Automated functional testing and control of Suomi 100 satellite systems and payload instrument control software

Juha-Matti Lukkari

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 16.1.2015

Thesis supervisor:

Prof. Esa Kallio

Thesis advisors:

D.Sc. (Tech.) Antti Kestilä

M.Sc. Juha Itkonen

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Juha-Matti Lukkari

Title: Automated functional testing and control of Suomi 100 satellite systems and payload instrument control software

Date: 16.1.2015 Language: English Number of pages: 7+11

Department of Radio Science and Technology

Professorship: Space Science and Technology

Supervisor: Prof. Esa Kallio

Advisors: D.Sc. (Tech.) Antti Kestilä, M.Sc. Juha Itkonen

Large portion of launched Cubesats have failed early on their missions. Complete lack or inadequate system level functional testing of the satellites has been thought of being one large contributor to these failures. Software failures could be another major contributor. This thesis represents potential solutions to mitigating these issues by the use of free open-source automated test frameworks.
Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained. Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained.

Keywords: For keywords choose concepts that are central to your thesis

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Juha-Matti Lukkari	
Työn nimi: Suomi100 satelliitin järjestelmien ja hyötykuorman instrumentin ohjausohjelmiston automaattinen funktionaalinen testaus ja ohjaus	
Päivämäärä: 16.1.2015 Kieli: Englanti	Sivumäärä: 7+11
Radiotieteen ja -tekniikan laitos	
Professuuri: Avaruus tiede ja tekniikka	
Työn valvoja: Prof. Esa Kallio	
Työn ohjaaja: TkT Olli Ohjaaja	

Tiivistelmässä on lyhyt selvitys (noin 100 sanaa) kirjoituksen tärkeimmästä sisällöstä: mitä ja miten on tutkittu, sekä mitä tuloksia on saatu. Tiivistelmässä on lyhyt selvitys (noin 100 sanaa) kirjoituksen tärkeimmästä sisällöstä: mitä ja miten on tutkittu, sekä mitä tuloksia on saatu.
Tiivistelmässä on lyhyt selvitys (noin 100 sanaa) kirjoituksen tärkeimmästä sisällöstä: mitä ja miten on tutkittu, sekä mitä tuloksia on saatu. Tiivistelmässä on lyhyt selvitys (noin 100 sanaa) kirjoituksen tärkeimmästä sisällöstä: mitä ja miten on tutkittu, sekä mitä tuloksia on saatu. Tiivistelmässä on lyhyt selvitys (noin 100 sanaa) kirjoituksen tärkeimmästä sisällöstä: mitä ja miten on tutkittu, sekä mitä tuloksia on saatu.

Avainsanat: Vastus, Resistanssi,
Lämpötila

AALTO-UNIVERSITETET
HÖGSKOLAN FÖR ELEKTROTEKNIK

SAMMANDRAG AV DIPLOMARBETET

Författare: Juha-Matti Lukkari

Titel: Arbetets titel

Datum: 16.1.2015 | Språk: Engelska | Sidantal: 7+11

Institutionen för radiovetenskap och -teknik

Professur: Kretsteori

Övervakare: Prof. Esa Kallio

Handledare: TkD Olli Ohjaaja

Sammandrag på svenska. Try to keep the abstract short, approximately 100 words should be enough. Abstract explains your research topic, the methods you have used, and the results you obtained.

Nyckelord: Nyckelord på svenska,
Temperatur

Preface

I want to thank Professor Esa Kallio and my instructors Antti Kestilä and Juha Itkonen for their good guidance.

Otaniemi, 16.1.2015

Eddie E. A. Engineer

Contents

Symbols and abbreviations

Symbols

$\mathbf{B}$	magnetic flux density
c	speed of light in vacuum $\approx 3 \times 10^8$ [m/s]
ω_{D}	Debye frequency
ω_{latt}	average phonon frequency of lattice
$\uparrow$	electron spin direction up
$\downarrow$	electron spin direction down

Operators

$\nabla \times \mathbf{A}$	curl of vectorin $\mathbf{A}$
$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\frac{\partial}{\partial t}$	partial derivative with respect to variable t
$\sum_i$	sum over index i
$\mathbf{A} \cdot \mathbf{B}$	dot product of vectors $\mathbf{A}$ and $\mathbf{B}$

Abbreviations

OBC	On board computer
EPS	Electric power system
BCS	Bardeen-Cooper-Schrieffer
DC	direct current
TEM	transverse eletromagnetic

1 Introduction

Cubesats have in recent years emerged as a new viable platform for carrying out space missions. They usually are produced by Universities and over 200 have already been launched. Yet, many of those missions have ended in failure due to various reasons.

The satellite involved in our research is called *Suomi100*.

- About cubesats in general
- Failures with cubesats
- Suomi100 satellite mission
- Doing test automation for Suomi100 satellite
- Robot framework, CUnit

2 Background

- Generally about spacecraft failures
- Failures with Cubesats
- Cubesat failure database, Swartwout research and conclusions -> Inadequate functional system level testing
- Earlier research done on flight software reliability
- Earlier research done on system level functional testing
- Testing done on satellites with big budgets vs testing on Cubesats
- About different testing methodologies, black box, white box

"Looking at the failure reports more closely, a com- mon thread is discovered, accounting for almost half of all failures: a configuration or interface failure be- tween communications hardware (27%), the power subsystem (14%) and the flight processor (6%). Typical examples of such failures: batteries and/or solar panels not connected properly to the power bus; insufficient power generation to operate the transmitter at a level needed to close the link; and unrecoverable processor errors. These can be classified as failures in functional integration; the spacecraft was not operated in a flight- equivalent state before launch, and thus these easily- caught mistakes were not discovered. Though this allegation obviously cannot be proven, it is strongly be- lieved that a large fraction of the “no contact” failures is due to poor functional integration"[1]

"As noted above, it is believed that the failure-rate problem is solvable, given that most failures can be traced back to insufficient system-level functional testing on the ground. Furthermore, it is thought that a “day in the life” operational demonstration is just as essential as vibration testing to certify a CubeSat for flight. Operational tests that demonstrate startup sequences, power management and graceful recovery from resets are all necessary"[1]

- Just as vibrational testing is done by automated machines to verify that the satellite can withstand the launch, automated software and system tests can verify that the satellite operates as expected. Robot framework is free and open-source tool for automated testing, so it is suitable for Cubesat projects.
- Present in Results section some philosophy or methodology for future Cubesat projects on how to do functional system level testing in practice? Some API on the satellite (like csp-client) -> code to send commands to the API automatically -> Some way or some script to run the commands -> Some way to gather information from the satellite on what is going on (csp-client prints to the console).

3 Automated testing done on Suomi100 satellite

- About Suomi100 subsystems
- About Gomspace software and instrument software
- About functional testing
- About unit testing
- Robot framework, CUnit
- Operation modes of the satellite
- Testing environment, gomspace API client
- Automating control of Suomi100 with Robot and gomspace client

4 Results

- Test results
- How operation modes behaved during tests
- How the different subsystems behaved during tests
- How the instrument behaved during tests
- Importantly: What issues with the software and subsystems were found with the tests and what corrections were implemented based on the test results
- AFTER LAUNCH:
- Operation modes
- Things that worked, commands that went through
- About failures if those happened, software crashes
- What was found during testing and comparison to how the satellite performed in orbit. Did the implemented changes to the software improve reliability.

5 Summary

- Suomi100 mission
- Failures with cubesats
- We tried to improve overall system reliability with automated functional tests
- We tried to improve instrument software reliability with automated unit and functional tests
- Did we improve the functionality and reliability of the satellite?

References

[1] Swartwout, Michael
The First One Hundred CubeSats: A Statistical Look Journal of small satellites, 2013.

A Esimerkki liitteestä

```
/*
Radio Operation modes
*/
#include <radio.h>

#include <inttypes.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <pthread.h>
#ifdef linux
#include <time.h>
#else
#include <FreeRTOS.h>
#include <task.h>
#endif

#include <util/console.h>
#include <util/log.h>
#include <util/color_printf.h>
#include <util/clock.h>

#include <csp/csp.h>
#include <csp/csp_endian.h>
#include <csp/arch/csp_thread.h>
#include "radio_config.h"
#include "libradio.h"
#include "radio_calculation.h"
#include "radio_property.h"


// Syntax for different arguments val1;val2;val3;
// NOTE: Frequency in kilohertz
void raw_data_mode(char *mode_args)
{
        uint32_t t_start = 0;
        uint16_t f_const = 0;
        uint32_t n_times = 0;
        uint32_t t_sleep = 0;
        uint8_t ferrite = 0;
```

```
char temp[32] = {0};
unsigned int i = 0;
unsigned int j = 0;
unsigned int argc = 0;
while(i < strlen(mode_args))
{
        if(mode_args[i] != ';')
        {
                temp[j] = mode_args[i];
                j++;
        }
        else
        {
                if(argc == 0)
                        t_start = strtol(temp, NULL, 10);
                else if(argc == 1)
                        f_const = strtol(temp, NULL, 10);
                else if(argc == 2)
                        t_sleep = strtol(temp, NULL, 10);
                else if(argc == 3)
                        n_times = strtol(temp, NULL, 10);
                else if(argc == 4)
                        ferrite = strtol(temp, NULL, 10);
                j = 0;
                argc++;
                printf("temp:%s\n", temp);
                memset(temp, 0 ,sizeof(temp));
                if(argc > 4)
                        break;
        }
        printf("%c", mode_args[i]);
        i++;
}
printf("Arguments␣received␣for␣operation:%d␣%d␣%d␣%d␣%d\n", t_s
// If values invalid for some reason, use these default values
if(t_start < 0)
        t_start = 0;
if(f_const < 149 || f_const > 50000)
        f_const = 5000;
if(t_sleep < 1)
        t_sleep = 100;
if(n_times < 1)
        n_times = 100000;
if(ferrite < 0 || ferrite > 1)
        ferrite = 0;
```

```
printf("Arguments␣corrected␣for␣operation:%d␣%d␣%d␣%d␣%d\n", t_
// sleep or vTaskDelay
#ifdef linux
        usleep(t_start*10);
#else
        vTaskDelay(t_start);
#endif
printf("Radio␣raw␣data␣mode␣commencing\n");
//Choose ferrite with the switch command, how?

// Open file for appending/writing
// in FreeRTOS create file in /flash/...

// Read from spi
#ifdef linux
        uint32_t start_time = 0;
        uint32_t end_time = 0;
#else
        timestamp_t *start_time = malloc(sizeof(timestamp_t));
        timestamp_t *end_time = malloc(sizeof(timestamp_t));
#endif

#ifdef linux
        start_time = (int)time(NULL);
#else
        clock_get_time(start_time);
#endif
uint16_t *data = calloc(10, sizeof(int));
FILE *dfp;
// Get system time to the filename, FreeRTOS has some command ?
char filename[64] = {0};
char filetime[64] = {0};
#ifdef linux
        sprintf(filetime, "%d", (int)time(NULL));
#else
        sprintf(filetime, "%d", start_time->tv_sec);
#endif
strcat(filename, "m1_");
strcat(filename, filetime);
strcat(filename, ".csv");
if((dfp = fopen(filename, "a")) == NULL)
        fprintf(stderr, "Could␣not␣open␣%s\n", filename);
fprintf(dfp, "Val,Freq\n");

// si chip takes frequency in khz
```

```
        radio_am_tune_freq("0", f_const, 0);
        // Wait for chip stabilization
        #ifdef linux
                usleep(t_sleep*10);     // usleep is deprecated in POS
        #else
                vTaskDelay(t_sleep);    // Milliseconds
        #endif
        for(unsigned int n = 0; n < n_times; n++)
        {
                //spi_16read(data);

                // Append to file
                fprintf(dfp, "%d,%d\n", n, f_const);
        }
        #ifdef linux
                end_time = (int)time(NULL);
                fprintf(dfp, "\nStart: %d End: %d\n", start_time, end_t
        #else
                clock_get_time(end_time);
                fprintf(dfp, "\nStart: %d End: %d\n", start_time->tv_se
                free(start_time);
                free(end_time);
        #endif

        fclose(dfp);
        free(data);
}
```

B Toinen esimerkki liitteestä

```
*** Settings ***
Library          libclient.py
Suite Teardown   Client Close     ${sock}          {proc}

*** Test Cases ***

Test Setup
        ${proc}=                  Client Start     /home/juha/S100/EGSE/EC
        #${proc}=                 Client Start     /home/juha/S100/EGSE/EC
        Set Suite Variable        ${proc}
        Sleep                     5
        ${sock}=                  Connect Socket   s100-juha         5000
        Set Suite Variable        ${sock}


Target Mode
        [Documentation]           The payload radio performs several swee
        [Tags]                    OPMODE-TARGET
        Store Client Responses    ${proc}  Target Mode
        # Are we sure that we start the thread in the satellite?
        #Run Radio Mode           ${sock}  ${proc}  /home/juha/S100/confs
/home/juha/S100/confs/radio_props.cfg  3  0;0;0;0;0;0;0;0;0;
        Run Radio Mode            ${sock}  ${proc}  /flash/radio_params.c
/flash/radio_props.cfg  3  0;0;0;0;0;0;0;0;0;
        Sleep                     2
        # Add proper hk and beacon commands here with flight planner
        # csp-client doesnt have flight planner
        #Get HK                   ${sock}          ${proc}          30
        #Send Beacon              ${sock}          ${proc}          10
        Verify Radio Results      Target Mode
        Send Message              ${sock}          ${proc}          exit_cl
        Close Connection          ${sock}
```