**Protocol: Telemetry Performance Assessment**

|  |  |
| --- | --- |
| Abel Corver  *December 2016* | [corvera@janelia.hhmi.org](mailto:corvera@janelia.hhmi.org) [abel.corver@gmail.com](mailto:abel.corver@gmail.com) |

**This protocol summarizes the steps involved in:**

* automatically assessing the performance of the Telemetry Transceiver hardware
* automatically generating performance plots and documents

**Step 1.** Attach the FlySim line to the FlySim system. Ideally, make sure the bead is on the *top* line, rather than the bottom. Then attach either a rigid or flexible backpack to the bead, making sure that the backpack’s hardware components are *pointing up*.

**Step 2.** All files used in the protocol can be found in the directory below. Navigate to this directory:

|  |
| --- |
| Z:/people/Abel/arena-automation/telemetry\_assessment |

**Step 3.** Make sure the FlySim Arduino is connected to the computer. Open the following Arduino file in the *Arduino IDE*, and upload it to the FlySim Arduino:

|  |
| --- |
| telemetry\_assessment.ino |

**Step 4.** Open a Serial Monitor to the FlySim Arduino. To confirm we’re connected to the right Arduino, type “h” in the command window. The monitor should display the response: “*FlySim Arduino Controller.*”

**Step 5.** Now type “z 300” (without quotes), followed by a newline. Then type “x -200” (without quotes), followed by a newline.

**Step 6.** Now enable the Transceiver’s RF power. Don’t enter the arena room or come near the antenna after this point!

**Step 7.** Now make sure that the Telemetry Transceiver hardware is plugged into the PC, and execute the Bug3 software:

|  |
| --- |
| bug3\_spikegl.exe |

In the Bug3 window, click on “*File 🡪 Log Performance Statistics*,” and select the name of the output file that will log the performance statistics.

**Step 8.** In the Bug3 window, press “Start” to start displaying the neural spike trains received from the backpack, which is still in the Z=300, X= - 200 position.

**Step 9.** Now tune the Amplitude and Phase of the Telemetry Transceiver, minimizing the logErrorRate and Missing Frames.

**Step 10.** Now make sure the Arduino IDE’s Serial Monitor is closed, thus making sure the serial connection to the FlySim Arduino interface is not taken. The best approach is to close the entire Arduino IDE.

**Step 11.** Run the following script:

|  |
| --- |
| telemetry\_assessment.bat |

This will establish a connection with the FlySim Arduino, send the command that will initiate the backpack’s slow motion throughout the X-Z plane, and log all backpack positions as a function of time.

The script will prompt for a location to store the position log in. The script will proceed once an appropriate file is chosen.

**Step 12**. Let the whole system run! This usually takes 1 to 2 hours, depending on settings.

**Step 13**. After the system has explored all available positions, the backpack will stop moving. Notice that no new positions will appear in the console window brought up by the command in *Step 11*.

**Step 14.** Repeat the recording for both the original and new telemetry device.

*Note: Currently the scripts are designed to compare two Transceiver devices. Functionality to simply plot performance for a single device will be added soon. One temporary solution would be to duplicate the data files, to fool the scripts below into producing graphs without errors.*

**Step 15.** We now have 2 data files per Telemetry Transceiver device. A log file containing the position of the backpack at various moments in time, and a log file containing telemetry performance measurements. Put all these files in the “*data*” folder, and rename them as follows:

* new\_transceiver\_rigid\_backpack.arduino.position.log
* new\_transceiver\_rigid\_backpack.telemetry.log
* old\_transceiver\_rigid\_backpack.arduino.position.log
* old\_transceiver\_rigid\_backpack.telemetry.log

**Step 16.** Now run the following file:

|  |
| --- |
| process\_performance.bat |

This will take a while to complete. All output files will be created in the same “*data”* directory. In particular, the following file will contain a report of all performance statistics:

|  |
| --- |
| telemetry\_performance\_[date\_of\_recording time\_of\_recording]{.docx, .pdf} |

In addition, several image files (.png’s) will have been created containing the performance graphs.

*Note:* The Python scripts require Python 3.5, in addition to various additional Python libraries. Although most Python code is platform independent, a few libraries currently make this code Windows-only.

*Email Abel Corver for additional questions.*