Recording Sensor Offsets

June 14, 2015

The accelerometers and gyros in UDB4 and UDB5 are stable enough so that you do not have to hold your aircraft level and motionless every time you power up. Instead, if you are using a version of MatrixPilot that supports defining the sensor offsets at compile time, it is possible to carefully record the offsets once and for all for each of your boards. To find out if you are using a version of MatrixPilot with this feature, do a search in the project for CUSTOM_OFFSETS. If it is found in libUDB.c, you will see something similar to the following, and the version you are using supports the option for recording sensor offsets:

```
C:\Users\bill\Desktop\MatrixPilot_wjp_helicalTurns_working\libUDB\libUD
B.c:145: #ifdef CUSTOM_OFFSETS
C:\Users\bill\Desktop\MatrixPilot_wjp_helicalTurns_working\libUDB\libUD
B.c:161: #endif   // CUSTOM_OFFSETS
Search complete. 2 matches found.
```

As of this writing, the only versions of MatrixPilot that support this feature are the branch MatrixPilot_wjp_helicalTurns and the tag MatrixPilot_helicalTurns_Beta. It is expected that the feature will be soon ported to MatrixPilot trunk.

For each board that you want to setup this way, there is a two step process. First you will need to run the roll pitch yaw demo with an option to report the accelerometer and gyro signals and record the information using OpenLog. From the reported data, you can determine the offsets. The second step is to specify the offsets in your options.h file. You can maintain a separate file for each board that that you use, or you can put the values for all boards in one options.h file, and comment out the ones that you are not using when you program a particular board.

The first step is to run the roll pitch yaw demo with the following option set in the file main.c:

```
#define RECORD_OFFSETS    ( 1 )
```

Most likely that option is already set to one, but if it is not, set it. Program your UDB4 or UDB5 to run the roll pitch yaw demo. Then connect an OpenLog to it set to capture data at 19200 baud, place the UDB on a flat level surface, power it up, and leave it motionless. Let it run for about one minute, and then power it down. If you happen to jiggle it a bit at the start or the end, you can edit that data out of the file. Here are a few lines of a typical set of data:

```
8314, 306, 41, 7363, -17, -69, -36
8314, 311, 51, 7357, -15, -70, -33
8314, 287, 44, 7361, -13, -66, -38
```

The first element in each line is two times the computed value of gravity in UDB units, furnished in the file as a reference. It is the same value in every line. The next three elements are raw accelerometer signals for X, Y, and Z, based on two times gravity. The next three elements are raw gyro signals.

There are three variations of a method to determine the sensor offsets from the data. The first variation is to pick one of the lines that looks consistent with the others, and compute the offsets from it. The second variation, is to average over all the data taken while the board is level and upright. The third variation is to take data for both normal and inverted orientation, and take the average.

For the first variation suppose we take the first line:

8314, 306, 41, 7363, -17, -69, -36

The first value, 8314, is the nominal value of two times gravity. The fourth value, 7363, is the Z axis accelerometer value, so it includes 8314. To find the Z axis offset, we subtract 8314 from 7363 to get -951. So the sensor offsets are:

306, 41, -951, -17, -69, -36

These are the values that will be put into the options.h file when compiling MatrixPilot.

The second method is similar to the first method, except first compute average values, which is easy to do with a spreadsheet program.

Both the first and the second method assume the accelerometer calibration matches the assumed value of gravity. If it does not, there will be a slight error in the estimate of the Z axis accelerometer offset. A slightly more accurate method in that case is the third variation of the method in which you collect data both normal and inverted. You will have to figure out a way to hold the board level for inverted measurements. The author made a simple board mount using a plastic wall switch plate.

The following is typical data for inverted orientation:

8314, 270, -68, -9377, -18, -70, -38
8314, 266, -98, -9406, -13, -70, -37
8314, 281, -71, -9412, -12, -68, -42

If you are using a spreadsheet program, a convenient way to average the normal and inverted data is to first place the normal and inverted data side by side and delete a few rows from one side or the other to produce the same number of data rows of each type. Then copy the inverted data to below the normal data, and take the average for each column, which will produce the offsets without having to use the computed value of two times gravity. In that case, using the full set of data (not shown here), the offsets were computed as:

278, -17, -1018, -14, -68, -37

Once you have the offsets, the next step is to place them in your options.h file for MatrixPilot, and rebuild MatrixPilot project for your board. For this example, the following appears in options.h:

```
#define CUSTOM_OFFSETS
#define XACCEL_OFFSET  ( 278 )
#define YACCEL_OFFSET  ( -17 )
#define ZACCEL_OFFSET  ( -1018 )
#define XRATE_OFFSET   ( -14 )
#define YRATE_OFFSET   ( -68 )
#define ZRATE_OFFSET   ( -37 )
```

With the custom offsets set in your options.h file, you will no longer have to hold your plane level and motionless during power up. That said, it is still a good idea to have your plane closer to right side up than inverted during power up, because the software assumes your plane is level during the initialization of the direction cosine matrix. If it is not, it will take a few seconds for the software to re-orient the matrix values to the actual orientation.

Also, you will still have to have your Tx sticks in their trim position during initialization, because the software needs to record the trim values.