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

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/sgpp/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.

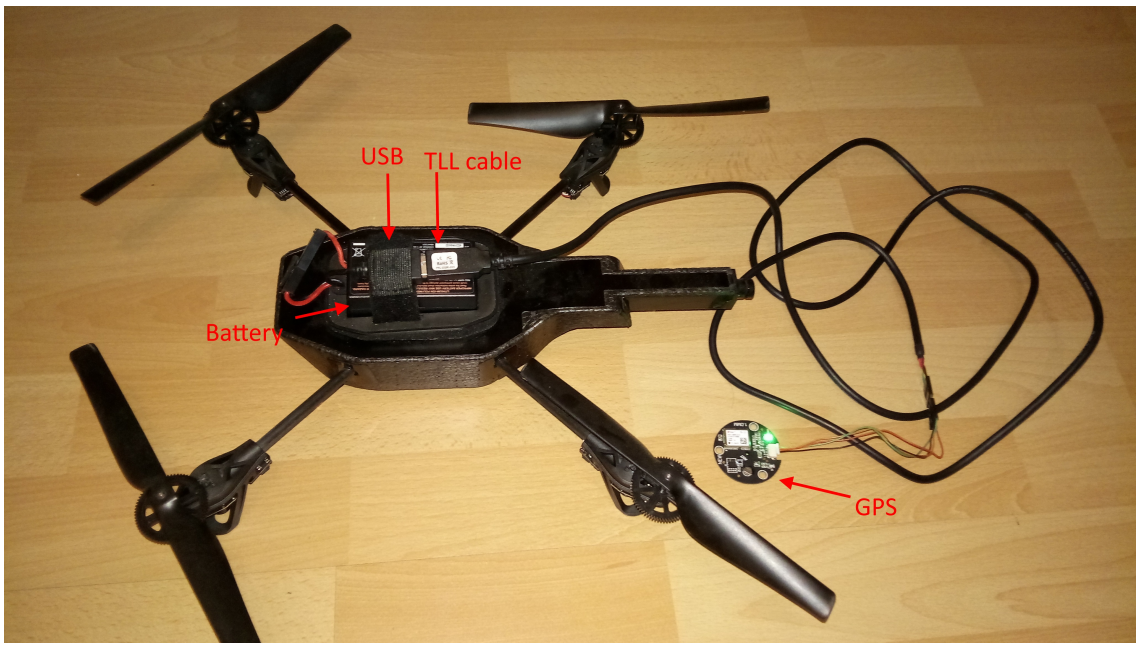

Figure 1: The configuration of the quad-copter and GPS module.

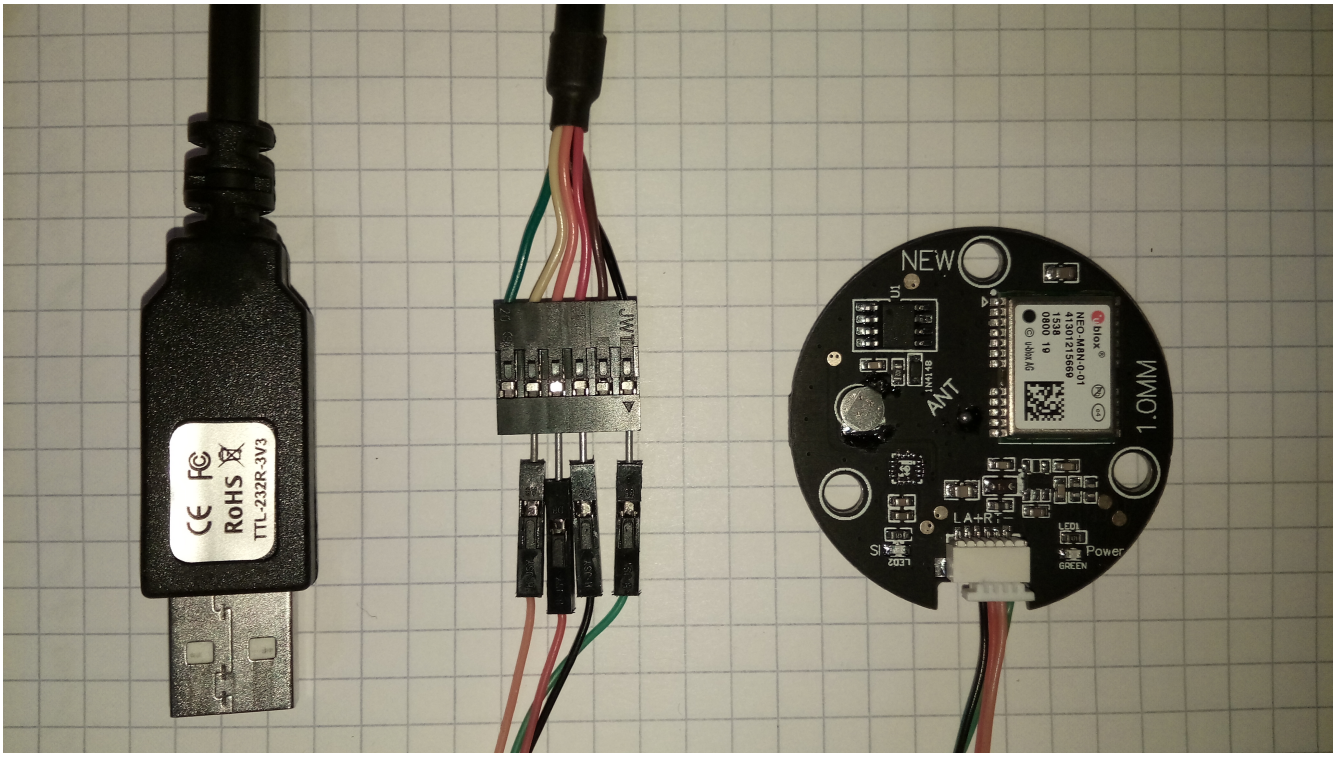

Figure 2: The used NEO-M8N-0-01 GPS module with the FTDI TTL 'ttl-232r-3v3' cable with the colour-coded transferring shown.

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/sgpp/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
```

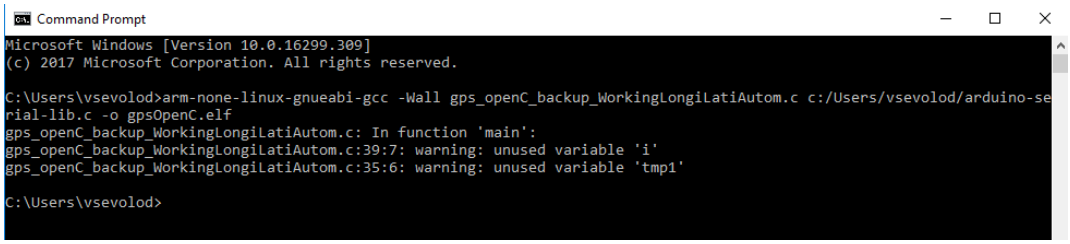

Figure 3: The cmd with the compiler code and the return

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_WorkingLongiLatiAutom.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.

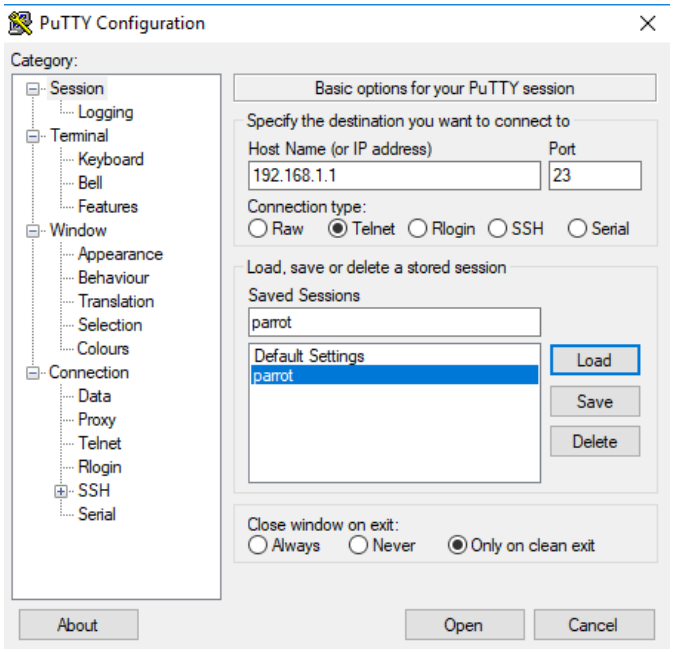

Figure 4: The PuTTY starting window

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.

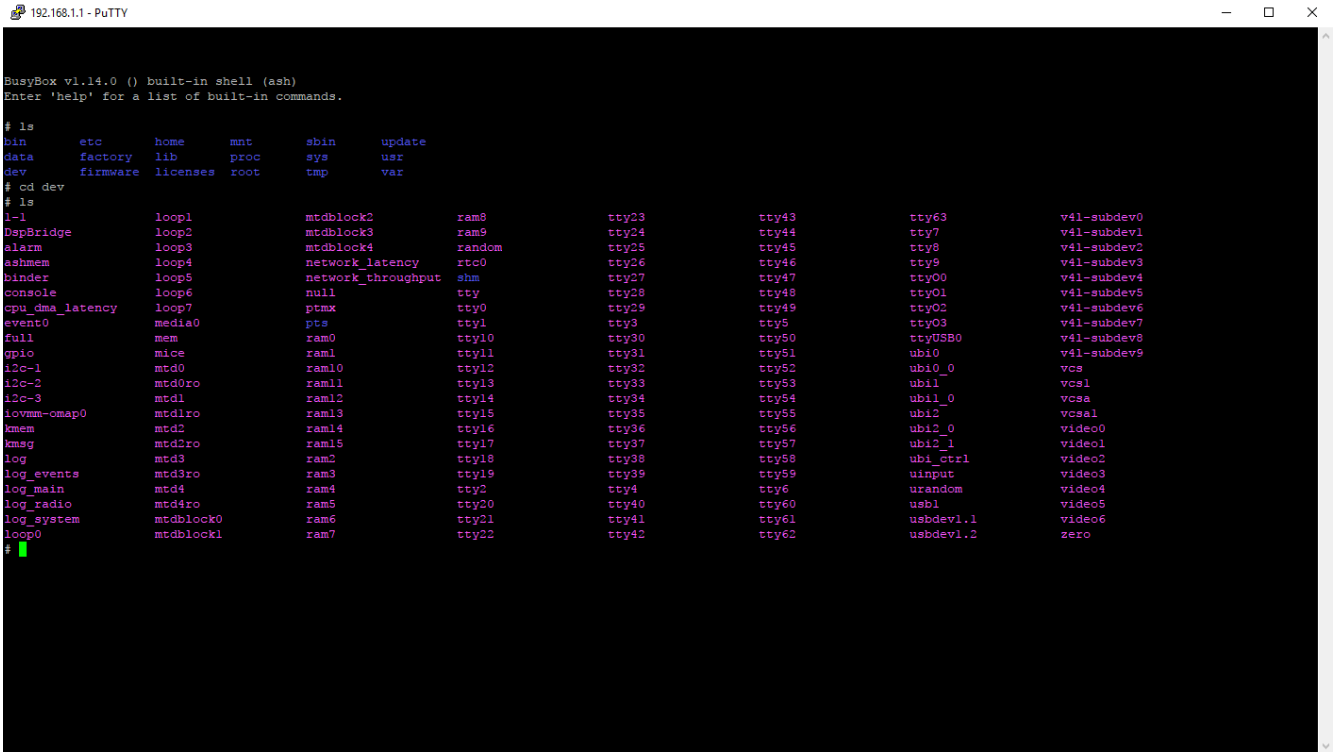

Figure 5: The PuTTY opened window

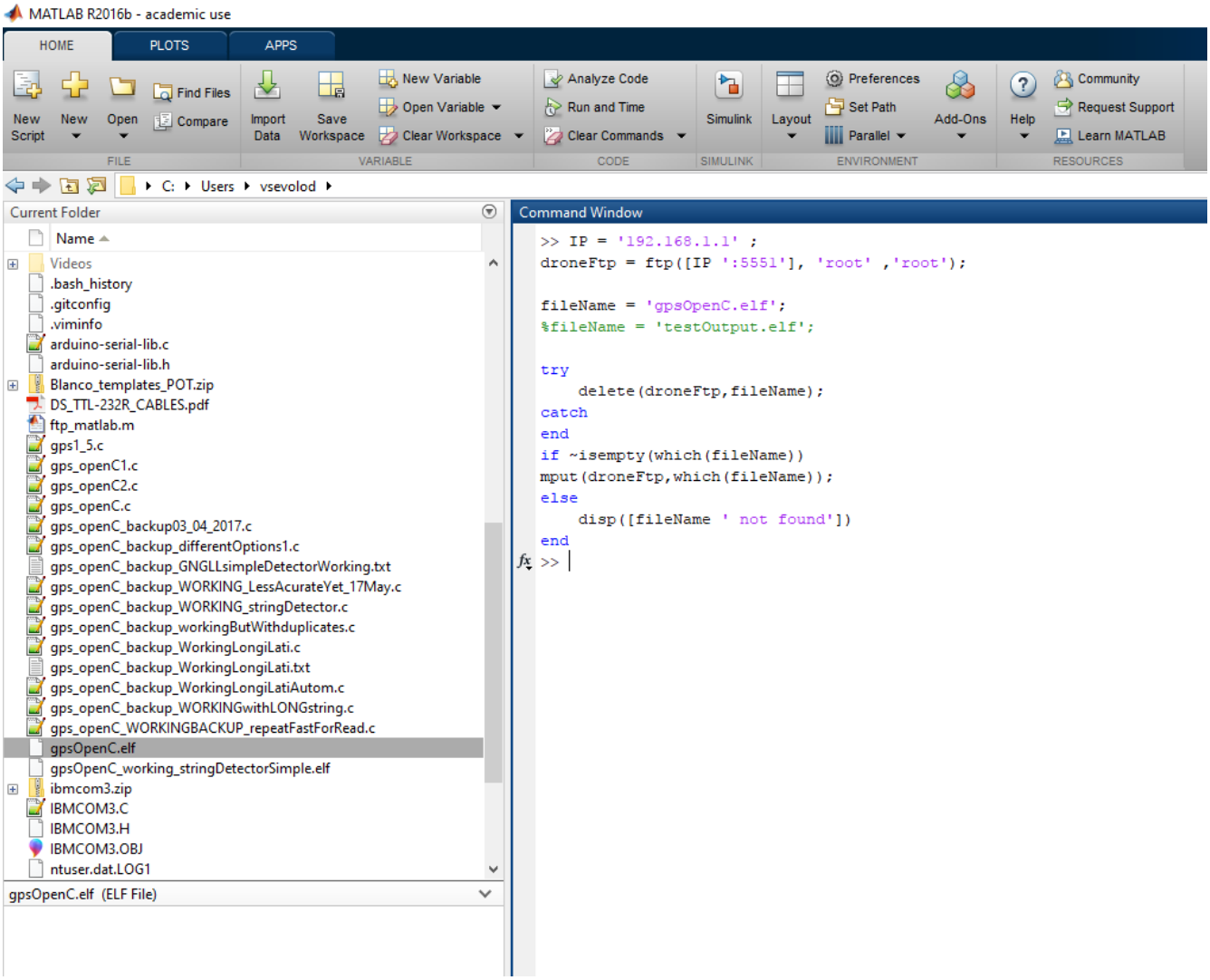

Figure 6: FTP via Matlab

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.

Figure 7: Green file

Next the file is run, using:

`./gpsOpenC.elf`

If used correctly the GPS gives the location:

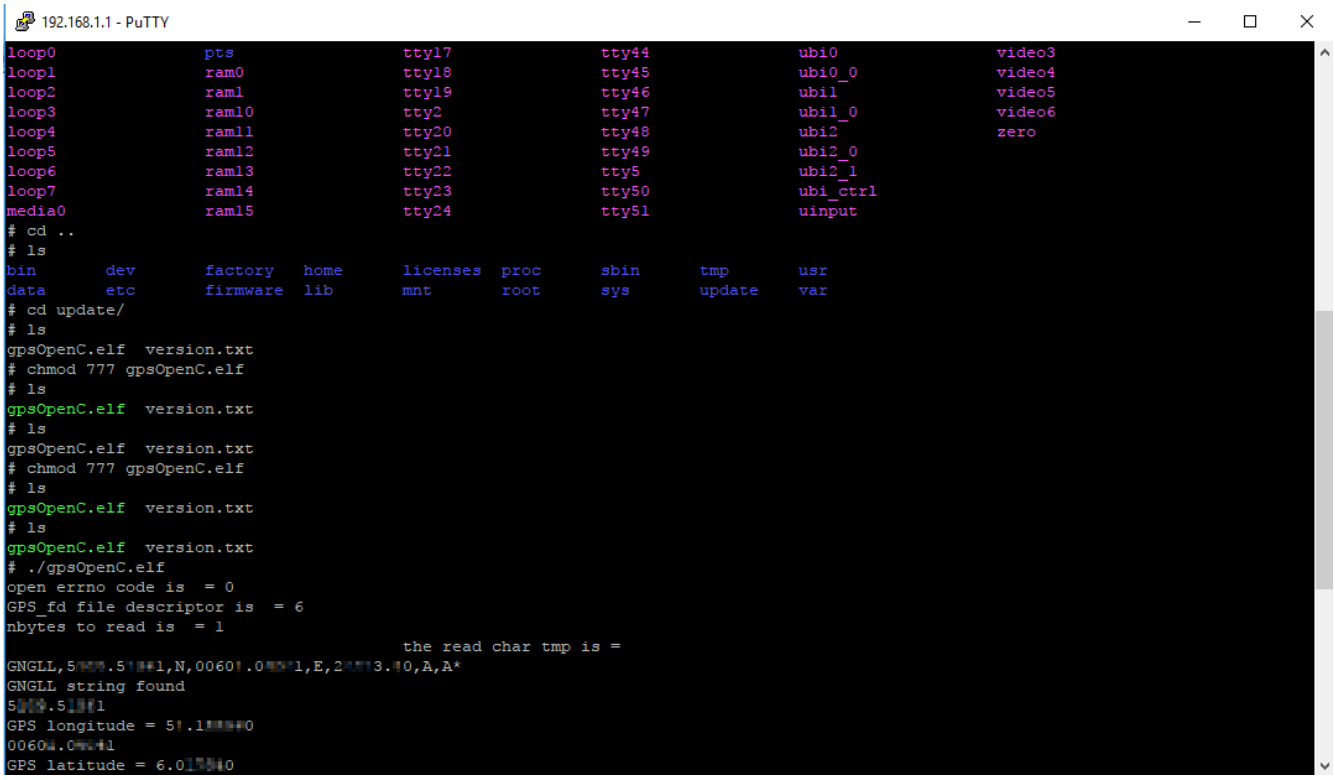

Figure 8: The GPS gives location

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 inforamtion needs to be send 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 usefull when used in Matlab with the filters described in [1].

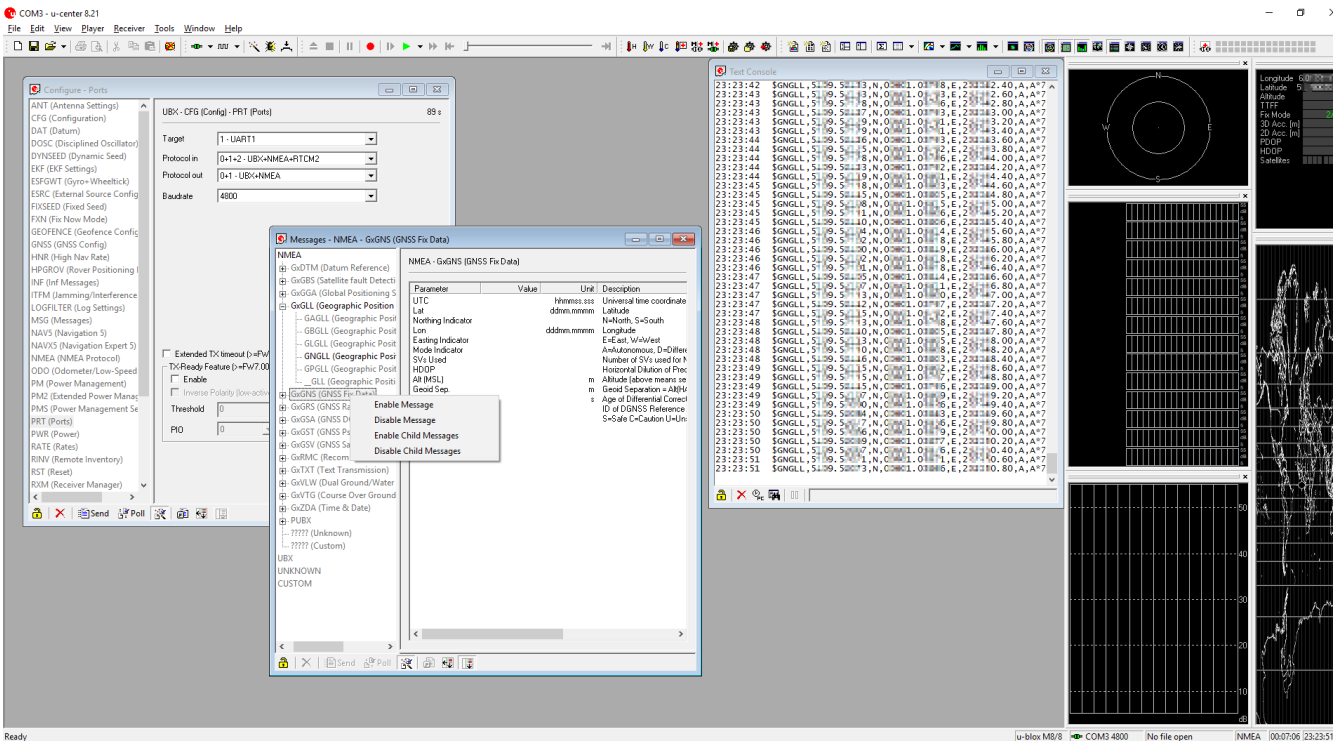

Figure 9: The GPS menu

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 Maltab 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

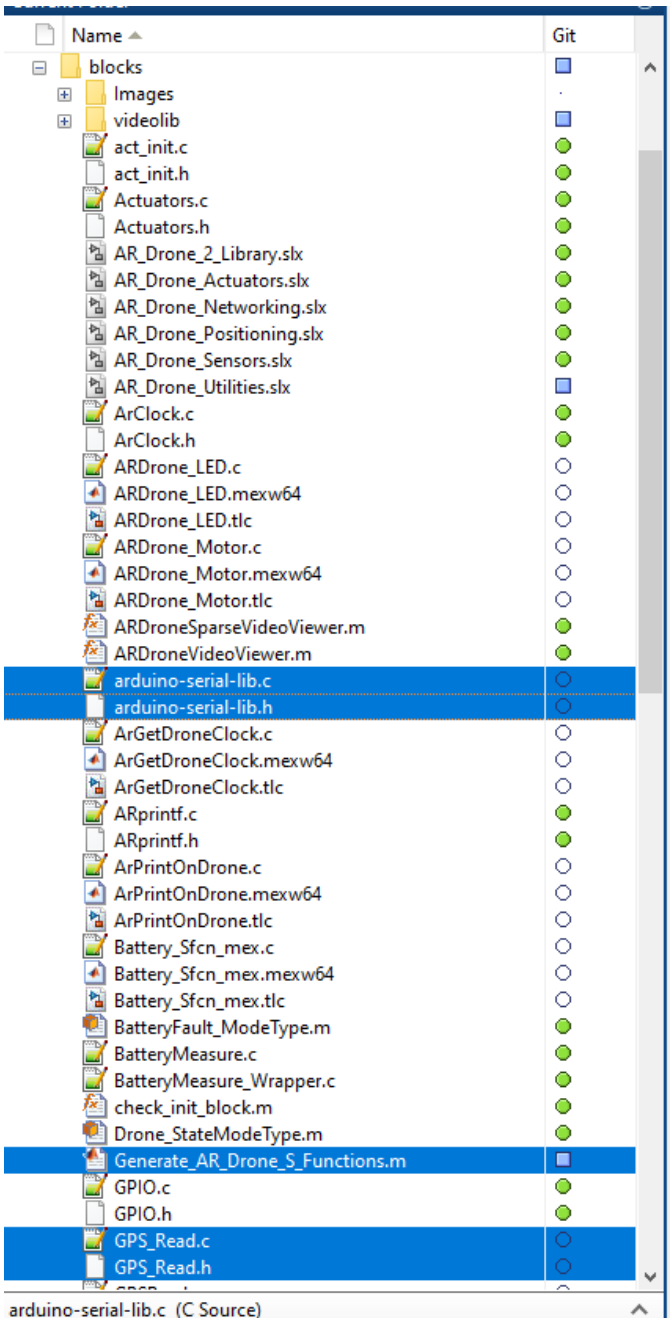

Figure 10: The files copied

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).

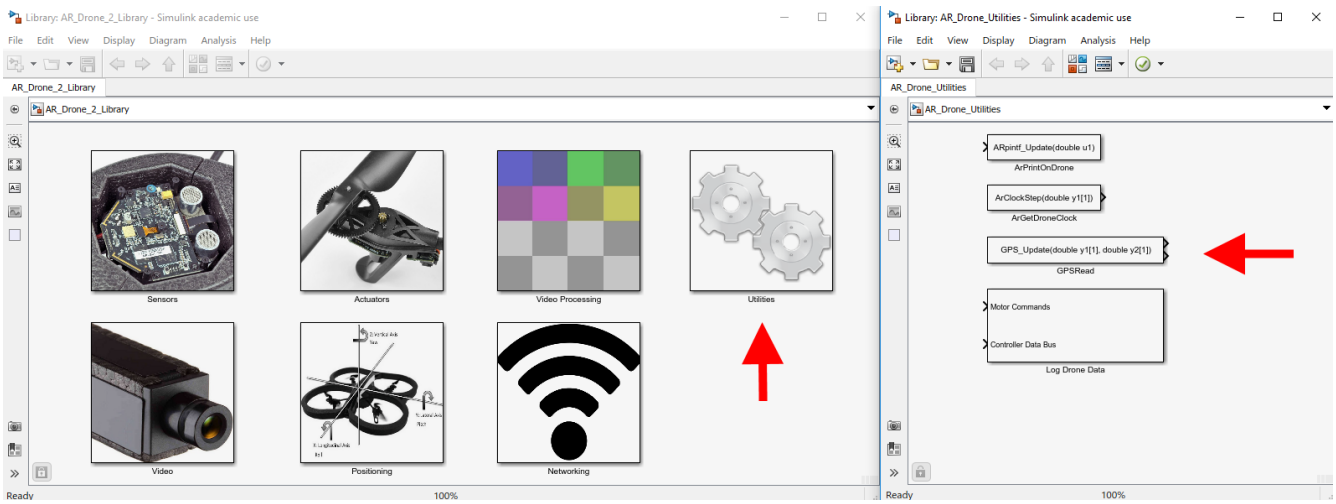

Figure 11: Utilities, you can click the lock button in the bottom left to move the blocks

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 sen in figure 12.

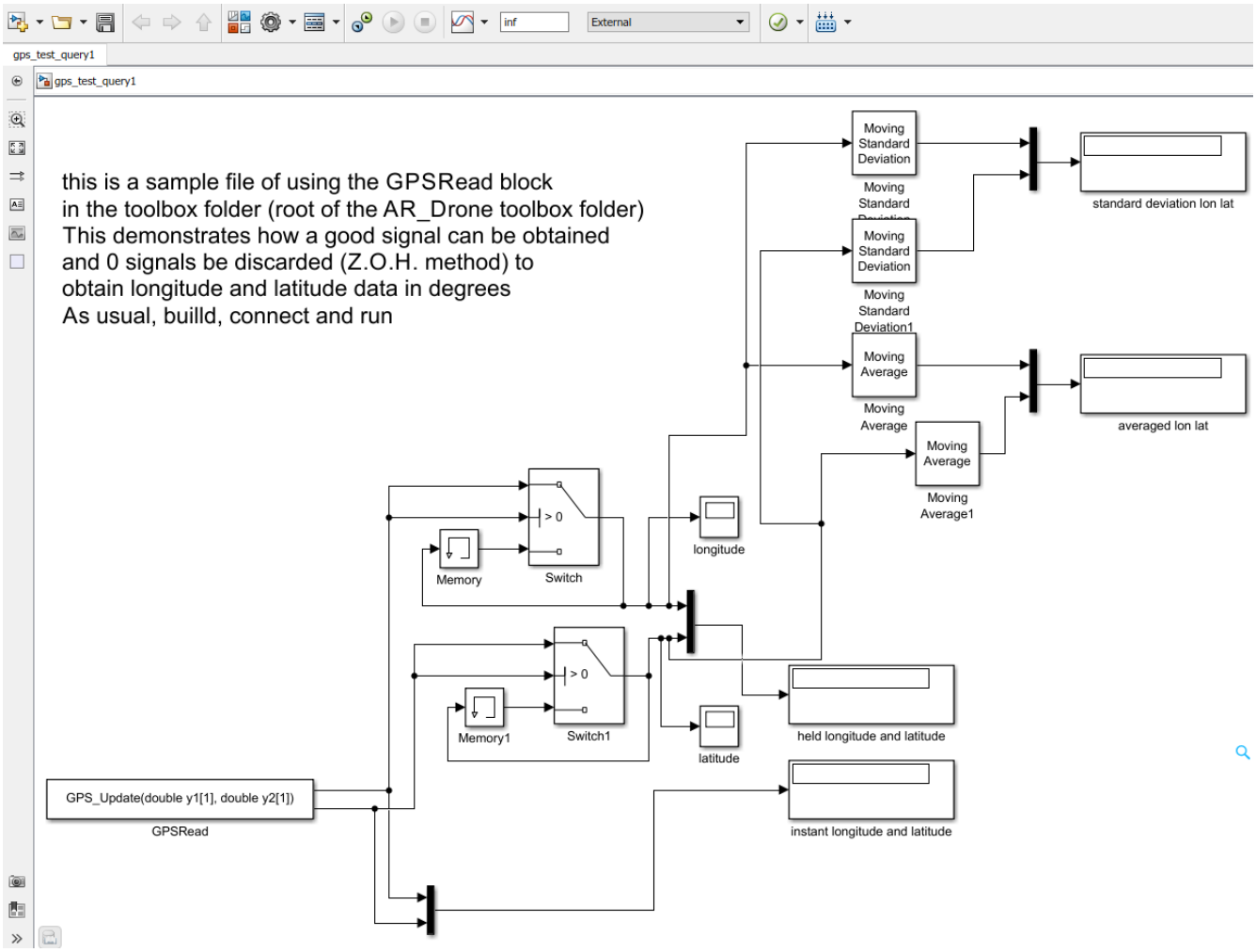

Figure 12: An example

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 htis 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 blanc 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.

```
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 :

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

arduino-serial-lib.h :

```
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`:

```
% to compile a c file test.c into an elf file to rn in linux on drone:
% use cmd, go to folfer of test.c, and run the following command:
% arm-none-linux-gnueabi-gcc test.c -o testOutput.elf
% then testOutput.elf will show up

%to link libraries, such as , use -Wall cfile.c c:/pathtolibrary/libraryname.c between
%gcc and -o , for example :
% arm-none-linux-gnueabi-gcc -Wall gps_openC.c c:/Users/vsevolod/arduino-serial-lib.c -o gpsOpenC.
    elf

% run the following in matlab to make connection with quad-copter via ftp
% (also possible through winSCP with ftp mode), (you  have to be in the
% folder in which testOutput.elf is in when running the code below):

an ftp in matlab placeholder

% droneTcpip =

% in putty look up with 'ls' what is in there
% go to update folder with: cd update
% ls again to see testOutput.elf
% to make testOutput.elf possible to run, use the following command line in
% putty:
% chmod 777 testOutput.elf
% or
% chmod 777 gpsOpenC.elf
% then to run the elf file, give in putty:
% ./testOutput.elf
% to remove the file, for putting a new one on:
% 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 .

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

The code for `GPS_Read.h`.

```
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

```
placeholder for the m file
```

Bibliography

[1] Vsevolod I. Kiriouchine "*Sensing and control design for high speed autonomous agile manoeuvring with quadrotors.*" 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.