|  |  |  |
| --- | --- | --- |
| RD/1 | AHNS-AP-DD-003 | QUAV Project, 2009, Autonomous Helicopter Navigation System, Autopilot Software, Design Document for |
| RD/2 | Embedded Linux Primer: A Practical Real-World Approach | Hallinan, C. 2007. Embedded Linux Primer: A Practical Real-World Approach: Prentice Hall PTR Upper Saddle River, NJ, USA. |
| RD/3 | OpenEmbedded | OpenEmbedded. 2009. Welcome to OpenEmbedded. Available: <http://wiki.openembedded.net/index.php/Main_Page> (access March 10 2010) |
| RD/4 | GNU Pth - The GNU Portable Threads | Engelschall, R. S. 2010. GNU Pth - The GNU Portable Threads. Available: <http://www.gnu.org/software/pth/> (accessed March 10 2010) |
| RD/5 | Native POSIX Threads Library (NPTL) Support for uClibc | Hill, S. J. 2006. "Native POSIX Threads Library (NPTL) Support for uClibc." Available: <http://www.kernel.org/doc/ols/2006/ols2006v1-pages-409-420.pdf> (accessed March 10 2010) |

# Software Architecture

The software architecture (Figure 3.1) consists of threaded airborne and ground based segments. The airborne segment is almost exclusively to be run on the flight computer whilst all ground control station code is to run on the GCS computer.



Figure . - High Level Block Diagram of Proposed Software Architecture

## Ground Control Station Software

A ground control station GUI and HMI GUI was designed in 2009 for AHNS using the Qt C++ library. In operation these two GUIs were run on separate computes linked by UDP [RD/1]. The lack of HMI system requirements removes the need to develop a separate HMI for a non-expert users and the move to a more powerful on board flight computer removes the need for control and state estimation to take place on the ground. Whilst it necessary to heavily modify the code-base to remove the UDP, control and state estimation features it is desired to continue development of the autopilot GUI developed.

The new proposed architecture still calls for a C++ Qt thread based architecture to run on a Ubuntu Linux desktop. The tasks preformed by the two GCS threads and their information sharing flow shows that very little processing other than logging, display and user input needs to be achieved through the GCS. Indeed the tasks for the GCS are well defined in SR-B-8 and SR-B-09.

## Airborne Software

The flight computer is an embedded device which is required to interface with the hardware sensors, actuators and communications. The requirement for considerable hardware interfacing has lead to the decision to program all embedded code with the C programming language. The overhead of object-oriented code is unwarranted thus an object-oriented programming language is not required. The importance of C is also clear when considering that the Linux kernel may have to be rebuilt for specific hardware configurations [RD/2].

The airborne software operating system will be developed on the flight computer with the aid of the OpenEmbedded Linux build environment [RD/3]. There is extensive support available for its use with the Overo Fire since it is recommended by the manufacturer. OpenEmbedded provides many pre-compiled programs and utilities. Aside from the Linux operating system, the airborne autopilot software has been formulated as a single process but its tasks have been divided into three logical threads; state estimation, control and downlink. The roles of each thread are shown in . It is anticipated that the state estimation and control threads will have the highest priority, approaching the requirement for soft real-time scheduling. The downlink thread will be scheduled at a considerably lower priority.

Implementation of the threads has been chosen to conform to the POSIX standard to guarantee functionality under the Linux operating system. The thread library desired and most widely distributed is the Native POSIX Threads Library (NPTL) [RD/2]. It is preferable because of its kernel creation of the threads, thus enabling pre-emptive scheduling which will switch between threads without waiting for events [RD/4]. The thread library to be used will need to be finalised based on the C library chosen as the base of the distribution.

The choice of C –libraries is essentially between GNU C (glibc) and micro C (uclibc) [RD/3]. From an embedded system perspective uclibc is the standard and hence preferable and does offer support for the desired POSIX NTPL thread library [RD/3;RD/5]. The C library is chosen at the time of operating system deployment from OpenEmbedded. The feasibility of using uclibc, NTPL and OpenEmbedded will become apparent early in the process of airborne code development nevertheless a threaded airborne software architecture is proposed.