gregcope / L76micropyGPS

README.md

L76micropyGPS

A `pytrack L76` class that feeds a `micropyGPS` object POC. Based on the excellent `micropyGPS`, that is in turn based on the most excellent `tinyGPS++` `C++` Ardunio class.

The idea is to take advantage of microPython threading to setup a `GPS/GNS` system that in the background is constantly updating its data based on the `NMEA` data is receives, without having to constantly call it to update itself. It does this by having a thread that constantly reads the `NMEA` sentences from the devices and parses them into `GPS/GNS` variables so that they can be read by python code.

This allows you to fire up the `GPS/GNS pytrack L76` module, go do other things like check other sensors, or setup a WIFI/HTTP[S]/LTE/LoRaWAN connection, then go back and see if you have `GPS/GNS` data worth using yet.

Intro

Usage

1. Checkout the repo
2. Flash your device
3. Read the `main.py` for examples. `boot.py` does very little bar start a timer for testing and the UART to print them out.

Own usage

Use the `main.py` code as an example. The `print` statements can be removed as they are there for memory / displaying to console.

NOTES

Presently the `pycoproc` code assumes an `i2c` ID of `0` here; https://github.com/pycom/pycom-libraries/blob/master/lib/pycoproc/pycoproc.py#L78 If you use another `i2c` device give it an ID above 0! `thing.i2c = I2C(C1, foo, bar)`

Thanks to `tttadam` on the pycom forums for spotting; https://forum.pycom.io/topic/3870/pytrack-gps-library/10

Todo

- Add something to stop thread