```
1 /*!
 2 ***@file*DFRobot_sim808.cpp
 3 ***@A*library··for*DFRobot's*SIM808*GPS/DFRobot_SIM808/GSM*Shield
 4 •*
5 •*•@copyright→[DFRobot](http://www.dfrobot.com),•2016
 6 •*
 7 ***@author*[Jason](jason.ling@dfrobot.com)
8 ** @version • V1.0
9 ** @date · · 2016-09-23
10 •
11 ** The MIT License (MIT)
12 ***Permission*is*hereby*granted,*free*of*charge,*to*any*person*obtaining*a*copy
13 ***of*this*software*and*associated*documentation*files*(the*"Software"),*to*deal
14 **·in·the·Software·without·restriction, including without·limitation the rights
15 ***to*use,*copy,*modify,*merge,*publish,*distribute,*sublicense,*and/or*sell
16 ** copies of the Software, and to permit persons to whom the Software is
17 ** furnished to do so, subject to the following conditions:
19 ***The above copyright notice and this permission notice shall be included in
20 ** all copies or substantial portions of the Software.
21 •*
22 ***THE*SOFTWARE*IS*PROVIDED*"AS*IS",*WITHOUT*WARRANTY*OF*ANY*KIND,*EXPRESS*OR
23 ***IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
24 ***FITNESS*FOR*A*PARTICULAR*PURPOSE*AND*NONINFRINGEMENT.*IN*NO*EVENT*SHALL*THE
25 **-AUTHORS-OR-COPYRIGHT-HOLDERS-BE-LIABLE-FOR-ANY-CLAIM, DAMAGES-OR-OTHER
26 ***LIABILITY, "WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
27 ***OUT*OF*OR*IN*CONNECTION*WITH*THE*SOFTWARE*OR*THE*USE*OR*OTHER*DEALINGS*IN
28 ** THE SOFTWARE.
29 •*/
30 •
31
32 #include < stdio.h>
33 #include "DFRobot sim808.h"
35 extern Stream ** serial SIM808;
36
37 DFRobot SIM808**DFRobot SIM808::inst;
38 char*receivedStackIndex*=*0;
39 char*receivedStack[130];
40 const char *des = "$GPRMC";
41
42 //char**receivedStack="$GPRMC,165445.000,A,3110.8635,N,12133.4627,E,0.58,70.26,220916,,,A*57";
43
44 DFRobot SIM808::DFRobot SIM808(HardwareSerial **mySerial)
45 {
46 ....inst = this; -
47 ──serialFlag = 1;
48 —→hgprsSerial = mySerial;
49 ····sim808_init(mySerial, 1);
50 }
51
52 DFRobot_SIM808::DFRobot_SIM808(SoftwareSerial**mySerial)
53 {
57 ***sim808_init(mySerial, *0);
58 }
59
60 bool DFRobot_SIM808::init(void)
61 {
62 ····//???AT?????????
63 ──if(!sim808_check_with_cmd("AT\r\n","OK\r\n",CMD)){····
64 ------
65 → return•false;
66 ------
```

```
67 ....}
 68 →//???SIM????????????????
 70 ···if(!sim808_check_with_cmd("AT+CFUN=1\r\n","OK\r\n",CMD)){
 71 ····return•false;
 72 · · · · }
 73
 74 → //???SIM????
 75 ····if(!checkSIMStatus()) •{
 76 → return•false;
 77 · · · · }
 78 ····return•true;
 79 }
 80
 81 bool DFRobot_SIM808::checkPowerUp(void)
 83 • return sim808_check_with_cmd("AT\r\n", "OK\r\n", CMD);
 84 }
 85
 86 void DFRobot_SIM808::powerUpDown(uint8_t pin)
 87 {
 88 ··//*power*on*pulse*for*SIM900*Shield
 90 ··delay(1000);
 91  ··digitalWrite(pin,HIGH);
 92 ··delay(2000);
 94 ··delay(3000);
95 }
96
 97 void DFRobot SIM808::powerReset(uint8 t pin)
98 {
99 **//*reset*for*SIM800L*board.
100 --//*RST*pin*has*to*be*OUTPUT,*HIGH
101 ··digitalWrite(pin,LOW);
102 ··delay(1000);
104 ···delay(3000); ···
105 }
106 ...
107 ..
108 bool DFRobot_SIM808::checkSIMStatus(void)
110 ····char•gprsBuffer[32];
111 ....int.count.=.0;
113 ••••while(count•<•3)•{</pre>
115 ....sim808_read_buffer(gprsBuffer,32,DEFAULT_TIMEOUT);
116 ....if((NULL*!="strstr(gprsBuffer,"+CPIN:"READY")))"{..//???SIM??????
117 ....break;
118 .....
119 · · · · · · count++;
120 ·····delay(300);
121 ....}
123 ····return•false;
124 ....}
125 ····return•true;
126 }
127
128 bool DFRobot_SIM808::sendSMS(char * number, * char * * data)
130 ···//char*cmd[32];
131 ····if(!sim808_check_with_cmd("AT+CMGF=1\r\n", "OK\r\n", "CMD)) - {-//-Set-message-mode-to-ASCII
132 ····return false;
```

```
133 ....}
134 ····delay(500);
138 ····//sprintf(cmd, "AT+CMGS=\"%s\"\r\n", number);
140 //...if(!sim808_check_with_cmd(cmd,">",CMD)) {
141 ....if(!sim808_check_with_cmd("\"\r\n",">",CMD)) • {
142 ····return•false;
143 ....}
144 ····delay(1000);
145 ····sim808_send_cmd(data);
146 ····delay(500);
147 ····sim808_send_End_Mark();
148 ····return*sim808_wait_for_resp("OK\r\n",*CMD);
149 }
150
151 char*DFRobot_SIM808::isSMSunread()
153 ****char*gprsBuffer[48];***//48*is*enough*to*see*+CMGL:
154 ····char**s;
155 ....
156 --- sim808_check_with_cmd("AT+CMGF=1\r\n","OK\r\n",CMD);
157 ····delay(1000);
159 ····//List.of.all.UNREAD.SMS.and.DON'T.change.the.SMS.UNREAD.STATUS
160 ****sim808_send_cmd(F("AT+CMGL=\"REC*UNREAD\",1\r\n"));
161 ····/*If "you" want "to "change "SMS" status "to "READ "you" will "need "to "send:
162 *******AT+CMGL=\"REC*UNREAD\"\r\n
163 ······This command will list all UNREAD SMS and change all of them to READ
164 .....
165 ....If there is not SMS, response is (30 chars)
166 .....AT+CMGL="REC*UNREAD",1..--> 22*+2
169
170 ·····If there is SMS, response is like (>64 chars)
171 ....AT+CMGL="REC*UNREAD",1
172 ·····+CMGL:*9, "REC*UNREAD", "XXXXXXXXXX", "", "14/10/16, 21:40:08+08"
173 ······Here SMS text.
174 ....OK..
175 .....
176 ....or
177
178 ·····AT+CMGL="REC*UNREAD",1
179 ······+CMGL:*9,"REC*UNREAD","XXXXXXXXX","","14/10/16,21:40:08+08"
180 ·····Here SMS text.
181 ······+CMGL: 10, "REC*UNREAD", "YYYYYYYY", "", "14/10/16, 21:40:08+08"
182 ······Here second SMS ·····
183 · · · · · · · OK · · · · · · · · ·
184 · · · · */
185
186 ····sim808_clean_buffer(gprsBuffer,31);
187 ....sim808_read_buffer(gprsBuffer,30,DEFAULT_TIMEOUT);
188 ····//Serial.print("Buffer isSMSunread: ");Serial.println(gprsBuffer);
189
190 ····if(NULL•!=•(•s•=•strstr(gprsBuffer, "OK")))•{
191 ·····//In-30-bytes-"doesn't"-fit-whole-+CMGL:-response,-if-recieve-only-"OK"
192 ·····//···means.you.don't.have.any.UNREAD.SMS
193 ·····delay(50);
194 •••••return•0;
195 ····}•else•{
196 ·····//More buffer to read
197 ·····//We-are-going-to-flush-serial-data-until-OK-is-recieved
198 ·····sim808_wait_for_resp("OK\r\n", •CMD); ······
```

```
199 .....//sim808 flush serial();
200 ·····//We-have-to-call-command-again
201 ....sim808_send_cmd("AT+CMGL=\"REC=UNREAD\",1\r\n");
202 ·····sim808_clean_buffer(gprsBuffer,48);
203 ....sim808_read_buffer(gprsBuffer,47,DEFAULT_TIMEOUT);
204 →→//Serial.print("Buffer is SMS unread 2: "); Serial.println(gprs Buffer); · · · · · ·
205 ·····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CMGL:")))*{
206 ·····//There is at least one UNREAD SMS, get index/position
207 ....s==strstr(gprsBuffer,":");
208 ....if (s != NULL) {
209 ·····//We-are-going-to-flush-serial-data-until-OK-is-recieved
210 ....sim808_wait_for_resp("OK\r\n", CMD);
211 ·····return•atoi(s+1);
212 .....
213 ······} •else • {
214 ····return -1;
215
217 ••••}•
218 ····return--1;
219 }
220
221 bool*DFRobot_SIM808::readSMS(int*messageIndex,*char**message,*int*length,*char**phone,*char**datetime)**
223 ··/* Response is like:
224 • AT+CMGR=2
225 ...
226 --+ CMGR: "REC - READ", "XXXXXXXXXXXX", "", "14/10/09, 17:30:17+08"
227 •• SMS • text • here
229 ··So·we·need·(more·or·lees), *80·chars·plus·expected·message·length·in·buffer. CAUTION·FREE·MEMORY
230 ..*/
231
232 ....int:i:=:0;
233 ****char*gprsBuffer[80*+*length];
234 ····//char*cmd[16];
235 —>char*num[4];
236 ••••char•*p,*p2,*s;
237 ....
238 ****sim808_check_with_cmd("AT+CMGF=1\r\n","OK\r\n",CMD);
239 ····delay(1000);
240 →//sprintf(cmd,"AT+CMGR=%d\r\n",messageIndex);
241 ..../sim808_send_cmd(cmd);
246 ....sim808_clean_buffer(gprsBuffer,sizeof(gprsBuffer));
247 ····sim808_read_buffer(gprsBuffer,sizeof(gprsBuffer));
249 ····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CMGR:"))){
250 ·····//•Extract•phone•number•string
251 .....p==strstr(s,",");
252 ·····p2·=·p·+·2; ·//We·are·in·the·first·phone·number·character
253 ·····p==strstr((char**)(p2), "\"");
254 ·····if (NULL != p) {
255 ······i·=·0;
256 ·····while (p2 < p) {
257 ·····phone[i++]==**(p2++);
258 .....
259 ·····phone[i]=='\0';······
260 ......
261 ·····//•Extract•date•time•string
262 ******p*=*strstr((char**)(p2),",");
263 ·····p2·=·p·+·1; ·
264 ******p*=*strstr((char**)(p2),*",");*
```

```
265 ·····p2*=*p*+*2;*//We*are*in*the*first*date*time*character
266 ·····p==*strstr((char**)(p2),*"\"");
267 ·····if (NULL != p) {
268 ······i·=·0;
269 ·····while (p2 < p) {
270 .....datetime[i++] = * * (p2++);
271 .....}
272 ·····datetime[i] •=• '\0';
273 .....}....
274 ·····if(NULL•!=•(•s•=•strstr(s,"\r\n"))){
275 ·····i·=•0;
276 · · · · · · · · · p = · s · + · 2;
277 ·····while((*p*!=*'\r')&&(i*<*length-1))*{
278 .....message[i++] == *(p++);
279 · · · · · · · · · }
280 ·····message[i] = '\0';
281 .....
282 ····return•true;
283 · · · · }
284 ····return•false;····
287 bool*DFRobot_SIM808::readSMS(int*messageIndex,*char**message,int*length)
289 ....int:i:=:0;
290 ····char•gprsBuffer[100];
291 ····//char cmd[16];
292 \longrightarrow char \cdot num[4];
293 ····char**p,*s;
295 ····sim808 check with cmd("AT+CMGF=1\r\n","OK\r\n",CMD);
296 ····delay(1000);
297 → sim808 send cmd("AT+CMGR=");
298 →itoa(messageIndex, num, 10);
299 → sim808_send_cmd(num);
300 → sim808_send_cmd("\r\n");
301 // sprintf(cmd, "AT+CMGR=%d\r\n", messageIndex);
302 //****sim808_send_cmd(cmd);
303 ····sim808_clean_buffer(gprsBuffer,sizeof(gprsBuffer));
304 ····sim808_read_buffer(gprsBuffer,sizeof(gprsBuffer),DEFAULT_TIMEOUT);
305 ····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CMGR:"))){
306 ·····if(NULL*!=*(*s*=*strstr(s,"\r\n"))){
307 ·····p==s++2;
308 ·····while((*p"!="'\r')&&(i"<"length-1))"{
309 ····*message[i++] = **(p++);
310 .....
311 ....message[i] = '\0';
312 ····return•true;
313 .....
314 · · · · }
315 ····return•false;···
316 }
317
318 bool*DFRobot_SIM808::deleteSMS(int*index)
319 {
320 ···//char cmd[16];
321 \longrightarrow char \cdot num[4];
322 ····//sprintf(cmd,"AT+CMGD=%d\r\n",index);
323 ****sim808_send_cmd("AT+CMGD=");
\rightarrowitoa(index, num, 10);
325 → sim808_send_cmd(num);
326 \rightarrow\//snprintf(cmd,sizeof(cmd),"AT+CMGD=%d\r\n",index);
327 ..../sim808_send_cmd(cmd);
328 ···//return•0;
329 ····//•We•have•to•wait•OK•response
330 →//return•sim808_check_with_cmd(cmd, "OK\r\n", CMD);
```

```
\rightarrowreturn•sim808 check with cmd("\r","OK\r\n",CMD);\rightarrow
331 -
332 }
333
334 bool DFRobot SIM808::callUp(char *number)
335 {
336 ....//char*cmd[24];
337 ···if(!sim808_check_with_cmd("AT+COLP=1\r\n","OK\r\n",CMD)) = {
338 ····return•false;
339 ....}
340 ····delay(1000);
341 → //HACERR quitar SPRINTF para ahorar memoria ????
342 ····//sprintf(cmd,"ATD%s;\r\n",•number);
343 ..../sim808_send_cmd(cmd);
344 \longrightarrow sim808\_send\_cmd("ATD");
345 \longrightarrow sim808\_send\_cmd(number);
346 \longrightarrow sim808\_send\_cmd(";\r\n");
347 ····return•true;
348 }
349
350 void DFRobot_SIM808::answer(void)
352 ····sim808_send_cmd("ATA\r\n"); ··//TO-CHECK: ATA-doesnt-return-"OK"-????
353 }
355 bool DFRobot_SIM808::hangup(void)
357 ····return sim808_check_with_cmd("ATH\r\n","OK\r\n",CMD);
359
360 bool DFRobot SIM808::disableCLIPring(void)
362 ***return*sim808 check with cmd("AT+CLIP=0\r\n","OK\r\n",CMD);
363 }
365 bool*DFRobot SIM808::getSubscriberNumber(char**number)
      \rightarrow//AT+CNUM\longrightarrow
                                   → → → → - - > • 7 • + • CR • = • 8
369 \longrightarrow // \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow
370 →//OK-
                                          371
372 ····byte·i·=·0;
373 ····char•gprsBuffer[65];
374 ••• char **p, *s;
375 → sim808_flush_serial();
376 ····sim808_send_cmd("AT+CNUM\r\n");
377 ····sim808_clean_buffer(gprsBuffer,65);
378 ····sim808_read_buffer(gprsBuffer,65,DEFAULT_TIMEOUT);
379 →//Serial.print(gprsBuffer);
380 ····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CNUM:")))*{
381 ....s="strstr((char**)(s),",");
382 ·····s==s++2; ··//We-are-in-the-first-phone-number-character-
383 ·····p==*strstr((char**)(s),"\"");*//p*is*last*character*"""
384 ·····if (NULL != s) {
385 ·····i·=·0;
386 ·····while (s < p) {
387 ·····number[i++]---*(s++);
388 ....}
389 ....number[i] = '\0';
390 .....}
391 ····return•true;
392 ....}..
393 ····return•false;
394 }
395
396 bool DFRobot_SIM808::isCallActive(char *number)
```

```
397 {
398 ····char*gprsBuffer[46]; ··//46*is*enough*to*see*+CPAS:*and*CLCC:
399 ••••char•*p,•*s;
400 ····int·i·=·0;
401
402 ····sim808_send_cmd("AT+CPAS\r\n");
403 ****/*Result*code:
404 · · · · · · · · 0: ready
405 ....2: unknown
406 .....3: ringing
407 ·····4: call in progress
408 ....
409 .....AT+CPAS...-> .7.+.2.=.9.chars
411 ....+CPAS: 3...-> 8.+.2.=.10.chars
412 ****************2*char
413 ·····OK·····--> ·2 · + · 2 · = · 4 · chars
414 ....
415 ·····AT+CPAS
416 .....
417 ----+CPAS: 0
418 .....
419 ·····OK
420 ****/
421
423 ****sim808_read_buffer(gprsBuffer,27);
424 ····//HACERR cuando haga lo de esperar a OK no me haría falta esto
425 ····//We are going to flush serial data until OK is recieved
426 ····sim808 wait for resp("OK\r\n", CMD); ····
427 ····//Serial.print("Buffer is Call Active 1: "); Serial.println(gprs Buffer);
428 ····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CPAS:")))*{
429 · · · · · · s · = · s · + · 7;
430 ******if*(*s*!=*'0')*{
431 ·····//There is something "running" (but number 2 that is unknow)
433 ·····//3*or*4, *let's*go*to*check*for*the*number
434 ....sim808 send cmd("AT+CLCC\r\n");
435 ..../*
436 ....AT+CLCC*-->*9
437 ......
438 ·····+CLCC: 1,1,4,0,0,"656783741",161,""
439 .....
440 · · · · · · · · OK · ·
442 ·····Without ringing:
443 .....AT+CLCC
444 · · · · · · · OK · · · · · · · · ·
445 ----*/
447 ....sim808_clean_buffer(gprsBuffer,46);
448 ····sim808_read_buffer(gprsBuffer,45);
449 →→→//Serial.print("Buffer•isCallActive•2:•");Serial.println(gprsBuffer);
450 ·····if(NULL•!=•(•s•=•strstr(gprsBuffer,"+CLCC:")))•{
451 ·····//There is at least one CALL ACTIVE, get number
453 ······s-=·s-+-1; ··//We-are-in-the-first-phone-number-character-····
454 .....p==strstr((char**)(s),"\""); //p=is=last=character="""
455 ....if (NULL != s) {
456 ·····i·=·0;
457 ....while (s < p) {
458 .....number[i++] = -*(s++);
459 .....}
460 ....number[i] == '\0'; .....
461 .....
462 ·····//I•need•to•read•more•buffer
```

```
463 ·····//We-are-going-to-flush-serial-data-until-OK-is-recieved
464 ·····return•sim808_wait_for_resp("OK\r\n",•CMD);•
465 .....
466 .....
467 .....}....
468 ••••}
469 ···return false;
470 }
471
472 bool DFRobot_SIM808::getDateTime(char *buffer)
           →//If it doesn't work may be for two reasons:
475 \longrightarrow // \longrightarrow 1. "Your"carrier"doesn't give that information
476 \longrightarrow // \longrightarrow 2. "You have to configurate the SIM808 IC.
477 →//→→→---First-with-SIM808_Serial_Debug-example-try-this-AT-command:-AT+CLTS?
478 \longrightarrow //\rightarrow \longrightarrow -•If•response•is•0,•then•it•is•disabled.
479 →//→→→--•Enable•it•by:•AT+CLTS=1
480 →//→→→--*Now*you*have*to*save*this*config*to*EEPROM*memory*of*SIM808*IC*by:*AT&W
481 \longrightarrow // \longrightarrow \longrightarrow -\text{"Now,"} you\text{"have"to"power"down"and"power"up"again"the "SIM808" in the "SIM
482 →//→→→--Try=now=again:=AT+CCLK?
483 \longrightarrow //\rightarrow \longrightarrow - • It • should • work
487 \longrightarrow //\rightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow
489
490 ····byte·i·=·0;
491 ····char*gprsBuffer[50];
492 ••••char•*p,*s;
493 \longrightarrow sim808 flush serial();
494 ....sim808 send cmd("AT+CCLK?\r");
496 ....sim808_read_buffer(gprsBuffer,50,DEFAULT_TIMEOUT);
497 ····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CCLK:")))*{
498 *******s*=*strstr((char**)(s),"\"");
499 ·····s==s=+1;··//We=are=in=the=first=phone=number=character=
500 .....p==strstr((char**)(s),"\"");"//p*is*last*character*"""
501 ·····if (NULL != s) {
502 ·····i·=·0;
503 ·····while (s < p) {
504 ·····buffer[i++] = * *(s++);
505 .....}
506 .....buffer[i] = '\0'; .....
507 .....
508 ····return•true;
509 ....}..
510 ····return•false;
511 }
513 bool DFRobot_SIM808::getSignalStrength(int *buffer)
514 {
515 \longrightarrow //AT+CSQ\longrightarrow \longrightarrow --> "6"+"CR"="10
516 \longrightarrow //+CSQ: "<rssi>, <ber> \longrightarrow --> "CRLF"+"5"+"CRLF"="9>
517 \longrightarrow //0K \longrightarrow \longrightarrow
                                                            → --> • CRLF • + • 2 • + • CRLF • = • • 6
518
521 → char ** p, ** s;
523 → sim808_flush_serial();
524 -->sim808_send_cmd("AT+CSQ\r");
525 → sim808_clean_buffer(gprsBuffer, 26);
\longrightarrow if (NULL*!=*(s*=*strstr(gprsBuffer,*"+CSQ:")))*{
528 \longrightarrow s = strstr((char*)(s), """);
```

```
\longrightarrow \longrightarrow s = s + 1; \cdot \cdot //We = are = in = the = first = phone = number = character = the standard form of the standar
\longrightarrow p==strstr((char**)(s),*",");*//p*is*last*character*""
531 →→if (NULL != s) {
532 \longrightarrow \longrightarrow i = 0;
\longrightarrow while (s < p)  {
534 -------------
                                    →buffers[i++]•=•*(s++);
535 \longrightarrow \longrightarrow \longrightarrow \}
\longrightarrow buffers[i] = '\0';
537 \longrightarrow \longrightarrow \}
538 →→*buffer = atoi(buffers);
539 \longrightarrow return true;
540 →}
541 ──return•false;
542 }
543
544 bool*DFRobot_SIM808::sendUSSDSynchronous(char**ussdCommand,*char**resultcode,*char**response)
           \longrightarrow //AT+CUSD=1,"{command}"
548 →//
\longrightarrow//+CUSD:1,"{response}",{int}
551 \longrightarrow byte i = 0;
552 ****char*gprsBuffer[200];
553 ····char**p,*s;
554 ····sim808_clean_buffer(response, sizeof(response));
555 →
558 ····sim808_send_cmd(ussdCommand);
559 ****sim808 send cmd("\"\r");
560 → if(!sim808_wait_for_resp("OK\r\n", "CMD))
561 → return•false;
563 ....sim808_read_buffer(gprsBuffer,200,DEFAULT_TIMEOUT);
564 ····if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CUSD:*")))*{
565 *******resultcode*=**(s+7);
\longrightarrow resultcode[1] = '\0';
\longrightarrow \longrightarrow if(!('0'"<="*resultcode"&&"*resultcode"<="'2'))
568 → → → return•false;
569 \longrightarrow s = strstr(s, "\"");
570 ·····s==s++1;··//Werarerin-the-first-phone-number-character
571 ·····p==strstr(s,"\""); //p=is=last=character="""
572 ·····if (NULL != s) {
573 ······i·=·0;
574 ·····while (s < p) {
575 ·····response[i++] = = *(s++);
576 · · · · · · · }
577 ·····response[i] == '\0'; ·········
578 .....
579 <del>→</del>return•true;
580 →}
581 ──return•false;
582 }
583
584 bool DFRobot_SIM808::cancelUSSDSession(void)
586 ····return*sim808_check_with_cmd("AT+CUSD=2\r\n","OK\r\n",CMD);
587 }
588
589 //Here-is-where-we-ask-for-APN-configuration,-with-F()-so-we-can-save-MEMORY
590 bool.DFRobot_SIM808::join(const.__FlashStringHelper.*apn,.const.__FlashStringHelper.*userName,.const.
          _FlashStringHelper•*passWord)
591 {
            →byte•i;
592 -
593 ••••char•*p,•*s;
```

```
594 ••••char•ipAddr[32];
595 ····//Select-multiple-connection
596 ···//sim808_check_with_cmd("AT+CIPMUX=1\r\n","OK",DEFAULT_TIMEOUT,CMD);
597
598 · · · · //set • APN. • OLD • VERSION
599 ····//snprintf(cmd,sizeof(cmd),"AT+CSTT=\"%s\",\"%s\",\"%s\"\r\n",_apn,_userName,_passWord);
600 ····//sim808_check_with_cmd(cmd, ""OK\r\n", "DEFAULT_TIMEOUT, CMD);
601 ····sim808_send_cmd("AT+CSTT=\"");
602 ****if*(apn)*{
603 ·····sim808_send_cmd(apn);
604 ....}
605 ....sim808_send_cmd("\",\"");
606 ····if (userName) {
607 ·····sim808_send_cmd(userName);
608 ....}
609 ····sim808_send_cmd("\",\"");
610 ····if (passWord) {
611 ....sim808_send_cmd(passWord);
612 ....}
613 ····sim808_check_with_cmd("\"\r\n", •"OK\r\n", •CMD);
614 ....
615
616 ····//Brings up wireless connection
617 ****sim808_check_with_cmd("AT+CIICR\r\n", "OK\r\n", "CMD);
619 ····//Get·local·IP·address
620 ****sim808_send_cmd("AT+CIFSR\r\n");
621 ····sim808_clean_buffer(ipAddr,32);
622 ····sim808 read buffer(ipAddr,32);
623 →//Response:
625 \longrightarrow // \r \n \longrightarrow \longrightarrow \longrightarrow -- \ \circ \circ \circ + \circ 2
627 →//Response error:
628 →//AT+CIFSR\r\n······
629 →//\r\n→ → → •
630 →//ERROR\r\n
631 ****if*(NULL*!=*strstr(ipAddr, "ERROR"))*{
632 \longrightarrow return false;
633 \longrightarrow 
634 ****s"="ipAddr"+"11;
635 ····p==strstr((char*)(s),"\r\n");"//p*is*last*character*\r\n
636 ····if (NULL != s) {
637 ·····i·=·0;
638 ·····while (s < p) {
639 ....ip_string[i++] = **(s++);
640 .....
641 ....ip_string[i] == '\0'; .....
642 ....}
643 ····_ip==str_to_ip(ip_string);
644 ····if(_ip•!=•0)•{
645 ····return•true;
646 ....}
647 ···return false;
648 }•
649
650 void DFRobot_SIM808::disconnect()
653 }
654
655 bool*DFRobot_SIM808::connect(Protocol*ptl,const*char***host,*int*port,*int*timeout,*int*chartimeout)
656 {
657 ···//char•cmd[64];
658 —→char•num[4];
659 ····char•resp[96];
```

```
660
661 ····//sim808_clean_buffer(cmd,64);
662 ••••if(ptl•==•TCP)•{
663 → sim808_send_cmd("AT+CIPSTART=\"TCP\",\"");
664 \longrightarrow sim808\_send\_cmd(host);
665 \longrightarrow sim808\_send\_cmd("\",");
\longrightarrow itoa(port, num, 10);
667 \longrightarrow sim808\_send\_cmd(num);
668 \longrightarrow sim808\_send\_cmd("\r\n");
669 //····sprintf(cmd, ""AT+CIPSTART=\"TCP\",\"%s\",%d\r\n",host, "port);
670 ····} • else • if (ptl • == • UDP) • {
671 → sim808_send_cmd("AT+CIPSTART=\"UDP\",\"");
672 \longrightarrow sim808\_send\_cmd(host);
673 \longrightarrow sim808\_send\_cmd("\",");
674 \longrightarrow  itoa(port, num, 10);
675 \longrightarrow sim808\_send\_cmd(num);
676 \longrightarrow sim808\_send\_cmd("\r\n");
677
679 ····} • else • {
680 ····return•false;
681 ....}
682 ....
683
684 ....//sim808_send_cmd(cmd);
685 ....sim808_read_buffer(resp, 96, timeout, chartimeout);
686 →//Serial.print("Connect•resp:•");•Serial.println(resp);····
687 ····if(NULL*!=*strstr(resp,"CONNECT"))*{*//ALREADY*CONNECT*or*CONNECT*OK
688 ····return•true;
689 ....}
690 ···return false;
691 }
692
693 //Overload*with*F()*macro*to*SAVE*memory
694 bool*DFRobot_SIM808::connect(Protocol*ptl,const*__FlashStringHelper**host,*const*__FlashStringHelper**
    port, int timeout, int chartimeout)
695 {
696 ····//char*cmd[64];
697 · · · · char · resp[96];
699 ····//sim808_clean_buffer(cmd,64);
700 ••••if(ptl•==•TCP)•{
701 .....sim808_send_cmd(F("AT+CIPSTART=\"TCP\",\""));...//%s\",%d\r\n",host,*port);
702 ····} else if(ptl == UDP) {
703 ·····sim808_send_cmd(F("AT+CIPSTART=\"UDP\",\""));···//%s\",%d\r\n",host,*port);
704 ····} • else • {
705 ····return false;
706 ....}
707 ····sim808_send_cmd(host);
708 ----sim808_send_cmd(F("\","));
709 ····sim808_send_cmd(port);
710 ----sim808_send_cmd(F("\r\n"));
711 //→Serial.print("Connect:•");•Serial.println(cmd);
712 *** sim808_read_buffer(resp, 96, timeout, chartimeout);
713 //→Serial.print("Connect•resp:•");•Serial.println(resp);····
714 ····if(NULL:!=:strstr(resp,"CONNECT")):{://ALREADY:CONNECT:Or:CONNECT:OK
715 ····return•true;
716 ....}
717 ····return•false;
718 }
719
720 bool DFRobot_SIM808::is_connected(void)
721 {
722 ····char•resp[96];
723 ****sim808_send_cmd("AT+CIPSTATUS\r\n");
724 ····sim808_read_buffer(resp,sizeof(resp),DEFAULT_TIMEOUT);
```

```
725 ····if(NULL•!=•strstr(resp, "CONNECTED"))•{
726 ·····//+CIPSTATUS::1,0,"TCP","216.52.233.120","80","CONNECTED"
727 ····return•true;
728 ····}•else•{
729 .....//+CIPSTATUS: 1,0,"TCP","216.52.233.120","80","CLOSED"
730 -----//+CIPSTATUS:-0,,"","","","INITIAL"
731 ····return•false;
732 · · · · }
733 }
734
735 bool DFRobot_SIM808::close()
737 ····//•if•not•connected,•return
738 ····if (!is_connected()) {
739 ····return•true;
740 ....}
741 ····return*sim808_check_with_cmd("AT+CIPCLOSE\r\n",*"CLOSE*OK\r\n",*CMD);
742 }
743
744 int DFRobot_SIM808::readable(void)
746 ****return*sim808_check_readable();
747 }
748
749 int DFRobot_SIM808::wait_readable(int wait_time)
751 ****return*sim808_wait_readable(wait_time);
752 }
753
754 int DFRobot SIM808::wait writeable(int reg size)
756 ····return•req_size+1;
757 }
759 int DFRobot_SIM808::send(const char * str, int len)
760 {
761 ....//char.cmd[32];
762 \longrightarrow char num[4];
763 ····if(len•>•0){
764 ·····//snprintf(cmd, sizeof(cmd), "AT+CIPSEND=%d\r\n",len);
765 → //sprintf(cmd, "AT+CIPSEND=%d\r\n",len);
766 → sim808_send_cmd("AT+CIPSEND=");
767 → → itoa(len, num, 10);
768 \longrightarrow sim808\_send\_cmd(num);
769 \longrightarrow if(!sim808_check_with_cmd("\r\n",">",CMD)) \in {
770 .....//if(!sim808_check_with_cmd(cmd,">",CMD))"{
771 ····return•0;
772 ....}
773 ·····/*if(0-!=-sim808_check_with_cmd(str,"SEND-OK\r\n",-DEFAULT_TIMEOUT-*-10-,DATA))-{
774 ·····return•0;
775 · · · · · · · · }*/
776 ·····delay(500);
778 ·····delay(500);
780 ······if(!sim808_wait_for_resp("SEND*OK\r\n",*DATA,*DEFAULT_TIMEOUT***10,*DEFAULT_INTERCHAR_TIMEOUT*** 7
   10))•{
781 ·····return•0;
782 .....}....
783 ....}
784 ···return·len;
785 }
786 ....
788 int DFRobot_SIM808::recv(char* buf, int len)
789 {
```

```
790 ····sim808 clean buffer(buf,len);
791 ····sim808 read buffer(buf,len);···//Ya·he·llamado·a·la·funcion·con·la·longitud·del·buffer·-·1·y·luego·le⊋
    •estoy•añadiendo•el•0
792 ····return•strlen(buf);
793 }
794
795 void DFRobot_SIM808::listen(void)
796 {
797 -
      →•if(serialFlag)
798 →→; •//hgprsSerial->listen();
799 →•else
800 -
      →→•gprsSerial->listen();
801
802 }
803
804 bool DFRobot_SIM808::isListening(void)
     →//•if(serialFlag)
807 → → //•return•hgprsSerial.isListening();
808 →//•else
809 → //•return•gprsSerial.isListening();
810 }
811
812 uint32_t*DFRobot_SIM808::str_to_ip(const*char**str)
813 {
814 ····uint32_t·ip·=·0;
815 ****char**p*=*(char*)str;
816 ····for(int·i·=·0;·i·<·4;·i++)·{
817 · · · · · · ip | = · atoi(p);
818 ....p = strchr(p, '.');
819 ********if*(p*==*NULL)*{
820 .....break;
821 .....
822 ....ip.<<=-8;
823 · · · · · · · p++;
824 ....}
825 ····return•ip;
826 }
827
828 char**DFRobot SIM808::getIPAddress()
830 ····//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);
831 ····return•ip_string;
832 }
834 unsigned long DFRobot_SIM808::getIPnumber()
835 {
836 ····return•_ip;
837 }
838 /*•NOT•USED•bool•DFRobot_SIM808::gethostbyname(const•char*•host,•uint32_t*•ip)
840 ····uint32_t•addr•=•str_to_ip(host);
841 ····char•buf[17];
842 ····//snprintf(buf, sizeof(buf), "%d.%d.%d.%d", (addr>>24)&0xff, (addr>>16)&0xff, (addr>>8)&0xff, addr&
   0xff);
843 ----if (strcmp(buf, host) === 0) {
844 ********ip*=*addr;
845 ····return•true;
846 ....}
847 ····return•false;
848 }
849 */
851 bool*DFRobot_SIM808::getLocation(const*__FlashStringHelper**apn,*float**longitude,*float**latitude)
852 {⋯⋯→
```

```
853 ——int:i:=:0;
854 ····char•gprsBuffer[80];
856 · · · · char • * s;
857 ....
858 ——//send AT+SAPBR=3,1, "Contype", "DFRobot_SIM808"
859 \longrightarrow sim808\_check\_with\_cmd("AT+SAPBR=3,1,\"Contype\",\"DFRobot\_SIM808\"\","OK\r\n",CMD);
860 —>//sen AT+SAPBR=3,1, APN", DFRobot_SIM808_APN"
861 \longrightarrow sim808_send_cmd("AT+SAPBR=3,1,\"APN\",\"");
862 →if•(apn)•{
863 ·····sim808_send_cmd(apn);
864 ....}
865 ....sim808_check_with_cmd("\"\r","OK\r\n",CMD);
866 \longrightarrow //send AT+SAPBR = 1,1
867 —>sim808_check_with_cmd("AT+SAPBR=1,1\r","OK\r\n",CMD);
869 —>//AT+CIPGSMLOC=1,1
870 → sim808_flush_serial();
873 —>sim808_read_buffer(gprsBuffer,sizeof(gprsBuffer),2*DEFAULT_TIMEOUT,6*DEFAULT_INTERCHAR_TIMEOUT);
874 →//Serial.println(gprsBuffer);
875 ....
876 →if(NULL*!=*(*s*=*strstr(gprsBuffer,"+CIPGSMLOC:")))
877 —→{
878 --> s == strstr((char **)s, * ", ");
879 \longrightarrow s = s+1;
880 \longrightarrow //Serial.println(*s);
881 \longrightarrow i=0;
882 \longrightarrow while(*(++s)*!=**',')
883 \longrightarrow buffer[i++]=*s;
884 \longrightarrow buffer[i] = 0;
885 → *longitude = atof(buffer);
886 -------
887 \longrightarrow i=0;
888 \longrightarrow while(*(++s)*!=**',')
889 \longrightarrow buffer[i++]=*s;
890 \longrightarrow buffer[i] = 0;
891 \rightarrow*latitude==atof(buffer); \rightarrow*
892 → return•true;
893 →}
894 —→return•false;
897 bool*DFRobot_SIM808::attachGPS()
      → if(!sim808_check_with_cmd("AT+CGNSPWR=1\r\n", "OK\r\n", "CMD)) " { "
900 ····return false;
901 ....}
903 ····return false;
904 ....}
905 —→return•true;
906 }
907
908 bool DFRobot_SIM808::detachGPS()
910 \longrightarrow if(!sim808_check_with_cmd("AT+CGNSPWR=0\r\n", "OK\r\n", "CMD)) "{"
911 ····return false;
912 ....}
913 —→return•true;
914 }
915
916 bool DFRobot SIM808::getGPRMC()
917 {
     →char*c;
918 -
```

```
919 → static bool endflag = false;
 920 → static • char • count;
 921 ------
 922 →while(serialSIM808->available())···//??????????
 923 \longrightarrow{\longrightarrowc==serialSIM808->read();
 924 \longrightarrow if(endflag)
 925 →→-----{
 926 \longrightarrow \longrightarrow
                    \rightarrowif(count--)
 928 \longrightarrow \longrightarrow \longrightarrow
                          →receivedStack[receivedStackIndex++] == c;
 929 \longrightarrow \longrightarrow \longrightarrow
 930 \longrightarrow \longrightarrow \stackrel{\cdot}{\text{else}}
 931 → → → endflag = false;
 932 → receivedStack[receivedStackIndex]*=*'\0';
 933 →→→→return*true;
 934 \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow
 935 \longrightarrow \longrightarrow
 936 \longrightarrow else
 937 →→{→
 938 \longrightarrow \longrightarrow switch(c)
 940 \longrightarrow \longrightarrow \longrightarrow
 941 \longrightarrow\longrightarrow\longrightarrow case '$':\longrightarrow\longrightarrow\longrightarrow\longrightarrow\longrightarrow\longrightarrow\longrightarrow
 942 → → → receivedStackIndex = • 0;
 943 \longrightarrow \longrightarrow \longrightarrowreceivedStack[receivedStackIndex++] == c; \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow
 944 \longrightarrow \longrightarrow \longrightarrow break;
 945 → → → case * '*':
 946 → → → → endflag = = true;
 947 \longrightarrow \longrightarrow \longrightarrow count = 2;
 948 \longrightarrow \longrightarrow \longrightarrowreceivedStack[receivedStackIndex++]"="c;\longrightarrow \longrightarrow \longrightarrow
 949 \longrightarrow \longrightarrow \longrightarrow break;
 950 \longrightarrow \longrightarrow \longrightarrow default:
 951 → → → if(receivedStackIndex < 120)
 952 → → → → receivedStack[receivedStackIndex++] == c;
 953 \longrightarrow \longrightarrow \longrightarrow \Longrightarrow break;
 955 \longrightarrow \longrightarrow \longrightarrow
 956 → → → return•false;
 957 \longrightarrow \longrightarrow
 958 → → return•false;
 959 →}
 960 —→return•false;
 961 }
 963 bool*DFRobot_SIM808::parseGPRMC(char**gpsbuffer)
→if(strstr(gpsbuffer,des) == NULL) · · //????$GPRMC???????????GPS???
 974 \longrightarrow else
 975 →→----{
 976 → → //Serial.print("NO•:");
 977 → → //Serial.println(gpsbuffer[18]);
 978 \longrightarrow \longrightarrow return•false;
 979 \longrightarrow \longrightarrow
 980 --------
 981 →}
 982 ------
 983 }
 984
```

```
985 // Parse a (potentially negative) number with up to 2 decimal digits -xxxx.yy
986 int32_t DFRobot_SIM808::parseDecimal(const char *term)
987 {
988 • • bool • negative • = • * term • = = • ' - ';
989 ··if (negative) ++term;
990 ..int32_t.ret.=.100.*.(int32_t)atol(term);
991 • while (isdigit(*term)) + +term;
992 **if*(*term*==*'.'*&&*isdigit(term[1]))
993 • • {
994 ····ret*+=*10***(term[1]*-*'0');
995 ····if (isdigit(term[2]))
996 · · · · · ret · + = · term[2] · - · '0';
997 ••}
998 ··return negative ? - ret : ret;
999 }
1001 •void•DFRobot_SIM808::getTime(uint32_t•time){
1002 → •GPSdata.hour · · · · = · · time • / • 1000000;
1004 → GPSdata.second = (time / 100) % 100;
1005 → GPSdata.centisecond = · · time · % · 100;
1007
1009 • void DFRobot_SIM808::getDate(uint32_t date){
1011 \longrightarrow GPSdata. year year + 2000; \longrightarrow
1012 ---- GPSdata.month -- = (date -/ -100) -% -100;
1014 • }
1015 •
1017 bool DFRobot SIM808::getGPS()
1020 → return•false;
1021 →//*Serial.println(receivedStack);
1022 → if(!parseGPRMC(receivedStack)) · · //????$GPRMC????????GPS???
1023 → return•false; ••
1024 ------
1025 \longrightarrow // skip mode
1026 ----char**tok*=*strtok(receivedStack,*",");-----//??'??????*
1027 ····if*(!*tok)*return*false;
1029 ···//*grab*time·······//<1>*UTC????????hhmmss.sss??
1030 ---//*tok*=*strtok(NULL,*",");
1031 —→char**time*=*strtok(NULL,*",");
1032 ****if*(!*time)*return*false;
1034 → getTime(newTime);
1035
1036 ••••//•skip•fix
1037 ····tok==strtok(NULL,=",");······//<2>-???????A=???????V=??????
1038 ····if (! • tok) • return • false;
1039
1040 ···//•grab•the•latitude
1041 ----char-*latp-=-strtok(NULL,-",");---//<3>-???ddmm.mmmm(???)???(j???0?????????)
1042 ***if*(!*latp)*return*false;
1043
1044 ····//*grab*latitude*direction······//*<4>*??????N(??????)??S(?????)
1045 ****char**latdir*=*strtok(NULL,*",");
1046 ****if*(!*latdir)*return*false;
1047
1048 ···//-grab-longitude·······//<5>-????dddmm.mmmm(???)???(j???0????????)
1049 ****char**longp*=*strtok(NULL,*",");
1050 ····if (! longp) return false;
```

```
1051
1052 ···//•grab•longitude•direction·····//<6>•???????E(????)??W(????)
1053 ****char**longdir*=*strtok(NULL,*",");
1054 ····if (! longdir) return false;
1055
1056 ····double latitude = atof(latp);
1057 ····double longitude = atof(longp);
1059 ···//*convert*latitude*from*minutes*to*decimal
1060 ****float*degrees*=*floor(latitude*/*100);
1061 ····double minutes = latitude - (100 * degrees);
1062 ....minutes /= 60;
1063 ****degrees*+=*minutes;
1065 ····//•turn•direction•into•+•or•-
1066 ....if (latdir[0] == 'S') degrees = -1;
1068 ....//*lat = degrees;
1069 → GPSdata.lat = degrees;
1071 ····//•convert•longitude•from•minutes•to•decimal
1072 ****degrees*=*floor(longitude*/*100);
1073 ****minutes*=*longitude*-*(100***degrees);
1074 ····minutes*/=*60;
1075 ····degrees += minutes;
1077
1078 ···//*turn*direction*into*+*or*-
1079 ****if*(longdir[0]*==*'W')*degrees**=*-1;
1081 ****//*lon*=*degrees;
1082 → GPSdata.lon=•degrees;
1084 ····//*only*grab*speed*if*needed········//<7>*???????(000.0~999.9???j???0???????)
1085 ---//*if*(speed kph*!=*NULL)*{
1087 ····//•grab•the•speed•in•knots
1088 *****char**speedp*=*strtok(NULL,*",");
1089 ******if*(!*speedp)*return*false;
1091 -----//-convert-to-kph
1092 *****//*speed kph*=*atof(speedp)***1.852;
1094
1095 ...//*}
1098 ---//*if*(heading*!=*NULL)*{
1100 ····//•grab•the•speed•in•knots
1101 •••••char•*coursep•=•strtok(NULL,•",");
1102 ·····if*(!*coursep)*return*false;
1103
1104 ·····//*heading==atof(coursep);
1105 → GPSdata.heading = atof(coursep);
1106 ...//-}
1107 <del>→</del>
1108 →•//•grab•date
1110 ***if*(!*date)*return*false;
1112 ──getDate(newDate);
1113
1114 ····//•no•need•to•continue
1115 ···//•if•(altitude•==•NULL){
1116 ···//···return•true;
```

```
1117 →//}
1118 → return•true;
1119 }
1120
1121
1122
```