Python GPS Module - cgps.py

This library provides access to the PITS+ / PITS Zero GPS module.

Basic Usage

```
from cgps import *
from time import sleep

mygps = GPS()

while 1:
    time.sleep(1)
    print ("Position: ", mygps.Position())
```

When the GPS object is created, it starts to care of all the GPS communications. The current time and position can then be retrieved as required, however note that depending on the position of the GPS aerial, it can take some time before this information is available.

The time and position are returned as a dict, for example

```
 {'lat': 51.95014, 'alt': 171, 'sats': 12, 'lon': -2.54445, 'time': '10:42:35', 'fix': 2}
```

Callbacks

Rather than poll the library (as in the above example), it's neater to provide callback functions so that the library informs us when a new position has arrived.

```
from cgps import *
import time

def NewPosition(Position):
    print("Callback: ", Position)

def LockChanged(GotLock):
    print("Lock " + ("GAINED" if GotLock else "LOST"))

print("Creating GPS object ...")
mygps = GPS(WhenNewPosition=NewPosition, WhenLockChanged=LockChanged)

print("loop ...")
while 1:
    time.sleep(1)
```

Reference

Object Creation

```
GPS(WhenNewPosition=None, WhenLockChanged=None)
```

WhenNewPosition and WhenLockChanged are callbacks (see below).

Functions

```
Position()
```

returns the current GPS position

Callbacks

```
WhenNewPosition(Position)
```

This is called when a new position is received from the GPS (one per second). "Position" is a dict:

```
{'lat': 51.95014, 'alt': 171, 'sats': 12, 'lon': -2.54445, 'time': '10:42:35', 'fix': 2}
```

Note that all fields are zeroes until the GPS gains a lock.

```
WhenLockChanged(HaveLock)
```

This is called when the GPS lock status changes. "HaveLock" is True or False

Dependencies

- Needs the external **gps** program be present in ../gps
- Requires that the Python module **psutil** be installed

Behind the scenes

This is a wrapper from the separate C GPS program, providing basic GPS functions intended for use in a HAB (High Altitude Balloon) tracker. It starts the program automatically as required.

That program works with a the UBlox GPS module on a PITS+ or PITS Zero board, using a software i2c implementation. We cannot use the Pi I2C ports because of a hardware limitation in the BCM processor, and we cannot use the Pi serial port if (as is likely) that is in use for RTTY transmissions. It handles the required "flight mode" of the UBlox module, so that it works at altitudes of up to 50km.

The GPS program is a socket server, sending JSON position data at a rate of 1 position per second, using port 6005. Currently it accepts one client at a time only.

Testing

Run the supplied test program:

```
python3 test_cgps.py
```