GPS – Sensor mounting

* Just connect the black socket to the pin.
* Black mark on the wire should be connected to the 1st pin.

GPS – Function

* Include the four QH\_GPS\_\* files into the project.
* Initialize the sensor by adding “gpsInit()” in main.c
* There are several important functions integrated in source codes.
* Include “QH\_GPS\_app.h”
  + unit8 getCurrentGeogrPos(GeograficPosition \* m)
  + bool getCurrentTime(GPSTime \* m)
  + bool getCurrentDate(GPSDate \* m)
* Include “QH\_GPS\_navicontrol.h”
  + uint32 calcDistance( GeograficPosition pos1, GeograficPosition pos2);
* Include “GPSControl.h”
  + void TestGPS(void)
    - This function is used for testing the GPS in our semester. It updates the current position (Latitude, Longitude, altitude and time, with and without filter) to copter states (included in copter.h) as global variables and could be used for control algorithm at any time.
* If more data from GPS-sensor is needed, use following steps to access new data:

1. Check in data sheet (*ls20030-3\_datasheet\_v10*), in which protocol the data is included. (If it is included in RMC or GGA protocol, go to step 8)
2. Then check in data sheet (*EB-230-Data-Sheet-V1.2*) page 9, Type 314, and find out the parameter to set the corresponding protocol’s output frequency.
3. In “void setGPSOutputFreq()” under “QH\_GPS\_app.c”, change the corresponding parameter.
4. Define new GPS message type in “QH\_GPS\_app.h”. (Use RMCmsg as template)
5. Add new static variable of the new GPS message type to “QH\_GPS\_app.c” to store the GPS data.
6. Define a new function, e.g. processGLLmessage() in “QH\_GPS\_app.c”. (use the current existing function processRMCmessage() as template)
7. Put in this new function into “void process\_message(void)” in “QH\_GPS\_app.c”.
8. If data is included in RMC or GGA protocol, just change the processRMCmessage() or processGGAmessage(). (Update the GPS message type, RMCmsg or GGAmsg, if necessary)

* Tips: There are a few useful functions defined in “QH\_GPS\_util.h” which could be helpful in getting new data.

GPS – Simulation model (GPSModel)

* This model simulates the noise of GPS signal.
* There are three files: GPS.mat, GPS\_database.m, and GPSModel,mdl. All should be included in same working directory in Matlab.
* You can copy the GPSModel to be used in any model, but remember to copy the GPS.mat and GPS\_database.m into the same directory and run the GPS\_database.m before you use it.
* To add new signal into the model:

1. Get the GPS-signal from xBee. (you should know which reading rate the signals are recorded)
2. First load GPS.mat by double clicking on it. And delete the corresponding old data. (e.g. you are adding new latitude, then you should only delete the old latitude from workspace.)
3. Import the data into Matlab by right-click on the signal file and choose “Import Data…”.
4. Click “Next” and check only the “Data” (uncheck the other two. You can rename the variable here or afterwards in workspace). The import signal should be an array.
5. Rename the signal correspondingly. (either Longitude, Latitude or Height)
6. Then type in “save GPS Latitude Longitude Height” in command window.
7. Update the data\_rate\_\* in GPS\_database.m and run it.
8. Now the new signal should be shown in the lookup table in GPSModel.

GPS – AutoCruiseControl

* This model simulates the auto cruising mode of copter. Further details see….
* You can turn on the auto cruising mode by switching Switch 4.
* You can input destination in the “Constant” Block (x,y,z) under “Set Coordination”.
* Remark:
  + This mode is not stable yet. You should only run it at the start of simulation. You should not switch to this mode after flying in other modes, it will results in instability.