

Instructions and guide to the AR Drone 2.0 GPS addition

Vsevolod Igorevitch Kiriouchine

20-06-2020

ORIGINAL 02-11-2017

With thanks to:

DR. IR. R. TOTH

DR. IR. I. BAROSAN

IR. S. MARX

IR. P.B. COX

DAREN LEE

TOD KURT

Eindhoven University of Technology

Faculty of Mechanical Engineering

In cooperation with the Faculty of Electrical Engineering

Contents

0.1	Introduction	2
0.2	Standalone code	2
0.2.1	Software for standalone code	2
0.2.2	Compiling the standalone C driver	3
0.3	Matlab version	9
0.4	Code	13
0.4.1	standalone C GPS code	13
0.4.2	arduino-serial-lib	13
0.4.3	FTP via Matlab	13
0.4.4	GPS driver Matlab	14
0.4.5	GPS driver Matlab	14

0.1 Introduction

This document describes the GPS C driver and the use of the GPS incorporation of [1] into the Matlab ARDrone 2.0 toolbox of [2] as well as a standalone version. In this document all needed software is either linked if provided by another company, or is pasted inside this document in the Code section 0.4 and should be in the accompanying archived folder **Files_GPS_ardrone.zip**.

0.2 Standalone code

0.2.1 Software for standalone code

The standalone version does not require the use of Matlab. It uses the following software:

- A gcc compiler: here the Code Sourcery Gnueabi compiler is used from <https://sourcery.mentor.com/srgpp/lite/arm/portal/package8738/public/arm-none-linux-gnueabi/arm-2011.03-41-arm-none-linux-gnueabi.exe>, make sure to execute in Windows 7 compatibility mode when installing
- PuTTY: a free telnet program from <http://www.putty.org/>
- Ublox ucenter: the free GPS configuration (and preview) program
- Possibly WinFTP or another FTP program: <https://winscp.net/eng/download.php>

The incorporation of an external GPS module is based on the use of the USB port on top of the Parrot AR Drone. The configuration of the quad-copter and the GPS is shown in figure 1. How the TLL 'ttl-232r-3v3' module is connected to the GPS module (note the wire colours) is shown in figure 2.

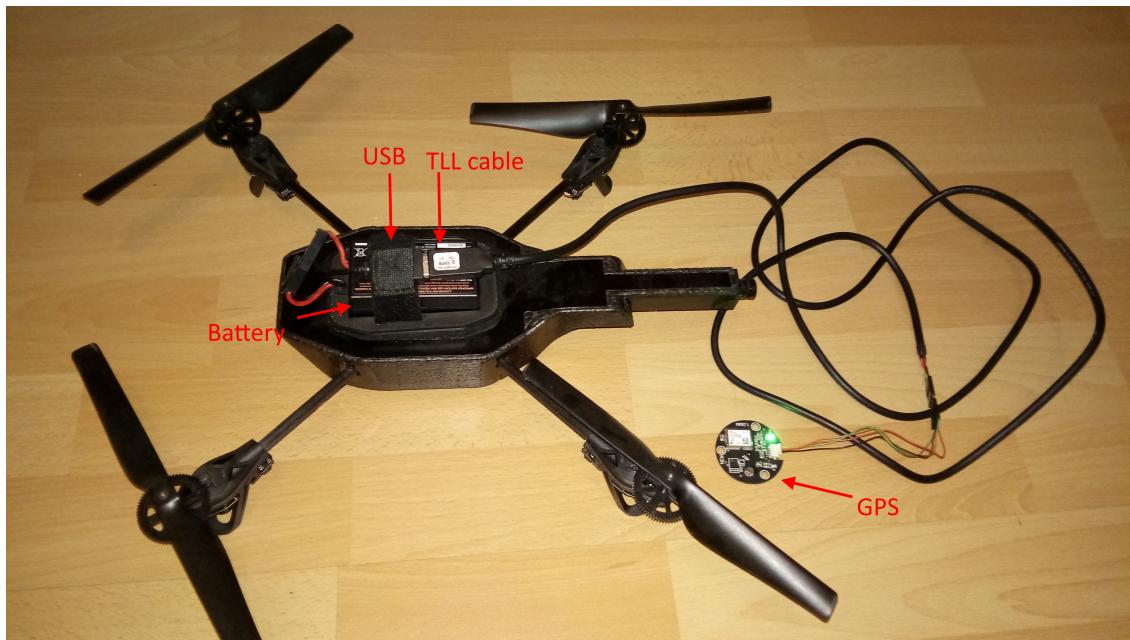


Figure 1: The configuration of the quad-copter and GPS module.

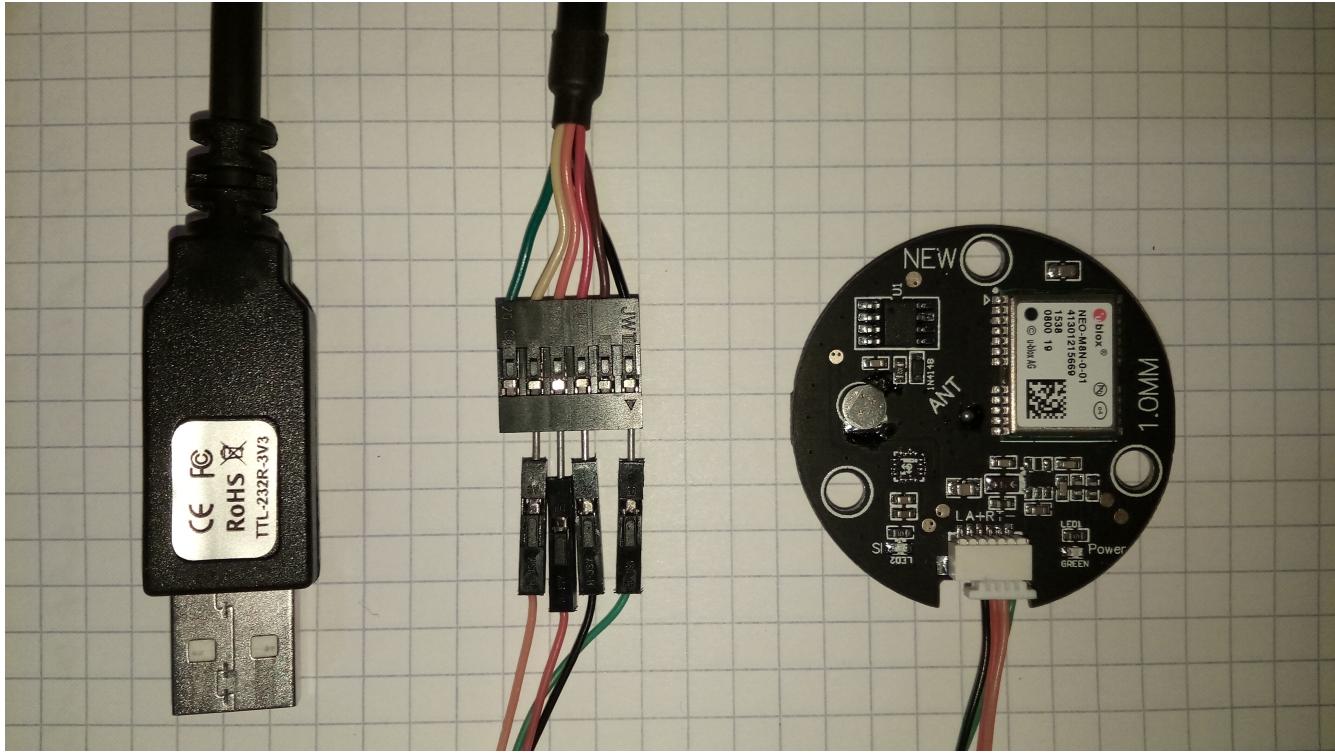


Figure 2: The used NEO-M8N-0-01 GPS module with the FTDI TTL 'ttl-232r-3v3' cable with the colour-coded transferring shown.

Before use, the module should be tested with Ublox ucenter software (available for free on Windows) using the usb to connect with your computer or laptop to make sure the Module is working well.

0.2.2 Compiling the standalone C driver

The first step in using the stand alone version is in installing a gcc compiler, such as: the Code Sourcery Gnueabi compiler is used from <https://sourcery.mentor.com/sgrpp/lite/arm/portal/package8738/public/arm-none-linux-gnueabi/arm-2011.03-41-arm-none-linux-gnueabi.exe>, make sure to execute in Windows 7 compatibility mode when installing.

The compiling happens through your command prompt. For this search in windows for 'cmd' and open it. First make sure that the corresponding code is in the right folder. In this case there are 3 files needed: the C GPS driver ,named `gps_openC_backup_WorkingLongiLatiAutom.c` , shown in section 0.4.1. The `arduino-serial-lib.c` from <https://todbot.com/blog/2013/04/29/arduino-serial-updated/> [3]. And it's header `arduino-serial-lib.h` (in section 0.4.2)

In command prompt, navigate to the folder where you have placed the 3 aforementioned files and input the command:

```
arm-none-linux-gnueabi-gcc -Wall gps_openC_backup_WorkingLongiLatiAutom.c c:/Users/vsevolod/arduino-serial-lib.c -o gpsOpenC.elf
```

```

Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\vsevolod>arm-none-linux-gnueabi-gcc -Wall gps_OpenC_backup_WorkingLongLatAutom.c c:/Users/vsevolod/arduino-seglib-lib.c -o gpsOpenC.elf
gps_OpenC_backup_WorkingLongLatAutom.c: In function 'main':
gps_OpenC_backup_WorkingLongLatAutom.c:39:7: warning: unused variable 'i'
gps_OpenC_backup_WorkingLongLatAutom.c:35:6: warning: unused variable 'tmp1'

C:\Users\vsevolod>

```

Figure 3: The cmd with the compiler code and the return

Multiple enter presses may be needed after copy paste or a long time to wait . The compiler also warns that there are some unused variables in some lines. The compiler could also give errors if your code does not compile, in which case the line and place number is given where the error occurred to help with trouble shooting .

This uses the 'arm-none-linux-gnueabi-gcc' compiler with the main code `gps_OpenC_backup_WorkingLongLatAutom.c` and the extra code `arduino-serial-lib.c` to create the Linux executable file `gpsOpenC.elf`.

In this case the path is `c:/Users/vsevolod/` , to compile substitute your own path to the files.

Now that the `gpsOpenC.elf` is made, the PuTTY can be installed and opened. Upon opening PuTTY, make a wifi connection to the quad-copter and configure Putty. Create a new session with the IP 192.168.1.1 and the port 23 with the type Telnet, as seen in figure 4, then you can save the configuration for future use. Press 'Open' to connect to the quad-copter.

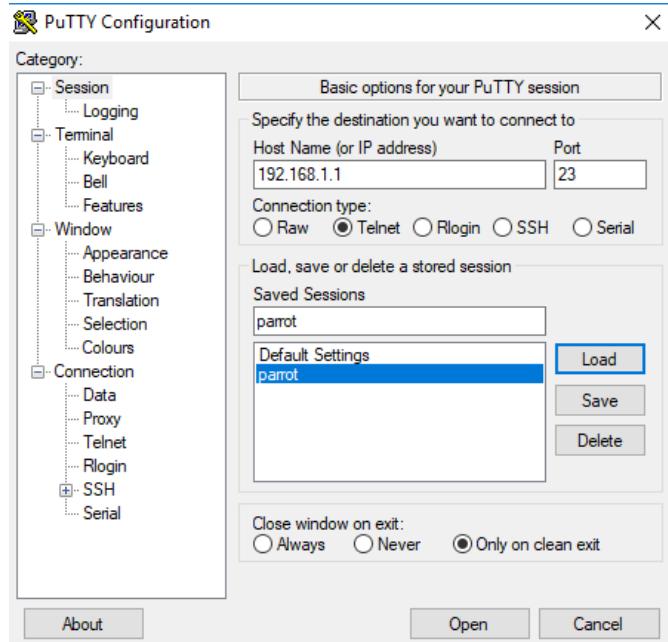


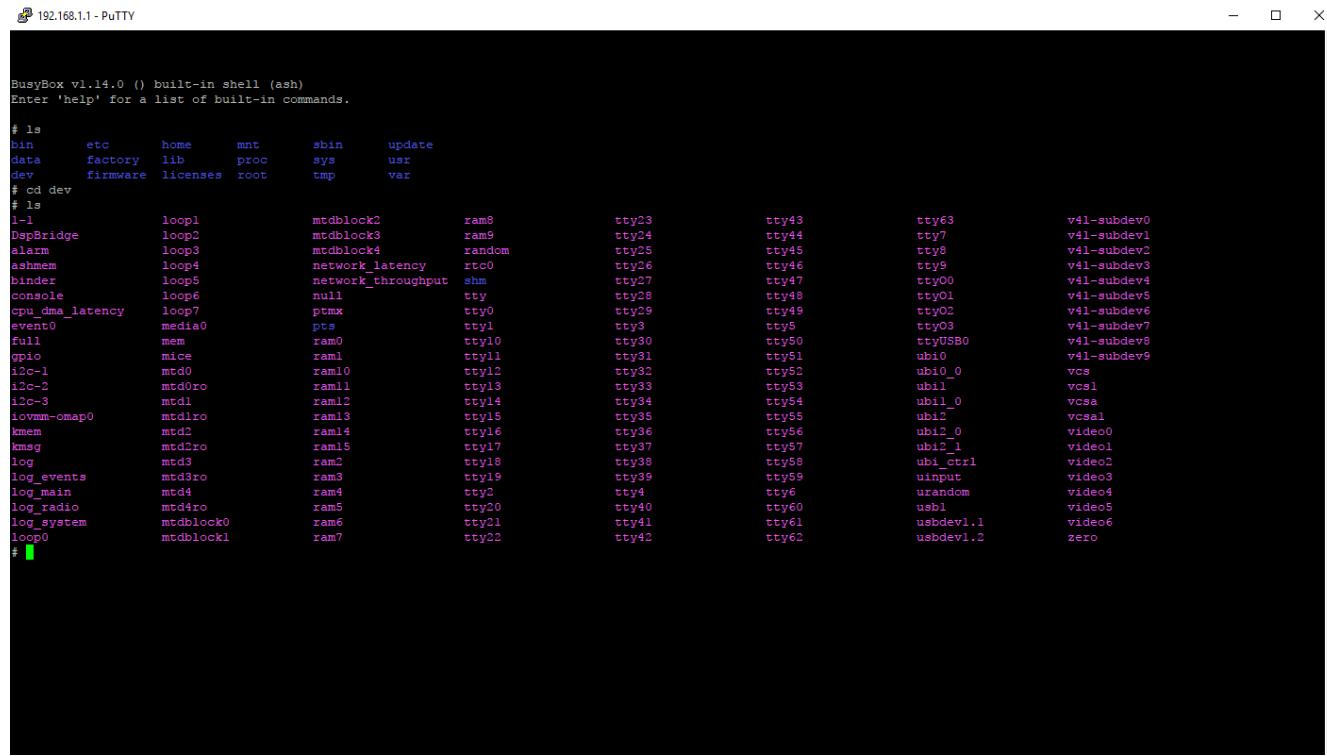
Figure 4: The PuTTY starting window

A new black window should appear. This window uses Linux commands to navigate, such as; 'ls' to view a list of folders, 'cd dev' to move to the dev folder (or any other), 'cd ..' to go up the folder path, and others.

For now type 'ls' and Enter to view the folders, then type 'cd dev' to go to the dev folder and type 'ls' again, as in figure 5. If the GPS module is connected properly the pink name `ttyUSB0` or `ttyUSB1` should be seen, this is the virtual directory path that the driver uses, it should be `ttyUSB0`, if it is not, reconnect the GPS usb module from the quad-copter until the `ttyUSB0` is present here (usually this happens automatically and this step can be skipped).

Now move to the update folder, by using 'cd ..' and 'cd update'. When 'ls' is used, the only file present is `version.txt` or none. Here the file `gpsOpenC.elf` should be put onto the drone. This can be done using the WinSCP program or using Matlab's build in FTP as seen in figure 6. Note that the code is also given in the last section for using Matlab's FTP (Note that the transferred file `gpsOpenC.elf` originated from the same folder as is the working folder in Matlab), again however Matlab is not necessarily required for this step and another (free) FTP can be

used, hence the name standalone.



```
BusyBox v1.14.0 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls
bin      etc      home      mnt      sbin      update
data     factory   lib       proc      sys       usr
dev      firmware licenses  root      tmp       var

# cd dev
# ls
l-1      loop1    mtdblock2    ram8      tty23    tty43    tty63    v4l-subdev0
DspBridge  loop2    mtdblock3    ram9      tty24    tty44    tty7     v4l-subdev1
alarm    loop3    mtdblock4    random   tty25    tty45    tty8     v4l-subdev2
ashmem   loop4    network_latency rtc0    tty26    tty46    tty9     v4l-subdev3
binder   loop5    network_throughput shm    tty27    tty47    tty00   v4l-subdev4
console  loop6    null        tty     tty28    tty48    tty01   v4l-subdev5
cpu_dma_latency  loop7    ptmx       pts      tty0    tty49    tty02   v4l-subdev6
event0   media0   ram0       ram1    tty1    tty5    tty03   v4l-subdev7
full     mem      ram0       ram1    tty0    tty30   tty50   ttyUSB0  v4l-subdev8
gpio     mice     raml       ram1    tty1    tty31   tty51   ubi0   v4l-subdev9
i2c-1    mtd0     raml0      ram1    tty2    tty32   tty52   ubi0_0  vcs
i2c-2    mtd0rro  raml1      ram1    tty3    tty33   tty53   ubi1   vcs1
i2c-3    mtd1     raml2      ram1    tty4    tty34   tty54   ubi1_0  vcsa
iovmm-omap0  mtd1rro raml3      ram1    tty5    tty35   tty55   ubi1_~  vcsal
kmem    mtd2     raml4      ram1    tty6    tty36   tty56   ubi2_0  video0
kmag    mtd2rro  raml5      ram1    tty7    tty37   tty57   ubi2_1  video1
log     mtd3     raml6      ram1    tty8    tty38   tty58   ubi_ctrl video2
log_events  mtd3rro  raml7      ram1    tty9    tty39   tty59   uinput  video3
log_main   mtd4     ram2       ram4    tty0    tty40   tty60   urandom video4
log_radio  mtd4rro  ram3       ram4    tty1    tty41   tty61   usb1   video5
log_system  mtdblock0 ram6       ram7    tty2    tty42   tty62   usbdev1.1 video6
loop0   mtdblock1 ram7       ram7    tty3    tty43   tty63   usbdev1.2 video7
#
```

Figure 5: The PuTTY opened window

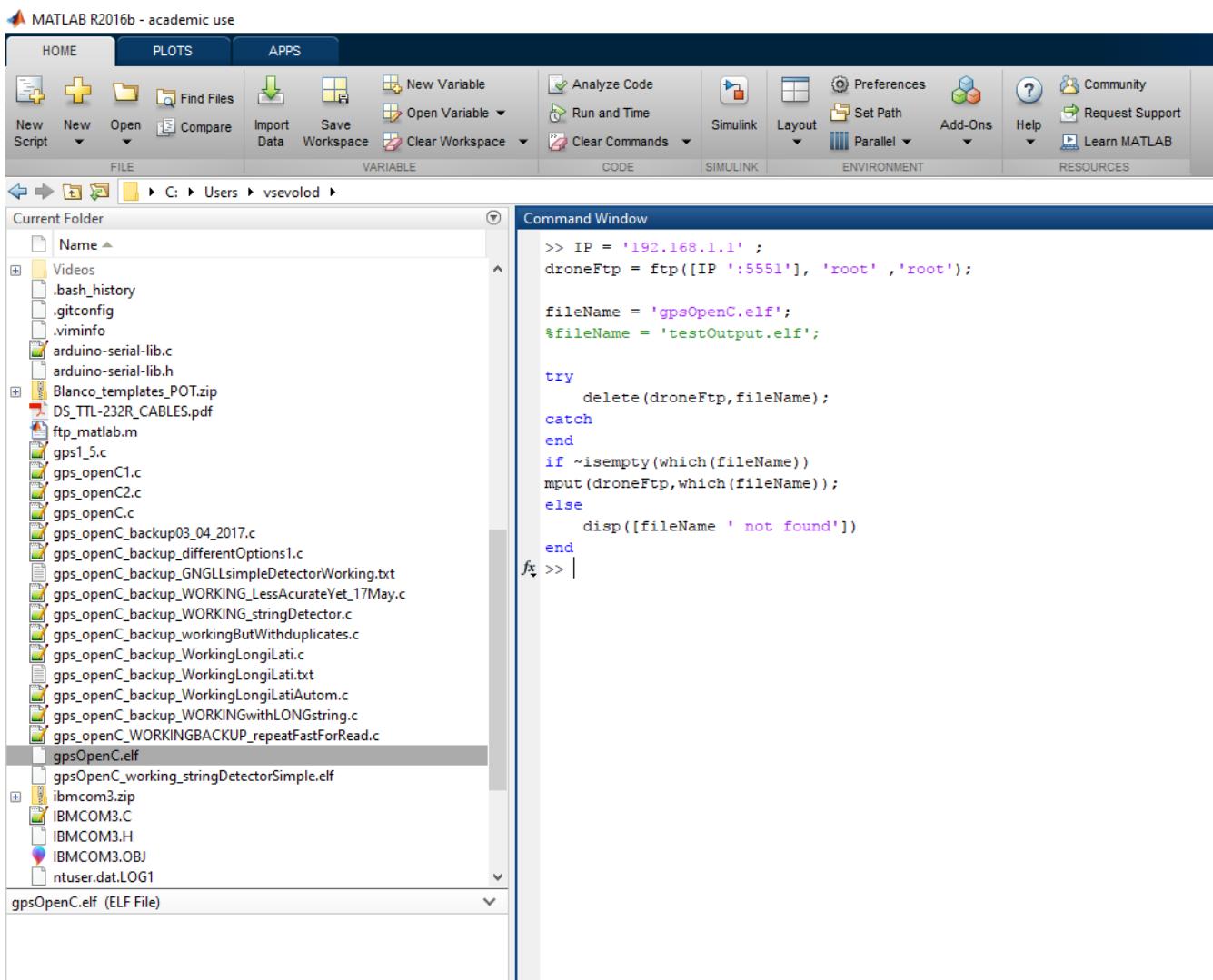


Figure 6: FTP via Matlab

Once the file `gpsOpenC.elf` is transferred into the update folder it should be made executable using the command:

```
chmod 777 gpsOpenC.elf
```

When the command 'ls' is used again, it should appear green in putty.



```
binder          mtdl          ram6          tty29          tty56          v4l-subdev0
console         mtdlro        ram7          tty3           tty57          v4l-subdev1
cpu_dma_latency mtd2          ram8          tty30          tty58          v4l-subdev2
event0          mtd2ro        ram9          tty31          tty59          v4l-subdev3
full            mtd3          random        tty32          tty6           v4l-subdev4
gpio             mtd3ro        rtc0          tty33          tty60          v4l-subdev5
i2c-1            mtd4          shm           tty34          tty61          v4l-subdev6
i2c-2            mtd4ro        tty            tty35          tty62          v4l-subdev7
i2c-3            mtdblock0   tty0          tty36          tty63          v4l-subdev8
iovmm-omap0     mtdblock1   tty1          tty37          tty7           v4l-subdev9
kmem             mtdblock2   tty0          tty38          tty8           vcs
kmsg             mtdblock3   tty11         tty39          tty9           vcs1
log              mtdblock4   tty12         tty4           tty00          vcsa
log_events       network_latency  tty13         tty40          tty01          vcsal
log_main         network_throughput  tty14         tty41          tty02          video0
log_radio        null          tty15         tty42          tty03          video1
log_system       ptmx          tty16         tty43          ttyUSB0        video2
loop0            pts           tty17         tty44          ubi0           video3
loop1            ram0          tty18         tty45          ubi0_0         video4
loop2            raml          tty19         tty46          ubil           video5
loop3            raml0         tty2          tty47          ubil_0         video6
loop4            raml1         tty20         tty48          ubi2           zero
loop5            raml2         tty21         tty49          ubi2_0        video7
loop6            raml3         tty22         tty5           ubi2_l        video8
loop7            raml4         tty23         tty50          ubi_ctrl      video9
media0           raml5         tty24         tty51          uinput
#
# cd ..
# ls
bin      dev      factory   home    licenses  proc      sbin      tmp      user
data    etc      firmware  lib      mnt      root      sys      update  var
# cd update/
# ls
gpsOpenC.elf  version.txt
# chmod 777 gpsOpenC.elf
# ls
gpsOpenC.elf  version.txt
#
```

Figure 7: Green file

Next the file is run, using:

```
./gpsOpenC.elf
```

If used correctly the GPS gives the location:

```

192.168.1.1 - PuTTY

loop0 pts    tty7      tty44      ubi0      video3
loop1 ram0   tty18     tty45      ubi0_0    video4
loop2 raml   tty19     tty46      ubi1      video5
loop3 raml0  tty2      tty47      ubi1_0    video6
loop4 raml1  tty20     tty48      ubi2      zero
loop5 raml2  tty21     tty49      ubi2_0   ubi_ctrl
loop6 raml3  tty22     tty5      ubi2_1
loop7 raml4  tty23     tty50      ubi_input
media0 raml5  tty24     tty51      ubi_input

# cd ..
# ls
bin      dev      factory   home    licenses  proc      sbin      tmp      usr
data    etc      firmware lib      mnt      root      sys      update  var

# cd update/
# ls
gpsOpenC.elf  version.txt
# chmod 777 gpsOpenC.elf
# ls
gpsOpenC.elf  version.txt
# ls
gpsOpenC.elf  version.txt
# ls
gpsOpenC.elf  version.txt
# ./gpsOpenC.elf
open errno code is = 0
GPS_fd file descriptor is = 6
nbytes to read is = 1
the read char tmp is =
GNGLL,51.1511,N,00601.0000,E,2013.00,A,A*
GNGLL string found
51.1511
GPS longitude = 51.1511
00601.0000
GPS latitude = 6.01540

```

Figure 8: The GPS gives location

It is the longitude and latitude in degrees, named 'GPS longitude' and 'GPS latitude', and the degree and minute decimated output before it.

It is also possible that the program times out, in this case rerun it with `./gpsOpenC.elf` (for fast use press the up arrow to see the previous command), this stand alone version namely searches for the GNGLL string for a set amount of iterations, the Matlab version does this continuously and provides an updating stream of data.

It is also useful to use the Ublox ucenter software (available for free on Windows) for configuring the GPS module via USB using your computer. The module can be configured with Baud rates, for example the standalone version uses 9600.

The Matlab version uses 4800. The reason why Matlab uses this rate is update frequency. For higher accuracy the gps module can be configured in Ublox ucenter to update at 5Hz and only send the GNGLL string, this means that less information needs to be sent and thus for robustness the baud rate is lowered to 4800. Thus when using the Matlab version, either configure the module in ucenter to the 4800 baud rate (in the PRT(Ports) configuration menu) or adapt the software.

Below is a screen with the menu in Ublox ucenter, for more information read the user guide. The configuration uses two screens. The first is where the PRT (port) menu is used to adapt the baud rate , once done click the send button below. The next is the NMEA menu with the to send strings, right click on all the strings and Disable Message for all the string except the GNGLL, and also click Send on the bottom of the screen to save the preferences. The GNSS(GNSS Config) on the left window can be used to use both GLONASS and GPS for higher accuracy. The CFG(Configuration) menu is used to select the devices for long storage with the Send command. And finally RATE(Rates) menu can be used to set the Measurement Period to 200ms to obtain a 5Hz sampling period for fast and accurate estimation.

The result should be a high frequency update signal, this is very useful when used in Matlab with the filters described in [1].

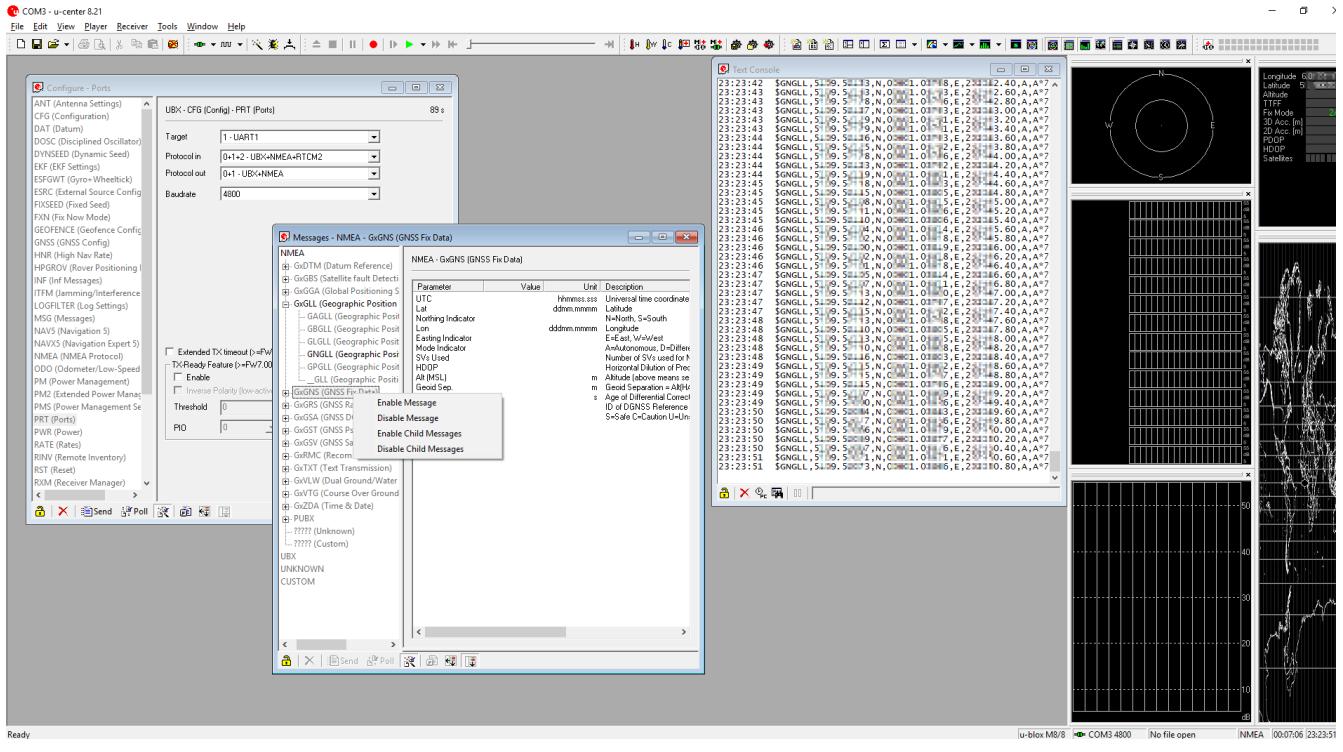


Figure 9: The GPS menu

0.3 Matlab version

For the Matlab the code does not need to be compiled manually and the Matlab version 2016a or 2016b is needed. Later and earlier versions will not work as the pathing of the Simulink files is different in other versions. This means that the entire toolbox will not be buildable [2], however the GPS driver portion will still work if the project path is adapted correctly, hence the high usefulness of the standalone GPS driver (as well as the usefulness of the Matlab version of the driver which could also be used in other applications).

For the Matlab version the toolbox needs to be installed, for the description view [2] . Once the toolbox is installed, files simply need to be copied and pasted in place.

Inside of C:\Users\vsevolod\Desktop\presentation_a\AR_Drone_Target\blocks (or your own path instead of C:\Users\vsevolod\Desktop\presentation_a) paste the following files:

- arduino-serial-lib.c
- arduino-serial-lib.h
- GPS_Read.c (see section 0.4.4)
- GPS_Read.h
- Generate_AR_Drone_S_Functions.m (Rewrite with the new version, seen in the code section 0.4.5)

Name	Git
blocks	.
Images	
videolib	
act_init.c	green
act_init.h	green
Actuators.c	green
Actuators.h	green
AR_Drone_2_Library.slx	green
AR_Drone_Actuators.slx	green
AR_Drone_Networking.slx	green
AR_Drone_Positioning.slx	green
AR_Drone_Sensors.slx	green
AR_Drone_Utils.slx	blue
ArClock.c	green
ArClock.h	green
ARDrone_LED.c	blue
ARDrone_LED.mexw64	blue
ARDrone_LED.tlc	blue
ARDrone_Motor.c	blue
ARDrone_Motor.mexw64	blue
ARDrone_Motor.tlc	blue
ARDroneSparseVideoViewer.m	green
ARDroneVideoViewer.m	green
arduino-serial-lib.c	blue
arduino-serial-lib.h	blue
ArGetDroneClock.c	blue
ArGetDroneClock.mexw64	blue
ArGetDroneClock.tlc	blue
ARprintf.c	green
ARprintf.h	green
ArPrintOnDrone.c	blue
ArPrintOnDrone.mexw64	blue
ArPrintOnDrone.tlc	blue
Battery_Sfcn_mex.c	blue
Battery_Sfcn_mex.mexw64	blue
Battery_Sfcn_mex.tlc	blue
BatteryFault_ModeType.m	green
BatteryMeasure.c	green
BatteryMeasure_Wrapper.c	green
check_init_block.m	green
Drone_StateModeType.m	green
Generate_AR_Drone_S_Functions.m	blue
GPIO.c	green
GPIO.h	green
GPS_Read.c	blue
GPS_Read.h	blue
GPSRead.c	blue
GPSRead.h	blue

Figure 10: The files copied

Run the file `Generate_AR_Drone_S_Functions.m` inside Matlab to generate the blocks inside the library. The matlab m-file also creates the extra generated '`GPSRead.c`', this however is not interesting and happened automatically, thus should be paid no attention to.

The generated block should be inside the Utilities library (some moving of the blocks might be necessary to see the gps block).

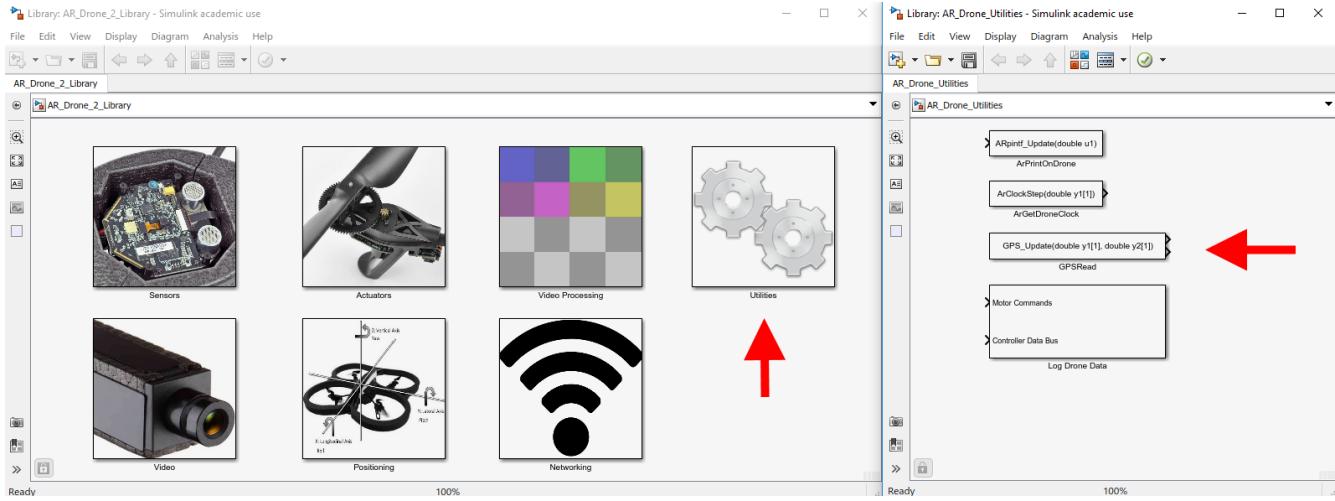


Figure 11: Utilities, you can click the lock button in the bottom left to move the blocks

Now the `GPS_Update` block can be used, where the first output `y1` is the longitude (in degrees) and the `y2` is the latitude (also in degrees), the data is not in the degree and minute decimated form, but is all in degrees.

The data comes in a spiked continuation, since the reading of the usb mostly outputs 0,0 when the GNGLL string is not found, and is only a relevant value when the GPS module is sending a string. To combat this a simple example is made in Simulink that uses memory blocks and triggers on a non zero value, thus always giving a relevant answer using the zero order hold method. This model can be in any folder, such as the root folder of the project, it is called `gps_test_query1.slx` and is seen in figure 12.

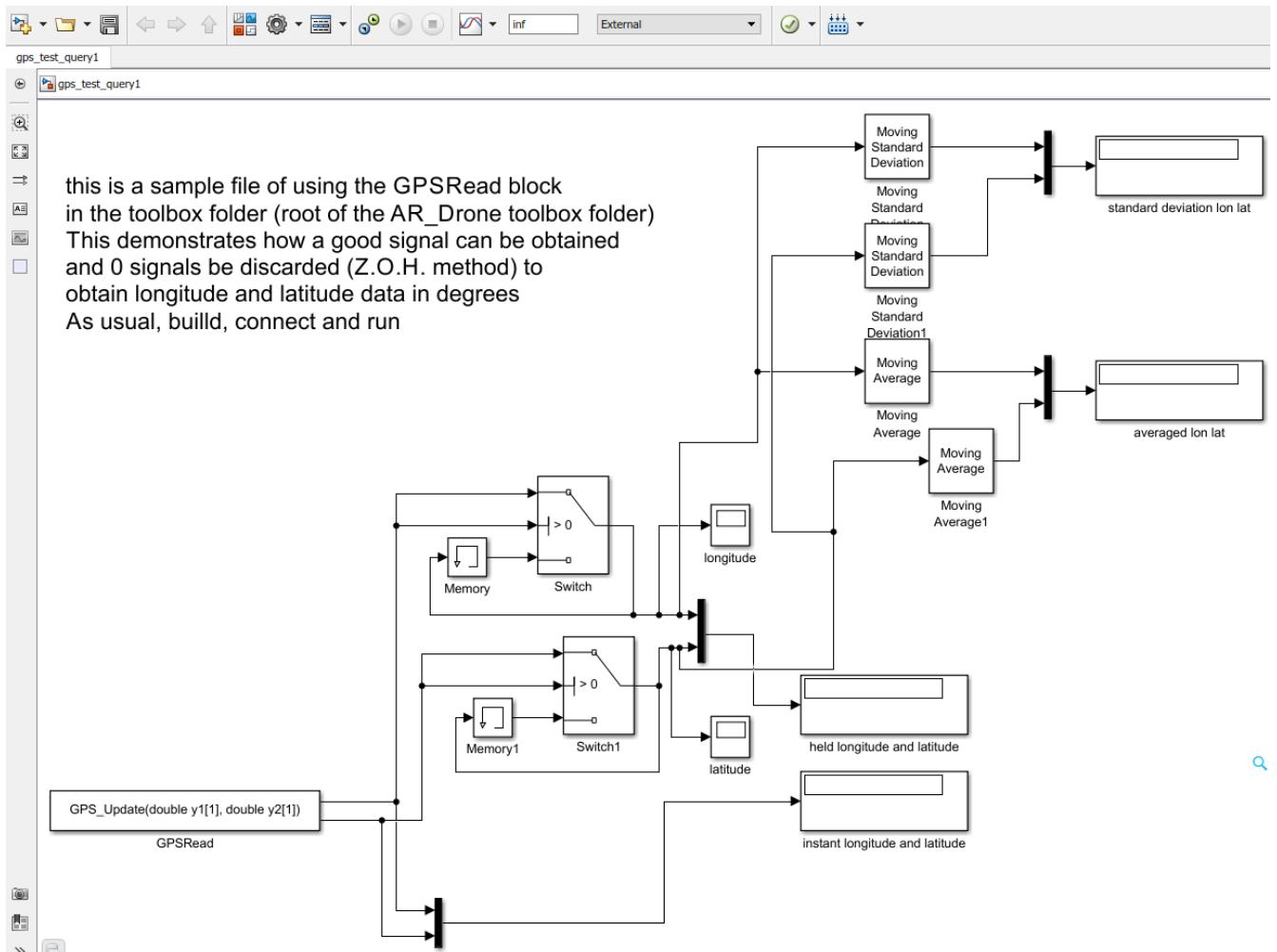


Figure 12: An example

The **GPS_Update** gives the values as seen in the blocks 'instant longitude and latitude' , when the project is Build, Connected and Ran (in External mode in this particular example) (like with other simulations in this toolbox), this block mainly outputs 0,0 and briefly the correct value.

To view the correct value continuously (with a 5Hz update frequency) a 'Switch' and 'Memory' block is used in tandem to skip the 0,0 values and only obtain the relevant values which are seen in the scopes. In this simple example moving average filters are used for checking the accuracy, however it is recommended that other state observers are used for flight in the open [1].

Thank you for reading, for further information please contact V.I.Kiriouchine*at*alumnus.tue.nl, also please use proprietary and other code and software with the permission of the author(s) of their respective work.

0.4 Code

In the original (2017) this section would have code, however, seeing as how a published version is not allowed to have code due to licensing from several parties, it is left blank for reference. For more information please contact me at v.i.kiriouchine*at*alumnus.tue.nl .

0.4.1 standalone C GPS code

The standalone C GPS code, named `gps_openC.c`: To view the cleaned up code please visit my newer github at <https://github.com/Kiriouchine/GPS-code-for-multirotor>, where a c file is presented and an elf file that runs well.

```
1 A standalone c file driver for a gps on the ardrone
```

0.4.2 arduino-serial-lib

The 'arduino-serial-lib.c' from <https://todbot.com/blog/2013/04/29/arduino-serial-updated/>, courtesy of Tod Kurt [3]. (this is an external library, it handles the options and opening of the ttyUSB0 port for reading. It is not necessary to use this specific library as the options and UNIX port opening commands can be written into the code manually, however it offers an organised way of handling the opening).

Note the Baud rate, which is given in the GPS C driver in the previous subsection.

`arduino-serial-lib.c` :

```
1 a serial library c code placeholder
```

`arduino-serial-lib.h` :

```
1 a serial library header placeholder
```

0.4.3 FTP via Matlab

The m-file of using FTP through Matlab to transfer the standalone driver to the AR Drone (alternatively WinSCP (<https://winscp.net/eng/download.php>) or another FTP program can be used), Note that the transferred file `gpsOpenC.elf` originated from the same folder as is the working folder in Matlab, named `ftp_matlab.m`:

```
1 % to compile a c file test.c into an elf file to run in linux on drone:
2 % use cmd, go to folder of test.c, and run the following command:
3 % arm-none-linux-gnueabi-gcc test.c -o testOutput.elf
4 % then testOutput.elf will show up
5
6 %to link libraries, such as , use -Wall cfile.c c:/pathtolibrary/libraryname.c between
7 %gcc and -o , for example :
8 % arm-none-linux-gnueabi-gcc -Wall gps-openC.c c:/Users/vsevolod/arduino-serial-lib.c -o gpsOpenC.
9   elf
10
11 % run the following in matlab to make connection with quad-copter via ftp
12 % (also possible through winSCP with ftp mode), (you have to be in the
13 % folder in which testOutput.elf is in when running the code below):
14
15 an ftp in matlab placeholder
16
17 % droneTcpip =
18
19 % in putty look up with 'ls' what is in there
20 % go to update folder with: cd update
21 % ls again to see testOutput.elf
22 % to make testOutput.elf possible to run, use the following command line in
23 % putty:
24 % chmod 777 testOutput.elf
25 % or
26 % chmod 777 gpsOpenC.elf
27 % then to run the elf file, give in putty:
28 % ./testOutput.elf
29 % to remove the file, for putting a new one on:
30 % rm testOutput.elf
```

0.4.4 GPS driver Matlab

The code for `GPS_Read.c`.

It uses a better searching algorithm for a more accurate string detection, as well as the 4800 baudrate .

```
1 A matlab compatible c file placeholder for the toolbox
```

The code for `GPS_Read.h`.

```
1 A matlab compatible c file header placeholder for the toolbox
```

0.4.5 GPS driver Matlab

The code for `Generate_AR_Drone_S_Functions.m`.

The new part is marked with %% GPS

```
1 placeholder for the m file
```

Bibliography

- [1] Vsevolod I. Kiriouchine "Sensing and control design for high speed autonomous agile manoeuvring with quad-rotors." Eindhoven, Eindhoven University of Technology, November 2017
- [2] ArDrone2Target Toolbox by; Daren Lee, Sanne Marx. <https://github.com/darenlee/SimulinkARDroneTarget> , <https://gitlab.com/ARDrone/ArDrone2Target>, 2014, 2017.
- [3] The arduino-serial-lib, courtesy of; Tod Kurt. <https://github.com/todbot/arduino-serial/> , <https://todbot.com/blog/2013/04/29/arduino-serial-updated/>, 2013, 2006.