SHIELD GSM/GPRS PER ARDUINO

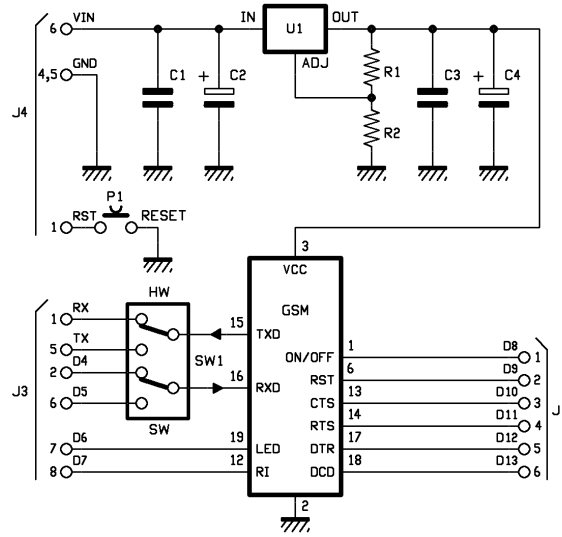
(cod. GSMSHIELD)

Shield per Arduino basata sul modulo GSM/GPRS più economico attualmente disponibile sul mercato, il SIM900 della SIMCom, fornito già montato sulla scheda TDGGSM_900. Per interfacciare il modulo con Arduino è stato realizzato un PCB che include un LM317 (fornisce al modulo circa 3,9 V), qualche condensatore di filtro e poco altro. Per controllare facilmente il modulo attraverso i pin 4 e 5, abbiamo realizzato una versione modificata della libreria di HWKitchen che comprende anche il NewSoftSerial. Con la nostra versione possiamo inviare e leggere SMS, effettuare chiamate, controllare lo stato del GSM, ecc. Il kit comprende la scheda premontata TDGGSM_900 (completa di modulo GSM SIM900), la scheda base forata e serigrafata, tutti i componenti da saldare e l'antenna GSM stilo.

Realizzazione pratica

Lo shield è di facile costruzione, dato che si tratta di un semplice stampato a singola faccia che contiene pochissimi componenti. Con un saldatore da non più di 30 watt, saldate le due resistenze e i quattro condensatori occorrenti, prestando attenzione alla polarità degli elettrolitici, poi sistemate il doppio deviatore a slitta ed il pulsante miniatura per c.s., da collocare lateralmente al circuito. Procedete inserendo e saldando l'integrato LM317, che dovete sdraiare sulla basetta dopo averne piegato i terminali ad angolo retto. Per le connessioni, utilizzate delle file di connettori single-in-line femmina a passo 2,54mm (adatte ai pin-strip di pari passo...) di due tipi: tradizionale e con terminali lunghi 20mm; del primo

Schema elettrico

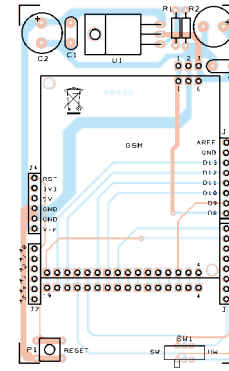


Piano di montaggio

Elenco Componenti:

R1: 470 ohm
R2: 1 kohm
C1: 100 nF multistrato
C2: 470 μF 25 VL elettrolitico
C3: 100 nF multistrato
C4: 470 μF 16 VL elettrolitico
U1: LM317
GSM: Modulo TDGGSM_900
SW1: Deviatore 2 vie miniatura
P1: Microswitch

Varie:
- Pin strip M/F 6 vie (2 pz.)
- Pin strip M/F 8 vie (2 pz.)
- Pin strip femmina 3 vie



- Pin strip femmina 16 vie
- Circuito stampato

servono due file da 3 e due da 16 contatti, mentre del secondo tipo ci occorrono due file da 6 ed altrettante da 8 contatti. Noterete che introducendo i connettori femmina coi terminali lunghi nei rispettivi fori dello stampato dello shield, i terminali stessi faranno da pin-strip e permetteranno allo shield stesso di introdursi nei connettori del modulo Arduino. Completate le saldature e verificato che non vi siano falsi contatti o cortocircuiti, il vostro shield è pronto; inseritevi il modulo col SIM900 sfruttando i contatti disposti trasversalmente e potete poi inserire il tutto su Arduino.

Usare lo shield

La prima cosa da fare prima di utilizzare lo shield è caricare lo sketch firmware in Arduino, in modo da poterlo gestire; se oltre a quello qui descritto, il sistema ospita altri shield, potrebbe essere necessario verificare se la seriale di Arduino è già occupata: se lo è, dovete spostare il doppio deviatore SW1 sulla posizione SW, assegnando, così, la comunicazione tra SIM900 e Arduino alle linee D4 e D5 di quest'ultimo. In tal caso dovete attivare l'emulazione firmware della seriale, assegnandola a D4 e D5. Nel caso il nostro sia l'unico shield montato o l'unico, tra quelli montati, che adopera la seriale hardware di Arduino, spostate pure SW1 sulla posizione HW. Fatto ciò, siete pronti per lavorare con lo shield GSM. Per la gestione del cellulare SIM900 abbiamo messo a punto la libreria "***GSM_Shield_Library"*** (scaricabile direttamente dalla scheda del prodotto presente sul nostro sito *www.futurashop.it)* derivata dalla libreria di HWKitchen, modificata per il nostro modulo.
Con la nostra versione controlliamo il modulo attraverso i contatti 5 e 6 del connettore J3 (sono le linee digitali D4 e D5 di Arduino) perché la nostra libreria GSM include NewSoftSerial; ciò vi permette di gestire il modulo, inviare e leggere SMS, effettuare chiamate, verificare lo stato del cellulare (rete agganciata ecc.) ed altro ancora.
Trovate tutte le funzioni di libreria nella **Tabella 1** di seguito riportata, opportunamente commentate: ognuna corrisponde a una funzionalità permessa dallo shield. Eventuali aggiornamenti sono disposnibili su *www.open-electronics.org.*

Tabella 1 *- Funzioni utilizzate nel controllo dello shield.*

Funzione	Descrizione
int LibVer(void)	restituisce la versione della libreria in formato XYY, che significa X.YY: ad esempio 100 vuol dire vers. 1.00)
void TurnOn(baud)	accende il modulo GSM e invia i comandi AT di inizializzazione inviabili senza bisogno di registrazione
void InitParam (byte group)	invia i parametri per l'inizializzazione del modulo GSM
void Echo(byte state)	abilita o disabilita l'eco dei comandi: Echo(1) abilita ed Echo(0) disabilita l'eco
byte CheckRegistration(void)	verifica se il GSM è stato registrato nella rete cellulare; questa funzione comunica direttamente col modulo, diversamente da IsRegistered() che legge il flag da module_status
byte IsRegistered(void)	ritorna un flag se il GSM è registrato nella rete; la procedura è veloce (richiede solo 20 ms.) e conviene utilizzarla quando è necessario verificare che il GSM sia registrato nella rete quali la verifica degli SMS e delle chiamate in arrivo
byte CallStatus(void)	verifica lo stato delle chiamate e fornisce CALL_NONE se non c'è chiamata, CALL_INCOM_VOICE se c'è una chiamata a voce in arrivo, CALL_ACTIVE_VOICE se è in corso una chiamata vocale e CALL_NO_RESPONSE in caso di chiamata senza risposta
byte CallStatusWithAuth(char *phone_number, byte first_authorized_pos, byte last_authorized_pos)	verifica lo stato delle chiamate (in entrata o in corso) e autorizza le posizioni di SIM; ritorna: - phone_number: puntatore dove viene collocato il numero telefonico della chiamata in corso in modo da memorizzarlo; - first_authorized_pos: prima posizione della rubrica nella SIM da cui sono iniziano i numeri delle chiamate autorizzate; - last_authorized_pos: ultima posizione della rubrica SIM dove finiscono i numeri autorizzati alle chiamate
void PickUp(void)	risponde alla chiamata in arrivo
void HangUp(void)	rifiuta la chiamata in arrivo o interrompe quella in corso
void Call(char *number_string)	chiama il numero indicato
void Call(int sim_position)	chiama il numero memorizzato nella posizione indicata della SIM
char SendSMS(char *number_str, char *message_str)	invia l'SMS di cui è specificato il testo, al numero indicato
char SendSMS(byte sim_phonebook_position, char *message_str)	invia l'SMS di cui è specificato il testo, al numero di cui è indicata la posizione in SIM

Funzione	Descrizione
char IsSMSPresent(byte required_status)	verifica se è presente almeno un SMS nello stato spefcificato; se c'è un SMS prima che la funzione sia eseguita e risulta non letto, viene automaticamente convertito in "letto"; i valori che la funzione ritorna sono: - SMS_UNREAD = SMS non ancora letto); - SMS_READ = SMS già letto; - ALL_SMS = tutti gli SMS memorizzati
char GetSMS(byte position, char *phone_number, char *SMS_text, byte max_SMS_len)	legge l'SMS dalla posizione indicata della SIM, ritornando i seguenti valori: - position = posizione SMS (<1..20>); - phone_number: puntatore dove viene messa il numero da cui proviene l'SMS, in modo che la posizione possa essere riservata; - SMS_text : puntatore dove viene collocato il testo dell'SMS; - max_SMS_len: massima lunghezza dell'SMS, escludendo il carattere terminatore 0×00
char GetAuthorizedSMS(byte position, char *phone_number, char *SMS_text, byte max_SMS_len, byte first_authorized_pos, byte last_authorized_pos)	legge l'SMS dalla posizione specificata della SIM e dà l'autorizzazione ad eseguirlo se proviene da un numero telefonico autorizzato; in tal caso ritorna GETSMS_AUTH_SMS, altrimenti restituisce GETSMS_NOT_AUTH_SMS
char DeleteSMS(byte position)	cancella l'SMS che si trova nella posizione specificata
char GetPhoneNumber(byte position, char *phone_number)	legge il numero telefonico dalla posizione specificata della SIM
char WritePhoneNumber(byte position, char *phone_number)	scrive il numero telefonico nella posizione specificata della SIM
char DelPhoneNumber(byte position)	cancella il numero telefonico dalla posizione specificata della SIM
char ComparePhoneNumber(byte position, char *phone_number)	confronta il numero specificato con quello memorizzato nella posizione specificata della SIM

A tutti i residenti nell'Unione Europea. Importanti informazioni ambientali relative a questo prodotto

Questo simbolo riportato sul prodotto o sull'imballaggio, indica che è vietato smaltire il prodotto nell'ambiente al termine del suo ciclo vitale in quanto può essere nocivo per l'ambiente stesso. Non smaltire il prodotto (o le pile, se utilizzate) come rifiuto urbano indifferenziato; dovrebbe essere smaltito da un'impresa specializzata nel riciclaggio.
Per informazioni più dettagliate circa il riciclaggio di questo prodotto, contattare l'ufficio comunale, il servizio locale di smaltimento rifiuti oppure il negozio presso il quale è stato effettuato l'acquisto.

Prodotto e distribuito da:
FUTURA ELETTRONICA SRL
Via Adige, 11 - 21013
Gallarate (VA)
Tel. 0331-799775
Fax. 0331-778112
Web site: *www.futurashop.it*
Info tecniche: *supporto@futurel.com*

L'articolo completo del progetto è stato pubblicato su: Elettronica In n. 155