GprsHttp.py - a-gsm 2.064 HTTP client over GPRS example utility

COPYRIGHT (c) 2014 Dragos Iosub / R&D Software Solutions srl

# You are legally entitled to use this SOFTWARE ONLY IN CONJUNCTION WITH a-gsm DEVICES USAGE. Modifications, derivates and redistribution
# of this software must include unmodified this COPYRIGHT NOTICE. You can redistribute this SOFTWARE and/or modify it under the terms
# of this COPYRIGHT NOTICE. Any other usage may be permitted only after written notice of Dragos Iosub / R&D Software Solutions srl.

# This SOFTWARE is distributed is provide "AS IS" in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

# Dragos Iosub, Bucharest 2014.
http://itbrainpower.net

# Raspberry PI - a-gsm wiring connection:
# Legal disclaimer:
# Incorrect or faulty wiring and/or connection can damage your RPi and/or your a-gsm board!
# Following directives are provided "AS IS" in the hope that it will be useful, but WITHOUT ANY WARRANTY!
# Do the wiring on your own risk!

#name RPi a-gsm shield
#
#POWER a-gsm 16 D7 - power(UP/DOWN)
#RESET a-gsm 18 D6 - reset
#a-gsm STATUS 12 D5 - status
#
#serial TXD0 08 D4 - tx(rxd)
#serial RXD0 10 D3 - rx(txd)
#
#5V 02/04 5V - on Arduino power IN connector
#GND 06/14 GND - on Arduino power IN connector
#
#IMPORTANT:
# a-gsm's POWER supply input selector must be in "use 5V pin" position

# this utility must be runned as root (just use: sudo python GprsHttp.py)

GPRS_context="live.vodafone.com"
GPRS_user=""
GPRS_password=""

GPRS_context="internet"
#GPRS_user="internet-gprs"
#GPRS_password="internet"
GPRS_user=""
GPRS_password=""

SERVER_address="itbrainpower.net"# 4 socket open
SERVER_port="80"# 4 socket open
HTTP_server="http://itbrainpower.net"# http server URL
serverfile="/a-gsm/test/agsm.php"# file name on server
fst_par="imei="# 1'st GET param
sec_par="&par="# second GET param
message = "Hello world!"# value of the second param

serialSpeed = 19200# we recommend usage of 19200 bps speed. If you want to use other speed,
first set the a-gsm speed using setSerial.py
usePoweringControl = 1# change it to 0 if you do not want to control powerUP/powerDown the
a-gsm board. In this case, please be sure the a-gsm board is powered UP(the a-gsm green led
lights continuous) before run this utility

# Do not change under following line! Instead make one copy of the file and play with!
###################################################
#definitions for a-gsm control(RPi GPIO mode)
POWER = 16
RESET = 18
STATUS = 12

i=0
buffd ="

sreadlen = 100# how many chars to read in one try over serial

fileBuffer = ""

import os
import serial
from time import sleep, time
from string import replace
import RPi.GPIO as GPIO

if not os.getuid() == 0:
    print("please use root privileges! try: \"sudo python fileHandling.py\"")
    exit(0)

agsm = serial.Serial("/dev/ttyAMA0", serialSpeed, timeout=1)
agsm.open()

print "Hi folks. Let's send some data over Gprs using Http GET method. The server will
return some data, that will be listed."

# poweron() - power up the modem
def poweron():
    #...function code here

# poweroff() - shutdown the modem
def poweroff():
    #...function code here

# restartModem() - restart the modem
```python
def restartModem():
    #...function code here

#recUARTdata(endchars,to,tm)
# read from modem - read string is loaded in global var buffd
# looking for endchars [SUCCESS STRING,FAILURE STRING] and to - TIMEOUT
# return 0 for SUCCESS, 1 for FAILURE, -1 for timeout
# tm how many chars to read(maximum) in one loop from serial
def recUARTdata(endchars,to,tm):
    #...function code here
    return SuccessErrorTimeout

#sendATcommand(command, endchars,to)
# command +"\r\n" is forwarded to modem
# looking for endchars [SUCCESS STRING,FAILURE STRING] and to - TIMEOUT
# return 0 for SUCCESS, 1 for FAILURE, -1 for timeout
# modem response is loaded in global var buffd
def sendATcommand(command, endchars,to):
    global sreadlen
    agsm.write(command+"\r\n")
    return (recUARTdata(endchars,to,sreadlen))

#aGsmWRITE(command)
# just write command to serial without CR LF
def aGsmWRITE(command):
    agsm.write(command)

#setupMODEM()
# just set and look at modem to be ready for usage
def setupMODEM():
    #...function code here

#getIMEI()
# utility that read and IMEI (MODEM related identifier)
# value is loaded in global var IMEI
def getIMEI():
    #...function code here
    return IMEI

#wait4GSMReg(to)
# read GSM registration status
# to -timeout in seconds
def wait4GSMReg(to):
    #...function code here
    return GsmRegistrationStatus

#wait4GPRSReg(to)
# read GPRS registration status
# to -timeout in seconds
def wait4GPRSReg(to): #to -timeout in seconds
    #...function code here
    return GprsRegistrationStatus

#RaspberryPI hardware setup section start
if usePoweringControl==1:
    GPIO.setmode(GPIO.BORD)```
GPIO.setwarnings(False)
try:
    GPIO.setup(STATUS, GPIO.IN)
    GPIO.setup(POWER, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(RESET, GPIO.OUT, initial=GPIO.LOW)
except:
    GPIO.cleanup() #free GPIO
    GPIO.setup(STATUS, GPIO.IN)
    GPIO.setup(POWER, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(RESET, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setwarnings(True)
#RaspberryPI hardware setup section stop

#here start the main code
sleep(2)#some delay...
state = 0
tm = time()
count=0

while(1):
    sleep(0.5)
    if (state==0):
        tm = time()
        print("check powering")
        if usePoweringControl==1:
            poweron()
        else:
            print("No powering control has been enabled! Be sure that a-gsm board is on!")
            sleep(10)
            state=state+1
    elif(state==1):
        tm = time()
        setupMODEM()
        print("Read IMEI")
        IMEI=getIMEI()
        run=1
        print("Query State of Initialization")
        while(run==1):
            res = sendATcommand("AT+QINISTAT",["OK","ERROR"],3)
            if(buffd.find("3")):
                print("ready...")
                state=state+1
                tm=time()
                run=0
            else:
                if(time()-tm > 30):
                    if usePoweringControl==1:
                        restartModem()
                        state = 0
                        print("timeout state 1, reboot")
                    else:
                        sleep(0.5)
                        print("not ready, retry...")
                elif(state==2):
                    print("checking gprs registration")
res = wait4GPRSReg(10)
if(res==1):
    state=state+1
tm=time()
    print("ready...")
elif(time()-tm > 40):
    if usePoweringControl==1:
        if usePoweringControl==1:
            restartModem()
        state = 0
        print("timeout state 2, reboot")
    else:
        sleep(1)
        print("not ready, retry...")
elif(state==3):
    print("checking gsm registration")
    state=state+1
elif(state==4):
    print("try attach GPRS")
    sendATcommand("AT+CGATT=1","OK","ERROR"), 3)
if (buffd.find("OK")>0):
    state=state+1
tm=time()
    print("success...")
elif(time()-tm > 40):
    if usePoweringControl==1:
        restartModem()
    state = 0
    print("timeout state 4, reboot")
else:
    sleep(1)
    print("let's try again")
elif(state==5):
    print("try to set the GPRS context")
aGsmWRITE("AT+QIDEACT\r\n")
sleep(5)
    sendATcommand("AT+QIREGAPP="+GPRS_context+"","+GPRS_user+"","+GPRS_password+"","OK","ERROR"],10)
if (buffd.find("OK")>0):
    state=state+1
tm=time()
    print("success...")
elif(time()-tm > 10):
    if usePoweringControl==1:
        restartModem()
    state = 0
    print("timeout state 5, reboot")
else:
    sleep(1)
    print("let's try again")
elif(state==6):
    print("GET local IP")
    res = sendATcommand("AT+QIACT","OK","ERROR"],10)
    if (buffd.find("ERROR") < 0):
        sendATcommand("AT+QIDNSIP=1","OK","ERROR"],5)
state = state + 1  
tm = time()  
print ("success...")  
# print buffd  
elif (time() - tm > 12):  
    if usePoweringControl == 1:  
        restartModem()  
    state = 0  
    print ("timeout state 6, reboot")  
else:  
    sleep(1)  
    print ("let's try again")  
elif (state == 7):  
    print ("try to open the socket")  
    sendATcommand ("AT+QIOPEN="/"TCP","+SERVER_address="/"","+SERVER_port="/"",[
"CONNECT OK","ERROR"],10)  
    # print buffd  
    if (buffd.find ("CONNECT OK") > 0 or buffd.find ("ALREADY CONNECT") > 0):  
        tm = time()  
        state = state + 1  
        print ("success...")  
    elif (time() - tm > 600):  
        if usePoweringControl == 1:  
            restartModem()  
        state = 0  
        print ("timeout state 7, reboot")  
    else:  
        print ("timeout open socket...wait and retry")  
        sleep(5)  
        sendATcommand ("AT+QICLOSE","OK","ERROR"],10)  
elif (state == 8):  
    print ("let's send some data socket")  
    res = sendATcommand ("AT+QISTAT","CONNECT OK","ERROR"],10)  
    # print buffd  
    if (res == 0):  
        message = replace (message," ", 
totalChars = len (HTTP_server)  
totalChars += len (serverfile)  
totalChars += len (fst_par)  
totalChars += len (IMEI)  
totalChars += len (sec_par)  
totalChars += len (message)  
res = sendATcommand ("AT+QHTTPURL=\"+str(totalChars)+\",40","CONNECT","ERROR"],10)  
    # print buffd  
aGsmWRITE (HTTP_server+serverfile+fst_par+IMEI+sec_par+message+\"\r\n\r\n")  
sleep(0.5)  
agsm.flushInput()  

sendATcommand ("AT+QHTTPGET=15","OK","ERROR"],16)  
# print buffd  

res = sendATcommand ("AT+QHTTPREAD=25","OK","ERROR"],26)  
print (buffd)  
print ("GET PERFORMED")  
state = 7
aGsmWRITE("AT+QICLOSE\r\n")
count+=1

else:
    print("error in socket status. get not performed.")
aGsmWRITE("AT+QICLOSE\r\n")
aGsmWRITE("AT+QIDEACT\r\n")
state = 2

if(count>2):
    break
    print("wait 53 seconds")
sleep(53);

agsm.close() # close serial

sleep(5)

if usePoweringControl==1:
    poweroff() # shutdown a-gsm

GPIO.cleanup() # free GPIO