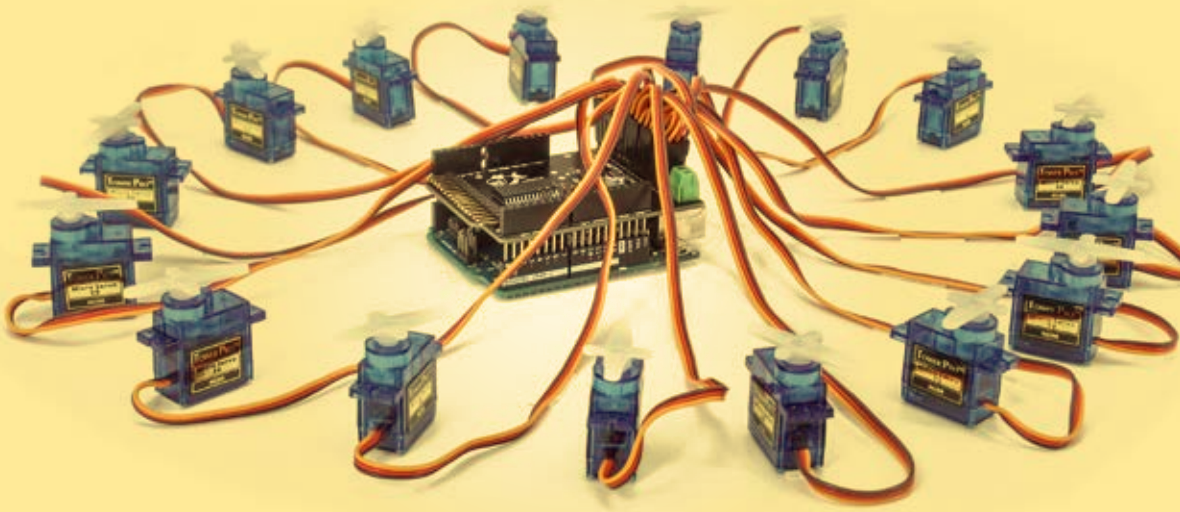
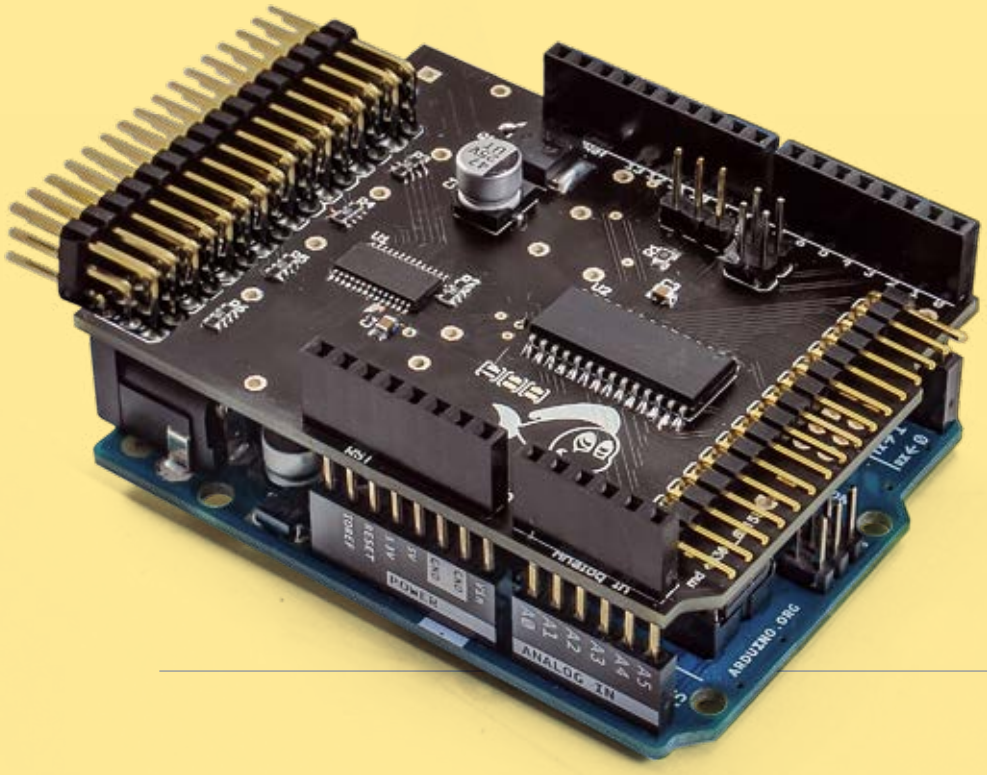
Shield che rende disponibili 16 I/O digitali o altrettanti PWM impiegando un solo canale I²C-Bus di Arduino. Ideale per gestire LED multicolore o servomotori.



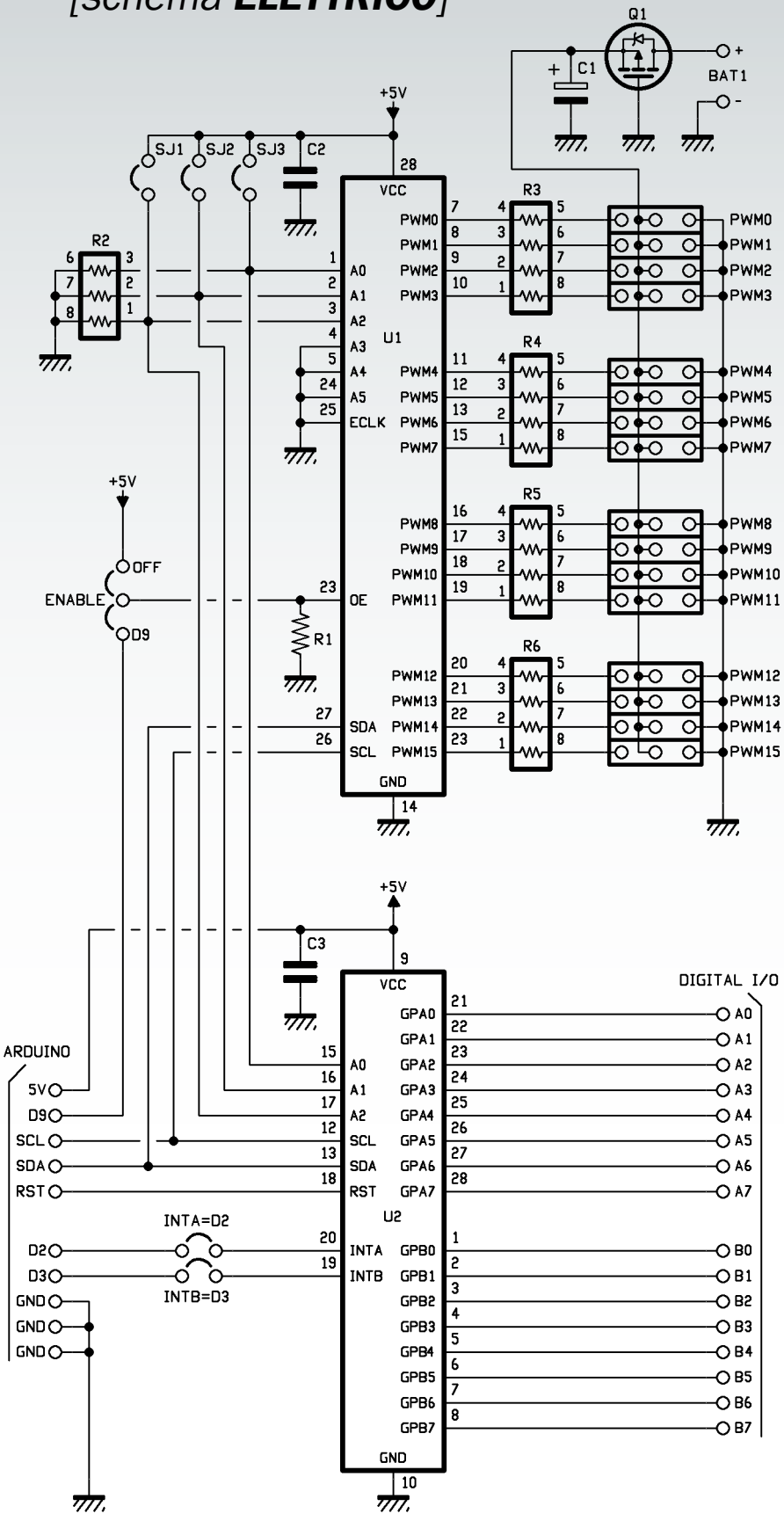
OCTOPUS, ESPANSIONE "TENTACOLARE" PER ARDUINO/FISHINO

di MASSIMO DEL FEDELE



Per quanto pratiche e capaci di realizzare innumerevoli applicazioni, le schede Arduino e compatibili hanno due limiti: la memoria di programma relativamente ridotta e la ridotta quantità di uscite disponibili, specie di I/O cui si può assegnare un segnale PWM. Per esempio, una Arduino/Fishino UNO dispone di sole sei uscite PWM e, a meno di non generare i relativi segnali via software (con notevole impegno del processore), permette il pilotaggio di un solo driver e quindi un solo LED RGBW di potenza, o in alternativa di sei led monocromatici.

[schema **ELETTRICO**]



Lo stesso limite emerge quando si vogliono pilotare più di 6 servomotori con le stesse schede; la realizzazione di un robot tipo "hexapod", che richiede ben 12 servi, risulta problematica, se non impossibile. Anche gli ingressi e le uscite digitali sono limitati; sempre parlando delle schede Arduino, abbiamo un totale di 13 I/O digitali e 6 ingressi analogici, utilizzabili anch'essi in digitale; sembrerebbero anche abbondanti, se non fosse che molti di questi vengono utilizzati per le periferiche a bordo o dagli shield di espansione. In pratica, realizzando un progetto con uno shield Ethernet/WiFi, una memoria SD e che necessita dell' uscita seriale e di qualche ingresso analogico, restano a disposizione solo sei I/O digitali che sono spesso insufficienti per progetti di media complessità.

Per tutti questi motivi abbiamo pensato di progettare uno shield di espansione, compatibile con le varie schede Arduino e con la nostra Fishino UNO (e la Fishino Mega che vi presenteremo a breve) che, senza praticamente impegnare risorse hardware, permette di avere a disposizione ben 16 uscite in PWM e 16 ingressi/uscite digitali aggiuntivi. Non solo, le schede sono sovrapponibili fino ad un massimo di 8, consentendo di gestire con Arduino fino a 128 I/O digitali e 128 uscite PWM aggiuntive; il tutto reso completamente trasparente all'utilizzatore tramite una libreria realizzata ad-hoc.

SCHEMA ELETTRICO

Lo shield qui proposto è composto da tre blocchi:

- espansione I/O digitali;
- espansione PWM;
- circuito di protezione dall'inversione di polarità sull'alimentazione esterna.

ESPANSIONE I/O DIGITALI

Questa sezione si basa sul ben noto integrato MCP23017, nell'abituale versione controllata tramite interfaccia I²C-Bus, quindi tramite due soli fili, il clock (SCL) ed il filo di data (SDA).
Esiste anche una versione controllabile tramite interfaccia SPI, più veloce ma che impegna quattro linee della CPU e che quindi non abbiamo utilizzato.
Come tutti i dispositivi I²C, prevede un indirizzamento tramite una sequenza di bit inviati sulle due linee, che attivano l'integrato solo se l'indirizzo corrisponde.
Nel caso dell' MCP23017 l'indirizzo è composto da una parte fissa, indicata come MCP23017_I2C_BASE_ADDRESS nella nostra libreria, pari al numero esadecimale 0x20, i cui ultimi TRE bit sono però impostabili mediante tre pin esterni (A0, A1 ed A2) ai valori binari da 000 (0 decimale) a 111 (7 decimale), permettendo quindi di variare l'indirizzo completo tra 0x20 e 0x27 inclusi. Questo indirizzamento consente il collegamento allo stesso I²C-Bus di otto MCP23017, a patto che per ogni dispositivo venga scelto un indirizzo differente tra gli otto disponibili.
L'integrato è dotato di due porte I/O digitali ad 8 bit (GPA0÷GPA7 e GPB0÷GPB7), per un totale di 16 linee complessive, impostabili singolarmente come ingresso o uscita, con o senza resistenze di pullup interne, tramite la libreria software di cui parleremo in seguito.
Per ogni canale ad 8 bit il chip è in grado di generare un segnale di interrupt, sulle uscite **INTA** e **INTB**, al verificarsi di ogni cambio di stato degli ingressi, sia separatamente per ogni gruppo di 8 porte (permettendo quindi di distinguere gli interrupt a livello hardware tra i 2 canali) oppure congiuntamente, un solo segnale di interrupt per tutti gli ingressi.
La generazione degli interrupt al cambio di stato degli ingressi è un grande vantaggio quando è necessario ottenere risposte immediate al verificarsi di eventi esterni, e consente di compensare la limitazione delle schede come Arduino/Fishino UNO, dotate di due soli ingressi di interrupt.
Le uscite **INTA** ed **INTB** sono collegabili tramite due ponticelli distinti agli indirizzi, rispettivamente 2 e 3 di Arduino, gli unici in grado di rilevare un cambio di stato; tramite software sarà poi possibile scegliere se utilizzare entrambi, solo uno o nessuno.
Completa la descrizione di questa sezione il condensatore di disaccoppiamento C3, situato in prossimità del chip per evitare che disturbi sulle linee di alimentazione ne influenzino il funzionamento, e la linea RST che, connessa al reset di Arduino, consente di ripristinare il funzionamento all'accensione e/o alla pressione del tasto **RESET**.

SEZIONE PWM

Questa sezione si basa sul meno noto ma non per questo meno versatile PCA9685, un integrato in grado di generare 16 segnali PWM quasi indipendenti (vedremo nella descrizione delle librerie che la frequenza deve essere la stessa per tutte le uscite).
Questo integrato consente un indirizzamento di ben sei bit, tramite gli ingressi A0..A5, che permetterebbe la sovrapposizione di 64 schede diverse; essendo però il limite imposto dall' MCP di 8 schede (e risultando 64 schede connesse ad un singolo bus decisamente troppe) abbiamo sfruttato i soli bits A0÷A2, ottenendo così anche qui la possibilità di connettere otto dispositivi diversi, collegando i rimanenti pins di indirizzamento direttamente a massa. La frequenza del PWM è impostabile tramite un registro interno (prescaler) con la seguente formula:

$$f_{PWM} = \frac{f_{clock}}{4096 \cdot (prescale + 1)}$$

L'integrato è in grado sia di generare la frequenza di clock internamente, tramite un oscillatore da 25 MHz che consente di ottenere frequenze PWM da 24 a 1.526 Hz, sia di operare con un clock esterno massimo di 50 MHz, che però non abbiamo utilizzato nella nostra scheda.
A differenza dell' MCP23017, questo integrato non dispone di un ingresso di reset; la relativa funzionalità va quindi realizzata tramite software (è previsto un indirizzo I²C speciale per il reset) oppure semplicemente ripristinando via software il contenuto dei registi (cosa da noi effettuata tramite la libreria).
È invece disponibile un ingresso OE in grado di disabilitare tutte le uscite contemporaneamente, che abbiamo ritenuto opportuno portare all'esterno dando la possibilità, tramite un ponticello, di collegarlo all'I/O D9 di arduino nel caso se ne presenti la necessità.
Una possibilità interessante di questo integrato è di poter operare sulle uscite sia in modalità totem-pole, con una coppia di MOSFET complementari, in grado di fornire una corrente sia positiva (source) che negativa (sink), con valori massimi rispettivamente di 10 mA e 25 mA a 5 volt, che in modalità open-drain (o collettore aperto, in italiano), in grado di fornire solo una corrente negativa fino a 25 mA, modalità questa utile in alcuni casi.

Dalle caratteristiche (e da altre che qui tralasciamo per brevità) si evince che l'integrato è nato per pilotare in PWM dei LED, connessi direttamente alle uscite tramite resistenze; ciò non toglie che, come vedremo, è possibile utilizzarlo per innumerevoli altre applicazioni. Sullo schema potrete notare, infatti, una serie di resistenze in serie alle uscite (inserite per proteggere le medesime da sovracorrenti) che portano ad un connettore a tre pin per uscita sui quali, oltre al segnale PWM vengono riportate la massa ed una tensione positiva della quale parleremo più avanti. Questo connettore, realizzato ad angolo di 90°, permette di connettere direttamente alle uscite 16 servocontrolli senza impedire la sovrapposizione di ulteriori schede Octopus.

PROTEZIONE DALL'INVERSIONE DI POLARITÀ

Come abbiamo accennato nelle pagine precedenti, la scheda è stata pensata per poter pilotare direttamente dei servocomandi, collegandoli al connettore angolato. Questi componenti assorbono una notevole corrente di picco ed è impensabile poterli alimentare con i 5 volt di Arduino; è stato quindi previsto un connettore per fornire alla scheda una tensione esterna con corrente adeguata. Per evitare che un'inversione di polarità distrugga scheda e servocontrolli, abbiamo inserito in serie all'alimentazione il Mosfet Q1, a canale P, in grado di lasciar passare la corrente solo nella direzione corretta.
Perchè un mosfet e non un semplice diodo? Il motivo è che il diodo ha una caduta di tensione fissa ai suoi capi (leggermente variabile con la corrente) che, se risulta ininfluente per tensioni medio-elevate e basse correnti, causa perdite e surriscaldamenti notevoli nel caso opposto di elevate correnti e basse tensioni.
Ad esempio, scegliendo un diodo Schottky con una caduta di tensione intorno a 0,5 V e facendo scorrere una corrente di 4 ampere, esso si trova a dover dissipare ben 2 W: e un diodo da 2 watt ha dimensioni non trascurabili e scalda molto.
Per contro, il MOSFET selezionato ha una resistenza di conduzione particolarmente bassa, pari a 51 mOhm (milliohm) alla stessa corrente di drain. La dissipazione diviene quindi, sempre nelle condizioni di cui sopra, pari a:

$$P = I^2 \cdot R = 4Ampere \cdot 0.051ohm = 0.816Watt$$

Una potenza quindi decisamente inferiore, con conseguente minor surriscaldamento e maggior durata di eventuali batterie. Il condensatore elettrolitico C1 filtra la tensione in ingresso da eventuali disturbi. Il valore scelto, 100 µF, è sufficiente fino a correnti di qualche ampere; nel caso di correnti più elevate e/o di assorbimenti molto impulsivi consigliamo di inserire un condensatore aggiuntivo di capacità elevata esternamente sulla linea di alimentazione.

REALIZZAZIONE PRATICA

Lo shield è realizzato su un circuito stampato a doppia faccia con fori metallizzati e fa uso di componenti in SMD; trattandosi di un montaggio superficiale, occorre la consueta attenzione, in particolar modo per i componenti più piccoli e quelli con i pin molto ravvicinati.
La maggiore attenzione è richiesta dall'integrato PCA9685, fornito in package TSSOP28 a 28 pin con spaziatura di soli 0,65 mm, e dalle le reti resistive R3÷R6, che hanno pin piccoli e molto ravvicinati; attenzione anche al condensatore elettrolitico C1 che, pur non essendo di dimensioni particolarmente ridotte, ha i pin sotto il corpo che sporgono solo pochissimo dal medesimo.
Con un minimo di pratica, una lente d'ingrandimento e una pinzetta, un buon saldatore a punta fine (da non più di 20 W di potenza) ed un ottimo flussante (che aiuta a fondere e a distribuire lo stagno su piedini e piazzole) risulta comunque abbastanza semplice montare il circuito; nel caso coli troppa lega saldante e faccia cortocircuiti tra i pin degli integrati è possibile rimediare cospargendoli di flussante, pulendo molto bene la punta del saldatore e scaldandoli in modo che lo stagno in eccesso risalga per sulla punta liberandoli. Solitamente con 2-3 passaggi anche i cortocircuiti più ostici si risolvono in questo modo.
Occorre fare solo attenzione alla temperatura del chip, evitando di soffermarsi troppo tempo col saldatore e, nel caso occorra ripetere il procedimento, lasciando raffreddare per qualche decina di secondi il componente.
Chi non se la sentirà di realizzare lo shield potrà acquistarlo (presso la Futura Elettronica, *www.futurashop.it*) in versione montata e collaudata, con i connettori laterali e frontali non montati; ciò per offrire la possibilità di cambiare tipologia di connettore, per avere i contatti delle uscite (tipo servo, quindi con PWM, +5V e massa in linea) accessibili in verticale o frontalmente.

LA LIBRERIA

Come già accennato, per questa scheda abbiamo realizzato un'apposita libreria software, denominata **Octopus**, dotata di alcune particolarità che ne rendono semplicissimo l'utilizzo.

La prima particolarità interessante della libreria si può notare dalle linee dell'include file (**Octopus.h**):

```
#define Octopus __octopus()
OctopusClass &__octopus();
```

e dalle linee del file sorgente (**Octopus.cpp**):

```
OctopusClass &__octopus()
{
        static OctopusClass octo;
        return octo;
}
```

Questa modalità apparentemente strana di utilizzo della variabile **Octopus** permette di ovviare ad uno dei problemi del C++, ovvero dell'inizializzazione delle variabili globali che non avviene in un ordine predeterminato ma è casuale, e viene effettuata al caricamento dello sketch. Nel nostro caso, dovendo inizializzare l'interfaccia I²C tramite le istruzioni:

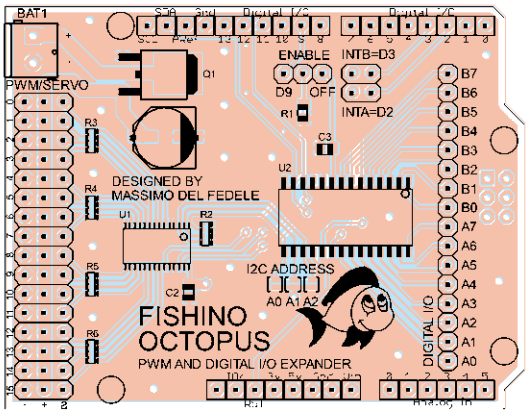
```
Wire.begin();
Wire.setClock(400000);
```

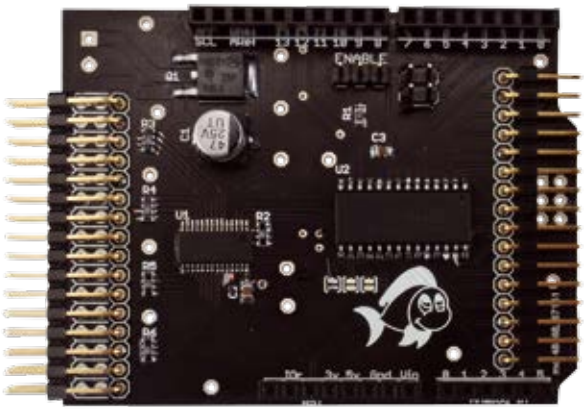
prima dell'utilizzo della libreria, risulta impossibile creare la variabile statica **Octopus** al momento del caricamento del programma, visto che l' interfaccia **Wire** in quel momento non è ancora stata inizializzata. La soluzione prescelta permette per contro di ottenere la creazione della variabile al primo utilizzo della medesima, e quindi dopo aver inizializzato correttamente l'interfaccia I²C-Bus; questo ci ha permesso di realizzare un codice che, senza nessuna riga di programma aggiuntiva, è in grado di contare ed inizializzare correttamente tutti gli shield Octopus connessi e di numerarne automaticamente le uscite in ordine di indirizzo I²C. Ad esempio, se alla nostra Arduino applichiamo due shield, avremo a disposizione 32 I/O digitali, numerati da 0 a 31, e 32 PWM, numerati anch'essi da 0 a 31.
La libreria fornisce due funzioni che permettono di conoscere il numero di schede connesse ed il numero di I/O e PWM disponibili:

```
// return number of boards found
uint8_t getNumBoards(void) const;

// return number of available I/O
uint8_t getNumIO(void) const;
```

Come detto in precedenza, la frequenza del PWM è unica per ogni scheda, quindi per ogni gruppo di 16 uscite PWM; è impostabile tramite le due funzioni seguenti, la prima scheda per scheda e





Elenco Componenti:

R1: 10 kohm (0805)
R2: Array 4x10 kohm (0603)
R3÷R6: Array 4x220 ohm (0603)
C1: 100 µF 16 VL elettrolitico (Ø6mm)
C2. 1 µF ceramico (0805)
C3: 1 µF ceramico (0805)
Q1: NTD25P03L (DPACK)
U1: PCA9685PW
U2: MCP23017-E/SO

Varie:
- Strip M/F 6 vie
- Strip M/F 8 vie (2 pz.)
- Strip M/F 10 vie
- Strip M/F 2x3 vie
- Strip Maschio 3x16 vie 90°
- Strip Maschio 16 vie 90°
- Jumper (3 pz.)
- Strip Maschio 3 vie
- Strip Maschio 2 vie (2 pz.)
- Morsetto 2 poli passo 3mm
- Circuito stampato S1226

Listato 1

```
// Progetto : OctopusTest
#include <Wire.h>
#include <Octopus.h>

uint16_t i = 1;
uint16_t valueTable[16];

// Codice di inizializzazionecode
void setup(void)
{
        Serial.begin(115200);
        Serial.println("OCTOPUS TEST APP");

        Wire.begin();
        Wire.setClock(400000);

        delay(500);
        Serial.print("Found #");
        Serial.print((int)Octopus.getNumBoards());
        Serial.println(" OCTOPUS boards");
        delay(500);

        Octopus.setPWMFreq(1526);

        for(int k = 0; k < 16; k++)
                valueTable[k] = (- (double)k * k / 64 +
0.25 * k) * 4096;
}

// ciclo infinito
void loop(void)
{
        for(uint8_t p = 0; p < 16; p++)
                Octopus.analogWrite(p, valueTable[(i +
p) % 16]);
        i++;
        delay(50);
}
```

la seconda per tutte le schede connesse in un solo comando:

```
// set pwm frequency for a single connected board
// valid values 24 Hz...1526 Hz
void setPWMFreq(uint8_t board, uint16_t freq);

// set pwm frequency for ALL connected boards
void setPWMFreq(uint16_t freq);
```

Nella prima occorre indicare il numero di scheda (che va da 0 a *Octopus.getNumBoards()*) e la frequenza di PWM, da 24 Hz a 1.526 Hz; nella seconda è sufficiente indicare la frequenza e tutte le schede verranno impostate su quella. All'accensione, la frequenza preimpostata è di 200 Hz, adatta ai servocontrolli ma anche ai LED. Il valore delle uscite PWM è impostabile, analogamente alle librerie di Arduino, tramite la funzione seguente:

```
// pwm output
void analogWrite(uint8_t port, uint16_t val, bool in-
vert = false);
```

Ad esempio, per impostare l'uscita **30** (la terzultima della seconda scheda connessa) al **50%** del valore massimo, occorre scrivere

```
Octopus.analogWrite(30, 2048);
```

Il terzo parametro opzionale, **invert**, è utile nel caso si connettano dei led in uscita sfruttando le uscite in modalità open collector e collegandone gli anodi al positivo; in questo caso si ha bisogno di un'uscita inversa (più tempo resta alta, meno corrente scorre nel led) ed è quindi necessario impostare il parametro a true per ottenere una luminosità crescente con il valore **val**.
Come avrete sicuramente notato, il valore del PWM a differenza delle uscite di Arduino è a 16 bit, dei quali ne vengono utilizzati 12, permettendo quindi una variazione di intensità a 4.096 livelli al posto dei 256 di Arduino; questo consente, per esempio, un controllo molto più graduale dell'intensità della luce emessa da LED connessi alla scheda.
Per contro occorre ricordarsi di questo quando si imposta il valore, visto che un numero pari a 255, che su Arduino corrisponde all'intensità massima, qui corrisponde ad un valore piuttosto basso (255/4.096 del massimo).
Per la gestione degli I/O digitali, la libreria fornisce le seguenti funzioni, praticamente identiche a quelle delle librerie standard, se non per il fatto di poter utilizzare un numero anche molto grande di porta:

```
// digital I/O
void pinMode(uint8_t port, uint8_t mode);
bool digitalRead(uint8_t port);
void digitalWrite(uint8_t port, bool value);

// read/write all digital pins of given board at once
uint16_t digitalReadAll(uint8_t board);
void digitalWriteAll(uint8_t board, uint16_t val);
```

Le ultime due funzioni permettono di leggere e scrivere tutti i ports digitali di una scheda in un colpo solo, tramite una variabile a 16 bit; questo risulta molto comodo nel caso si abbia bisogno di modificare o leggere molto rapidamente le porte digitali.
Al momento della pubblicazione di questo articolo la libreria (scaricabile dal nostro sito *www.elettronicain.it* insieme ai file del progetto) è in fase di completamento, e le funzioni degli interrupt e della modifica di modalità delle uscite PWM (open drain/totem pole) devono essere completate.

UNO SKETCH DI PROVA

Per concludere, presentiamo un semplice sketch che permette di visualizzare una "coda" luminosa utiliz-

zando 16 LED connessi alle uscite PWM (**Listato 1**). Il codice inizializza l'interfaccia seriale, stampa un messaggio, inizializza l'I²C, stampa il numero di schede rilevate e, utilizzando la prima di esse (uscite PWM da 0 a 15) crea una sorta di 'serpente' luminoso sfruttando 16 led.
I valori di luminosità del 'serpente' sono preventivamente calcolati nella setup ed inseriti in una tabella contenente 16 valori; a seconda del punto di partenza della tabella (variabile 'i' nel loop) la 'testa del serpente' si trova in un punto differente, creando quindi l'effetto visivo voluto.
Come si nota, a parte il dover inserire '**Octopus.**' davanti ai comandi **analogWrite()** l'utilizzo è praticamente identico alla libreria n ativa di **Arduino**.
L'unica cosa degna di nota è il calcolo dei valori di luminosità, effettuato qui con un polinomio del secondo ordine in modo da creare un effetto di onda; sono possibili altri metodi con funzioni trigonometriche o semplicemente con una variazione lineare su cui potrete sperimentare; ad esempio:

```
for(int k = 0; k < 16; k++)
        sinTable[k] = 4096 * sin(M_PI / 16 * k);
```

per un andamento sinusoidale, oppure

```
for(int k = 0; k < 16; k++)
        if(k <= 8)
                sinTable[k] = 4096 * (double)k / 8;
        else
                sinTable[k] = 4096 * (double)(15 - k)
/ 8;
```

per un andamento bilineare.
Concludiamo qui la descrizione della nostra scheda **Octopus**; potete ora sperimentare i diversi effetti luminosi utilizzando ad esempio il driver Colibrì presentato nel numero scorso di Elettronica In. ■

per il MATERIALE

Lo shield Octopus (cod. OCTOPUS) è disponibile presso Futura Elettronica nella versione con la componentistica SMD premontata a 24,00 Euro (con i connettori laterali e frontali da montare). Software e sketch sono scaricabili gratuitamente sul sito *www.elettronicain.it*. Il prezzo si intende IVA compresa.

Il materiale va richiesto a:
Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287 - www.futurashop.it