DEPENDABLE SOFTWARE FOR THE BEESAT PICO SATELLITE

Sergio Montenegro*, Klaus Briess**, Hakan Kayal**

*FhG FIRST
Kekulestr 7
12489 Berlin
www.first.fhg.de/~sergio
sergio@first.fhg.de

**Technical University Berlin
Institute of Aeronautics and Astronautics
Marchstr 12
10587 Berlin
Klaus.briess@ilr.tu-berlin.de
Hakan.Kayal@TU-Berlin.de

1. ABSTRACT

BEESat is a picosatellite project (CubeSat standard) under development at the Institute of Aeronautics and Astronautics of TUB (Technical University of Berlin), Berlin, Germany, in cooperation with AstroFein technik (Berlin) and the Fraunhofer institute FIRST . The overall objective is to demonstrate and verify new component technologies for picosatellites, in particular the qualification of microwheels in orbit - an enabling actuation device to improve the attitude control capabilities to a new functional level in the very restricted confines of a CubeSat.

Although there are currently many developers of CUBESAT's, many of them lack of a precise attitude control system. This in turn is mandatory for more sophisticated applications, which require pointing capability. Having a precise attitude control system opens the door to new potential and very cost effective applications, which includes areas such as earth observation, space science, astronomy and on-orbit verification of new technologies.

The combination of highly sophisticated miniaturised technologies and the advantage of building swarms or clusters of pico satellites will result in completely new applications at very low cost. Beesat will demonstrate the use of coin size micro wheels for the attitude control of pico satellites in orbit as one of the key elements on which TU-Berlin is working. Further missions with a focus on other key technologies are planed for the future.

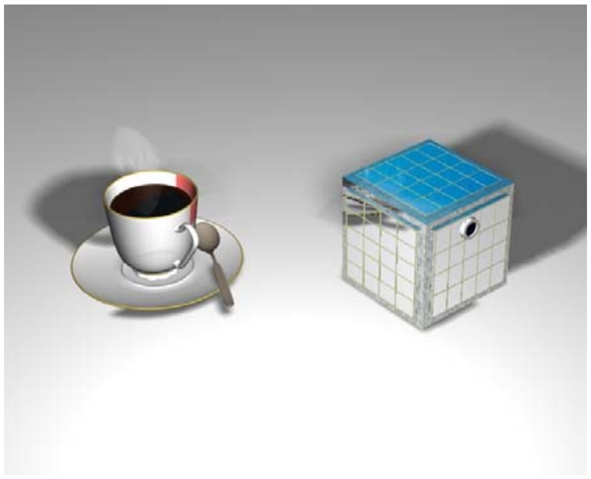

Figure 1: BEESAT

The main design requirements of BEESat can be summarized as follows:
- LEO orbit in the range 450 - 850 km
- Lifetime of 1 year

- 3-axis stabilization
- RF communications in UHF band
- Spacecraft operations at TUB

The launch of BEESat is planned for the year 2007.

2. Spacecraft:

BEESat conforms to the CubeSat standard having a size of 10 cm x 10 cm x 10 cm, and a mass limit of 1 kg, permitting a launch with the P-POD deployment mechanism of CalPoly. The spacecraft is 3-axis stabilized utilizing "microwheels" designed and developed by TUB in cooperation with Astro- und Feinwerktechnik Adlershof GmbH (Astrofein).

Each microwheel (See figure 2) is having an angular momentum of at least 2 x 10-4 Nms with a minimum torque of 3 x 10-6 Nm; the size of a single device is 20 mm in diameter and 14 mm in height. Three or four of the microwheels are suitable for an actuator system of a picosatellite providing a 3-axis stabilization capability. The microwheels are being controlled with a common electronics board which is connected to the spacecraft bus via a CAN (Controller Area Network) interface.

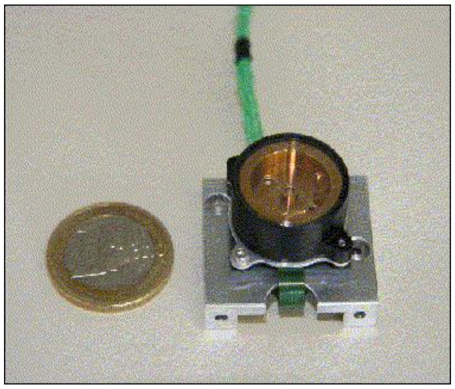

Figure 2: microwheel compared with a 1 EURO coin

The BEESat ADCS (Attitude Determination and Control Subsystem) utilizes 3 microwheels for 3-axis stabilization. In addition, a magnetic coil system is being used to desaturate the wheels and to control the satellite during orbital periods when the wheels might not be available. Attitude sensing is provided by a sun sensor system based on 6 PSD (Position-Sensitive Detector) photocells of Hamamatsu, and a magnetometer of Honeywell (HMC1023). The ADCS software supports 4 main attitude modes: a) inertial pointing, b) maximum power, c) Earth pointing, and d) rotation mode (as well as intermediate modes).

The OBDH (Onboard Data Handling Subsystem) utilizes a microcontroller (Philips LPC2292), a flash memory for data storage, and a redundant CAN bus interface for the communication with the microwheels. The OBDH employs the TinyBOSS operating system of FIRST (Fraunhofer Institute of Computer Architecture and Software Technology), Berlin. TinyBOSS is an adapted version of BOSS, a real-time operating system being successfully flown on the BIRD (Bi-Spectral Infrared Detection) microsatellite mission of DLR. Note: the acronym BOSS stood originally for "BIRD Operating System (Simple)" - it is now being used for many other applications, not only for those in space.

Power is being provided by surface-mounted solar cells. Four Li-ion battery cells are being used for spacecraft operations during the ecliptic phase of the orbit. The power bus provides a voltage of 3.3 V and 5 V to the S/C subsystems, utilizing DC/DC converters.

3. THE OPERATING SYSTEM BOSS AND ITS MIDDLEWARE

BOSS (from FhG FIRST) is a real-time embedded operating system and middleware, which were designed for safety and simplicity and to allow their own mathematically formal verification. Complexity is the root of most development errors – if you eliminate complexity, you eliminate most development errors. This was one target of BOSS. The BOSS-middleware simplicity allows the system to be easily understood, used and ported to other platforms even to FPGA (Field Programmable Gate Array) logic. The BOSS middleware has already been implemented and is being used in software for real-time dependable systems. We now aim to implement the same middleware in FPGA hardware. This will bring hardware and software developers together in an environment that is familiar to both. It makes no difference whether applications running on top of the BOSS middleware are implemented in software or hardware. For communication purposes, it makes no difference whether the communication partner is implemented in hardware, software or both. The BOSS middleware allows any combination of communication (SW/SW, SW/HW, HW/SW, HW/HW) and creates a (Flying Laptop) satellite standard interface between software and hardware without needing different device drivers for different devices.

3.1. The BOSS-Middleware principle

The BOSS-Middleware provides very simple communication mechanisms for applications running on the top of it, regardless if they are implemented in software or in (FPGA) hardware (now in development). It was designed to support fault tolerance. All processes running on top of the BOSS-Middleware can exchange messages asynchronously using a subscriber protocol: a process or a hardware device can subscribe to one or more message types by name. When a process or a hardware device sends a message of a given type (name), each subscriber to this name receives a copy of the message. For communication purposes, the node and even the software/hardware barriers/boundaries are transparent. The messages are distributed across these barriers. Using this approach, we obtain very high flexibility and users do not have to differentiate between local/remote functions or hardware and software functionality. The system can be configured or reconfigured simply by plugging software modules or hardware devices into/out of the middleware.

The BOSS-Middleware provides transparent support for fault tolerance. The simplest example of this is a controller sending commands (messages) to a device. As a first step, we insert the middleware between the device and the controller by implementing the same interface on both sides of it. Neither the controller nor the device notices this intervention. The middleware forwards the messages across node boundaries, which means that controller and device no longer need to be located in the same node. Furthermore, messages can be replicated if there is more than one subscriber to a message type (name). Now we can add a monitor to hear messages of the same type, like the device. The monitor can create a log file and/or execute an online diagnosis of the system. Again, no one will notice this intervention.

The next step is to replicate the controller, simply by creating several instances of it, if possible running on different nodes. They need not know about the existence of the other replicas. What are needed now are voters that intercept all messages to the device, compare them and send only those that are most likely to be right (a democratic decision, e.g. two of three) to the device. If required, it is possible to replicate the voter, too. One voter – the worker (as in BIRD) – is in charge and the other one – the supervisor (as in BIRD) – is a hot redundancy. The supervisor is ready to take control if the voter in charge fails to respond .

The routing of messages depends only on the types/names of the messages and on who is subscribed to each name.

4. REFERENCES

2005: Kayal, H: BEESat internet presentations:
http://directory.eoportal.org/pres_BEESatBerlinExperimentalEducationalSatellite.html
http://www.beesat.de

1999: Montenegro, S.: Entwicklung sicherheitsrelevanter Systeme, Hanser Verlag,
ISBN: 3-446-21235-3

2003: Briess, K., Baerwald, W., Gill, E., Halle, W., Kayal, H., Montenbruck, O., Montenegro, S., Skrbek, W., Studemund, H., Terzibaschian, T., Venus, H.:
Technology Demonstration by the BIRD Mission
4th IAA Symposium on Small Satellites for Earth Observation, April 7-11, 2003
ISBN 3-89685-569-7

2002: Briess, K., Bärwald, W., Hartmann, M., Kayal, F., Krug, H.3, Lorenz, E., Lura, F., Maibaum, O., Montenegro, S., Oertel, D., Röser, H.P., Schlotzhauer, G., Schwarz, J., Studemund, H., Turner, P., Zhukov, B.:
Orbit Experience and First Results of the BIRD Mission

53rd International Astronautical Congress The World Space Congress 2002, October 10-19, 2002, Houston, Texas, USA

2002: Briess, K., Montenegro, S., Bärwald, W., Halle, W., Kayal, H., Lorenz, E., Skrbek, W., Studemund, H., Terzibaschian, T., Walter, I.:
Demonstration of Small Satellite Technologies by the BIRD Mission
16th Annual AIAAA/USU Conference on Small Satellites, Logan, Utah, USA 2002

2002: Montenegro, S., Barr, V.:
BOSS/Ada: An Open Source Ada 95 Safety Kit
Ada Deutschland Conference 2002, March 6-8, 2002, Jena, Germany