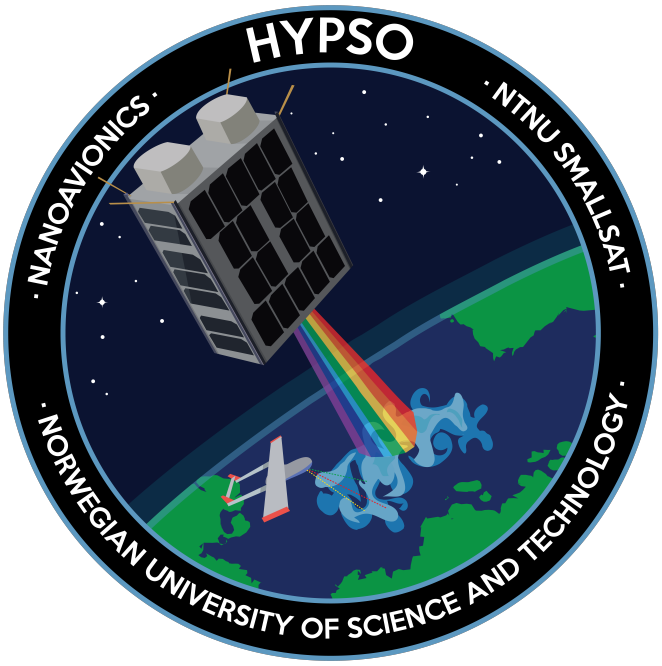
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Simen Berg

Maximizing Payload Utilization of the HYPSO-1 Satellite

Master's thesis in MTELSYS
Supervisor: Milica Orlandic
Co-supervisor: Roger Birkeland
July 2022



HYPSO

NTNU
Norwegian University of
Science and Technology

Simen Berg

Maximizing Payload Utilization of the HYPSO-1 Satellite

Master's thesis in MTELSYS
Supervisor: Milica Orlandic
Co-supervisor: Roger Birkeland
July 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Maximizing Payload Utilization of the HYPSO-1 Satellite

Simen Berg

July 26, 2022

Abstract

This thesis follows the HYPSO-1 satellite – from an operational perspective – from its launch in January 2022 until June of 2022. The HYPSO team utilized spacecraft scripts to automatically capture images of target areas. The thesis shows how operational tools were developed to reduce the sources and number of errors, the required knowledge for the operator, and the time needed to make these spacecraft scripts. With the tools implemented, the HYPSO-1 satellite achieved an average of 2.94 image captures per day in the month of May, reaching a maximum of six captures in a single day. The thesis also shows how operators were trained during the ever-changing operational workflow. It was found that an academic team with a high turnover of personnel had to accommodate different learning styles and focus on providing a fundamental understanding of the system of systems. Lastly, the thesis investigated the possibility of operating HYPSO-1 using only scheduled commands. It was found to be possible, but not achieved within the span of the thesis.

Sammendrag

Denne oppgaven følger HYPSO-1 satellitten fra oppskytning i januar 2022 frem til juni i 2022, fra et operasjonelt perspektiv. HYPSO teamet benytter skript i satellitten for å automatisk ta bilder av ønskede områder. Denne avhandlingen viser hvordan et sett av verktøy ble utviklet for å redusere antall feil og feilkilder, nødvendig forhåndskunnskap hos operatørene og tid brukt for å lage disse skriptene. Med verktøyene implementert oppnådde HYPSO-1 et snitt på 2.94 bilder per dag i mai, med et maksimum på seks bilder på en enkelt dag. Avhandlingen viser også hvordan satellittoperatører ble opplært for en arbeidsflyt i konstant endring. Det ble funnet ut at en akademisk arbeidsgruppe med høy utskiftning av medlemmer må legge til rette for flere ulike lærestiler, og fokusere på å gi medlemmene en fundamental forståelse av hele systemet. Til slutt undersøkte avhandlingen muligheten for å operere HYPSO-1 med bare å bruke planlagte kommandoer. Dette er mulig, men ikke oppnådd i løpet av avhandlingen.

Preface

Prior to starting this thesis, I have been fortunate enough to have the opportunity of working with CubeSats through the student organization Orbit NTNU. It has been four amazing years of working on the SelfieSat, which unfortunately is yet to be deployed after its launch on SpaceX's Transporter 5 mission. Satellites really sparked my interest and I am grateful that I got the chance of working with the operations of the HYPSO-1 satellite.

Working with the operations of a satellite can be a bit overwhelming because of all the systems involved, but also very rewarding. I learned a whole lot in such a small amount of time by getting the chance to be a part of so much. Building the ground station, follow the launch, be there during the first ping, power on the payload, and see the first images. Just thinking about it gives me goosebumps. It is crazy to me that this is my job. Seeing the progress over the thesis period also gave me motivation since I directly saw the effects of my contributions. I really felt like I was solving a problem, which is why I started my engineering degree in the first place.

The title of the thesis changed multiple times. Because operations of a satellite include many systems, there is a lot of overhead which made it hard to focus on a specific area at times. Controlling the satellite gives a lot of insight but is also time-consuming. Additionally, trying to troubleshoot problems that were not contained to the payload was hard due to my lack of insight into the systems developed by NanoAvionics. Even though events like this were not directly productive for my thesis, I obtained a lot of valuable knowledge and experience. I also got the opportunity to work on papers alongside members with a lot more experience than me. Looking back at it, I could not have wished for a better thesis, team, or satellite to work with.

Contents

Acronyms

ADCS Attitude Determination and Control Subsystem. 10, 27, 28, 32, 46

AMOS Centre for Autonomous Marine Operations and Systems. 5

API Application Programming Interface. 7, 9

ASPEN Automated Scheduling and Planning Environment. 18, 38

AUV Autnonomous Underwater Vehicle. 5, 55

AWS Amazon Web Services. 7, 9, 35

CAN Controller Area Network. 9, 31

CLAW Colored Littoral Zone and Algae Watcher. 5, 9

CLI Command Line Interface. ix, 9, 10

CRR Commisioning Results Review. 14

CSP CubeSat Space Protocol. ix, 7, 9, 10, 12, 13, 29, 31, 42, 45, 46, 48, 50, 51

ECSS European Cooperation for Space Standardization. 13, 14

ELR End-of-Life Review. 14

EPS Electrical Power Subsystem. 10–12, 27, 46

ESD Electrostatic Discharge. 12

FC Flight Computer. 10–12, 23, 28–30, 34, 35, 37, 46, 50, 87

FPS Frames Per Second. 45, 48

FRR Flight Readiness Review. 14

FTM File Transfer Manager. 7, 8, 36, 59

GENIE generic inferential executor. 18, 38

GenSAA Generic Spacecraft Analyst Assistant. 18, 38, 78

GS Ground Station. 3, 8, 12, 31, 41, 55, 62, 63, 71, 75, 77

GUI Graphical User Interface. 8

HAB Harmful Algal Bloom. 1

HIL Hardware In the Loop. 28

HSI HyperSpectral Imager. 1, 2, 5, 10, 12, 27–29, 31–33, 55, 56, 61, 66, 78

HYPSO HYPer-spectral Smallsat for Ocean observation. iii, v, 1, 3, 5, 6, 10, 13–15, 19, 21, 24–28, 31, 32, 38, 62, 65, 72, 76, 79

IMU Inertial Measurement Unit. 10, 67, 68

IntelliSTAR Intelligent Satellite Technology Automated Reasource. ix, 19, 23, 72

JPL Jet Propulsion Laboratory. 18

KSAT Kongsberg Satellite Services. 7–9, 63, 65

LEO Low Earth Orbit. 1

LEOP Launch and Early Operations. 27

LGPL GNU Lesser General Public Licence. 7

LRR Launch Readiness Review. 14

LUSV Light Unmanned Surface Vehicle. 55

M6P Multi Purpose 6. 7, 9–11, 32, 42, 45, 73

MASSIVE Mission-oriented autonomous systems with small satellites for maritime sensing, surveillance and communication. 5, 6, 53, 63, 79

MOP Mission Operations Plan. 27, 31–33

MTQ Magnetorquer. 10

NA NanoAvionics. vii, 5, 7, 9, 10, 12, 25–27, 30–32, 36, 59

NTNU Norwegian University of Science and Technology. vii, 1, 2, 5, 8, 10, 12, 31, 63, 79

OPU On-Board Processing Unit. 5, 10, 12, 27, 28, 31, 34, 38, 42, 43, 45, 54, 59

PC Payload Controller. 7, 10–12, 23, 27–29, 31, 34–37, 42, 43, 45, 46, 50, 54, 59, 62, 67, 74, 88

RGB Red-Green-Blue. 5, 10, 12, 27–29, 31, 37, 56, 61, 66, 67

RW Reaction Wheels. 10

SDR Software Defined Radio. 12

SSL SmallSat Laboratory. 1, 5, 10, 79

STK System Tool Kit. 29, 34

TLE Two Line Element. 27, 40, 68, 76

TRL Technology Readiness Level. 14, 21

UAV Unmanned Aerial Vehicle. 5, 6

UHF Ultra High Frequency. 10, 12, 27

USV Unmanned Surface Vehicle. 5, 55

VPN Virtual Private Network. 12

XP Extreme Programming. 14, 15, 21, 22

Glossary

CubeSat A small satellite consisting of cubes called units [U]. One unit is 10cm x 10cm x 10cm and is standardized dimensions for small satellites.. vii, 2, 3, 5

Cycle Life The number of charge/discharge cycles a battery can withstand before decreasing in performance [1]. 46

FlatSat A FlatSat is a horizontally integrated system model. Instead of being stacked vertically as in the satellite, the subsystems are connected horizontally to ease the implementation and connection of systems.. 12

Grafana Grafana is open source visualization software [2]. It lets users make custom, insightful dashboards to easily visualize information for other team members [3].. 7, 8

LidSat A FlatSat setup of selected subsystems of the HYPSO-1 satellite at the SmallSat Laboratory in Trondheim.. 12, 22, 28, 29, 34, 73, 74

MCS Mission Control Software (MCS) is a service provided by NanoAvionics that connects ground infrastructure. MCS encompasses a suite of systems developed by NanoAvionics, and is integrated with other services.. 7–10, 28, 29, 31, 34–36, 38, 44, 59, 71

metodology A specific series of steps to be followed [4]. 14, 15, 21, 22

NNG A lightweight broker-less messaging library.. 7, 9

S-band The frequency range spanning 2 GHz - 4 GHz [5].. 7, 9, 10, 27, 31, 55, 63

SatLab SatLab is a company providing sband tranceiver and ground equipment. The ground equipment consists of rotor controller, modem, antenna, and a graphical user interface.. 7–9

Sun Synchronous Orbit A particular class of a polar orbit. Satellites travelling in this type of orbit is synchronous with the Sun, meaning they will always have a fixed position relative to the Sun [6]. . 2

Chapter 1

Introduction

Satellites have been launched since 1957 [7], thus leading to satellite operations being a necessity for around 65 years. After a satellite has been developed, launched, and commissioned, the predominant cost of the mission is the operations of the satellite [8]. Therefore, tools and workflows have been developed to improve the capacity of operators, reduce the number of errors, and better the monitoring of satellite health. This chapter will introduce the motivation behind this thesis and provide a problem statement with the objective of this work.

1.1 Motivation

Harmful Algal Blooms (HABs) have become a more prominent concern over the last century because of a dramatic rise in the severity and geographic extent of these events caused by environmental degradation[9]. HABs are therefore a big concern in countries like Norway since the fish market is one of their biggest exports [10] [11]. In 2019, Norway experienced a HAB, killing nearly eight million Atlantic salmon in fish farms in northern Norway [12]. Monitoring the ocean to get an early warning can help mitigate the severity of such outbreaks and increase the understanding of algae blooms.

On the 13^{th} of January 2022, the HYPer-spectral Smallsat for Ocean observation (HYPSO) team at the SmallSat Laboratory (SSL) at Norwegian University of Science and Technology (NTNU) launched HYPSO-1, a research satellite equipped with a novel HyperSpectral Imager (HSI) payload designed to detect Harmful Algal Blooms (HABs) from a Low Earth Orbit (LEO). HYPSO-1 was their first satellite and the team did not have any previous experience in satellite operations. Operating a satellite is time-consuming, labor-intensive, and requires special knowledge and qualifications [13]. In a small, resource-limited team like the HYPSO team, it is therefore essential to make tools to ease operational challenges in order to maximize the utilization of the satellite. Additionally, it is desired to work asynchronously, meaning having the ability to troubleshoot or upload commands without having to wait for the satellite to pass over a ground station. By

working asynchronously the flexibility of the team increases.

CubeSats are small, inexpensive, and have a shorter development time compared to more traditional, monolithic satellites [14]. CubeSats are therefore often used by academic institutions, like the HYPSO-1 satellite at NTNU [14]. Since the satellites are so small, they have optical limitations when equipped with an imaging payload [14]. The camera settings can be adjusted, and are therefore a trade-off between resolution, coverage, revisit time, and swath width. The HyperSpectral Imager (HSI) on HYPSO-1 can have a higher spectral resolution than traditional satellites like Sentinel, but not with the same coverage. HYPSO-1, being a 6U CubeSat, can therefore not cover the same areas as traditional satellites, but its utilization can be more agile. The HYPSO-1 satellite was launched to a Sun Synchronous Orbit with a mean altitude of about 540km with the ability to capture off-nadir images (internal information), and therefore covers the entire earth within 24 hours. The satellite can image specific areas more frequently than more traditional satellites that serve a wide range of users. In order to utilize the potential responsiveness of the HYPSO-1 satellite, there is a need for tools that enable members to quickly, and easily, operate the satellite to capture images of specific targets. The concept of operations is shown in Figure 1.1.

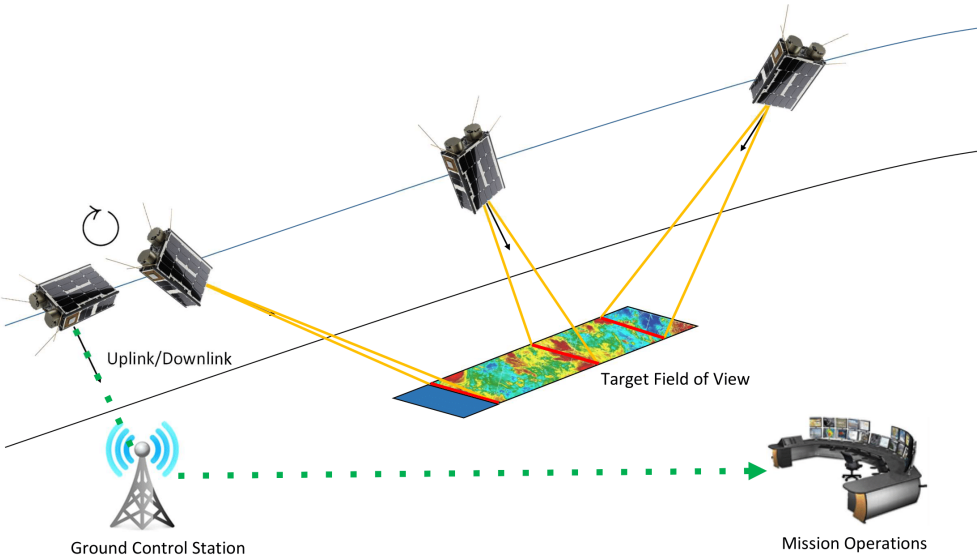


Figure 1.1: Concept of operations for the HYPSO-1 satellite. The figure is an alteration of a figure made by Mariusz E. Grøtte in [15]

This thesis starts by introducing the stakeholders, relevant tools, and theory used throughout the thesis. Afterward, the methods and workflow are described. Then the results and experiences are provided, before ending with a discussion and conclusion.

1.2 Problem Statement & Research Questions

Automating satellite operations can increase the responsiveness of captures and overall data throughput from a satellite. Additionally, automation can lower the knowledge needed to operate the satellite, thus making it more accessible. However, the payload is what makes satellites unique, and all satellites are equipped with at least one mission-specific payload. Therefore, satellites have different operational patterns and requirements, thus implying a need for mission-specific tools. The HYPSO team launched their first satellite, HYPSO-1, on the 13^{th} of January 2022. This thesis investigates and implements tools to simplify operations in order to maximize the utilization of the HYPSO-1 satellite payload. This objective has been divided into three research questions.

- **RQ1**: *To what degree is it possible to work with operations of HYPSO-1 outside of passes over a ground station?*
  Having to wait for the satellite to be in the range of a Ground Station (GS) takes up a lot of time and causes stressful situations if one has a set of tasks to perform within the satellite pass. Being able to schedule such commands asynchronously to the satellite passes is desired.
- **RQ2**: *To what degree is it possible to reduce the number of imaging capture errors whilst also reducing the amount of testing on the ground?*
  Capture scripts can have different types of errors. They can have errors in the spacecraft scripts i.e., the wrong name, incorrect timing, or typos in commands. The capture scripts can have erroneous pointing parameters due to human error when making the parameters. Capture scripts can also be uploaded to the wrong script engine due to human error.
- **RQ3**: *How can the project organization best facilitate the operator training?*
  An academic CubeSat team consist of members from a variety of study fields and commonly has a big turnover of personnel. In academic teams, members also have their own research to tend to and cannot focus solely on operating the satellite. Members have different levels of understanding of the systems to use. Facilitating training of operators can therefore be challenging.

Chapter 2

Background

This section will describe the background of this work. It will start by describing the stakeholders and system design in the HYPSO project, then the applicable theory will be presented.

2.1 Stakeholders

HYPSO-1 is a 6U CubeSat consisting of a satellite bus provided by NanoAvionics (NA), and a payload developed at Norwegian University of Science and Technology (NTNU).

2.1.1 NTNU

HYPSO-1 is the first satellite made within the HYPer-spectral Smallsat for Ocean observation (HYPSO) project. HYPSO is a part of the Mission-oriented autonomous systems with small satellites for maritime sensing, surveillance and communication (MASSIVE) project, which is associated to the Centre for Autonomous Marine Operations and Systems (AMOS) [16] [17]. The HYPSO-1 satellite is a 6U CubeSat equipped with a novel pushbroom HyperSpectral Imager (HSI) as its main payload [18], called Colored Littoral Zone and Algae Watcher (CLAW). The payload was custom built by the HYPSO team, which is a part of the SmallSat Laboratory (SSL) at NTNU [18]. The payload consists of the On-Board Processing Unit (OPU), a Red-Green-Blue (RGB) camera, and the HSI. The SSL consists of about 20 students, ranging from Bachelor's to Post-doc level, with an ever-evolving mixture of skills and study disciplines [19]. Collectively, the student-driven SSL has been able to develop, test, qualify, and integrate the HSI payload into a 6U CubeSat platform provided by NA, and have a successfull launch of the HYPSO-1 satellite on the 13^{th} of January 2022 [18] [19].

As part of the MASSIVE project, the HYPSO-1 satellite was part of a bigger infrastructure for maritime surveillance. Together with Autnonomous Underwater Vehicles (AUVs), Unmanned Surface Vehicles (USVs), and Unmanned Aerial

Vehicles (UAVs), the HYPSO-1 satellite forms what has been called *the observational pyramid*, shown in Figure 2.1. The HYPSO-1 satellite gives the best overview, while aerial-, surface-, and underwater vehicles can get more accurate measurements and can operate below cloud cover [18].

Satellite Operators

The HYPSO team developed the payload but also operated the satellite. Satellite operators were the users of the operational tools and training methods developed in the thesis. The operators had no previous experience in satellite operations, came from different study fields, and had varying levels of knowledge about the satellite platform and ground systems.

End Users

The end users of the satellite data are primarily separated into two categories. Category one consists of oceanographers who want raw data. Category two consists of aquaculture groups, i.e., Grieg Seafood and SalMar. Category two prefers processed data. Other groups interested in processed data are i.e., the environmental directorate. End users are not constrained to be in Norway.

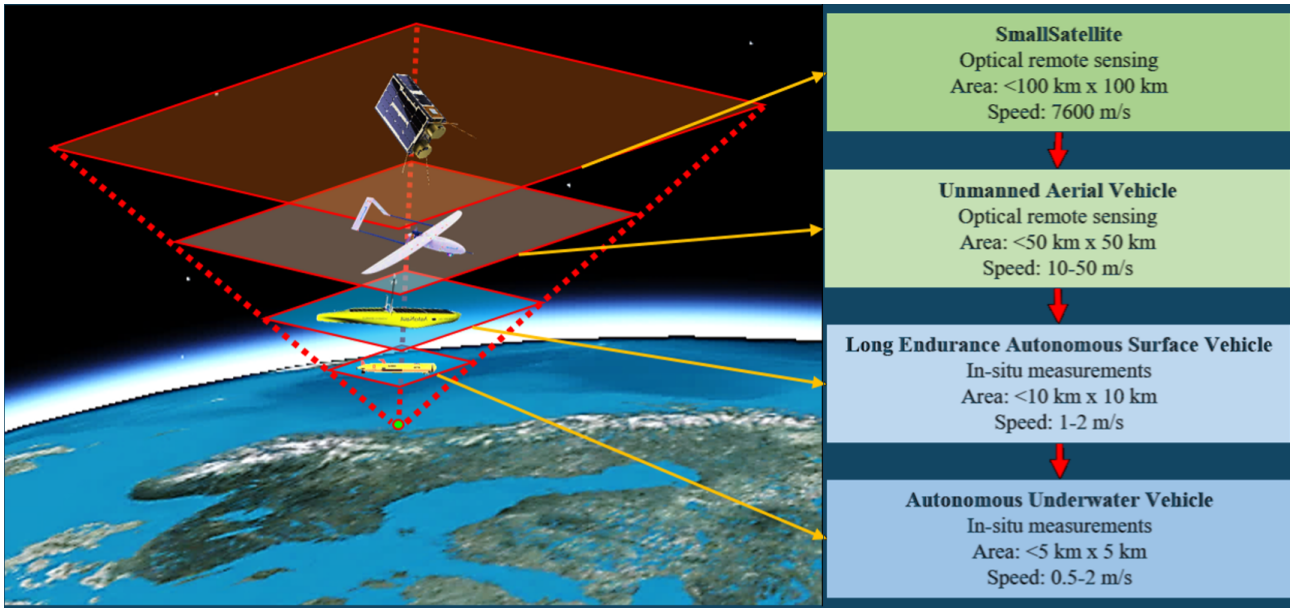


Figure 2.1: The observational pyramid showing the different autonomous agents used in the MASSIVE project. The figure is obtained from [18], and is an updated version of a figure originating from Tor Arne Johansen through the MASSIVE project.

2.1.2 NanoAvionics

NanoAvionics (NA) is a Lithuanian[1] small satellite mission integrator providing satellite platforms and subsystems [20]. One of the satellite platforms they provide is the 6U nanosatellite bus, called Multi Purpose 6 (M6P) [21]. The M6P platform is used by two other satellites launched before the HYPSO-1 satellite [21]. The M6P platform includes a Payload Controller (PC), which was the interface between the satellite payload and the M6P bus. As mission integrator, NA provided launch integration, payload data storage, and ground station software called MCS for the HYPSO-1 mission.

2.2 Tools and System Design

This section will introduce tools and applicable modelling of the tools and systems surrounding the HYPSO-1 satellite, including the relevant ground systems.

2.2.1 CubeSat Space Protocol

The CSP protocol was used for communication in the HYPSO-1 satellite. The CubeSat Space Protocol (CSP) is a network-layer delivery protocol designed for CubeSats originating from the university of Aalborg [22]. The main developer of the protocol started working for GOMSpace, which used it in their products [22]. The CSP protocol was made to ease communication between distributed embedded systems by introducing a service-oriented network topology [22]. Since the communication bus itself is the interface to other subsystems, the developers only need to define a service contract and set of port-numbers that the subsystem will be responding on [22]. GOMSpace released a basic implementation of the CSP functionality as a software library under GNU Lesser General Public Licence (LGPL), which allowed the implementation to be copied and used by the public [23].

2.2.2 Mission Control Software

The Mission Control Software (MCS) is ground station software provided by NA that connects the different ground services as shown in Figure 2.2. MCS runs on the Amazon Web Services (AWS) and includes a task scheduler, `nanoMCS`, pass scheduler service, File Transfer Manager (FTM), data storage, and Grafana. MCS is also connected to SatLab's and Kongsberg Satellite Services (KSAT)'s pass schedulers for ground station booking and operations. The operator can use the remote connection to get access to all parts of MCS, use the direct access through NNG to get a direct connection to the S-band modem or UHF radio, or use a RESTful Application Programming Interface (API) to access the data storage. By connecting directly to the S-band modem, the operator can communicate with the satellite without the use of the subsystems that are a part of MCS.

[1] Recently acquired by Kongsberg Defence and Aerospace

The task scheduler can schedule `nanoMCS` tasks, further described in subsection 2.2.3, or upload/download tasks using the File Transfer Manager (FTM). A task is a set of commands to be executed on the satellite during a pass, i.e., change configurations or download/upload files. The pass scheduler service communicates with SatLab and KSAT pass scheduler to schedule satellite passes over the different ground stations. The data storage is where all the data from the satellite is stored, and Grafana is a Graphical User Interface (GUI) for visualizing the telemetry data. The operator can connect the NTNU Ground Station (GS) directly without using MCS as well. The operator can also connect to KSAT's pass booker and administration portal. In short, MCS allows operators to communicate seamlessly with the satellite by connecting all of the services running on ground. (Internal information).

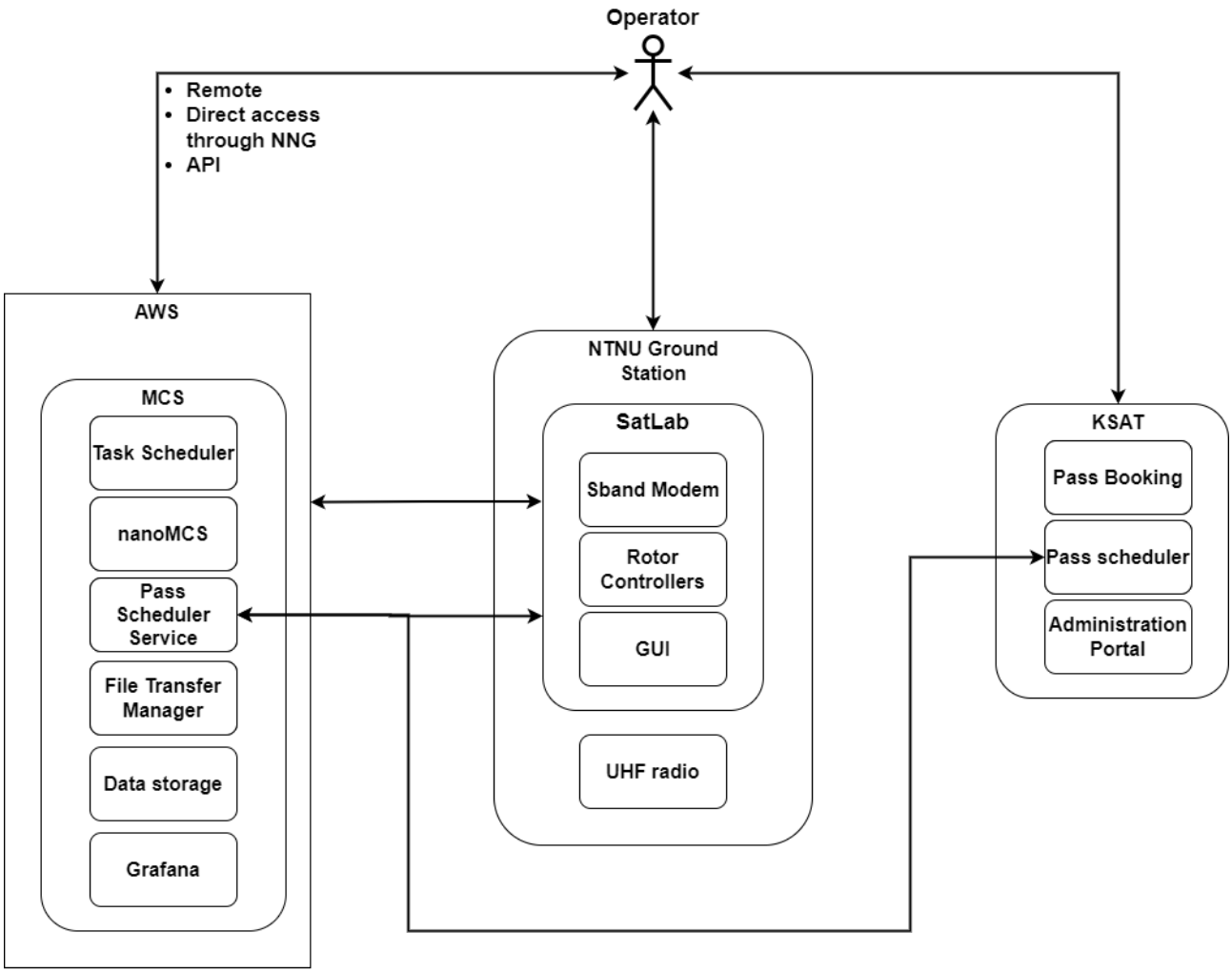


Figure 2.2: Overview of ground services. MCS is running on the AWS server and communicates with SatLab and the pass scheduler at KSAT. The operator can utilize MCS through a remote connection, a RESTful API, or connect to the S-band modem directly using NNG.

2.2.3 Command Line Interfaces

The HYPSO-1 satellite uses an implementation of the CSP layer on top of a Controller Area Network (CAN) bus, which is the communication interface between the HYPSO-1 payload, CLAW, and the M6P satellite platform. `nanoMCS` and `hypso-cli` are tools that translate human-readable commands to CSP commands that give access to the shell commands of different subsystems.

`nanoMCS` is software provided by NA that is as a Command Line Interface (CLI), and functions as an interface to the satellite subsystems. `nanoMCS` translates

human-readable commands into low-level commands for the satellite subsystem, which contains a toolset of functions used in mission control [24]. With `nanoMCS` one can log into into the shell of subsystems produced by NA. To log into the shell of subsystems is called remote shell in the rest of this thesis. Furthermore, MCS supports setting up `nanoMCS` tasks with the task scheduler. All subsystems of the M6P bus provided by NA supports CSP, but by using `nanoMCS` one can remote shell into them and use human-readable version of the commands [24].

`hypso-cli` is a CLI developed by the HYPSO team at the SSL that fulfills many of the same functions as `nanoMCS`, but is tailored to direct payload operations. The CLI allows users to communicate with human-readable commands with the payload. The OPU supports CSP, and can therefore communicate with the M6P satellite bus. Using `hypso-cli`, the operator can use high-level functions with human-readable input instead of communicating directly through CSP. `hypso-cli` also provide access to the shell of all subsystems on the M6P platform but is not connected to the task scheduler in MCS [25].

2.2.4 Spacecraft Subsystems

The spacecraft consists of the M6P satellite platform provided by NA with a payload developed by NTNU. The satellite platform is shown in Figure 2.3. As described in subsection 2.1.1, the payload consists of the OPU, the RGB camera, and the HSI. There are different ways of partitioning a spacecraft into subsystems [26]. The one used in this project divided the satellite platform consisting of seven main subsystems [25]:

- The Flight Computer (FC)
- The Payload Controller (PC)
- The Electrical Power Subsystem (EPS)
- The Thermal Subsystem
- The Attitude Determination and Control Subsystem (ADCS)
- The Communication Subsystem
- The Structural Subsystem

The FC is the main controller of the satellite. The PC is the interface to the payload. The EPS includes the solar panels and provides the whole satellite with power. The thermal subsystem distributes heat energy and heats up the battery pack. The thermal subsystem has passive, structural parts and active heaters. The ADCS determines and controls the attitude of the satellite. The subsystem includes Reaction Wheelss (RWs), Magnetorquers (MTQs), a star tracker, sun sensors, and Inertial Measurement Units (IMUs). The communication subsystem consists of two Ultra High Frequency (UHF) radios, an S-band transceiver, and antennas. The structural subsystem is the satellite frame providing structural supports.

Compared to the partitioning proposed in [26], the on-board processing or command and data handling subsystem is covered by the PC and FC. The tele-

metry, tracking and command subsystem is covered by the EPS, FC, and communication subsystem. The M6P platform did not include any propulsion subsystem, thus also not including a position and orbit determination and control subsystem.

For operations of the satellite, the FC is used to control the satellite platform, and the PC is used to control the payload.

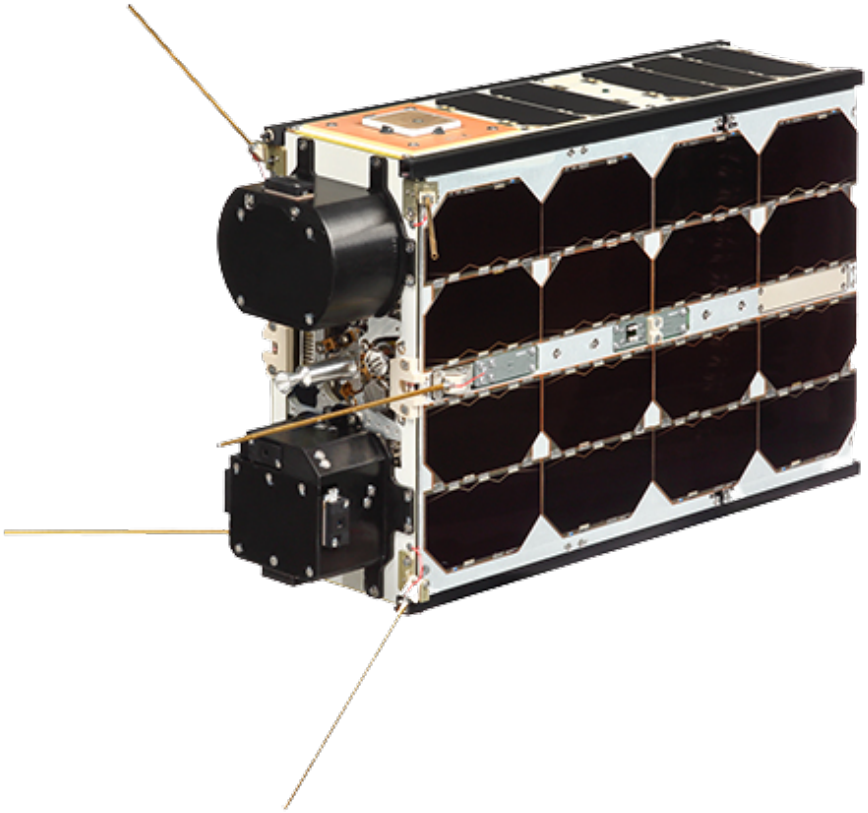


Figure 2.3: M6P Satellite platform. Figure obtained from [21].

2.2.5 Spacecraft Scripting

The PC and FC subsystems on the M6P platform are equipped with two independent script engines each. These script engines can run script files consisting of subsystem shell commands. The script files are text files with MS Windows style line terminators (CRLF). The FC script engine runs in the shell of the FC and vice versa for the PC. Therefore, the FC script engines have access to the human-readable version of commands on the FC only. So to access/send commands to

other subsystems, the shell commands must be sent as raw CSP messages, using the `csp txrx` function.

The subsequent lines/commands in the scripts are run sequentially from start to end. There are two different ways of triggering or delaying a command. The first way is to delay until a specific unix timestamp, and the second method waits for a defined amount of milliseconds.

The `csp txrx` command is defined with a delay that can be set up to 3000 ms and the script engine waits until an acknowledge is received, or until the specified delay has passed, before going to the next line in the script file. The script engines therefore allow for automatic control of the satellite when the satellite is not in range of a GS, but all timing between commands needs to be calculated on ground. (Internal information).

2.2.6 Mirror on Ground

The LidSat is an assembly of selected engineering models of satellite sub-systems on ground. The LidSat is a FlatSat, but is named LidSat because it is placed on a lid of an Electrostatic Discharge (ESD)-box for more tidy mounting. The LidSat is connected to the FlatSat set up in NA's laboratory in Lithuania through ethernet over a Virtual Private Network (VPN) tunnel. The LidSat consists of the payload, FC, PC, Ultra High Frequency (UHF) radio, and two Electrical Power Subsystems (EPSs). One EPS is at NTNU and the other one is in Lithuania. The LidSat has been used for testing on target hardware during development. The LidSat setup with CSP IDs are shown in Figure 2.4. The OPU is the payload. The RGB and HSI is connected to the OPU. The secondary EPS is not depicted since it is not a part of the satellite but needs to be connected in order to power the FC. The Software Defined Radio (SDR) is part of the HYPSO-2 satellite and is therefore added in the figure, but not relevant in this thesis.

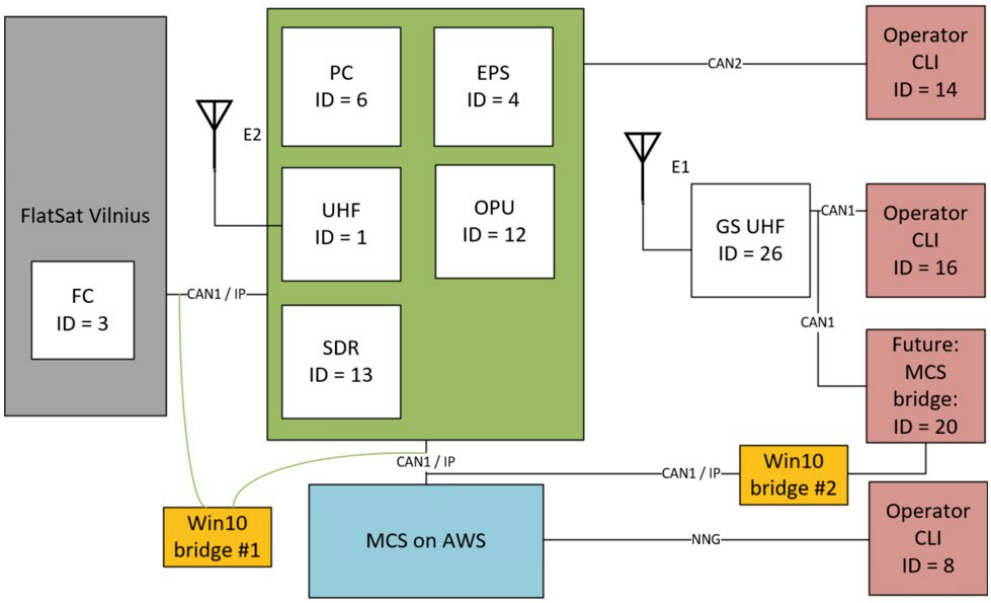


Figure 2.4: The LidSat setup with CSP IDs. Figure is obtained from [27]

2.3 Systems Engineering and Development

The thesis was done as part of the HYPSO team which consisted of students coming from a variety of different fields of study, as described in subsection 2.1.1. In order to develop and collaborate efficiently in such a team, there was a need to establish workflows and training procedures. It was also important to understand the phase of the project as a whole.

2.3.1 Project Life Cycle

In order to effectively improve the operations of the HYPSO-1 satellite, there is a need to understand the current stage of the project. There are multiple ways of modeling the project life cycle. Organizations such as NASA and ESA have made their own standards and implementations. [26] shows the NASA project life cycle, but in this thesis, ESA's representation in European Cooperation for Space Standardization (ECSS) was used. The ECSS is a European collaboration to develop a coherent, single set of user-friendly standards for use in European space activities [28]. These standards include a description of the project life cycle of satellite missions divided into different phases. Phase 0, phase A and phase B contain mission definition and preliminary design, phases C and D span the detailed design, production, and qualification, phase E considers the utilization, and phase F consists of the disposal [29]. Phase E comprises reviews prior to launch, commissioning, utilization, maintenance of orbital elements, and maintenance of the associated ground segment [29]. The major tasks of this phase vary widely depending on the

type of project concerned [29]. General tasks are: preparing space and ground segments for launch, launch, and early orbital operations, commissioning, in-orbit operations to achieve the mission objectives, ground segment activities to support the mission, and finalizing the disposal plan [29]. Associated reviews are Flight Readiness Review, Launch Readiness Review, Commisioning Results Review, and End-of-Life Review [29]. The project life cycle is defined to assist project managers with controlling the cost, schedule, and technical objective [29]. At the same time, the ECSS is generalized, thus leading to tailoring for the specific mission being necessary [30].

2.3.2 Agile Software Development

The software development used in HYPSO is an agile approach. Agile software development is a conceptual framework that starts with a planning phase and utilizes an iterative and incremental approach to reach the deployment phase [31]. The overall objective of the agile framework is to reduce overhead and increase the adaptability to changes without excessive rework, even late in the development [31]. The agile software development manifesto was developed in 2001 [32], and consists of four principles:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

These four principles have proven themselves to be valuable in a competitive environment with high degrees of uncertainty often associated with low-Technology Readiness Level (TRL) components [32]. The agile software development framework has inspired different trade-offs of these four principles leading to different metodologys [4]. Adopting these metodologys has proven to increase productivity and quality [33].

Extreme Programming

Extreme Programming (XP) is one of the most popular approach to agile software development [4]. XP is based on four values [4]:

1. Communication: Software development is all about solving the problem of the user [34]. The users are therefore part of the project team such that everyone share the same view of the system [4].
2. Feedback: Continuous feedback from previous work help identify areas for improvement [34]. Feedback can be from the system it itself or other project members [4].
3. Simplicity: Features are not implemented before they are needed to help developers concentrate [33]. It promotes working on current requirements and not predicting the future [34]. The code is also preferred to be simple [4].

4. Courage: In case of something not working, the previous principles should be kept in mind for the results to not harm the team [34].

According to [34], minimum accompanying measures are required in XP, meaning creating documentation and project requirements are not required. However, [4] and [33] rather claim that XP allows for flexible requirements through frequent change. The metodology life cycle is presented in Figure 2.5.

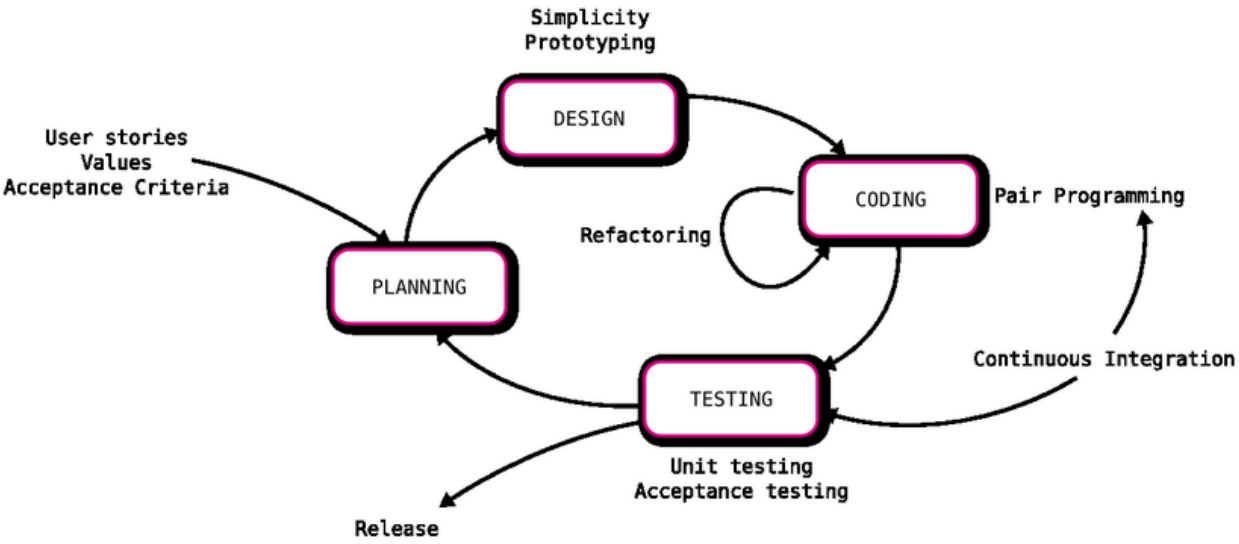


Figure 2.5: Extreme Programming life cycle. Figure obtained from [4]

The HYPSO Software Development Method

The HYPSO team used an agile approach to software development. They arranged weekly software meetings and bi-weekly sprints. During the sprint, the team reviewed the past sprint and planned the new sprint. To keep track of the sprint, the team used a KanBan board on GitHub. Bugs and feature requests were submitted as GitHub issues and graded based on how much time the team thought it would take to solve. For further description of the sprint process, the reader is directed to [18].

The software meetings were meant to plan software releases, ask questions and update everyone on what members were working on and if they had any problems. The team also included updating the KanBan board used in the sprint. The meetings were arranged to be updated on milestones and the current stage of development. The meetings were allocated time to get help, give help, and provide input on current development goals, milestones, and development methodology. For further description of the software meetings and how the software methodology of the HYPSO team was formed, the reader is directed to [18].

2.3.3 System Design Evaluation

System design evaluation is the inner compass of system design [35]. System design requires integration and iteration coordinated by synthesis, analysis, and evaluation between all steps [35]. The analysis is performed to quantify the performance and is essential for the evaluation process [26]. Using design-dependent parameters like reliability, maintainability, and sustainability, analysis, the evaluation of a system design is made possible [35]. These design-dependent parameters are not necessarily the same as the performance parameters defined in [26]. A representation of the synthesis, analysis, and evaluation cycle is shown in Figure 2.6.

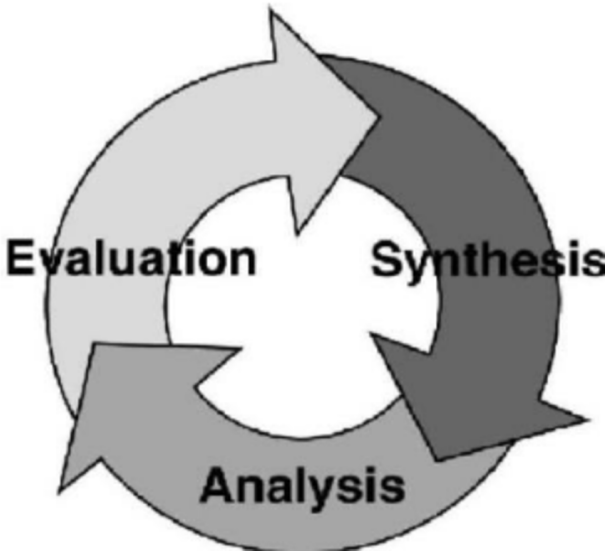


Figure 2.6: Synthesis, analysis, and evaluation cycle. Figure obtained from [35]

[35] defined the three terms and how they are best used. Synthesis is the process of building the system by using creativity to put known things together into a new and more useful combination. The analysis concerns the function of estimating and predicting values for design-dependent parameters and design-independent parameters. [35] stated that system analysis and operations research is a part of the system evaluation, but insufficient. Evaluation requires alternative designs to be compared to each other and their compliance with the user requirements. It is the design-dependent parameters that differ between the alternatives, and the user should make the final decision [35].

2.3.4 Learning Styles

Due to the many different backgrounds the satellite operators had, it was looked into theory about learning styles. In 1984, David A. Kolb published his work on experimental learning styles, including Kolb's learning cycle [36]. The learning

cycle consists of four steps: concrete experience, reflective observation, abstract conceptualization, and active experimentation [37]. The learning cycle assists with transforming experience into usable knowledge. Within the same publication, Kolb discussed four different learning styles [37]. The authors of [37] adapted the learning cycle to also include the learning styles, resulting in Figure 2.7. Each learning style occupies a quadrant in the learning cycle[37]. The learning styles are a combination of the adjacent stages and are a function of circumstances and preferences [37]. In [38], Kolb found that the pattern associated with the four learning styles comes from the transaction between people and their environment at five different levels: personality, education, specialization, professional career, current job role, and adaptive competencies [38]. The divergent learning style is associated with valuing skills like helping others and sense-making through concrete experience and reflective observations [37] [38]. The assimilative learning style combines reflective observation and abstract conceptualization [37], and is associated with thinking skills [38]. Assimilative learners are good at information-gathering, analysis, and theory building [38]. The convergent learning style combines abstract conceptualization and active experimentation and is associated with decision skills like the use of technology and goal-setting [38] [37]. Lastly, the accommodative learning style combines active experimentation and concrete experience and is associated with practical, and intuitive problem-solving [37]. Individuals often thrive in leadership and taking initiatives [38].

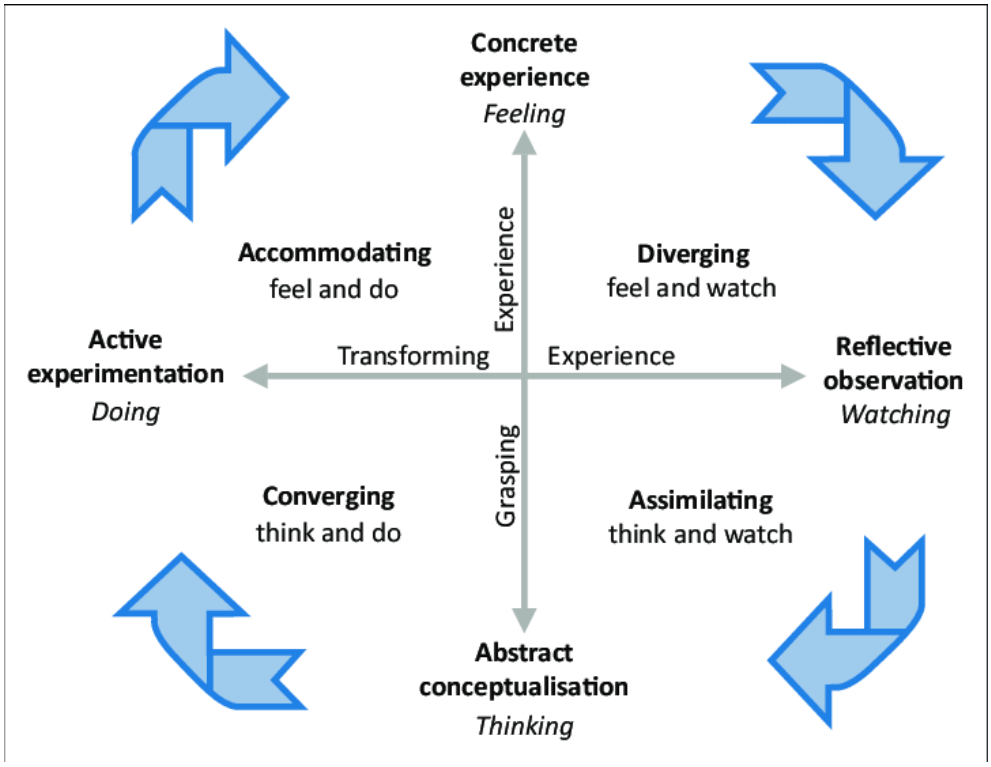


Figure 2.7: Kolb's learning cycle and learning styles. Figure obtained from [37]

2.4 Existing Tools and Team Structures in Satellite Operations

Since satellite operation is demanding there have previously been developed team structures and tools to simplify and automate processes.

2.4.1 Selection of Tools

To combat the resource-consuming satellite operations, NASA JPL have developed tools called generic inferential executor (GENIE), Generic Spacecraft Analyst Assistant (GenSAA), and Automated Scheduling and Planning Environment (ASPEN). GENIE is a pass automation tool that mimics normal operators and graphically displays the progress during a pass [39]. GenSAA that informs operators about potential issues with the satellite as well as probable causes [40]. ASPEN is a tool that provides low-level commands from high-level input from operators [40]. GENIE and GenSAA automate passes, assist in troubleshooting the satellite, and helps monitor the health of the satellite. ASPEN reduce the knowledge base the operation team must have by replacing the role of subsystem specialists. Together, these tools allow for downsizing and changing the knowledge base needed for operations by not needing to have subsystem specialists available for nominal

operations. However, when put in use the operators, which often did not have a software background, could rarely figure out why it stopped working when it ran into problems [8].

2.4.2 The IntelliSTAR Team Structure

The structure of HYPSO's operational structure was inspired from the Intelligent Satellite Technology Automated Reasource (IntelliSTAR) structure. In [41], Thomas P. Gathmann and Linas Raslavicius look at the full operations domain and have proposed the IntelliSTAR architecture. The IntelliSTAR architecture divides the satellite operations domain into four distinct classes: **planner**, **controller**, **subsystem specialist**, and **analyst** [41]. The **planners** develop and maintain a plan to satisfy the overall objectives, the **controllers** coordinate activities and follow the execution, the **subsystem specialists** manage and maintain specific subsystems, and the **analyst** assess the degree of success or failure after a mission [41]. Together, these classes cooperate to develop missions and schedules, follow execution, and assess the results [41].

Chapter 3

Methods and Workflow

This chapter will describe what method and workflow were used, why they were chosen, and adaptations throughout the project period.

3.1 Development Workflow

The HYPSO team did not have any previously existing tools for payload and satellite operation made for automating the payload utilization of the HYPSO-1 satellite. Additionally, the team did not have any previous experience in operating the satellite. Features, implementation, and integration to the satellite utilization were therefore uncertain. It was natural to go for an agile approach to development due to the uncertainty in the use cases for operations. As described in subsection 2.3.2, an agile approach provides high adaptability to changing requirements. The users of the tools were the operators. Since the author was an operator during the project period, the author had good communication with the users of the tools, which is required in an agile approach. Because of the small size of the HYPSO team, contacting project members was easy. Instead of defining how the supporting operational tools were to be designed at an early stage, it was rather chosen to solve problems as they occurred. This aligns with the Extreme Programming (XP) metodology described in section 2.3.2. The chosen methodology was an adapted XP methodology.

The XP metodology allows for rapid changes to the requirements and does not include new features before they are needed. The user of the tools to be developed is the operators part of the HYPSO team. By focusing on solving the issues the users are currently experiencing, it becomes easy to focus on the correct parts of development. Feedback from the users also locates new areas to improve. As described in section 2.3.2, the XP metodology is often used on low-TRL products, something that corresponds well with the supporting operational tools set to be developed. However, the development of these tools are primarily done by the author. It is therefore not possible to follow the XP metodology exactly since the pair-programming and in-depth code reviews are not possible. The development

method used is therefore an adaptation of XP where pair-programming and code reviews are exchanged with testing on the LidSat and letting other members test the tools.

The users were part of the evaluation process in the development. Features and issues to be resolved were communicated to the author. Multiple alternatives were proposed, then evaluated by the author in collaboration with other team members before the author implemented the chosen alternative. The design-dependent parameters were mainly maintainability and development time since the resources were limited. This process has similarities with the system design evaluation shown in subsection 2.3.3. With inspiration from both XP and the system design evaluation cycle, the chosen development workflow was therefore an adaptation of these. Mainly, the method was iterative, allowed for rapidly changing requirements, and focused on swift deployment. The workflow chosen is illustrated in Figure 3.1. Features to implement and issues to correct were found by the author or other parts of the team. The team adopted an ad-hoc approach to testing due to a lack of resources and time. After changes were implemented, test scripts were tested on the LidSat. If errors were found during testing or after deployment, it went back to requested changes. The metodology valued rapid deployment over reliability. The tests on LidSat only proved the sequence was functional but did not contain tests for all the parameters that affect the satellite in orbit such as pointing, position, and timing of the capture. Instead of testing for all such possibilities, circumstantial errors were fixed when they were found.

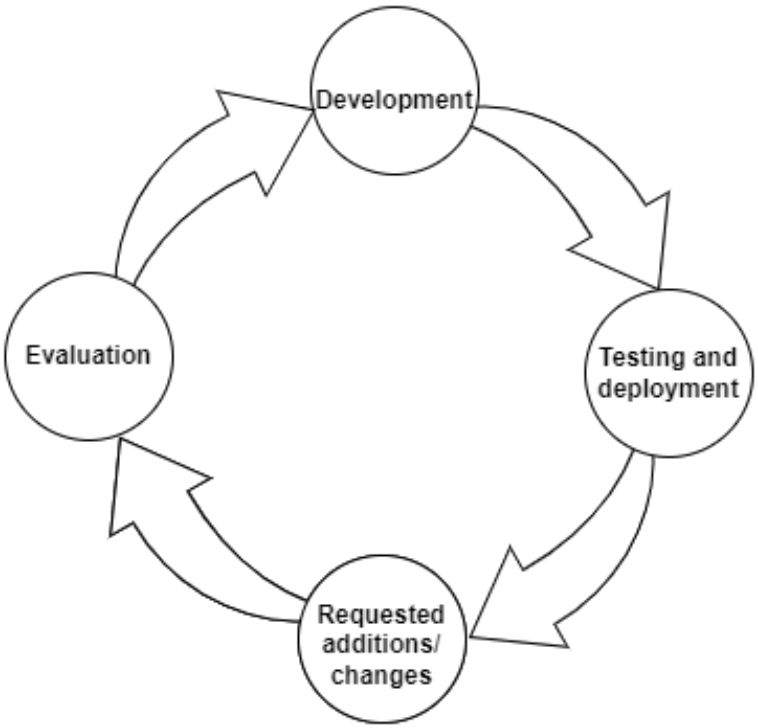


Figure 3.1: Development workflow used in this thesis.

3.2 Satellite Operator Team Structure

The HYPSO team organized their operations team with inspiration from the IntelliSTAR architecture. The team was divided into three categories: **mission planner**, **operator**, and **data analyst**. Multiple members had several of these roles i.e., all **operators** were also trained as **mission planners**. The **mission planners**' role was to generate capture scripts, meaning FC and PC scripts, and coordinate them by merging them all into one master script to be uploaded. The **operators**' role was to upload the capture scripts, monitor the health of the satellite through telemetry data, pull the payload data downloaded from the satellite, and monitor the execution of the scripts. The **data analysts**' role was to process the downloaded captures and provide feedback on capture settings.

These classes do not cover the entirety of the classes described in the IntelliSTAR architecture. Instead of having a specific class responsible for the planning of all objectives and targets, everyone was able to request specific targets through the GitHub issue templates described in subsection 4.2.3. Coordination of objectives for the week was decided upon at a weekly satellite operations meeting with the entire operations team. The **analyst** role specified by the IntelliSTAR architecture was filled by the team as a whole. Additionally, after capture campaigns, the **analyst** role was filled by members from all parts of the observational pyr-

amid described in subsection 2.1.1. The **subsystem specialist** role was partially covered by understanding the payload capabilities of the **mission planners** and **data analysts** when setting capture configurations. However, making the script compatible with the script generator was mostly the author's responsibility. If a problem was found, the team coordinated themselves to set up ad-hoc task forces for debugging if needed.

3.3 Training of Operators

Since the HYPSO team is a student team with high turnover, the members come from a variety of different backgrounds and fields of study with no previous experience in operating a satellite. It is therefore challenging to make suitable training programs for all new operators. The operational workflow is ever-changing in the beginning. The chosen approach is therefore to make a document describing the high-level components of the HYPSO-1 satellite in order to give members a general understanding of the system to be operated. Then, new operators watch as the experienced operator works and explained what they are doing. After some time, the new operator tries the work themselves with support from an experienced operator. The training of operators is therefore thought to follow the stages shown in Figure 3.2. As an independent operator, the person is expected to be capable of searching through documentation independently but that does not mean the person cannot ask other operators for help. If other adaptations are necessary, they will be added iteratively.

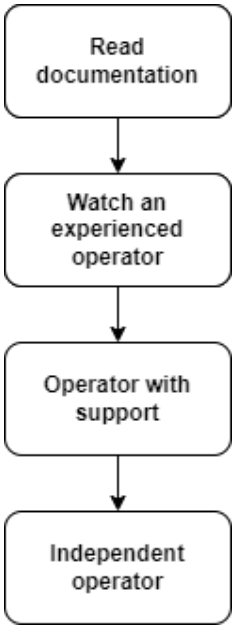


Figure 3.2: Stages of operator training

Chapter 4

HYPSO-1 Launch & Commissioning

Design is an iterative process. The necessary number of iterations is one more than the number you have currently done. This is true at any point in time.

Dave Akin

HYPSO-1 was launched on the 13th of January 2022, and the first contact with the satellite was made on the 14th of January 2022. This chapter describes the first two months in the orbital life of HYPSO-1, focusing on the early operations and the beginning of the commissioning of the satellite. This chapter will not go deeply into the commissioning performed by NanoAvionics (NA), it will rather be mentioned what was being done if applicable, and focus on the work performed by the HYPer-spectral Smallsat for Ocean observation (HYPSO) team. During this point in time, the complete focus was put on checking that all parts of the satellite were functional and learning how to operate it. A timeline showing the different stages is shown in Figure 4.1.

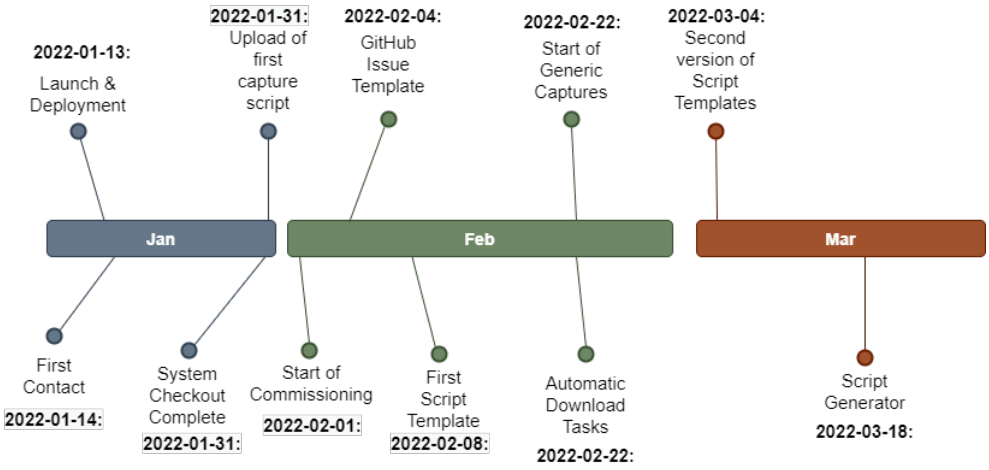


Figure 4.1: Timeline of the first three months after launch

4.1 System Checkout

The system checkout is marked in the timeline by the yellow arrow in Figure 4.2.

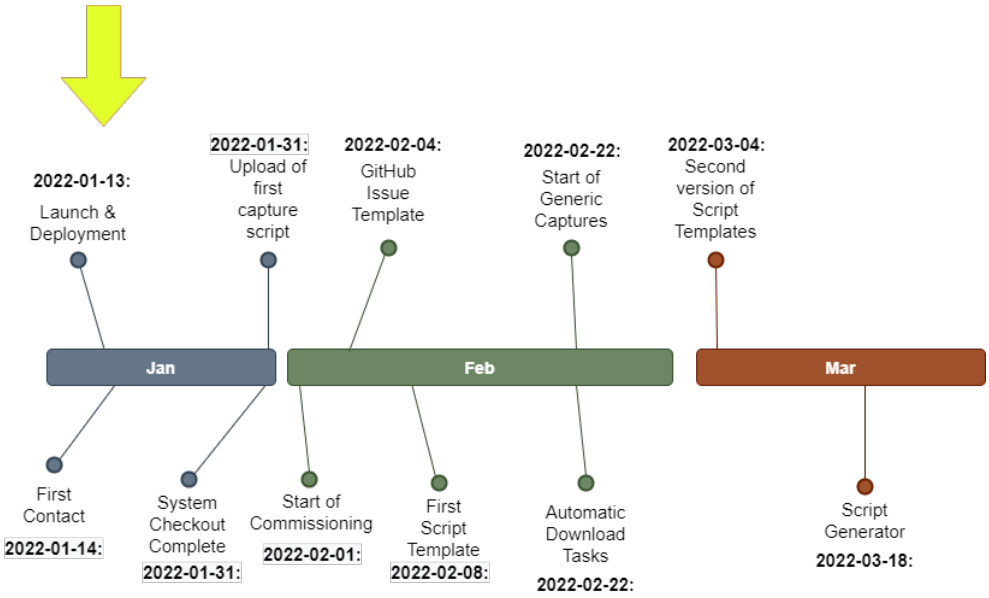


Figure 4.2: Timeline showing the start of system checkout with the yellow arrow.

The commissioning phase is intended to verify that all systems work as intended and optimize configurations [42]. Since the HYPSO team acquired the satellite platform from NA, they performed system checkout and commissioning of the satellite bus. NA's early operations commenced by getting contact with the satellite and pinging subsystems [43]. Then NA downloaded telemetry and

updated the Electrical Power Subsystem (EPS) and Ultra High Frequency (UHF) configurations [43]. After configurations were verified, NA initiated S-band communication, set the satellite in pointing mode, uploaded Two Line Element (TLE), verified the Attitude Determination and Control Subsystem (ADCS) and Payload Controller (PC), before ending with uploading an end-of-LEOP configuration to all subsystems [43]. After NA had performed successful system checkout on the satellite platform, the On-Board Processing Unit (OPU) of the payload was powered on for the HYPSO team to perform their system checkout on the 28th of January. The Red-Green-Blue (RGB) camera and HyperSpectral Imager (HSI) were powered on for system checkout on the 31st of January 2022. All system checkout was successful, leading into the next phase of fully commissioning the subsystems, and learning to operate the satellite.

4.2 Development of the Script Generator

In preparation for the commissioning, the HYPSO team made an Mission Operations Plan (MOP). The commissioning phase was not purely driven by the MOP since the MOP included testing of features that were not possible to achieve until several months later. This section will therefore cover the commissioning as a timeline and show how the HYPSO team learned to operate the satellite and utilize their payload. Commissioning of the satellite platform was gradual and went on until the official handover meeting on the 27th of April while commissioning of the payload was not achieved within the thesis period. The following subsections will follow the timeline starting from the yellow arrow shown in Figure 4.3.

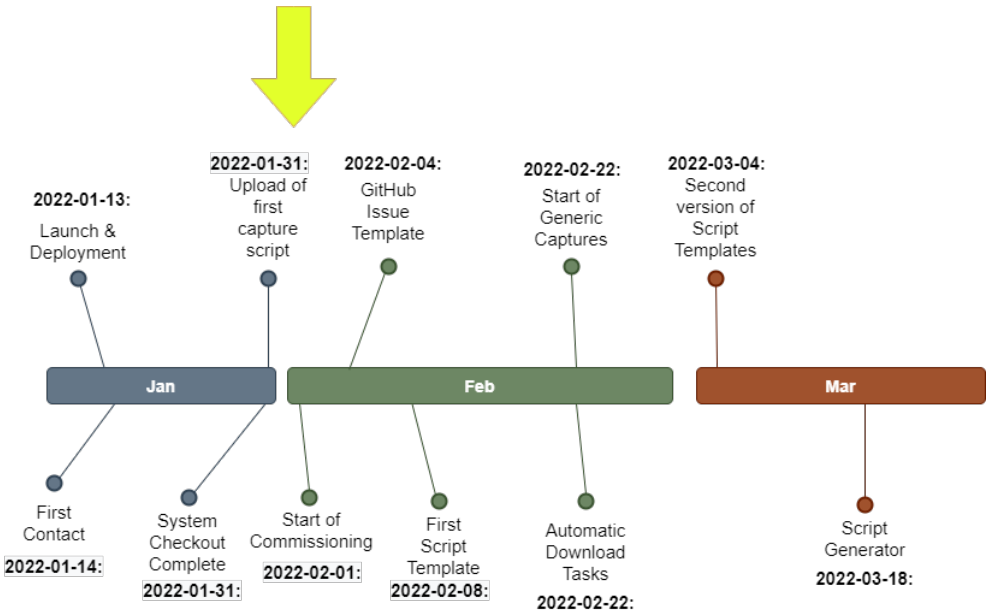


Figure 4.3: Timeline with a yellow arrow showing the start of the commissioning phase

4.2.1 Generation of the Early Capture Scripts

The first capture scripts were made at a time when there was a lot of uncertainty about how to use the platform. The HYPSO team did not know how to use MCS and was not comfortable with using `nanoMCS`. Only a few members had used `nanoMCS` and tried making a few simple script engine scripts prior to launch. At this point in time, the pointing of the satellite was stable, but the HYPSO team was not sure about how to use the ADCS subsystem. Additionally, attempts at making pointing scripts were impossible to test on the ground [1] and there had been made design changes to the configurations on the FC to control the ADCS shortly before launch. At the same time, the payload commissioning was at its very beginning, so the accuracy of targets was not a major concern since the team focused on finding suitable camera configurations. The target of captures was therefore nadir passes over land with as many features as possible. Since the functionalities of the OPU, RGB camera, and HSI were verified during the system checkout, the team focused on the objective of finding suitable exposure times for both the HSI and RGB camera.

The first Payload Controller (PC) and Flight Computer (FC) scripts, described in subsection 2.2.5, were made using a tool the author has developed. The tool parsed through the log generated from an instance of running `hypso-cli` and

[1] The LidSat was not equipped with an ADCS. Testing the ADCS in Earth gravity was not possible without e.g., a Helmholtz cage, but could have been simulated in a Hardware In the Loop (HIL) setup.

wrote the command and corresponding CubeSat Space Protocol (CSP) packet to a text file. It also fetched the timestamps of the command and wrote script delays into the output file in order to make a script that recreated the sequence the operator inputted to the LidSat. The outputted text file would therefore recreate the same operation executed in the instance of `hypso-cli` in the same sequence with an identical delay between commands. This tool was seen as a semi-autonomous tool at best since the script converted a human-readable version to a script version, but would only recreate the sequence and timing of what was typed in. Therefore, the operator had to act like a script engine or modify the scripts afterward. Additionally, the scripts depended on the file structure of the LidSat being an exact copy of the file structure on HYPSO-1.

Even though accuracy was not a major concern at the early stage, it was necessary to learn how to capture at the correct time and have the satellite pointing the camera instruments towards Earth. The default pointing of the satellite had the camera payloads pointing towards nadir as a constrained vector, which was deemed sufficient for the first captures, thus leading to not making a FC script to change the satellite's attitude. The Payload Controller (PC) script was started at a time that gave the Red-Green-Blue (RGB) camera and the HyperSpectral Imager (HSI) a seemingly good pointing towards an area with a lot of features to find suitable camera configurations using the System Tool Kit (STK) software. Before uploading the PC scripts, the scripts were tested on the LidSat and verified by another team member to ensure they were functional. The upload was done using the `hypso-cli` and not the task scheduler in the MCS. From the first scripts, it became clear that the RGB camera was not performing as expected. It had passed the system checkout since it was capturing pictures, but experienced unforeseen anomalies. The pseudocode of one of the first capture scripts is shown in Code listing 4.1. The script captured 10 RGB captures with the same configuration, and two HSI captures with different configurations. The capture script was made using the tool that parsed through the `hypso-cli` log. From the timestamps, one can see that the timing has been manually altered after generation.

Code listing 4.1: Capture script for Sahara capture

```
#13:59:48   Power on the OPU
script delay 9000

#13:59:57   Power on the RGB camera
script delay 34000

#14:00:31   Initialize the RGB camera
script delay 22000

#14:00:53   rgb configure 0.0202 -1 -1 -1 -1
script delay 1000

#14:01:04   rgb capture n rgb png sahara0
script delay 10000
```

```
#14:01:17   rgb capture n rgb png sahara1
script delay 10000

#14:01:30   rgb capture n rgb png sahara2
script delay 10000

#14:01:42   rgb capture n rgb png sahara3
script delay 10000

#14:01:54   rgb capture n rgb png sahara4
script delay 10000

#14:02:06   rgb capture n rgb png sahara5
script delay 10000

#14:02:19   rgb capture n rgb png sahara6
script delay 10000

#14:02:31   rgb capture n rgb png sahara7
script delay 10000

#14:02:42   rgb capture n rgb png sahara8
script delay 10000

#14:02:54   rgb capture n rgb png sahara9
script delay 10000

#Added after generation    rgb print
script delay 1000

#14:03:06   rgb deinit
script delay 14000

# tar cvmof sahara_rgb.tar rgb-images/sahara*
script delay 5000

#14:03:20   hsi capture -n 50 -f 10 -e 10 -s
script delay 51000

#14:04:11   hsi capture -n 2 -f 1 -e 10 -b 1 -s -w 1936 -h 1216
script delay 55000

#14:05:06   opu shutdown
```

4.2.2 Reusing Previous Scripts

After some time of generating specific scripts for each capture, the team figured out some functional capture configurations and sequences which made it possible to reuse previous scripts and exchange a few commands instead of generating a whole new script from scratch. Commands and sequences that worked were used in new versions while errors were corrected. The author also attempted making pointing scripts for the FC even though NA made their own scripts. The FC scripts were sent to NA for feedback. Eventually, these FC scripts were at the level where NA deemed them good enough to use and added them to the script file they made

for their commissioning of the satellite bus. The pointing scripts used quaternion-pointing mode, which allowed the team to point the HSI off-nadir for captures, thus opening up more possible target locations. Capturing off-nadir images was also an objective during the payload commissioning. For more information about quaternion pointing, the reader is directed to [44]. The HYPSO team slowly started becoming more confident with operating the satellite. At this point in time, NA used MCS for their operations while HYPSO manually connected to the satellite using the direct connection method that circumvented the rest of the MCS subsystems during passes over the NTNU Ground Station (GS).

Throughout these scripts, the team also tried buffering data from the payload to the PC. Buffering of the capture to the PC was desired since the PC had a faster communication bus to the S-band transceiver than the OPU. The Controller Area Network (CAN)-bus between the PC and payload had a data rate of 1000 kbps while the S-band transceiver had a data rate of about 1.4 Mbps. Additionally, the CAN-bus was structured as shown in Figure 4.4. The CAN-bus used headers and footers totaling 8 bytes of overhead to send 8 bytes of data in one frame. The information rate of the CAN-bus was therefore 500 kbps, which was much lower than the data rate of the S-band transceiver [45].

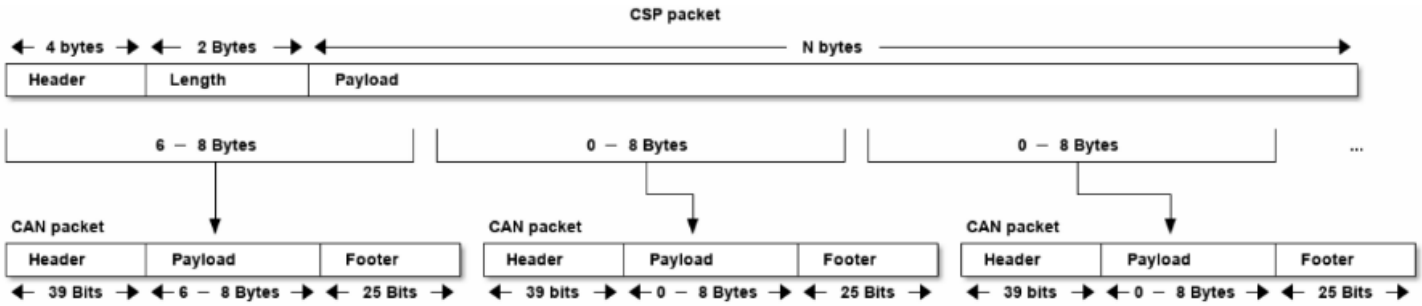


Figure 4.4: Mapping of a CSP packet to the CAN-bus. Figure obtained from [46]

By instead buffering the capture to the PC, the utilization of the S-band download speed was maximized. Buffering the capture to the PC was less power efficient, but the HYPSO-1 satellite had a more constrained download time than power budget, thus leading to all captures being manually buffered to the PC one pass after the capture was executed. The increased downlink capacity led to the team testing capture of a full hyperspectral cube using hardware compression, thus marking off another capability defined in the MOP.

Within these scripts, it was also concluded that the RGB imager was not useful due to the anomalies experienced, thus its utilization ceased. Instead, RGB composites of the hyperspectral images were made with the HSI. This made it even more important that images had to be taken along coastlines or land to have enough features to georeference the captures. Throughout this period, it became

clear that the capture modes in the MOP defined prior to launch were not suitable to follow. Instead of testing these capture modes, the team decided to apply the newly obtained knowledge from the captures that were taken to improve the capture quality, which was the essence of the MOP.

4.2.3 GitHub Issue Template for Requesting Captures

One of the major pain points of script generation was for the operator to get all the necessary information. Members interesting in the data asked for captures of specific targets leading to a lot of overhead for the operator. To ensure that enough information was given to the operator the author made a GitHub issue-template for requesting captures. The issue template limited the workload of the operator by providing coordinates, pointing type, defining time constraints for helping with prioritizing script generation, and HSI capture configurations. By including all this information, the operator avoided having to look into all requested captures to figure out what scripts to make first. The operators also did not need to have a deep understanding of suitable capture configuration for that specific target. It was also made a KanBan board with the swimlanes:

- *todo*
- *script writing and planning*
- *scripts ready to be uploaded*,
- *scripts uploaded and in execution until all files are downloaded*
- *verified captures - ready to delete files on spacecraft*
- *done*

These swimlanes made it easy for the operator to get an overview of the captures to be done, and once a week the HYPSO team planned objectives for the week. Organizing it on GitHub was also compatible with HYPSO's software workflow.

4.2.4 First Script Templates

NA used a lot of time to commission the ADCS system compared to other M6P subsystems. One of the reasons was that the HYPSO team had requested a mission-specific maneuver called *slew*. The HSI is placed on the Z-axis of the satellite, and the slew maneuver is a pointing mode rotating the z-axis across the nadir vector at a specific angular velocity, shown in Figure 1.1. Therefore, as the satellite flies past a target, the rotation compensates for the movement of the satellite. For a more thorough explanation of the slew maneuver, the reader is directed to [47]. Due to the *slew* mode still being under testing, the HYPSO team did not use the mode, and instead used quaternion pointing or nadir pointing for captures.

When the scripts were at a point where the team settled on a functional sequence of commands, the author made script templates where timing was calculated for that specific sequence of commands. One only had to find the timestamp of the middle of the capture and then fill out the unix-times described in the script

template. However, updating all these unix times was tedious work and some errors were made because of this. At the same time, having these unix-times precalculated made it easier for the operator to generate the scripts and for others to verify them. Buffering was also included as part of the script instead of being done manually, saving one pass per capture before having it downloaded. In addition, since the command sequence had become static, the `hypso-cli` parser was not used anymore since it was only needed to update a few commands between scripts. Standard delays were also incorporated leading to improved timing of captures, thus also improved accuracy.

Scripts were still slightly updated when errors or improvements were found. One of the scripts had an erroneous capture. The capture command automatically named the capture folder as *hsi<index>* where the index comes from the number of existing folders following this structure. Therefore, when the buffering command was run, the file to buffer from was hard-coded into the script. Since one capture failed, all the rest was named with the wrong index, causing them to fail. Therefore, the following scripts were updated to rename all capture folders after the capture was completed to force all new captures to be named *hsi0* such that one erroneous capture would not propagate to other captures. The problem with continuously updating the script template was that it made other members uncertain about what version to use. Some members opted to use an earlier version of scripts that had been functional previously instead of using the latest version of the script templates. This led to different versions of scripts being used, thus not forming unified naming conventions and sequences. These discrepancies led to confusion, particularly for operators with limited knowledge of how to use the payload.

At this point, the team had tested all the short-term functionalities described in the MOP. There were still tasks to be done on the calibration of the HSI, in-orbit software updates, capture using the slew maneuver, dimensionality reduction, and further improved exposure times. Slew capture was completed on the 7^{th} of April and exposure times have been improved, but the software update and dimensionality reduction were not completed during the project period of this thesis.

4.3 Increasing the Number of Captures

In order to maximize payload utilization of the HYPSO-1 satellite, operations of the satellite had to be made more accessible for new members, and the time-consuming tasks had to be mitigated. However, the satellite also had to produce payload data, not just telemetry. It was therefore made generic capture scripts that took, and buffered, one capture each night. There were no specific targets meant for these generic targets other than randomly selected coastlines. Since there was not a specific target, the scripts were quicker and easier to produce. The names

of the captures were also set to be generic, thus leading to scripts only needing updated timestamps between scripts. The scripts cleaned up after themselves on the OPU. However, this also meant that if one download failed, the cube would be deleted. Having the capture deleted was not deemed an issue since the generic captures were of a lower priority, thus leading to less workload per downloaded capture. Manual cleanup and updating names each week would take a too long time to be worthy of possibly saving a few extra captures without integrity. The time consumed doing so would be put to better use by generating more targeted captures of higher quality instead.

There were three major changes from the targeted scripts to the generic scripts. The first one being that the FC script started the PC script. Since the FC script started the PC script, and all capture configurations and names were the same, the PC script did not have to be updated. The script engine utilization is shown in Figure 4.5. Not updating the PC script reduced the number of timestamps to be updated from four timestamps down to one per capture. This change not only reduced the amount of work but also the chance of human error. The second major change was that the author scheduled automatic download tasks through MCS to download the captures, a change that also was implemented in the script templates afterward. The captures, therefore, relied on MCS to successfully download the captures each day. The third and final major change was that verification of scripts became excessive since only the timestamps of the captures itself were updated and the timestamp could not be verified by testing on the LidSat. Only timing within a script could be tested on the LidSat. Since there were no specific targets to aim towards, all captures were taken nadir, thus resulting in it being easy to georeference the captures. The timestamp of capture start was chosen and put into the orbit simulation software STK, and the capture location was found.

Being reliant on the MCS reduced the amount of work, but MCS also crashed sometimes. It was decided to do cleanup and download manually. It became clear that being reliant on MCS might also be a bad thing. Instead of following the plan of not caring about the generic captures, the author put in the effort to manually clean up.

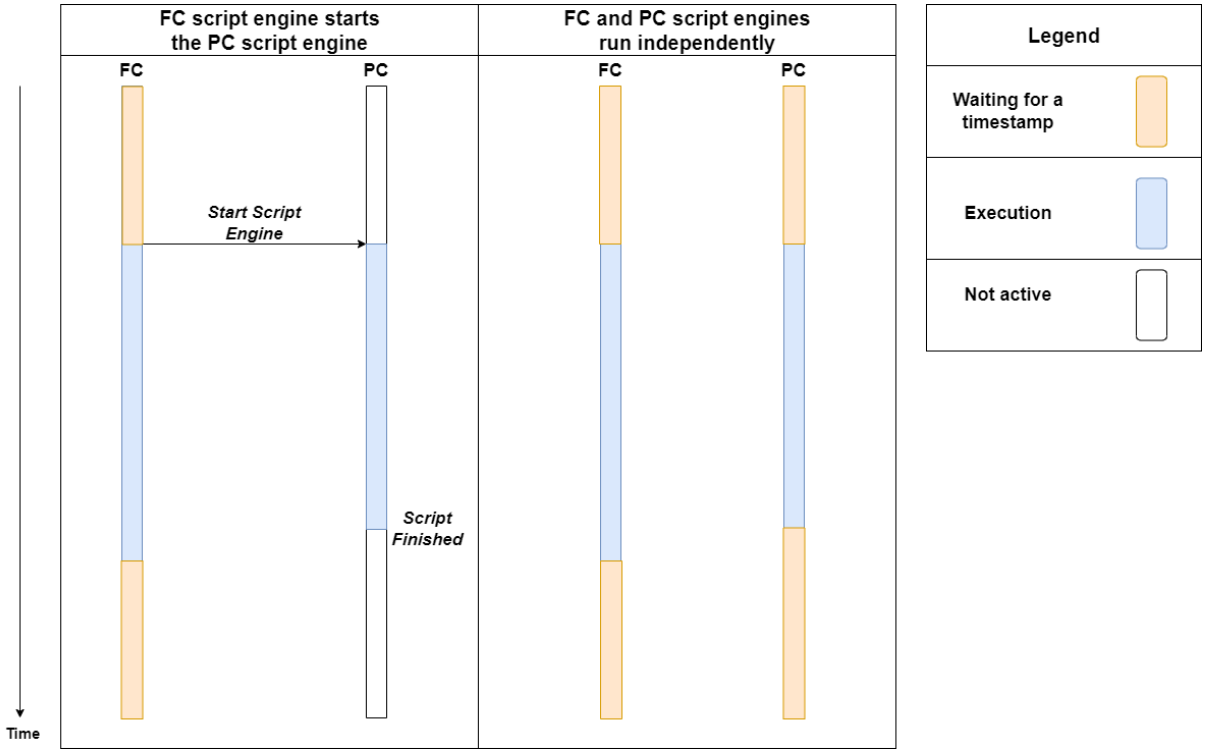


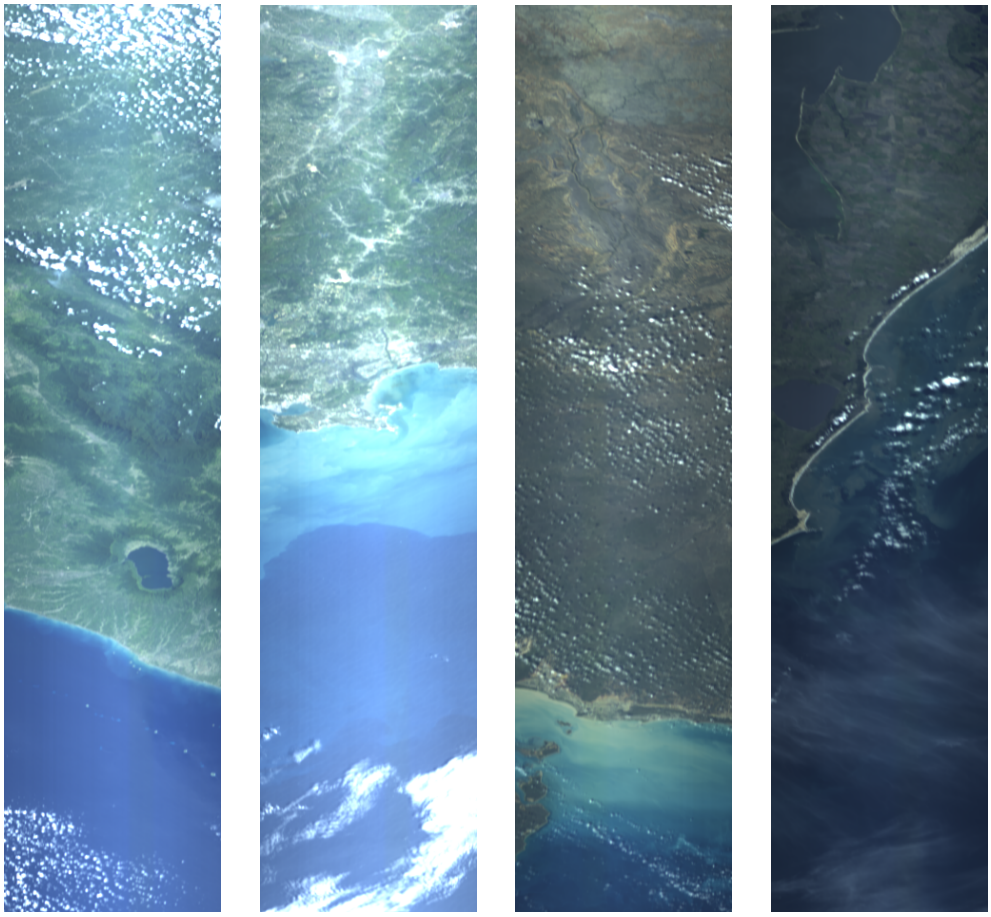
Figure 4.5: Two different ways of using the script engine. The version to the left use the script engine on the FC to start the script engine on the PC. The example to the right shows them running independently.

4.3.1 Second Version of Script Templates

Throughout the period of using script templates, improvements were done to the scripts when needed, based on the feedback from the operators. The most notable improvement was that the scripts buffered the capture as part of the PC script such that the operator only had to download the capture and not manually buffer them. The author also started getting familiar with MCS and set up automatic download tasks by using the task scheduler on the 22nd of February. By doing so, operators only had to asynchronously pull the payload data from the database on AWS. The second version of script templates also had improved timing of capture to further improve accuracy. It was implemented with another script delay command that made the payload wait until a specified unix-time before issuing the capture command. It added an extra unix-time to update when making scripts but made the timing of the capture have less uncertainty. The second version of script templates also included more information about the calculation of the delays, thus making it possible for other operators to change the capture configuration and calculate updated delays correspondingly. At last, the second generation included an improved metazip that collected more of the useful telemetry data surrounding the capture, thus leading to fewer files needing to be downloaded manually.

4.3.2 Improved Generic Capture Scripts

The second version of the generic captures made and buffered a metazip with supporting information surrounding the capture, which previously had been downloaded manually. Additionally, the scripts cleared the buffer files used on the PC after download. The buffer files were log files with a file size of 100 MB that rewrote itself from the beginning if it was full. They did not have to be cleared between each use since the File Transfer Manager (FTM) in MCS continuously checked the entries corresponding to different file versions. However, if MCS crashed, manually figuring out these entry numbers was challenging. By clearing the files between all captures, manual download or cleanup would be easier in case of an error happening. This change did not follow the plan that generic captures were of a lower priority than targeted captures. The change was an easy preventative measure to make, and gave the operators an option in case the capture seemed to be of a good quality based on the file size. Having the generic script on the PC also made it easy to take ad-hoc captures during passes without knowing how to operate the payload. One only had to start the PC script and the capture would be taken, buffered, and downloaded automatically. Figure 4.6 shows four of the captures taken by the generic scripts. At this point in time, NA did not test much other than their slew maneuver. All scripts were therefore sent to NA on a weekly basis to merge with their scripts and NA planned their tests around the capture times.



(a) Low-resolution RGB-composite the generic capture on the 1st of April

(b) Low-resolution RGB-composite the generic capture on the 5th of April

(c) Low-resolution RGB-composite the generic capture on the 5th of May

(d) Low-resolution RGB-composite the generic capture on the 13th of May

Figure 4.6: Low-resolution RGB-composites of four generic captures

4.3.3 Training of Operators

During the whole timeline from Figure 4.1, a prominent problem was that a lot of new members struggled to learn how to be an operator. The author wrote a user manual ([48]) introducing the PC and FC subsystems, the *csp txrx* command, timing considerations and the script templates. Since the workflow of operations rapidly changed and operating the satellite required reactiveness in addition to proactiveness, it was attempted to give the reader an understanding in order to apply the knowledge to specific situations instead of following a specific sequence. The document explained all the involved parts of making scripts. It became clear that the members that understood the document either had a previous understanding of the system or systematically understood all of the documentation by

asking specific questions when something was unclear. The members without this previous knowledge of the system instead made their own bullet-point list of how to make scripts. The bullet-point list made them capable of making scripts but lacked an understanding of what they did. This led to scripts using the wrong parameter values or being uploaded to the wrong script engine since they copied the values from the example in the bullet-point list. Additionally, the members never obtained the fundamental knowledge of how the system worked, thus leading to errors and time-consuming troubleshooting.

4.4 Automating Communication with the Satellite

MCS had the potential for automating parts of the time-consuming operation that was done manually by the HYPSO team. This section will cover the initiatives taken in order to utilize these capabilities.

Compared to subsection 2.4.1, MCS covered the same areas as GENIE and partially GenSAA and ASPEN through having a task scheduler, automatic download of telemetry and payload data, and visualizing the telemetry data using Grafana. However, it did not provide information about potential errors and probable causes. Additionally, `nanoMCS` did not have access to the remote shell of the OPU. Nevertheless, it was decided to use MCS since it was already integrated with the satellite platform. If another system were to be used, tailoring for utilization of the mission-specific payload would still have had to be done. In [49] and [50], they found that integration is time-consuming in academic CubeSat missions, and advised other academic CubeSat missions to start easy and iteratively build upon the functionality. Using MCS gave a functional baseline during the development of additional tools to further automate operations. It was therefore chosen to build upon MCS by starting to utilize its capabilities and developing supporting tools with new functionality.

As mentioned above, MCS fully covered pass automation through the task scheduler. However, it did not cover assisting in troubleshooting or generating low-level scripts based on high-level input. It was chosen to begin by automating the utilization of the satellite before automating troubleshooting since the author had more knowledge about the payload than the satellite platform. Additionally, there is no need for a healthy satellite if it does not produce any useful payload data. To partially cover the troubleshooting it was instead scheduled automatic downloads of all necessary telemetry needed to manually troubleshoot asynchronously.

4.4.1 Task Scheduling

The use of the task scheduler is shown by the yellow arrow in the timeline in Figure 4.7

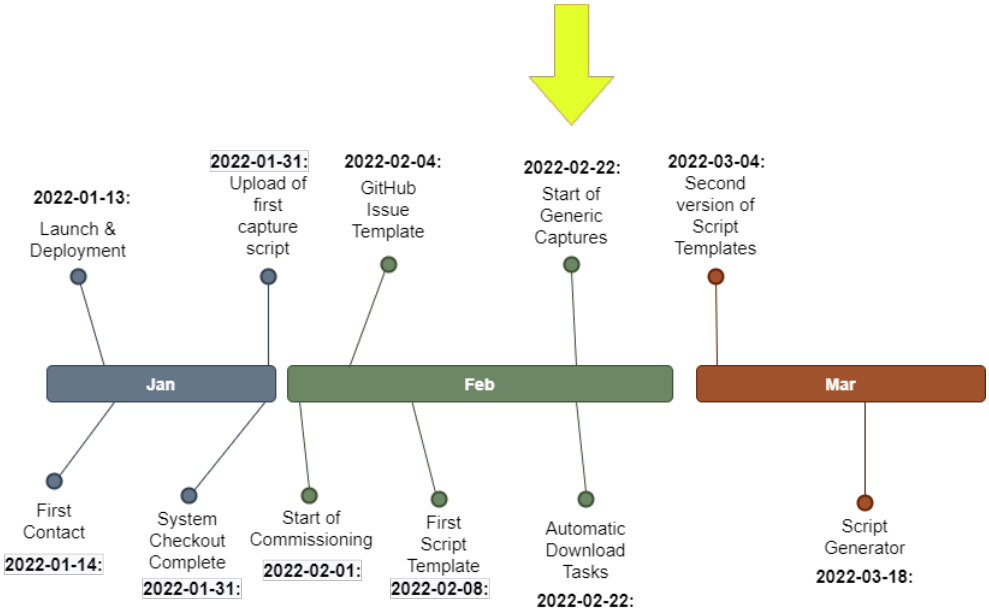


Figure 4.7: Timeline showing where the work with the task scheduler began indicated by the yellow arrow.

The task scheduler could be used for file upload, file download, and `nanoMCS` tasks. The file upload task uploads the defined file to the specified subsystem and file id. The file download task downloads the content of a specified file id on the defined subsystem of the satellite. The `nanoMCS` tasks were used for other communication with the satellite. All tasks could be given a name, a priority ranging from 1 - 100, a duration it was allowed to run for, a timestamp of when to activate, if it should be repeated on all passes, and if the task depended on the outcome of any other tasks. Additionally, the task scheduler supported the Microsoft common parameters (described in [51]). The task scheduler produced logs from all satellite passes and had the ability to show the live progress of a pass. If a task was scheduled to run at a specific timestamp, it was attempted at the first possibility after that given timestamp. If the task failed, it was attempted again during the following pass until it reaches success.

Instead of having all operators learn how to use the task scheduler and the syntax of `nanoMCS`, the author made generic tasks that could be resumed with minimal edits. File uploads were set up using `nanoMCS` tasks instead of file upload tasks. The `nanoMCS` tasks were used since they allowed for ensuring that the script engine the file was uploaded to was enabled, running the correct script file, and

could start the script engine. The file upload task was not suitable for the upload of script files since it only uploaded the script file without starting the script engine. The file upload was therefore only suitable for uploading files that were not meant to be executable, such as Two Line Element (TLE) files. A pseudocode example of one such script is provided in Code listing 4.2. The task started by ensuring that the script engine was correctly configured by enabling it, aborting any current scripts if one was running, and setting it to the correct script file. Then the task uploaded the capture script before starting the script engine and ended by checking the status of the script engine. All commands were running until success with a delay of 1000 ms between each attempt. For the script upload, the script file was allocated two seconds to complete the upload, with a 10 ms period between entries. In order to upload a new file, other operators only had to change the file path in the upload command, then resume the task in order to activate it.

Code listing 4.2: script upload task

```
#enable script engine

#abort current script in case any are running

#reconfigure script engine to run file ID 15

#Upload of the capture script

#start script

#Check status
```

Such generic tasks were made for upload to all script engines on the satellite. There were also made tasks to reboot the different subsystems on the satellite in case that became a necessity. Automatic download of telemetry data from all subsystems of the satellite platform was scheduled to be executed on every satellite pass. The tasks that were set to always be active are shown in Table 4.1

Table 4.1: Overview of the task scheduler task that are always on

Name	**Purpose**
eps-gs-wdt-reset	Reset watchdog timer
comm-gs-wdt-reset	Reset watchdog timer
srs4-gs-wdt-reset	Reset watchdog timer
srs4-read-stats	Download telemetry
read-config-status	Check when- and how many times configs has changed
read-configs	Check all active-, main-, and fallback configurations
read-versions	Get githash and bootloader versions
check-time	Check RTC and GPS time
FT-FC-SE0-LOG	The log output of script engine 0 on the FC
FT-FC-SE1-LOG	The log output of script engine 1 on the FC
FT-EPS-Telemetry	EPS telemetry download
FT-COMM-1-Telemetry	UHF telemetry download
FT-FC-Telemetry	FC telemetry download
FT-COMM-2-Telemetry	UHF telemetry download
FT-EPS-Startup-Telem	EPS startup telemetry download
FT-COMM-1-Startup-Telem	UHF startup telemetry download
FT-FC-Startup-Telem	FC startup telemetry download
FT-PC-SE1-log-18	The log output of script engine 0 on the PC
FT-PC-SE0-log-17	The log output of script engine 1 on the PC
SE-status-FC-PC	Status of the script engines on the FC and PC
FT-PC-Telemetry	PC telemetry download
FT-PC-Startup-Telem	PC startup telemetry download
FT-FC-ADCS-Telemetry	ADCS telemetry download
FT-PC-File-33	Download of buffer-file 33
FT-PC-File-34	Download of buffer-file 34
FT-PC-File-35	Download of buffer-file 35
FT-PC-File-36	Download of buffer-file 36
FT-PC-File-37	Download of buffer-file 37
FT-PC-File-38	Download of buffer-file 38
FT-PC-File-39	Download of buffer-file 39
FT-PC-File-40	Download of buffer-file 40
FT-FC-GPS-log	Download of GPS data

All these tasks were attempted to execute on all passes the HYPSO-1 satellite had above any of the GSs in use. Tasks were added continuously when the author found of need for them.

Key points from this section are summarized below:

- The task scheduler made it possible to work on the satellite without waiting for HYPSO-1 to be in the range of a GS.
- The task scheduler was used to download telemetry data for monitoring and troubleshooting the satellite continuously.
- The author made generic tasks that anyone could easily resume, thus not

needing to make their own tasks.

4.4.2 Interfacing with the Payload Using the Task Scheduler

As stated in subsection 4.4.1, the task scheduler could run `nanoMCS` scripts. The PC was made to interface with the payload, but only with peripherals and CSP. The interface was therefore done through the PC shell. One problem that occurred was that even though the PC and the OPU supported CSP, the implementation differed. The M6P subsystems implemented the use of CSP through a `csp txrx` command. The `csp txrx` command was declared as *csp txrx node<int> port<int> delay<ms> hex_code*, where the `node` was the CSP id given to specific subsystems, `port` was the service port used to route the message correctly, the delay was the amount of time the command could wait for an acknowledge up until 3000 ms. The `hex-code` was the command itself in hex format. The command was denoted as an error if no acknowledgment was received within the timeframe given. However, the OPU was implemented with functions sending acknowledgments when the function was completed successfully. The two different implementations led to some commands timing out and being denoted as an error even though it was a success. This issue is visualized in Figure 4.8.

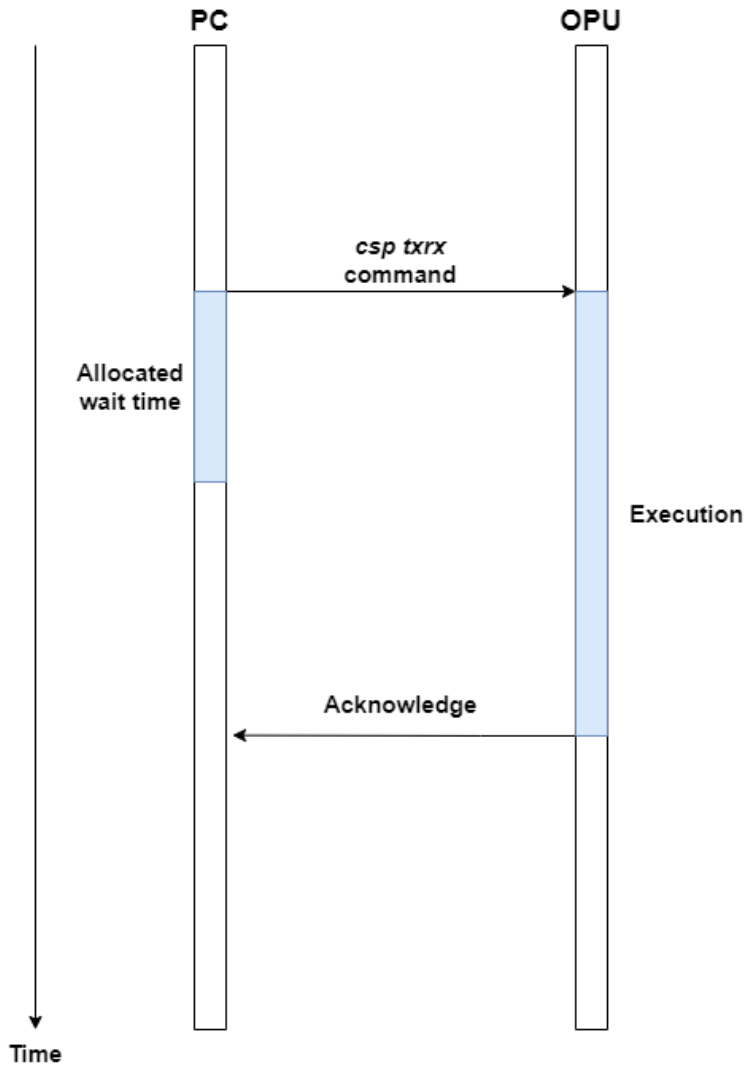


Figure 4.8: Sequence diagram of how the PC denotes a successfull command as a failure since the timeout is exceeded.

The response time limitation led to a problem with the command for buffering captures from the OPU to the PC since it only sent an acknowledge after buffering was complete, and the timing used for the command was dependent on the size of the file. This led to it being impossible to use the repeat-until-success functionality in the task scheduler since it would denote the buffering as a failure and re-send another buffering command. Not only would such an error potentially cause the payload to become confused, but it would also prevent further scheduled tasks from running since it would re-try the same command over and over. Since the task itself would be denoted as an error, it would be attempted during all future passes until the task was stopped by an operator. Additionally, it would buffer more unnecessary data to the file which would also waste resources by down-

loading if not stopped by an operator. Interfacing with the payload was possible but was only done with caution. The author's solution was to submit a GitHub issue on sending an acknowledgment for the start of buffering instead of waiting until the buffering was finished such that these problems were circumvented. The change is implemented in the current software release on the ground that will eventually be uploaded to the HYPSO-1 satellite.

Even with the problems with the use of the buffering command when using the task scheduler, there was made an attempt to make a task to daily download the payload logs. Since the task scheduler only supported running a task on every pass or after a specifically given timestamp, the task had to be resumed daily. This was done by setting up a task using Windows task scheduler to open the interface to MCS and command it to resume the task every night. The task was resumed during the night since it would then be executed during the first pass of the morning, which always was a low elevation pass not used for imaging, thus not interfering with capture objectives. It could not use the repeat-until-success functionality, so it had to only send the buffering command once and hope it was successful. Pseudocode of the task is presented in Code listing 4.3. Even though it deleted old logs with no security in actually completing the buffering of the file, it was tried for about a month since all logs were backed up in the metadata of each capture. The times the satellite had imaged since the last download of logs and there was a capture pass on the second pass of the day, the logs download ran close to captures happening. The reason for it happening on the second pass was that the task was denoted as a failure on the first pass because of a large log file, thus resulting in the buffering not being completed within three seconds. The task was then executed again on the second pass. The task was therefore paused until the coming software update.

Code listing 4.3: Automatic payload logs download task

```
# Power on OPU

#Wait for OPU to boot
delay 25000

Clear the buffer-file

#ls -l

#delete old logs.tar file

#make new logs.tar

#delete logs/*

#buffer logs.tar to file id 33. This command use more than 3sec and cannot be
#repeated so we just have to hope it works

#Wait for buffering to complete
delay 20000
```

Key points from this section are summarized below:

- It was possible to interface with the payload using the task scheduler.
- The CSP implementation on the M6P platform differed from the implementation on the payload leading to some commands always being denoted as a failure.
- A software update on the payload will make them compatible.

4.4.3 Automating Script Generation

The automatic script generation is shown in the timeline by the yellow arrow in

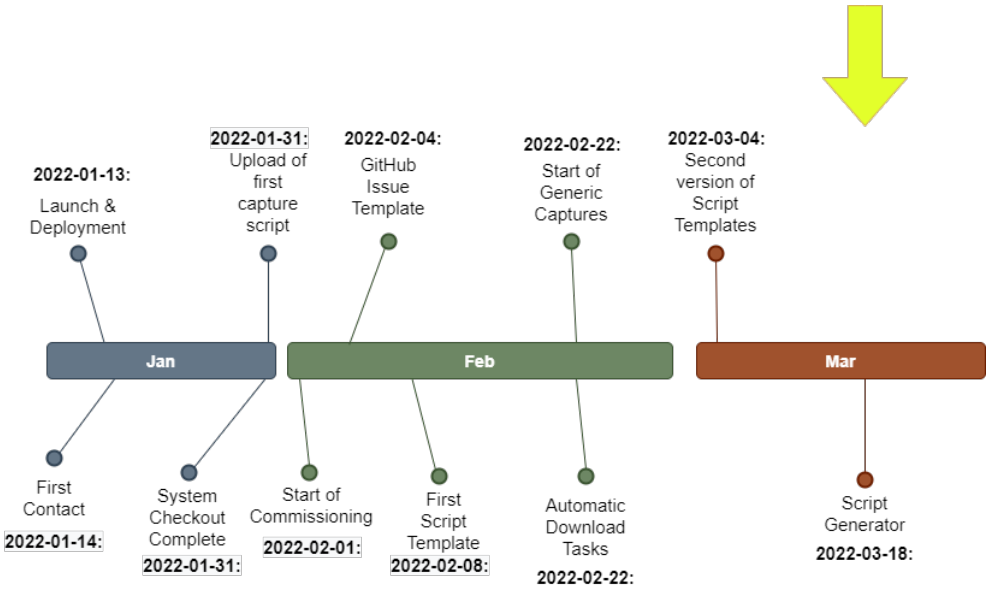


Figure 4.9: Timeline showing where the work with the script generator began indicated by the yellow arrow.

The first version of the script generator was a python script taking some high-level user-input to generate capture scripts. The initial script generator only made nadir capture script and was ready for use on the 18th of march. The input needed was the unixtime of highest elevation above the capture target, name of the capture, exposure time and the file-id to buffer the hsi-cube and metazip to. The tool made capture planning a lot easier and was therefore extended to cover more capture modes the next day. On the 19th of march it was therefore possible to select nadir-, quaternion-, generic-, or slew pointing. The script generator calculated FPS from the exposure-time input since it was desired to have the highest possible FPS in order to maximize the spatial resolution. This useful tool removed the knowledge barrier connected to using hypso-cli, understanding the csp interface between the PC and OPU, and previous capture scripts. Additionally, all timings were calculated and optimized for the different capture modes. Based on

EPS telemetry, the power budget was not a limiting factor, but the satellite lifetime is affected by the depth of discharge of the EPS [1]. The Cycle Life of the power supply differs between batteries, but the relationship between cycle life and depth of discharge is of an exponential nature [1].

Since all the script structures were static and made by the python script, the scripts did not have to be tested on the ground before uploading to the satellite, which removed the time needed for testing as well as severely mitigated the probability of errors. Additionally, if errors were found, they only had to be fixed once and they would never occur again. Since the team was still learning to utilize the HYPSO-1 satellite, the operational patterns slightly altered from day to day and week to week. These alterations led to a need for frequent, but small, adjustments.

Functionality of the Script Generator

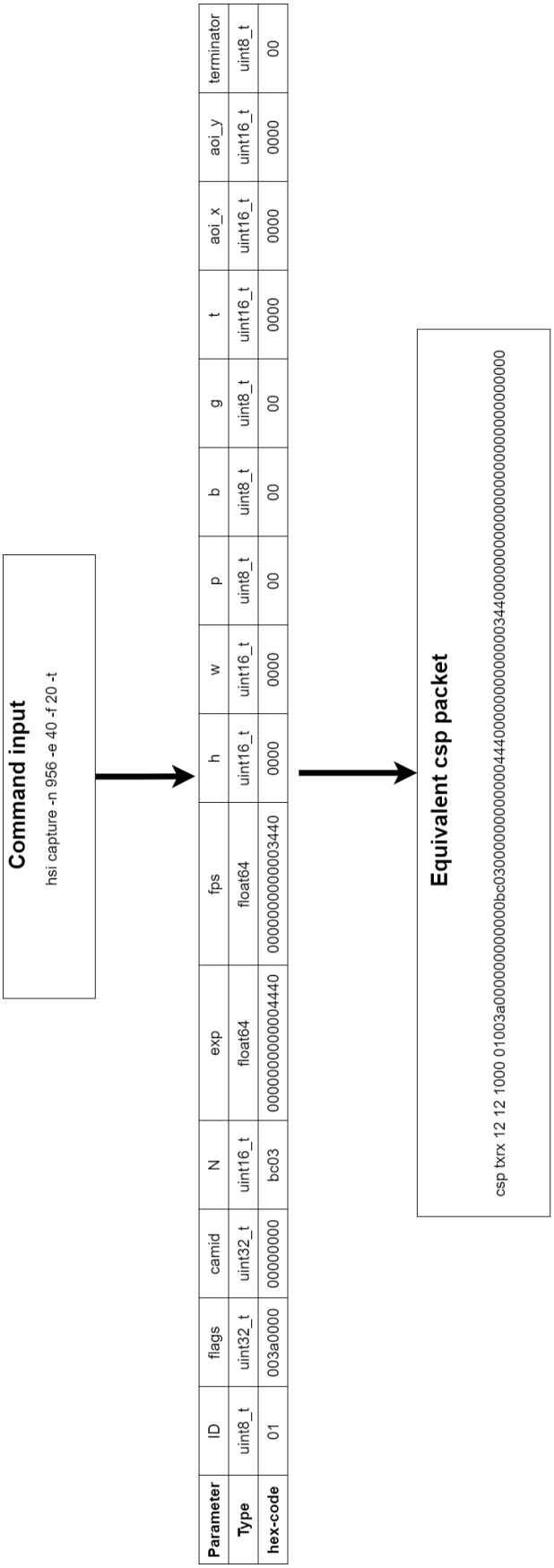
The script generator translates high-level user input into predefined capture structures and CSP packets. The output of the script generator is a PC- and FC script. As described in subsection 2.2.5, the PC script handles commands related to the payload while the FC script takes care of commands to the ADCS. The script engine on the PC runs the script in the PC shell, thus meaning it has access to shell commands on its internal subsystem, but has to use CSP packets to communicate with other subsystems. One example of how these hex-codes looks is provided in Figure 4.10 where the *hsi capture* command is defined.

With the addition of the script generator, the time used for generating capture scripts heavily decreased. Captures for the week were planned every Monday at noon, and scripts were uploaded the same evening.

Parameter	ID	flags	camid	N	exp	fps	h	w	p	b	g	t	aoi_x	aoi_y	terminator
Type	uint8_t	uint32_t	uint32_t	uint16_t	float64	float64	uint16_t	uint16_t	uint8_t	uint8_t	uint8_t	uint16_t	uint16_t	uint16_t	uint8_t

Figure 4.10: Definition of how the hex-code of the *hsi capture* command is built.

The script generator takes the exposure-time input, calculates the FPS and then translates the inputs to their correct hex-value to make the hex-string of the equivalent CSP packet of the shell command, shown in Figure 4.11.



Parameter	ID	flags	camid	N	exp	fps	h	w	p	b	g	t	aoi_x	aoi_y	terminator
Type	uint8_t	uint32_t	uint32_t	uint16_t	float64	float64	uint16_t	uint16_t	uint8_t	uint8_t	uint8_t	uint16_t	uint16_t	uint16_t	uint8_t
hex-code	01	003a0000	00000000	bc03	0000000000004440	0000000000003440	0000	0000	00	00	00	0000	0000	0000	00

Figure 4.11: Description of how a shell command is translated to its equivalent csp packet.

The script generator calculates the necessary parameters and then translates them into the CSP packets used by the *csp txrx* command. The CSP id and port are hardcoded since the structure of the scripts is always the same. All the variables lie within the hex-code part of the *csp txrx* command. An overview of the dataflow of the script generator is shown in Figure 4.12. The user input leads to a selection of a specific capture structure before generating separate PC and FC scripts. A pseudocode example of the outputted script files can be found in Code listing A.1 and Code listing A.1.

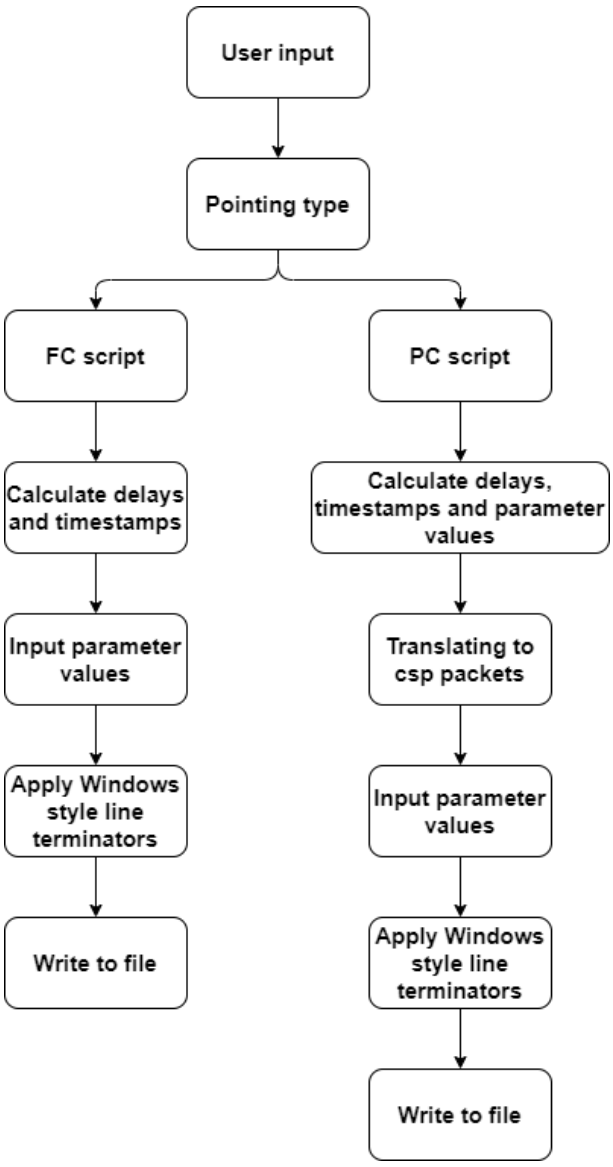


Figure 4.12: High-level dataflow of the script generator

Key points from this section are summarized below:

- The script generator makes deterministic capture scripts from high-level input.
- The script generator removes all human errors within the scripts.
- Since scripts are deterministic they do not have to be tested before being uploaded to the satellite.
- The introduction of the script generator removed the need for operators to have extensive knowledge about `hypso-cli`, CSP, and previous scripts in order to make capture scripts.

Chapter 5

Using the Script Generator for Mission Specific Capture Campaigns

(Shea's Law) The ability to improve a design occurs primarily at the interfaces. This is also the prime location for screwing it up.

Dave Akin

Throughout the project period, the HYPSO-1 satellite was used in two separate mission specific capture campaigns even though the payload commissioning was not complete. Due to the payload still being in commissioning, the main objectives of the capture campaigns were mostly set on showing the concept, not producing valuable payload data. The campaign captures had a higher priority than all other captures. Having a higher priority meant that other captures would be removed if they interfered with the campaign captures. This chapter will describe the preparations, execution, and experiences done within these campaigns.

5.1 Frohavet - The First Campaign

The Frohavet capture campaign for HYPSO-1 spanned from the 23rd of March up until the 2nd of May, but the active campaign spanned from the 28th of March until the 27th of April. The capture campaign was executed as part of the Mission-oriented autonomous systems with small satellites for maritime sensing, surveillance and communication (MASSIVE) project, including other remote agents from the observational pyramid shown in Figure 2.1. HYPSO-1 was still in commissioning, so the objective of HYPSO-1 in the campaign was to show the concept of the full observational pyramid. A timeline of the capture campaign is shown in Figure 5.1. An overview of the errors is shown in Table 5.1.

Table 5.1: Description of error IDs

ID	Description
E01	Reboots caused by simultaneously reading and writing to the memory card on the PC
E02	MCS crash caused by the FTM and Task Scheduler running out of memory
E03	PC not accessing the SD card since it loaded into the wrong bootloader
E04	Renaming error on the OPU caused by multiple captures having the same name
E05	Star Tracker anomalies

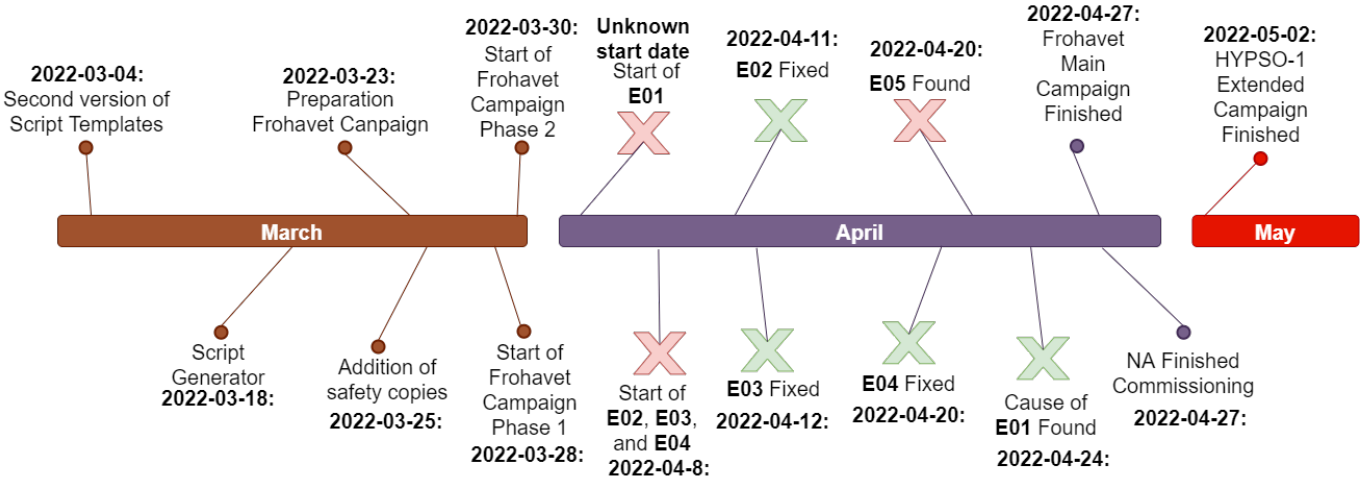


Figure 5.1: Timeline of the Frohavet capture campaign

5.1.1 Preparation

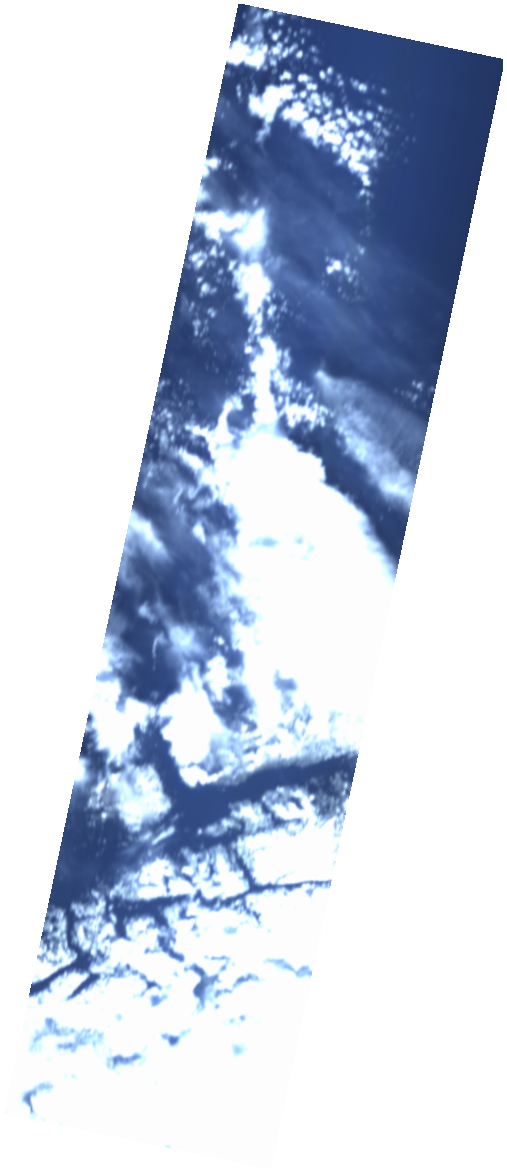
To prepare for the Frohavet capture campaign, HYPSO-1 started capturing hyperspectral images of the target area ahead of the other autonomous agents being deployed. This was done as a test to see if the correct target was hit or not. The first attempt missed the target due to calculating the wrong quaternions and the second attempt was lost due to a reboot, thus causing the script engines on the PC to stop. Reboots became a frequent problem during buffering of captures and were later found to be caused by **E01**. These buffering problems led to a change in the script generator. Instead of deleting old captures, old captures were now stored as safety captures on the OPU until an operator manually deleted them. The safety copies were made to have a backup of the capture in the case that buffering would fail such that it was not deleted before the next capture was scheduled to happen. At this point in time, the generic captures were happening

during the local nighttime, and Frohavet captures happened during the highest elevation pass over the GS since the target area was right beside the GS. As a result of the captures happening during the highest elevation pass, the capture timings of the script generator were improved to not point towards the target area with the cameras longer than necessary and instead get back to pointing the S-band antenna towards the GS to downlink payload data more efficiently.

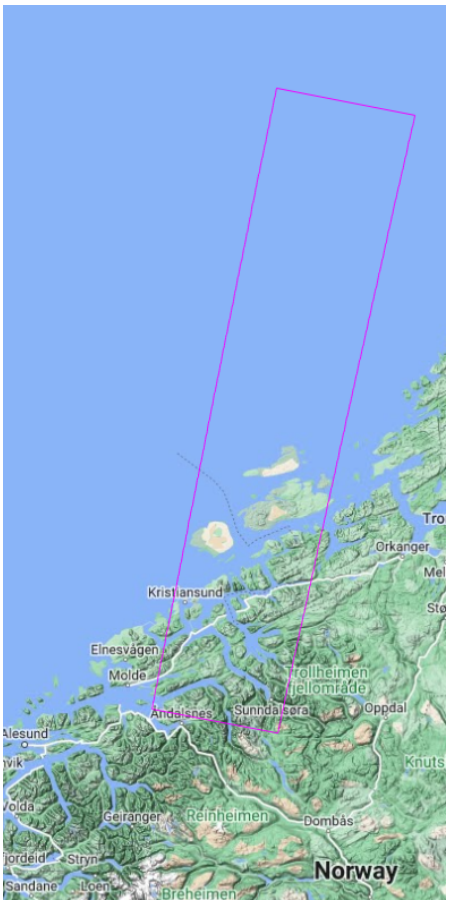
5.1.2 How the Workflow Affected the Main Campaign

The capture campaign consisted of a drone equipped with an HSI V4 instrument, a drone equipped with a hypserspectral SpecIm AFX10 imager, an Unmanned Surface Vehicle (USV), a Light Unmanned Surface Vehicle (LUSV), and the HYPSO-1 satellite [52]. All the remote agents were equipped with an HSI as part of their sensor suite. Additionally, radiometric measurements, water sampling, and a numerical simulation model were used [52]. The campaign consisted of two phases. During phase 1, all of the aforementioned techniques were used over a timespan of 36 hours [52]. Phase 2 consisted of the AUV, the numerical simulation model, and occasional water samples [52]. The objective of the first phase was to see what level of detail surveillance using HSIs could provide. The objective of the second phase was to validate the numerical model and HYPSO-1's capability of tracking changes in the coastal environment over time using the LUSV as a ground truth.

During phase 1, the capture on the 28^{th} of March was lost due to a reboot, but on the 29^{th} of March, HYPSO-1 captured the image shown in Figure 5.2. At the same time as the capture in Figure 5.2 was taken, the other remote agents were deployed in the same location. In [52], E. Oudijk made a figure showing the ground track of the other remote agents during the capture campaign. This overview is shown in Figure 5.3.



(a) Low resolution RGB-composite of the the image captured by the HSI on HYPSO-1



(b) Ground plot of the capture made by Dennis Langer

Figure 5.2: Low-resolution RGB-composite of the capture taken during the phase 1 of the Frohavet capture campaign

During phase 2 of the campaign time, HYPSO-1 still suffered from many reboots later found to be caused by **E01**. Most of the team went on easter vacation leaving only the author as the sole operator for about two weeks during this campaign. Additionally, since the algae bloom season was starting, the number of captures increased. The team had previously only executed one or two captures a

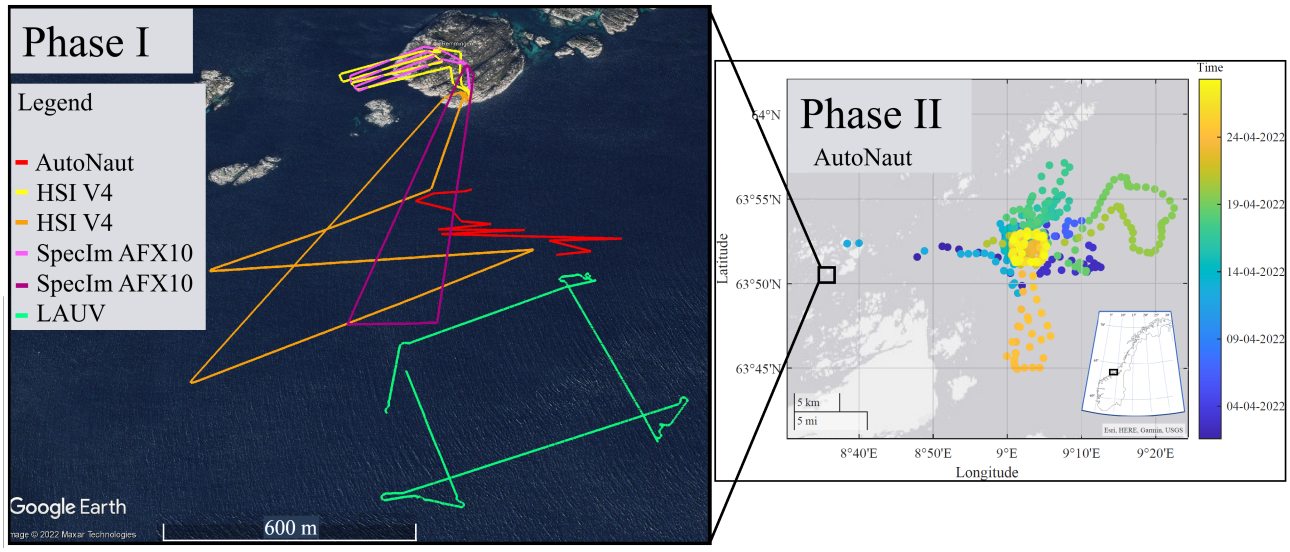


Figure 5.3: Map showing the ground track of the remote agents during both phases of the capture campaign. Figure obtained from [52]

day but was planning to take five captures on the 9th and 10th of April. Changes done when adding the safety copy led to captures not being deleted but rather renamed. These changes were not implemented for the generic captures since they were of a lower priority. The sequence of the resulting error caused by the addition of safety copies is shown in Figure 5.4

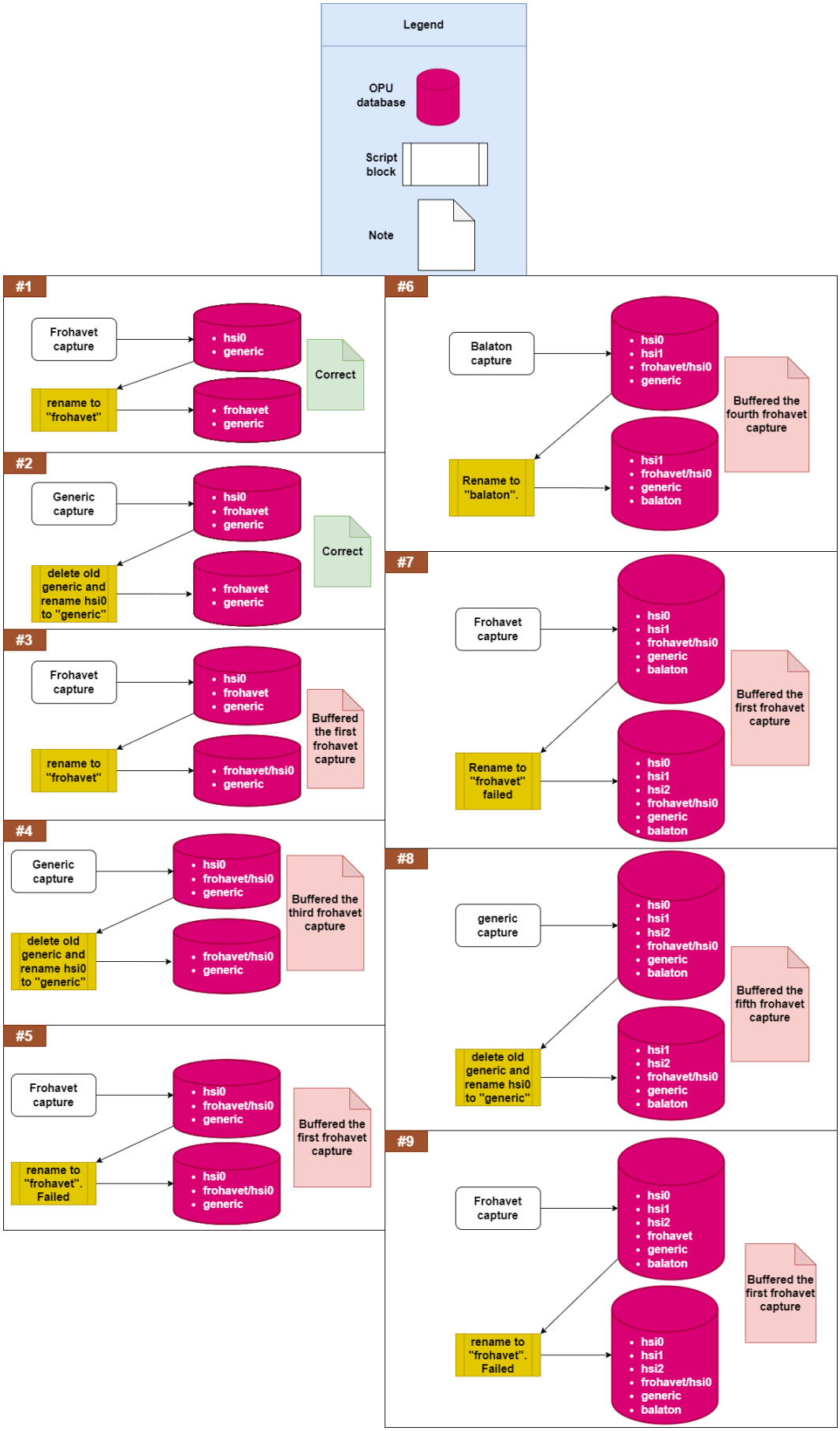


Figure 5.4: Sequence describing the **E04** error.

The safety copy became a problem since it resulted in a need for different names on all captures and was not communicated clearly to the **mission planners**, leading to the error named **E04**. The first capture happened as normal and renamed the capture folder from *hsi0* to the specified name. However, since the next capture was also named *frohavet*, the *hsi0* folder was put inside the existing *hsi0* folder, leading to a wrong path for the buffering command, thus buffering the first *frohavet* capture once more.

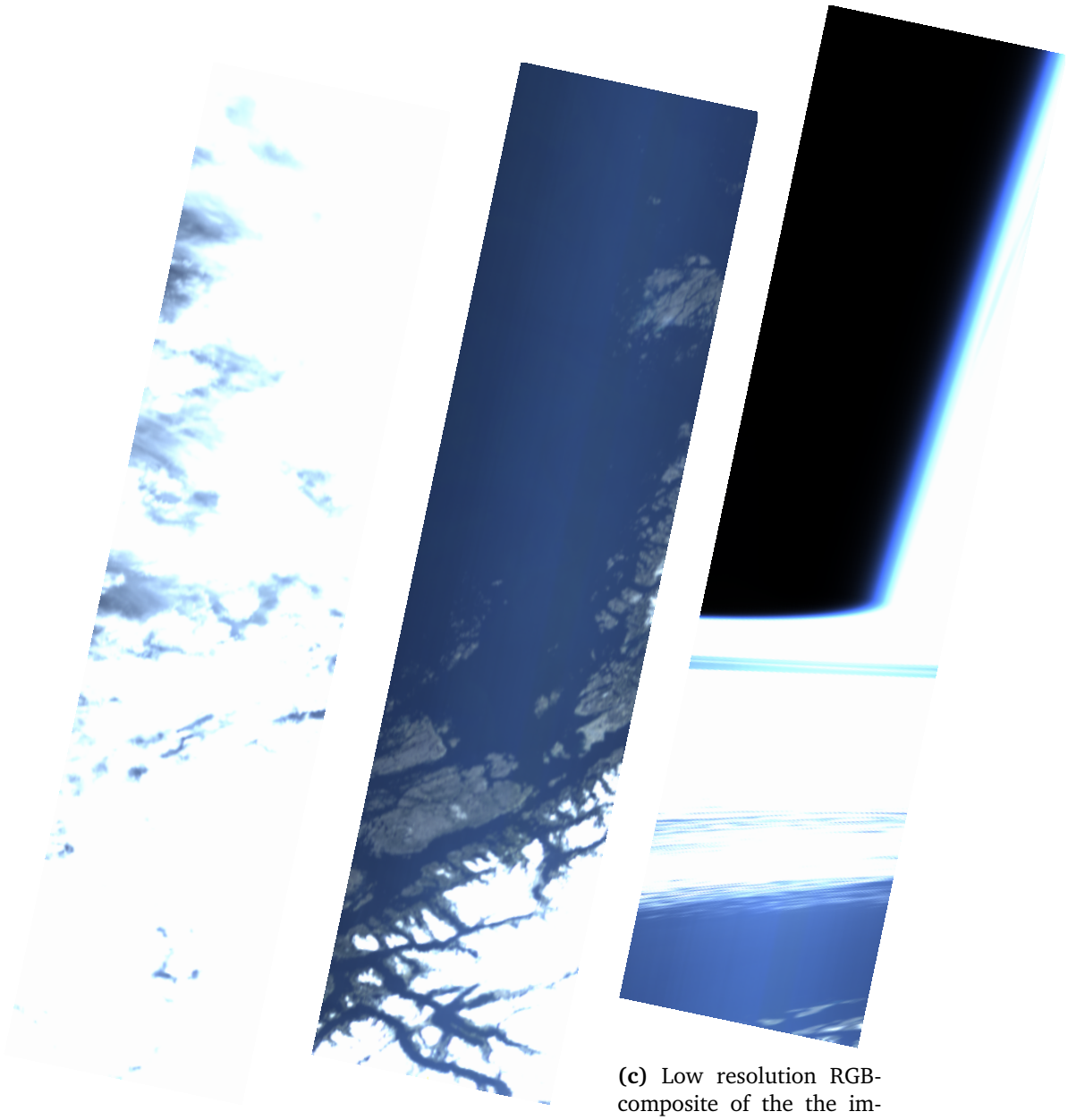
The third capture using the same name caused the renaming command to fail because of the existing folder. This meant the next *generic* capture was named *hsi1* because there already existed a folder named *hsi0*, as described in subsection 4.2.4. Nevertheless, the script renamed the capture in the *hsi0* folder to *generic* and buffered it. That meant the capture named *generic* was actually the third *forhavet* capture. Since the generic capture deleted the folder after itself, it did not cause any further issues with the generic captures. The next *frohavet* capture then had the same malfunction and buffered the first *frohavet* capture once more. The following capture was named *balaton*. Since there existed an *hsi0* folder, it was named *hsi1*. It then renamed the *hsi0* folder, which was actually the fourth *frohavet* capture, to *balaton* and buffered it. The frohavet capture was then named *hsi0* and again buffered the first frohavet capture. The following generic capture was then named *hsi2* and renamed the fifth frohavet capture under the name of *generic* before deleting the folder.

Lastly, the *frohavet* capture had the same malfunction as earlier and buffered the first *frohavet* capture once more. This became an alternating series where *hsi*-folders piled up and were stored under the incorrect names. This went on for a couple of more iterations before the author stopped the scripts and figured out the cause. The error went unnoticed for so long since there were other errors happening at the same time.

While the captures piled up, the download tasks stopped working and MCS crashed because the FTM and the task scheduler went out of memory, named **E02**. In communication with NA, that issue was fixed. At this point, the capture scripts were aborted and it was attempted to rebuffer the captures to the PC. However, the satellite stopped downloading the captures even though the logs from the OPU showed that buffering was completed. It was later found out that the PC had booted into the wrong bootloader, named **E03**, leading to it not having access to the memory card where the captures were stored. When this was figured out, the author found the cause for why the captures got intertwined and started downloading and cleaning up to resolve **E04**.

After setting up new scripts, the satellite started having major pointing issues because of anomalies with the star tracker (**E05**), leading to many of the captures missing the target. During the active campaign, HYPSO-1 captured a total

of 50 captures, where 24 were attempts at the target area. Many of the Frohavet-captures suffered from being overexposed because of cloud cover or missed the target area, but the campaign was denoted as a success since it did show the concept of the observational pyramid with the successful captures. Figure 5.5 shows a selection of three captures showing three different types of captures. Figure 5.5a shows an overexposed capture that hit cloud cover, Figure 5.5b shows a good capture, and Figure 5.5c shows a capture with pointing issues caused by the star tracker anomalies.



(a) Overexposed low resolution RGB-composite of the image captured by the HSI on HYPSO-1 on the 2nd of April

(b) Low resolution RGB-composite of the the image captured by the HSI on HYPSO-1 on the 19th of April

(c) Low resolution RGB-composite of the the image captured by the HSI on HYPSO-1 on the 24th of April showing the pointing issues caused by star tracker anomalies.

Figure 5.5: A selection of three low-resolution RGB-composites of captures taken during the Frohavet capture campaign by the HSI on the HYPSO-1 satellite. The captures show one that is overexposed, one that is good, and one that has pointing issues.

5.1.3 Experiences and Improvements

Towards the end of the capture campaign, it was found that one of the causes of the PC reboots was buffering captures within a GS pass. Since the captures happened within the range of the GS, the buffering commenced as previous captures were downloading, thus reading and writing to the memory card on the PC at the same time (**E01**). The HYPSO team did not consider that the position of the satellite during captures affected the sequence needed to successfully execute a capture script. It was therefore added new functionality to the script generator where the mission planners could set a delay flag causing captures to delay buffering of the hyperspectral cube by 20 minutes. However, it became clear that multiple mission planners did not understand the utilization of the delay flag, meaning it was used for all captures, and not only the captures happening within range of the GS.

Additional changes to the script generator were the addition of the checksum of the hyperspectral cube to the metazip of the captures. Using checksums to compare the downloaded captures with the checksum of the capture on the satellite was a preventative measure to help locate the corresponding captures in case of situations like the one described in Figure 5.4 happen again. The metazip and hyperspectral cube were also merged to be one file instead of two separate files to make it easier to match the correct metazip to the capture. This change also used fewer files on the PC, leading to coordination of captures being easier. The star tracker was also taken out of the script temporarily while waiting for the anomalies to be corrected. Lastly, the reboots pushed a change to add better comments in the scripts such that it was easier for operators to figure out what commands the script had executed in case of reboots happening.

From the capture campaign, the team experienced that one should always plan for errors. The team did not plan operator responsibilities for the whole campaign, but rather made the planning on the weekly meeting as usual. This resulted in a lack of operators during the Easter holiday. The lack of operators led to the author being the sole operator for around two consecutive weeks. Having only one operator for an extended period of time is demanding and makes troubleshooting slow. The author had to make decisions without discussing them with the rest of the team. By having more operators available, the workload could have been split up, leading to a faster recovery.

Key points from this section are summarized below:

- The team encountered three unpredictable errors (**E02**, **E03**, and **E05**) during the campaign. The team also encountered two errors (**E01** and **E04**) caused by human error due to lacking an understanding of the system of systems.
- The script generator was updated to make it easier to resolve the issues that

were encountered in the future, but could not prevent them.
- The team attempted five captures in a single day. Not all captures were successful, but the team was capable of generating scripts for all the captures.

5.2 Estimation of the Total Theoretical Captures per Day

After the Frohavet capture campaign where the team planned for a total of five captures on two of the days, it was desired to estimate the total amount of captures possible to capture per day. Using only the NTNU GS, the satellite has a line of sight to the GS for an average of 10 passes per day, with a total theoretical duration of 100minutes [53]. The satellite link is shared with the tasks described in Table 4.1, leading to approximately 60 minutes used for the download of captures [53]. As mentioned in subsection 4.2.2, the S-band on HYPSO-1 has a data rate of 1.4 Mbps. The information rate was approximately averaging an information rate of 1 Mbps, thus having a theoretical downlink capacity per day of

$$(60 * 60) * \frac{1 * 10^6 bit}{s} = 3600Mb = 450MB$$

One full hyperspectral cube captured by HYPSO-1 is 153 MB, but is compressed to 80 MB or less using the CCSDS123 compression algorithm [53]. Assuming full cubes, the thoretically number of captures per day HYPSO-1 can downlink is

$$\frac{450MB}{80MB} = 5.625$$

The team can increase this capacity by booking passes using the KSAT GSs. Additionally, most captures are below 80 MB. Therefore, HYPSO-1 is capable of downlinking about 6 captures per day.

5.3 Kongsfjorden - The Second Campaign

The Kongsfjorden capture campaign happened from the 19th of May until the 29th of May. The capture campaign was executed as part of the Mission-oriented autonomous systems with small satellites for maritime sensing, surveillance and communication (MASSIVE) project, including other remote agents from the observational pyramid shown in Figure 2.1. The objective of the campaign for HYPSO-1 was to get several captures of target areas with ground truth measurements provided by other remote agents. The timeline of the campaign is shown in Figure 5.6. An updated error table is provided in Table 5.2.

Table 5.2: Updated description of error IDs

ID	Description
E01	Reboots caused by simultaneously reading and writing to the memory card on the PC
E02	MCS crash caused by the FTM and Task Scheduler running out of memory
E03	PC not accessing the SD card since it loaded into the wrong bootloader
E04	Renaming error on the OPU caused by multiple captures having the same name
E05	Star Tracker anomalies
E06	Pointing issues caused by not updating the TLE on the satellite

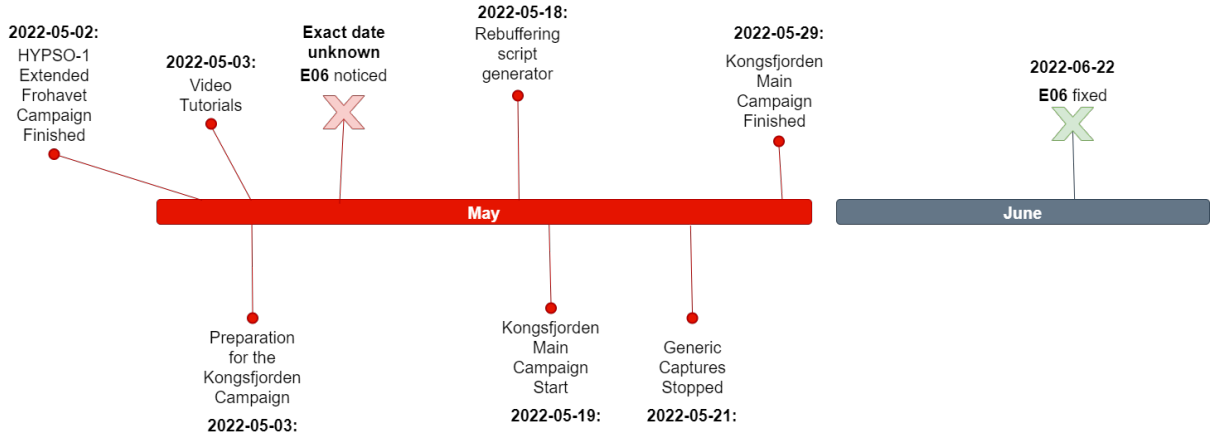


Figure 5.6: Timeline of the Kongsfjorden campaign.

5.3.1 Preparation

Ahead of the Kongsfjorden capture campaign, the team set up planning meetings to discuss what capture modes to use, how many captures to take, responsibilities, and test captures ahead of the campaign. It was decided upon taking one capture of the target area each day for a week ahead of the main campaign, and two captures of the target area during the main campaign. One member was chosen to be the main **mission planner**, and another member was selected to be the main **operator**. The author was not part of the staff during this campaign but was reachable for questions if necessary. As part of the preparation for the campaign, the author made video tutorials and flowcharts describing the operator's responsibilities and the most common troubleshooting. Each section in the flowcharts had a corresponding video describing how to execute the specific task

with explanations. Additionally, the author made a generator for making rebuffering scripts. Operators were not comfortable with making scripts from scratch themselves, thus meaning they manually buffered the files. However, manually buffering files meant that there was a high chance for buffering and download to happen at the same time since it was started within a pass, thus getting new instances of **E01**. The rebuffering generator took the folder name of the capture and the timestamp for rebuffering as input and outputted a script that would rebuffer the captures at the specified timestamp.

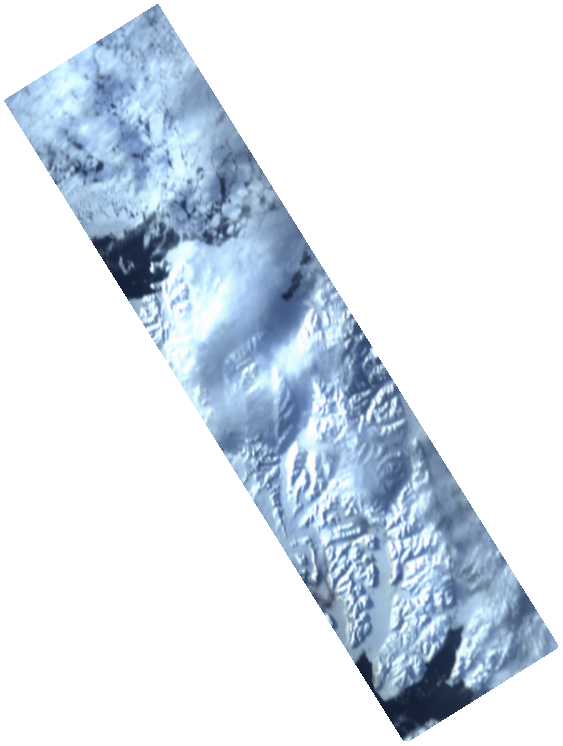
Safety copies of captures were still being used, thus meaning an operator had to delete captures after they were successfully downlinked. As part of preparing for the Kongsfjorden capture campaign, the author did not take on as many operator responsibilities as previously. Instead, the campaign team operated the satellite for an extended amount of time with the author being the operator when necessary. It quickly became clear that the cleanup on the satellite had been neglected, causing a total of 32 captures to be stored on the satellite. The author cleaned up the captures and told the team they need to clear up the satellite at least once a week.

One of the captures taken prior to the main capture campaigns is shown in Figure 5.7. The ground track in Figure 5.7b was calculated from the attitude determination telemetry from HYPSO-1, but did not correspond to the capture itself. The HYPSO team was not able to solve these pointing issues before the main capture campaign started.

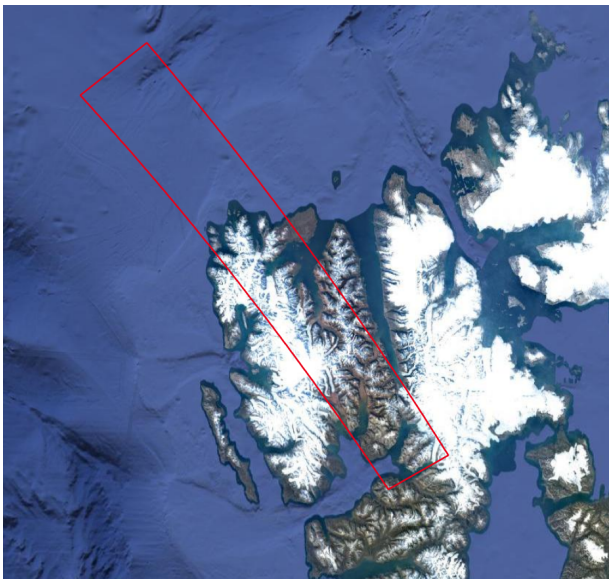
5.3.2 How the Workflow Affected the Main Campaign

During the main capture campaign, the HYPSO team planned for two captures of Kongsfjorden every day. The first capture was always during the morning passes, and the second capture was during the evening. In addition to these captures, it was planned captures of other targets. Throughout the main capture campaign, HYPSO-1 captured a total of 44 images. Of these, 27 captures were attempts at the target area. Instead of the planned two captures per day, the team started attempting to capture three captures per day. The third capture had a bigger off-nadir angle between the satellite and the target area. Nevertheless, it was decided to attempt as many captures as possible. In order to downlink all the captures, the team booked KSAT passes. Another measure by the campaign team was to intentionally aim the capture a bit off target as it seemed like the pointing error had a systematic error. On the 29^{th} of May, two of the three captures hit the intended target area, shown in Figure 5.8.

Several of the captures had buffering or downlink errors due to errors done during the coordination of the captures. These errors were solved by re-buffering the captures since they were stored in the safety copies. The main cause of the

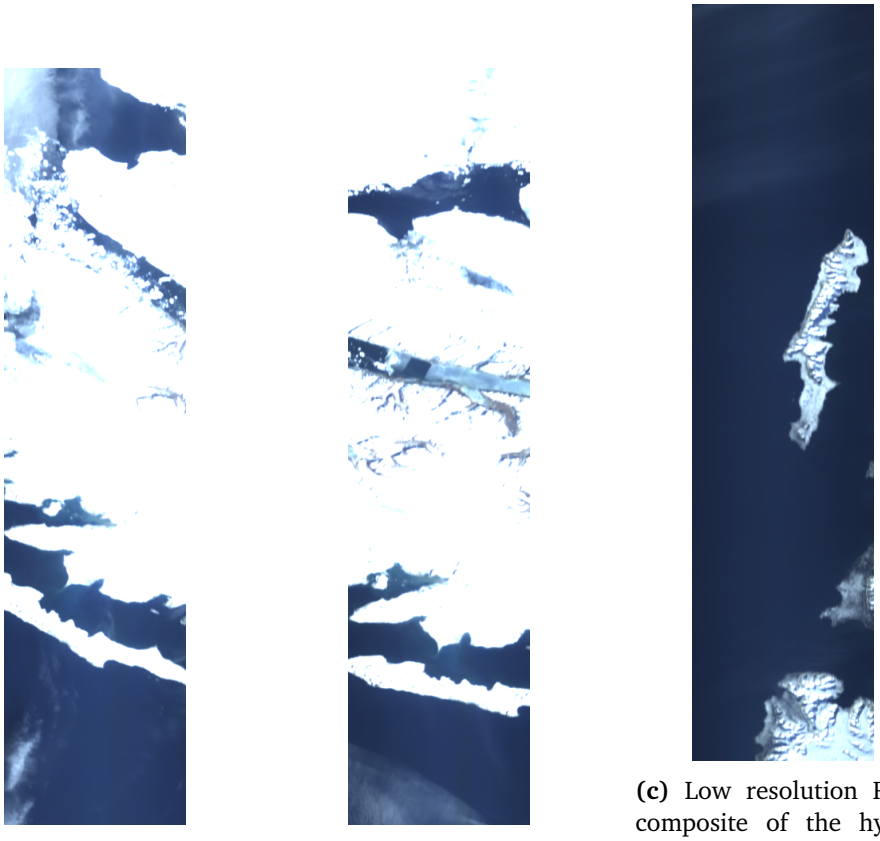


(a) Low resolution RGB-composite of an image captured by the HSI on HYPSO-1 on the 13th of May.



(b) Ground track of the low resolution RGB-composite calculated from the attitude determination telemetry of HYPSO-1 made by Dennis Langer

Figure 5.7: Low resolution RGB-composite of Kongsfjorden taken by HYPSO-1 on the 13th of May. The ground track incorrect.



(a) Low resolution RGB-composite of the hyperspectral cube captured by HYPSO-1 at 11:11:13 UTC on the 29th of May

(b) Low resolution RGB-composite of the hyperspectral cube captured by HYPSO-1 at 12:47:16 UTC on the 29th of May

(c) Low resolution RGB-composite of the hyperspectral cube captured by HYPSO-1 at 19:04:46 UTC on the 29th of May. The capture missed the intended target.

Figure 5.8: Low resolution RGB-composite of the captures taken of svalbard by HYPSO-1 on the 29th of May

buffering and downlink errors was a lack of coordination of the captures. The lack of coordination caused some captures to be too close in time and led to one of them overwriting the file on the PC before the downlink of the previous capture was completed. There were also errors done when merging the scripts together into one script file, thus resulting in lost captures. The problems were reduced by stopping the generic captures to instead focus on the targeted captures. Therefore, the generic captures were stopped on the 21st of May.

5.3.3 Experiences and Improvements

After the capture campaign, the cause of the **E06** pointing error was found. The star tracker was still not in use because of the anomalies found during the Frohavet campaign, meaning that the satellite relied upon the IMU for its attitude

determination. The IMU consisted of a gyroscope, accelerometer, and magnetometer. The magnetometer relied on geomagnetic models of the Earth to compare with its measurements [54]. Using positional knowledge, one could find the attitude of the satellite [54]. HYPSO-1 found its position by using Two Line Element (TLE) uploaded from the ground. On the 22nd of June, it was discovered that the automatic Two Line Element (TLE) upload had malfunctioned, thus leaving the satellite with outdated position information. The issue was resolved and the Two Line Element (TLE) upload was resumed. Afterward, the pointing accuracy increased notably.

During the capture campaign, HYPSO-1 captured up to six captures per day, which is at the theoretical limit of how many captures could be downlinked in a day, found in section 5.2. However, since a lot of the captures had overexposed areas, they were more aggressively compressed, thus resulting in smaller file sizes.

Key points from this section are summarized below:

- The Kongsfjorden campaign was performed without the author as a member of the campaign team. It was deemed a success, thus showing that the operator training and the developed tools eased operations.
- The generic captures were stopped due to an increased amount of targeted captures, thus showing script generation was simplified.
- The team captures up to six captures in a day.
- The team only suffered from one unresolved issue, but the issue lasted throughout the entire campaign.

5.4 Total Captures and Timing Accuracy

Over the thesis period spanning from the launch until the end of June 2022, HYPSO-1 has taken a total of 240 hyperspectral cubes. Figure 5.9 shows a timeline of all captures taken throughout the thesis period. The timeline clearly shows an increase in the number of captures over the project period, especially during the Kongsfjorden capture campaign. Figure 5.10 shows the total captures from each month, making the increasing trend up until May more noticeable, which averaged 2.94 captures per day.

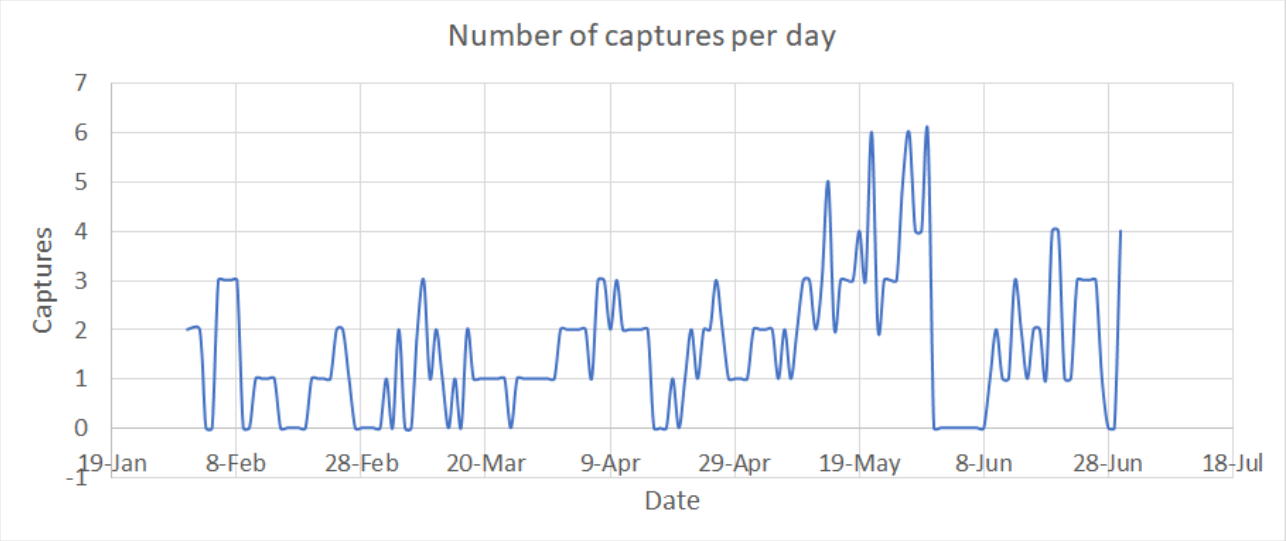


Figure 5.9: Capture timeline of all hyperspectral captures downloaded from the HYPSO-1 satellite from launch until then end of June 2022

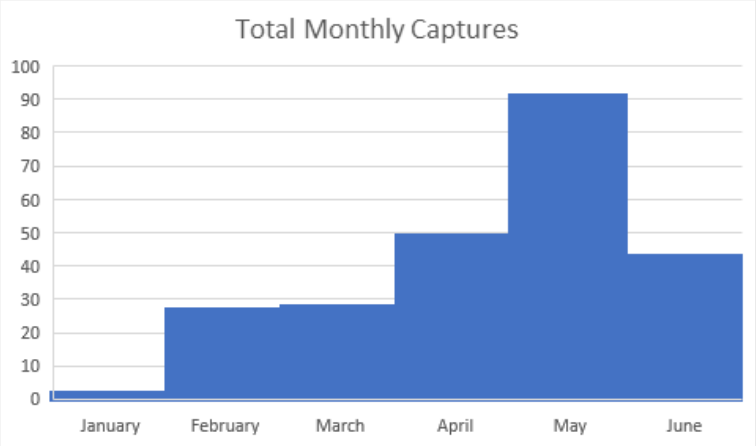


Figure 5.10: Monthly captures downloaded from the HYPSO-1 satellite

The timing accuracy of captures was collected from a set of randomly selected captures spanning different versions of the tools used for making scripts. The database only includes captures planned with a specified target timestamp for capture start. The results are shown in Figure 5.11. There was a sudden spike to 58seconds on the 18th and 19th of March that is not shown due to the axes being stretched out. The spike was caused by an error in calculating delays from the script generator.

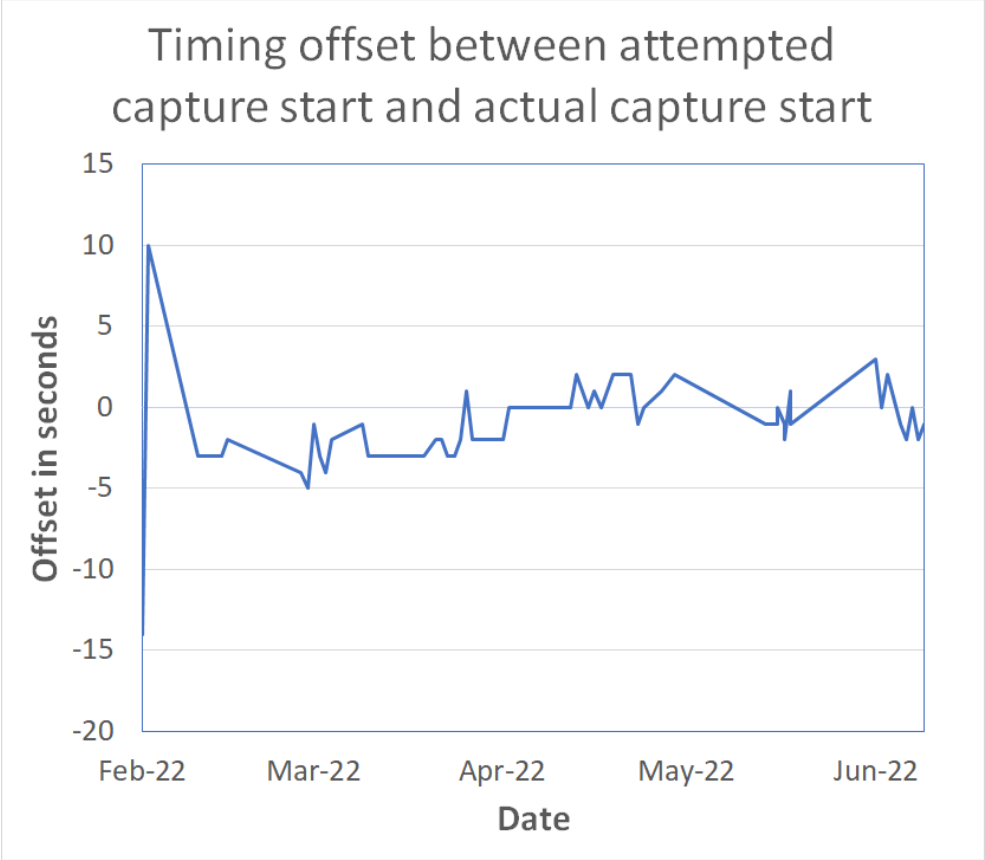


Figure 5.11: Timing offset between attempted start of capture and actual start of capture

Chapter 6

Discussion

6.1 Asynchronous Operations

As described in chapter 4, operations of the HYPSO-1 satellite was originally performed manually within a pass. This means operators waited for the satellite to be in the range of the GS and manually typed in commands. This way of operating the satellite was gradually replaced by utilizing the task scheduler in MCS, thus making most operations asynchronous, meaning scheduling commands ahead of satellite passes. Using automated download tasks, all troubleshooting and nominal operation of the satellite bus has become fully asynchronous. However, the payload still needs manual involvement. As described in subsection 4.4.2, the author made an attempt to automate the download of logs from the payload but was unsuccessful. Additionally, the inclusion of the safety copies meant that operators needs to manually delete old captures at least once a week.

The author did find a solution for how to automate the logs from the payload which is included in the upcoming software update, but the cleanup on the satellite still needs to be done manually. It is possible to make scripts to delete the files on the payload but automating deletion is unnecessarily risky. Something that could be done is to make a generator for cleanup scripts that require filenames and folder names as input, but checks the input and denies the user access to delete certain important files. This still requires the operators to have an overview of the current file structure on the payload, but that can be automatically downloaded using the same solution as the automated logs download. A fully asynchronous operational pattern of the HYPSO-1 satellite is therefore possible but was not achieved in this work.

For the automatic download of logs, one can utilize the Microsoft common parameters to disregard the error of the buffering command or set a time limitation of the task in order to mitigate the repercussions the task will have on other scheduled tasks. However, adapting the task to fail successfully does not seem like a good design and has the potential to confuse other operators.

6.2 Satellite Operator Team Structure

The HYPSO operational team being based upon having weekly coordination meetings makes members susceptible to confusion if they cannot participate. The operational workflow changes rapidly, thus meaning if a member cannot participate they might lose out on new knowledge and experiences. Additionally, the defined **mission planner**, **operator**, and **data analyst** roles have overlapping responsibilities. Most of the team members also had multiple of these roles at the same time, meaning **mission planners** were often also **operators** and/or **data analysts** and vice versa for the other roles. The fact that all **operators** were trained to also be **mission planners** clearly shows that the structure was not optimal. The reason for **operators** being trained as **mission planners** was to provide them with an understanding of how the scripts worked such that they were capable of troubleshooting. Vice versa, the **mission planners** had to operate the satellite to understand how different captures were affected by the position of the satellite to prevent errors like **E04**. Even though tools like the task scheduler and the script generator existed, the **operators** had to know how to use them. This directly contradicted the reasoning behind separating the responsibilities defined in the team structure in the first place. Additionally, all members had other commitments and roles in the HYPSO team as well. Therefore, the operational team structure defined in section 3.2 was not followed in practice. It was not possible with the small size of the team. Because of the small team size, it could be beneficial to instead define the skill set of each member instead. The IntelliSTAR team structure the HYPSO team based themselves upon was defined for established operational teams, and was maybe not suitable for such a young team.

The main problem could also be that satellite operations are normally described from a commercial perspective where operators have a functional pipeline and operators on duty at all times. This is not the case for academic organizations. The HYPSO team consisted of several part-time operators with other commitments to the organization. One can therefore not compare commercial satellite operations with academic satellite operations.

Throughout the two capture campaigns, the team experienced coordination and communication problems. Being a small team usually makes communication easier, but if the team is too small each member needs to have knowledge within multiple fields, thus making it hard to focus on a specific area. Having tools to automate parts of the chain only removes the need for specific knowledge of how the part is made, it does not remove the need to understand how the part works as part of the chain. Extending the team could improve the situation by allowing more specific roles. The problem could also come from the high turnover of personnel, meaning the best solution would be to focus on improving the training to give all members a better understanding of the system of systems.

6.3 Operator Training

Operators had to learn mission planning as part of their training in order to be able to troubleshoot if errors occurred, thus spanning a wide range of tasks. When operator training started, it was expected for them to have read the documentation, but it became clear that members from different backgrounds and fields of study did not get the same foundation from the documentation. Therefore, they made their own bullet-point list describing how to execute different tasks. The bullet-point list was then used instead of the documentation because it gave the direct answer. The documentation prepared by the author did not give direct answers since it instead explained the functionality. The bullet-point list did not provide an understanding of the system of systems. When the video tutorial was made, the problem was partially solved. It was probably because the video tutorial showed exactly the sequence of commands necessary to execute a specific task, but also explained why it did so. Therefore, the video tutorials covered the areas of both the bullet-point list and the documentation. It might also be because it gave operators an alternative learning method, thus covering more of the learning styles described in Figure 2.7.

6.4 Simplification of Operations

It is clear that the work done in this thesis has made operating the HYPSO-1 satellite easier. The first capture script took three days to make without pointing to a specific target whilst requiring special knowledge about `hypso-cli`, the payload, and the M6P satellite bus. Additionally, the first capture scripts were manually uploaded and payload data was manually downlinked within a pass. At the end of the thesis period, the team planned up to five captures a day for a week at a time in the timespan between the weekly meeting and the scheduled upload of scripts the same evening. The upload of scripts and downlink of captures was done automatically by using the task scheduler. Figure 5.9 shows an increase in the number of captures throughout the project period. Additionally, the diagram only shows the number of captures, not their size. In the beginning, captures were smaller to test camera configurations. Then, the generic captures without specific target areas were attempted daily from the 23rd of February until the 21st of May. The fact that the generic captures were aborted due to an increased amount of targeted captures clearly shows that script generation had been simplified.

The testing of scripts on the ground was seen as excessive with the addition of the script generator, but the team still experienced a lot of errors throughout the whole project period. The script generator was tested using the LidSat after every new addition, but the testing was not thorough enough. By generating a script and testing it on the LidSat, the only thing being tested was that the sequence was functional for that one script. The problems experienced did not concern the sequence or commands being wrong, but rather the utilization of it being wrong.

The addition of the safety copies led to a need for using different names on all captures. When tested on the LidSat this was done, but when it was used to generate capture scripts for HYPSO-1 this was not done. The testing on LidSat did not cover variables like the satellite position, thus leading to the **E04** error experienced in the Frohavet capture campaign. These factors are difficult to test for. One way of solving it would be for other operators to verify all capture scripts before their upload, but that is time-consuming in an already resource-constrained team. Therefore, it means the training of operators and communication with the mission planners should be improved to make all members aware of these operational dependencies.

One of the main problems with operational dependencies was that the **mission planners**, who made the scripts, quickly adopted changes to make the script generation easier, but did not easily adopt the changes made to account for the operational dependencies like naming conventions and the delay flag to prevent occurrences of **E04**. These changes were understood by some operators that had to correct the errors after they occurred, but not by the **mission planners** who had to make the preventative measures. These problems might have been resolved or reduced if the author focused more on the operator training instead of simplifying the tools to be used. By simplifying the utilization of the tools, the author enabled **mission planners** to make scripts without having a fundamental understanding of why it works. As mentioned, not all operators that had to fix the errors that occurred from operational dependencies understood what caused the problem. They simply followed the bullet-point list or the video tutorial describing how to correct it. I.e., an operator would figure out that a capture was not downloaded correctly due to a reboot, but not understand that the reboot was caused by not using the delay flag during mission planning, thus resulting in simultaneously reading and writing to the same memory on the PC. However, correctly classifying the cause of a reboot was hard and was often done by the author.

A preventative measure could be to discuss all reboots in the weekly meetings such that all team members get an explanation of the cause. Another measure could be to improve the tools, i.e., making the script generator calculate the position of the satellite during the capture and from that calculate if the delay flag is necessary or not. This is part of a trade-off between how much knowledge an operator should have and how much time should be used on automating operations.

If operators have less knowledge of the system it will become harder for them to notice and locate errors, just like [8] stated when using the tools developed by NASA. Therefore, overlapping roles were seen as a necessity to obtain an understanding of the rapidly changing system of systems. It is not possible for a single operator to be an expert in all fields, but it is possible to have a fundamental understanding of the system itself.

Operations of the HYPSO-1 satellite should be further automated, but it is important for operators to understand how to best operate the satellite while the tools are being developed in order to continuously utilize the satellite. When mission planning becomes a full pipeline the uncertainties associated with human involvement are removed, leading to a deterministic output. When a full pipeline is achieved, a more strict team structure can be defined and utilized. If an error is found, a **subsystem specialist** can fix the error in the pipeline. Until the full pipeline is achieved it is impossible to calculate or reason to find the best balance. It needs to be found by being open to changes and applying experiences.

6.5 Answering the Research Questions

6.5.1 RQ1 - Task Scheduling

It is possible to work fully asynchronously with the operations of HYPSO-1, meaning that one never has to wait for the satellite to be in reach of a GS to execute specific tasks. Working fully asynchronously is not achieved for the HYPSO-1 satellite at this time, but is possible. All monitoring and commands to the satellite platform is possible to do asynchronously, but commands on the payload that take more than three seconds to succeed are not possible to schedule using the task scheduler. In order to make the payload compatible with the three-second timeout of the task scheduler, the payload needs a software update to return an acknowledgment at the start of the command instead of at the end.

6.5.2 RQ2 - Reducing the Amount of Errors in Capture Scripts

It is not possible to prevent all types of errors while reducing the amount of testing on the ground. It is possible to prevent typos and logic errors within spacecraft scripts while not having to test the scripts by using tools to generate them, but these simple tools do not account for all faults that can be experienced. Capture scripts can miss the target area due to wrongful pointing parameters or erroneous capture timing. The tools developed in this thesis automated parts of the pipeline on the ground necessary to run capture scripts on the satellite, but members still had to use the tools correctly. The script generator and the generic task scheduler tasks removed all human error contained within these scripts but did not remove the human error when using these tools. To further remove human error the team needs to extend the tools on the ground to be a full pipeline.

To achieve a full pipeline on the ground, the team needs to make additional tools. The input to the pipeline should be a target area. Then, the pipeline should provide a set of possible capture timestamps of that target area which the **mission planner** can select. The automated parts include calculating the pointing parameters, generating the capture scripts, and scheduling uploads of the scripts

using the task scheduler. If all these parts are automated, errors are contained to unpredictable errors and the ability to select the wrong target area.

The HYPSO-1 satellite encountered multiple unpredictable errors like the star tracker anomalies and unintentional reboots. The number of errors can therefore be reduced to only these unpredictable errors while not having to test on the ground, given that a full pipeline from the selection of the target areas to the upload of scripts to the satellite is implemented. Until the pipeline is achieved, the HYPSO team needs to rely on coordination and communication to reduce the number of errors.

6.5.3 RQ3 - Training of Operators

The project organization should facilitate operator training by accommodating multiple learning styles. This is especially important for teams with members from different fields of study. In an academic team without a fully automatic pipeline of script generation - like HYPSO - it is important for members to have a fundamental understanding of the total system of systems since the members have to use the supporting tools correctly. If a full pipeline is achieved, members would not need to understand the system of systems in order to utilize the satellite payload, but it would be required for troubleshooting. Since **RQ2** concluded that unpredictable errors cannot be totally removed, it is important to have a fundamental understanding of the satellite to resolve issues like **E06**, where the automatic TLE upload malfunctioned. The academic project organization should therefore facilitate operator training by focusing on understanding the system of systems, and accommodating different learning styles. The project organization should also be aware of caveats connected with the learning methods, i.e., using video tutorials resolves one specific issue without necessarily providing an answer to the root cause of the issue.

Chapter 7

Conclusion

This thesis serves as the beginning of automating the utilization of the HYPSO-1 satellite in order to maximize payload utilization. It describes the functional baseline when the HYPSO-1 satellite was launched and shows the continuous improvements throughout the thesis period. It describes the steps going from doing everything manually to working almost fully asynchronously, meaning scheduling all commands to the satellite instead of waiting for it to be in the range of a GS. It shows a variety of problems the team was faced with, how they were fixed, and preventative measures that were implemented to prevent the errors from happening again. The method used focused on rapid development and fixing problems when they were encountered instead of testing for all possibilities before deployment. The thesis found that it is possible to work fully asynchronously with the operations of the HYPSO-1 satellite, but did not achieve it during the thesis period.

HYPSO-1 utilized spacecraft scripts to execute captures automatically. In order to reduce the number of errors during capture whilst also reducing the amount of testing on the ground, it was made a script generator that took high-level input to generate low-level output. The addition of the script generator reduced the time and knowledge needed to make capture scripts, and the number of errors in the scripts by having a deterministic output. The testing on the ground became excessive because the output of the script generator was deterministic, thus only needing to be tested once. However, it only prevented errors in the script itself, not errors in the utilization of the script generator. It was found that circumstantial errors were impossible to predict. Therefore, it was not possible to prevent all errors. Furthermore, the supporting tools used in the operations of the HYPSO-1 satellite should be integrated into a pipeline to prevent uncertainties associated with human interference.

During the development of the supporting tools, it was found that the project organization should adapt their operator training to cover multiple learning styles. The work also found it important for the **mission planners** to operate the satellite in order to understand how parameters like the position of the satellite

could affect the capture. It becomes a trade-off between the level of automation and the operator's capability of fixing the error when they occur. In conclusion, during the development of supporting tools to operate the satellite, there is a need for understanding the system of systems, thus resulting in it being a necessity with overlapping roles.

Overall, the HYPSO-1 satellite captured 240 hyperspectral cubes with the HSI during the thesis period. Script generation has gone from taking three days for a single capture down to less than one day to plan captures for an entire week. The accuracy of captures has increased from not being aimed towards a specific target at all, to within two seconds of the desired start time. HYPSO-1 reached an average of 2.94 captures per day in the month of May and has captured up to six captures in a single day. In conclusion, there are still improvements to be done concerning the capacity, quality, and resource utilization for operations of the HYPSO-1 satellite, but the payload utilization has been drastically improved over the thesis period.

7.1 Future Work

Future work includes executing a software update of the satellite such that the payload does not require manual download and deletion of logs. It also includes continuing to improve the timing accuracy of captures. More of the necessary logic used for mission planning should be automated, but thoroughly described in the documentation. The tools should be integrated into a pipeline that also calculates pointing parameters and accommodates for position dependencies of the satellite. Lastly, a tool like Generic Spacecraft Analyst Assistant (GenSAA) should be developed to assist in troubleshooting.
Other upgrades to be done concerns the capacity of captures. Features like the one proposed in [53], using the compression ratio of captures to determine if they should be buffered or deleted, should be added.

Acknowledgement

This thesis is a product of so much more than only my contribution. There have been so many people involved both prior- and during my thesis that deserves to be mentioned. I want to use this opportunity to thank my supervisors, fellow students, team members, and friends.

First and foremost, thank you Milica Orlandic and Roger Birkeland for being my supervisors. You not only helped me in my thesis, you felt like my colleagues by including me in decision making, meetings, and valuing my feedback. This thesis would not have been possible without your directions when I was struggling to find a path to pursue. Thank you, Roger, for being so enthusiastic about the operations of the satellite.

I also want to thank Evelyn Honoré-Livermore and Sivert Bakken for helping me, sharing my enthusiasm, motivating me, and giving me feedback. Thank you, Evelyn, for being an unbelievable project manager. Thank you, Sivert, for being an incredible Software Lead. I also want to thank you for taking on operator responsibilities such that I could work on my thesis and get some time off.

I want to thank everyone who has contributed to the operator team. There are a lot of people involved, but I want to give a special thanks to Esmeé Oudijk, Fredrik Gran-Jansen, Joseph Garret, Roger Birkeland, Milica Orlandic, Dennis Langer, Mariusz Grøtte, and Sivert Bakken for their contributions to operating the satellite.

Thanks to all my fellow students at the SSL, A491, and Orbit NTNU. It has been a pleasure working with all of you, learning about your fields of study and theses, making memes, playing table tennis with you, and eating lunch on the roof.

I would like to show my appreciation for everyone involved in HYPSO. The utilization and performance of the payload would not have been possible without all the hard work you have put in over the years. It has been a pleasure to use all that work in an operational context. I would also like to thank all members of MASSIVE for the opportunity of being part of exciting capture campaigns.

Lastly, I would like to thank Astrid Christine Zieritz, Sindre Sletten Øyen, Mats Even Jørgensen, Even Tobias Eriksen, Håkon Kindem, and Torbjørn Bravold for providing me with a space to think about other things than HYPSO-1.

Bibliography

[1] M. Sufyan, N. Abd Rahim, M. Aman, C. K. Tan and S. Raihan, 'Sizing and applications of battery energy storage technologies in smart grid system: A review,' *Journal of Renewable and Sustainable Energy*, vol. 11, p. 014 105, Feb. 2019. DOI: 10.1063/1.5063866.

[2] Grafana.com. 'We're on a mission.' (2022), [Online]. Available: https://grafana.com/about/mission/.

[3] Grafana.com. 'Why grafana?' (2022), [Online]. Available: https://grafana.com/grafana/.

[4] L. Castro, 'On the development life cycle of distributed functional applications: A case study,' Jan. 2010.

[5] ITU. 'Nomenclature of the frequency and wavelength bands used in telecommunications.' (Aug. 2015), [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/v/R-REC-V.431-8-201508-I!!PDF-E.pdf.

[6] T. E. S. Agency. 'Polar and sun-synchronous orbit.' (Mar. 2020), [Online]. Available: https://www.esa.int/ESA_Multimedia/Images/2020/03/Polar_and_Sun-synchronous_orbit.

[7] NASA. 'Sputnik 1.' (Apr. 2022), [Online]. Available: https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1957-001B.

[8] D. A. Thurman, D. M. Brann and C. M. Mitchell, 'Operations automation: Definition, examples, and a human-centered approach,' *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 43, no. 3, pp. 194–198, 1999. DOI: 10.1177/154193129904300314. eprint: https://doi.org/10.1177/154193129904300314. [Online]. Available: https://doi.org/10.1177/154193129904300314.

[9] S. Watson, B. Whitton, S. Higgins, H. Paerl, B. Brooks and J. Wehr, 'Harmful algal blooms,' in Jun. 2015, pp. 873–920, ISBN: 9780123858764. DOI: 10.1016/B978-0-12-385876-4.00020-7.

[10] J. Holland. 'Norway breaks seafood export records in 2021.' (Jan. 2022), [Online]. Available: https://www.seafoodsource.com/news/supply-trade/norway-breaks-seafood-export-records-in-2021.

[11] S. Sentralbyrå. 'External trade in goods.' (Jun. 2022), [Online]. Available: https://www.ssb.no/en/utenriksokonomi/utenrikshandel/statistikk/utenrikshandel-med-varer.

[12] Kongsberg. 'Satellite images of harmful algae bloom in norway.' (May 2019), [Online]. Available: https://www.kongsberg.com/no/newsandmedia/news-archive/2019/satellite-imagery-of-harmfull-algae-bloom/.

[13] O. Kalden and C. Bodemann, 'Eo satellite operations challenges and training simulator requirements,' in *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, 2013, pp. 1145–1150. DOI: 10.1109/RAST.2013.6581175.

[14] M. H. Azami, N. Örger, V. Schulz, T. Oshiro and M. Cho, 'Earth observation mission of a 6u cubesat with a 5-meter resolution for wildfire image classification using convolution neural network approach,' *Remote Sensing*, vol. 14, p. 1874, Apr. 2022. DOI: 10.3390/rs14081874.

[15] E. Honoré-Livermore, 'Conops,' Concept of Operations for the HYPSO-1 Satellite, 2018.

[16] NTNU. 'Research at ntnu amos.' (Jun. 2022), [Online]. Available: https://www.ntnu.edu/amos/research.

[17] NTNU. 'Ntnu amos - centre for autonomous marine operations and systems.' (Jun. 2022), [Online]. Available: https://www.ntnu.edu/amos.

[18] S. Bakken, 'Development of a small satellite with a hyperspectral imaging payload and onboard processing for ocean color,' Doctoral Thesis, Norwegian University of Science and Technology, Mar. 2022.

[19] Elizabeth Frances Prentice, Evelyn Honoré-Livermore, Sivert Bakken, Roger Birkeland, Marie Bøe Henriksen, Amund Gjersvik, Martine Hjertenæs, Tor Arne Johansen, Fernando Aguado Ageleta, and Fermín Navarro Medina, 'Assembly, integration, and testing strategy of a hyperspectral imaging cubesat, hypso-1,' To be submitted.

[20] https://nanoavionics.com/. 'About us.' (Jun. 2022), [Online]. Available: https://nanoavionics.com/about/.

[21] https://nanoavionics.com/. '6u nanosatellite bus m6p.' (Jun. 2022), [Online]. Available: https://nanoavionics.com/small-satellite-buses/6u-nanosatellite-bus-m6p/.

[22] GOMSpace. 'Cubesat space protocol (csp).' (Jan. 2011), [Online]. Available: https://bytebucket.org/bbruner0/albertasat-on-board-computer/wiki/1.%5C%20Resources/1.1.%5C%20DataSheets/CSP/GS-CSP-1.1.pdf?rev=316ebd49bed49fdbb1d74efdeab74430e7cc726a.

[23] E. R. Jahren, 'Design and implementation of a reliable transport layer protocol for nuts,' Masters Thesis, Norwegian University of Technology and Science, Jun. 2015.

[24] N. Avionics, 'Nanomcs - na-nanomcs-icd-r4,' Interface Control Document provided to customers, 2021.

[25] R. Birkeland, 'Manual for hypso-1,' Manual for HYPSO-1 version 1.5, May 2022.

[26] J. R. Wertz, *Space Mission Engineering: The New SMAD*. Hawthorne - CA 90250 USA: Microcosm Press, 2015, Second Printing.

[27] R. Birkeland, 'Hypso - learning to use,' Internal learning-to-use document in HYPSO, May 2021.

[28] S. Berg, 'Environmental qualification and development procedures to increase chance of mission success for a university cubesat,' A report written by the author as part of a course subject at NTNU, Jan. 2022.

[29] ECSS. 'Ecss-m-st-10c rev. 1.' (Mar. 2009), [Online]. Available: https://ecss.nl/standard/ecss-m-st-10c-rev-1-project-planning-and-implementation/.

[30] ECSS. 'Tailoring.' (Jun. 2020), [Online]. Available: https://ecss.nl/standard/ecss-s-st-00-02c-draft-1-tailoring-15-june-2020/.

[31] S. Alsaqqa, S. Sawalha and H. Abdel-Nabi, 'Agile software development: Methodologies and trends,' *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, p. 246, Jul. 2020. DOI: 10.3991/ijim.v14i11.13269.

[32] M. A. G. Darrin and W. S. Devereux, 'The agile manifesto, design thinking and systems engineering,' in *2017 Annual IEEE International Systems Conference (SysCon)*, 2017, pp. 1–5. DOI: 10.1109/SYSCON.2017.7934765.

[33] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza and Z. Sheikh, 'Agile software development: Impact on productivity and quality,' Jul. 2010, pp. 287–291. DOI: 10.1109/ICMIT.2010.5492703.

[34] A. Shrivastava, I. Jaggi, N. Katoch, G. Deepali and S. Gupta, 'A systematic review on extreme programming,' *Journal of Physics: Conference Series*, vol. 1969, p. 012 046, Jul. 2021. DOI: 10.1088/1742-6596/1969/1/012046.

[35] W. Fabrycky, 'Systems analysis: Its proper utilization in systems engineering education and practice,' Jun. 2015.

[36] N. Sue, 'Knowledge transfer and learning: Problems of knowledge transfer associated with trying to short-circuit the learning cycle,' *Journal of Information Systems and Technology Management*, vol. 2, Dec. 2005. DOI: 10.4301/S1807-17752005000300003.

[37] C. van der Horst and R. Albertyn, 'The importance of metacognition and the experiential learning process within a cultural intelligence–based approach to cross-cultural coaching,' *SA Journal of Human Resource Management*, vol. 16, May 2018. DOI: 10.4102/sajhrm.v16i0.951.

[38] D. Kolb and A. Kolb, *The Kolb Learning Style Inventory 4.0: Guide to Theory, Psychometrics, Research Applications*. Jan. 2013.

[39] E. C. L. D. S. J. Hartley. 'Spacecraft control center automation using the generic inferential executor.' (Nov. 1996), [Online]. Available: https://www.semanticscholar.org/paper/Spacecraft-control-center-automation-using-the-Hartley-Luczak/5f1cfc4eaa5a63f44ace76c19c8f3b4d828b291c.

[40] J. L. Anderson, F. J. Kurfess and J. Puig-Suari, 'A Framework for Developing Artificial Intelligence for Autonomous Satellite Operations,' in *ESA Special Publication*, H. Lacoste, Ed., ser. ESA Special Publication, vol. 673, Sep. 2009, 4, p. 4.

[41] T. Gathmann and L. Raslavicius, 'Systems approach to the satellite operations problem,' *IEEE Aerospace and Electronic Systems Magazine*, vol. 5, no. 12, pp. 20–24, 1990. DOI: 10.1109/62.61954.

[42] R. Torres, S. Løkås, D. Geudtner and B. Rosich, 'Sentinel-1a leop and commissioning,' in *2014 IEEE Geoscience and Remote Sensing Symposium*, 2014, pp. 1469–1472. DOI: 10.1109/IGARSS.2014.6946714.

[43] R. Blinovaitė, 'Hyspo-1 satellite / commissioning end report,' Commissioning report prepared by Nano Avionics delivered to HYPSO, May 2022.

[44] Y. Yang, 'Spacecraft attitude determination and control: Quaternion based method,' *Annual Reviews in Control*, vol. 36, pp. 198–219, Dec. 2012. DOI: 10.1016/j.arcontrol.2012.09.003.

[45] D. J. C. Shu Lin, *Error Control Coding*, ser. Second Edition. Pearson Prentice Hall, 2004, ISBN: 0-13-017973-6.

[46] N. Avionics, 'Payload controller m6p icd,' ICD of the payload controller used on the HYPSO-1 satellite, 2021.

[47] R. Wisniewski and P. Kulczycki, 'Slew maneuver control for spacecraft equipped with star camera and reaction wheels,' *Control Engineering Practice*, vol. 13, pp. 349–356, Mar. 2005. DOI: 10.1016/j.conengprac.2003.12.006.

[48] S. Berg, 'Hypso-um-010 $pc_f c_s cripting$,' User manual introducing the Payload Controller, Flight Computer, the csp txrx command, and timing considerations. It also includes script templates., Feb. 2022.

[49] L. Alminde, M. Bisgaard, D. Bhanderi and J. D. Nielsen, 'Experience and methodology gained from 4 years of student satellite projects,' Jul. 2005, pp. 94–99, ISBN: 0-7803-8977-8. DOI: 10.1109/RAST.2005.1512542.

[50] L. Berthoud, M. Sartwout, J. Cutler, D. Klumpar, J. Larsen and J. Nielsen, 'University cubesat project management for success,' English, in *Proceedings of the AIAA/USU Conference on Small Satellites*, Small Satellite Conference 2019 : Driving a Revolution ; Conference date: 03-08-2019 Through 08-08-2019, 2019.

[51] Microsoft. 'About$_{c}ommonparameters$.' (Aug. 2022), [Online]. Available: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_commonparameters?view=powershell-7.2.

[52] E. A. Oudijk, 'Campaign for hyperspectral data validation in north atlantic coastal waters,' Accepted for The Sound of Whispers 2022 conference, May 2022.

[53] J. L. Garret, 'On-board characterization of hyperspectral image exposure and cloud coverage by compression ratio,' Paper accepted to The Sound of Whispers' 22 conference. R. Birkeland, S. Berg, M. Orlandic are co-authors., Apr. 2022.

[54] S. Carletta, P. Teofilatto and M. S. Farissi, 'A magnetometer-only attitude determination strategy for small satellites: Design of the algorithm and hardware-in-the-loop testing,' 2020.

Appendix A

Additional Material

Code listing A.1: Pseudocode of an Flight Computer (FC) script of a slew maneuver generated by the script generator

```
#OPU active: 07/08/22 10:43:34 - 07/08/22 11:43:34
# Slew pointing script for targeted capture. Mid capture at 1657277040

#read rtc time

#Turn on star tracker 30 minutes ahead of capture script
#turn on star tracker at 07/08/22 10:09:34 UNIX: 1657274974
script delayuntil 1657274974

#elevating role

#Power on star tracker
script delay 1000

#Turn on accurate determination

#Script start at: 07/08/22 10:39:34 UNIX: 1657276774
# script start 866 seconds before mid-capture
script delayuntil 1657276174

# elevating role again in case it was changed

#set priority of the pointing target to 0

#defining target to be SLEW pointing. It needs to init at least 320sec in advance

# Set priority to 0 in safe-mode

#Set start time of the slew maneuver

#Set duration of the slew maneuver

#Set start off-nadir degree with respect to the y-axis in body frame

#Set stop off-nadir degree with respect to the y-axis in body frame

# Set pointing priority to highest priority => starting pointing when timestamp hits

#Waiting until 266 seconds before mid-capture
```

```
script delayuntil 1657276774

#Wait one minute for kalman filter to estimate bias

#Setting ADCS telemetry rate to 4Hz for 453 seconds. Needs to be lower than 10minutes

#Set EPS telemetry rate at 0.1Hz for 17 min

#Read rtc time

# waiting 453 seconds
script delay 453000

#Setting back to normal pointing by disabling this target

#Set ADCS telemetry rate back to normal rate

#Set ADCS mode back to coarse determination mode

#Power off the star tracker

#end script
#================================================================================
```

Code listing A.2: Pseudocode of an Payload Controller (PC) script of a slew maneuver generated by the script generator

```
#OPU active: 07/08/22 10:43:34 - 07/08/22 11:43:34
# Slew pointing script for targeted capture. Mid capture at 1657277040

#read rtc time

#Turn on star tracker 30 minutes ahead of capture script
#turn on star tracker at 07/08/22 10:09:34 UNIX: 1657274974
script delayuntil 1657274974

#elevating role

#Power on star tracker
script delay 1000

#Turn on accurate determination

#Script start at: 07/08/22 10:39:34 UNIX: 1657276774
# script start 866 seconds before mid-capture
script delayuntil 1657276174

# elevating role again in case it was changed

#set priority of the pointing target to 0

#defining target to be SLEW pointing. It needs to init at least 320sec in advance

# Set priority to 0 in safe-mode

#Set start time of the slew maneuver

#Set duration of the slew maneuver
```

```
#Set start off-nadir degree with respect to the y-axis in body frame

#Set stop off-nadir degree with respect to the y-axis in body frame

# Set pointing priority to highest priority => starting pointing when timestamp hits

#Waiting until 266 seconds before mid-capture
script delayuntil 1657276774

#Wait one minute for kalman filter to estimate bias

#Setting ADCS telemetry rate to 4Hz for 453 seconds. Needs to be lower than 10minutes

#Set EPS telemetry rate at 0.1Hz for 17 min

#Read rtc time

# waiting 453 seconds
script delay 453000

#Setting back to normal pointing by disabling this target

#Set ADCS telemetry rate back to normal rate

#Set ADCS mode back to coarse determination mode

#Power off the star tracker

#end script
#================================================================================
```

Simen Berg

Maximizing Payload Utilization of the HYPSO-1 Satellite

NTNU
Norwegian University of
Science and Technology