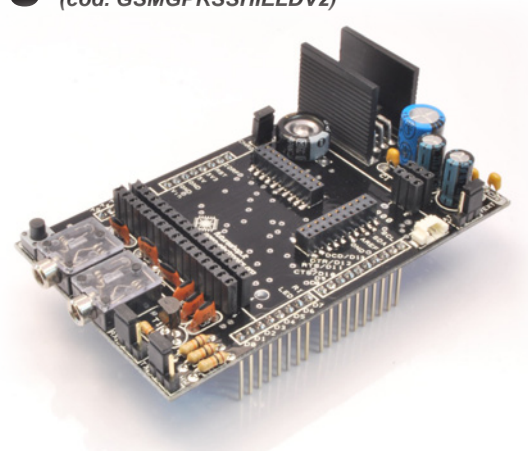
SHIELD UNIVERSALE PER MODULI GSM/GPRS/GPS *(cod. GSMGPRSSHIELDV2)*



Shield per Arduino in grado di accogliere diversi moduli GSM/GPRS/GPS della famiglia SIM900 della SIMCom che, opportunamente controllato tramite un'interfaccia seriale ed una UART per la gestione della connessione, permette di effettuare chiamate voce, inviare SMS o effettuare delle connessioni alla rete Internet. La disponibilità di nuove librerie di supporto allo sviluppo software, permettono grande flessibilità nella configurazione dei moduli e nello sviluppo di applicazioni voce, dati e basate su WEB. Con delle semplici chiamate a funzioni di alto livello è possibile, ad esempio, leggere un SMS ricevuto o avviare una chiamata ed effettuare tutte le altre operazioni eseguite da un normale cellulare. La scheda, compatibile con Arduino Duemilanove, Arduino UNO, Arduino MEGA, ecc... dispone di due prese per il flusso audio analogico. Grazie a un microfono e ad un auricolare dotati di jack da 3,5 mm (è sufficiente la cuffia standard per computer), è possibile effettuare una chiamata voce a tutti gli effetti. Nel caso di ricezione di una chiamata si udrà attraverso le cuffie una suoneria che avvisa della chiamata in arrivo. Lo shield alimenta il circuito dedicato al RTC (Real Time Clock) tramite un condensatore di elevata capacità. All'interno del SIM900, così come del SIM908, è presente un circuito che si occupa di aggiornare il contatore dell'ora anche in assenza dell'alimentazione. L'intero circuito funziona con una tensione di alimentazione di 12Vdc fornita direttamente dalla scheda Arduino. Durante le operazioni più pesanti dal punto di vista dei consumi di corrente, come l'utilizzo del GPRS, il modulo assorbe una corrente di circa 1A, dunque è necessario che la sorgente di alimentazione sia in grado di fornire tale intensità di corrente. Lo shield prevede un connettore per il collegamento di una batteria esterna ricaricabile, come fonte di alimentazione primaria, utilizzabili solamente con il SIM908. La tensione necessaria per la ricarica della batteria viene prelevata direttamente dalla scheda Arduino (dal pin Vin). L'ingombro massimo dello shield è di 100 x 57mm.

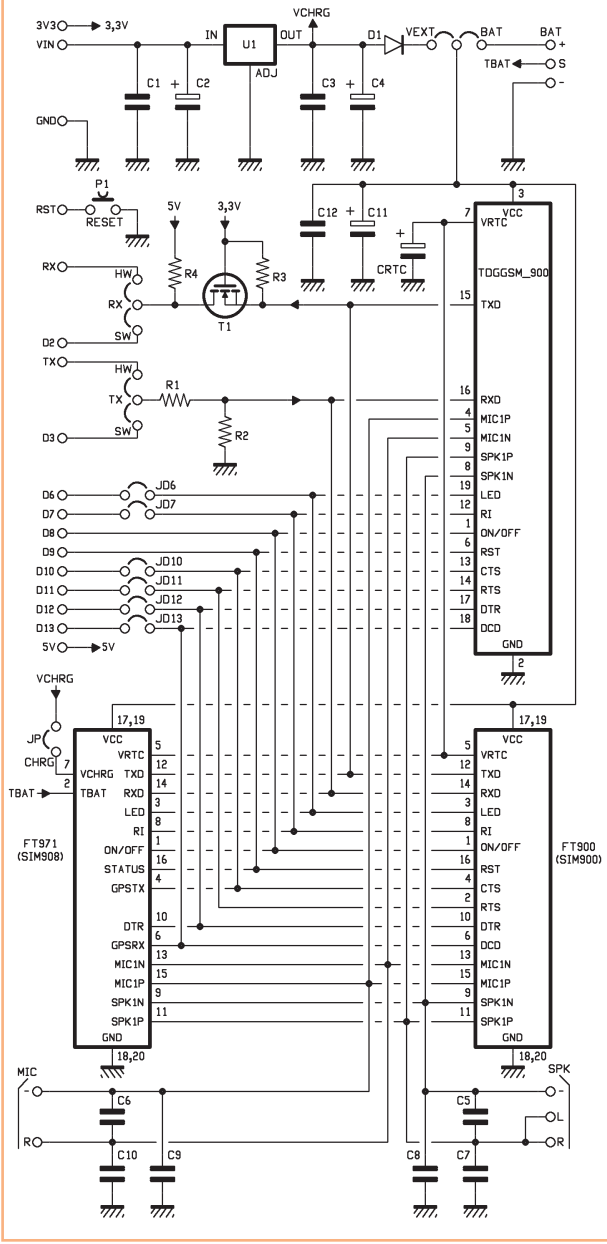
Realizzazione pratica

Lo shield è di facile realizzazione. Con un saldatore da non più di 30W, saldare sulla scheda i 2 connettori SMD (siglati SIM908 e SIM900), il connettore 3 poli per batteria e successivamente tutti gli altri componenti, dando precedenza a quelli con basso profilo (come resistenze, diodi, micropulsanti, prese jack, ecc). Posizionare sulla scheda il dissipatore termico sul quale dovrà essere fissato, mediante vite e dado, il regolatore U1 con i relativi pin piegati a 90°. Effettuare la saldatura del regolatore e di tutti i pin strip assicurandosi che quelli previsti per il collegamento con la scheda Arduino (6, 8 e 10 vie, lunghi) abbiano le rispettive punte rivolte verso il basso. Completate le saldature verificare attentamente che non vi siano falsi contatti o cortocircuiti. Applicate quindi lo shield sulla scheda Arduino ed effettuate le impostazioni seguendo le indicazioni riportate nel paragrafo "Configurazione".

Configurazione

Per preservare la compatibilità con l'Arduino Mega, la modalità di comunicazione con il SIM900 (seriale hardware, utilizzando i pin 0 e 1 dell'Arduino Uno o seriale software utilizzando invece i pin 2 e 3) è impostabile tramite jumper ma vi è anche la possibilità di utilizzare i pin a propria scelta tramite un semplice collegamento. In Fig. 1 sono mostrate le tre configurazioni possibili.

Schema elettrico



Sempre per preservare la massima flessibilità e personalizzazione, si è predisposta la scheda con delle piazzole da saldare sul retro (Fig. 2), che consentono di andare a effettuare i collegamenti per portare in ingresso alle porte digitali dell'Arduino, i segnali di controllo del flusso dati (CTS, RTS) o segnali di alert per chiamate in arrivo o SMS non letto (RI). È possibile dunque disabilitare questi collegamenti andando così a risparmiare la quantità di ingressi o uscite utilizzati, avendo sempre la possibilità, eventualmente, di sfruttarli come meglio si crede con la realizzazione di un ponticello.

La Fig. 3 mostra la presenza di due connettori a pettine sulla parte superiore. Questi ulteriori collegamenti consentono l'utilizzo dello shield anche con i nuovi moduli compatti del SIM900 e SIM908, dotati di un differente pin-out mirato al risparmio di spazio. Il nuovo modulo SIM908 sempre della SIMCom, si caratterizza per la presenza di un localizzatore GPS a 42 canali con un'accuratezza inferiore ai due metri e mezzo.

Mediante il jumper BAT/VEXT (Fig. 4) è possibile impostare la sorgente di alimentazione dello shield: posizionare il jumper su VEXT se si desidera alimentare il dispositivo tramite la scheda Arduino oppure su BAT se si utilizza una batteria esterna ricaricabile. Per consentire la ricarica di quest'ultima è necessario chiudere il jumper siglato CHRG (Fig. 5) posto in prossimità del condensatore CRTC.

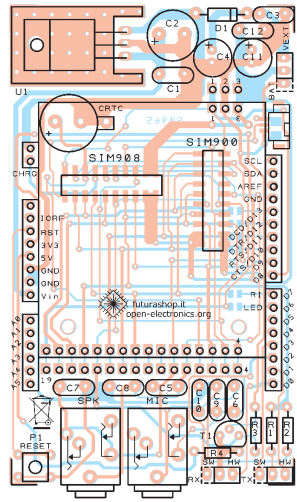
Libreria GSM GPRS

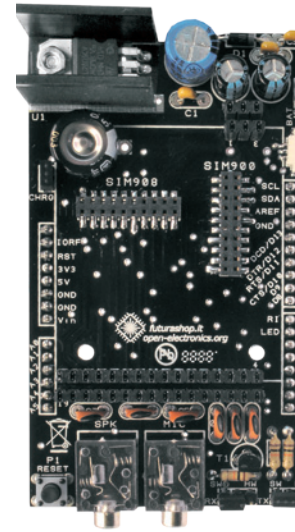
La libreria software legata alla scheda GSM GPRS shield è open-source e utilizza il servizio di hosting di Google Project, raggiungibile all'indirizzo *http://code.google.com/p/gsm-shield-*

Piano di montaggio

Elenco Componenti:

R1: 10 kohm
R2: 10 kohm
R3: 10 kohm
R4: 10 kohm
C1: 100 nF multistrato
C2: 470 μF 25 VL elettrolitico
C3: 100 nF multistrato
C4: 220 μF 16 VL elettrolitico
C5÷C10: 47 pF ceramico
C11: 220 μF 16 VL elettrolitico
C12: 100 nF multistrato
CRCT: Condensatore basso profilo 0,1F
U1: 7805
T1: BS170
D1: 1N4007
P1: Microswitch
MIC: Presa jack stereo 3,5 mm da CS
SPK: Presa jack stereo 3,5 mm da CS





Varie:
- Dissipatore (ML26)
- Vite 10 mm
- Dado 3 MA
- Strip Maschio 6 vie lungo
- Strip Maschio 8 vie lungo (2 pz.)
- Strip Maschio 10 vie lungo
- Strip Maschio 2 vie corto
- Strip Maschio 3 vie corto (3 pz.)
- Strip Femmina 3 vie (2 pz.)
- Strip Femmina 16 vie (2 pz.)
- Presa MMS SMT 2x10 vie passo 2 mm (2 pz.)
- Presa 3 vie 90° passo 1,25 mm
- Jumper (4 pz.)
- Circuito stampato

arduino/. La libreria viene costantemente aggiornata e migliorata con l'aggiunta di nuove funzioni e correzioni di eventuali bug, perciò si consiglia di controllare di avere sempre l'ultima release. È stata inoltre prevista la gestione della comunicazione TCP/IP attraverso GPRS. Grazie agli ormai diffusi smartphone e tablet, sono presenti in commercio numerose offerte per l'utilizzo di Internet sui dispositivi mobili. In base alle esigenze ed alle conoscenze dell'utente è possibile implementare diverse funzioni, atte a svolgere, ad esempio, funzioni quali: l'invio di email, la

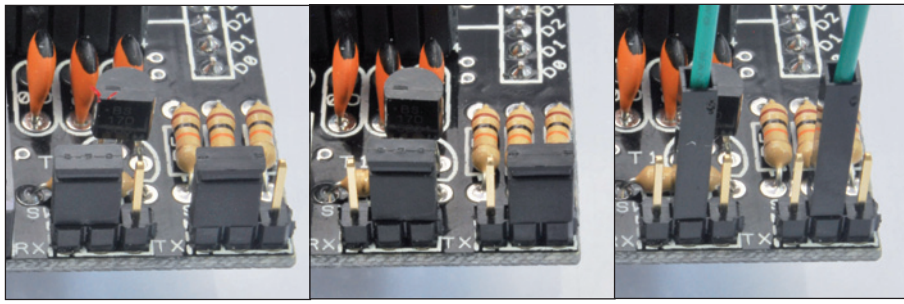


***Fig. 1** - A sinistra la configurazione per la seriale software, al centro hardware e, a destra, personalizzata.*

pubblicazioni di dati attraverso il metodo GET o lo streaming di flussi audio/video. Per una panoramica delle principali funzioni attualmente implementate, si rimanda alle tabelle contenenti l'elenco dei comandi.

Innanzi tutto è necessario avere all'interno della directory *libraries*, contenuta nella directory principale di Arduino, la cartella GSM_GPRS, contenente tutte le funzioni utilizzabili. Inoltre per poter utilizzare la seriale software, solamente nel caso si utilizzi un IDE di Arduino precedente alla 1.00, bisogna avere anche la libreria *NewSoftSerial* (reperibile da *www.arduiniana.com*) alla quale questo shield si appoggia per la comunicazione. Ora nel

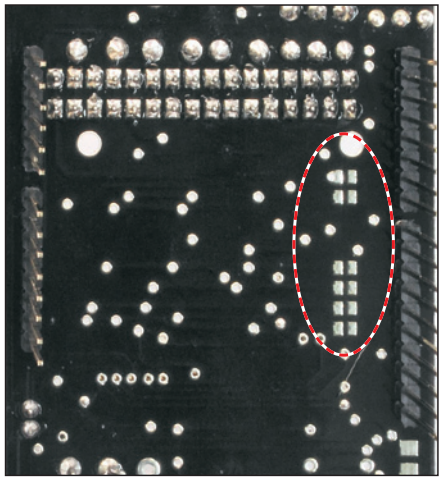


***Fig. 2** - Dettaglio delle piste dei segnali di controllo del SIM900, con la possibilità di portarli in ingresso ad Arduino tramite una semplice saldatura.*

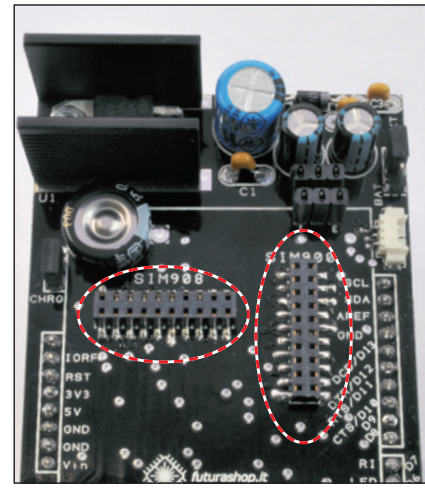


***Fig. 3** - Strip per il collegamento dei nuovi moduli che equipaggiano i chip della famiglia SIMCom in versione compatta.*

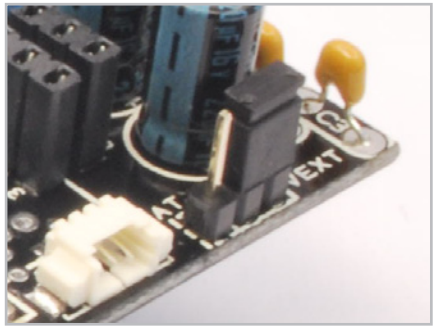


***Fig. 4** - Jumper per la scelta della sorgente di alimentazione (BAT= alimentazione tramite batteria; VEXT= alimentazione tramite scheda Arduino).*

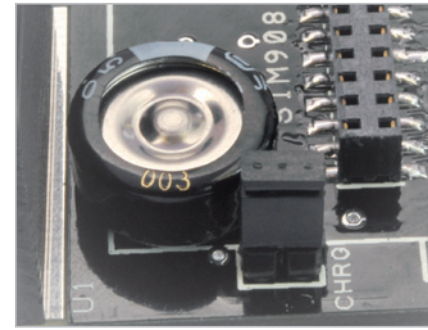


***Fig. 5** - Jumper CHRG. Chiudendo questo ponticello il sistema provvede alla ricarica della batteria esterna eventualmente collegata allo shield.*

SIM900.h
È sempre necessario includere questo file in quanto contiene le funzioni base per l'avvio e la configurazione del modulo SIM900. Per queste funzioni non è necessario istanziare un oggetto gsm, in quanto questa operazione è stata già effettuata all'interno della libreria. Sarà sufficiente chiamare le funzioni utilizzando "gsm." come prefisso.

int gsm.begin(**int** baudrate)	Accende il modulo e stabilisce il baud-rate per la comunicazione.	gsm.TurnOn(9600);
void SimpleRead()	Legge un byte dal buffer della seriale.	gsm.SimpleRead();
void SimpleWrite(**char*** cmd)	Scrive la stringa (o un intero) sulla porta seriale.	gsm. SimpleRead("AT+CSQ");
char SendATCmdWaitResp (**char const*** AT_cmd_string, **uint16_t** start_comm_tmout, **uint16_t** max_interchar_ tmout, **char const*** respon- se_string, **byte** no_of_attempts)	Invia un comando AT passato come parametro, controllando i tempi di timeout per la ricezione di una risposta e tra un carattere e l'altro. Dopo aver ricevuto la risposta la confronta con quella aspettata e se diversa riesegue il comando per il numero di volte impostato. I possibili output sono riportati di seguito, con la relativa enumerazione. AT_RESP_ERR_NO_RESP oppure -1: Nessuna risposta ricevuta. AT_RESP_ERR_DIF_RESP oppure 0: Risposta differente da quella attesa. AT_RESP_OK oppure 1: La risposta contiene la stringa attesa.	If(gsm.SendATCmdWa itResp("AT",500,50," OK",3)== AT_RESP_OK) Serial.println("OK");

call.h
Nel caso si vogliano effettuare delle telefonate, rispondere o semplicemente rifiutare una telefonata in arrivo, è necessario l'utilizzo di questa classe. Per l'utilizzo di tali funzioni è sufficiente istanziare l'oggetto all'interno dello sketch. Nelle funzioni riportate nella tabella seguente si fa riferimento a un oggetto creato con il seguente comando all'inizio dello sketch: *CallGSM call*;

void Call(**char*** number_string)	Effettua una chiamata al numero contenuto nella stringa	call. Call("+393471234567")
void PickUp(**void**)	Durante una chiamata in arrivo, risponde e attiva la comunicazione audio.	call.PickUp();
void HangUp(**void**)	Durante una chiamata attiva, riaggancia e disattiva la comunicazione audio.	call.HangUp();
byte CallStatus(void);	Restituisce lo stato riguardo le telefonate. I possibili byte di output sono enumerati con i seguenti nomi: CALL_NONE oppure 0: Nessuna chiamata. CALL_INCOM_VOICE oppure 1: Chiamata in arrivo. CALL_ACTIVE_VOICE oppure 2: Chiamata attiva.	If(call.CallStatus()== CALL_ACTIVE_VOICE) Serial. println("CHIAMATA IN CORSO");
byte CallStatu- sWithAuth (**char*** phone_number, **byte** first_authori- zed_pos, **byte** last_ authorized_pos);	A differenza del precedente, distingue se la chiamata attiva, o in arrivo, appartiene a un numero salvato sulla SIM, in una posizione compresa tra quella iniziale e finale, passate come parametri. I possibili byte di output sono enumerati con i seguenti nomi: CALL_NONE oppure 0: Nessuna chiamata. CALL_INCOM_VOICE_AUTH oppure 3: Chiamata in arrivo da un numero autenticato. CALL_INCOM_VOICE_NOT_AUTH oppure 4: Chiamata in arrivo da un numero non autenticato.	If(call.CallStatusWi- thAuth ()==CALL_IN- COM_VOICE_AUTH) Serial. println("CHIAMATA IN ARRIVO DA UN NUME- RO AUTENTICATO");

caso si volessero cambiare i pin della seriale software, attraverso l'apposito jumper, è opportuno riportare tale cambiamento anche all'interno della libreria stessa. Tale modifica va effettuata nelle prima riga del file GSM.cpp. Per motivi legati all'utilizzo di memoria, si è scelto di suddividere le funzioni in classi differenti contenute in file differenti, in modo da consentire all'utente di includere o meno le parti di codice necessarie, andando così a risparmiare memoria RAM, lasciandola libera per il resto del programma. Per il funzionamento di base è sempre necessario includere i file SIM900.h e SoftwareSerial.h (o NewSoftSerial.h nel caso di IDE precedente alla 1.00), mentre a seconda delle necessità si può includere call.h (per la gestione delle chiamate), sms.h (per l'invio, ricezione e salvataggio degli SMS) e inetGSM.h (contenente funzioni legate all'HTTP e alla connessione GPRS). Laddove non diversamente specificato si sottintende che la funzione restituisce 1 in caso di operazione completata con successo, altrimenti 0.

Esempio per SMS e chiamate

Affrontiamo ora, passo-passo, la realizzazione del nostro primo sketch per l'utilizzo dello shield utilizzando la versione 1.00 dell'IDE di Arduino. Realizzeremo un programma che controlla il modulo in modo tale che quando riceve una telefonata da un numero preimpostato (salvato in una determinata posizione sulla SIM), rifiuta la telefonata e invia un SMS in risposta al numero chiamante con il valore letto da un input. Innanzitutto bisogna estrarre dalla cartella compressa i file della libreria all'interno della cartella *libraries* contenuta all'interno della cartella d'installazione di Arduino. Ora siamo pronti

SMS.h

Per la gestione dei messaggi di testo bisogna utilizzare questa classe apposita. Come in precedenza è necessario richiamarla all'interno dello sketch ed istanziarne successivamente un oggetto. Ad esempio nelle funzioni seguenti si fa riferimento a un oggetto creato, all'inizio dello sketch, con il comando *SMSGSM sms;*

char SendSMS(**char** *number_str, **char** *message_str)	Tramite il seguente comando viene inviato un SMS al numero contenuto nella prima stringa passata come parametro con il testo contenuto nella seconda.	sms.SendSMS ("+393471234567", "Hello Arduino");
char SendSMS(**byte** sim_phonebook_position, **char** *message_str)	Invia un SMS come in precedenza, dove al posto della stringa del destinatario viene passata la posizione del contatto salvato sulla SIM.	sms.SendSMS (1,"Hello Arduino");
char GetSMS(**byte** position, char *phone_number, **char** *SMS_text, **byte** max_SMS_len)	Legge l'SMS salvato sulla SIM nella posizione passata come parametro, salvando il numero del mittente nella prima stringa passata e il contenuto nella seconda di lunghezza indicata.	char number[13]; char text[200]; sms.GetSMS (1,number,text,200);

inetGSM.h

In questa classe vengono incluse le funzioni per stabilire una connessione e gestire le comunicazioni tramite protocollo HTTP. Negli esempi seguenti è stata creato un oggetto con il comando *InetGSM inet;*

int httpGET(**const char*** server, **int** port, **const char*** path, **char*** result, **int** resultlength)	Invia una richiesta di tipo GET al server indicato, sulla porta specificata, richiedendo un determinato path e salvando la risposta in una stringa di lunghezza specificata. Restituisce il numero di byte letti.	char text[200]; inet.httpGET ("www.elettronicain.it", 80,"/",text,200);
int httpPOST(**const char*** server, **int** port, **const char*** path, **const char*** parameters, **char*** result, **int** resultlength)	Invia una richiesta di tipo POST al server indicato, sulla porta specificata, richiedendo un determinato path, passando i parametri indicati e salvando la risposta su stringa di lunghezza specificata. Restituisce il numero di byte letti.	char text[200]; inet.httpGET ("www.elettronicain.it",80,"/",text,200);
int attachGPRS(**char*** domain, **char*** dom1, **char*** dom2)	Avvia la connessione GPRS utilizzando l'APN passato come primo parametro. Il secondo e terzo parametro sono due stringhe che contengono rispettivamente username e password. Se non è richiesta l'autenticazione è sufficiente passare le ultime due stringhe vuote.	inet. attachGPRS("internet.wind","","");
int deattachGPRS(**void**)	Disconnette il modulo dalla rete GPRS.	inet.deattachGPRS();
int connectTCP(**const char*** server, **int** port)	Stabilisce una connessione come client al server passato come parametro, sulla porta definita dal secondo parametro.	inet.connectTCP("www.elettronicain.it",80);
int disconnectTCP(**void**)	Chiude la comunicazione con il server.	inet.disconnectTCP();
int connectTCPServer (**intport**)	Pone il modulo in ascolto sulla porta specificata in attesa di una connessione da parte di un client.	inet.connectTCPServer(80);
boolean connectedClient(**void**)	Restituisce true se un client è connesso al modulo, altrimenti false.	inet.connectedClient();

per la realizzazione del nostro primo programma. Lo sketch dovrà come prima cosa caricare le librerie tramite i comandi

#include "SIM900.h"
#include <SoftwareSerial.h>

Successivamente carichiamo, decommentando opportunamente, i file relativi alle classi contenenti le funzioni che vogliamo usare per la gestione delle telefonate e degli SMS.

#include "sms.h"
#include "call.h"

Dopo aver definito le variabili che ci serviranno eseguiamo la procedura di inizializzazione nel setup. Impostiamo il pin da cui vogliamo leggere il valore che verrà poi inviato tramite SMS, configuriamo la comunicazione seriale e avviamo la procedura di inizializzazione del modulo con la funzione *gsm.begin,* in cui passiamo il baud-rate (solitamente per una corretta comunicazione dei dati attraverso GPRS è consigliabile non salire sopra i 4800 baud). A questo punto entriamo nel vivo del programma, che dovrà controllare periodicamente lo stato delle telefonate entranti. Per fare questo all'interno del ciclo *loop* utilizzeremo la funzione call. *CallStatusWithAuth* salvando il return nel byte dichiarato precedentemente. In caso di chiamata in arrivo o in corso, viene salvato il numero del mittente (o destinatario) della telefonata all'interno della stringa *number*.
Una volta confrontato il valore salvato con CALL_INCOM_VOICE_AUTH, che descrive una chiamata in arrivo da un numero tra quelli impostati, rifiutiamo la chiamata con la funzione *gsm.HangUp* e dopo aver aspettato 2 secondi, leggiamo il valore dell'input e inviamo il messaggio.

Ricordiamo che il valore letto è un intero e deve essere prima convertito in stringa utilizzando la funzione *itoa*. Ricordiamoci di inserire un ritardo, all'interno della funzione *loop*, per far si che il modulo venga interrogato a intervalli non inferiori al secondo. Comandi inviati in rapida sequenza potrebbero compromettere la stabilità del modulo stesso. Nel caso non ricevessimo l'esito di inizializzazione corretta, sarà necessario controllare l'alimentazione. Ricordiamo che è consigliato utilizzare anche una fonte di alimentazione esterna poiché a volte la sola alimentazione fornita dalla porta USB non risulta sufficiente. Se l'alimentazione dovesse risultare corretta, bisogna controllare che all'interno del file GSM.cpp, contenuto nella libreria siano dichiarati correttamente i pin per la seriale. Di base la nuova versione utilizza i pin 2 e 3, mentre la precedente utilizzava i pin 4 e 5.

#define _GSM_TXPIN_ 2
#define _GSM_RXPIN_ 3

L'intero programma risulta essere quello visibile nel Listato 1.

Esempio per internet

Analizziamo uno degli esempi contenuti all'interno della libreria per la connessione a internet. Realizzeremo un programma in grado di ricevere il contenuto HTML da una pagina web e salvarne i primi 50 caratteri. Poiché utilizzeremo solo le funzioni riguardanti la connessione internet e HTTP caricheremo oltre ai file standard della libreria, il file inetGSM.h
Istanziamo un oggetto per la gestione delle funzioni Internet tramite

InetGSM inet;

e come in precedenza eseguiamo la routine di inizializzazione. Successivamente stabiliamo una connessione GPRS. In questo passo è necessario eseguire il comando "AT+CIFSR" che richiede al provider l'indirizzo IP assegnato al SIM900. Questo passo è fondamentale in quanto con alcuni provider potrebbe non funzionare la connessione

Listato 1

```
#include "SIM900.h"
#include <SoftwareSerial.h>
//carichiamo i file necessari allo sketch
#include "sms.h"
#include "call.h"

CallGSM call; //instanzia un oggetto per le chiamate
SMSGSM sms;   //instanzia un oggetto per gli SMS

char number[20];
byte stat=0;
int value=0;
int pin=1;
char value_str[5];

void setup()
{
  pinMode(pin,INPUT);
  Serial.begin(9600);
  Serial.println("GSM GPRS Shield");
  //inizializziamo il modulo e configuriamo il baud-
rate
  if (gsm.begin(2400))
    Serial.println("\nstatus=READY");
  else Serial.println("\nstatus=IDLE");
};

void loop()
{
  stat=call.CallStatusWithAuth(number,1,3);
  //se il mittente della chiamata in arrivo rientra
  //tra quelli autorizzati (salvati sulla SIM tra la
  //posizione numero 1 e la 3)
  if(stat==CALL_INCOM_VOICE_AUTH){
    //rifiuta la chiamata
    call.HangUp();
    delay(2000);
    //leggi lo stato dell'input
    value=digitalRead(1);
    //converti il valore in stringa
    itoa(value,value_str,10);
    //invia un SMS al mittente della telefonata
        //salvato in precedenza
    sms.SendSMS(number,value_str);
  }
  //eseguiamo questa operazione con
  //un tempo di attesa di un secondo
  delay(1000);
};
```

se non viene precedentemente effettuata questa richiesta. Tramite la funzione *gsm.WhileSimpleRead*, contenuta nella classe gsm, chiediamo di leggere, sulla porta seriale, l'intero contenuto del buffer. Una volta svuotato il buffer, lo sketch passerà alle funzioni successive.
A questo punto siamo connessi, non ci resta che stabilire una connessione TCP con il server, inviare una richiesta GET per una pagina internet e memorizzare il contenuto di risposta in un array precedentemente dichiarato. Tutto questo viene svolto dalla funzione *httpGET* nella classe inetGSM. Oltre al server e la porta (80 nel caso del protocollo HTTP), dovremo indicare il path dove è contenuta la pagina richiesta. Ad esempio nel caso si voglia scaricare la pagina di Wikipedia relativa ad Arduino, raggiungibile con il seguente indirizzo it.wikipedia.org/wiki/Arduino_(hardware) il path sarà descritto da /wiki/Arduino_(hardware) mentre il server it.wikipedia.org.

numdata=inet.httpGET("it.wikipedia.org ", 80, "/wiki/Arduino_(hardware) ", msg, 50);

Ovviamente se desideriamo salvare un numero maggiore di caratteri della risposta, basterà inizializzare una stringa di dimensioni maggiori, ponendo attenzione a non saturare la RAM messa a disposizione da Arduino, altrimenti rischieremo di ottenere comportamenti anomali, come stalli o riavvii (Listato 2).

Note

Lo shield non comprende il modulo GSM, l'antenna e la batteria, acquistabili separatamente. La scheda è predisposta per accogliere uno dei seguenti moduli GSM (la scelta è funzione delle proprie esigenze): FT900M,

Listato 2

```
#include "SIM900.h"
#include <SoftwareSerial.h>
//carichiamo i file necessari allo sketch
#include "inetGSM.h"

InetGSM inet; //instanzia un oggetto per Internet

char msg[50];
int numdata;
char inSerial[50];
int i=0;
boolean started=false;

void setup()
{
  Serial.begin(9600);
  Serial.println("GSM Shield testing.");
  //inizializziamo il modulo e configuriamo il baud-
rate
  if (gsm.begin(2400)){
    Serial.println("\nstatus=READY");
    started=true;
  }
  else Serial.println("\nstatus=IDLE");

  if(started){
    //stabilisci una connessione attraverso l'APN
     //specificato, solitamente i campi username e
password
    //non sono richiesti
    if (inet.attachGPRS("internet.wind","", ""))
      Serial.println("status=ATTACHED");
    else Serial.println("status=ERROR");
    delay(1000);

    //leggi l'indirizzo IP
    gsm.SimpleWriteln("AT+CIFSR");
    delay(5000);
    //leggi finché il buffer non è vuoto
    gsm.WhileSimpleRead();

     //invia una richiesta GET al server e salva i
primi 50
    //byte all'interno della stringa msg
     numdata=inet.httpGET("www.google.com", 80, "/",
msg, 50);
    //stampa i risultati
    Serial.println("\nNumero di byte ricevuti:");
    Serial.println(numdata);
    Serial.println("\nDati ricevuti:");
    Serial.println(msg);
  }
};
void loop()
{
};
```

Comandi **AT Hayes**

Molti modem in commercio, siano essi destinati ad utilizzi di tipo mobile, o meno, utilizzano per lo scambio di informazioni con l'utente, determinati comandi sviluppati originariamente per il modem Hayes Smartmodem. Tali stringhe di controllo prendono il nome di comandi AT (dalla parola AT-tention) Hayes e sono per la maggior parte universali a tutti i modem.
Nel caso di modem GSM (come per il SIM900), la gestione della rete, l'utilizzo di SMS e il controllo delle chiamate voce è regolamentato da comandi standard universali. Mentre per altre funzioni il comando può essere strettamente proprietario dell'azienda produttrice e non sempre viene reso pubblico attraverso i datasheet.
Di seguito sono riportati alcuni dei più utilizzati comandi AT universali nel campo dei modem GSM:

AT&F	Ripristina la configurazione di fabbrica
AT+CSQ	Fornisce un numero che indica la potenza del segnale: 0 potenza di segnale inferiore a -113 dBm 1 potenza di segnale tra -113 e -109 dBm 2-30 potenza di segnale tra -109 e -53 dBm 31 potenza di segnale superiore a -51 dBm 99 potenza di segnale sconosciuta
AT+COPS=?	Restituisce i dati dell'operatore telefonico a cui il modulo è agganciato
AT+CREG?	Fornisce informazioni riguardo lo stato della registrazione della SIM sulla rete.
AT+CMGR=n	Legge l'SMS in posizione n
AT+CMGS="1234567890"	Invia la stringa digitata successivamente tramite SMS al numero indicato
ATD1234567890;	Chiama il numero indicato
AT+CFUN=1	Piena funzionalità
AT+CSCLK=0	Disattiva la modalità standby seriale
AT&W	Salva i parametri correnti

Per l'utilizzo standard del GSM GPRS shield, non è necessario conoscere questa sintassi di comandi, in quanto la libreria è stata sviluppata per rendere il più trasparente possibile l'utilizzo del modem senza alcuna conoscenza specifica nel campo dei comandi AT Hayes.
Configurazione seriale dei moduli SIMCOM (es.SIM900): Baudrate 115200bps/8/N/1
I moduli SIMCOM sono configurati di default per andare in standby su lettura seriale nel caso non ricevano proprio su seriale dei dati entro 5 secondi dall'ultimo.

FT971 e TDGGSM_900. La batteria da utilizzare deve essere di tipo ricaricabile al litio con tensione nominale di 3,7V (es. cod. BATTGC1800).

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*