DOI. No. 10.1109/MAES.2019.170217

Feature Article:

Developing a Linux-Based Nanosatellite On-Board Computer: Flight Results from the Aalto-1 Mission

***Hannu Leppinen, Petri Niemelä, Nuno Silva, Henry Sanmark, Henrik Forstén, Adrian Yanes, Rafal Modrzewski, Antti Kestilä, Jaan Praks*, Aalto University, Finland**

INTRODUCTION

The on-board computers of a spacecraft handle telecommands sent by the ground, provide telemetry to the ground, process on-board data, and control the spacecraft platform and payloads. A spacecraft may have one or several on-board computers that handle different tasks.

On-board software often makes use of specialized real-time operating systems and is usually tailored to the needs of each mission, leading to software projects spanning several years. Additionally, often only parts of the developed software can be used in following missions, demanding a new long software project for the new mission—even when much of the functionality may be very similar from one mission to another.

One method to facilitate software reuse is to use existing off-the-shelf software modules to accomplish many of the mission tasks while only writing a small portion of mission-specific application code. One way to achieve this, used especially by universities and "new space" companies, has been to use embedded Linux [1]. However, only some of these projects have published how they built their Linux-based on-board software systems. This article describes the development, design, initial flight results, and lessons learned of the Linux-based on-board computer (OBC) of Aalto-1.

BACKGROUND

EMBEDDED LINUX

Linux is a general-purpose operating system used in embedded systems because of quality and availability of code, wide hardware support, implementations of communication protocols and application programming interfaces, available development tools, favorable licensing conditions, vendor independence, and cost. The Linux kernel can be run in various Linux systems from very small embedded computers to supercomputer clusters. Each system has different purposes, software packages, and applications [2]. Linux provides operating system facilities such as virtual memory, processes, communication sockets, and file systems, and the Linux kernel has support for many data buses and communication protocols useful for spacecraft developers [3].

One of the first CubeSats to use Linux was QuakeSat, launched in 2003. Since then, Linux has been used in several CubeSats and other spacecraft, perhaps most notably in Falcon and Dragon space vehicles built by SpaceX. CubeSats that have used Linux have been studied in [1], and include at least QuakeSat, UWE-1 and UWE-2, IPEX, Lightsail-1, PhoneSat satellites, and the Dove satellite constellation. The STRaND-1 mission that used both a Linux-based single-board computer and an Android smartphone has been described by Bridges et al. [4].

THE AALTO-1 PROJECT

Aalto-1, shown in Figure 1, is a 4 kg nanosatellite designed and built by students and researchers at Aalto University. The project started in 2010 [5] and the satellite was launched on June 23, 2017. In addition to the on-board computer, platform subsystems include Ultra High Frequency (UHF) [6] and S-band radios [7], a global positioning system (GPS)-based navigation system [8], Electrical Power System (EPS) [9] and Attitude Determination and Control System (ADCS) [10].

The satellite carries three payloads. Aalto Spectral Imager (AaSI), built at the VTT Technical Research Centre of Finland, is a technology demonstration that aims to produce images of the Earth in several narrow visual and near-infrared wavelength bands [11]. Radiation Monitor (RADMON), built at the University of Turku, is used to gather data on the 10–200 MeV proton and 0.7–10 MeV electron environment around Earth [12]. Electrostatic Plasma Brake (EPB), built by a consortium led by the Finnish Meteorological Institute, is a charged tether that will be deployed by spinning the satellite near the end of the mission. It is expected that when moving through the ionosphere, the charged tether produces an observable braking effect due to Coulomb drag [13].

The design of Aalto-1 has been based around the 3U CubeSat standard, which sets the satellite size at 340.5 mm × 100.0 mm × 100.0 mm and maximum mass at 4 kg. Using the CubeSat standard has allowed cooperation and learning from other academic CubeSat projects.

The satellite is intended to be operated for approximately two years. In the first part of the mission, AaSI and RADMON are op-

Authors' current address: Aalto University, School of Electrical Engineering, Otakaari 5A, Espoo, 02150 Finland, E-mail: (hannu.leppinen@aalto.fi).
Manuscript received December 13, 2017, revised April 22, 2018 and June 12, 2018, and ready for publication June 12, 2018.
Review handled by E. Blasch.

erated in three-axis stabilized mode, and ADCS is used to point the AaSI instrument toward ground targets of interest. In the second phase, the ADCS is used to spin up the satellite in order to deploy the EPB using centrifugal force [14].

ON-BOARD COMPUTER DEVELOPMENT

TIMELINE

The Aalto-1 satellite concept was the outcome of a space technology course at Aalto University during spring 2010. Linux was already chosen as the flight computer operating system at the beginning of the project [15]. Off-the-shelf single-board computers such as Gumstix or Beagleboard were considered for the OBC, but it was decided to develop custom hardware for educational purposes. By the preliminary design review in late 2011, the on-board computer design included two cold-redundant AT91RM9200 processors and an arbiter microcontroller selecting which processor is active. By this stage, the main architecture and responsibilities of the software had been outlined, including kernel configuration and use of the Unsorted Block Image File System (UBIFS).

Flight software development began with first commits to the Git version control system in early 2012, focusing on making all the required drivers work. Software development started AT91RM9200-based single-board computers such as BF220 and Centipad. First versions of the on-board computer, modeled after these two existing computers, were produced in 2012 [16].

Five versions of the hardware were produced iteratively during the project. For simplicity, first versions of the custom on-board computer hardware only included one of the two redundant processor branches. By the critical design review in mid-2013, versions 2 and 3 of the on-board computer had been built. Due to the rapid and iterative development, much of the on-board computer hardware and software designs, including documentation, were placed in GitHub under version control, while Google Drive was used elsewhere in the project.

Most of the application software and the flight hardware were developed between 2013 and 2015. Versions 4 and 5 of the on-

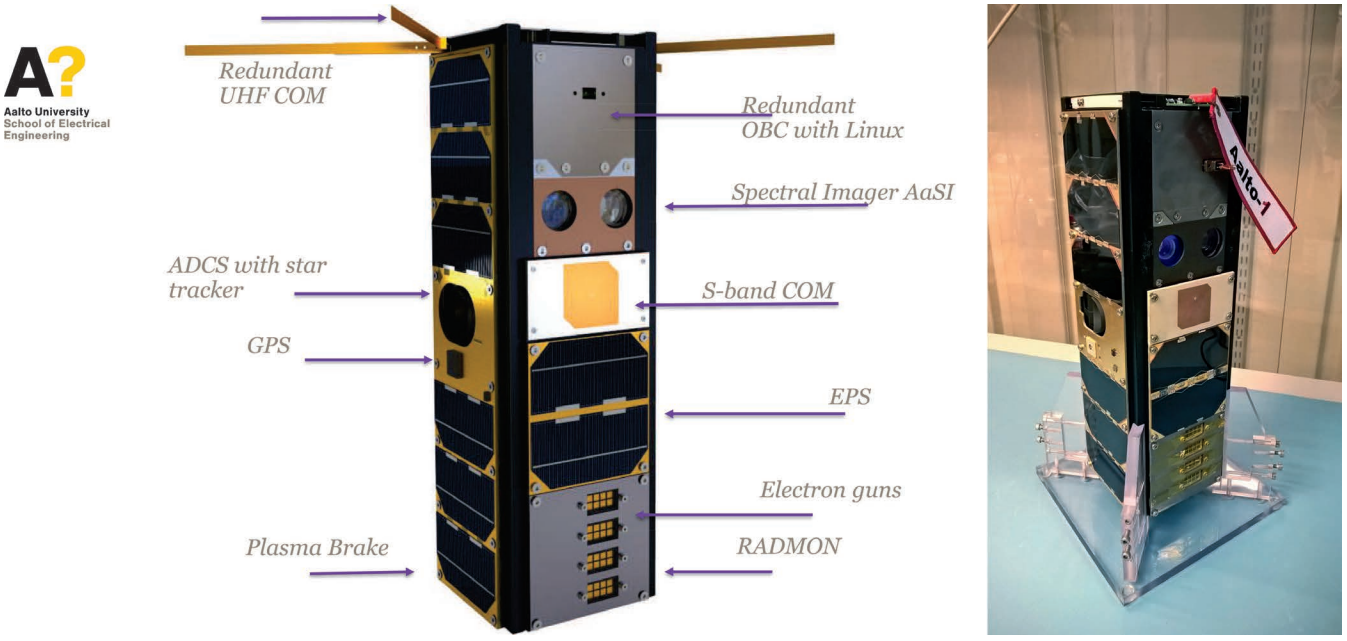

Figure 1.
Flight model of the Aalto-1 nanosatellite, with various subsystems highlighted.

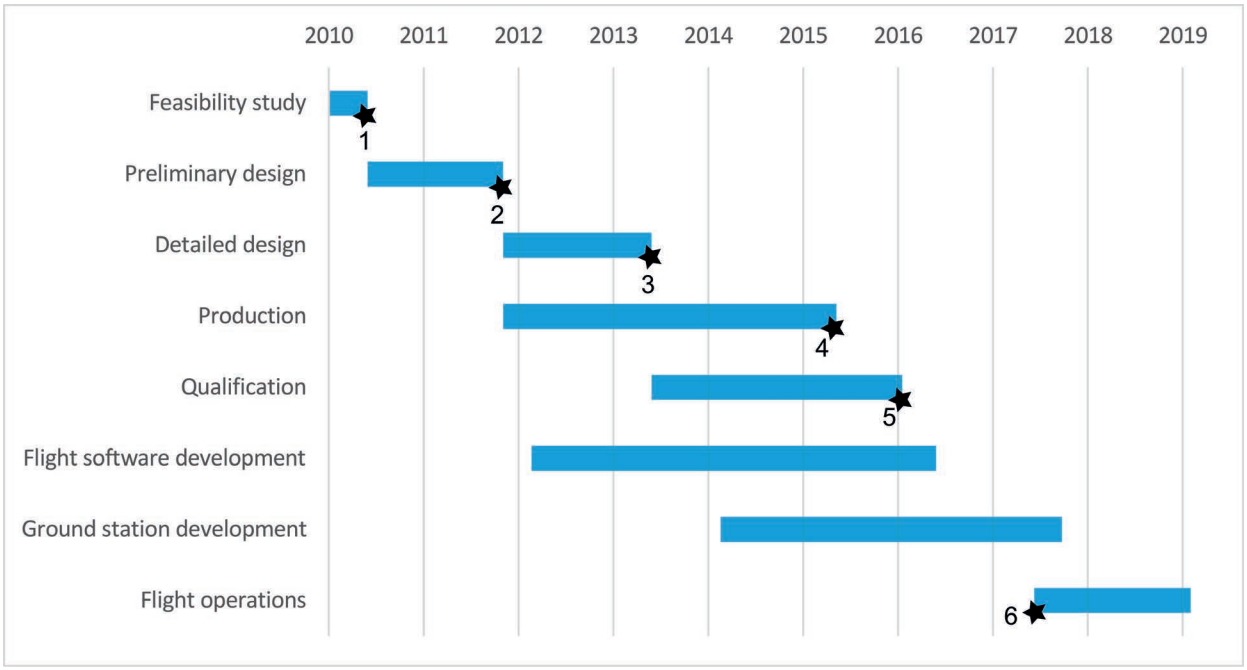

Figure 2.
Aalto-1 development phases and milestones, with software and ground station development in focus. The on-board computer hardware production aligns with the overall satellite production phase. (1) Initial report published; (2) preliminary design review; (3) critical design review; (4) test readiness review; (5) flight readiness review; and (6) launch.

board computer were developed in 2014, with version 5 becoming the flight model. The launch had been scheduled to occur already in 2015, but after being repeatedly delayed, the additional time was used to finalize the ground segment to match the on-board telecommand/telemetry implementation. By mid-2016, the on-board software design was final, but improvements to the ground station continued up to the launch and slightly beyond it.

Although the satellite development milestones have followed the waterfall model, the actual development process has been iterative, as subsystems and software have been prototyped several times and sometimes remade from scratch. On-board software development time has been estimated by analysing version control system commits to the project's Git repository. The five most active developers contributed 90% of the commits, and based on their commit rate per day, it was estimated that developing the Aalto-1 on-board software took around 6.195 person-years. The Aalto-1 development timeline is visualized in Figure 2.

IMPLEMENTATION

Designs for the Aalto-1 on-board computer circuit boards have been mostly done using Autodesk Easily Applicable Graphical Layout Editor (EAGLE). Like the Linux kernel, the on-board computer software has been written mostly in C, and high-level operational scripts—known as on-board control procedures in traditional space industry—have been implemented as shell scripts. The board designs and the software source code are maintained in GitHub.

Much of the satellite software has been developed with desktop PCs with common tools used for C development on Linux, such as GNU Compiler Collection and other tools in the GNU toolchain. Development tool versions were frozen to ensure repeatable results across developers. To facilitate remote development and testing with real hardware, the Aalto-1 satellite engineering model was connected to the internet using secure shell (SSH) to provide remote access. This connection has allowed developers to upload the latest versions of application programs and operation scripts to the on-board computer to test them in the presence of actual subsystem hardware models.

The engineering model consists of the latest working models of platform and payload subsystems. The model is still maintained at the university laboratory as a "twin" of the orbiting satellite, and procedures can be tried on it first before performing them on the orbiting satellite. The engineering model is nearly identical to the flight model, but some subsystems such as the ADCS and the EPB are only partially implemented.

QUALITY ASSURANCE

To ensure that the satellite and the on-board computer hardware can function in space, environmental test campaigns have been performed. The tests, described in more detail in [17], included full vibration, shock, and thermal tests and limited radiation tests. The tests have provided confidence that the designed systems will function in space.

Quality assurance of the Aalto-1 on-board software has focused on following a software development process and performing high-level functional tests on the developed systems. The code has been made available to all software team members in GitHub for cross-review. Compiler warnings and static analysis with clang

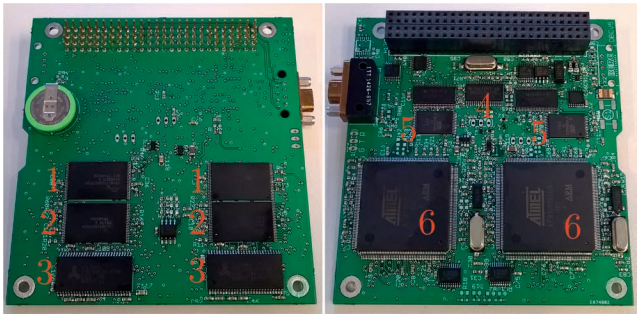

Figure 3.
The bottom and top sides of the Aalto-1 on-board computer flight model, with the microcontrollers and memory chips annotated: (1) 256 MB NAND Flash, (2) 8 MB NOR Flash, (3) 32 MB SDRAM, (4) MSP430 arbiter, (5) 8 MB DataFlash, (6) AT91RM9200 microcontroller.

scan-build tool have been used to detect potential problems in the source code. Developers have been expected to perform at least basic unit testing on their component under development before uploading the software to the engineering model for integration testing.

Integration testing has been done by driving the developed programs manually on the engineering model, or by writing shell scripts that use several programs together. So-called "health check scripts," written as a part of mission operations development, have also been used. These scripts can be run before starting some scientific activity, to ensure that all required subsystems are responsive and healthy. The health check scripts were run, for example, during and after environmental qualification tests to ensure that the health of subsystems is not affected by the environmental stresses. As the scripts consist of very high-level checks, they require much of the satellite to work perfectly; unfortunately, when problems appear, they may not be sufficient to track down the exact cause of problems.

Testing has focused on only those software parts that have been written for the project, as Linux and other off-the-shelf software components are expected to be reliable. While all software components written for the project have been tested, most testing has focused on the communication software, since it is the most critical.

ON-BOARD COMPUTER DESIGN

HARDWARE

The Aalto-1 OBC is a single-board computer designed to fit on a single 9.0 cm × 9.6 cm CubeSatKit circuit board [18]. (While Aalto-1 has several microcontrollers with various firmware, the term OBC refers to the satellite main computer.) It has two cold-redundant AT91RM9200 microcontrollers running at 150 MHz as the main processors. An MSP430FR5729 microcontroller acts as an arbiter, selecting which of the main processors is active. The mass of the on-board computer is only 75.0 grams, and power consumption is between 250 and 450 mW from a 3.3 V power supply. The Aalto-1 OBC flight model is depicted in Figure 3.

Each processor is externally interfaced to its own 8 MB DataFlash, 8 MB Parallel NOT-OR (NOR) Flash, 32 MB random access memory (RAM), and 256 MB NOT-AND (NAND) Flash. Each NAND flash stores two file system images (a nominal 200 MB file system and a backup 56 MB file system), thus the satellite has in total four file system images. The file systems are used to store Linux system files and mission data. A software Error Correction Code (ECC) driver in the Linux kernel is used to protect the data on the NAND flashes. The DataFlash and NOR Flash are used as reprogrammable boot memory. Both contain a bootloader and Linux kernel, and both can be used to boot the system. The Linux kernel and rest of the on-board software are loaded to RAM at boot. Components with CubeSat or other spaceflight heritage have been preferred when selecting parts for the on-board computer; a comprehensive list can be found in [19].

The CubeSatKit stack connector, shown on top of Figure 3, is used to connect the remaining satellite systems to the on-board computer via I^2C, serial peripheral interface (SPI), and universal asynchronous receiver/transmitter (UART). OBC is the master of the I^2C and SPI buses. A micro-D connector on the side of the board provides UART access to the OBC and allows recharging the satellite batteries. The connections are shown in Figure 4.

The satellite platform has been developed mostly in Aalto University, while some parts such as the attitude control system and electrical power system have been purchased from outside providers. The scientific payloads, as mentioned previously, have been provided by various institutes and universities around Finland.

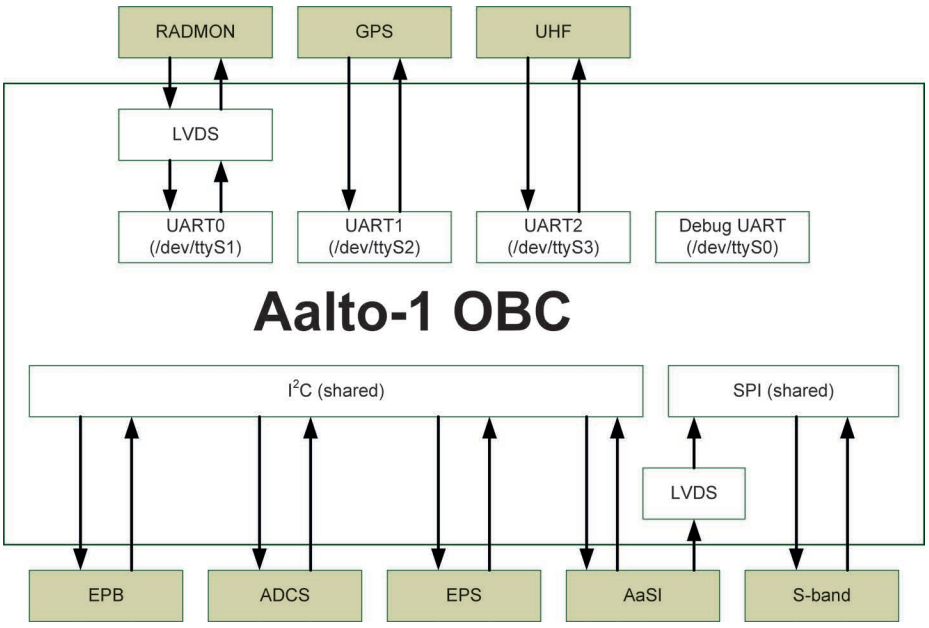

Figure 4.
Subsystems and data buses connected to the on-board computer [19].

SOFTWARE

The Aalto-1 on-board software is built around applications running on Linux. As the on-board computer has no hard real-time deadlines, it is possible to use Linux without any special real-time adaptations. The applications utilise libraries to communicate with satellite subsystems and data buses. A diagram of the software modules is presented in Figure 5. Linux kernel 3.4.106 is used, and some patches have been applied to support the Aalto-1 hardware. Drivers have been added for an external real-time clock and configurations have been modified for SPI, I²C, light-emitting diodes, and heartbeat timers. Kernel patches were applied to add userland interfaces for ECC statistics and for the I²C bus reset functionality.

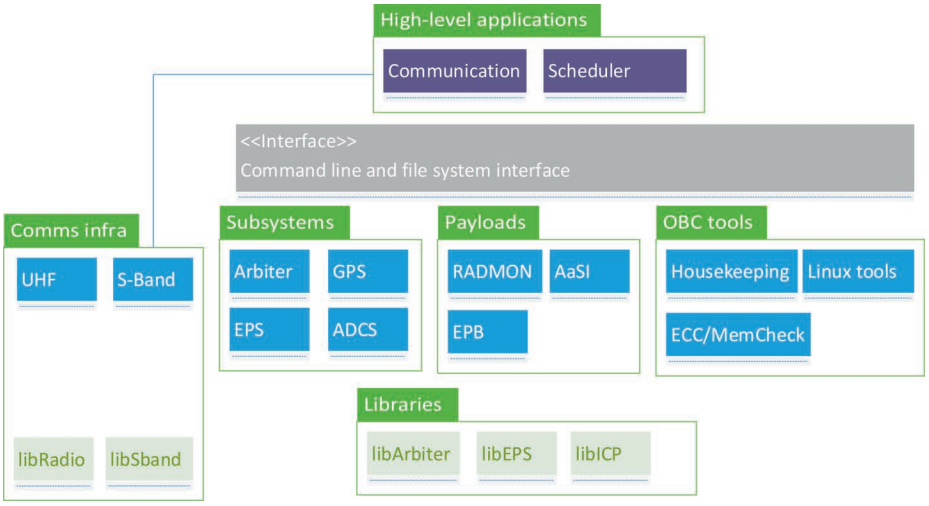

Figure 5.
The Aalto-1 on-board software is based on applications running on Linux. The communications infrastructure allows commanding rest of the subsystems via the command line.

A custom bootloader was developed to minimize the size of the boot partition. On each boot, the bootloader checks a file system selection signal from the arbiter and performs an ECC check on the kernel image. The kernel and the included applications have been carefully tailored to avoid existence of unused software. Buildroot 2014.02 has been used to configure and build the Linux kernel for Aalto-1. BusyBox is used to provide lightweight implementations of several general-purpose Linux programs and utilities. For application programming, uClibc 0.9.33.2 is used as a small footprint version of the C standard library. This selection of tools is quite common for embedded Linux systems [2], [3]. In-house developed utilities include a memory checking program and a housekeeping utility that maintains a data pool of housekeeping values and updates them to the radio beacon.

Libraries developed for the project include libarbiter for communicating with the arbiter, libeps for communicating with the electrical power system, libicp for an internal communication protocol over I²C, and libradio and libsband for communicating with the UHF and S-band radios.

Application programs provide the mission-specific logic to determine how the various subsystems are handled. Subsystem-related application programs include electrical power system manager, GPS manager, attitude determination and control system manager, arbiter manager, S-band and UHF managers, and managers for the scientific payloads. Two highest-level applications are communications manager and mission scheduler. The communications manager handles the telecommand-telemetry link with ground. The telecommand-telemetry protocol is bespoke for the Aalto-1 mission, but it has been inspired by the European Cooperation for Space Standardization (ECSS) packet utilization standard [20].

Most of the programs have been written in C, but shell scripts are used to automate Aalto-1 mission operations. Since many of the application programs controlling various payloads operate with standard input/output, it allows scripts to pipe outputs from some subsystem programs into files or other programs and add logic in between; this follows the UNIX pipeline principle.

Portable Operating System Interface (POSIX) syslog facilities are used to produce log files about system health and software status; these logs are downlinked as a part of housekeeping data.

USING THE COMPUTER

When the ground operator sends a telecommand over the UHF link to the receiver on the satellite, libradio and the communications manager are activated to maintain the ground-satellite data link. The communications manager identifies the incoming telecommand packet and forwards it to the correct on-board service and produced telemetry packets are relayed to the ground. Available services are listed in Table 1. Much of the satellite operations will be handled either by scheduling operations using the scheduler or by uploading shell scripts to the satellite, and by triggering their execution either by the scheduler or by a subsequent telecommand.

An on-board scheduler provides capability for scheduling one-time or periodic operations. The scheduler program runs in a continuous loop and accepts commands for adding and deleting one-time and recurring tasks, and also listing what commands have been currently scheduled. The scheduler can also be switched between safe mode and nominal mode, controlling what list of tasks can be activated, and where scheduler logs are stored. At startup, the scheduler is by default in safe mode, and has to be separately commanded to nominal mode.

Much of the on-board computer software can be updated during flight. The application program binaries and operations script text files are residing on the root filesystem and can be easily replaced by transmitting new files from ground. Since the satellite has redundant boot chains on different memories, the kernels and bootloaders in NOR and DataFlash can be safely updated by transmitting a new kernel image and writing it to the desired memory. The arbiter firmware and most of subsystem firmware, however, cannot be updated during flight.

Table 1.

Aalto-1 Telecommand (TC) and Telemetry (TM) Services		
ID	**Name**	**Description**
1	TC verification	Acknowledge execution of TC packets
2	Ping	Connection test
3	Housekeeping	Transmit health information
5	Execute	Execute a command in the Linux shell
6	Scheduling	Manage the mission scheduler
7	File transfer	Transfer files
8	Retransmission	Retransmit TM packets

Table 2.

Watchdogs Employed in Aalto-1			
Name	**Location**	**Monitored Item**	**Recovery Action**
Bus	EPS firmware	I²C bus	Power cycle
Arbiter	Arbiter firmware	OBC heartbeat	OBC branch logic
Kernel	Linux kernel	Userland watchdog	Reboot OBC
Userland	Linux userland	Comms SW main loop	Stop kernel watchdog
Contact	Comms SW	Time of last communication	Change UHF branch

FAULT DETECTION, ISOLATION, AND RECOVERY

Fault detection, isolation, and recovery (FDIR) measures aim to recover Aalto-1 from any error situation to a known, safe configuration [21]. On a conceptual level, the on-board software contains two high-level modes: *safe mode*, which is entered after boot and in which only the most essential equipment needed for communication are enabled, and *nominal mode*, which is entered when the satellite is commanded to enable further equipment. The satellite stays in nominal mode until it is rebooted again.

FDIR focus has been on the custom code developed for Aalto-1, as common Linux code is expected to be dependable. The aim of FDIR is to maintain the satellite in a state in which communication with the ground station is possible, and thus the outcome of many FDIR functions is to reboot the computer to safe mode. FDIR is handled primarily by the watchdogs listed in Table 2.

The *bus watchdog* in the EPS firmware expects that the EPS is regularly sent commands over one of the redundant I²C buses; this is handled by the OBC. If the EPS does not receive any commands within a specified timeout of a few minutes, it will try to unblock the bus by power cycling the entire satellite.

The *arbiter watchdog* expects a periodic heartbeat signal from the OBC, which is activated when OBC successful boots to userland level. After activation, only kernel panic can stop this heartbeat. After 500 heartbeats, the present OBC configuration is considered a working one. If heartbeat timeout occurs before that, a different OBC configuration will be selected by the arbiter; otherwise, the current configuration is restarted. Each processor has access to two kernels and two filesystems, thus there are in total eight different bootable configurations.

The *kernel watchdog* uses a hardware watchdog in the AT91RM9200 processor supported by the Aalto-1 Linux kernel. It is enabled after booting to userland, and it expects to be regularly refreshed by the userland watchdog. On timeout, the kernel watchdog increments a counter and tries to reboot the OBC without alarming the arbiter. After a certain number of consecutive reboots, the arbiter switching logic is triggered intentionally by inhibiting the heartbeat signal.

The *userland watchdog* is an independent program started at boot that enables the OBC heartbeat (unless it has been decided to trigger the switching logic), activates the kernel watchdog, and monitors the date stamp of a watchdog file on the file system. The date stamp is regularly updated by the main loop of the communications manager. If the watchdog file timestamp is updated, the userland watchdog program resets the kernel watchdog; if the file is not updated, the kernel watchdog is not updated, which will trigger a reboot of the OBC [21].

The *contact watchdog* monitors the time since last communication; the watchdog is triggered if no communication has taken place in the last 12 hours. If communication has not occurred, the active branch of the UHF radio is switched. As 12 hour pauses in communication are quite normal, both UHF radio branches are regularly used.

Fault detection and prevention features also include ECC checking of memories and use of checksums in the communication protocols; a comprehensive list can be found in [21].

GROUND STATION

Aalto-1 is operated from a ground station that is located at the university campus in Espoo, Finland. The ground station has uplink and downlink capability in VHF and UHF bands and downlink capability in S-band. The ground station operations have been semi-automated, so that a single operator sitting at the satellite control room can control the ground station and operate the satellite using a single desktop computer. Remote operations over SSH are also possible. The ground station software has been written in Python, as it has many software modules for radio operations and satellite tracking available off-the-shelf. The telecommand-telemetry protocols written on-board in C are written on the ground side in Python. The automated protocols allow fast exchange of telecommands and telemetry, including uplinking and downlinking files.

Figure 6.
The operator console of the Aalto-1 ground station. The middle display shows the command line interface used to operate the satellite, and the right display shows a waterfall view of the radio traffic.

Due to its sun-synchronous orbit, there are up to six communication passes per 24 h with Aalto-1, each lasting approximately 10 min or less. The operator console of the ground station is shown in Figure 6.

FLIGHT RESULTS

Aalto-1 was launched to orbit on an Indian Polar Satellite Launch Vehicle rocket from the Satish Dhawan Space Centre at 03:59 Coordinated Universal Time (UTC) on June 23, 2017. The launch vehicle carried a total of 31 satellites of various sizes to orbit. Aalto-1 was deployed from the rocket at 04:22:08 UTC, and an on-board timer switched on the satellite approximately 30 min after deployment [22]. The satellite beacon was observed from the ground station immediately after the satellite came over the horizon over Finland, and housekeeping data in the beacon indicated that the on-board computer had started. Two-way telecommand-telemetry link was established on subsequent passes during the same day, showing that the OBC is responsive to commands. The ground station was fine-tuned during the communication sessions, improving the link and reducing packet loss.

Initial operations focused on ensuring that all the subsystems were healthy. After startup and establishing a successful telecommand-telemetry link, it was concluded from the received telemetry that the antennas, electrical power system, UHF radio, and the on-board computer worked as expected. Attitude control system readings indicated that the satellite was slowly tumbling around its long axis. Activating the attitude control was considered too risky to be attempted immediately, and it was decided to maintain the satellite in the slow tumble while other systems were tested.

Other subsystems were tested one by one: the first GPS fix was obtained six days after launch on June 29, making the GPS the first noncritical subsystem to be successfully operated. The GPS measurements helped identify the correct two-line element set from the published set of orbits for satellites launched on the same rocket. The exact orbit is critical for communications; as the UHF carrier wave of Aalto-1 is disabled when no data is transmitted, the Doppler shift of the satellite coming over the horizon needs to be predicted beforehand. Housekeeping data was also requested from the AaSI, EPB, and RADMON payloads to ensure that they also were responsive to commands. The devices were powered up in steps, and housekeeping values from the EPS were monitored.

The first image from Aalto-1, shown in Figure 7, was taken with the AaSI instrument on July 5; the image shows the coastline of Denmark as viewed from over Norway. Transmitting the image over the S-band link was tested. The on-board housekeeping values indicated a nominal transmission, but as the satellite was tumbling and the S-band link requires accurate pointing, no S-band signal was observed at the ground station. Only the UHF downlink was available, and the image was downlinked over a few weeks with the slow bandwidth; the image was fully received on July 21. Further images were taken, and a histogram program was uploaded to the computer to analyse which of the taken images might contain the Earth instead of black space, in order to decide which images to downlink. A script was later developed that triggers imaging when sun sensors indicate that the satellite points toward Earth, eliminating images that only show black space.

Figure 7.
The first image obtained by Aalto-1, with an overlaid map from Google Earth.

The on-board computer housekeeping telemetry values such as central processing unit (CPU) load, RAM usage, file system usage, and power consumption have been nominal. File system usage has been consistent with operations, increasing when test images have been taken and again decreasing when obsolete files have been deleted. Some of the housekeeping values from the first month of the mission

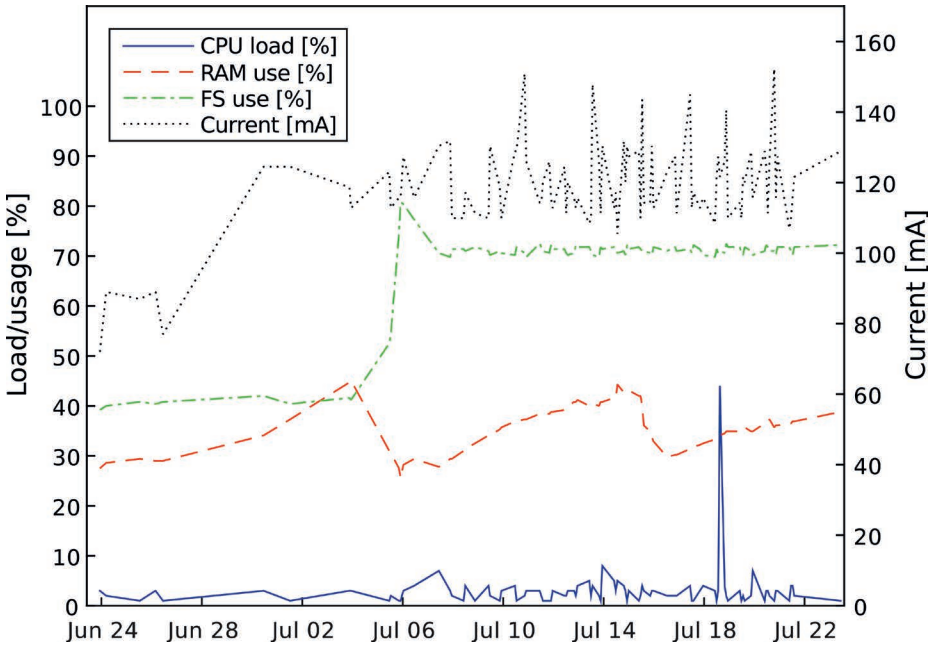

Figure 8.
CPU load, RAM and file system usage, and electrical current from the Aalto-1 OBC during the first month in space.

are visualised in Figure 8. File system usage increased from 40% to 70% during image operations on July 4 and July 5 as images were stored on the computer; the files remained on the OBC as they were being downlinked over the UHF link for much of July. The images were deleted from the on-board computer later on after they had been downlinked. Power consumption of the OBC increased in early July after more features of the computer were enabled after first contact. CPU and RAM usage have been mostly stable; the CPU usage spike nearly July 19 was due to housekeeping data being sampled when AaSI data was being compressed.

The on-board computer branch (OBC1) that was enabled at satellite deployment had around 30-day uptime after launch, before it reset itself in the end of July over the so-called South Atlantic Anomaly; radiation was first suspected, but the problem was found to be caused by a bug activated by direct execution of a certain command on the command line. The communications manager main loop got stuck long enough for the userland watchdog to activate a reboot. The processor restarted without the arbiter watchdog noticing, and the OBC boot count maintained by the arbiter did not increase while the uptime counted by the OBC was reset. Using the specific command has since been avoided but uploading a fixed binary to the satellite is a possibility.

From early September to mid-October, the satellite suffered from instability in the electrical power system, resulting in several resets of the electrical power system and the arbiter. On September 2, the redundant on-board computer (OBC2) had become the active one, and restoring OBC1 was attempted unsuccessfully; an electrical hardware malfunction was suspected. However, restoring OBC1 was later successful, indicating that the failure was not permanent.

A Coronal Mass Ejection (CME) occurred in early September 2017, providing an excellent opportunity to test the radiation monitor. The satellite was quickly retasked to collect as much data as possible with RADMON. Elevated proton counts were detected as a result of the solar storm; however, the instrument had not yet been fully calibrated, and the proton energies could not yet be accurately categorized. Additionally, data collection was sometimes interrupted due to OBC reboots; data collection was restarted from ground when necessary.

A total of 38 boot events occurred during the first five months of the mission. (A boot event refers to a group of one or more boots, with the group having a likely common cause.) The first boot event occurred after deployment as planned, starting up the OBC for the first time during the flight. In July, as mentioned, there was one boot due to a software (SW) bug and one commanded boot. On September 15, another boot was commanded, as this is the only way to change the active file system. In September and October, a total of 11 EPS boot events and 20 boot events of the arbiter occurred. An EPS boot event is caused either by a blocked I^2C bus or battery voltages becoming too low; since no I^2C troubles have been observed, the battery voltage is considered the likelier cause. A boot event of the arbiter is caused by a power loss or unstable input voltage, thus actually 31 boot events are likely related to the battery voltage problems. Cause of the battery problems is unknown; no battery problems were observed in July, August, or November. Certain devices may drain more power when activated, but no clear pattern has been established.

A few unexplained boot events of the OBC, which were resolved by the kernel watchdog without involvement of the arbiter, occurred during the CME event in September. It is suspected that these may be related to radiation, and it is also possible that some of the EPS and arbiter boot events are related to radiation. Boot events during the first five months of the mission, overlaid on the near-Earth proton flux [23], are visualised in Figure 9.

The satellite has mostly been commanded by either executing direct commands on the command line, or by uploading small scripts and then executing them. Some existing program binaries have also been updated. Existing scripts on the satellite have also been modified: for example, the AaSI script was modified to compute histograms and to compress the obtained images automatically. In addition to direct commanding and scripts, the on-board mission scheduler has been used to operate the mission.

After collecting as much calibration and science data as possible with no attitude control, detumbling was started in late September, around three months after launch. A problem was discovered in the attitude control system that caused it to reset itself 90 s after activation, interrupting detumbling. Implementing the attitude control algorithms on the OBC and driving the attitude

control system "manually" using only low-level commands was considered, but instead a script was developed that restarts the attitude control system with correct settings after the 90 s reset, removing the need for the ground operator to manually restart the attitude control system. However, detumbling was still not working, and a firmware problem in the ADCS was suspected.

In June 2018, a working ADCS firmware was finally uploaded to the satellite, allowing successful detumbling. Scientific operations continue, and the results so far are enough to know the performance and characteristics of the Linux-based command and data handling system in orbit.

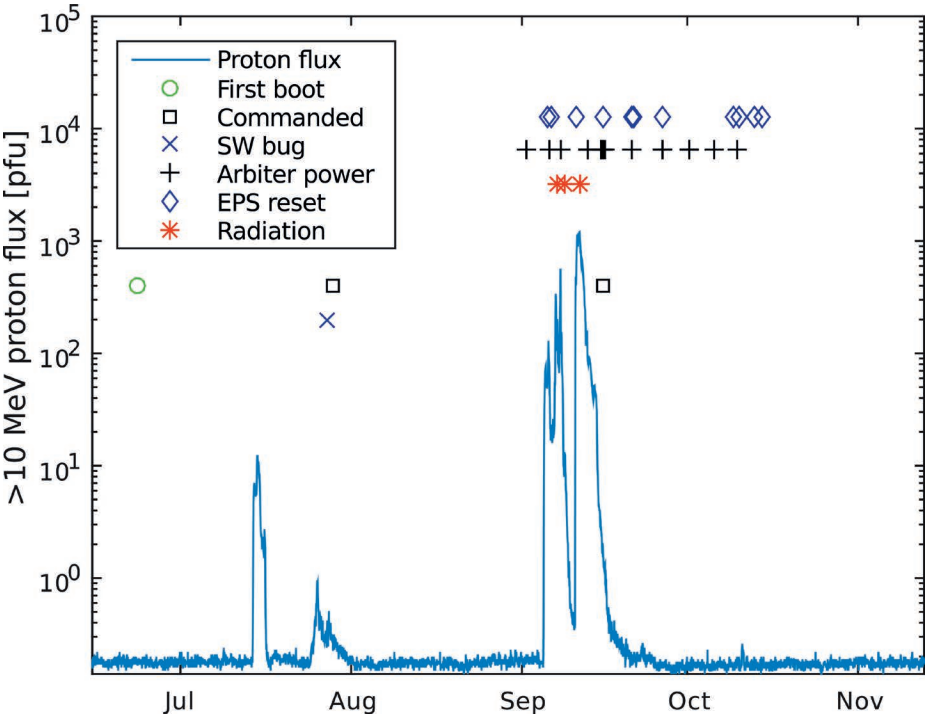

Figure 9.
Various boot events affecting the OBC from the first five months in space, overlaid on proton flux values obtained from [23].

DISCUSSION AND LESSONS LEARNED

Aalto-1 has been the first nanosatellite project for most of the students involved, resulting in some suboptimal decisions affecting quality and schedule. Some more obvious lessons learned by the student team included selecting and freezing development tools early on, appreciating the importance of diagnostics functions in the software, and using checklists during ground station operations. The I^2C main bus was found to be prone to lockups, and Controller Area Network bus was selected to be used in future Aalto University satellites, starting from Aalto-2.

Real-time adaptations of Linux were not needed in Aalto-1 as the OBC did not have hard real-time deadlines. In missions where hard real-time deadlines exist, one option is to use a dedicated microcontroller to handle them; another option is to use some real-time Linux variant [1]. Resources lost to unused packages were not a problem in Aalto-1, as the OBC had performance and disk space to spare, and the Linux system created with Buildroot was tailored to remove most unused features.

Off-the-shelf computer hardware should preferably be used instead of designing custom hardware, unless desirable from an educational point of view. This would also allow more focus on developing the application software, as the embedded computer likely already has tools for basic Linux setup.

Using the UNIX pipeline principle allowed developing smaller, modular programs and quickly integrating them with a wide range of existing utilities for Linux. Existing off-the-shelf utilities on the computer have also provided a useful operations toolkit for example for compressing data to be downlinked. Modularity also facilitated in-orbit updates.

The satellite file systems could have been partitioned better. For example, there could be a read-only root file system that would always be bootable if the memory hardware is intact. Science and mission data could be maintained in a separate file system, and the basic system could be bootable even without the science file system. In Aalto-1, both boot-critical data and modifiable mission data are maintained on the same file system, unnecessarily adding risk of unintended writes to the boot-critical files which could make that configuration unbootable.

An interface control spreadsheet, maintained in Google Drive, sped up the development of the software [24]. The most important interface details of each subsystem, used as the baseline for development, were maintained in the spreadsheet. The interface control spreadsheet was managed by the system engineer who either accepted or rejected incoming changes. Having a centralized document to which changes were proposed avoided the process of preparing lengthy interface control documents and reviewing them; such documents could also have become outdated very quickly [25].

Development of the on-board data handling system should heavily focus on providing a working and robust telecommand-telemetry link with ground, including high throughput services such as file transfer. It is also prudent to develop a dedicated housekeeping service to obtain information about the satellite health without invoking too many other high-level functions. When the basic communication system is in place, in a Linux-based system it is very easy to develop modular programs that communicate with each other for example over the command line.

As the Linux-based computer is a very complex and capable system, it is easy to lose track of the current system state. It takes several telecommands and telemetry back and forth to establish, for example, what is the current filesystem state, which programs

are running, etc. The limited communication bandwidth should have been considered more during development, for example by integrating the slow bandwidth to all executed tests.

FDIR on satellite level was considered unnecessarily late in Aalto-1 development, and while separate systems were able to handle their own failures, FDIR integration on the satellite level proved difficult; for example, the arbiter firmware could not be directly commanded to change OBC branch, thus the branch changing had to be triggered by inhibiting the OBC heartbeat [21].

The on-board scripts have proved very useful and versatile when unexpected situations have occurred, and they were found to be excellent tools for functional testing of the satellite. The mission planners were able to produce operations scripts which formalised the mission plan into a testable format, allowing a test-driven development approach. While not done in Aalto-1, such scripts could be taken into use as automated test scripts, executed, for example, every night as part of continuous integration as suggested in [26].

CONCLUSIONS

This article outlined the development process, ultimate design, and flight results of the Aalto-1 on-board computer and software. Based on Git commits, development of the Linux-based on-board software took approximately 6.195 person-years of work; this development pace by a student team was made possible by the large existing catalogue of off-the-shelf software that could be taken into use as a part of the flight software. The existence of freely available and powerful development environments for Linux was also a contributing factor.

The satellite was successfully launched on June 23, 2017, and the results as of mid-2018 have indicated that the satellite and its on-board computer continue to perform well. No hardware problems within the OBC have been observed, and software problems have been circumvented with in-flight patches and avoiding use of certain configurations and commands. During the first five months of the mission, the on-board computer encountered 38 boot events, of which 35 have been unplanned; however, 31 out of 35 seem to be related to the electrical power system rather than the on-board computer, one boot event is explained by an identified software bug, and the remaining three could be radiation-related, as they were observed during a solar storm in September 2017.

Lessons learned during development included favoring off-the-shelf computer hardware over in-house development, developing modular programs using the UNIX pipeline principle, partitioning file systems to read-only boot partitions and mission data partitions, maintaining a centralized interface control spreadsheet, keeping in mind that the slow communication link limits the observability of the system state during operations, taking FDIR design into account early in the development process, and preparing automated test scripts that can later be reused as mission operations scripts. ◈

ACKNOWLEDGMENTS

The authors would like to thank Konsta Hölttä, Joonas Javanainen, Nemanja Jovanovic, Toni Kangas, Shengchang Lan, Seppo Mäntylä, Leo Nyman, Antti Näsilä, Billy Pitiot, Arttu Punkkinen, Mikael Ranta, and Elyas Razzaghi for their contributions to the Aalto-1 on-board computer and this article. The research has been funded by Aalto MIDE programme and Aalto University, and the launch of Aalto-1 has additionally been funded by Space Systems Finland, Nokia, University of Turku, and RUAG.

REFERENCES

[1] Leppinen, H. Current use of Linux in spacecraft flight software. *IEEE Aerospace and Electronic Systems Magazine,* Vol. 32, 10 (Oct. 2017), 4–13. [Online] Available: http://dx.doi.org/10.1109/MAES.2017.160182.

[2] Yaghmour, K., Masters, J., Ben-Yossef, G., and Gerum, P. *Building Embedded Linux Systems* (2nd ed.). Sebastopol, CA: O'Reilly, 2008.

[3] Hallinan, C. *Embedded Linux Primer* (2nd ed.). Upper Saddle River, NJ: Prentice Hall, 2011.

[4] Bridges, C., Yeomans, B., Iacopino, C., Frame, T. E., Schofield, A., Kenyon, S. et al. Smartphone qualification and Linux-based tools for CubeSat computing payloads. In *Proceedings of the 2013 IEEE Aerospace Conference*, Mar. 2013. [Online] Available: http://dx.doi.org/10.1109/AERO.2013.6497349.

[5] Praks, J., Kestilä, A., Hallikainen, M., Saari, H., Antila, J., Janhunen, P. et al. Aalto-1—An experimental nanosatellite for hyperspectral remote sensing. In *Proceedings of the 2011 IEEE International Geoscience and Remote Sensing Symposium*, Jul. 2011, 4367–4370. [Online] Available: http://dx.doi.org/10.1109/IGARSS.2011.6050199.

[6] Lankinen, M. Design and testing of antenna deployment system for Aalto-1 satellite. Master's thesis, Aalto University, Espoo, Finland, 2015.

[7] Jussila, J., Ben Cheikh, S., Holopainen, J., Lankinen, M., Kestilä, A., Praks, J. et al. Design of high data rate, low power and efficient S-band transmitter for Aalto-1 nanosatellite mission. In *Proceedings of the 2nd IAA Conference on University Satellites Missions and CubeSat Workshop*, Feb. 2013, 811–829.

[8] Leppinen, H. Integration of a GPS subsystem into the Aalto-1 nanosatellite. Master's thesis, Aalto University, Espoo, Finland, 2013.

[9] Hemmo, J. Electrical power systems for Finnish nanosatellites. Master's thesis, Aalto University, Espoo, Finland, 2013.

[10] Tikka, T., Khurshid, O., Jovanovic, N., Leppinen, H., Kestilä, A., and Praks, J. Aalto-1 nanosatellite attitude determination and control system end-to-end testing. In *Proceedings of the 6th European CubeSat Symposium*, Estavayer-le-Lac, Switzerland, Oct. 2014.

[11] Mannila, R., Näsilä, A., Viherkanto, K., Holmlund, C., Näkki, I., and Saari, H. Spectral imager based on Fabry-Perot interferometer for Aalto-1 nanosatellite. In *Proceedings of the SPIE*, Vol. 8870, 2013. [Online] Available: http://dx.doi.org/10.1117/12.2023299.

[12] Peltonen, J., Hedman, H.-P., Ilmanen, A., Lindroos, M., Määttänen, M., Pesonen, J. et al. Electronics for the RADMON instrument on the Aalto-1 student satellite. In *Proceedings of the 10th European Workshop on Microelectronics Education (EWME)*, May 2014, 161–166. [Online] Available: http://dx.doi.org/10.1109/EWME.2014.6877418.

[13] Khurshid, O., Tikka, T., Praks, J., and Hallikainen, M. Accommodating the plasma brake experiment on-board the Aalto-1 satellite. In *Proceedings of the 2014 Estonian Academy of Sciences*, Vol. 63, 2S (May 2014), 258–266. [Online] Available: http://dx.doi.org/10.3176/proc.2014.2S.07.

[14] Kestilä, A., Tikka, T., Peitso, P., Rantanen, J., Näsilä, A., Nordling, K. et al. Aalto-1 nanosatellite - Technical description and mission objectives. *Geoscientific Instrumentation, Methods and Data Systems*, Vol. 2, 1 (2013), 121–130. [Online] Available: http://dx.doi.org/10.5194/gi-2-121-2013.

[15] Näsilä, A., Hakkarainen, A., Praks, J., Kestilä, A., Nordling, K., Modrzewski, R. et al. Aalto-1: A hyperspectral Earth observing nanosatellite. 2011. [Online] Available: http://dx.doi.org/10.1117/12.898125.

[16] Razzaghi, E. Design and qualification of on-board computer for Aalto-1 Cube-Sat. Master's thesis, Aalto University, Espoo, Finland, 2012.

[17] Leppinen, H., Kestilä, A., Tikka, T., and Praks, J. The Aalto-1 nanosatellite navigation subsystem: Development results and planned operations. In *Proceedings of the 2016 European Navigation Conference (ENC)*, May 2016, 1–8. [Online] Available: http://dx.doi.org/10.1109/EURONAV.2016.7530545.

[18] CubeSat Kit PCB Specification. Pumpkin Inc. Std. 810-00 253, 2003. [Online] Available: http://www.cubesatkit.com/docs/CSK_PCB_Spec-A5.pdf.

[19] Silva, N. Reliability analysis of satellite on-board software. Master's thesis, Aalto University, Espoo, Finland, 2017.

[20] Ground Systems and Operations—Telemetry and Telecommand Packet Utilization. European Cooperation for Space Standardization Std. ECSS-E-70-41A, 30 Jan. 2003.

[21] Javanainen, J. Reliability evaluation of Aalto-1 nanosatellite software architecture. Master's thesis, Aalto University, Espoo, Finland, 2016.

[22] Praks, J., Kestilä, A., Niemela, P., Näsilä, A., Leppinen, H., Riwanto, B. et al. Aalto-1 nanosatellite mission status and initial observations. In *Proceedings of the 9th European CubeSat Symposium*, 2017, 125–128.

[23] OMNIWeb—Hourly near-Earth solar wind magnetic field and plasma data, energetic proton fluxes (>1 to >60 MeV), and geomagnetic and solar activity indices. Available: https://omniweb.gsfc.nasa.gov/ow.html, last access Nov. 23, 2017.

[24] Praks, J., Kestilä, A., Tikka, T., Leppinen, H., Khurshid, O., and Hallikainen, M. Aalto-1 Earth observation CubeSat mission: Educational outcomes. In *Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Jul. 2015, 1340–1343. [Online] Available: http://dx.doi.org/10.1109/IGARSS.2015.7326023.

[25] Praks, J., Tikka, T., Kestilä, A., and Hieta, M. Online documentation approach for assisted system engineering and assessment in student projects. In *Proceedings of the 2015 IEEE Global Engineering Education Conference (EDUCON)*, Mar. 2015, 608–611. [Online] Available: http://dx.doi.org/10.1109/EDUCON.2015.7096032.

[26] Space Engineering—Agile Software Development Handbook. European Cooperation for Space Standardization Std. ECSS-E-HB-40-01A DIR1, 6 Mar. 2017.