```
################################################################################################################
################################################
#a-gsmUtilities.py - a-gsm 2.064 SIM/MODEM/MISCELLANEOUS usage example utility
#COPYRIGHT (c) 2014 Dragos Iosub / R&D Software Solutions srl
#
#You are legaly 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 permited 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
################################################################################################################
################################################
#HEALTH AND SAFETY WARNING!!!!!!!!!!!!!!!!!!!
#High power audio (around 700mW RMS)! You can damage your years! Use it with care when
headset is connected.
#We recomend to use AT+CLVL=25, audio setup command in order to limit the output power.
#
#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             04/02   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 a-gsmUtilities.py)

useSIM = 0#                                     0(zero)- TOP SIM, or 1(one)- BOTTOM SIM


destinationNumber="+40123456789"#usually phone number with International prefix eg. +40 for
Romania
```

```
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! Insteed 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 "HEALTH AND SAFETY WARNING!!!!!!!!!!!!!!!!!!!HEALTH AND SAFETY
WARNING!!!!!!!!!!!!!!!!!!!"
print "\r\n\r\nHigh power audio (around 700mW RMS)! You can damage your years! Use it with
CARE when HEADSETS are USED."
print "We recomend to use AT+CLVL=25, audio setup command in order to limit the output power."
print "\r\n"
sleep(10)
print "Hi folks. Here are some utilities that can be helpful in your developing/porting some
application."
print "Only some functions are called from the code. Just take a look inside, in order to
find other implemented utilities. Also, for other functions implementation, take a look
inside the M85 AT command manual."
sleep(3)


activeSIM = 0 #store the active SIM id; 0(zero)- TOP SIM, or 1(one)- BOTTOM SIM

#setAUDIOchannel()
#   Prepare a-gsm for audio part usage
#   High power audio (around 700mW RMS)! You can damage your years! Use it with care when
headset is connected.
#   We recomend to use AT+CLVL=25, audio setup command in order to limit the output power.
def setAUDIOchannel():
```

```
  #...function code here

#activateTopSIM()
#   call this function if you want to test/use SIM inserted in the SIM SOCKET placed on TOP
of a-gsm
def activateTopSIM():
    setActiveSIM(0)

#activateBottomSIM()
#   call this function if you want to test/use SIM inserted in the SIM SOCKET placed on
BOTTOM of a-gsm
def activateBottomSIM():
    setActiveSIM(1)

#dial(number)
#   just dial "number"
#   use it in conjunction with getcallStatus() to see if remote ANSWER/BUSY....
def dial(number):
    return sendATcommand("ATD"+str(number)+";",["OK","ERROR"],3)

#hangup()
#   hangup the call
def hangup():
    return sendATcommand("ATH",["OK","ERROR"],3)

#answer()
#   answer the call
def answer():
    return sendATcommand("ATA",["OK","ERROR"],3)

#enableAutoAnswer()
#   set auto answer at ringCnt RING
def enableAutoAnswer(ringCnt):
    return sendATcommand("ATS0="+str(ringCnt),["OK","ERROR"],3)

#disableAutoAnswer()
#   disable auto answer
def disableAutoAnswer():
    return sendATcommand("ATS0=0",["OK","ERROR"],3)


#setActiveSIM(SIM)
#   Set the active SIM, Active SIM value will be stored in activeSIM var
def setActiveSIM(SIM):
    #...function code here
    activeSIM = SIM#load active SIM value



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

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

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

#int getcallStatus()
#   detects if the voice call is CONNECTED
#   returns:
#       0 Active   CONNECTED   BOTH
#       1 Held                 BOTH
#       2 Dialing (MO call)    OUTBOUND
#       3 Alerting (MO call)   OUTBOUD
#       4 Incoming (MT call)   INBOUD
#       5 Waiting (MT call)    INBOUD
def getcallStatus():
    #...function code here
    return callStatus;

#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

#setupMODEMforDTMFusage()
#   just set and look at modem to be ready for DTMF usage
#   run this before encode&send or receive&decode DTMF
def setupMODEMforDTMFusage():
    #...function code here

#next example for DTMF send -in that 100,100 means: 100ms DTMF lenght and 100 DTMF pause,
best for decoding too. The last 3* are used as terminator string in listen4DTMF(terminator)
#sendATcommand("AT+QWDTMF=6,0,\"ABCD0123456789*#***,100,100\"",["OK","ERROR"],10)#send some
DTMF
```

```
#sendDTMF(DTMFstring, DTMFterminator, DTMFlenght, DTMFpause)
#   ...send some DTMF ...be sure you did before setupMODEMforDTMFusage()
#   DTMFstring      ... value to be transmitted ABCD0123456789*#
#   DTMFterminator  can be null....string used as terminator listen4DTMF(terminator, to))
*** can be a good choise
#   DTMFlenght      ... in msec (100 best for decoding too)
#   DTMFpause       ... in msec (100 best for decoding too)
def sendDTMF(DTMFstring, DTMFterminator, DTMFlenght, DTMFpause):
    #...function code here


#DTMF=""
#listen4DTMF("***", 55)#listen for DTMF see the Arduino C code --just port yourself it to
python
#/*
#Listen for DTMF until "terminator" has been found or "to" (in secs) timeout reached
#return: int
#  -1 TIMEOUT
#  1  SUCCESS
#read DTMF string => DTMF
#*/

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

#getIMSI()
#   utility that read and IMSI (SIM related identifier)
#   value is loaded in global var IMSI
def getIMSI():
    #...function code here
    return IMSI

#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

#getSignalStatus()
#   read GSM signal status
#   return 0 no signal, positive values for certain intervals--see bellow
#   print bar like signal level
def getSignalStatus():
    #...function code here
```

```
    return SignalLevel


#RaspberryPI hardware setup section start
if usePoweringControl==1:
    GPIO.setmode(GPIO.BOARD)
    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

if usePoweringControl==1:
    poweron()

sleep(2)

print("a-gsm will use SIM "+str(useSIM)+" as active...")
setActiveSIM(useSIM)
sleep(2)

setupMODEM()
print("Read IMEI (modem id)")
IMEI=getIMEI()
print(IMEI)
print ""

print("Read IMSI (SIM id)")
IMSI = getIMSI()
print IMSI


print("checking 4 gsm registration")
res = wait4GSMReg(10)
if(res==1):
    print("ready...")

sleep(1)

print("let's set audio channel")
setAUDIOchannel()
print("done...\r\n")
```

```
sleep(1)

print("Let's check the signal level")
res = getSignalStatus()
print "Signal: "+ str(res)
sleep(1)

print("There is any a-gsm processing call?")
res=getcallStatus()
sleep(1)

res=wait4GSMReg(1)
sleep(5)

#run=1
#res = dial(destinationNumber)#res =
sendATcommand("ATD"+destinationNumber+";",["OK","ERROR"],3)
#while(run==1):
#    res=getcallStatus()
#    if(res==0):#here the other part answer the call...status active
#        sleep(30)#you can talk 30 seconds
#        hangup()#hang up the call
#        run=0
#    elif(res<0):#no calls, you may want to redial??
#        sleep(2)
#        print("redial")
#        dial(destinationNumber)#
sendATcommand("ATD"+destinationNumber+";",["OK","ERROR"],3)

print("That's all folks! Next, you may want to play!\r\n")
print("Do not forget to read the \"GSM_M85_AT_Commands_Manual_V1.0.pdf\" available for
download on http://itbrainpower.net!\r\n")

agsm.close()#close serial

sleep(5)

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

GPIO.cleanup()#free GPIO
```