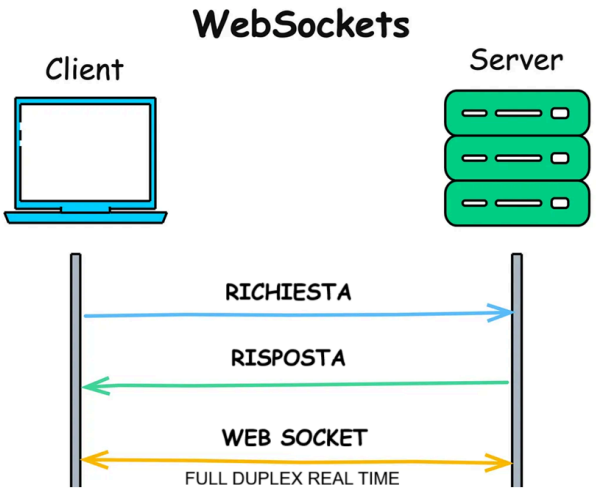
- **esterno:** viene creato un file .css all’interno del quale si inserisce il CSS e si crea un link nella sezione <head> dell’HTML con il tag <link href="nomefile.css" rel="stylesheet" type=”text/css”>
- **interno:** nella sezione <head> dell’HTML si inserisce il tag <style></style> all’interno del quale si inserisce il CSS utilizzando i **puntatori** (h1 {}, p {}, div {} ecc.), le **classi** (.nomeclasse {}) o gli **id** (#nomeid {})
- **in-line:** direttamente nel tag dell’elemento <p style=”color: red; font-weight: boder;”>

Il termine Cascading dell’acronimo **CSS** (**C**ascading **S**tyle **S**heet) intende che a ‘cascata’ verrà preso in considerazione il comando più prossimo all’elemento. Per chiarire, **quello in-line prevale su quello interno che, a sua volta, prevale su quello esterno**.

Aggiornamento pagina Client Web Server

Se è necessario ricaricare la pagina sul Client, il metodo più semplice è quello di inserire nell’<head> della pagina web la stringa **<meta http-equiv="refresh" content="tempoespressoinsecondi">**, per farlo premendo un pulsante va inserito nel <body> **<button onclick="location.reload()">Ricarica Pagina Ora</button>**

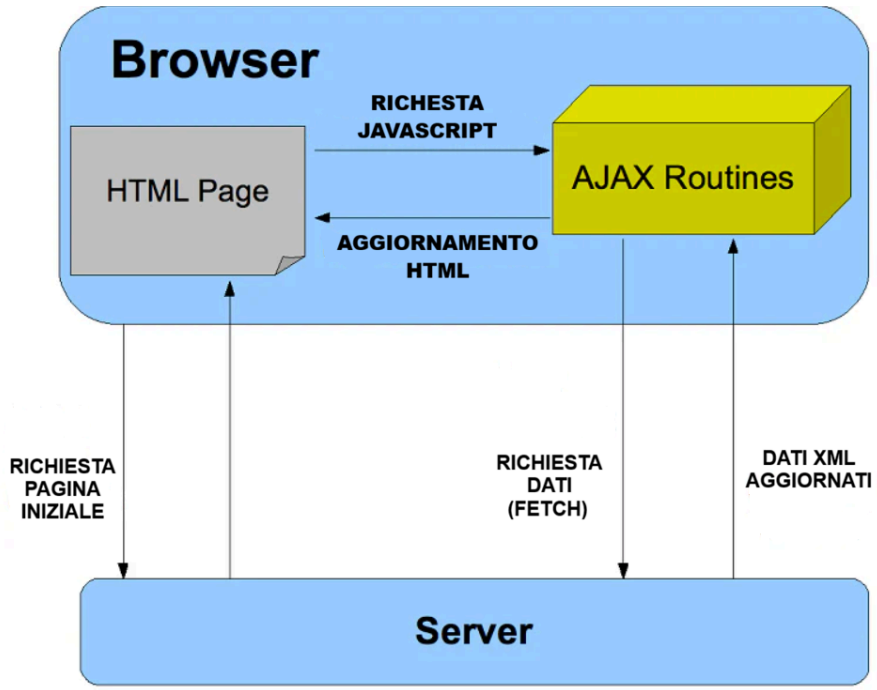
per farlo fare invece direttamente dal Server in caso di eventi rilevati da esso, è necessario ricorrere al protocollo **Websockets**



Il protocollo **WebSockets** consente, una volta che è stata stabilita la connessione tra Client e Server, di **aprire un canale comunicativo Full Duplex** che consente ai dispositivi di comunicare tra di essi **in tempo reale**, **senza necessità di effettuare richieste da parte del client** o refresh periodici automatici o manuali.

Tecnologia AJAX Asynchronous **J**avaScript **A**nd **X**ML

AJAX (si pronuncia ey-jaks) **è una tecnica di sviluppo web che permette di aggiornare parti di una pagina web senza ricaricarla, rendendo le applicazioni più veloci, fluide e reattive.** Le richieste di dati in questo caso è sempre Client/Server e viene fatto attraverso il cosiddetto **'fetch'** che significa letteralmente **'far portare'**, infatti, il **Client effettua la richiesta di dati** (come se si rivolgesse in un ristorante ad un cameriere) e continua ad occuparsi delle sue cose, **quando i dati sono disponibili** (Asincroni quindi non in tempo reale come il Websocket) **il Server li invia, il Client li riceve ed aggiorna solo quei valori senza ricaricare tutta la pagina**, da qui la maggior fluidità della pagina.



Nello sviluppo web moderno, per la comunicazione dei dati, alla tecnologia **AJAX** si associa la codifica **JSON .**

JSON (JavaScript Object Notation)

La **codifica JSON** (si pronuncia geison) **è la più diffusa per il trasferimento dei dati nel Web**, **è un formato 'letterale' ed è quindi facilmente interpretabile dagli esseri umani,** anche se è destinato alle macchine, ed è **supportata da quasi tutti i linguaggi di programmazione**.
La sintassi è la seguente:

- il codice è contenuto tra parentesi graffe **{ }**
- ogni elemento è caratterizzato da **"chiave" : "valore"** racchiusi tra doppi apici " " e separati dai due punti **:**
- gli elementi vengono separati con la virgola

{"nome":"Mario", "cognome":"Rossi", "età": 35} da notare che, **se il valore non è una stringa** (insieme di caratteri) **ma un numero, non vanno inseriti i doppi apici**

Una cosa molto importante è che i dati possono essere 'annidiati' (in inglese 'nested') e per farlo è necessario racchiudere tali dati tra altre due parentesi graffe.

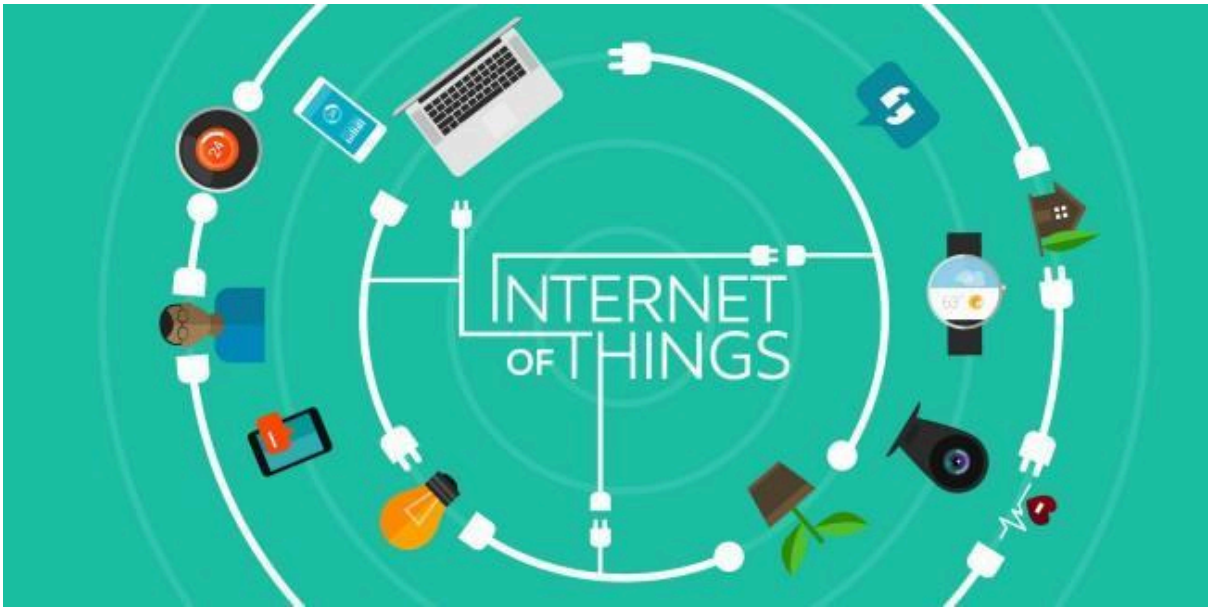
```
{"nome":"Mario","cognome":"Rossi","indirizzo":{"città":
"Milano","Via":"Napoleone","numero":12}}
```

Se un elemento contiene più valori dello stesso tipo, questi possono essere inseriti in un Array delimitato da parentesi quadre **[ ]**

```
{"sensore":"gps","orario":15.00,"coordinate":[48.756080
,2.302038]}
```

Per consentire ad un dispositivo di trasmettere dei dati in formato JSON è necessario che il dispositivo effettui la **Serializzazione** di tali dati, **effettui cioè la codifica dei dati in tale formato**.
La procedura inversa, cioè **la ricezione dei dati e la possibilità quindi di poterli estrarre dalla comunicazion**e, si dice **Deserializzazione.**
formattatore JSON memoria occupata JSON

Indice



46. IoT Internet of Things (Internet delle Cose)
L'Internet of Things è un insieme di tecnologie che consentono **ai dispositivi** ('cose') di **comunicare tra di loro attraverso la rete internet**.

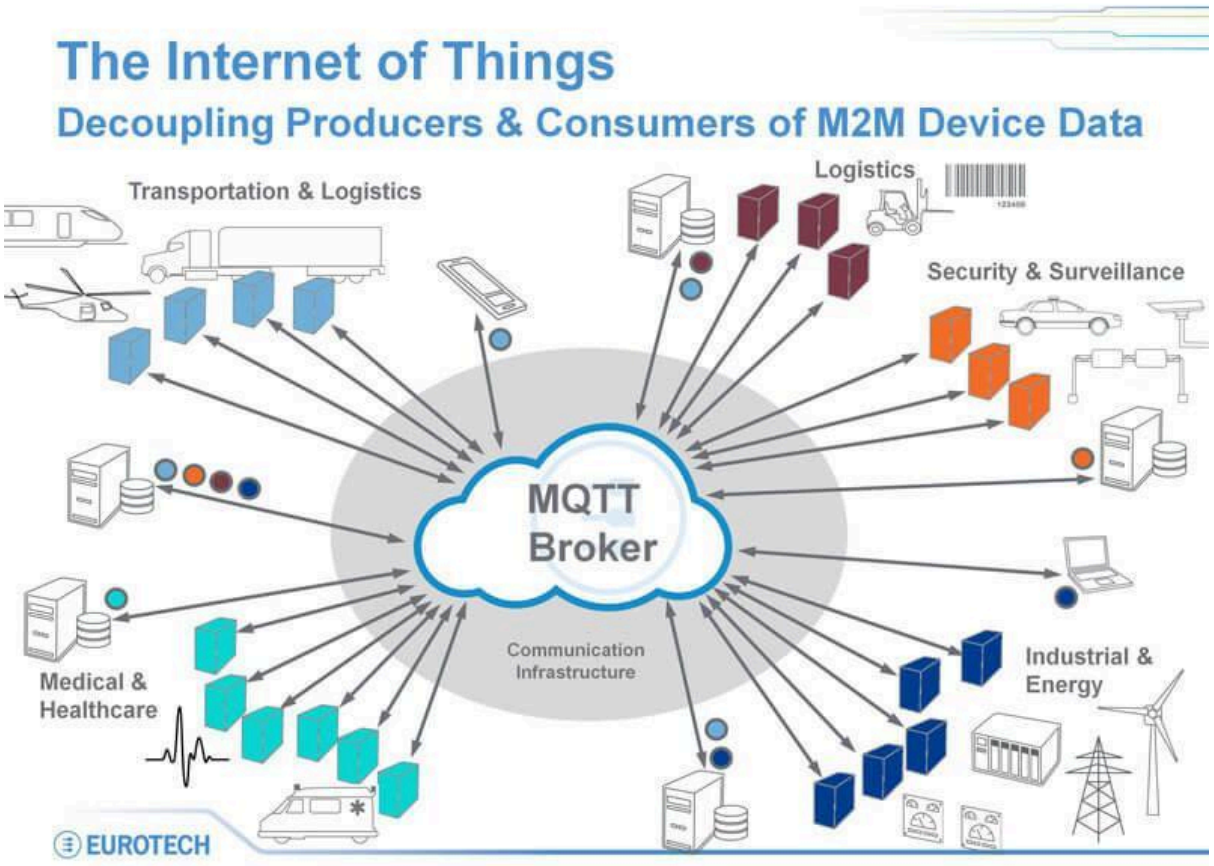
Le più diffuse tecnologie che fanno parte dell'Internet of Things sono:

- **RFId (Radio Frequency Identification)**: è la tecnologia più semplice con cui un oggetto può integrarsi nell'Internet of Things. Racchiude tutti gli standard di identificazione automatica in radiofrequenza, il metodo più diffuso è il **Near Field Communication (NFC)** ;
- **Reti PAN (Personal Area Network)**: si tratta di reti a corto raggio e tra queste rientrano le reti **Bluetooth BLE** (Bluetooth Low Energy) e **Zigbee**;
- **WiFi**: protocolli per l'accesso wireless alle reti a banda larga che però **trovano poco impiego nell'IoT data la grande mole di informazioni che trasportano** (occupazione di banda) ed il consumo energetico;
- **Power Line Communication**: trasmissione delle informazioni attraverso i conduttori di alimentazione dei dispositivi, detti anche *Sistemi ad onde convogliate*;
- **Protocollo MQTT (Message Queue Telemetry Transport):** l'**MQTT** è un protocollo di messaggistica leggero di tipo **publish-subscribe** (*pubblicazione-iscrizione*). È stato progettato per occupare pochissima banda di trasmissione ed è diventato uno standard nel campo dell'**IIoT** (Industrial Internet of Things) e della **Domotica** e per la sua 'leggerezza' viene utilizzato anche per il funzionamento dei social come ***Facebook e Twitter***.

CAMPI DI APPLICAZIONE				
				
SMART CAR	SMART CITY	SMART HOME	INDUSTRIAL IOT	SMART AGRICULTURE
Connessione delle auto per comunicare informazioni in tempo reale al consumatore, connessione tra veicoli o tra questi e l'infrastruttura circostante per la prevenzione e la rilevazione degli incidenti	Monitoraggio e gestione degli elementi di una città (ad esempio mezzi per il trasporto pubblico, illuminazione pubblica e parcheggi) e dell'ambiente circostante per migliorare vivibilità, sostenibilità e competitività	Soluzioni per la gestione in automatico e/o da remoto degli impianti e degli oggetti connessi dell'abitazione, con il fine di ridurre i consumi energetici e migliorare il comfort e la sicurezza e delle persone al suo interno	Connessione dei macchinari M2M (Machine to Machine), degli operatori e dei prodotti per abilitare nuove logiche di gestione della produzione.	Monitoraggio di parametri microclimatici a supporto dell'agricoltura per migliorare la qualità dei prodotti, ridurre le risorse utilizzate e l'impatto ambientale

La 'cosa', integrata nella rete, deve essere innanzitutto **identificabile**, cioè dotata di un **identificativo univoco nel mondo digitale** e poi deve essere **connessa** per poter trasmettere e ricevere informazioni.
Per poter affrontare l'argomento dell'**IoT** si rende quindi necessario trattare gli argomenti indispensabili relativi al funzionamento della rete Internet.

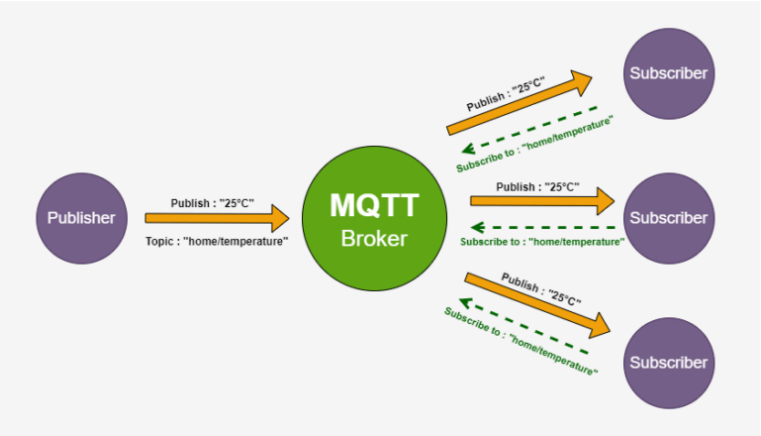
47 Protocollo MQTT (Message Queue Telemetry Transport)



Il modello su cui si basa il protocollo **MQTT** è il **publish/subscribe (pubblicare/iscriversi)** che **è un'alternativa al modello client/server** dove il client comunica direttamente con un endpoint, cioè il server. **Il problema del modello client/server è che per far comunicare tra di loro i client è indispensabile che i loro IP siano statici e noti**.
Con l'**MQTT**, invece, **si possono mantenere separati i client tra loro** in modo che un certo client (che prende il nome di "publisher", colui che pubblica) possa inviare un determinato messaggio agli altri client che invece sono iscritti (si chiameranno "subscriber", coloro che sono iscritti). In questo modo il publisher e il subscriber possono ignorare l'esistenza di altri client dato che viene aggiunto un nuovo attore, un client filtro detto "**broker**", **che sarà l'unico a dover avere un indirizzo statico** ed avrà il compito di gestire il flusso dei messaggi tra i client. **Il nodo di scambio dell'informazione tra publisher e subscriber viene detto topic** (argomento).
Tra le peculiarità del protocollo quelle più rilevanti sono quelle di essere **molto 'leggero' in termini di occupazione di banda** (richiede poche risorse di rete per il suo flusso di dati) e di essere **'scalabile'** (con il termine scalabile si intende la **capacità di un sistema di adattarsi in base alle esigenze**) .

– Il **publisher** è un dispositivo (ad esempio un sensore) in grado di pubblicare dei messaggi nel Sistema MQTT;
– Il **subscriber** è un dispositivo che ha bisogno di ricevere i messaggi 'postati' dal/dai publisher (ad esempio un Sistema di controllo remoto);

– Il **broker** è un software che si inserisce tra il **publisher** ed il **subscriber** e che si occupa di gestire lo scambio dei messaggi tra loro.

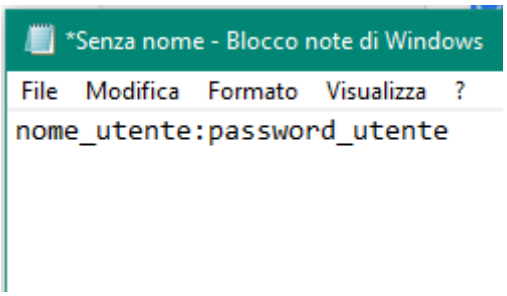


Uno dei broker più diffusi è sicuramente **Mosquitto**. Si tratta di un software a 'riga di comando' (senza interfaccia grafica) ed è tra i pochi ad essere freeware ma richiede conoscenze approfondite per poter essere usato.

Uso di Mosquitto

Per configurare il Broker Mosquitto per accedervi con user e password bisogna procedere nel seguente modo:

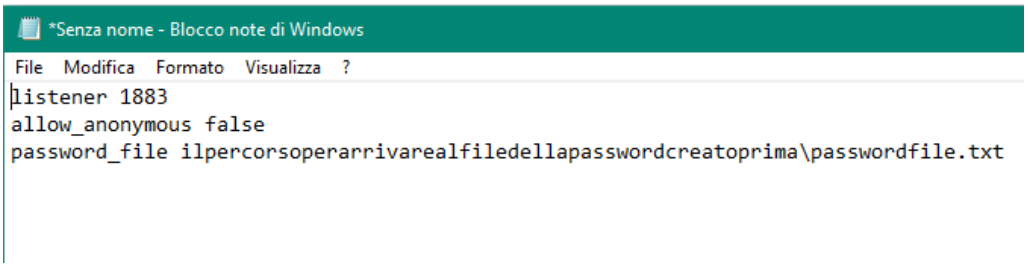
- Nella cartella di mosquitto creare un file di testo come quello che segue e salvarlo con il nome che si preferisce, in questo esempio lo chiameremo 'passwordfile.txt'



- bisogna procedere ora con la criptazione eseguendo il comando

mosquitto_passwd -U passwordfile.txt

- bisogna ora creare il file di configurazione (un file di testo con estensione .conf), da utilizzare in fase di lancio di Mosquitto, che contenga queste tre righe

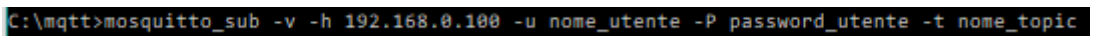


in questo esempio lo salveremo come 'nomefile.conf'

il comando per lanciare Mosquitto come Broker sarà

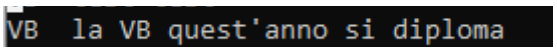
mosquitto -v -c nomefile.conf

I dispositivi che dovranno **'iscriversi'** al Topic nella finestra di prompt digiteranno, sempre dopo essersi portati nella cartella di mosquitto, il seguente comando:
mosquitto_sub -v -h indirizzo_ip_broker -u nome_utente -P password_utente -t nome_topic



Per **'pubblicare'** un messaggio nel topic da inviare a tutti gli **'iscritti'** il comando è il seguente:
mosquitto_pub -h indirizzo_ip_broker -u nome_utente -P password_utente -t nome_topic -m "la VB quest'anno si diploma" -i nome_di_identificazione

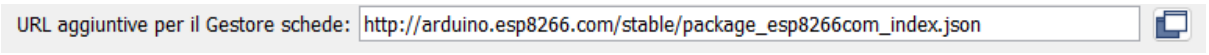
Per inviare una frase è necessario racchiuderla "tra i doppi apici" e questo ciò che comparirà nella finestra dell'iscritto al topic



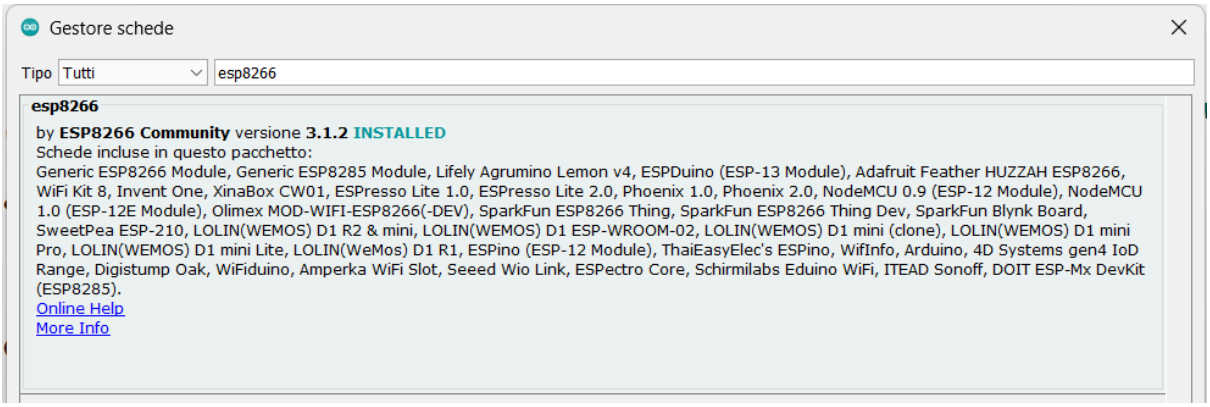
48. Wemos D1 Mini Pro

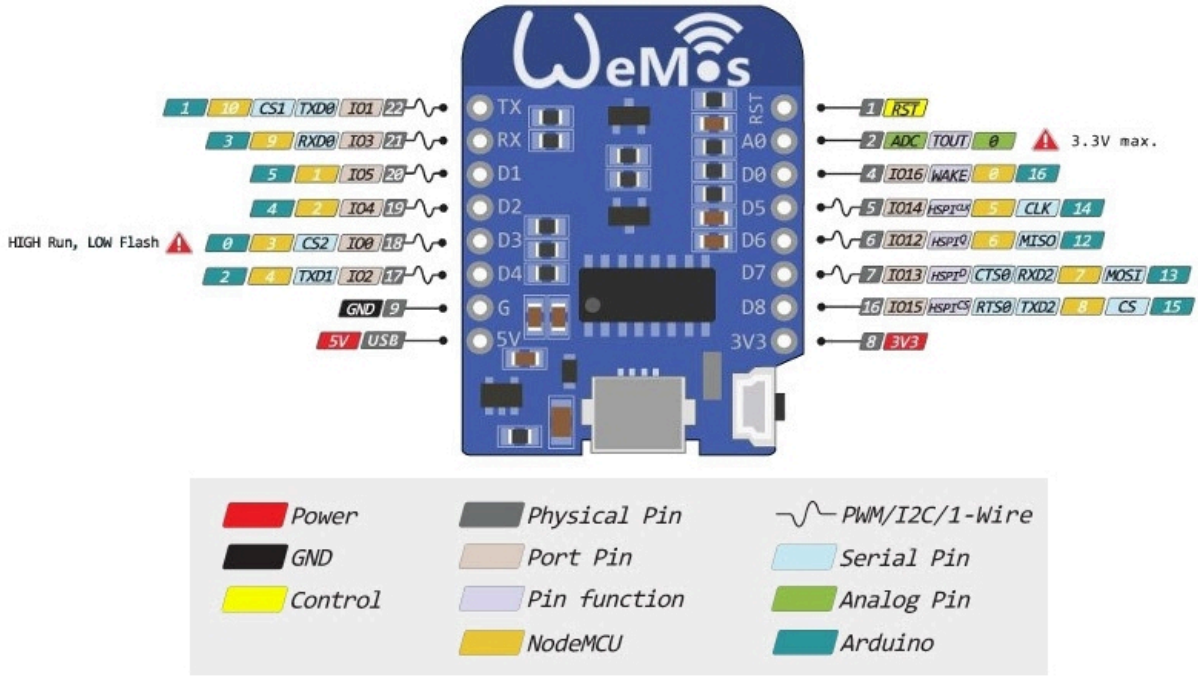
La scheda **Wemos D1 Mini** ha a bordo il chip **Wi-Fi ESP8266** che è l'integrato più diffuso nel campo dell'**IoT**, è una scheda che può essere programmata utilizzando l'**IDE** di Arduino ed è in grado di svolgere **tutte le funzioni di microcontrollore.**

Per aggiungere questa scheda è necessario aggiungere questo Url **http://arduino.esp8266.com/stable/package_esp8266com_index.json** dal menù File->Impostazioni



ed installare dal gestore schede il seguente file





Le differenze tra Arduino e Wemos D1 mini sono che la sua logica di funzionamento è a **3,3V** e che la **corrispondenza dei pin è diversa**, infatti, in fase di

programmazione, **dovremo fare riferimento ai pin marcati in verde ma, nei collegamenti fisici, a quelli riportati sulla scheda.**
Esempio: se devo far passare stato logico alto l'uscita D1 della Wemos, nel software di programmazione dovrò scrivere digitalWrite(5, HIGH);.

Specifiche tecniche:

- 11 I/O digitali, tutti i pin sono provvisti di interrupt e supportano PWM, I2C , One-Wire (D1-SCL-D2 -SDA)
- 1 Pin analogico A0 (tensione massima di ingresso 3.3V)
- Pin D4 Pulled-Up e collegato a LED on board
- Pin D3 Pulled-Up
- Connessione Micro USB

Interfaccia I2C:

- SDA => D2
- SCL => D1

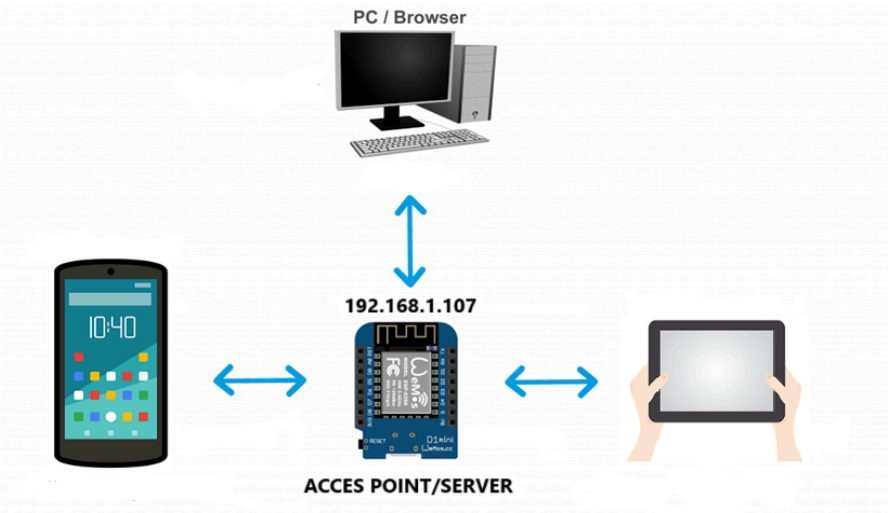
Interfaccia SPI

- M-CLK => D5
- MISO => D6
- MOSI => D7

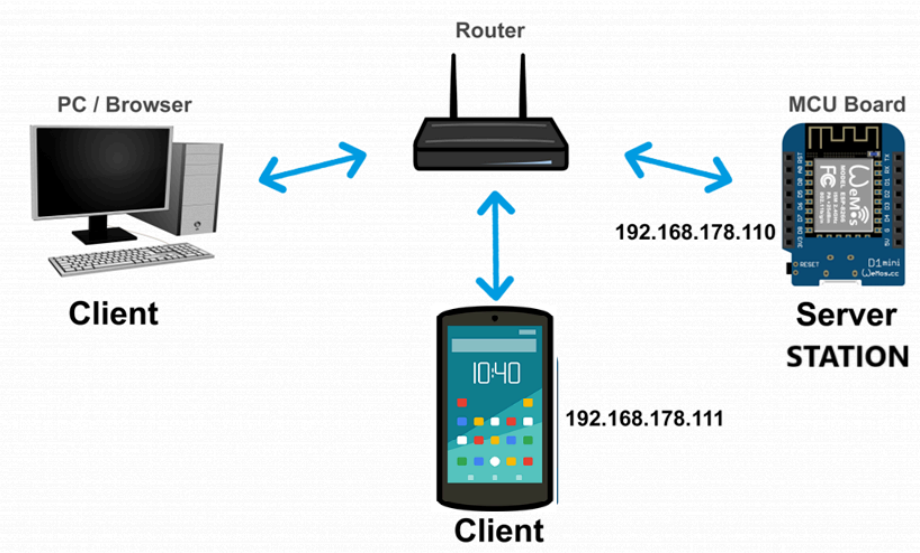
- (SPI Bus SS (CS) è D8)

Modalità di funzionamento
La scheda offre l'opportunità di poter funzionare sia in modalità Access Point che Station.

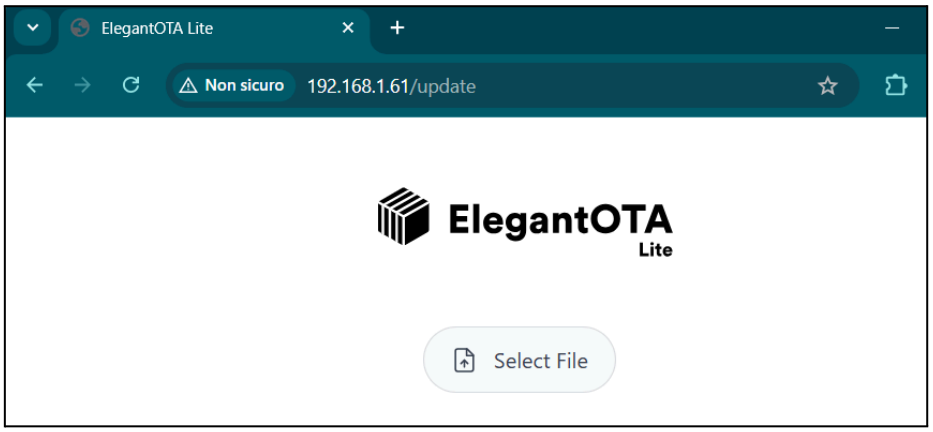
Access Point
In questa modalità la scheda **si comporterà sia da Router che da Server** consentendo ai Client di accedervi, in pratica, ci permette di **creare una nostra rete LAN.**



Station
In questa modalità la scheda si comporterà da **Server** raggiungibile attraverso il **Router** di una rete **LAN** esistente



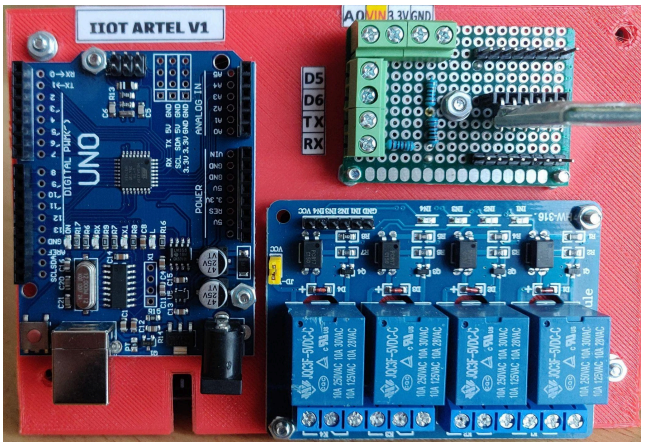
OTA (**O**ver **T**he **A**ir)
Questa straordinaria scheda **consente il caricamento del programma direttamente wireless senza usare il cavo seriale.**
Questa funzione permette di fare modifiche allo scketch e di caricarlo anche se la scheda è già posizionata e magari non è facilmente raggiungibile fisicamente.
Per poterlo fare, bisogna ricorrere alla libreria *ElegantOTA.h*. Questa libreria contiene al suo interno una pagina web che può essere richiamata digitando nel browser l'indirizzo ip seguito dall'header **/update**. Dopo aver esportato lo scketch in formato .bin, questo dovrà essere caricato dalla pagina web di cui sopra.



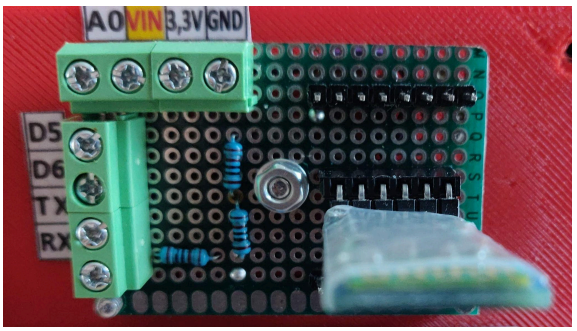
Indice

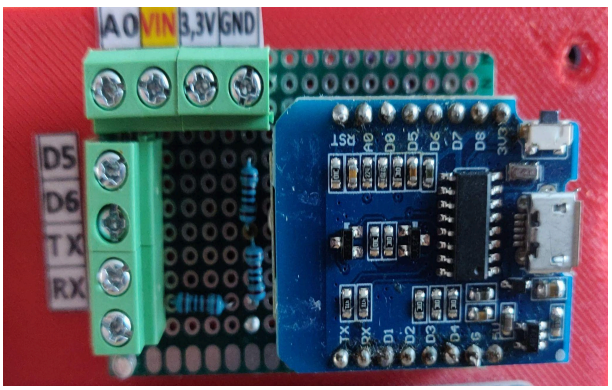
49. IIOT ARTEL V1

L'apparecchiatura nasce per facilitare l'interfacciamento tra i dispositivi domotici, come l'antenna HC-05 (Bluetooth Low Energy) o la scheda Wemos D1 mini (IIOT), con circuiti di potenza superiore. Nella versione V1 tale interfacciamento viene realizzato mediante 4 Relè dotati di contatto di scambio (spdt).
Per ampliare il numero di pin disponibili è stato aggiunto un Arduino Uno che ha anche la funzione di alimentare gli altri componenti.



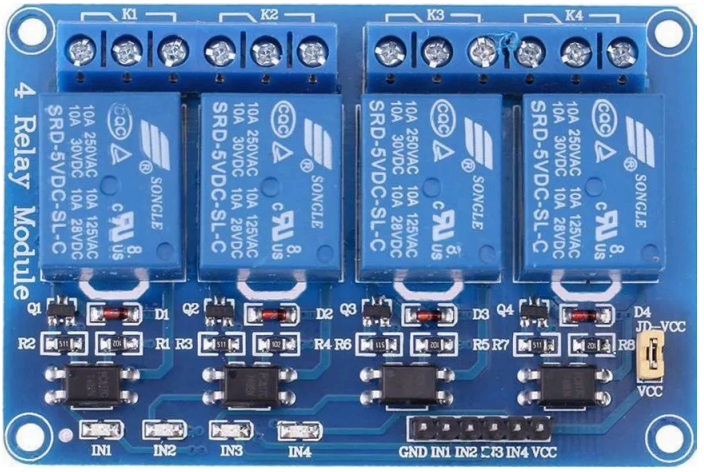
Nel caso dell'antenna **HC-05** collegheremo i **5V*** di Arduino al morsetto VIN ed il Ground di Arduino al morsetto GND.
I morsetti TX ed RX, quest'ultimo mediante partitore di tensione che riduce il segnale a 5V di Arduino ai 3,3V dell'antenna BT, sono collegati ai corrispondenti pin dell'antenna.





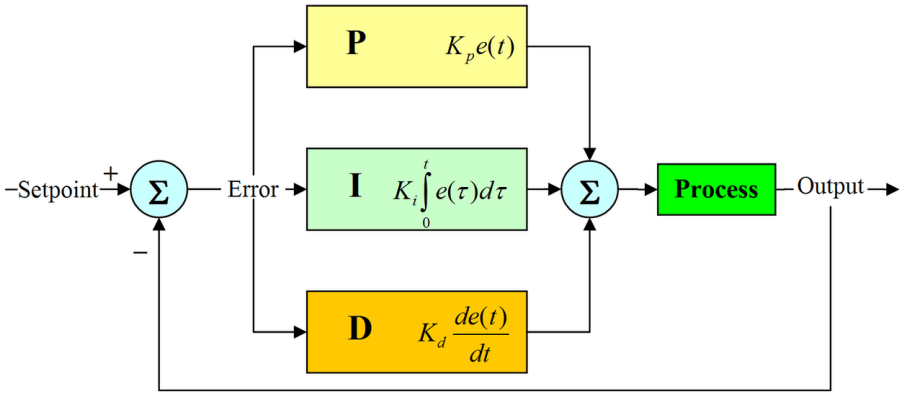
Per la **Wemos D1 mini**, valgono le stesse considerazioni del sistema Bluetooth sia per l'alimentazione che per i morsetti **TX** (**D2** che corrisponde al **5** dell'IDE Arduino) ed **RX** (**D1** che corrisponde al **4** dell'IDE Arduino), in questo caso, abbiamo disponibili il morsetto 3,3V in uscita ed il morsetto A0 che fa capo al corrispondente pin Analogico della scheda così come i morsetti dei pin digitali D5 e D6.

Il modulo dei quattro relè presenta i pin GND e VCC, che saranno alimentati da Arduino, ed i pin IN1, IN2, IN3 e IN4 ognuno dei quali aziona il corrispondente relè quando il relativo pin verrà posto a potenziale 0V (LOW, logica negativa).
Ogni relè ha a bordo un contatto di scambio spdt (single pole double throw) 'pulito' (non sotto tensione, in inglese 'dry contact')



Indice

18. Regolazione PID (Proporzionale, Integrativo, Derivativo)



Ogni volta che un dispositivo deve mantenere costante un determinato valore, detto **valore di Set-Point**, ad esempio una velocità, una temperatura, un livello, una rotta, ecc. serve un 'regolatore', che ha il compito di correggere gli inevitabili errori rispetto al valore prefissato.
Se pensiamo al pilota automatico di una nave che deve mantenere o seguire la rotta, è facile comprendere come venti, correnti e onde siano fonti di errore che il regolatore deve contrastare.

Regolatore Proporzionale
Per il pilota automatico può sembrare sufficiente qualcosa che legga la bussola e giri il timone dal lato giusto e dell'angolo giusto per correggere l'errore.
Avremmo quindi una **correzione Proporzionale al valore dell'errore**.
Un comportamento quindi legato ad una relazione tipo:

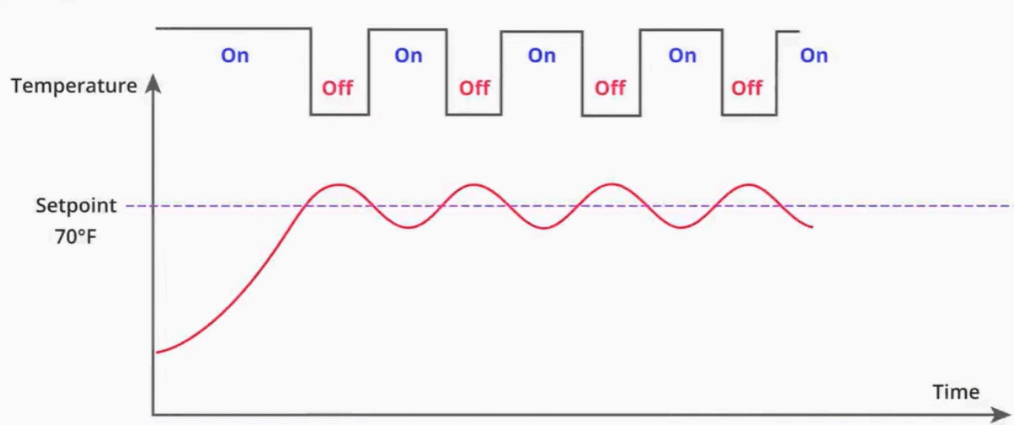
correzione= K*errore

dove **K** è ovviamente da impostare in funzione dei parametri meccanici del timone e della velocità di risposta che vogliamo.
Sfortunatamente, un regolatore che preveda una correzione solo proporzionale, non è sempre ciò che ci serve. Ritornando al nostro esempio, sarebbe facile verificare che la nave continua a virare attorno alla rotta assegnata causando ai passeggeri attacchi di mal di mare oscillando continuamente lungo la rotta ideale.
E' evidente che **il regolatore Proporzionale è attivo solo quando c'è un errore,** e **se si eccede nella intensità della correzione, il sistema diventa instabile ed inizia ad oscillare** attorno al valore giusto con oscillazioni sempre più ampie.
Di seguito la formula

P = Kp * ε

P è il valore della correzione che vogliamo dare al nostro sistema.
ε è il valore dell'errore rispetto al Set-Point e **Kp** è la costante che imposteremo per regolare la risposta rispetto alla natura del sistema.

L'equazione è quella tipica di una retta dove **Kp** rappresenta la pendenza della retta. In pratica, **maggiore sarà Kp maggiore sarà "l'intensità" della correzione e più veloce il ritorno verso il valore di Setpoint e maggiore sarà però il rischio di instabilità.** In sostanza la parte proporzionale è fondamentale per un regolatore in quanto lega la correzione in modo diretto all'errore, **da sola però questa parte non riesce a dare stabilità al sistema e nemmeno a prevenire l'errore.**



animazione

La I (Integrativo) di PID
L'integrale altro non è se non la somma di tutti gli errori nel tempo e in pratica si ottiene semplicemente facendo:

$$I = I + Ki * \varepsilon$$

Questo parametro, "ricordando" gli errori precedenti, aumenta o diminuisce la correzione in funzione di ciò smorzando così le oscillazioni tipiche del sistema Proporzionale. Se prima il sistema ci forniva in uscita una variabile che era:

$$OUT=Kp*\varepsilon$$

e che avevamo chiamato P ora abbiamo:

$$OUT = P + I$$

Con **I** calcolata come nella formula precedente.

L'effetto di rendere più stabile il sistema è dato da una sorta di ritardo con la quale la variabile out viene integrata. Questo ritardo dipende dalla **KI**, costante che dovremo scegliere e bilanciare affinché l'accoppiata **P** e **I** sia efficace.
Questo ritardo ha però un prezzo, **rende l'insieme meno reattivo e veloce ma più fluido.**

La D (Derivativo) di PID

La derivata è una funzione matematica che opera sulla tendenza dell'errore e non sull'errore assoluto. Usando la derivata si ottiene una risposta più rapida da parte del Sistema ed una previsione di quanto stia accadendo.
Questo significa che, ad esempio, riconoscendo che l'errore sta crescendo, il Sistema darà più energia alla correzione di quanto previsto dal solo parametro proporzionale mentre ridurrà la correzione se l'errore sta diminuendo dato che ciò significa che si sta avvicinando al valore di Setpoint . **La derivata** per così dire **accelera o decelera l'intervento del regolatore in modo dinamico seguendo la tendenza dell'errore** e permettendoci di prevedere che, senza altri interventi, alla prossima lettura l'errore sarà minore o maggiore in base appunto alla tendenza.
A livello di formule la derivata si calcola così:

$$D = Kd \bullet \frac{\delta\varepsilon}{\delta t}$$

Qui abbiamo che:

- **D** è il valore della correzione;
- **Kd** è la solita costante che andremo a settare noi in funzione del peso che vogliamo dare alla parte derivativa del nostro regolatore;
- **δε** (delta epsilon) è la differenza tra i due errori;
- **δt** dovendo ripetere il controllo ad intervalli regolari, a questo parametro viene assegnato il valore 1 e diventa quindi ininfluente nel calcolo.

Ad ogni controllo avremo un errore, se fissiamo in **t** il momento dell'ultimo controllo, avremo che il controllo precedente sarà stato effettuato in **t-1**, facendo la differenza tra l'errore precedente **εt** meno quello attuale **εt-1** avremo un valore positivo se l'errore aumenta ed un valore negativo se l'errore diminuisce.
Avremo quindi che D assumerà un valore positivo o negativo in funzione della tendenza e non del valore assoluto dell'errore.

Finalmente il PID
Abbiamo finalmente i nostri tre elementi che costituiscono il nostro PID, non ci resta che combinarli.
Dal punto di vista matematico è molto semplice:

OUT = P + I + D

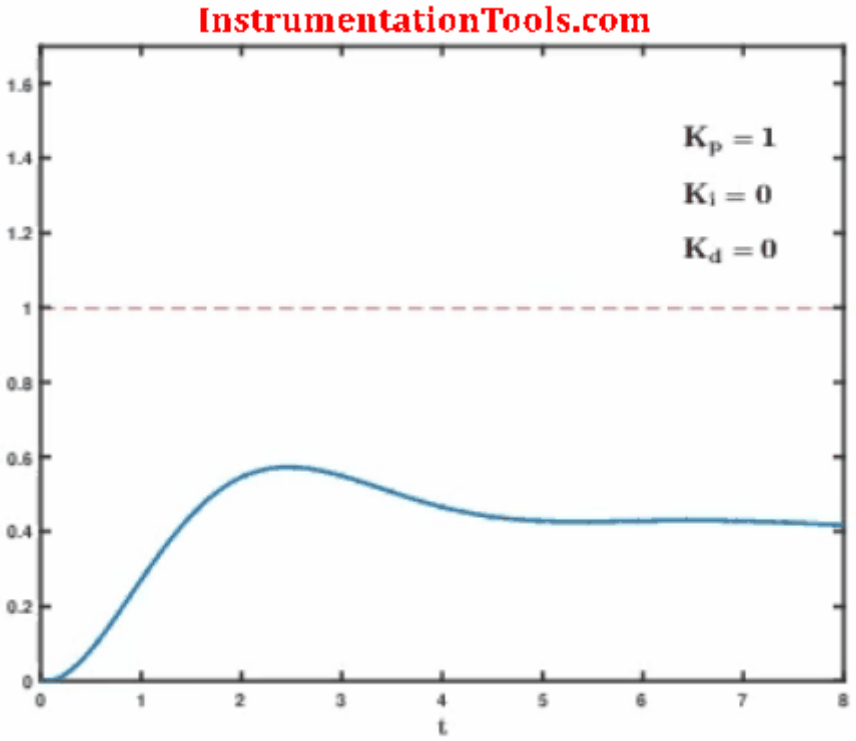
In effetti è tutto quello che serve, i calcoli dei tre parametri li abbiamo fatti prima, li riassumiamo aggiungendo che ovviamente vanno fatti ad ogni ciclo di misura della nostra grandezza da regolare.

$$P=Kp*\varepsilon$$

$$I=I+Ki*\varepsilon$$

$$D = Kd \bullet \frac{\delta\varepsilon}{\delta t}$$

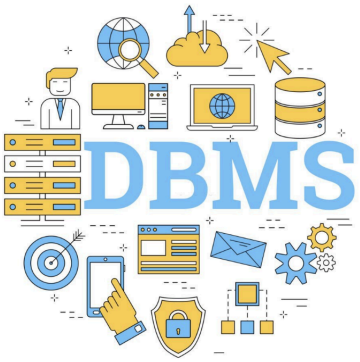
Il difficile in tutto questo è regolare le tre K. Queste determinano infatti il peso dei tre parametri sul totale della nostra regolazione, per fare una cosa scientifica dovremmo avere i parametri meccanici, di risposta, ed altre variabili, non ci resta che il metodo empirico e cioè la sperimentazione e le prove.
Seppur empirico è comunque un metodo e va fatto non per tentativi casuali ma con cognizione di causa e criterio.
IL suggerimento è quello di impostare un parametro per volta ovvero disabilitare le regolazioni **I** e **D** e cercare di ottenere il massimo dell'efficienza dalla sola regolazione proporzionale, aggiungere poi la regolazione **I** e poi la **D** cercando sempre di fare piccole variazioni.



Siccome iniziamo dal parametro proporzionale potrebbe risultare necessario tenere questo un po' più basso rispetto alla prova iniziale e questo per il semplice motivo che comunque andiamo ad aggiungere gli altri parametri e ciò potrebbe portare il tutto oltre le possibilità del regolatore o di risposta del sistema controllato.

Indice

50. DataBase Management System

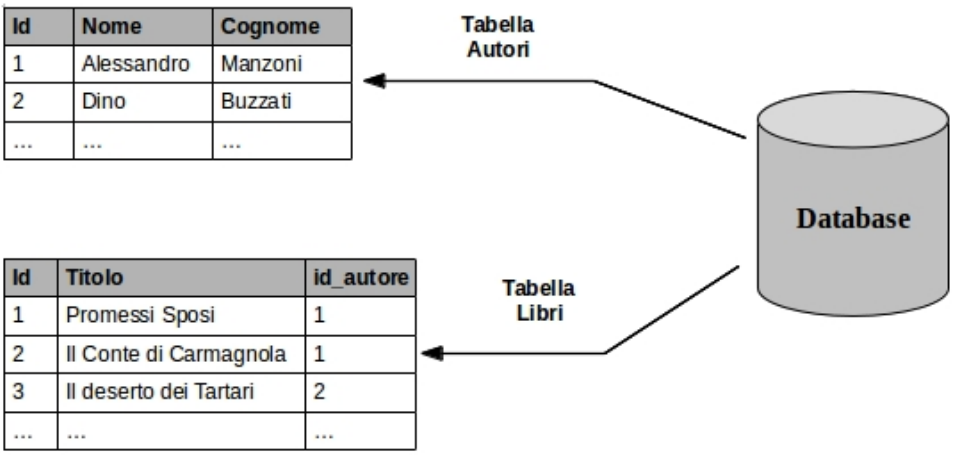


Nel campo dell'IoT è indispensabile avere le competenze per gestire i dati dei Sistemi controllati attraverso la rete Internet e renderli raggiungibili da qualunque host. Vi sono diverse opzioni ma, quella più efficiente e dinamica, è sicuramente quella dei **DBMS** ospitati su server ai quali poter accedere.
Esistono diverse modalità di organizzare un database, nel nostro caso, il più adatto è il **Database Relazionale**.

Un **Database Relazionale** è una raccolta di dati (**Record**) suddivisi in più aree di archiviazione separate (**Tabelle**) che un software dedicato, il **D**ata **B**ase **M**anagement **S**ystem (DBMS), consente di memorizzare, organizzare, cancellare e recuperare.

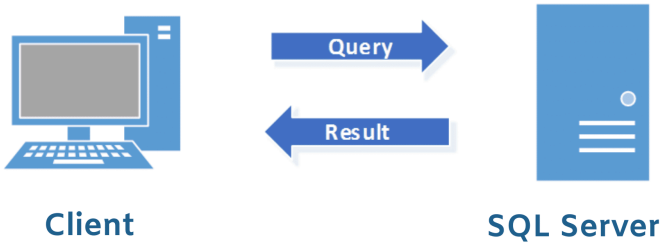
Il software **DBMS** open source più diffuso è **MySQL**.
SQL (**Structured Query Language**) è invece il linguaggio di programmazione per i Database relazionali, **SQL**, letteralmente, significa **linguaggio di interrogazione strutturato**.

Le strutture del database sono le **Tabelle** la cui corretta progettazione, assegnazione univoca di **ID** (**Chiave Primaria**) ai dati (**Record**) , ne consente poi la gestione senza errori.



Le **Query** (Interrogazioni) sono le azioni che consentono di **accedere** al database per **memorizzare**, **organizzare**, **cancellare o recuperare i dati**.

MySQL utilizza il modello **Client-Server**

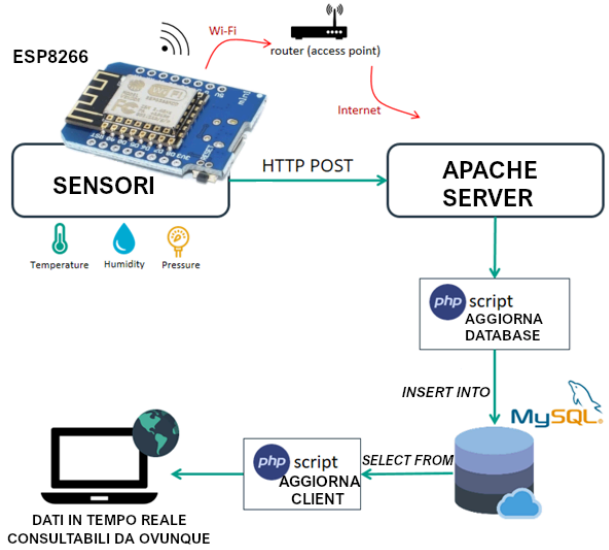


Per comprendere il funzionamento dei DBMS è necessario distinguere tra **sito web statico** e **sito web dinamico**.

- **Sito web statico:** viene creato per restituire, su richiesta del Client, sempre la stessa pagina web contenente quella determinata informazione richiesta (programmazione HTML). **Se la pagina deve essere modificata ciò andrà fatto manualmente da chi gestisce il server.**

- **Sito web dinamico:** in questo caso il client ha la possibilità di modificare la pagina web, o alcune parti, grazie all'invio del codice scritto con il linguaggio PHP che sul lato server viene interpretato ed eseguito generando la pagina che verrà inviata e visualizzata dal client. **In questo caso si parla di programmazione lato server da parte del client stesso.**

Quindi, sul server che dovrà gestire il database, dovrà esserci un interprete PHP che si occuperà di manipolarlo e di restituire ciò che il Client ha richiesto mediante l'invio della Query sotto forma di codice PHP.



Il Sistema che verrà implementato in laboratorio prevede l'uso della scheda Wemos (ESP8266) abbinata ad un sensore DHT11, che avrà il compito di inviare al Web Server i dati di temperatura ed umidità da inserire nel DBMS (**INSERT INTO**) mediante l'invocazione di un apposito file **PHP**, all'interno del quale, vi sarà anche una porzione di codice HTML che consentirà a qualunque Client di estrarre i dati (**SELECT FROM**) e visualizzarli.

Per consentire il funzionamento del Sistema DBMS è necessario:

- **un *web server*** (Apache) **che lo ospiti;**
- **un *interprete PHP*** (phpMyAdmin)**;**
- **ed il *database* stesso** (MySQL),

per fortuna, esistono delle suite (gruppi di software) che, con una sola installazione, predispongono tutto il necessario. Nel nostro caso useremo XAMPP (si pronuncia Shamp).

Sintassi

Per interagire con il database è necessario utilizzare le relative parole chiave per richiamare i record che si vogliono manipolare.
Le più comuni sono:

SELECT: Serve per leggere i dati. (Esempio: *"Prendi gli utenti dal database"*).

FROM: Indica da quale tabella prendere i dati.

WHERE: La condizione di filtro. (Esempio: *"Solo dove l'ID è uguale a 5"*).

INSERT INTO: Serve per aggiungere una nuova riga.

VALUES: Specifica i valori effettivi da inserire.

UPDATE: Serve per modificare dati già esistenti.

DELETE: Serve per eliminare una riga.

Se ad esempio dobbiamo recuperare i records da intere colonne da una determinata tabella

```
SELECT nomecolonna1, nomecolonna2, … FROM nometabella;
```

In questo caso vengono recuperati i records che rispondo ad una determinata condizione contenute in determinate colonne

```
SELECT nomecolonna1, nomecolonna2, … FROM nometabella
WHERE condizione;
```

In questo caso vengono inseriti i valori specificati in VALUES in una specifica tabella e nelle rispettive colonne

```
INSERT INTO nometabella (nomecolonna1, nomecolonna2, …)
VALUES (valore1, valore2,...);
```

Questo comando aggiorna i valori di una determinata tabella inserendo il valore1 nella colonna 1, il valore2 nella colonna2 corrispondenti ad una condizione

```
UPDATE nometabella SET nomecolonna1 = valore1, nomecolonna2 =
valore2, … WHERE condizione;
```

Con DELETE possiamo rimuovere una riga da una tabella che risponde ad una condizione

```
DELETE FROM nometabella WHERE condizione;
```
o cancellare tutti i records di una data tabella
```
DELETE FROM nometabella;
```
mentre, per rimuovere la tabella dal database, si usa il comando
```
DROP TABLE nometabella;
```

Abbiamo ora gli elementi per comprendere il funzionamento del Sistema che implementeremo

Vediamo quali sono i passaggi che consentono il funzionamento del **D**ata **B**ase **M**anagement **S**ystem:

- la scheda ESP8266 si connette ad Internet ed invia la richiesta HTTP di connessione al server;
- il server accetta la connessione e si mette in attesa dei dati;
- la scheda ESP8266 invia i dati da inserire nel database posto sul server ed ‘invoca’ l’esecuzione di un file PHP che inserisce materialmente i dati all’interno del database utilizzando il linguaggio SQL (INSERT INTO);
- una pagina HTML, posta sul server, conterrà una porzione di linguaggio PHP che preleva dal database i dati mediante il linguaggio SQL (SELECT FROM), ed andrà ad aggiornare la pagina stessa;

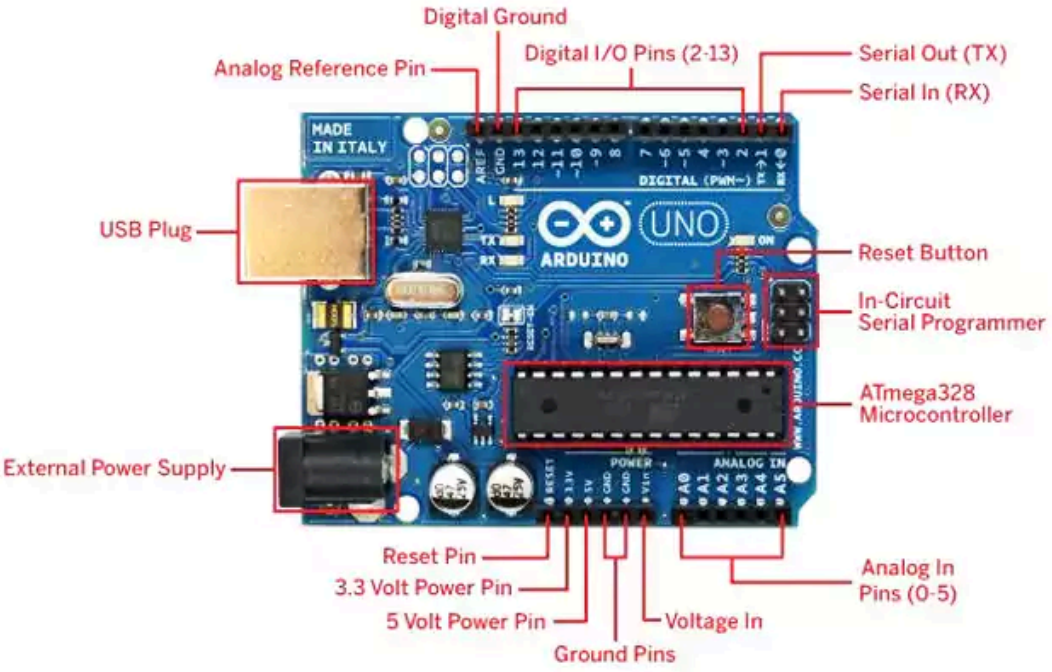
a questo punto, chiunque abbia il link di tale pagina HTML, potrà visualizzare i dati aggiornati da qualunque dispositivo connesso ad Internet .

Indice

(´•⁔•`)
٩(●◡●)۶

CLIL (Content and Language Integrated Learning)

Arduino Microcontroller



History. Arduino was born at the 'Ivrea Interaction Design Institute' (Italy) as an easy tool for fast prototyping to help students without a background in electronics and programming.

What is Arduino? Arduino is an open-source electronics platform. Open-source means that anyone can create new software, modify and distribute it without copyright, this has made Arduino a global phenomenon. Arduino is able to read inputs like buttons, sensors, limit switches or messages coming via email, whatsapp or directly from the web and, after processing the loaded software, turn on a led, start an engine or post something on the web. Anything can be implemented with this board!

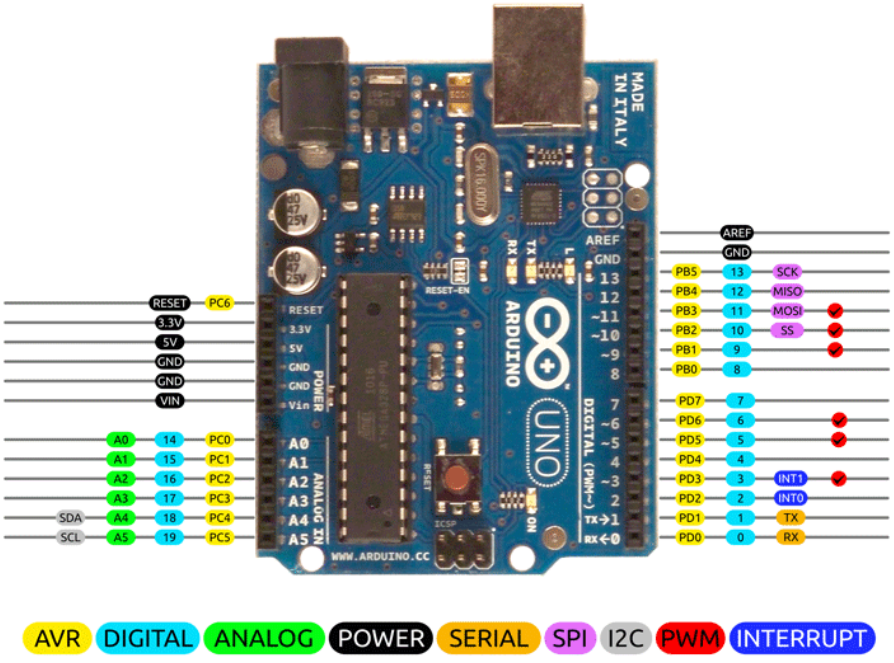
How Arduino Works. Microcontrollers are integrated circuits where instructions can be recorded, which you write with the programming language that you can use in the Arduino **IDE** environment. These instructions allow you to create programs that interact with the circuitry on the board.

The programming language is called **Wiring**, derived from C and C++.

Advantages of Arduino

- **Inexpensive** Arduino boards are relatively inexpensive compared to other microcontroller platforms. Arduino modules cost less than 10€.
- **Cross-platform** The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple** The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.

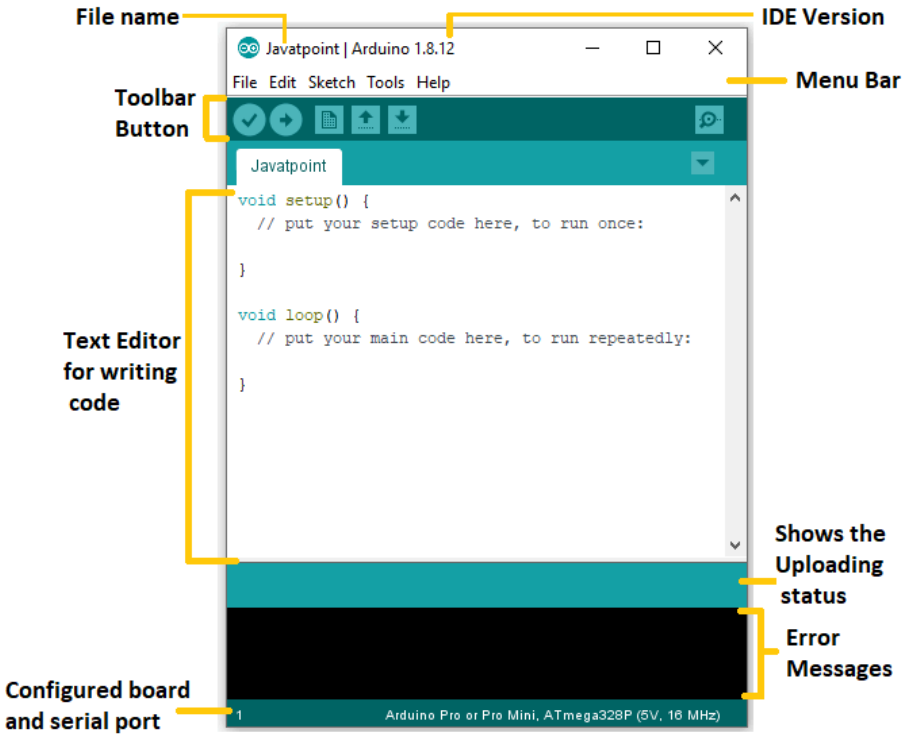
What on board? It has 14 digital pins which can be programmed as both input and output (six of them can be used for PWM driving) and 6 analog inputs. What is the difference between digital or analog pins? The digital ones can only assume the HIGH logic state (5V) or the LOW logic state (0V) while, the analog ones, all the values between 0V and 5V



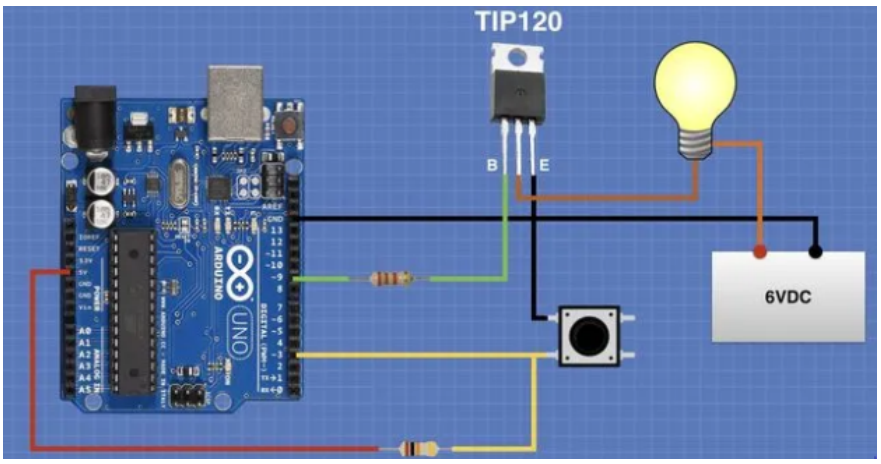
The IDE (Integrated Development Environment)

The IDE is a simple text editor, the first programming phase consists in defining the variables that will be used to obtain the desired result and to establish the inputs and outputs used. The second step is to declare,

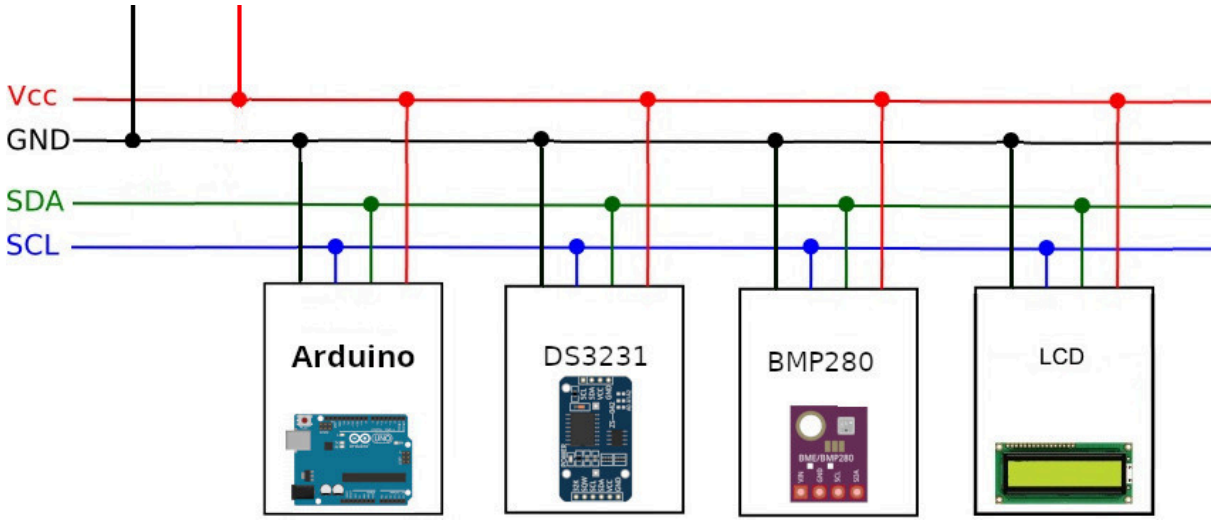
within the main void setup() structure, which pins are inputs and which outputs. This part of the program will be read only on startup of the microcontroller. The third phase consists in writing the part of the program that will be read cyclically and is inside the main void loop() structure.



Interfacing Arduino Arduino is a low power electronic device, in fact the maximum current that can be supplied by its pins is 20mA, and, for this reason, when it has to drive loads that require higher currents, it is necessary to interpose a buffer device between the microcontroller and the load. If the load requires only an On/Off command, the buffer device can be a relay, while, if the load requires a Pulse Wide Modulation drive, it is essential to use the transistors.



Supported Communication Protocols Arduino supports all the most popular communication protocols such as UART, I2C, SPI and is able to interface using BLE (Bluetooth Low Energy) technology. For these reasons it has become the most widespread board in the field of civil and industrial Domotics. By adding a special Ethernet shield, the microcontroller can connect to the network allowing it to develop systems with Internet of Things technology.



Importance of Arduino community There is another important factor in the success of Arduino, it is the community that supports all this development, shares knowledge, develops libraries to facilitate the use of Arduino and publishes its projects so that they can be replicated, improved or be the basis for another related project.