AAS – MTRN4010

This document describes the format of the messages, used by the UGV (and by the playback module).
In order to avoid using a predefined system for interacting with the platform, we use standard UDP protocol for receiving sensors' data from the UGV (this is different to previous years, when an API was provided for that purpose). In this way, you would be able to receive those measurements in any usual OS (Windows, Mac, Linux, Arduino, etc.), without needing to install any software in your computer. In addition, although we perform the processing in Matlab, you may use it from other frameworks (C/C++, Python, etc.)
Your program (Matlab program) will receive UDP messages which will contain the sensors' measurements, in real time.
There can be two possible senders (or "data producers"). One is a UGV platform, whose on-board software is set to transmit certain measurements. An alternative sender can be a simulator program, which you can use for testing your program before you try it with the real platform. When used via that playback module (i.e. the simulator), you can configure that program for targeting a particular IP address and port number. Those parameters are defined in the configuration file of that playback program.
When we use the UGV (i.e. UGV is producing those messages), then the UDP port number will be fixed, so your client program (in Matlab) should allow to be configured (i.e. for listening a particular UDP port number).

Message structure

The content of all the messages, which you will receive via UPD protocol, is composed by a header whose format is defined by the following structure:

```
struct headerUDP4010
{       unsigned char HH,ID;
        unsigned short cx,sz,CS16 ;
};
//note: This structure and all the ones mentioned in this document are BYTE aligned.
```

The meaning and purpose of each of these fields is as follows,

HH : fixed = 0x33

ID : for indicating type of message, which in our case are:

ID=31: Dead-Reckoning (DR)(Longitudinal velocity and GyroZ). ID=26: LIDAR

cx: just a counter. (not relevant)

sz: length, in bytes, of the data part of the message

CS16 : Checksum of the full message, excluding HH and CS16.

This header (whose size is 8 bytes), is followed by a block of "sz" bytes.
We have, in 4010, two usual messages: LIDAR and DR (dead reckoning data)

The data associated to a LIDAR message has the following structure:

```
struct LIDAR_packet                 // laser packet
{       unsigned time;              // timestamp (32 bits unsigned, 1 count = 0.1ms)
        unsigned short cx ;         //(irrelevant for MTRN4010)
        unsigned short range[361];  // range & intensity data (as in projects 1,2)
        unsigned char irrelevant[8] ;  //(irrelevant for MTRN4010)
};      //size = 736 bytes
```

The data associated to a DR message has the following structure:

```
struct DR_packet
{       unsigned time ;          // timestamp (32 bits unsigned, 1 count = 0.1ms)
        short int V,W ;          //16 bits signed integers,..
                                 //.. for indicating longitudinal velocity (V) and angular rate (W).
} //size = 8 bytes
//scaling: V : 1 count = 1 mm/second ; W: 1 count =0.02 degree/second. ;
```

Note: all the measurements contain timestamp, based on the same clock.

Using the Playback module

Before you are able to test your programs by processing data generated by the UGV, in real time; you should test your program using the simulator. The simulator is a module which plays data back, emulating the real UGV. The simulator can send the data (real data, previously recorded in experiments), at the same rate in which the data (i.e. measurements) were originally generated. In fact, the same data you used (OFF-LINE) in previous projects, which was provided in Matlab files, can be played back, using this simulator.

Although the simulator program can be configured to play different datasets, and can be operated in different ways, you are provided with it already having a default configuration, ready to run (i.e. just running it).

The program is a CONSOLE mode program, which offers certain capabilities. For knowing those capabilities, just press ***h.***

The most useful of its commands are the following ones:

pausing playback	any of the following keys:	p, P, s, S
for continuing playing,	any of the following keys:	c, C
for ending the program,	any of the following keys:	e, E, q, Q
for jumping in time	press key:	j
for help	press key:	h

(there are some other less useful commend; just press 'h' for help in the console itself)

Settings

In the configuration file ("simu_Possum_ViaUDP.cfg"), which is a TEXT file (so you may use NOTEPAD or any other plain text editor, for editing it), you may need to modify certain settings. The ones which are of interest are the target UDP port and the target IP address. Read the comments in the file itself, for understanding those settings.

OS Compatibility

The simulator program (which is named "PossumSimulator08_UDP.exe") is a Windows' program (for XP, W7, W8, W10). If your computer does have a different OS you may try the following solutions:

1) Run the simulator from a Windows virtual machine (hosted in your computer/main OS), so that you can receive the messages in your main/host OS (where your client program does work)
2) Run the simulator in Windows, running in another computer, but targeting your computer.
3) Other combinations of VMs and real machines may also be feasible.

For avoiding misuse or promiscuous/aggressive use of the program, I have disabled its capacity of sending data outside the LAN (usual IPs will be limited to the family 192.168.XXX.ZZZ and loopback addresses). Broadcast addresses are also rejected by the program.

Receiving the sensors' data in your program

You will need to implement the reception of messages in your program. Matlab offers some basic API for TCP communications; one of these is for the UDP protocol.
Students are allowed to work in small teams (up to 3 members) for implementing this part of the program. The rest of the parts of your program are considered individual work.

Questions: Ask the Lecturer, via Moodle or via email (j.guivant@unsw.edu.au)