

NMEA_(GPS)_Location_Viewer

Good starter code for working with GPS data:

Connects to a bluetooth GPS receiver and parses the NMEA sentence, printing the location to the screen.

From [Nick's Code](#) - check there for any updates.

```
# Basic script to grab the current location from an NMEA bluetooth GPS,
# and log it to the screen
# Also optionally logs periodic location data to a file, so you can see
# where you were
#
# GPL
#
# Nick Burch - v0.01 (15/01/2006)

import appuifw
import socket

# Bluetooth address to connect to
# Change this to your GPS's bluetooth address!
gps_addr='00:15:4B:01:14:EE'

# How many GGA sentences between logging
# Set to 0 to prevent logging
gga_log_interval = 10
gga_log_count = 0

# File to log GGA sentences into
gga_log_file = 'e:\\nmea_gga_log.txt'

# We want icons etc
appuifw.app.screen='normal'

# Define title etc
appuifw.app.title=u"NMEA Location Disp"

# Alert them to the GPS we're going to connect to
appuifw.note(u"Will connect to GPS %s" % gps_addr, 'info')

# Connect to the bluetooth GPS using the serial service
sock=socket.socket(socket.AF_BT, socket.SOCK_STREAM)
target=(gps_addr,1)
sock.connect(target)

appuifw.note(u"Connected to the GPS")

# Open the GGA log file, in append mode
gga_log_fh = open(gga_log_file, 'a');

# This is set to 0 to request a quit
going = 1

#####

# Generate the checksum for some data
# (Checksum is all the data XOR'd, then turned into hex)
def generate_checksum(data):
    csum = 0
    for c in data:
        ordsum = csum ^
```

NMEA_(GPS)_Location_Viewer

```
    hex_csum="% csum
return hex_csum.upper()

# Format a NMEA timestamp into something friendly
def format_time(time):
    time_h0=2]
    time_m12=4]
    time_s4=]
return "%s:%s:%s UTC" % (hh,mm,ss)

# Format a NMEA date into something friendly
def format_date(date):
    months='Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')
    d0=2]date
    m12=4]date
    yy4=6]date
    yyyy= + 2000
return "%s %s %d" % (dd, months[(int(mm)-1)], yyyy)

#####

def readline(sock):
    """Read one single line from the socket"""
    line =
while 1:
    recv(sock)
if not char: break
    line += char
if char == "\n": break
return line

def exit_key_pressed():
    """Function called when the user requests exit"""
# TODO Figure out why going in the main thread isn't updated
going = 0
app.quit_key_handler = None

#####

# Get the location from a GGA sentence
def get_gga_location(data):
    dspldata(á,')
    {}ret =
    ['type' = 'GGA'
    ['lat' = "%s%s" % (d[1],d[2])
    ['long' = "%s%s" % (d[3],d[4])
    ['time' = format_time(d[0])
return ret

# Get the location from a GLL sentence
def get_gll_location(data):
    dspldata(á,')
    {}ret =
    ['type' = 'GLL'
    ['lat' = "%s%s" % (d[0],d[1])
    ['long' = "%s%s" % (d[2],d[3])
    ['time' = format_time(d[4])
return ret

# Get the location from a RMC sentence
def get_rmc_location(data):
    dspldata(á,')
```

NMEA_(GPS)_Location_Viewer

```
{ }ret =
['type'] = 'RMC'
['lat'] = "%s%s" % (d[2],d[3])
['long'] = "%s%s" % (d[4],d[5])
['time'] = format_time(d[0])
return ret

#####

# Loop while active
appuifw.app.exit_key_handler = exit_key_pressed
while going == 1:
    rawdata = sock.recv(1024)
    if not rawdata: break
    data = rawdata

# Ensure it starts with $GP
if not data[0:3] == '$GP':
    continue

# If it has a checksum, ensure that's correct
# (Checksum follows *, and is XOR of everything from
# the $ to the *, exclusive)
if data[-3] == '*':
    exp_checksum = get_checksum(data)
    if not exp_checksum == data[-2:]:
        print "Invalid checksum %s, expecting %s" % (data[-2:], exp_checksum)
        continue

# Strip the checksum
[:-3]data = data

# Grab the parts of the sentence
talker = data[1:3]
sentence_id = data[3:6]
sentence_data = data[7:]

# The NMEA sentences we're interested in are:
# GGA - Global Positioning System Fix Data
# GLL - Geographic Position
# RMC - GPS Transit Data
location = {}
if sentence_id == 'GGA':
    location = get_gga_data(sentence_data)

# Log GGA packets periodically
gga_log_count = gga_log_count + 1
if gga_log_count == gga_log_interval:
    gga_log_count = 0
    write(rawdata, gga_log_fh)
if sentence_id == 'GLL':
    location = get_gll_data(sentence_data)
if sentence_id == 'RMC':
    location = get_rmc_data(sentence_data)

# If we got a location, print it
if not location == {}:
    # Check the location is valid
    if location['lat'] == '0000.0000N' and location['long'] == '0000.0000E':
        print "Invalid GPS location found"
    else:
        print "Source of location is %s" % location['type']
```

NMEA_(GPS)_Location_Viewer

```
print "Lat is %s" % location['lat']
print "Long is %s" % location['long']
print "Time is %s" % location['time']
print ""
```

```
# Future:
# Grab satellites in view from GSV
# Grab satellites in use from GSA
# Grab speed and direction from VTG
```

```
# For debugging
#print data
```

```
# All done
sock.close()
gga_log_fh.close()
```

```
print "All done"
appuifw.app.set_exit()
```