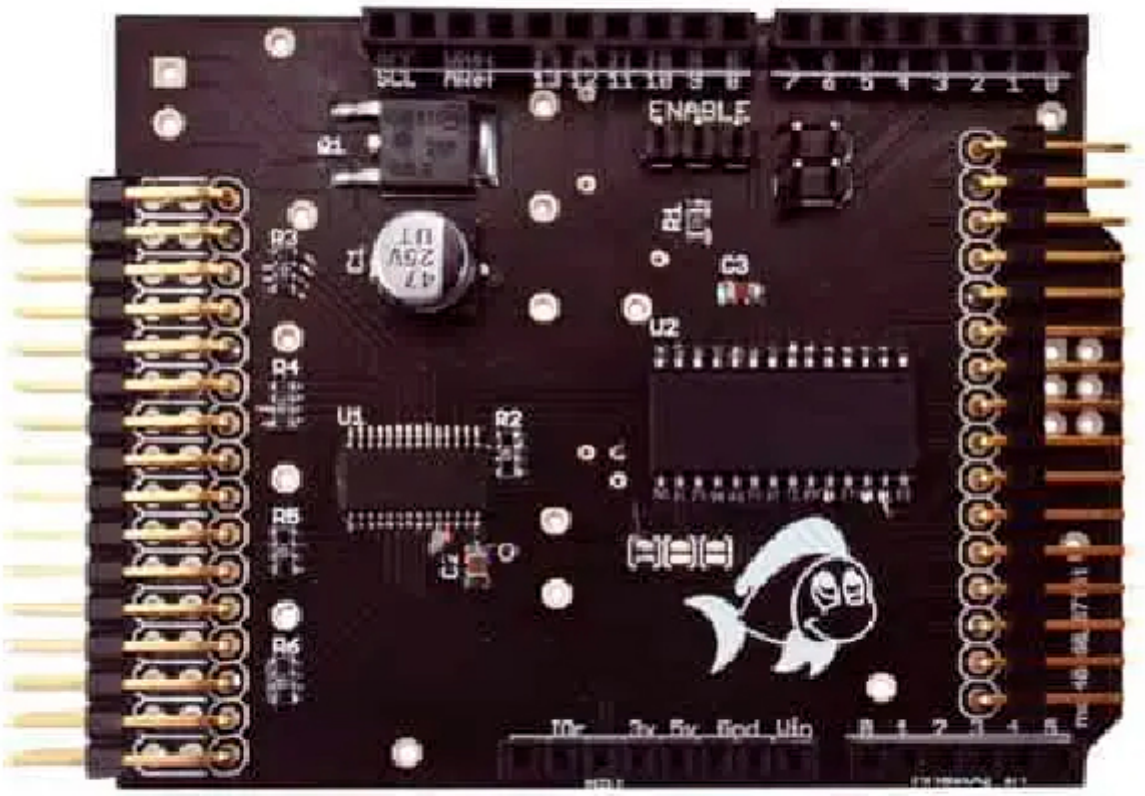
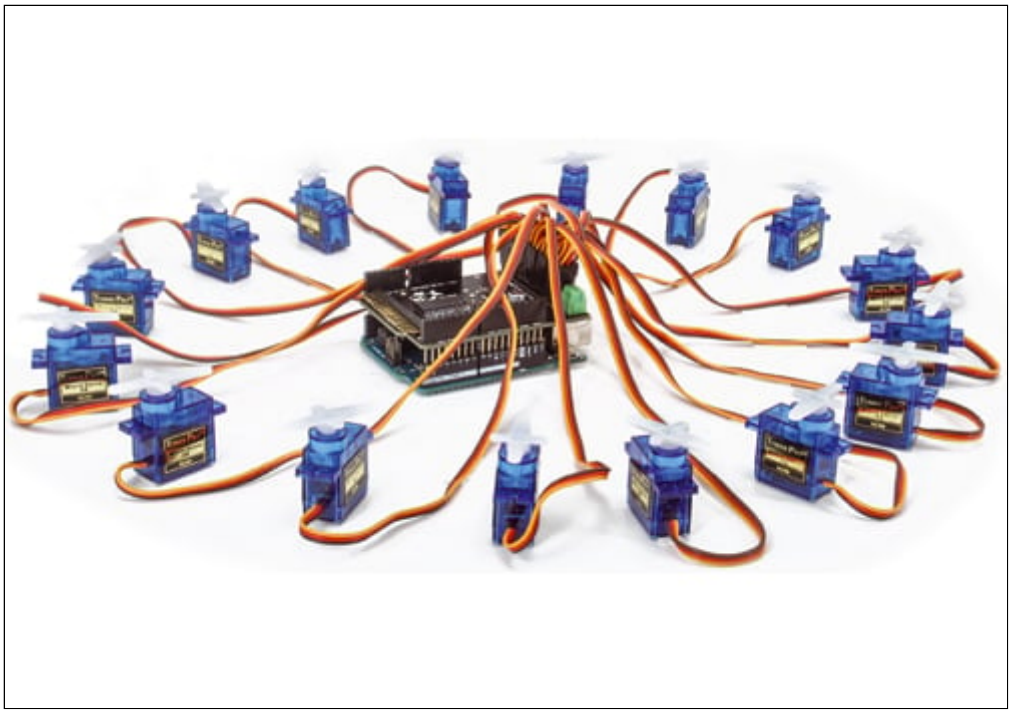
Octopus - Shield 16 I/O - in kit

Prezzo: 23.77 €

Tasse: 5.23 €

Prezzo totale (con tasse): 29.00 €





Shield di espansione, compatibile con le varie schede Arduino e con la nostra Fishino UNO 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. Il prodotto viene fornito completamente montato per quanto concerne tutta la parte SMD, mentre rimarrà da saldare la sola parte dei connettori verso Arduino e verso i dispositivi esterni. **N.B.** Arduino/Fishino e i servo RC non sono compresi (vedere prodotti correlati).

Perchè è stata realizzata

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. 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.

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 founduint8_t getNumBoards(void) const; // return number of available I/Ouint8_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 la seconda per tutte le schede connesse in un solo comando: // set pwm frequency for a single connected board // valid values 24 Hz...1526 Hzvoid setPWMFreq(uint8_t board, uint16_t freq); // set pwm frequency for ALL connected boardsvoid 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 invert = 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/Ovoid 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 onceuint16_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 utilizzando 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 nativa 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.

Documentazione e link utili

- **Sketch**
- **Libreria Octopus, per la gestione dello shield di espansione I/O Octopus**