WADGNSS Client User Guide

August 2020

Contents

Abstract

This document describes software that enables Wide Area Differential Global Navigation System (WADGNSS) for users on a global basis, without the user needing access to a local GNSS reference station. The software functions in a client-server architecture. Typical users will only use the client portion of the software, which will connect to a remote server. At the time of connection, the user supplies an IP address and a position $\mathbf{P}_b$ for a virtual base station that is within 20 km of where the user will operate. The server provides artificial measurements at the location $\mathbf{P}_b$ of the virtual reference station that include Observation Space Representation (OSR) corrections. Those measurements are distributed in RTCM format using message types ?? and ?? using the NTRIP protocol.

At present, the method works only for the GPS constellation. The theoretical approach behind the software extends to other GNSS systems. At present, the SSR data required to compute the OSR corrections is only available in real-time for GPS.

The software is released in open source format under the ???? license. The client software is available in source code and executable formats at

`https://github.com/jaffarrell/WADNGSS_Client_Server/Code/WADGNSS_Client`

The server software is available in source code and executable formats at

`https://github.com/jaffarrell/WADNGSS_Client_Server/Code/WADGNSS_Server`

Acknowledgements

This software was developed under funding from the California Department of Transportation under the project "Network Differential GNSS Corrections for Connected and Autonomous Vehicles." (Contract 65A0767).
The software utilizede BKG ???? .
The software utilized RTKLIB ???? .

1 Introduction

The DGNSS communicates measurements applicable to a virtual reference station in the vicinity of the user. The architecture of the system is illustrated in Figure 1. The client software is hosted on a user computer with a direct connection to the users receiver. The purpose of the client software is to establish and maintain the connection with the server. At start up, it communicates the users IP address and virtual base location $\mathbf{P}_b$. After that point, the client receives the virtual base station measurements from the Ethernet connection with the server and sends them to the user receiver over the local connection (e.g. serial port or USB).

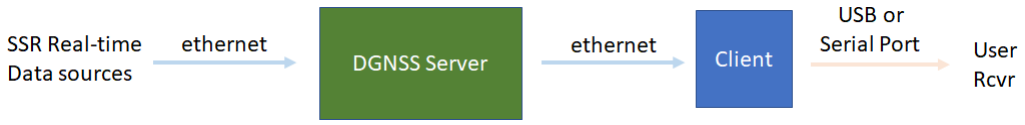

Figure 1: Client Server DGNSS Architecture.

2 Client

In the following sections we outline compilation, installation, GPS receiver setup, execution, and troubleshooting of the `WADGNSS_client` application. We assume most users will read the guide sequentially once and then use it only as a reference document.

We recommend that most users begin from the **Installation** subsection, which describes the procedure for installing a pre-compiled executable application. We also distribute the source code. Users who wish to start from the source code to compile the `WADGNSS_client` application should begin in the **Compilation** subsection.

2.1 Client Software Compilation

The following procedure outlines compilation on the Ubuntu 18.04 operating system. The procedure may be compatible with other Linux distributions, but is untested as we are actively developing on Ubuntu.

1. Download the necessary prerequisite software to compile `WADGNSS_Client` source. You will need the `cmake` build tool and a `C++` compiler such as `g++`. If not already installed on your local system, you can run the following commands to download the most recent releases.

   ```
   $ sudo apt update && sudo apt upgrade
   $ sudo apt clean && sudo apt auto-remove
   $ sudo apt install make cmake g++
   ```

2. Compile the client software on your local machine.

   (a) Navigate to the `WADGNSS_Client` root directory. The root directory contains a `CMakeLists.txt` file.

   (b) Execute the following to make a build directory and navigate into the newly created build directory.

   ```
   $ mkdir build && cd build
   ```

   (c) Compile the `WADGNSS_Client` by executing the following commands. The first command will configure the `cmake` build system and the second will invoke `make` to start the compilation process.

   ```
   $ cmake .. && make
   ```

   (d) If compilation is successful an executable named `WADGNSS_Client` will be visible in the build directory. The client software is now successfully compiled.

You can now proceed to Section 2.3.

2.2 Client Software Installation

This section describes the recommended installation procedure for the `WADGNSS_Client` executable. At present, we support installation on Ubuntu 18.04 (Ubuntu 18.04.5 LTS). However, we are working to provide support for other operating systems and are focusing our efforts on releasing a Windows compatible client soon.

1. In any directory on your machine create a new directory and navigate into the new directory with the following command.

   ```
   $ mkdir WADGNSS && cd WADGNSS
   ```

2. Copy the `WADGNSS_Client` executable into the `WADGNSS` directory. The `WADGNSS_Client` executable is now successfully installed.

You can now proceed to Section 2.3.

2.3 GPS Receiver Setup

Before running the `WADGNSS_Client` executable, the user must initialize and configure their receiver using its specific software or other interface. At present, `WADGNSS_Client` software supports automated configuration for u-Blox ZED-F9P and u-Blox M8P. All other receivers need to be configured manually.

The following example on manual configuration is for a u-blox receiver using U-center. Manual configuration of other receivers can be accomplished by configuring the same parameters using the device specific configuration software:

- Enable GPS.
- Turn off all non-GPS navigation systems.
- Enable NMEA protocol.
- Enable PUBX 00 message.[1]
- Enable message output through a receiver USB port.[2]
- Enable RTCM version 3 for input through a receiver USB port.[3]

An example for u-blox receivers using U-center software is shown below.

Step 1: Enable GPS and set GPS only.

Go to *View* → *Messages View* → *UBX* → *GNSS*. Select enable GPS and disable others. Then click "*Send*"[4]

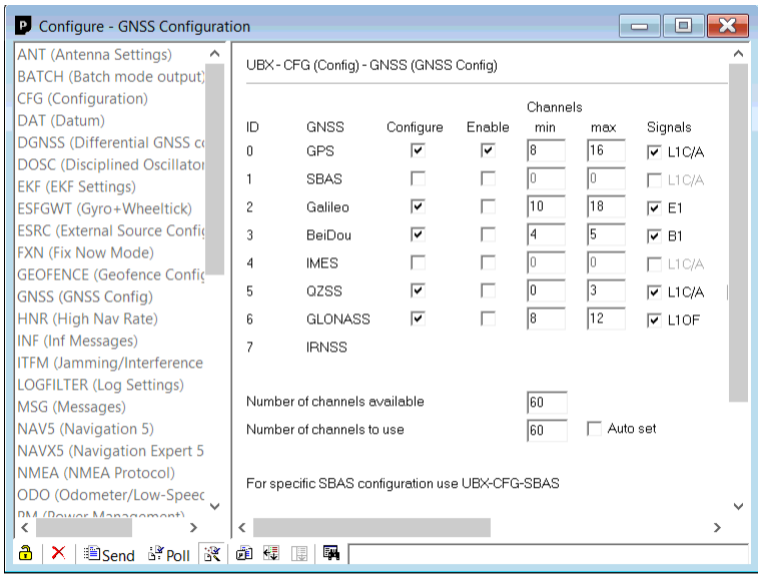

Figure 2: Step 1

Step 2: Enable PUBX 00 via NMEA protocol.

Go to *View* → *Messages View* → *NMEA* → *PUBX*. Right click "*00*" then select "*Enable Message*".

[1] TO DO: This sounds like a u-blox specific parameter. Please add a description of the goal of the step that would apply to other receivers.

[2] TO DO: Explain why. If their receiver does not have USB, what to they do? What is the big picture goal of the step

[3] TO DO: Explain why. If their receiver does not have USB ports, what to they do? What is the big picture goal of the step

[4] TO DO: Why are several of the non-GPS systems checked under 'Configure'? Does it matter which signals are checked?

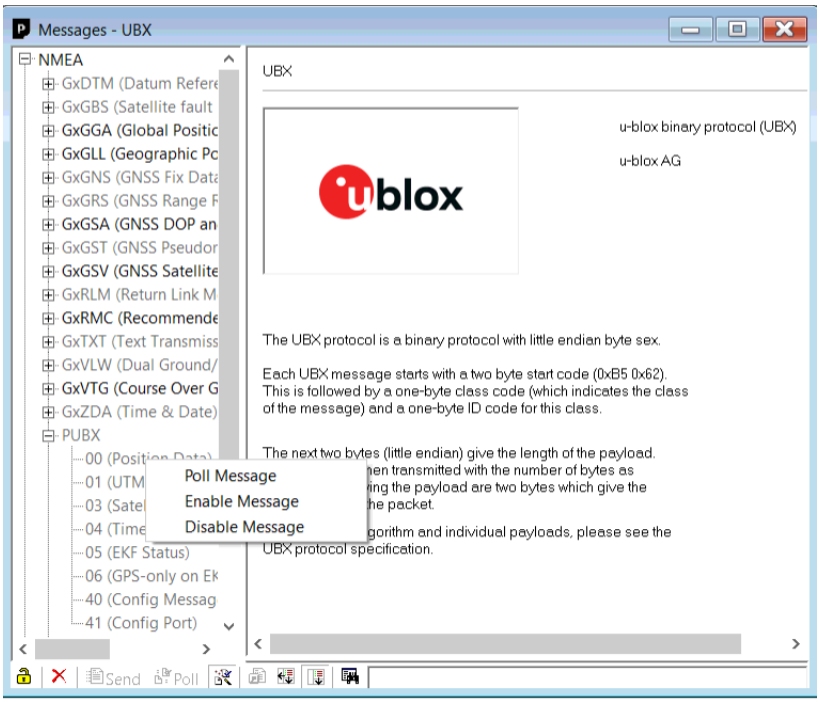

Figure 3: Step 2

Step 3: Enable a receiver USB port for RTCM version 3 input.
Go to *View* → *Messages View* → *UBX* → *CFG* → *CFG* → *PRT*. Select the options based on Fig. 4. Then click "*Send*".

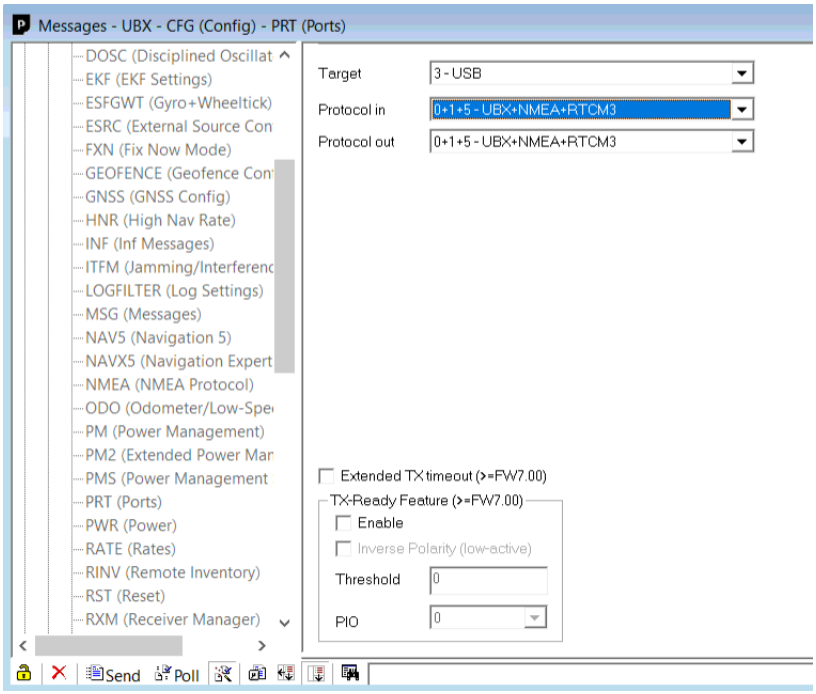

Figure 4: Step 3

If the receiver is turned off or its power is disconnected, the receiver should remain reconfigured the next time that it starts.

2.4 WADGNSS_Client Execution

The WADGNSS_Client executable expects four command line arguments: com port, configuration option, server ip and server port. An example usage template is shown below.


```
$ ./WADGNSS_Client [com port] [configuration] [server ip] [server port]
```

The following sections discuss these command line parameters and how to find the correct values for your system.

2.4.1 [com port] Paramater

The [com port] option designates on which of the computers USB communication ports the WADGNSS_Client software receive messages from the GPS receiver. Determining port number to which the GPS receiver is broadcasting is operating system specific. It may take a couple attempts to discover the correct filepath.

In Ubuntu, serial data from the GPS receiver is available at a filepath matching the pattern \dev\ttyACMX where X is a number in the range [0-9]. For example, data from our receiver might be available at \dev\ttyACM0 so we provide the filepath \dev\ttyACM0 in place of the [com port] argument.

In Windows, the com port can be determined by opening Device Manager and expanding Ports (COM & LPT) option. For example, suppose your GPS receiver is visible as device COM6. Then, the filepath COM6 is provided in place of the [com port] argument.

2.4.2 [configuration] Paramater

The [configuration] parameter indicates the desired configuration of the supported GPS receiver. If supported, the WADGNSS_Client will initialize the connected GPS receiver to the corresponding configuration. We currently support these automated configuration options:

- 0: Receiver configured by its software,
- 1: u-blox M8P module
- 2: u-blox ZED-F9P module

2.4.3 [server ip] and [server port] Parameters

The [server ip] and [server port] parameters indicate the network location of the WADGNSS_Server to which the WADGNSS_Client will connect. Determining the ip address and port of the desired server is outside the scope of this manual.[5]

2.4.4 Example Execution

Bringing all this together, an example execution command for the WADGNSS_Client where USB data is available at \dev\ttyACM0, the connected GNSS receiver is u-blox M8P, and the client will connect to a WADGNSS_Server at ip 63.45.125.1 on port 8772 is as follows:

```
$ WADGNSS_Client \dev\ttyACM0 1 63.45.125.1 8772
```

Congratulations you have successfully launched the WADGNSS_Client. If there are error messages, please see the section on troubleshooting[6]. To stop the WADGNSS_Client press the "ctrl + c" keys, simultaneously on the host computer.

[5] TO DO: We need to get this info and fill it in.
[6] TO DO: This does not exist. Are you working on it or should teh cross-reference be removed.

