```
/*!
 * @file DFRobot_sim808.cpp
 * @A library  for DFRobot's SIM808 GPS/DFRobot_SIM808/GSM Shield
 *
 * @copyright   [DFRobot](http://www.dfrobot.com), 2016
 *
 * @author [Jason](jason.ling@dfrobot.com)
 * @version  V1.0
 * @date  2016-09-23
 
 * The MIT License (MIT)
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
 

#include <stdio.h>
#include "DFRobot_sim808.h"

extern Stream *serialSIM808;

DFRobot_SIM808* DFRobot_SIM808::inst;
char receivedStackIndex = 0;
char receivedStack[130];
const char *des = "$GPRMC";

//char *receivedStack="$GPRMC,165445.000,A,3110.8635,N,12133.4627,E,0.58,70.26,220916,,,A*57";

DFRobot_SIM808::DFRobot_SIM808(HardwareSerial *mySerial)
{
    inst = this;    
    serialFlag = 1;
    hgprsSerial = mySerial;
    sim808_init(mySerial, 1);
}

DFRobot_SIM808::DFRobot_SIM808(SoftwareSerial *mySerial)
{
    inst = this;
    serialFlag = 0;
    gprsSerial = mySerial;
   sim808_init(mySerial, 0);
}

bool DFRobot_SIM808::init(void)
{
    //???AT????????
    if(!sim808_check_with_cmd("AT\r\n","OK\r\n",CMD)){   
        ....
        return false;
        
    }
	//???SIM????????????????
	// 1 : OK
    if(!sim808_check_with_cmd("AT+CFUN=1\r\n","OK\r\n",CMD)){
        return false;
    }

	//???SIM????
    if(!checkSIMStatus()) {
		return false;
    }
    return true;
}

bool DFRobot_SIM808::checkPowerUp(void)
{
  return sim808_check_with_cmd("AT\r\n","OK\r\n",CMD);
}

void DFRobot_SIM808::powerUpDown(uint8_t pin)
{
  // power on pulse for SIM900 Shield
  digitalWrite(pin,LOW);
  delay(1000);
  digitalWrite(pin,HIGH);
  delay(2000);
  digitalWrite(pin,LOW);
  delay(3000);
}

void DFRobot_SIM808::powerReset(uint8_t pin)
{
  // reset for SIM800L board.
  // RST pin has to be OUTPUT, HIGH
  digitalWrite(pin,LOW);
  delay(1000);
  digitalWrite(pin,HIGH);
  delay(3000);  
}
  
  
bool DFRobot_SIM808::checkSIMStatus(void)
{
    char gprsBuffer[32];
    int count = 0;
    sim808_clean_buffer(gprsBuffer,32);
    while(count < 3) {
        sim808_send_cmd("AT+CPIN?\r\n");
        sim808_read_buffer(gprsBuffer,32,DEFAULT_TIMEOUT);
        if((NULL != strstr(gprsBuffer,"+CPIN: READY"))) {  //???SIM??????
            break;
        }
        count++;
        delay(300);
    }
    if(count == 3) {
        return false;
    }
    return true;
}

bool DFRobot_SIM808::sendSMS(char *number, char *data)
{
    //char cmd[32];
    if(!sim808_check_with_cmd("AT+CMGF=1\r\n", "OK\r\n", CMD)) { // Set message mode to ASCII
        return false;
    }
    delay(500);
    sim808_flush_serial();
    sim808_send_cmd("AT+CMGS=\"");
    sim808_send_cmd(number);
    //sprintf(cmd,"AT+CMGS=\"%s\"\r\n", number);
    //snprintf(cmd, sizeof(cmd),"AT+CMGS=\"%s\"\r\n", number);
//    if(!sim808_check_with_cmd(cmd,">",CMD)) {
    if(!sim808_check_with_cmd("\"\r\n",">",CMD)) {
        return false;
    }
    delay(1000);
    sim808_send_cmd(data);
    delay(500);
    sim808_send_End_Mark();
    return sim808_wait_for_resp("OK\r\n", CMD);
}

char DFRobot_SIM808::isSMSunread()
{
    char gprsBuffer[48];  //48 is enough to see +CMGL:
    char *s;
    
     sim808_check_with_cmd("AT+CMGF=1\r\n","OK\r\n",CMD);
    delay(1000);

    //List of all UNREAD SMS and DON'T change the SMS UNREAD STATUS
    sim808_send_cmd(F("AT+CMGL=\"REC UNREAD\",1\r\n"));
    /*If you want to change SMS status to READ you will need to send:
          AT+CMGL=\"REC UNREAD\"\r\n
      This command will list all UNREAD SMS and change all of them to READ
      
     If there is not SMS, response is (30 chars)
         AT+CMGL="REC UNREAD",1  --> 22 + 2
                                 --> 2
         OK                      --> 2 + 2

     If there is SMS, response is like (>64 chars)
         AT+CMGL="REC UNREAD",1
         +CMGL: 9,"REC UNREAD","XXXXXXXXX","","14/10/16,21:40:08+08"
         Here SMS text.
         OK
         
         or

         AT+CMGL="REC UNREAD",1
         +CMGL: 9,"REC UNREAD","XXXXXXXXX","","14/10/16,21:40:08+08"
         Here SMS text.
         +CMGL: 10,"REC UNREAD","YYYYYYYYY","","14/10/16,21:40:08+08"
         Here second SMS
         OK
    */

    sim808_clean_buffer(gprsBuffer,31);
    sim808_read_buffer(gprsBuffer,30,DEFAULT_TIMEOUT);
    //Serial.print("Buffer isSMSunread: ");Serial.println(gprsBuffer);

    if(NULL != ( s = strstr(gprsBuffer,"OK"))) {
        //In 30 bytes "doesn't" fit whole +CMGL: response, if recieve only "OK"
        //    means you don't have any UNREAD SMS
        delay(50);
        return 0;
    } else {
        //More buffer to read
        //We are going to flush serial data until OK is recieved
        sim808_wait_for_resp("OK\r\n", CMD);
        //sim808_flush_serial();
        //We have to call command again
        sim808_send_cmd("AT+CMGL=\"REC UNREAD\",1\r\n");
        sim808_clean_buffer(gprsBuffer,48);
        sim808_read_buffer(gprsBuffer,47,DEFAULT_TIMEOUT);
        //Serial.print("Buffer isSMSunread 2: ");Serial.println(gprsBuffer);
        if(NULL != ( s = strstr(gprsBuffer,"+CMGL:"))) {
            //There is at least one UNREAD SMS, get index/position
            s = strstr(gprsBuffer,":");
            if (s != NULL) {
                //We are going to flush serial data until OK is recieved
                sim808_wait_for_resp("OK\r\n", CMD);
                return atoi(s+1);
            }
        } else {
            return -1;

        }
    }
    return -1;
}

bool DFRobot_SIM808::readSMS(int messageIndex, char *message, int length, char *phone, char *datetime)
{
  /* Response is like:
  AT+CMGR=2

  +CMGR: "REC READ","XXXXXXXXXXX","","14/10/09,17:30:17+08"
  SMS text here

  So we need (more or lees), 80 chars plus expected message length in buffer. CAUTION FREE MEMORY
  */

    int i = 0;
    char gprsBuffer[80 + length];
    //char cmd[16];
    char num[4];
    char *p,*p2,*s;

    sim808_check_with_cmd("AT+CMGF=1\r\n","OK\r\n",CMD);
    delay(1000);
    //sprintf(cmd,"AT+CMGR=%d\r\n",messageIndex);
    //sim808_send_cmd(cmd);
    sim808_send_cmd("AT+CMGR=");
    itoa(messageIndex, num, 10);
    sim808_send_cmd(num);
    sim808_send_cmd("\r\n");
    sim808_clean_buffer(gprsBuffer,sizeof(gprsBuffer));
    sim808_read_buffer(gprsBuffer,sizeof(gprsBuffer));

    if(NULL != ( s = strstr(gprsBuffer,"+CMGR:"))){
        // Extract phone number string
        p = strstr(s,",");
        p2 = p + 2; //We are in the first phone number character
        p = strstr((char *)(p2), "\"");
        if (NULL != p) {
            i = 0;
            while (p2 < p) {
                phone[i++] = *(p2++);
            }
            phone[i] = '\0';
        }
        // Extract date time string
        p = strstr((char *)(p2),",");
        p2 = p + 1;
        p = strstr((char *)(p2), ",");
        p2 = p + 2; //We are in the first date time character
        p = strstr((char *)(p2), "\"");
        if (NULL != p) {
            i = 0;
            while (p2 < p) {
                datetime[i++] = *(p2++);
            }
            datetime[i] = '\0';
        }        
        if(NULL != ( s = strstr(s,"\r\n"))){
            i = 0;
            p = s + 2;
            while((*p != '\r')&&(i < length-1)) {
                message[i++] = *(p++);
            }
            message[i] = '\0';
        }
        return true;
    }
    return false;    
}

bool DFRobot_SIM808::readSMS(int messageIndex, char *message,int length)
{
    int i = 0;
    char gprsBuffer[100];
    //char cmd[16];
	char num[4];
    char *p,*s;
	
    sim808_check_with_cmd("AT+CMGF=1\r\n","OK\r\n",CMD);
    delay(1000);
	sim808_send_cmd("AT+CMGR=");
	itoa(messageIndex, num, 10);
	sim808_send_cmd(num);
	sim808_send_cmd("\r\n");
//  sprintf(cmd,"AT+CMGR=%d\r\n",messageIndex);
//    sim808_send_cmd(cmd);
    sim808_clean_buffer(gprsBuffer,sizeof(gprsBuffer));
    sim808_read_buffer(gprsBuffer,sizeof(gprsBuffer),DEFAULT_TIMEOUT);
    if(NULL != ( s = strstr(gprsBuffer,"+CMGR:"))){
        if(NULL != ( s = strstr(s,"\r\n"))){
            p = s + 2;
            while((*p != '\r')&&(i < length-1)) {
                message[i++] = *(p++);
            }
            message[i] = '\0';
            return true;
        }
    }
    return false;   
}

bool DFRobot_SIM808::deleteSMS(int index)
{
    //char cmd[16];
	char num[4];
    //sprintf(cmd,"AT+CMGD=%d\r\n",index);
    sim808_send_cmd("AT+CMGD=");
	itoa(index, num, 10);
	sim808_send_cmd(num);
	//snprintf(cmd,sizeof(cmd),"AT+CMGD=%d\r\n",index);
    //sim808_send_cmd(cmd);
    //return 0;
    // We have to wait OK response
	//return sim808_check_with_cmd(cmd,"OK\r\n",CMD);
	return sim808_check_with_cmd("\r","OK\r\n",CMD);	
}

bool DFRobot_SIM808::callUp(char *number)
{
    //char cmd[24];
    if(!sim808_check_with_cmd("AT+COLP=1\r\n","OK\r\n",CMD)) {
        return false;
    }
    delay(1000);
	//HACERR quitar SPRINTF para ahorar memoria ???
    //sprintf(cmd,"ATD%s;\r\n", number);
    //sim808_send_cmd(cmd);
	sim808_send_cmd("ATD");
	sim808_send_cmd(number);
	sim808_send_cmd(";\r\n");
    return true;
}

void DFRobot_SIM808::answer(void)
{
    sim808_send_cmd("ATA\r\n");  //TO CHECK: ATA doesnt return "OK" ????
}

bool DFRobot_SIM808::hangup(void)
{
    return sim808_check_with_cmd("ATH\r\n","OK\r\n",CMD);
}

bool DFRobot_SIM808::disableCLIPring(void)
{
    return sim808_check_with_cmd("AT+CLIP=0\r\n","OK\r\n",CMD);
}

bool DFRobot_SIM808::getSubscriberNumber(char *number)
{
	//AT+CNUM								--> 7 + CR = 8
	//+CNUM: "","+628157933874",145,7,4		--> CRLF + 45 + CRLF = 49
	//										-->
	//OK									--> CRLF + 2 + CRLF = 6

    byte i = 0;
    char gprsBuffer[65];
    char *p,*s;
	sim808_flush_serial();
    sim808_send_cmd("AT+CNUM\r\n");
    sim808_clean_buffer(gprsBuffer,65);
    sim808_read_buffer(gprsBuffer,65,DEFAULT_TIMEOUT);
	//Serial.print(gprsBuffer);
    if(NULL != ( s = strstr(gprsBuffer,"+CNUM:"))) {
        s = strstr((char *)(s),",");
        s = s + 2;  //We are in the first phone number character 
        p = strstr((char *)(s),"\""); //p is last character """
        if (NULL != s) {
            i = 0;
            while (s < p) {
              number[i++] = *(s++);
            }
            number[i] = '\0';
        }
        return true;
    }  
    return false;
}

bool DFRobot_SIM808::isCallActive(char *number)
```

```
{
    char gprsBuffer[46];  //46 is enough to see +CPAS: and CLCC:
    char *p, *s;
    int i = 0;

    sim808_send_cmd("AT+CPAS\r\n");
    /*Result code:
        0: ready
        2: unknown
        3: ringing
        4: call in progress

      AT+CPAS   --> 7 + 2 = 9 chars
                --> 2 char
      +CPAS: 3  --> 8 + 2 = 10 chars
                --> 2 char
      OK        --> 2 + 2 = 4 chars

      AT+CPAS

      +CPAS: 0

      OK
    */

    sim808_clean_buffer(gprsBuffer,29);
    sim808_read_buffer(gprsBuffer,27);
    //HACERR cuando haga lo de esperar a OK no me haría falta esto
    //We are going to flush serial data until OK is recieved
    sim808_wait_for_resp("OK\r\n", CMD);
    //Serial.print("Buffer isCallActive 1: ");Serial.println(gprsBuffer);
    if(NULL != ( s = strstr(gprsBuffer,"+CPAS:"))) {
      s = s + 7;
      if (*s != '0') {
         //There is something "running" (but number 2 that is unknow)
         if (*s != '2') {
           //3 or 4, let's go to check for the number
           sim808_send_cmd("AT+CLCC\r\n");
           /*
           AT+CLCC --> 9

           +CLCC: 1,1,4,0,0,"656783741",161,""

           OK

           Without ringing:
           AT+CLCC
           OK
           */

           sim808_clean_buffer(gprsBuffer,46);
           sim808_read_buffer(gprsBuffer,45);
           //Serial.print("Buffer isCallActive 2: ");Serial.println(gprsBuffer);
           if(NULL != ( s = strstr(gprsBuffer,"+CLCC:"))) {
             //There is at least one CALL ACTIVE, get number
             s = strstr((char *)(s),"\"");
             s = s + 1;  //We are in the first phone number character
             p = strstr((char *)(s),"\""); //p is last character """
             if (NULL != s) {
                i = 0;
                while (s < p) {
                    number[i++] = *(s++);
                }
                number[i] = '\0';
             }
             //I need to read more buffer
             //We are going to flush serial data until OK is recieved
             return sim808_wait_for_resp("OK\r\n", CMD); 
           }
         }
      }        
    } 
    return false;
}

bool DFRobot_SIM808::getDateTime(char *buffer)
{
    //If it doesn't work may be for two reasons:
    //      1. Your carrier doesn't give that information
    //      2. You have to configurate the SIM808 IC.
    //          - First with SIM808_Serial_Debug example try this AT command: AT+CLTS?
    //          - If response is 0, then it is disabled.
    //          - Enable it by: AT+CLTS=1
    //          - Now you have to save this config to EEPROM memory of SIM808 IC by: AT&W
    //          - Now, you have to power down and power up again the SIM808 
    //          - Try now again: AT+CCLK?
    //          - It should work
    
    //AT+CCLK?                          --> 8 + CR = 9
    //+CCLK: "14/11/13,21:14:41+04"     --> CRLF + 29+ CRLF = 33
    //
    //OK                                --> CRLF + 2 + CRLF =  6

    byte i = 0;
    char gprsBuffer[50];
    char *p,*s;
    sim808_flush_serial();
    sim808_send_cmd("AT+CCLK?\r");
    sim808_clean_buffer(gprsBuffer,50);
    sim808_read_buffer(gprsBuffer,50,DEFAULT_TIMEOUT);
    if(NULL != ( s = strstr(gprsBuffer,"+CCLK:"))) {
        s = strstr((char *)(s),"\"");
        s = s + 1;  //We are in the first phone number character 
        p = strstr((char *)(s),"\""); //p is last character """
        if (NULL != s) {
            i = 0;
            while (s < p) {
              buffer[i++] = *(s++);
            }
            buffer[i] = '\0';            
        }
        return true;
    }  
    return false;
}

bool DFRobot_SIM808::getSignalStrength(int *buffer)
{
    //AT+CSQ                            --> 6 + CR = 10
    //+CSQ: <rssi>,<ber>                --> CRLF + 5 + CRLF = 9
    //OK                                --> CRLF + 2 + CRLF =  6

    byte i = 0;
    char gprsBuffer[26];
    char *p, *s;
    char buffers[4];
    sim808_flush_serial();
    sim808_send_cmd("AT+CSQ\r");
    sim808_clean_buffer(gprsBuffer, 26);
    sim808_read_buffer(gprsBuffer, 26, DEFAULT_TIMEOUT);
    if (NULL != (s = strstr(gprsBuffer, "+CSQ:"))) {
        s = strstr((char *)(s), " ");
        s = s + 1;  //We are in the first phone number character 
        p = strstr((char *)(s), ",");//p is last character """
        if (NULL != s) {
            i = 0;
            while (s < p) {
                buffers[i++] = *(s++);
            }
            buffers[i] = '\0';
        }
        *buffer = atoi(buffers);
        return true;
    }
    return false;
}

bool DFRobot_SIM808::sendUSSDSynchronous(char *ussdCommand, char *resultcode, char *response)
{
    //AT+CUSD=1,"{command}"
    //OK
    //
    //+CUSD:1,"{response}",{int}

    byte i = 0;
    char gprsBuffer[200];
    char *p,*s;
    sim808_clean_buffer(response, sizeof(response));
    
    sim808_flush_serial();
    sim808_send_cmd("AT+CUSD=1,\"");
    sim808_send_cmd(ussdCommand);
    sim808_send_cmd("\"\r");
    if(!sim808_wait_for_resp("OK\r\n", CMD))
        return false;
    sim808_clean_buffer(gprsBuffer,200);
    sim808_read_buffer(gprsBuffer,200,DEFAULT_TIMEOUT);
    if(NULL != ( s = strstr(gprsBuffer,"+CUSD: "))) {
        *resultcode = *(s+7);
        resultcode[1] = '\0';
        if(!('0' <= *resultcode && *resultcode <= '2'))
            return false;
        s = strstr(s,"\"");
        s = s + 1;  //We are in the first phone number character
        p = strstr(s,"\""); //p is last character """
        if (NULL != s) {
            i = 0;
            while (s < p) {
              response[i++] = *(s++);
            }
            response[i] = '\0';            
        }
        return true;
    }
    return false;
}

bool DFRobot_SIM808::cancelUSSDSession(void)
{
    return sim808_check_with_cmd("AT+CUSD=2\r\n","OK\r\n",CMD);
}

//Here is where we ask for APN configuration, with F() so we can save MEMORY
bool DFRobot_SIM808::join(const __FlashStringHelper *apn, const __FlashStringHelper *userName, const __FlashStringHelper *passWord)
{
    byte i;
    char *p, *s;
    char ipAddr[32];
    //Select multiple connection
    //sim808_check_with_cmd("AT+CIPMUX=1\r\n","OK",DEFAULT_TIMEOUT,CMD);

    //set APN. OLD VERSION
    //snprintf(cmd,sizeof(cmd),"AT+CSTT=\"%s\",\"%s\",\"%s\"\r\n",_apn,_userName,_passWord);
    //sim808_check_with_cmd(cmd, "OK\r\n", DEFAULT_TIMEOUT,CMD);
    sim808_send_cmd("AT+CSTT=\"");
    if (apn) {
      sim808_send_cmd(apn);
    }
    sim808_send_cmd("\",\"");
    if (userName) {
      sim808_send_cmd(userName);
    }
    sim808_send_cmd("\",\"");
    if (passWord) {
      sim808_send_cmd(passWord);
    }
    sim808_check_with_cmd("\"\r\n", "OK\r\n", CMD);
    

    //Brings up wireless connection
    sim808_check_with_cmd("AT+CIICR\r\n","OK\r\n", CMD);

    //Get local IP address
    sim808_send_cmd("AT+CIFSR\r\n");
    sim808_clean_buffer(ipAddr,32);
    sim808_read_buffer(ipAddr,32);
    //Response:
    //AT+CIFSR\r\n        -->  8 + 2
    //\r\n                 -->  0 + 2
    //10.160.57.120\r\n  --> 15 + 2 (max)   : TOTAL: 29
    //Response error:
    //AT+CIFSR\r\n
    //\r\n
    //ERROR\r\n
    if (NULL != strstr(ipAddr,"ERROR")) {
        return false;
    }
    s = ipAddr + 11;
    p = strstr((char *)(s),"\r\n"); //p is last character \r\n
    if (NULL != s) {
        i = 0;
        while (s < p) {
            ip_string[i++] = *(s++);
        }
        ip_string[i] = '\0';
    }
    _ip = str_to_ip(ip_string);
    if(_ip != 0) {
        return true;
    }
    return false;
}

void DFRobot_SIM808::disconnect()
{
    sim808_send_cmd("AT+CIPSHUT\r\n");
}

bool DFRobot_SIM808::connect(Protocol ptl,const char * host, int port, int timeout, int chartimeout)
{
    //char cmd[64];
    char num[4];
    char resp[96];
```

```cpp

    //sim808_clean_buffer(cmd,64);
    if(ptl == TCP) {
        sim808_send_cmd("AT+CIPSTART=\"TCP\",\"");
        sim808_send_cmd(host);
        sim808_send_cmd("\",");
        itoa(port, num, 10);
        sim808_send_cmd(num);
        sim808_send_cmd("\r\n");
//        sprintf(cmd, "AT+CIPSTART=\"TCP\",\"%s\",%d\r\n",host, port);
    } else if(ptl == UDP) {
        sim808_send_cmd("AT+CIPSTART=\"UDP\",\"");
        sim808_send_cmd(host);
        sim808_send_cmd("\",");
        itoa(port, num, 10);
        sim808_send_cmd(num);
        sim808_send_cmd("\r\n");

    //        sprintf(cmd, "AT+CIPSTART=\"UDP\",\"%s\",%d\r\n",host, port);
    } else {
        return false;
    }
    

    //sim808_send_cmd(cmd);
    sim808_read_buffer(resp, 96, timeout, chartimeout);
    //Serial.print("Connect resp: "); Serial.println(resp);    
    if(NULL != strstr(resp,"CONNECT")) { //ALREADY CONNECT or CONNECT OK
        return true;
    }
    return false;
}

//Overload with F() macro to SAVE memory
bool DFRobot_SIM808::connect(Protocol ptl,const __FlashStringHelper *host, const __FlashStringHelper *port, int timeout, int chartimeout)
{
    //char cmd[64];
    char resp[96];

    //sim808_clean_buffer(cmd,64);
    if(ptl == TCP) {
        sim808_send_cmd(F("AT+CIPSTART=\"TCP\",\""));   //%s\",%d\r\n",host, port);
    } else if(ptl == UDP) {
        sim808_send_cmd(F("AT+CIPSTART=\"UDP\",\""));   //%s\",%d\r\n",host, port);
    } else {
        return false;
    }
    sim808_send_cmd(host);
    sim808_send_cmd(F("\","));
    sim808_send_cmd(port);
    sim808_send_cmd(F("\r\n"));
//  Serial.print("Connect: "); Serial.println(cmd);
    sim808_read_buffer(resp, 96, timeout, chartimeout);
//  Serial.print("Connect resp: "); Serial.println(resp);    
    if(NULL != strstr(resp,"CONNECT")) { //ALREADY CONNECT or CONNECT OK
        return true;
    }
    return false;
}

bool DFRobot_SIM808::is_connected(void)
{
    char resp[96];
    sim808_send_cmd("AT+CIPSTATUS\r\n");
    sim808_read_buffer(resp,sizeof(resp),DEFAULT_TIMEOUT);
    if(NULL != strstr(resp,"CONNECTED")) {
        //+CIPSTATUS: 1,0,"TCP","216.52.233.120","80","CONNECTED"
        return true;
    } else {
        //+CIPSTATUS: 1,0,"TCP","216.52.233.120","80","CLOSED"
        //+CIPSTATUS: 0,,"","","","INITIAL"
        return false;
    }
}

bool DFRobot_SIM808::close()
{
    // if not connected, return
    if (!is_connected()) {
        return true;
    }
    return sim808_check_with_cmd("AT+CIPCLOSE\r\n", "CLOSE OK\r\n", CMD);
}

int DFRobot_SIM808::readable(void)
{
    return sim808_check_readable();
}

int DFRobot_SIM808::wait_readable(int wait_time)
{
    return sim808_wait_readable(wait_time);
}

int DFRobot_SIM808::wait_writeable(int req_size)
{
    return req_size+1;
}

int DFRobot_SIM808::send(const char * str, int len)
{
    //char cmd[32];
    char num[4];
    if(len > 0){
        //snprintf(cmd,sizeof(cmd),"AT+CIPSEND=%d\r\n",len);
        //sprintf(cmd,"AT+CIPSEND=%d\r\n",len);
        sim808_send_cmd("AT+CIPSEND=");
        itoa(len, num, 10);
        sim808_send_cmd(num);
        if(!sim808_check_with_cmd("\r\n",">",CMD)) {
        //if(!sim808_check_with_cmd(cmd,">",CMD)) {
            return 0;
        }
        /*if(0 != sim808_check_with_cmd(str,"SEND OK\r\n", DEFAULT_TIMEOUT * 10 ,DATA)) {
            return 0;
        }*/
        delay(500);
        sim808_send_cmd(str);
        delay(500);
        sim808_send_End_Mark();
        if(!sim808_wait_for_resp("SEND OK\r\n", DATA, DEFAULT_TIMEOUT * 10, DEFAULT_INTERCHAR_TIMEOUT * 10)) {
            return 0;
        }
    }
    return len;
}


int DFRobot_SIM808::recv(char* buf, int len)
{
    sim808_clean_buffer(buf,len);
    sim808_read_buffer(buf,len);   //Ya he llamado a la funcion con la longitud del buffer - 1 y luego le estoy añadiendo el 0
    return strlen(buf);
}

void DFRobot_SIM808::listen(void)
{
	 if(serialFlag)
		; //hgprsSerial->listen();
	 else
		 gprsSerial->listen();

}

bool DFRobot_SIM808::isListening(void)
{
	// if(serialFlag)
		// return hgprsSerial.isListening();
	// else
		// return gprsSerial.isListening();
}

uint32_t DFRobot_SIM808::str_to_ip(const char* str)
{
    uint32_t ip = 0;
    char* p = (char*)str;
    for(int i = 0; i < 4; i++) {
        ip |= atoi(p);
        p = strchr(p, '.');
        if (p == NULL) {
            break;
        }
        ip <<= 8;
        p++;
    }
    return ip;
}

char* DFRobot_SIM808::getIPAddress()
{
    //I have already a buffer with ip_string: snprintf(ip_string, sizeof(ip_string), "%d.%d.%d.%d", (_ip>>24)&0xff,(_ip>>16)&0xff,(_ip>>8)&0xff,_ip&0xff); 
    return ip_string;
}

unsigned long DFRobot_SIM808::getIPnumber()
{
    return _ip;
}
/* NOT USED bool DFRobot_SIM808::gethostbyname(const char* host, uint32_t* ip)
{
    uint32_t addr = str_to_ip(host);
    char buf[17];
    //snprintf(buf, sizeof(buf), "%d.%d.%d.%d", (addr>>24)&0xff, (addr>>16)&0xff, (addr>>8)&0xff, addr&0xff);
    if (strcmp(buf, host) == 0) {
        *ip = addr;
        return true;
    }
    return false;
}
*/

bool DFRobot_SIM808::getLocation(const __FlashStringHelper *apn, float *longitude, float *latitude)
{    	
    int i = 0;
    char gprsBuffer[80];
    char buffer[20];
    char *s;
    
    //send AT+SAPBR=3,1,"Contype","DFRobot_SIM808"
    sim808_check_with_cmd("AT+SAPBR=3,1,\"Contype\",\"DFRobot_SIM808\"\r","OK\r\n",CMD);
    //sen AT+SAPBR=3,1,"APN","DFRobot_SIM808_APN"
    sim808_send_cmd("AT+SAPBR=3,1,\"APN\",\"");
    if (apn) {
      sim808_send_cmd(apn);
    }
    sim808_check_with_cmd("\"\r","OK\r\n",CMD);
    //send AT+SAPBR =1,1
    sim808_check_with_cmd("AT+SAPBR=1,1\r","OK\r\n",CMD);
    
    //AT+CIPGSMLOC=1,1
    sim808_flush_serial();
    sim808_send_cmd("AT+CIPGSMLOC=1,1\r");
    sim808_clean_buffer(gprsBuffer,sizeof(gprsBuffer));
    sim808_read_buffer(gprsBuffer,sizeof(gprsBuffer),2*DEFAULT_TIMEOUT,6*DEFAULT_INTERCHAR_TIMEOUT);
    //Serial.println(gprsBuffer);
    
    if(NULL != ( s = strstr(gprsBuffer,"+CIPGSMLOC:")))
    {
        s = strstr((char *)s, ",");
        s = s+1;
        //Serial.println(*s);
        i=0;
        while(*(++s) !=  ',')
            buffer[i++]=*s;
        buffer[i] = 0;
        *longitude = atof(buffer);
        
        i=0;
        while(*(++s) !=  ',')
            buffer[i++]=*s;
        buffer[i] = 0;
        *latitude = atof(buffer);
        return true;
    }
    return false;
}

bool DFRobot_SIM808::attachGPS()
{
     if(!sim808_check_with_cmd("AT+CGNSPWR=1\r\n", "OK\r\n", CMD)) { 
        return false;
    }
     if(!sim808_check_with_cmd("AT+CGNSTST=1\r\n", "OK\r\n", CMD)) { 
        return false;
    }
    return true;
}

bool DFRobot_SIM808::detachGPS()
{
     if(!sim808_check_with_cmd("AT+CGNSPWR=0\r\n", "OK\r\n", CMD)) { 
        return false;
    }
    return true;
}

bool DFRobot_SIM808::getGPRMC()
{
    char c;
    static bool endflag  = false;
    static char count;

    while(serialSIM808->available())   //??????????
    {   c = serialSIM808->read();
        if(endflag)
        {
            if(count--)
            {
                receivedStack[receivedStackIndex++] = c;
            }
            else{
                endflag = false;
                receivedStack[receivedStackIndex] = '\0';
                return true;
            }
        }
        else
        {
            switch(c)
            {

                case '$':
                    receivedStackIndex = 0;
                    receivedStack[receivedStackIndex++] = c;
                    break;
                case '*':
                    endflag = true;
                    count = 2;
                    receivedStack[receivedStackIndex++] = c;
                    break;
                default:
                    if(receivedStackIndex < 120)
                        receivedStack[receivedStackIndex++] = c;
                    break;

            }
            return false;
        }
        return false;
    }
    return false;
}

bool DFRobot_SIM808::parseGPRMC(char *gpsbuffer)
{
    if(strstr(gpsbuffer,des) == NULL)  //????$GPRMC??????????GPS???
    {
        receivedStackIndex = 0;
        return false;
    }
    else
    {
        if(gpsbuffer[18] == 'A')        //?????????????
            return true;
        else
        {
            //Serial.print("NO :");
            //Serial.println(gpsbuffer[18]);
            return false;
        }

    }

}
```

```
// Parse a (potentially negative) number with up to 2 decimal digits -xxxx.yy
int32_t DFRobot_SIM808::parseDecimal(const char *term)
{
  bool negative = *term == '-';
  if (negative) ++term;
  int32_t ret = 100 * (int32_t)atol(term);
  while (isdigit(*term)) ++term;
  if (*term == '.' && isdigit(term[1]))
  {
    ret += 10 * (term[1] - '0');
    if (isdigit(term[2]))
      ret += term[2] - '0';
  }
  return negative ? -ret : ret;
}

 void DFRobot_SIM808::getTime(uint32_t time){
	 GPSdata.hour     =  time / 1000000;
	 GPSdata.minute  = (time / 10000) % 100;
	 GPSdata.second = (time / 100) % 100;
	 GPSdata.centisecond =  time % 100;
 }

 
  void DFRobot_SIM808::getDate(uint32_t date){
	 uint16_t year = date % 100;
	 GPSdata. year    =  year + 2000;	 
	 GPSdata.month  = (date / 100) % 100;
	 GPSdata.day = date / 10000;
 }
 
 
bool DFRobot_SIM808::getGPS() 
{
	 if(!getGPRMC())  //û??õ?$GPRMC??????????GPS???
		 return false;
	// Serial.println(receivedStack);
	 if(!parseGPRMC(receivedStack))  //????$GPRMC??????????GPS???
		 return false;  
		
	// skip mode
    char *tok = strtok(receivedStack, ",");     //??'?????? 
    if (! tok) return false;

   // grab time                                  //<1> UTC???????hhmmss.sss??
   // tok = strtok(NULL, ",");
	char *time = strtok(NULL, ",");
    if (! time) return false;
	uint32_t newTime = (uint32_t)parseDecimal(time);
	getTime(newTime);

    // skip fix
    tok = strtok(NULL, ",");              //<2> ???????A=????????V=??????
    if (! tok) return false;

    // grab the latitude
    char *latp = strtok(NULL, ",");   //<3> ???ddmm.mmmm(???)???(j???0?????????)
    if (! latp) return false;

    // grab latitude direction              // <4> ??????N(??????)??S(?????)
    char *latdir = strtok(NULL, ",");
    if (! latdir) return false;

    // grab longitude                          //<5> ????dddmm.mmmm(???)???(j???0?????????)
    char *longp = strtok(NULL, ",");
    if (! longp) return false;

    // grab longitude direction            //<6> ???????E(????)??W(????)
    char *longdir = strtok(NULL, ",");
    if (! longdir) return false;

    double latitude = atof(latp);
    double longitude = atof(longp);

    // convert latitude from minutes to decimal
    float degrees = floor(latitude / 100);
    double minutes = latitude - (100 * degrees);
    minutes /= 60;
    degrees += minutes;

    // turn direction into + or -
    if (latdir[0] == 'S') degrees *= -1;

    //*lat = degrees;
	GPSdata.lat = degrees;

    // convert longitude from minutes to decimal
    degrees = floor(longitude / 100);
    minutes = longitude - (100 * degrees);
    minutes /= 60;
    degrees += minutes;
	

    // turn direction into + or -
    if (longdir[0] == 'W') degrees *= -1;

    //*lon = degrees;
	GPSdata.lon= degrees;

    // only grab speed if needed                  //<7> ????????(000.0~999.9???j???0?????????)
   // if (speed_kph != NULL) {

      // grab the speed in knots
      char *speedp = strtok(NULL, ",");
      if (! speedp) return false;

      // convert to kph
      //*speed_kph = atof(speedp) * 1.852;
	  GPSdata.speed_kph= atof(speedp) * 1.852;

   // }

    // only grab heading if needed             //??????(000.0~359.9????????????????j???0?????????)
   // if (heading != NULL) {

      // grab the speed in knots
      char *coursep = strtok(NULL, ",");
      if (! coursep) return false;

      //*heading = atof(coursep);
		GPSdata.heading = atof(coursep);
   // }
	
	 // grab date
	char *date = strtok(NULL, ",");   //<3> ???ddmm.mmmm(???)???(j???0?????????)
    if (! date) return false;
	uint32_t newDate = atol(date);
	getDate(newDate);

    // no need to continue
   // if (altitude == NULL){
   //   return true;
	//}
	return true;
}
```