```
#!/usr/bin/env python
# LX200gps.py
# accessing LX200gps
# scripts implementing the Meade command protocol
# including control structures
# set scope up and input
# location (Zurich), date (19May2007), time (20:30),target (terrestial)
# check current DEC and RA values first !!!
# move the final version of this code into
# /usr/lib/python2.5/site-packages
# mb 05/2007
# -----------------------------------------------
import os, sys
import serial
import time
import string, re
#globals
twrite = 0.25       #might need to increase this..
twait = 1.5
tslew = 35.0        #was 25
#-----------------------------------------------

def start_telescope(port):
        #connect to the scope on a port
    #linux: /dev/ttyUSB0 (check with lsusb)
    #win: COMn-1 (check in device manager)
    print "trying to connect to port: " , port
    try:
        serialobject = serial.Serial(port, 9600, timeout = 1)
        if(serialobject.isOpen()):
                print "port is connected"
                #wakeup sleeping telescope
                serialobject.write("#:hW#")
                print "trying to wake scope"

        else:
                print "-- scope not ready! --"
                serialobject = "notready"
    except:
        print "serial port open error  - already open?"
        serialobject = "serialerror"


        return(serialobject)

#-----------------------------------------------

def stop_telescope(serialobject):
        #stop all motion by making scope sleep
```

```
        if(serialobject.isOpen()):
                serialobject.write("#:Q#")
            time.sleep(twrite)
            #sleep telescope
            serialobject.write("#:hN#")
            time.sleep(twrite)
            serialobject.close()
            done = "telescope stopped and sleeping"
        else:
                done = "telescope already disconnected..."
        return(done)
#--------------------------------------------------

def get_telescope_position(serialobject):
        #get RA and DEC
        serialobject.write("#:GR#")
        RA = serialobject.read(10)
        time.sleep(twrite)
        serialobject.write("#:GD#")
        Dec = serialobject.read(10)
        pos = [RA, Dec]
        return (pos)

#--------------------------------------------------

def move_telescope(serialobject, directionNS, directionEW, duration, speed):
        #move the scope in one or two directions
        #north, south, east west OR
        #north-east, north-west, south-east, south-west,
        #for n seconds stop after that
        directionNS = directionNS.lower()
        directionEW = directionEW.lower()

        if(speed == "fast"):
                serialobject.write("#:RS#")
        else:
                #set the slew rate (second fastest)
                serialobject.write("#:RM#")
        time.sleep(twrite)

        print "scope moving.."

        if(directionNS == 'north'):
                serialobject.write("#:Mn#")
        elif(directionNS == 'south'):
                serialobject.write("#:Ms#")

        if(directionEW == 'east'):
```

```
                serialobject.write("#:Me#")
        elif (directionEW == 'west'):
                serialobject.write("#:Mw#")

        if(duration <= 0):
                print "-- warining: set duration! --"
                duration = 1

        time.sleep(duration)

        #stop all motion
        serialobject.write(":Q#")

#-----------------------------------------------------

def get_time(serialobject):
        #get local time in 24hr format
        serialobject.flushInput()
        serialobject.flushOutput()
        serialobject.write("#:GL#")
        time.sleep(twrite)
        #03:49:33 -> 8chars
        ntime = serialobject.read(8)
        time.sleep(twrite)
        return (ntime)

#-----------------------------------------------------

def set_time(serialobject, ntime):
        #get local time
        serialobject.flushInput()
        serialobject.flushOutput()
        newtime = "#:SL" + ntime + "#"
        serialobject.write(newtime)
        time.sleep(twrite)
        serialobject.flushInput()
        time.sleep(twait)

#-----------------------------------------------------

def set_UTC_offset(serialobject, offset):
        #set the time difference (+/- n.m hours)
        serialobject.flushInput()
        serialobject.flushOutput()
        tag = "#:SG"
        command = tag + str(offset) + "#"
        serialobject.write(command)
        time.sleep(twrite)
        serialobject.flushInput()
```

```
        #might need a bit more..
        time.sleep(twait)

#-----------------------------------------------------

def get_UTC_offset(serialobject):
        #get UTC offset in decimal hours
        serialobject.flushInput()
        serialobject.flushOutput()
        serialobject.write("#:GG#")
        time.sleep(twrite)
        #sHH.H -> 5char
        offset = serialobject.read(5)
        time.sleep(twrite)
        #drop the leading "1"
        if(offset[0] == "1"):
           offset = offset[1:len(offset)]
        #drop the pound
        n_offset = offset.split("#")
        utc_offset = n_offset[0]
        return (utc_offset)

#-----------------------------------------------------

def get_date(serialobject):
        #get local time in 24hr format
        serialobject.flushInput()
        serialobject.flushOutput()
        serialobject.write("#:GC#")
        time.sleep(twrite)
        #22/03/07 -> 8chars
        ndate = serialobject.read(8)
        time.sleep(twrite)
        return (ndate)

#-----------------------------------------------------

def set_date(serialobject, ndate):
        #get local date (in UTC !)
        serialobject.flushInput()
        serialobject.flushOutput()
        newdate = "#:SC" + ndate + "#"
        serialobject.write(newdate)
        time.sleep(twrite)
        serialobject.flushInput()
        time.sleep(twait)

#-----------------------------------------------------
```

```
def get_site(serialobject, sitenum):
        #get the currently selected site
        serialobject.flushInput()
        serialobject.flushOutput()
        if(sitenum == 0):
           serialobject.write("#:GM#")
        elif(sitenum == 1):
           serialobject.write("#:GN#")
        elif(sitenum == 2):
           serialobject.write("#:GO#")
        elif(sitenum == 3):
           serialobject.write("#:GP#")
        else:
           serialobject.write("#:GM#")
           print "error in sitenum ..."

        time.sleep(twrite)
        nsite = serialobject.read(10)   #(15max)
        #get everything before the pound sign
        nsite = nsite.split("#")
        sitename = nsite[0]
        time.sleep(twrite)

        return (sitename)

#--------------------------------------------------------

def get_sites(serialobject):
        #get list of sites
        serialobject.flushInput()
        serialobject.flushOutput()
        siteList = []
        #first (0-entry)
        serialobject.write("#:GM#")
        time.sleep(twrite)
        serialobject.flushInput()   #NEW
        nsite = serialobject.read(10)
        nsite = nsite.split("#")
        sitename = nsite[0]
        siteList.append(sitename)
        time.sleep(twrite)
        #second site
        serialobject.write("#:GN#")
        time.sleep(twrite)
        serialobject.flushInput()   #NEW
        nsite = serialobject.read(10)
        nsite = nsite.split("#")
        sitename = nsite[0]
        siteList.append(sitename)
```

```
        time.sleep(twrite)
        #third site
        serialobject.write("#:G0#")
        time.sleep(twrite)
        serialobject.flushInput()   #NEW
        nsite = serialobject.read(10)
        nsite = nsite.split("#")
        sitename = nsite[0]
        siteList.append(sitename)
        time.sleep(twrite)
        #forth site (3-entry)
        serialobject.write("#:GP#")
        time.sleep(twrite)
        serialobject.flushInput()   #NEW
        nsite = serialobject.read(10)
        nsite = nsite.split("#")
        sitename = nsite[0]
        siteList.append(sitename)
        time.sleep(twrite)

        return (siteList)

#--------------------------------------------------------

def set_site(serialobject, site, sitenum):
        #--------------------------------------
        #does not work ->set site via handbox...
        #if messed up -> Setup-Reset
        #--------------------------------------
        serialobject.flushInput()
        serialobject.flushOutput()

        if(sitenum == 0):
           tag = ":SM"
        elif(sitenum == 1):
           tag = ":SN"
        elif(sitenum == 2):
           tag = ":SO"
        elif(sitenum == 3):
           tag = ":SP"
        else:
           tag = ":SM"
           print "error in sitenumber..."

        #newsite = "#:S"+ chr(ord('M')+ sitenum) + " " + site + "#"
        newsite = "#" + tag + site + "#"
        print "site, sitenum and site string: ", site, sitenum, newsite
        serialobject.write(newsite)
        time.sleep(twait)
```

```
        result = serialobject.read(3)
        serialobject.flushInput()
        print "result of set site: ", result

#-------------------------------------------------------

def select_site(serialobject, sitenum):
        #select one of the previously set sites, sitenum 0,1,2,3
        serialobject.flushInput()
        serialobject.flushOutput()

        newsite = "#:"+ "W" + str(1) + "#"
        serialobject.write(newsite)
        time.sleep(twait)
        serialobject.flushInput()

#-------------------------------------------------------

def set_gps(serialobject, on_off):
        #set gps on/off - returns nothing from the scope; leider..
        serialobject.flushInput()
        serialobject.flushOutput()
        if(on_off):
           command = "#:g+#"
           print "gps on.."
        else:
           command = "#:g-#"
           print "gps OFF..."

        serialobject.write(command)
        time.sleep(twait)
        serialobject.flushInput()

#-------------------------------------------------------

def set_alignment(serialobject, amode):
        #set to land, polar or altaz mode
        serialobject.flushInput()
        serialobject.flushOutput()
        if(amode == "land"):
           command = "#:AL#"
        elif(amode == "polar"):
           command = "#:AP#"
        elif(amode == "altaz"):
           command = "#:AA#"
        else:
           print " alignmode not recognized...defaulting to altaz.."
           command = "#:AA#"
```

```
        serialobject.write(command)
        time.sleep(twait)
        serialobject.flushInput()

#-----------------------------------------------------

def get_alignment(serialobject):
        #find the alignment mode
        serialobject.flushInput()
        serialobject.flushOutput()
        command = chr(0x06)
        serialobject.write(command)
        time.sleep(twait)
        result = serialobject.read(2)
        time.sleep(twrite)
        serialobject.flushInput()
        if(result == 'A'):
           mode = "altaz"
        elif(result == 'L'):
           mode = "land"
        elif(result == 'P'):
           mode = "polar"
        else:
           mode = "unknown"

        return (mode)

#-----------------------------------------------------

def move_telescope_to_location(serialobject, location):
        #>>>depreciated (move_telescope_to_location_p instead)
           #move telescope to a precise RA (first parameter) and
           #DEC (second parameter) location
           #only int values (no minutes, seconds)


           RA = location[0]
           DEC = location[1]
           RAs = str(int(RA))
           DECs = str(int(DEC))
           RAs = ":Sr" + RAs + ":00#"

           if(DEC > 0):
                   DECs = ":Sd+" + DECs + "*00#"
           else:
                   DECs = ":Sd-" + DECs + "*00#"

           controls = ":Gr#:GR#:Gd#:GD#"
```

```
        #set the slew rate (fastest)
        serialobject.write("#:RS#")
        time.sleep(twrite)

        print "scope moving to new position..."

        serialobject.write(RAs + DECs + controls)
        time.sleep(twrite)
        returnval = serialobject.read(42)

        #if (returnval[0:1] == str(1)):
        # only then execute -
        # else: print ("-- check the coordinates: not moving! --")

        #else:
        #execute the command
        serialobject.write(":MS#")
        time.sleep(tslew)

#-------------------------------------------------------

def move_telescope_to_location_p(serialobject, location):
        # uses the HH:MM:SS (RA) and sDD*MM:SS (DEC) format
        # RA (first parameter) DEC (second parameter) of location
        #--->high precision move here<---

    #parse the input strings by colon
        RA = location[0]
        DEC = location[1]
        DECval = DEC.split(':')
        RAval = RA.split(':')
        DEClength = len(DECval)
        RAlength = len(RAval)

    #figure out the sign
        if(int(DECval[0]) > 0):
        Sd = ":Sd+"
    else:
        Sd = ":Sd"  #we already have a '-'sign

        #round up or down and convert back to a string
        DECval_2 = str(int(round(float(DECval[DEClength - 1]))))
        RAval_2 = str(int(round(float(RAval[RAlength - 1]))))
        if(len(DECval_2) < 2):
        DECval_2 = "0" + DECval_2
        if(len(RAval_2) < 2):
        RAval_2 = "0" + RAval_2

    #create the strings
```

```
        nDEC = DECval[0] + "*" + DECval[1] + ":" + DECval_2
        nRA = RAval[0] + ":" + RAval[1] + ":" + RAval_2
        DECs = Sd + nDEC + "#"
        RAs = ":Sr" + nRA + "#"

           #combine to a single string
           controls = ":Gr#:GR#:Gd#:GD#"
        command = RAs + DECs + controls

        #debugging...
        #print RAs, DECs, RAval, DECval

        #set the slew rate (fastest)
           serialobject.write("#:RS#")
           time.sleep(twrite)

        print "scope moving to RA, DEC: ", nRA, nDEC
           serialobject.write(command)
           time.sleep(twrite)
           returnval = serialobject.read(42)

           #if (returnval[0:1] == str(1)):
           # only then execute -
           # else: print ("-- check the coordinates: not moving! --")

           #else:
           #execute the command
           serialobject.write(":MS#")
           time.sleep(tslew)

#--------------------------------------------------------

def get_battery_level(serialobject):
    #returns an value of the percentage of battery power available
    #cast to int - but check first non-int char if in single digits..

    found = 0
    attempt = 0
    limit = 24
    s_off = 17
    s_len = 2
    battery_val = -1
    str_len = 100
    tag = re.compile(r"Battery")

    #test battery level
    serialobject.flushInput()
    serialobject.flushOutput()
    serialobject.write(":EK11#:ED#")
```

```
    time.sleep(twrite)
    val = serialobject.read(str_len)
    time.sleep(twrite)

    #try first command version
    #m=re.search('Battery', val)
    m=re.search(tag, val)
    if(m>0):
        battery_val = (val[m.start()+s_off : m.start()+s_off+s_len])
        print battery_val
        found = 1

    while((found == 0) & (attempt < limit)):
        #try the alternate command
        print "checking battery level..."
        serialobject.flushInput()
        serialobject.flushOutput()
        serialobject.write(":EK68#:ED#")
        time.sleep(twrite)
        val = serialobject.read(str_len)
        time.sleep(twrite)
        m=0
        #m=re.search('Battery', val)
        m=re.search(tag, val)
        if(m>0):
            battery_val = (val[m.start()+s_off : m.start()+s_off+s_len])
            print battery_val
            found = 1
        else:
            print "check:", val, m

        attempt = attempt+1


    #escape from the status display
    serialobject.write(":EK9#")

    return (battery_val)

#-------------------------------------------------------

def telescope_motion(destination, port):
    # high level scope move with correction
    serialobject = start_telescope(port)
    time.sleep(twrite)
    [RA,DEC] = get_telescope_position(serialobject)
    print "start RA:", RA, "start DEC:", DEC
    #set the slew rate (fastest)
    serialobject.write("#:RS#")
```

```
    #move to the destination
    move_telescope_to_location_p(serialobject, destination)
    [RA,DEC] = get_telescope_position(serialobject)
    print "current RA:", RA, "current DEC:", DEC

    #new: check the difference between desired and achieved goal
    currentposition = RA,DEC
    val = close_enough(destination, currentposition, threshold)
    if(val == 0):
       print "correcting..."
          move_telescope_to_location_p(serialobject, destination)
          [RA,DEC] = get_telescope_position(serialobject)
          print "end RA:", RA, "end DEC:", DEC
    else:
          print "no correction required."

    finished = stop_telescope(serialobject)
    print finished
#-------------------------------------------------------------
```