Investigations into the Filter-Wheel of the CoCa Experiment on the Comet Interceptor Mission

Master Thesis
Faculty of Science, University of Bern

handed in by

Linus Leo Stöckli

2021

Supervisor

Prof. Dr. Nicolas Thomas

University of Bern

Faculty of Science

Physics Institute

Investigations into the Filter-Wheel of the CoCa Experiment on the Comet Interceptor Mission

Master Thesis

Author:
Linus Leo Stöckli

supervised by:
Prof. Dr. Nicolas Thomas

October 1, 2021

Abstract

Comet Camera (CoCa) is a scientific instrument currently being developed by the University of Bern for the upcoming Comet Interceptor mission. It will be able to take high resolution images at four different wavelengths of a comet nucleus. The first part of this project focuses on the control algorithm of the filter wheel. Second and third order motion profiles haven been compared and after the advantages of the third order motion profile have been evaluated an open loop controller has been implemented in the prototype and tested.
The second part of this project is dedicated to the centers and bandwidths of the four filters used by CoCa. The reflectance of the comet was simulated using a *Hapke* model and the filter bandwidths have been determined by minimising motion smear and maximising Signal to Noise Ratio (SNR). One of the selected options for the filters: BLU centered at 465.4 nm (width 219.7 nm), ORA centered at 650.0 nm (width 149.5 nm), RED centered at 781.6 nm (width 113.7 nm), NIR centered at 946.9 nm (width 217.0 nm).

Contents

Acronyms

ADC Analog Digital Converter. 6, 7, 12, 31

ASIC Application Specific Integrated Circuits. 6

BLDC Brushless Direct Current Motor. 5–8, 13, 14

CaSSIS Colour and Stereo Surface Imaging System. 4

CLB Configurable Logic Block. 6

CMOS Complementary Metal-Oxide-Semiconductor. 4

CoCa Comet Camera. 3–6, 8, 12, 17, 19, 20, 31

DNC Dynamically New Comet. 3, 31

DPM Digital Processing Module. 15

EMF Electromotive Force. 7

ESA European Space Agency. 3

FPGA Field Programmable Gate Array. 5, 6, 8, 12, 15, 31

HK Housekeeping. 5, 10

HMC Halley Multicolor Camera. 17

LUT Look Up Table. 6, 11, 31

LVDS Low Voltage Differential Signaling. 15, 31

MICAS Miniature Integrated Camera Spectrometer. 17

NAC Narrow Angle Camera. 4, 17

OSIRIS Optical, Spectroscopic, and Infrared Remote Imaging System. 4, 17

PLB Processor Local Bus. 12

PMSM Permanent Magnet Sync Motor. 7, 8

PWM Pulse Width Modulation. 8, 12, 14, 16, 31

SNR Signal to Noise Ratio. 19, 24–31

SPI Serial Peripheral Interface. 12

UART Universal Asynchronous Receiver Transmitter. 5, 6, 10, 12, 15, 31

VHDL Very High-Speed Integrated Circuit Hardware Description Language. 6, 10

WAC Wide Angle Camera. 4

Glossary

ARIEL Ariel, the Atmospheric Remote-sensing Infrared Exoplanet Large-survey, is an ESA mission to be launched in 2029 and will study what exoplanets are made of, how they formed and how they evolve. 3

Deep Space 1 Deep Space 1 was a NASA mission that launched in 1998 and flew by comet 19P/Borrelly in 2001. 17

Giotto Giotto was an ESA mission, launched in 1985, that flew by comet 1P/Halley. 17

MANIaC The Mass Analyzer for Neutrals and Ions at Comets instrument is a mass spectrometer to sample the gases released from the comet on-board the Comet Interceptor to be launched in 2029. 3

ModelSim ModelSim is a multi-language simulation tool by MentorGraphics. It can be used in conjunction with software synthesis tools like Xilinx ISE to develop designs for FPGAs. 6

NIM The Neutral gas and Ion Mass spectrometer (NIM) is one of six instruments of the Particle Environment Package (PEP) on board the JUICE spacecraft, planned to launch in 2022. 5, 6, 8, 12

Python Python is a high level programming language, which is widely used amongst scientists due to its easy access, platform independence and the large amount of libraries for various applications. 5, 15, 26

Rosetta Rosetta was an ESA mission, launched in 2004, that flew to 67P/Churyumov-Gerasimenko. It carried the scientific imaging system OSIRIS along with the lander Philae. 2, 17, 31

Stardust Stardust was a NASA mission that launched in 1999 and the first spacecraft to return a cometary sample. 17

TGO The Trace Gas Orbiter is a spacecraft part of the ESA ExoMars mission and currently in Mars orbit. It was launched in 2016 and carries among other instruments the CaSSIS stereographic camera. 4

Xilinx ISE 14.7 Xilinx ISE 14.7 is a software synthesis and analysis tool from Xilinx used to synthesize designs for Xilinx FPGAs. The last release was in 2013 (discontinued). 6

1 Introduction

1.1 Comets

Comets and asteroids are remnants of the creation of the Solar System. Asteroids mostly occur in the asteroid belt between Mars and Jupiter, where debris has been trapped under the influence of Jupiter's gravitational field and combined to form rocky clumps, known as asteroids. Comets on the other hand are not just made up of rocks and metals, but are known as "dirty snowballs". They contain a large amount of frozen materials, mostly H_2O, CO_2 and CO. As a comet approaches the Sun, the ices start to sublime, creating a cloud around the nucleus, known as the coma. While the nucleus is usually only a couple of kilometers in diameter, the coma can reach diameters of millions of kilometers. As the ices sublime, they also carry dust from the surface with them, the dust particles then get pushed away from the nucleus under the influence of radiation pressure, thus forming the curved dust tail. The second tail, which points in the radial direction away from the sun, is caused by the solar wind ionizing gas molecules in the coma, thus creating a plasma tail, which is usually dimmer than the dust tail [Thomas, 2020]. This phenomenon is shown in Figure 1.1. As these tails and the coma can be visible from Earth by the naked

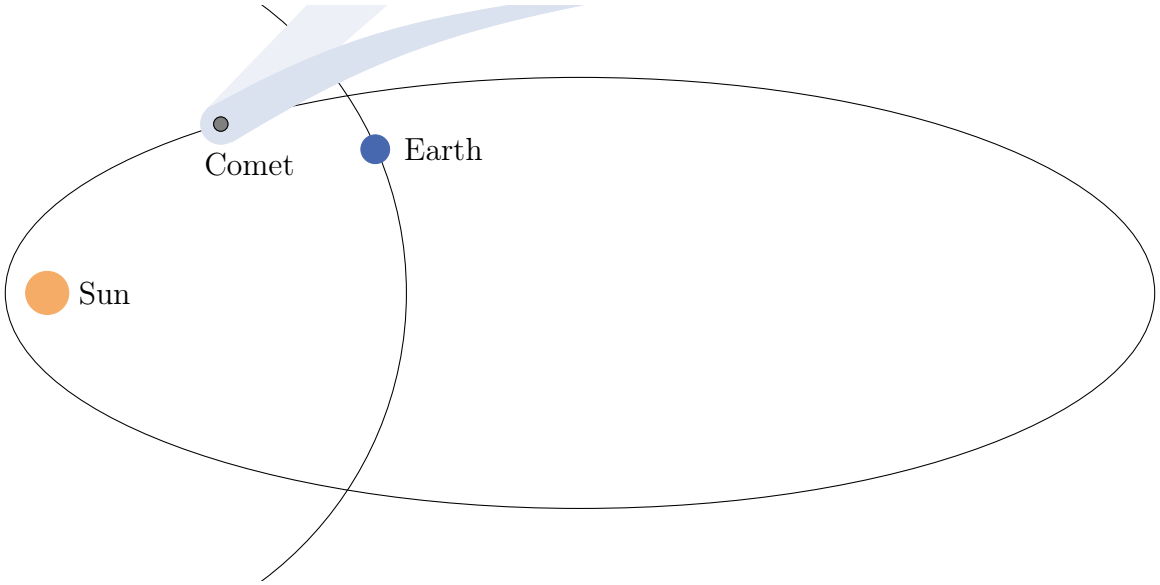


Figure 1.1: Orbit of a comet (not to scale), which is usually highly elliptical, with the plasma tail in light shade and the curved dust tail in dark shade, both pointing away from the Sun.

eye, comets have been observed by mankind for thousands of years. The most popular is 1P/Halley, which passes the sun every 76.1 years and is expected to pass again in 2061. [NASA, 2021a]
The material lost in the tails can be responsible for other visible phenomena as for example the Perseids, a meteor shower caused by 109P/Swift–Tuttle. [NASA, 2021b]
The Oort cloud and the Kuiper belt are believed to be the reservoir of most comets. For Kuiper belt objects, it is assumed that collisions can put them in elliptical orbits with a close approach to the sun. The mechanism for Oort cloud objects is not fully understood, but the gravitational influence of large planets could contribute.
It has been proposed that comets could be the source of water on Earth, however, this is

still debatable and measurements of the Rosetta mission do not support this hypothesis. [Altwegg et al., 2014]
Comets are a possible source of organic material on Earth [Sandford et al., 2006] and cometary research can tell us a lot about the early days of our solar system. Hence, there have been numerous missions to visit comets, most of them by fly-bys. An image of a longer visit of comet 67P/Churyumov–Gerasimenko is shown in Figure 1.2.

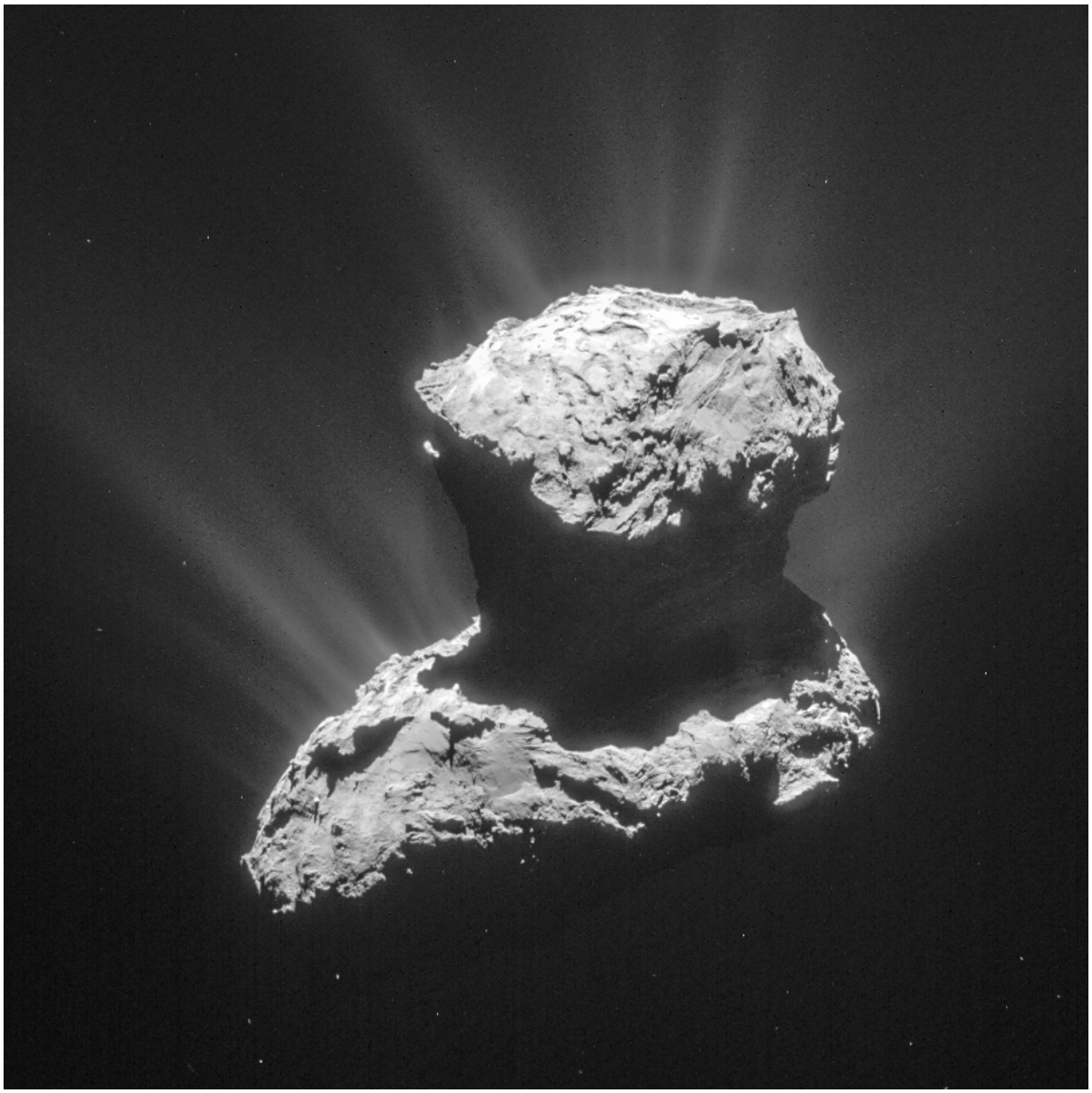


Figure 1.2: This single frame from Rosetta's navigation camera of Comet 67P/Churyumov–Gerasimenko was taken on 25 March 2015 from a distance of 86.6 km from the comet centre. The activity on the comet's surface is well visible. [ESA, 2021]

1.2 Comet Interceptor

Previous missions only visited short-period comets (< 200 years) originating from the Kuiper Belt region, because their orbits are well known and there is enough time to design and manufacture a space-probe. The goal of the Comet Interceptor mission is to visit a long-period comet, coming from the Oort Cloud and entering the inner part of the Solar System for the first time. Its surface has therefore not yet been altered by high amounts of solar radiation. Such an object is called a Dynamically New Comet (DNC) and is usually detected about one year before perihelion, which is too little time to design and manufacture the probe. Comet Interceptor circumvents this issue by being the first of many ESA F-class (fast class) missions. An F-class mission will hitch a ride on a larger M-class (medium class) mission and must be built within approximately eight years [ESA, 2019]. Comet Interceptor will launch together with the ARIEL exoplanet telescope and then remain in the Sun-Earth Lagrange point L2. After a target has been selected, Comet Interceptor will be deployed and depart from L2 to intercept the comet.
Comet Interceptor is made up of three separate spacecrafts: A mothership (A) and two smaller daughterships (B1 and B2). The three will detach weeks prior to closest approach and adjust their course in such a way, that the mothership can perform a distant fly-by at around 1000 km whereas the daughterships will perform high-risk/high-return measurements much closer to the nucleus. The approach phase, where the spacecrafts travel perpendicular to the direction of the nucleus is shown in Figure 1.3. Comet Interceptor can potentially reach the comet in a range of 0.85 to 1.35 au. from the Sun [Sánchez et al., 2021]
Various institutions have been selected to contribute to the mission and develop instruments for the three spacecrafts. The University of Bern has been chosen to supply a mass analyzer MANIaC and a camera CoCa, both will be mounted on the mothership A.

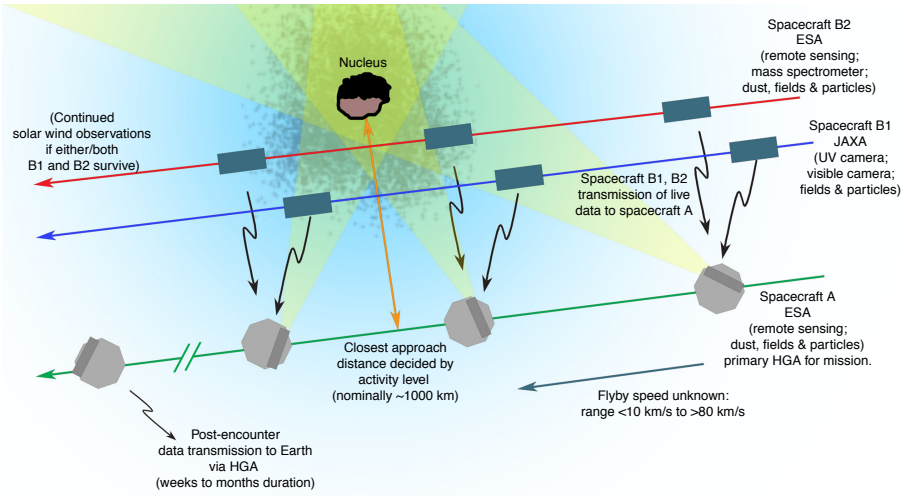


Figure 1.3: The proposed general encounter sequence for Comet Interceptor, not to scale. Green, blue and red lines are the trajectories of spacecraft (SC) A, B1 and B2, respectively. SC A passes farthest from the nucleus, performing remote sensing observations and receiving data from SC B1 and B2, which follow near-parallel trajectories nearer the nucleus. Post-encounter, SC A transmits stored data from all three platforms to Earth. The final SC, payload and precise flyby scenario may change as the mission development proceeds. [Snodgrass and Jones, 2019]

1.3 CoCa

In order to meet the deadlines of an F-class mission, it was decided not to build a new instrument from scratch, but to build CoCa based on the stereographic camera Colour and Stereo Surface Imaging System (CaSSIS) on the ExoMars TGO, which was also developed at the University of Bern. The focal plane assembly remains similar to CaSSIS but the filter-wheel has been added.
Common digital cameras use Bayer filters on top of their sensors [RED, 2021]. To determine the color of a pixel, each is split up into four sub-pixels; two green, one blue and one red as shown in Figure 1.4. This is done to mimic the physiology of the human eye, which is more light sensitive to green wavelengths.

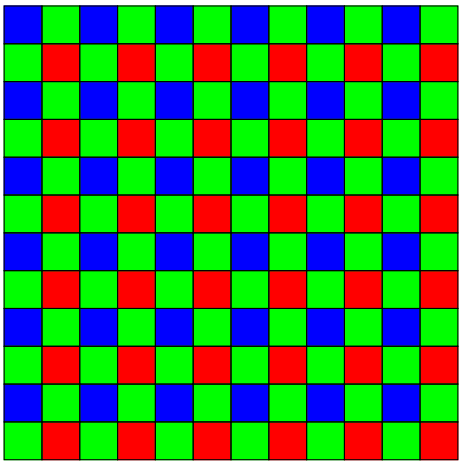


Figure 1.4: Top view of a Bayer filter on a 12×12 pixel sensor. The colored image will only have 6×6 pixels.

This process requires no moving parts and has been used for many years in the industry. Space hardware needs to be certified for different conditions than Earth-based hardware. Sensors that are built into space experiments tend to be much larger, it is thus not feasible to sacrifice four sub-pixels in order to get color information for one pixel. CoCa will use a CIS115 [Teledyne-e2v, 2017] CMOS sensor with 1504×2000 pixels and a pixel size of $49\,\mu m^2$. Common practise for cameras used in space is to change the filter in front of the sensor with a filter-wheel. Optical, Spectroscopic, and Infrared Remote Imaging System (OSIRIS) had two cameras, Wide Angle Camera (WAC) and Narrow Angle Camera (NAC). WAC had two filter wheels with eight positions (one empty) which made it possible to use 14 different filters in total, NAC used twelve different filters [Oklay, N. et al., 2015]. CoCa will use four different filters to cover the visible and near-infrared spectrum: BLU (in the blue range), ORA (in the orange range), RED (in the red range) and NIR (in the near-infrared range). The bandwidths of these filters are calculated in section 3. In section 2, we discuss the control of the filter-wheel, which will move the filters in front of the sensor.

2 Filter-Wheel Prototype

The electronics of the filter-wheel prototype are based closely on the NIM shutter controller [Wüthrich et al., 2018], which has also been developed at the University of Bern. The prototype is connected to a Spartan-3E FPGA development board [Xilinx, 2006] as shown in Figure 2.1. The Spartan-3E development board is connected via UART to a PC.

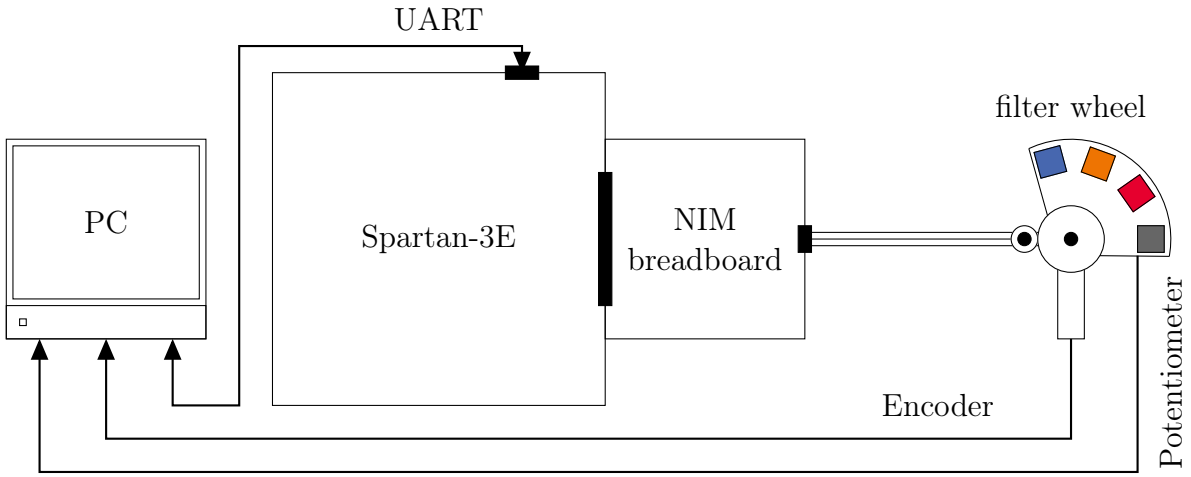


Figure 2.1: The setup of the CoCa filter-wheel prototype.

A Python script, which sends the appropriate commands and reads out the Housekeeping (HK) data, is provided. The BLDC motor and planetary gearbox assembly is manufactured by Faulhaber (No: 1226A012B 124 16:1) [Faulhaber, 2021a, Faulhaber, 2021b]. There are three analog Hall sensors with 120° dephasing built into the round motor PCB. This analog three phase sinus signal is then converted by the Faulhaber controller of type *MCBL3002 S* into the absolute digital position. The controller is connected to the PC. The filter wheel position can be read out through a potentiometer and controller of type *MAP4000* which is connected to the PC. These read out methods are only implemented in the prototype and are used exclusively for verification purposes. The position of the motor can be read out to a accuracy of $\pm 360/3000°$ which corresponds to $\pm 0.0015°$ on the filter-wheel. The filter-wheel can be read out to an accuracy of $\pm 0.01°$. Gear backlash is only visible on the filter-wheel readout.
The requirements for the open loop controller are as follows:

- The acceleration must not exceed ± 6000 rad/s^2 and the velocity must not exceed ± 5000 rpm.
- The filter wheel must move 35°, 70° and 105° to switch between filters.
- All movements must be completed within 0.5 seconds.
- The accuracy must lie within $\pm 0.19°$.
- Power must be variable.
- The system should be based on the NIM project with a Faulhaber BLDC motor

A detailed description of this project, the conducted tests and the control software is provided in the CoCa Open Loop Controller documentation [Stöckli et al., 2021] in the Appendix F.

2.1 Hardware

2.1.1 FPGA

A Field Programmable Gate Array (FPGA) consists of a matrix of Configurable Logic Blocks (CLBs) with programmable interconnects. The logic blocks consists mainly of Look Up Tables (LUTs), multiplexers and flip-flops. They can be (re-)programmed to individual needs after manufacturing. This differentiates them from Application Specific Integrated Circuits (ASIC) which are custom manufactured for one specific task and cannot be adapted. For space missions, radiation hard FPGAs are used. The prototype for the filter-wheel is built around a Spartan-3E development board [Xilinx, 2006] which is connected through a Hirose FX2 Pin connector to the NIM motor controller breadboard. The code is written in VHDL using the Xilinx ISE 14.7 software package and simulated in ModelSim. A Microblaze softcore processor handles the UART communication. The CoCa controller is lighter than the NIM controller since it does not use the ADCs on the motor control breadboard.

2.1.2 BLDC Motor

The cheapest and most simple type of motor is the widely known brushed DC-motor. The coils are mounted on the rotor and the permanent magnets outside on the stator. The coils are connected via brushes to the power lines, as shown in Figure 2.2. When a voltage is applied to the motor, current flows through the coils and a magnetic field is created, which interacts with the magnetic field of the permanent magnets, resulting in a rotation until the magnetic fields are aligned. Due to the geometry of the brushes, the polarity of the coils is inverted at the correct time, which prevents the magnetic fields from ever aligning and blocking the motor. Since these brushes wear down quickly, brushed DC-motors should not be used on space hardware. [Hughes and Drury, 2013]

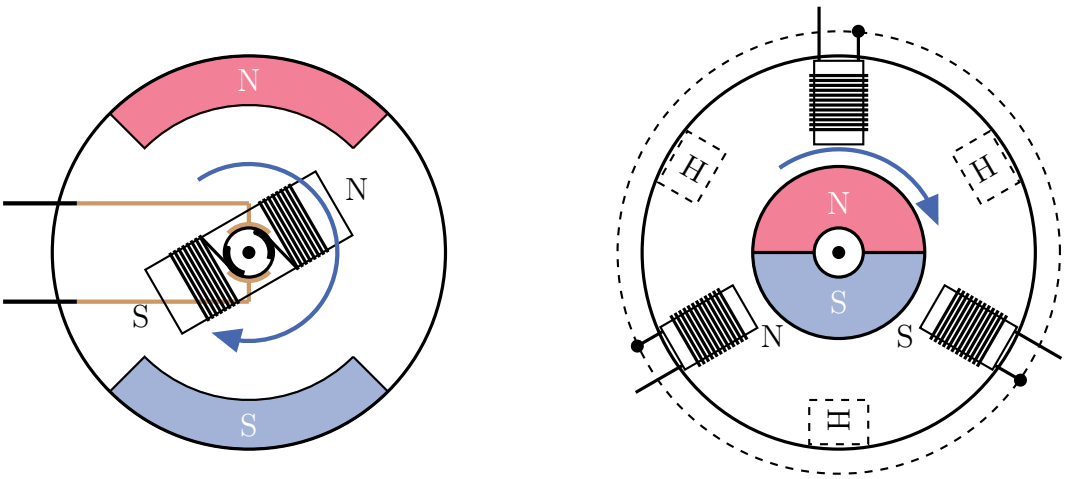


Figure 2.2: Simplified schematic view of a DC and BLDC motor. The DC motor is connected to only two wires, the polarity sets the direction and the applied voltage sets the torque/rotational velocity. The brushes are indicated in brown. The BLDC motor needs all three phases to be connected to a controller that sets the direction and torque/rotational velocity. Hall sensors are shown as dashed squares marked with H. The BLDC motor in this diagram is connected in a star pattern with a virtual ground marked by the dashed circle.

A good alternative is a Brushless Direct Current Motor (BLDC) motor which consists of three coils (phases) in a number of pairs on the stator and permanent magnets on the rotor, as shown in Figure 2.2. To drive the motor, a controller will change the magnetic field created by the coils. The permanent magnets allow the rotor to follow this field. Typically a closed loop control is used, where the controller will measure the position of the rotor and change the voltages on the three phases according to said measurement. This can be done by either sensor-less or sensor-based control. In sensor-less control, the back-Electromotive Force (EMF) on the three phases is measured and the zero-crossing determined. In sensor-based control, three hall-sensors are placed close to the rotor and thus measure the absolute position. Both options require additional circuitry, either an Analog Digital Converter (ADC) or Hall sensors. In Figure 2.3 the three phase voltages and hall sensor signals are shown for one electrical rotation. A BLDC motor is driven with trapezoidal phase curves [Piotr, 2014].

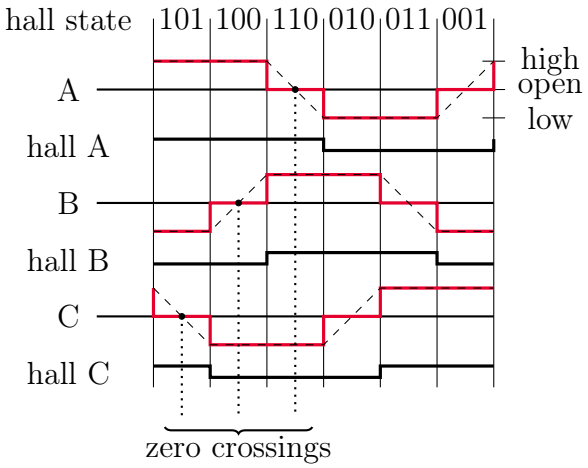


Figure 2.3: Block commutation with the Hall sensor states shown. The dashed line represents the phase voltage including back-EMF while the red line shows the applied voltage.

A very similar type of motor is a Permanent Magnet Sync Motor (PMSM) which due to different coil windings is driven by sinusoidal phase curves [Eriksson, 2019] as shown in Figure 2.4. To determine whether a trapezoidal or a sinusoidal control should be used, one can turn the motor by hand and measure the back-EMF on the phases with an oscilloscope.

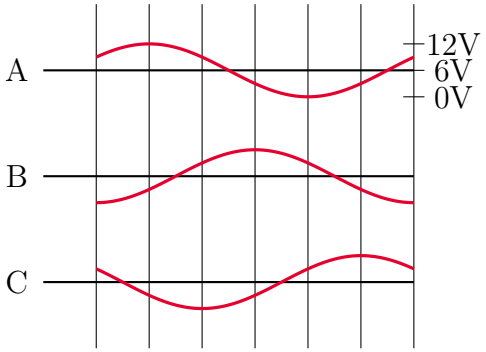


Figure 2.4: Open loop controller with three sinusoidal phases for the PMSM use-case.

Often the terms BLDC and PMSM are used interchangeably. The motor used for CoCa is a PMSM but we will refer to it as a BLDC motor henceforth.
Due to simplicity, an open-loop control design has been chosen for CoCa, where the magnetic field is set without feedback of the actual position. To make sure that the stator will follow the rotating field, it is required that the torque is sufficient. Hence, we need to supply higher voltages and thus higher currents to the phases than what would be required with closed loop control. The sine voltages are created by a Pulse Width Modulation (PWM) signal with variable duty cycle on the digital output pins of the FPGA. These low voltage signals go to the H-bridge on the controller breadboard to control the higher voltage. Finally the PWM is filtered using an LC-filter as shown in Figure 2.5. The schematic of the breadboard is shown in the NIM Design Description documentation. [Lüthi and Brüngger, 2017]

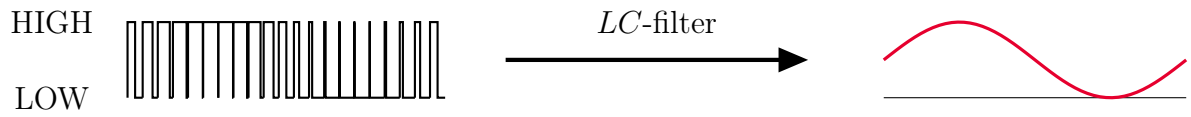


Figure 2.5: PWM signal and filtered signal after the LC low pass filter on the breadboard.

2.1.3 Hall Sensor Placement

The filter-wheel prototype contains four Hall sensors that act as emergency stop switches and homing reference switches. The placement is shown in Figure 2.6.

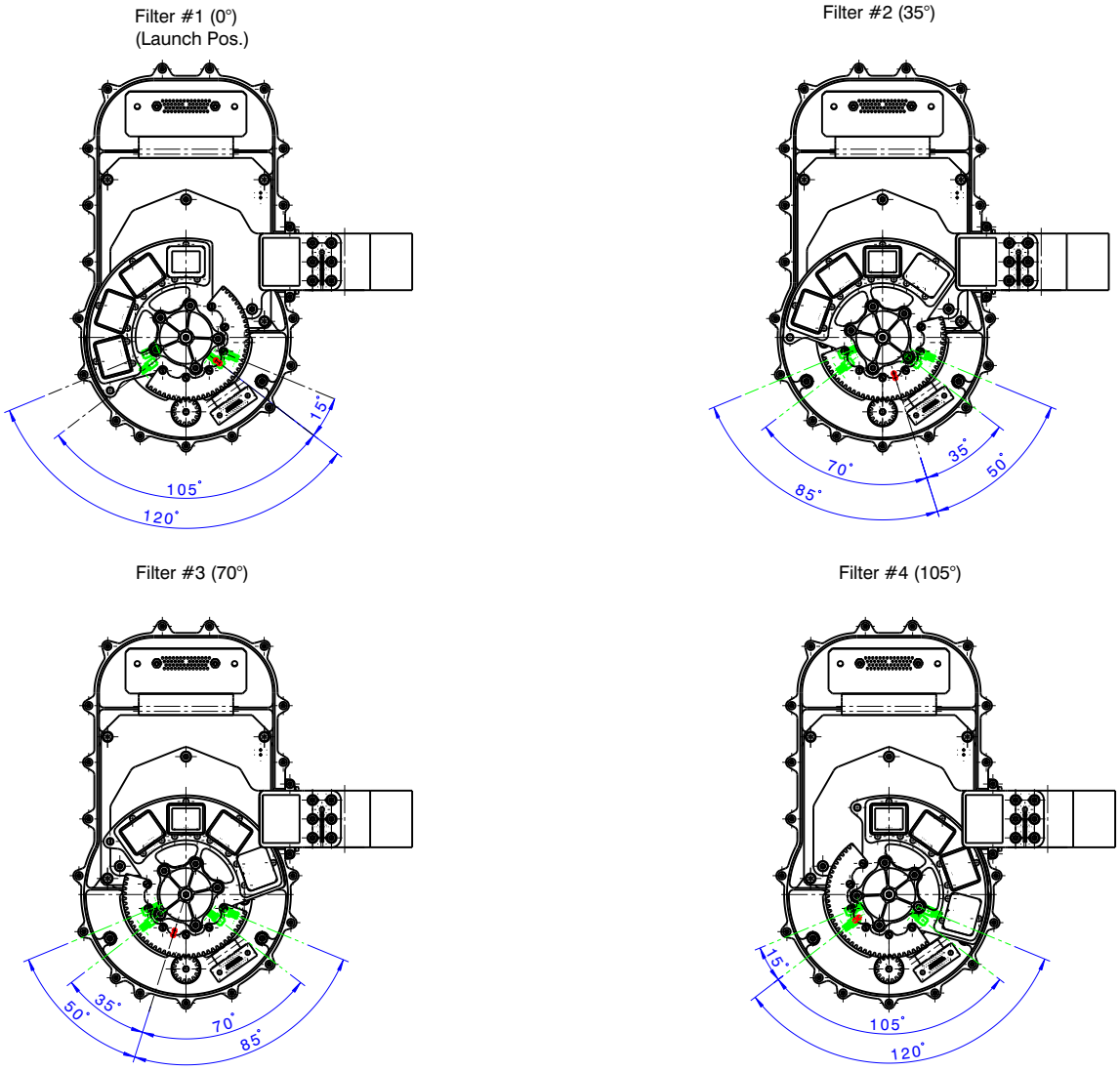


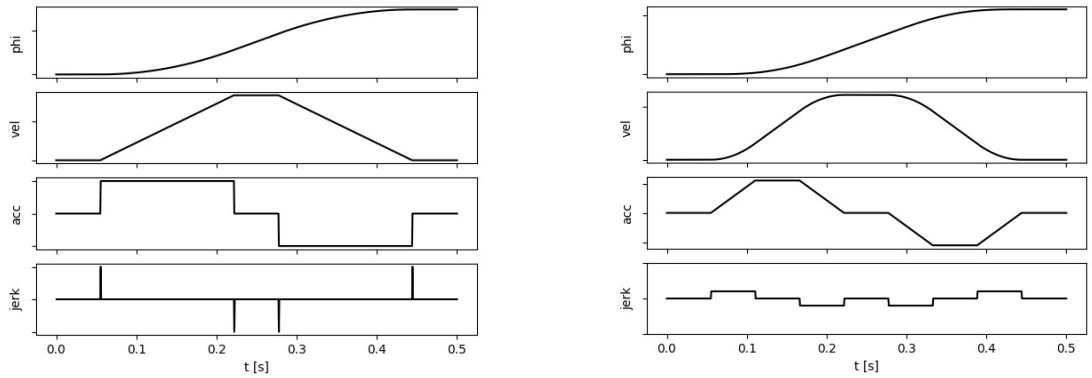
Figure 2.6: The placement of the Hall sensors on the flight model shown in green for all four filter positions. The magnet is shown in red. The magnet and Hall sensors are located on a circle of radius 22.5 mm around the center. It is evident that the magnet aligns with the corresponding Hall sensor when the filter is located directly above the detector. The placement on the prototype for the two inner sensors (homing sensors) is similar, the two outer sensors (emergency stop) are not connected on the prototype.

2.2 Open Loop Controller

The Open Loop Controller is written using Very High-Speed Integrated Circuit Hardware Description Language (VHDL). The Xilinx Wizard allowed for an easy implementation of the Microblaze softcore which handles the UART communication with the PC. C-code runs on the Microblaze softcore and handles the decoding of the received commands and sends back HK data. Two main movement modes are implemented; *MOVE* and *HOME*.

2.2.1 Motion Profiles

To move the shutter, the NIM controller used a rectangular acceleration profile as shown in 2.7a. This will however induce a lot of vibrations since the jerk is in theory a delta peak, in practice the jerk will certainly be finite, but still high, which can lead to faster ware on the structure. To mitigate this, a third order motion profile [JPE, 2021], shown in 2.7b, was chosen. It has a trapezoidal acceleration curve and leads to a low finite jerk value. Since the slope for the acceleration can be chosen, the jerk can be directly controlled for each movement. This reduces vibrations and wear on the structure but requires a more advanced control system. For simplicity, the profiles are symmetrical.



(a) Second order (b) Third order

Figure 2.7: Second and third order motion profiles. The second order profile has a rectangular acceleration profile and high jerk peaks, while the third order profile has a trapezoidal acceleration profile with marginal jerk. [Stöckli et al., 2021]

The motion profile is calculated by integrating the acceleration. The acceleration profile is defined by the eight time parameters t_i as shown in Figure 2.8. The values for t_i are given in equations (2.1) and depend solely on the target angle $\varphi_{\max}$, maximum velocity $v_{\max}$, maximum acceleration $a_{\max}$ and acceleration slope m.

$$
\begin{aligned}
t_1 &= 0 & t_5 &= \frac{\varphi_{\max}}{v_{\max}} \\
t_2 &= \frac{a_{\max}}{m} & t_6 &= t_5 + t_2 \\
t_3 &= \frac{v_{\max}}{a_{\max}} & t_7 &= t_5 + t_3 \\
t_4 &= t_1 + t_2 + t_3 & t_8 &= t_5 + t_4
\end{aligned}
\tag{2.1}
$$

To mitigate arithmetic errors, the time parameters t_i and the acceleration timer `accel_step`

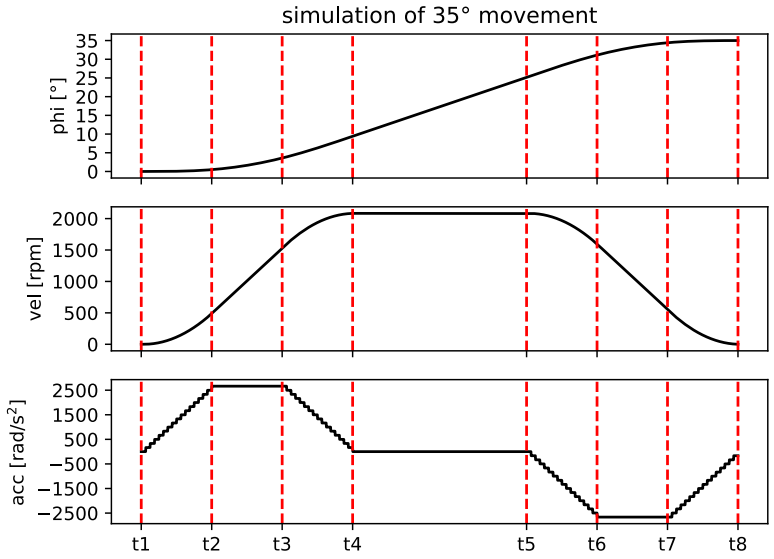


Figure 2.8: The t-parameters illustrated on a third order motion profile of 35°. A maximum velocity of 2000 rpm and a maximum acceleration of $\pm 2500\,\mathrm{rad\,s^{-2}}$ are reached. [Stöckli et al., 2021]

$= 1/m$ need to be integers. For that to be the case, only certain maximum velocities and accelerations are allowed. All possible parameter combinations are shown in Figure 2.9 for a 35° movement. The control software loads a simple Look Up Table (LUT) containing the pre-set maximum velocity, maximum acceleration and acceleration step parameters for each movement.

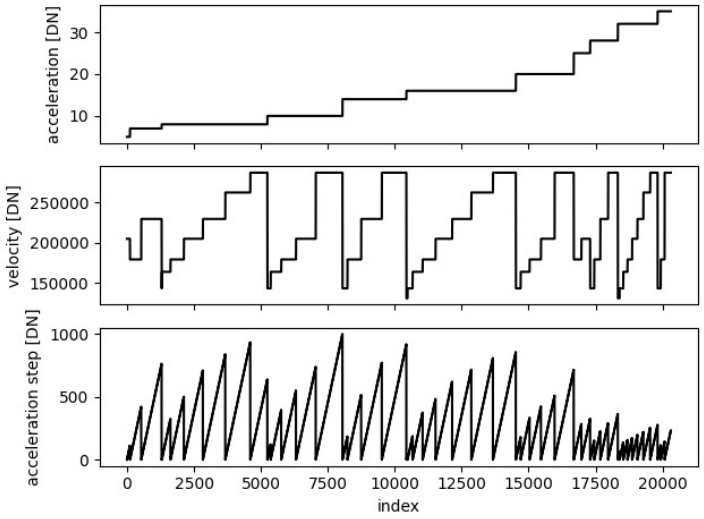


Figure 2.9: All possible combinations of velocity, acceleration and acceleration step parameters for one 35° movement. Each index represents a combination of velocity, acceleration and acceleration step. [Stöckli et al., 2021]

2.2.2 FPGA Architecture

The softcore Microblaze receives and sends commands over Universal Asynchronous Receiver Transmitter (UART) and will write into the registers of the FPGA via a Processor Local Bus (PLB) bus to the statemachine entity. Depending on the register entries, different PWM duty cycles are generated in the statemachine entity and sent over to the pwm entity. The pwm entity will generate the PWM signal on the outputs and send a sync pulse to the statemachine entity at each new PWM-cycle. The top level diagram is shown in Figure 2.10.

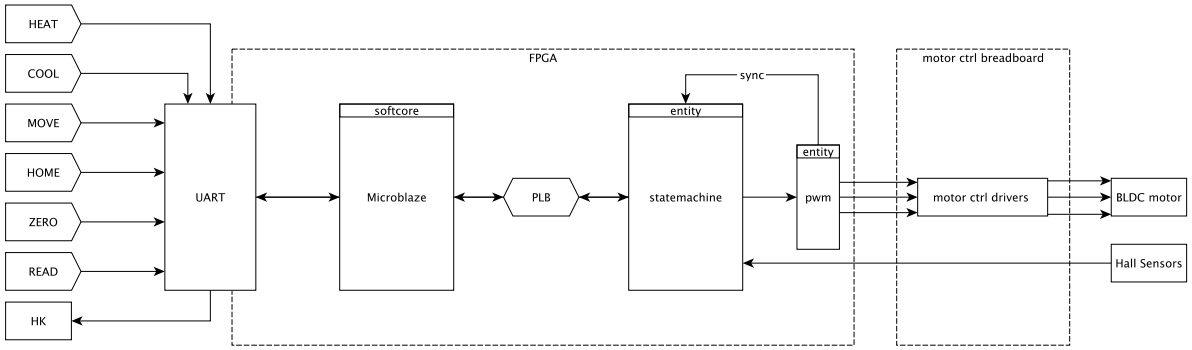


Figure 2.10: Top level diagram of the control setup. [Stöckli et al., 2021]

The inputs and outputs are listed in Table 2.1. Many other inputs used by the NIM shutter controller are not mapped in the CoCa prototype architecture (for example SPI bus for the ADCs).

FPGA Pin	FX2 Pin	Board Signal	Purpose
B11	A36	`bldc_ctrl_0_pwm_a_pin`	Motor Phase A
A11	A37	`bldc_ctrl_0_pwm_b_pin`	Motor Phase B
G9	A39	`bldc_ctrl_0_pwm_c_pin`	Motor Phase C
E13	A34	`bldc_ctrl_0_end_switches_pin[2]`	Homing Sensor 1
C4	A35	`bldc_ctrl_0_end_switches_pin[1]`	Homing Sensor 2
E9	A19	`bldc_ctrl_0_end_switches_pin[3]`	Stop Sensor 1
C11	A21	`bldc_ctrl_0_end_switches_pin[0]`	Stop Sensor 2

Table 2.1: I/O connections to the control breadboard. [Stöckli et al., 2021] Only the two homing sensors are connected to the sensors, the stop sensors are not yet connected on the prototype.

The architecture is designed with the two process method, where all values are stored in two records: `pres`, which contains the present values, and `nxt`, which contains the updated values. A simple clocked process `pc_fsm` updates the present values with the previously updated values at a frequency of 50 MHz. The `nxt` record is updated by the combinatorial process `p_fsm`, which contains the logic for the state-machine. The combinatorial process is synced to the `pwm` entity, which runs at a frequency of 200 kHz. This leads to a new duty cycle value being available at the start of the next PWM cycle. The diagram is shown

in Figure 2.11. For each cycle, the timestamp `t` is updated by one increment. Then, the state-machine is updated - the diagram describing this is shown in Figure 2.12. With the new acceleration values that are obtained from the state-machine, the integration step is performed to get the position value `phi`. A timing diagram is included in appendix B. To use the position value to index the sine-table, it needs to be gated:

$$\texttt{phi\_gated} = \texttt{phi} \mod \texttt{C\_TAU}$$

where `C_TAU` is given as the product of the sine-table length and the scaling factor. For the prototype this is

$$\texttt{C\_TAU} = \texttt{GC\_SINE\_TBL\_LENGTH} \cdot \texttt{C\_POS\_SCALE} = 360 \cdot 10^{22}$$

The length of the sine table sets the resolution of the "micro-steps" of the BLDC motor. No tests have been conducted to deduce the minimum working length, but in order to conserve memory, the length could be reduced in the future. For the prototype the resolution is given in equation (2.2). Note that this is the resolution of the filter wheel motion.

$$\Delta\varphi = \frac{360^\circ}{360 \cdot 80} = 0.0125^\circ \tag{2.2}$$

The corresponding sine values are then converted to duty cycles [Wüthrich, 2018] as

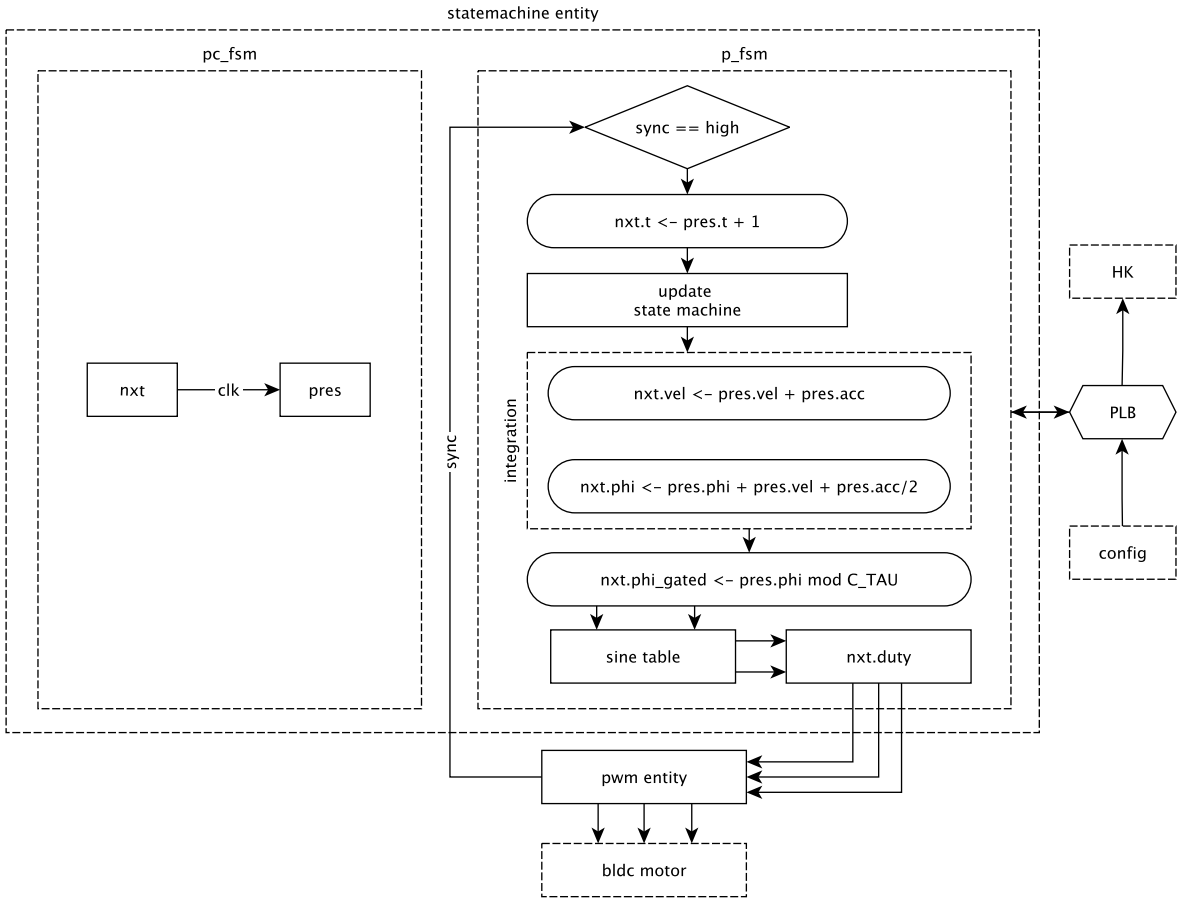


Figure 2.11: Simplified diagram of the clocking of the statemachine entity. The flow diagram of the "update state machine" block is shown in Figure 2.12. [Stöckli et al., 2021]

shown in equations (2.3) and sent to the `pwm` entity, which generates the PWM signals for the BLDC motor.

$$\begin{aligned} \varphi_1 &= \varphi \mod \texttt{C\_TAU} \\ \varphi_2 &= (\varphi + \texttt{C\_TAU}/3) \mod \texttt{C\_TAU} \\ \mathrm{PWM}_1 &= \frac{T_{\mathrm{PWM}}}{2}(1 + P \cdot \sin(\varphi_1)) \\ \mathrm{PWM}_2 &= \frac{T_{\mathrm{PWM}}}{2}(1 + P \cdot \sin(\varphi_2)) \end{aligned} \tag{2.3}$$

where T_{PWM} is set to 250 times the a clock period because we are using a PWM frequency of 200 kHz and a clock frequency of 50 MHz. P is the configured power value between 0 and 255. The third phase is then calculated according to equation (2.4).

$$\mathrm{PWM}_3 = \frac{T_{\mathrm{PWM}}}{2}(1 - P \cdot \sin(\varphi_1) - P \cdot \sin(\varphi_2)) \tag{2.4}$$

The state-machine remains in `IDLE` state, until the `cmd` bit and the configurations registers have been set. If the idle power has previously been set to zero, it will first ramp up the three phases to 6 V and then span the three phases to their 120° separated positions. If the idle power has been set to a non-zero value, it will directly span them up from idle power to their 120° separated positions. From then on, it will transition to `INCREASE_ACCEL1` state, where for each duty cycle, the timestamp and acceleration counter are increased. When the acceleration counter equals `accel_step`, the acceleration value will be increased by 1 increment. If the timestamp reaches the value of t_i, the state-machine will transition to the next state.

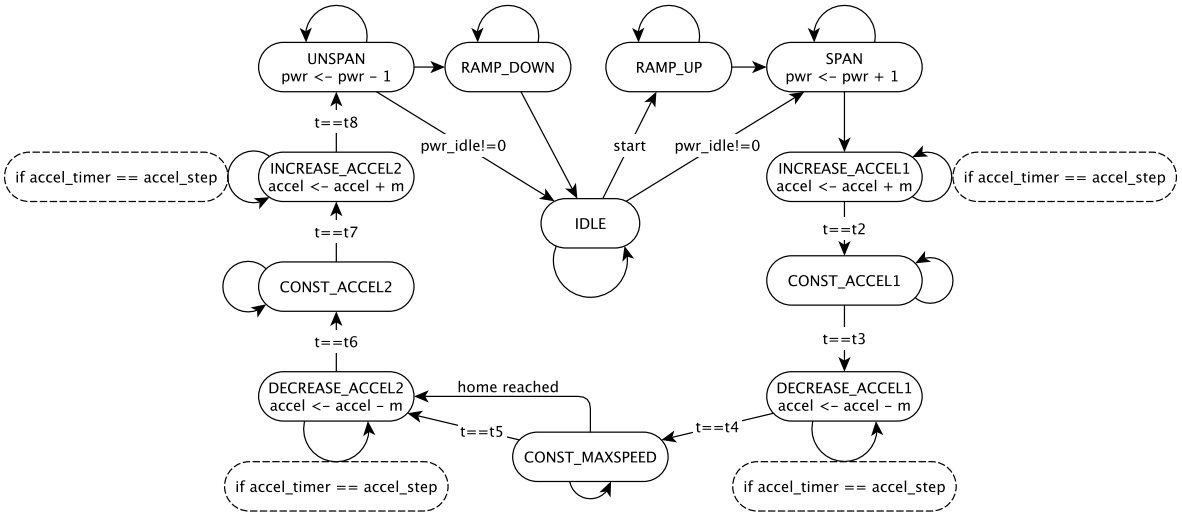


Figure 2.12: Simplified block diagram of the state-machine for a movement. [Stöckli et al., 2021]

If the system is in *HOME* mode, the state-machine will not exit the `CONST_MAXSPEED` state unless a signal from the end stop Hall sensors is detected. Thus, the filter-wheel will travel the remaining distance after hitting the end-stop. The hall sensors are mounted in the centers of the filters as shown in Figure 2.6, however, measurements have shown that the hall sensors are activated at 1.55 mm away [Zimmermann and Meier, 2020]. The

hall sensors are mounted at a radius of 22.5 mm. Thus, the remaining distance that the filter-wheel must travel is given by equation (2.5):

$$\varphi = \frac{1.55\,\text{mm}}{22.5\,\text{mm}} \cdot \frac{180^\circ}{\pi} = 3.95^\circ \tag{2.5}$$

The parameters for the movement need to be adjusted such that the filter-wheel travels 3.95° between t_5 and t_8. This has been done by trial and error and has not yet been fully tested on the prototype, since the positioning of the hall-sensors is not equivalent to the final flight model.

2.2.3 Backlash Compensation

In the confidence test, the backlash of the prototype has been measured to be 0.98° [Zimmermann and Meier, 2020]. However, backlash of the actual flight model will likely be smaller. The software for the prototype cannot generate a 0.98° movement without generating arithmetic errors, thus a backlash compensation of 1° has been chosen. Other possible options would be for example 0.8°, 0.9° 0.97° 0.99°.
Backlash needs to be compensated only when the direction of the rotation is changed. Before a direction change, an additional movement of 1° is commanded to compensate the backlash. After this movement, all the gears should touch but the filter wheel should not have moved yet. The fastest configuration for a 1° movement will take 0.038 seconds. A 105° movement will take 0.422 seconds thus making it possible to move 105° and compensate the backlash afterwards within the required 0.5 seconds.

2.2.4 Control Interface

The FPGA is connected to a PC through a USB to UART cable. The UART connection is configured as follows:

- Baud rate: 9600
- Byte size: 8 Bit
- Parity Bit: None
- Stop Bits: 1
- Flow control: None

The Microblaze handles the commands on the FPGA side and transmits the house keeping data. On the PC-side, a Python script sends the appropriate commands and parameters and logs the house keeping data. All registers, possible commands and housekeeping data implemented in the prototype are listed in the Appendix A but a detailed description can be found in the documentation [Stöckli et al., 2021].
The flight model will not include a Microblaze and UART combination, it will be connected to the Digital Processing Module (DPM) through a Low Voltage Differential Signaling (LVDS) port.

2.3 Results

Multiple repeatability tests have been conducted. The goal was to see, if there is any noticeable drift and if the rotations are repeatable. The individual tests are listed in Appendix C, a summary is shown in Table 2.2. The tests have all been conducted at approximately 25°C indoors. After a significant amount of consecutive rotations (> 100) a slight drift can be noticed in the measurements. This is strongly assumed to be due to increasing temperature in the potentiometer of the filter wheel, which leads to a change in the resistance. After letting the system cool down for around 15 minutes, the reading returns back to 0.00°.

Movement	Calculated Resolution	Standard Deviation	Backlash Compensation
35°	0.0125°	0.005°	Yes
35°	0.0125°	0.006°	No
70°	0.0125°	0.005°	Yes
105°	0.0125°	0.006°	Yes

Table 2.2: A summary of the measured accuracies of the prototype of the tests from Table C.1, Table C.2, Table C.3 and Table C.4. The backlash compensation does not have an influence on the standard deviation here. If we do not include backlash compensation, the target position will be off by the backlash-distance. [Stöckli et al., 2021]

The performed tests also give us information about the absolute accuracy of the system. It is visible that the results are in range of the targeted accuracy of 0.19° for the 35° and 70° tests, but slightly exceed this range for the 105° tests. Since the increments on the motor of a 105° test shown in Table C.5 remain within $\pm$ 1 increment, it is assumed that the potentiometer used to read out the angle is not completely linear.

2.4 Possible Adaptations

The code was written such that system parameters can easily be changed. The following possible adaptations are explained in detail in the documentation [Stöckli et al., 2021] in Appendix F:

- The clock frequency
- The PWM frequency
- The length of the sine table
- The scaling parameter
- The timeout value

3 Filter Selection

Previous missions to visit short period comets include Giotto with HMC to 1P/Halley, Deep Space 1 with MICAS to 19P/Borrelly, Stardust to 9P/Tempel 1, 81P/Wild and Rosetta with OSIRIS to 67P/Churyumov–Gerasimenko. The filters used in these missions are listed in Table 3.1.

	HMC		OSIRIS (NAC)		MICAS		Stardust Cam	
filter	c [nm]	w [nm]	c [nm]	w [nm]	c [nm]	w [nm]	c [nm]	w [nm]
CLEAR	652.9	372.6	640.0	520.0			698.8	400.0
SWIR					1800	1200		
IR			989.3	38.2				
Fe_2O_3			931.9	34.9				
Near-IR			882.1	65.9			874.6	30.0
ORTHO			805.3	40.5				
RED	813.0	165.0	743.7	64.1	750	500	712.9	6.0
Cont 2	738.1	37.4						
HYDRA			701.2	22.1				
NH_2							665.1	12.0
ORANGE	645.4	94.0	649.2	84.5				
Oxygen							633.6	12.0
Hi-Res							596.4	200
YELLOW							580.2	4.0
GREEN			535.7	62.4				
C-2	509.5	20.9					513.2	12.0
Cont 1	457.4	20.3						
BLUE	440.0	101.1	480.7	74.9				
C-3	408.4	16.6						
Near-UV			360.0	51.1				
OH	314.8	12.3						
Far-UV			269.3	53.6	132.5	105		

Table 3.1: Center c and bandwidth w of the filters used in previous missions. [Thomas and Keller, 1990, Oklay, N. et al., 2015, Ray L. Newburn, 2000, Soderblom et al., 2000] MICAS did not use a filter wheel but four different sensors (two in the RED spectrum) with different wavelength ranges.

We see that the orange-red part of the spectrum is well covered in all three missions. To allow for comparison between the already obtained results from past missions with the anticipated results from CoCa, at least one filter should also cover this region.
The bandwidth of the filter sets how much light will pass through to the sensor. Since CoCa does not use the same sensor as the missions from Table 3.1, we need to determine

the required bandwidth for our own specific use case. In the following subsection we set a filter with comparable center-line as previous missions, and calculate the required bandwidth, we then will be able to determine bandwidth and center of the adjacent filters.

3.1 Theory

3.1.1 Signal Strength

The signal in digital numbers [DN] is calculated by integrating the radiation that reaches the sensor over all wavelengths, then multiplying it by the aperture area A_a, solid angle Ω and exposure time t_{exp} and finally dividing by the gain factor G as given in equation (3.1).

$$S(\alpha) = \frac{A_a\,\Omega\,t_{\text{exp}}}{G} \int_0^\infty M(\lambda)^N\,T(\lambda)\,Q(\lambda)\,R(\lambda,\alpha)\,F'_\odot(\lambda)\,\mathrm{d}\lambda \tag{3.1}$$

The incident radiation $F'_\odot(\lambda)$ is determined by the solar flux [NREL, 2021], $F_\odot(\lambda)$ converted to eV and adapted for an arbitrary distance to the sun r_h in equation (3.2).

$$F'_\odot(\lambda) = \frac{\lambda}{h\,c} \cdot \frac{F_\odot}{{r_h}^2} \tag{3.2}$$

The mirror reflection function is given as $M(\lambda)$, the filter transmission function as $T(\lambda)$ and the phase angle dependant surface reflectance as $R(\lambda,\alpha)$. The surface reflectance is known as I/F where I is the reflected light and F the incident light. A schematic diagram is shown in Figure 3.1.

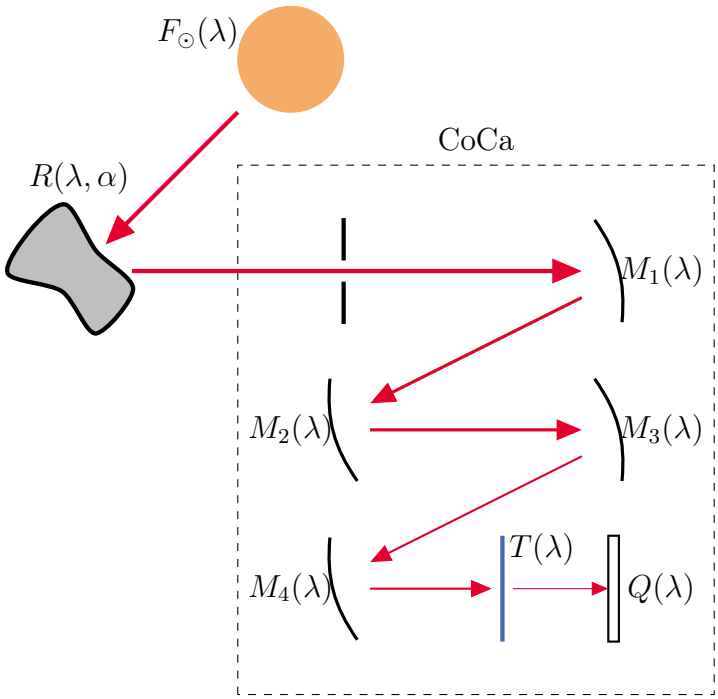


Figure 3.1: Schematic setup of the CoCa camera. The four mirrors M_i all have the same reflective properties.

The reflectance $R(\lambda,\alpha)$ is discussed in Section 3.1.3. The mirror reflection function $M(\lambda)$ and quantum efficiency $Q(\lambda)$ are given by the manufacturers and listed in Appendix E. We

will solve for the filter transmission function $T(\lambda)$ to determine the center and bandwidth of the filters.
The parameters used in equations (3.1) and (3.2) are given in Table E.3.

3.1.2 Signal to Noise Ratio

We implement the Signal to Noise Ratio (SNR) calculation of the previously developed CoCa model [Haslebacher, 2020] which is given by equation (3.3):

$$\mathrm{SNR} = \frac{S}{N_{\mathrm{shot}} + N_{\mathrm{readout}} + N_{\mathrm{darkcurrent}}} \tag{3.3}$$

where S is the signal, and the noises are given in Table 3.2. The full well capacity of the

N_{shot}	$\sqrt{S}$
N_{readout}	$5\mathrm{e}^-$
$N_{\mathrm{darkcurrent}}$	$20\mathrm{e}^-$

Table 3.2: The different noise contributions as used in equation (3.3). The signal S is given in e^-.

sensor is 33000 e^- and the peak linear capacity is 27000 e^-. This corresponds to a SNR of ~ 189, which determines the upper limit to our signal. As done in the CoCa model, we will aim for a SNR of 100.

3.1.3 Hapke Model

The surface reflectance of the nucleus is given by the *Hapke* Model [Hapke, 2012].

$$R(\alpha,\lambda) = k\frac{\omega}{4\pi}\frac{\mu_0}{\mu_0+\mu}\cdot\left[P(\alpha,g)\cdot\left(1+\frac{B_s}{1+1/h_s\tan(\alpha/2)}\right)\right.$$
$$\left.+H\left(\frac{\mu_0}{k},\omega\right)\cdot H\left(\frac{\mu}{k},\omega\right)-1\right]\cdot S(\mu,\mu_0,\theta) \tag{3.4}$$

$$I/F = \pi R(\alpha,\lambda) \tag{3.5}$$

where we used the single parameter *Henyey-Greenstein* function $P(\alpha, g)$, all functions are given in equations (D.1) - (D.4) in Appendix D. The disk-averaged parameters ω, g, B_0, h_s, θ at $\lambda = 645$ nm are taken from [Fornasier, S. et al., 2015]. The wavelength dependence is extracted from the average of measurements [Deshapriya, J. D. P. et al., 2018, Feller et al., 2016, Fornasier, S. et al., 2019] and then interpolated. The *Hapke* model is scaled for different wavelengths.
Since $\mu = \cos e$ and $\mu_0 = \cos i$ the reflectance depends on the emission angle e and the incidence angle i. However, Figure 3.2 shows that we can set $i = \alpha$ and $e = 0$ since CoCa is always looking at the sub-spacecraft point on the surface of the nucleus.

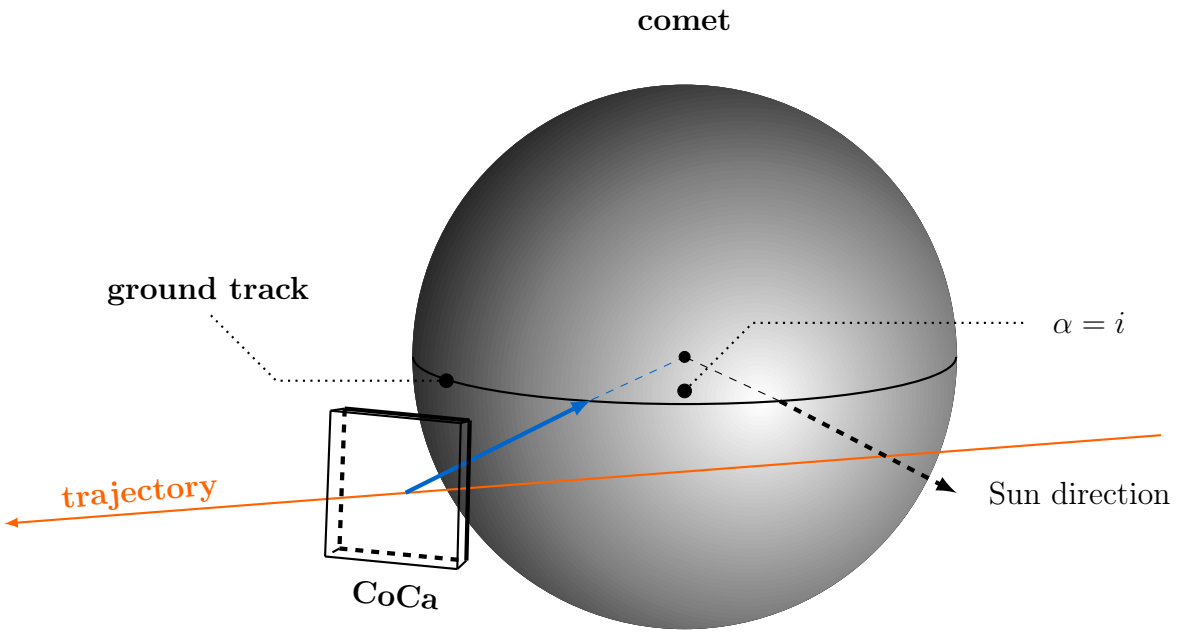


Figure 3.2: Trajectory of the Comet Interceptor in front of the comet nucleus. CoCa will always point to the sub-spacecraft point, normal to the surface. Thus the emission angle e is always 0 and the incidence angle i is equal to the phase angle α. This figure shows the orientation of CoCa shortly after closest approach, pointing back to the comet. Slight deviations occur because the modelled reflectance assumes $i = \alpha$ and $e = 0$ where as the measurements were taken at different photometeric angles.

This derived reflectance is shown in Figure 3.3. This deviation is most noticeable for the measurements at large phase angles. Previous research indicates that the reflectance of an ice-rich region can be approximated as three times the reflectance of a rocky region [Egger, 2018].

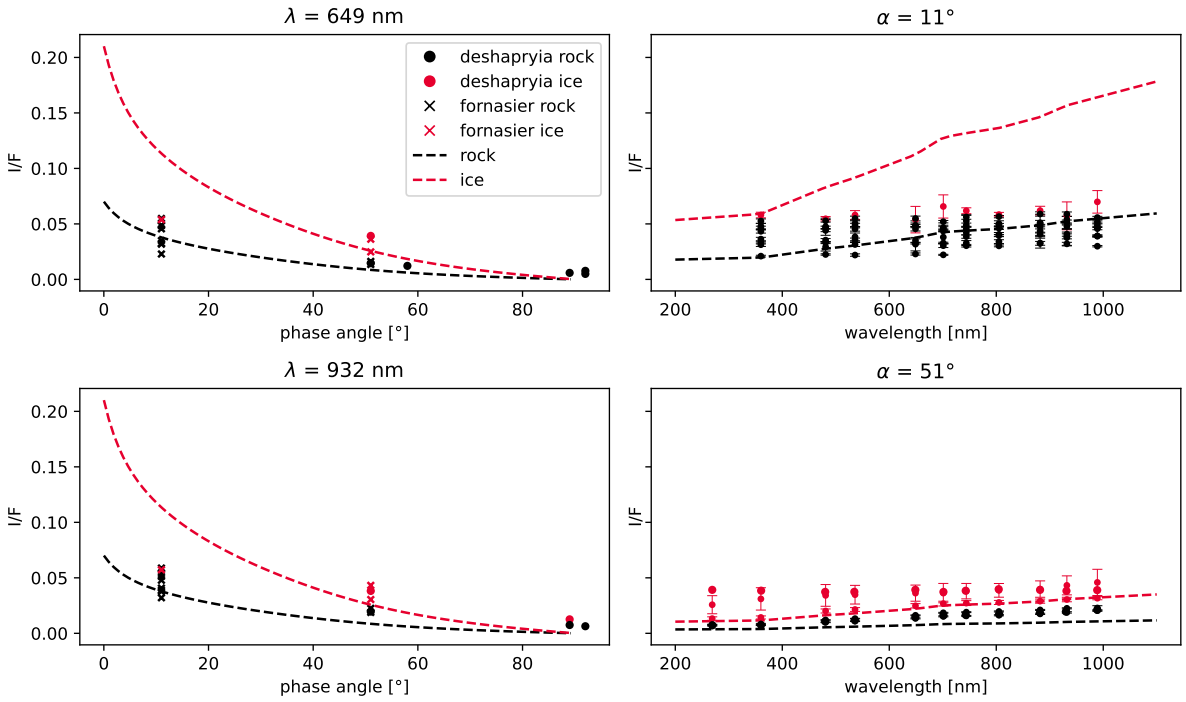


Figure 3.3: The reflectance for rock in black and ice in red is scaled and compared to data obtained from 67P/Churyumov-Gerasimenko [Deshapriya, J. D. P. et al., 2018, Feller et al., 2016, Fornasier, S. et al., 2019]. We have assumed ice to be 3× brighter than rock. This approximation does not always hold true, but is good enough for an estimate of the signal strength.

3.1.4 Exposure Time

The minimum exposure time of the sensor is $t_{\text{exp, min}} = 0.22\,\text{ms}$. The maximum possible exposure time depends on the motion smear introduced by the relative velocity v between the nucleus and the spacecraft, as well as the impact parameter b. The phase angle change can be approximated by equation (3.6) [Haslebacher, 2020].

$$\omega(t) \approx \frac{v/b}{1 + (vt/b)^2} \tag{3.6}$$

While the relative velocity is not yet known, it is assumed that the spacecraft will pass by at a distance $b = 1000\,\text{km}$ with a relative velocity in the range of $10\,\text{km}\,\text{s}^{-1} < v < 80\,\text{km}\,\text{s}^{-1}$. Motion smear is visible when the plane angle θ covered within the exposure time t_{exp} is larger than the $\sqrt{\Omega} = 4.58 \cdot 10^{-4\circ}$, where Ω is the solid angle of one pixel of the sensor [Haslebacher, 2020]. The upper limit for the exposure times for different phase angles are shown in Figure 3.4, where the phase angles are derived as the integrated phase angle change according to equation (3.7).

$$\alpha = \int \omega(t)\mathrm{d}t \tag{3.7}$$

The exposure times have been calculated using equation (3.8).

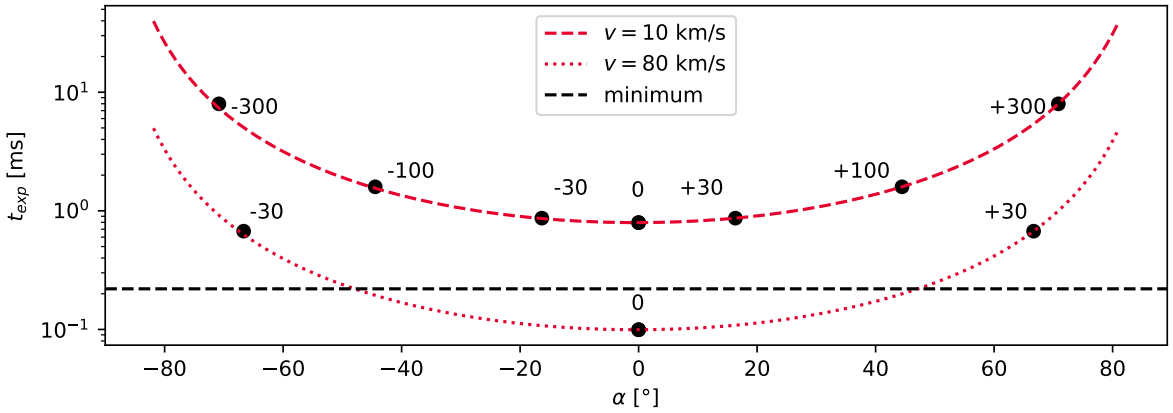


Figure 3.4: The exposure times are shown for different phase angles. The minimum possible exposure time $t_{\text{exp, min}} = 0.22\,\text{ms}$ of the sensor is shown as the dashed black line. The two maximum possible exposure times for the limits of 10 km/s and 80 km/s the relative velocities are shown as dashed and dotted red curves respectively. Note that for a relative velocity of 80 km/s, motion smear will occur for a phase angle within $\pm 48°$. The timestamps before and after closest approach are indicated by the black dots. Note that the encounter with $v = 10$ km/s relative velocity takes ≈ 600 seconds while the encounter with $v = 80$ km/s relative velocity only lasts ≈ 60 seconds.

$$t_{\text{exp}} = \frac{\omega(t)}{\sqrt{\Omega}} \tag{3.8}$$

The exposure times for $v = 10$ km/s and $v = 80$ km/s are listed in Table 3.3. As an approximation of equation (3.6), we can neglect higher order terms and see that the

phase angle change $\omega(t)$ scales linearly with velocity v as shown in equation (3.9):

$$\omega(t) \approx \frac{v/b}{1+(vt/b)^2} \approx \frac{v}{b} \propto v \tag{3.9}$$

This approximation makes further computation more efficient, since we can use the values for the exposure time t_{exp} for $v = 10$ km/s and scale them for any relative velocity v.

	t_{exp} [ms]	
α [°]	$v = 10\,\text{km}\,\text{s}^{-1}$	$v = 80\,\text{km}\,\text{s}^{-1}$
0	0.80	0.10
10	0.82	0.10
20	0.90	0.11
30	1.05	0.13
40	1.35	0.17
50	1.91	0.24
60	3.16	0.40
70	6.77	0.85
80	26.32	3.29
90	118.63	14.83

Table 3.3: Exposure times for different phase angles. Note that for a relative velocity $v = 80\,\text{km}\,\text{s}^{-1}$, for phase angles within $\pm 48°$, the necessary exposure time is lower than the minimal possible exposure time of 0.22 ms. The actual exposure time of the sensor can only be set as multiples of 0.22 ms, for the final implementation, it is advised to take the closest value to the numbers calculated by the model.

3.2 Method

The exposure time was fixed as the upper limit set by the motion smear calculation. Using this, one set of signal strength for all possible filter centers with a constant width of $w = 100$ nm was calculated. As an approximation, the signal strength was linearly scaled for various bandwidths w and exposure times t_{exp}. This enables us to load this data set and determine the signal strength for each filter center, filter bandwidth and exposure time without having to numerically integrate over the wavelengths. We aim for a SNR of 100, which is not always possible, thus the algorithm is set up to maximize the SNR.

Algorithm 1 Bandwidth Solver

function GET_SNR(w_1, w_2, w_3, w_3)
  $S_{\text{BLU}} = S(c_2 - w_2/2 - w_1/2, w_1, t_{\text{exp}})$
  $S_{\text{ORA}} = S(c_2, w_2, t_{\text{exp}})$
  $S_{\text{RED}} = S(c_2 + w_2/2 + w_3/2, w_3, t_{\text{exp}})$
  $S_{\text{NIR}} = S(c_2 + w_2/2 + w_3 + w_4/2, w_4, t_{\text{exp}})$
  $\text{SNR}_i = \text{SNR}(S_i)$
  $w_{\text{sum}} = \sum w_i$
  return SNR_i, w_{sum}
end function
Adjust w_i to maximize SNR_i with constraint $w_{\text{sum}} = 700$ nm
Export c_i, w_i

The procedure is shown in Algorithm 1. We choose an initial filter, in this case the orange filter c_2. We focus on three possibilities as shown in Figure 3.5.

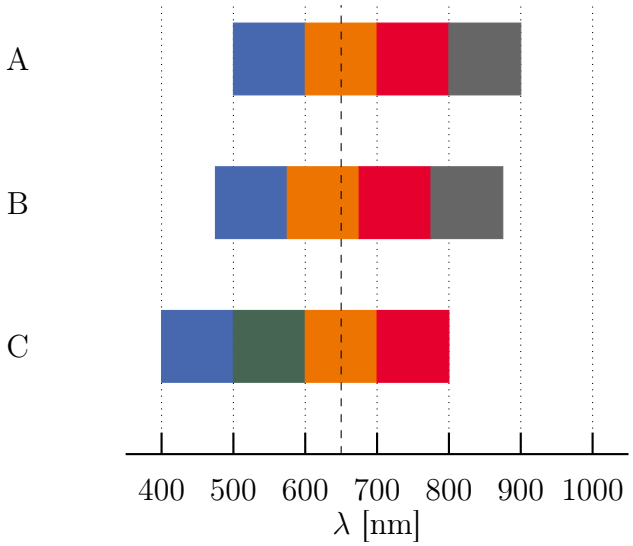


Figure 3.5: Option A will center the ORA filter at 650 nm, since this region is well covered as shown in Table 3.1. Option B sets the center of the ORA filter also as a free variable, but constraint within 575 and 675 nm. Option C will set the ORA filter at 650 nm, but set two filters on the left side (BLU and GRE) and just one filter on the right side (NIR). Since we want to cover the whole spectrum, it does not make sense to choose a filter in the 650 nm on the edge with the three other filters all on one side. The bandwidths of the filters are not displayed to scale.

The centers of the adjacent filters are now only determined by c_2 and the widths of the respective filters. The constraint $w_{\text{sum}} = 700\,\text{nm}$ is necessary to cover the whole spectrum from 350 nm to 1050 nm. We will discard option C and look in detail at options A and B with a blue (BLU), orange (ORE), red (RED) and near-infrared (NIR) filter, since option C would produce a very low SNR for the BLU and GRE (green) filter, while we would loose the red filter (RED). The selection is shown in Figure 3.11.
The flow diagram of this method is shown in Figure 3.6, we note here that the "integrate over wavelengths" step is simplified with the approximation mentioned above.

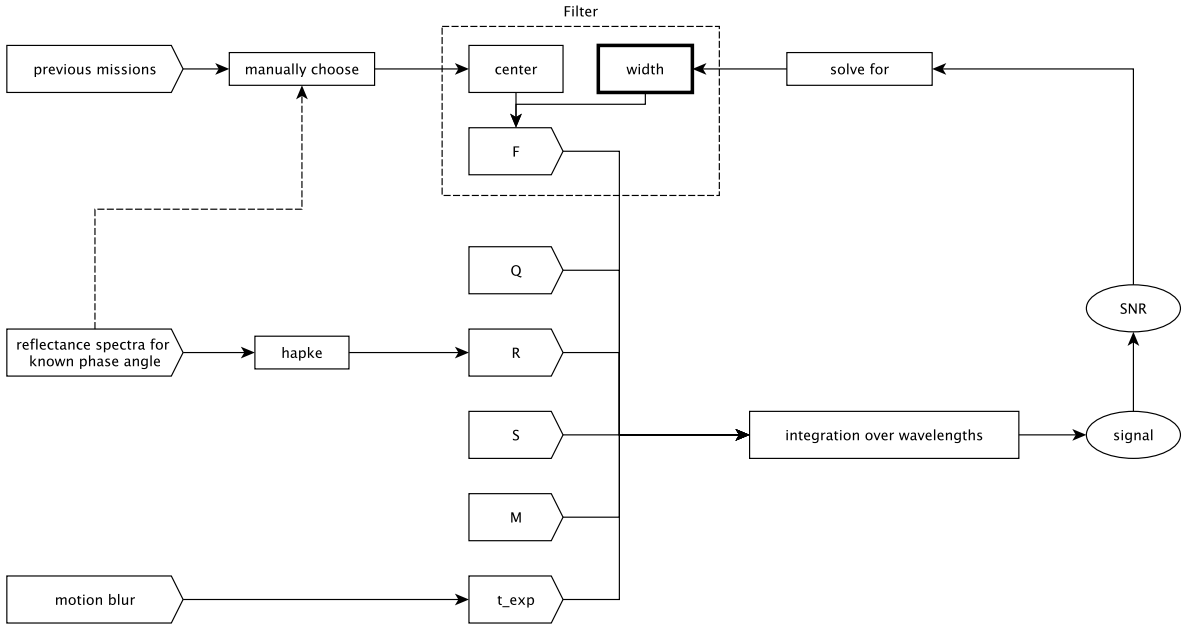


Figure 3.6: A flow diagram of the approach on selecting the filter centers and bandwidths. All parameters, including the exposure time, are fixed in a first approach. The filters are then selected by choosing a center and solving for the bandwidth.

3.3 Results

The algorithm is setup for a possible intercept with the parameters shown in Table 3.4.

r_h	v	b	t_{exp}
1 a.u.	30 km/s	1000 km	max

Table 3.4: Setup parameters for the filter selection. The exposure time has been chosen as the maximum possible for according to Table 3.3.

As mentioned in Section 3.1.4, the exposure time can be approximated to scale linearly with the ratio of the relative velocity and the impact parameter v/b. From equation (3.2) we see the dependence of the signal on the distance to the Sun r_h, as shown in equation (3.10):

$$S(\alpha) \propto 1/{r_h}^2 \tag{3.10}$$

It is also evident that the signal depends linearly on the exposure time. By knowing the coupling of all the parameters listed in Table 3.4, we are able to achieve similar SNR for intercept scenarios that differ from our nominal setup.
The selected filters can be imported to an interactive tool. The tool is platform independant as it is written in Python. The selected filters are displayed and the widths can be adjusted freely. The output of the filters derived with the orange filter fixed at 650 nm is given in Figure 3.7a. However, we notice that the SNR is not equivalent for all four filters as shown in Table 3.5.

filter	center [nm]	width [nm]	$\text{SNR}_{\alpha=11^\circ}$ [-] (rock)	$\text{SNR}_{\alpha=11^\circ}$ [-] (ice)
BLU	465.4	219.7	77.1	115.3
ORA	650.0	149.5	80.9	121.9
RED	781.6	113.7	74.6	110.9
NIR	946.9	217.0	75.4	112.3

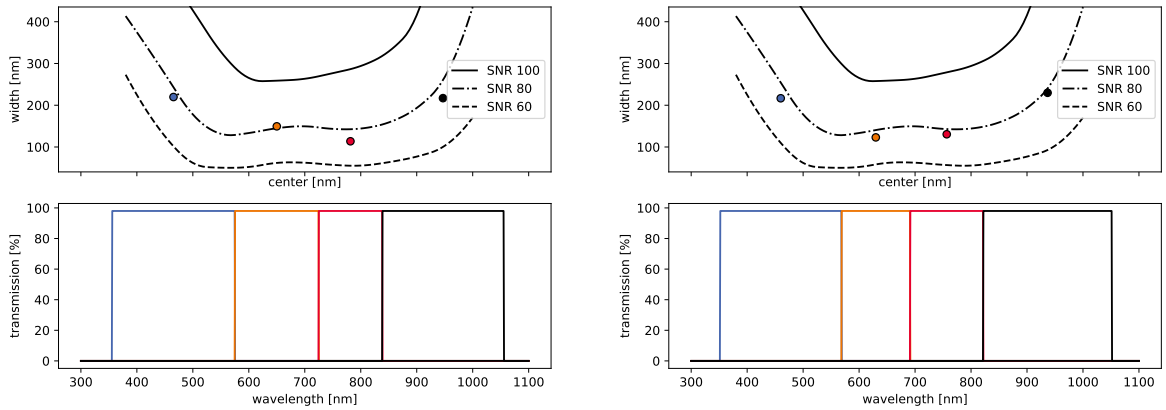
Table 3.5: The selected filters with orange being centered around 650 nm (option A). Note that the BLU and ORA filters have a higher SNR than the others.

A slightly different approach is to set the seed as a variable input like the widths in Algorithm 1. This results in a slightly different output as shown in Table 3.6 and Figure 3.7b.

filter	center [nm]	width [nm]	$\text{SNR}_{\alpha=11^\circ}$ [-] (rock)	$\text{SNR}_{\alpha=11^\circ}$ [-] (ice)
BLU	460.0	216.7	75.5	112.4
ORA	629.8	123.0	76.3	113.9
RED	756.6	130.5	77.7	116.3
NIR	936.7	229.8	78.7	118.0

Table 3.6: The selected filters with orange being shifted, but still covering 650 nm (option B). Note that all filters have a similar SNR.

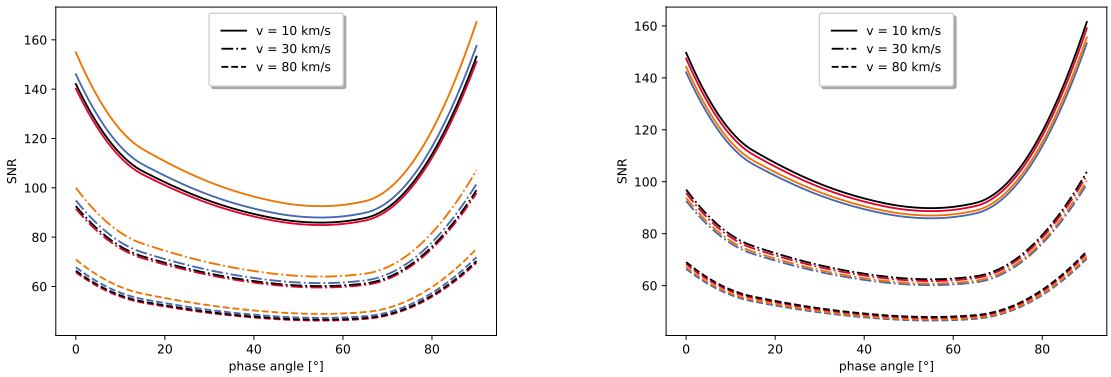
Since the exposure time is set to the maximum possible to eliminate motion smear, the SNR will vary for different phase angles. The calculated SNR for a rocky region is shown in Figure 3.8a and Figure 3.8b, and for an ice-rich region in Figure 3.9a and Figure 3.9b.



(a) ORA filter at 650 nm (option A)

(b) ORA filter not set (option B)

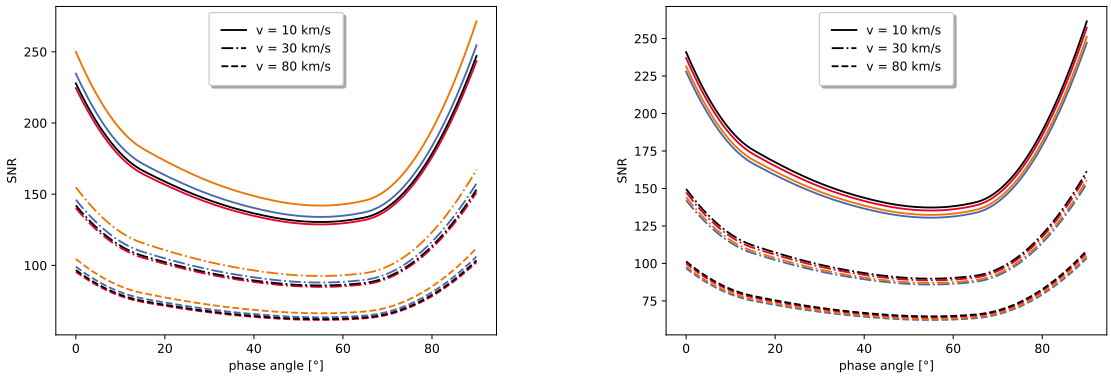
Figure 3.7: The output of the interactive tool for the two options. The upper plot shows the range of widths for a chosen center. The lower indicates the four filter centers and bandwidths. The SNR corresponds to a rocky region.



(a) ORA filter at 650 nm (option A)

(b) ORA filter not set (option B)

Figure 3.8: The SNR for a rocky region at 1 a.u. for each filter for different phase angles α and different relative velocities v. The SNR is highest as we approach the comet or are far away. This is because the reflectance increases at low phase angles while at large phase angles the motion smear decreases and thus the possible exposure time increases.



(a) ORA filter at 650 nm (option A)

(b) ORA filter not set (option B)

Figure 3.9: The SNR for a ice-rich region at 1 a.u. for each filter for different phase angles α and different relative velocities v. Note that since the reflectance of ice was set to roughly 3× the reflectance of rock, the SNR scales with a factor of $\approx \sqrt{3}$.

To always reach a SNR of 100, we need to adjust the exposure time t_{exp}. This can induce motion smear. To calculate the number of pixels involved, we used equation (3.11).

$$n_{\text{pixels}} = \frac{t_{\text{exp}} \cdot \omega(t)}{\sqrt{\Omega}} \tag{3.11}$$

Where $\omega(t)$ is given in equation (3.6), t is the corresponding time for a certain phase angle and Ω is the solid angle for one pixel. This lets us plot the motion smear for different phase angles as shown in Figure 3.10.

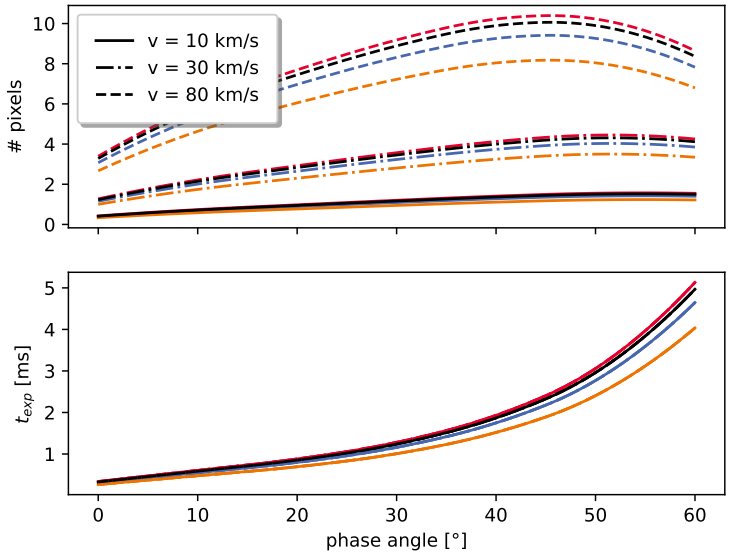
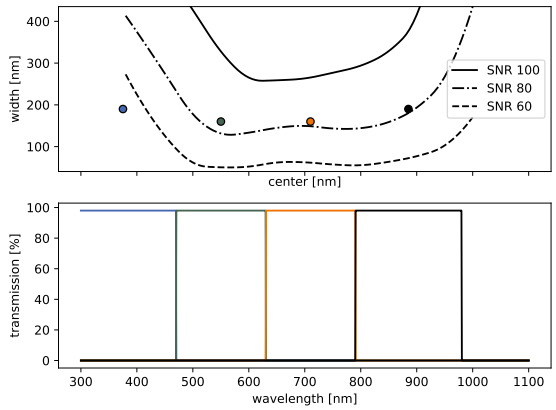
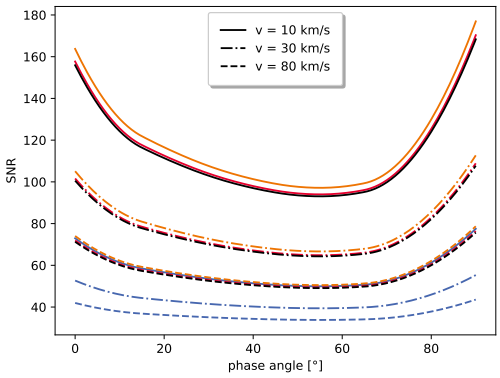


Figure 3.10: The exposure time required to achieve a SNR of 100 for a rocky region in the lower plot at 1 a.u. with the orange filter centered at 650 nm (option A). The number of pixels experiencing motion smear is shown for different relative velocities v in the upper plot. Motion smear seems to reach a local maximum at a phase angle of 40° - 60°.



(a) filters for option C



(b) SNR for option C

Figure 3.11: The filter selection for option C, where we have the orange filter in the 650 nm regime and two filters, BLU and GRE, on the lower end of the spectrum and NIR on the upper end. The SNR for the BLU filter is very low. This SNR corresponds to a rocky region.

4 Summary and Conclusion

4.1 Open Loop Controller

Preliminary tests on the prototype have shown that all the requirements from section 2 have been met. A movement from Filter BLU to Filter NIR and vice versa, which corresponds to 105°, can be performed within 0.422 seconds with a peak angular velocity of 4883 rpm and a peak angular acceleration of $\pm\,5326$ rad/s^2. Additionally, the backlash can be compensated within 0.037 seconds with a peak angular velocity of 651 rpm and a peak angular acceleration of $\pm\,5326$ rad/s^2. Both movements together can be performed within the required 0.5 seconds and do not exceed the velocity and acceleration limits of the motor. It follows that the smaller 35° and 70° movements can also be performed without violating the requirements. To prevent excessive wear on the motor, the two smaller movements are configured such that they also require 0.5 seconds, thus decreasing peak acceleration and velocity.
The accuracy of the movement has been measured on the prototype using a MAP4000 potentiometer which is attached to the filter wheel. The accuracy of the potentiometer is not entirely symmetrical, as it appears that the resistance is not truly linear. This explains why the values are more consistent on one side than on the other, and thus the offsets are larger for movements that go across both sides such as 70° and 105°.
The measured results are shown Table 2.2. For 35°- and 105°-degree movements, we see a slight overshoot, but for 70° we are slightly lower than the target. As mentioned before, the potentiometer to measure the angle is not truly linear and might deviate for different angles.

4.2 Filter Choices

The model is built to minimise the motion smear during the intercept of the comet. The phase angle dependent reflectance of the comet can be compensated by adjusting the exposure time accordingly in order to keep the SNR constant.
Two slightly different filter options, A and B, seen in Table 3.5 and Table 3.6, have been proposed. With the developed interactive tool they can easily be adjusted if needed. However, the model shows that the desired SNR value of 100 can not be reached in all conditions, as seen in Figure 3.8 and Figure 3.9. In Figure 3.9 we see that for an ice-rich region, the calculated SNR can exceed the peak linear capacity (SNR $\approx$ 189). This should not be an issue, since the exposure time can always be decreased.
The dependence of the SNR on the trajectory-specific parameters r_h, v, b has been determined. This enables us to calculate the SNR and adjust the exposure time t_{exp} for different scenarios. The dependence of the exposure time is summarized below:

$$t_{\text{exp}} \propto \frac{v}{b} \qquad\qquad t_{\text{exp}} \propto {r_h}^2$$

5 Outlook

As the open loop controller has only been tested for a couple of minutes at most, further long-term tests need to be conducted to verify the functionality of the open loop controller. The gears are required to be replaced for further tests in order to verify the backlash compensation with the flight-system gears. Furthermore, the two emergency stop Hall sensors must to be placed and connected and the two homing sensors need to be moved to the correct position of the flight system. The code will have to be improved to fit onto the Microsemi RT3PE600L chosen for the flight model. LVDS will be implemented in the future to replace the Microblaze – UART interface.
To read out the motor temperature, phase voltages and phase currents, the ADC readout should also be implemented.
As of now, there is no failure catch implemented in the FPGA. It will likely hang and require a reset if invalid t-parameters are sent. This problem should be addressed in the future.
The LUT for the movement parameters needs to be updated for the flight model, since essential parameters such as the sine table length, the clock- or PWM-frequency will likely be changed. Possible adaptations are further mentioned in section 2.4.

For the selected filters, the SNR and exposure time calculation can be improved by supplying a more accurate reflectance model. We should also note here, that the calculations are based on the data obtained from Rosetta's observation of 67P/Churyumov-Gerasimenko, a short-period comet, whereas CoCa will observe a DNC coming from the Oort cloud, which could have different surface properties. As the target is observed prior to the intercept, more information will be available to estimate the reflectance. The model is setup such that the reflectance can easily be changed.
Deviations from our model developed in this work can be compensated by adjusting the exposure time, at the expense of risking motion smear if the exposure time has to exceed the calculated limits.
One possibility to reduce motion smear is to rotate the spacecraft at the same angular velocity as calculated from equation (3.6), this would be the same principle as implemented in star trackers that astrophotographers use to compensate the Earth's rotation when taking images with long exposure times of astronomical objects.

6 Acknowledgement

This work would not have been possible without the support from various members of the Planetary Imaging Group and the staff of the University of Bern. I am especially thankful to Prof. Dr. Nicolas Thomas, my supervisor, who introduced me to the field of cometary research and has supported me in weekly meetings throughout this project.
I am grateful to all of those with whom I have had the pleasure to work during this project. I especially want to thank Claudio Zimmermann, who was eager to meet with me on a regular basis to discuss the progress on the open loop controller of the filter-wheel prototype, Matthias Lüthi, who spent countless hours introducing me to VHDL and Clément Feller and Antoine Pommerol who provided me with data and helped me implement the proper surface properties for the filter selection.
Additional thanks goes to Neville Mehta, Mirko Meier, Sébastien Hayoz and Thomas Beck who have supported me and discussed the progress of this work throughout the project in numerous meetings.
Last but not least, I want to thank my family and friends. They have provided me with tremendous emotional and scientific support.

References

[Altwegg et al., 2014] Altwegg, K., Balsiger, H., Bar-Nun, A., Berthelier, J. J., Bieler, A., Bochsler, P., Briois, C., Calmonte, U., Combi, M., Keyser, J. D., and et al. (2014). 67p/churyumov-gerasimenko, a jupiter family comet with a high d/h ratio. *Science*, 347(6220):1261952–1261952.

[Deshapriya, J. D. P. et al., 2018] Deshapriya, J. D. P., Barucci, M. A., Fornasier, S., Hasselmann, P. H., Feller, C., Sierks, H., Lucchetti, A., Pajola, M., Oklay, N., Mottola, S., Masoumzadeh, N., Tubiana, C., Güttler, C., Barbieri, C., Lamy, P. L., Rodrigo, R., Koschny, D., Rickman, H., Bertaux, J.-L., Bertini, I., Bodewits, D., Boudreault, S., Cremonese, G., Da Deppo, V., Davidsson, B. J. R., Debei, S., De Cecco, M., Deller, J., Fulle, M., Groussin, O., Gutierrez, P. J., Hoang, H. V., Hviid, S. F., Ip, W., Jorda, L., Keller, H. U., Knollenberg, J., Kramm, R., Kührt, E., Küppers, M., Lara, L., Lazzarin, M., Lopez Moreno, J. J., Marzari, F., Naletto, G., Preusker, F., Shi, X., Thomas, N., and Vincent, J.-B. (2018). Exposed bright features on the comet 67p/churyumov-gerasimenko: distribution and evolution. *A&A*, 613:A36.

[Egger, 2018] Egger, J. A. (2018). Colour variations on the nucleus of comet 67p/churyumov-gerasimenko.

[Eriksson, 2019] Eriksson, S. (2019). *Permanent Magnet Synchronous Machines*. MDPI AG.

[ESA, 2019] ESA (2019). ESA's new mission to intercept a comet. `https://sci.esa.int/web/cosmic-vision/-/61416-esa-s-new-mission-to-intercept-a-comet`. viewed on 9.9.2021.

[ESA, 2021] ESA (2021). Image of 67P/Churyumov–Gerasimenko. `https://www.esa.int/Space_in_Member_States/Germany/Rosettas_Komet_enthaelt_die_Bausteine_des_Lebens`. viewed on 6.9.2021.

[Faulhaber, 2021a] Faulhaber (2021a). *Faulhaber Brushless DC-Servomotor 1226...B Datasheet*.

[Faulhaber, 2021b] Faulhaber (2021b). *Faulhaber Planetary-Gear 12/4 Datasheet*.

[Feller et al., 2016] Feller, C., Fornasier, S., Hasselmann, P. H., Barucci, A., Preusker, F., Scholten, F., Jorda, L., Pommerol, A., Jost, B., Poch, O., and et al. (2016). Decimetre-scaled spectrophotometric properties of the nucleus of comet 67p/churyumov–gerasimenko from osiris observations. *Monthly Notices of the Royal Astronomical Society*, 462(Suppl 1):S287–S303.

[Fornasier, S. et al., 2019] Fornasier, S., Feller, C., Hasselmann, P. H., Barucci, M. A., Sunshine, J., Vincent, J.-B., Shi, X., Sierks, H., Naletto, G., Lamy, P. L., Rodrigo, R., Koschny, D., Davidsson, B., Bertaux, J.-L., Bertini, I., Bodewits, D., Cremonese, G., Da Deppo, V., Debei, S., De Cecco, M., Deller, J., Ferrari, S., Fulle, M., Gutierrez, P. J., Güttler, C., Ip, W.-H., Jorda, L., Keller, H. U., Lara, M. L., Lazzarin, M., Lopez Moreno, J. J., Lucchetti, A., Marzari, F., Mottola, S., Pajola, M., Toth, I., and Tubiana, C. (2019). Surface evolution of the anhur region on comet 67p/churyumov-gerasimenko from high-resolution osiris images. *A&A*, 630:A13.

[Fornasier, S. et al., 2015] Fornasier, S., Hasselmann, P. H., Barucci, M. A., Feller, C., Besse, S., Leyrat, C., Lara, L., Gutierrez, P. J., Oklay, N., Tubiana, C., Scholten, F., Sierks, H., Barbieri, C., Lamy, P. L., Rodrigo, R., Koschny, D., Rickman, H., Keller, H. U., Agarwal, J., A´Hearn, M. F., Bertaux, J.-L., Bertini, I., Cremonese, G., Da Deppo, V., Davidsson, B., Debei, S., De Cecco, M., Fulle, M., Groussin, O., Güttler, C., Hviid, S. F., Ip, W., Jorda, L., Knollenberg, J., Kovacs, G., Kramm, R., Kührt, E., Küppers, M., La Forgia, F., Lazzarin, M., Lopez Moreno, J. J., Marzari, F., Matz, K.-D., Michalik, H., Moreno, F., Mottola, S., Naletto, G., Pajola, M., Pommerol, A., Preusker, F., Shi, X., Snodgrass, C., Thomas, N., and Vincent, J.-B. (2015). Spectrophotometric properties of the nucleus of comet 67p/churyumov-gerasimenko from the osiris instrument onboard the rosetta spacecraft. *A&A*, 583:A30.

[Hapke, 2012] Hapke, B. (2012). *Theory of Reflectance and Emittance Spectroscopy*. Cambridge University Press, 2 edition.

[Haslebacher, 2020] Haslebacher, N. (2020). Development of a camera model for the comet camera (coca). Master's thesis, University of Bern.

[Hughes and Drury, 2013] Hughes, A. and Drury, B. (2013). *Electric Motors and Drives (Fourth Edition)*. Newnes, Boston, fourth edition edition.

[JPE, 2021] JPE (2021). Third Order Point-To-Point Motion Profile. `https://www.jpe-innovations.com/precision-point/third-order-point-to-point-motion-profile/`. viewed on 1.8.2021.

[Lüthi and Brüngger, 2017] Lüthi, M. and Brüngger, S. (2017). NIM Design Description, (JUI-UBE-PEP-DD-002). Technical report, University of Bern.

[Meier et al., 2021] Meier, M., Mehta, N., Piazza, D., Beck, T., and Thomas, N. (2021). Instrument Design and Performance Report (COMET-UBE-COC-DD-001). Technical report, University of Bern.

[NASA, 2021a] NASA (2021a). 1P/Halley. `https://solarsystem.nasa.gov/asteroids-comets-and-meteors/comets/1p-halley/in-depth/`. viewed on 6.9.2021.

[NASA, 2021b] NASA (2021b). Perseids. `https://solarsystem.nasa.gov/asteroids-comets-and-meteors/meteors-and-meteorites/perseids/in-depth/`. viewed on 6.9.2021.

[NREL, 2021] NREL (2021). MODTRAN Extraterrestrial Solar Spectra. `https://www.nrel.gov/grid/solar-resource/spectra.html`. viewed on 1.7.2021.

[Oklay, N. et al., 2015] Oklay, N., Vincent, J.-B., Sierks, H., Besse, S., Pajola, M., Bertini, I., Rickman, H., La Forgia, F., Barucci, A. M., Fornasier, S., Barbieri, C., Koschny, D., Lamy, P. L., Rodrigo, R., Agarwal, J., A´Hearn, M. F., Bertaux, J.-L., Cremonese, G., Da Deppo, V., Davidsson, B., Debei, S., De Cecco, M., Fulle, M., Groussin, O., Gutiérrez, P. J., Güttler, C., Hviid, S. F., Ip, W.-H., Jorda, L., Keller, H. U., Knollenberg, J., Kramm, J.-R., Kührt, E., Küppers, M., Lara, L. M., Lazzarin, M., Lopez-Moreno, J. J., Marzari, F., Michalik, H., Naletto, G., Thomas, N., and Tubiana, C. (2015). Characterization of OSIRIS NAC filters for the interpretation of multispectral data of comet 67P/Churyumov-Gerasimenko. *A&A*, 583:A45.

[Piotr, 2014] Piotr, W. (2014). *Dynamics and Control of Electrical Drives*. Springer Berlin.

[Ray L. Newburn, 2000] Ray L. Newburn, J. (2000). Calibration of the stardust navigation camera.

[RED, 2021] RED (2021). The Bayer Sensor Strategy. `https://www.red.com/red-101/bayer-sensor-strategy`. viewed on 16.8.2021.

[Sandford et al., 2006] Sandford, S. A., Aleon, J., Alexander, C. M. O., Araki, T., Bajt, S., Baratta, G. A., Borg, J., Bradley, J. P., Brownlee, D. E., Brucato, J. R., and et al. (2006). Organics captured from comet 81p/wild 2 by the stardust spacecraft. *Science*, 314(5806):1720–1724.

[Snodgrass and Jones, 2019] Snodgrass, C. and Jones, G. H. (2019). The European Space Agency's Comet Interceptor lies in wait. *Nature Communications*, 10(1):5418.

[Soderblom et al., 2000] Soderblom, L. A., Beauchamp, P. M., Chen, G.-S., Elliott, S. T., Fraschetti, G. A., Lee, M., Pain, B., Petrick, S. W., G.Ringold, P., Rodgers, D. H., Sandel, B. R., Soll, S. L., A.Thomas, D., and Wang, D. (2000). Miniature integrated camera spectrometer (micas). Technical report, JPL.

[Stöckli et al., 2021] Stöckli, L., Zimmermann, C., Beck, T., and Thomas, N. (2021). Open Loop Controller (COMET-UBE-COC-TN-006). Technical report, University of Bern.

[Sánchez et al., 2021] Sánchez, J. P., Morante, D., Hermosin, P., Ranuschio, D., Estalella, A., Viera, D., Centuori, S., Jones, G., Snodgrass, C., Levasseur-Regourd, A. C., and Tubiana, C. (2021). ESA F-Class Comet Interceptor: Trajectory design to intercept a yet-to-be-discovered comet. *Acta Astronautica*, 188:265–277.

[Teledyne-e2v, 2017] Teledyne-e2v (2017). *SIRIUS – CIS115 Back Illuminated CMOS Image Sensor Datasheet*.

[Thomas, 2020] Thomas, N. (2020). *An introduction to comets: post-Rosetta perspectives*. Springer.

[Thomas and Keller, 1990] Thomas, N. and Keller, H. U. (1990). Photometric calibration of the halley multicolour camera. *Appl. Opt.*, 29(10):1503–1519.

[Wüthrich, 2018] Wüthrich, J. (2018). Open-loop control of bldc-motor for space mass spectrometer. Master's thesis, EPFL.

[Wüthrich et al., 2018] Wüthrich, J., Gerber, M., and Wurz, P. (2018). Implementation of the NIM Shutter Mechanism Controller. Technical report, University of Bern.

[Xilinx, 2006] Xilinx (2006). *Spartan-3E FPGA Starter Kit Board User Guide (Xilinx UG230)*.

[Zimmermann and Meier, 2020] Zimmermann, C. and Meier, M. (2020). Filter Wheel Open-Loop Confidence Test (COMET-UBE-COC-TN-002). Technical report, University of Bern.

A Registers, HK and Commands

REG	name	value	type	content	size	description
0	new cmd	0 or 1	WT	[0]	1	trigger
	direction	0 or 1	RW	[1]	1	direction
	home	0 or 1	RW	[2]	1	home
	zero	0 or 1	RW	[3]	1	zero
	hall_config_mask	0b0110	RW	[7:4]	4	sensors
	heat_up	0 or 1	RW	[8]	1	heat mode
	cool_down	0 or 1	RW	[9]	1	cool mode
	ramp	0 to 63	RW	[15:10]	6	ramp up time
	power_idle	0 to 255	RW	[23:16]	8	idle power
	power	0 to 255	RW	[31:24]	8	power
1	idle_led	0 if busy	R	[0]	1	busy flag
		reserved		[4:1]	4	-
	status	0 to 5	R	[7:5]	3	status
2	t1	0 to 2^{17}	RW	[16:0]	17	t_1
3	t2	0 to 2^{17}	RW	[16:0]	17	t_2
4	t3	0 to 2^{17}	RW	[16:0]	17	t_3
5	t4	0 to 2^{17}	RW	[16:0]	17	t_4
6	t5	0 to 2^{17}	RW	[16:0]	17	t_5
7	t6	0 to 2^{17}	RW	[16:0]	17	t_6
8	t7	0 to 2^{17}	RW	[16:0]	17	t_7
9	t8	0 to 2^{17}	RW	[16:0]	17	t_8
10	accel_step	0 to 2^{17}	RW	[16:0]	17	accel_step
11						
12			deprecated			
13						
14	turns	-2^{31} to $+2^{31}$	R	[31:0]	32	full turns
15	angle	0 to `C_TAU`	R	[31:0]	32	angle
16			deprecated			
17	end_switches	0b1000	R	[3:0]	4	hall sensor

Table A.1: List of all registers implemented in the prototype. [Stöckli et al., 2021]

message	value	description
STATUS	0 - IDLE	status register
	1 - HOME	
	2 - MOVE	
	3 - EMG STOP LEFT	
	4 - EMG STOP RIGHT	
	5 - TIMEOUT	
LIMITS	0 - No Sensor	hall sensor readout
	1 - EMG LEFT	
	2 - HOME LEFT	
	3 - HOME RIGHT	
	4 - EMG RIGHT	
ANGLE	0 to `C_TAU`	angle
TURNS	-2^{31} to $+2^{31}$	turns

Table A.2: Housekeeping (HK) data that is sent back over UART. [Stöckli et al., 2021]

command	return	description
conf	"[OK] CONF"	set hall config
home	"[OK] HOME"	commence homing
	"[OK] target reached"	
move	"[OK] MOVE"	commence movement
	"[OK] target reached"	
heat	"[OK] HEATUP"	ramp up phases
	"[OK] target reached"	
cool	"[OK] COOLDOWN"	ramp down phases
	"[OK] target reached"	
zero	"[OK] ZERO"	zero turns register
read	"[OK] READ"	read out registers

Table A.3: Available commands to send over UART using the Python script. [Stöckli et al., 2021]

B Timing Diagram

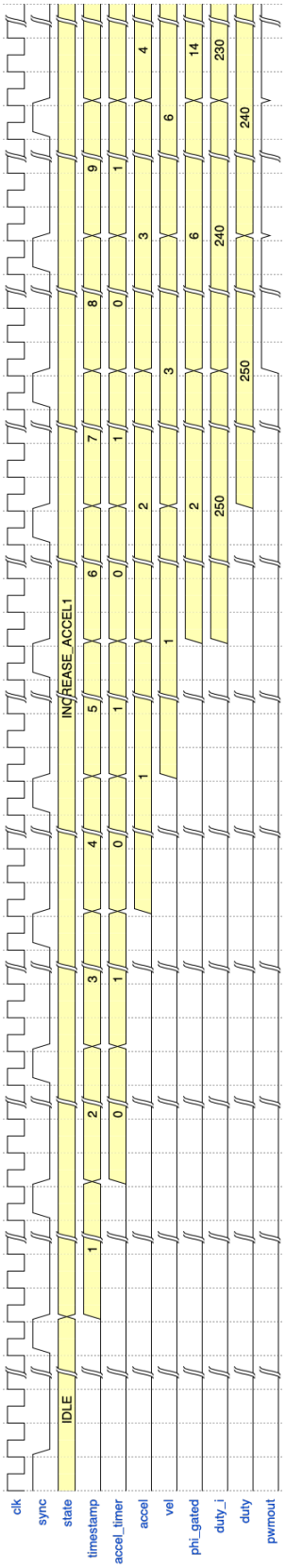


Figure B.1: Timing diagram based on the ModelSim simulations.

C Prototype Tests

φ_0 [°] ± 0.01°	φ_e [°] ± 0.01°	$\Delta\varphi$ [°] ± 0.01°
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.04	35.03
35.04	0.01	35.03
0.01	35.04	35.03

Table C.1: 35° Movements including backlash compensation resulted in an average distance moved of 35.037° ± 0.005°. Here φ_0 denotes the starting position, φ_e the end position and $\Delta\varphi$ the distance travelled. [Stöckli et al., 2021]

φ_0 [°] ± 0.01°	φ_e [°] ± 0.01°	$\Delta\varphi$ [°] ± 0.01°
35.11	0.82	34.29
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.09	34.27
35.09	0.82	34.27
0.82	35.09	34.27
35.09	0.82	34.27
0.82	35.09	34.27
35.09	0.82	34.27
0.82	35.09	34.27

Table C.2: 35° Movements without backlash compensation resulted in an average distance moved of 34.321° ± 0.006°. Here φ_0 denotes the starting position, φ_e the end position and $\Delta\varphi$ the distance travelled. [Stöckli et al., 2021]

φ_0 [°] ± 0.01°	φ_e [°] ± 0.01°	$\Delta\varphi$ [°] ± 0.01°
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.00	69.95
0.00	69.95	69.95
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.00	69.95
0.00	69.95	69.95
69.95	0.00	69.95
0.00	69.95	69.95
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.00	69.95

Table C.3: 70° Movements including backlash compensation resulted in an average distance moved of 69.944° ± 0.005°. Here φ_0 denotes the starting position, φ_e the end position and $\Delta\varphi$ the distance travelled. [Stöckli et al., 2021]

φ_0 [°] ± 0.01°	φ_e [°] ± 0.01°	$\Delta\varphi$ [°] ± 0.01°
105.20	0.01	105.19
0.01	105.21	105.20
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.20	105.20
105.20	0.00	105.20
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21

Table C.4: 105° Movements including backlash compensation resulted in an average distance moved of 105.207° +/- 0.006°. Here φ_0 denotes the starting position, φ_e the end position and $\Delta\varphi$ the distance travelled. [Stöckli et al., 2021]

movement #	target angle [°] ± 0.01°	encoder [inc] ± 2 counts
100	0.01	35855
150	0.02	35854
200	0.02	35854
250	0.02	35854
300	0.02	35853
350	0.02	35853
400	0.02	35854
450	0.02	35853
500	0.02	35854
550	0.02	35853
600	0.02	35854
650	0.02	35853
700	0.02	35855
750	0.02	35853
800	0.02	35854
850	0.02	35854
900	0.02	35853
950	0.03	35853
1000	0.03	35853

Table C.5: Extended run test of 105° back and forth where the target distance is noted down every 50 runs. The calculated angle on the FPGA remained constant. Additionally, the encoder position on the motor was measured. The measured 0.03° returned to 0.01° after 20 minutes of no movement. It is assumed that the drift was a result of a rise in temperature of the potentiometer. [Stöckli et al., 2021]

D Hapke Model

The following functions are used by the *Hapke* 2012 model. The single scattering phase function $P(\alpha, g)$, the H-function, the r-parameter and the surface roughness correction $S(\mu, \mu_0, \theta)$, which is detailed in *Hapke*'s book [Hapke, 2012].

$$P(\alpha, g) = \frac{1 - g^2}{(1 + 2g\cos(\alpha) + g^2)^{3/2}} \tag{D.1}$$

$$H(x, \omega) = \left[1 - \omega x \left(r(\omega) + \frac{1 - 2xr(\omega)}{2}\right) \log\left(\frac{1+x}{x}\right)\right]^{-1} \tag{D.2}$$

$$r(\omega) = \frac{1 - \sqrt{1-\omega}}{1 + \sqrt{1-\omega}} \tag{D.3}$$

$$S(\mu, \mu_0, \theta) = \frac{\mu'}{\eta}\frac{\mu_0}{\eta_0} \cdot \frac{\chi(\theta)}{1 - f + f\chi(\theta)\frac{\mu}{\eta}} \tag{D.4}$$

w	g	B_0	h_s	θ [°]	k
0.034	-0.42	2.25	0.061	28	1.2

Table D.1: The parameters for the *Hapke* 2012 model derived from Rosetta for 67P/Churyumov-Gerasimenko. [Fornasier, S. et al., 2015]

E CoCa Specifications

The quantum efficiency of the CIS115 [Teledyne-e2v, 2017] CMOS sensor is shown in Figure E.1 and Table E.1, the mirror reflection is shown in Figure E.2 and Table E.2.

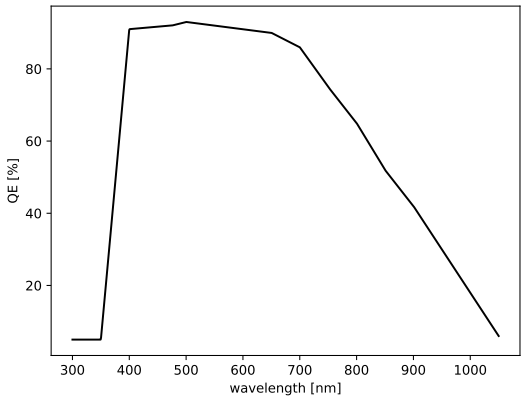


Figure E.1: The quantum efficiency $Q(\lambda)$ of the sensor.

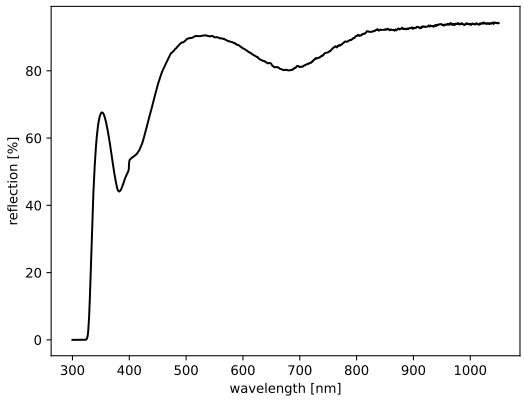


Figure E.2: The reflection $M(\lambda)$ of the mirrors.

The filter transmission has been modelled for a filter with center c and bandwidth w as shown in equation (E.1).

$$T(\lambda) = \begin{cases} 0.98 & c - w/2 < \lambda < c + w/2 \\ 0 & \text{else} \end{cases} \tag{E.1}$$

The raw data and Python programs can be found on GitHub: `https://github.com/hacknus/master_thesis`.

wavelength [nm]	QE [%]
300	5
350	5
400	91
450	92
500	93
550	92
600	91
650	90
700	86
750	75
800	65
850	52
900	42
950	30

Table E.1: The quantum efficiency of the sensor.

wavelength [nm]	transmission [%]
300	0
350	67
400	53
450	75
500	89
550	89
600	86
650	82
700	81
750	86
800	90
850	91
900	92
950	93

Table E.2: The transmission of the mirrors.

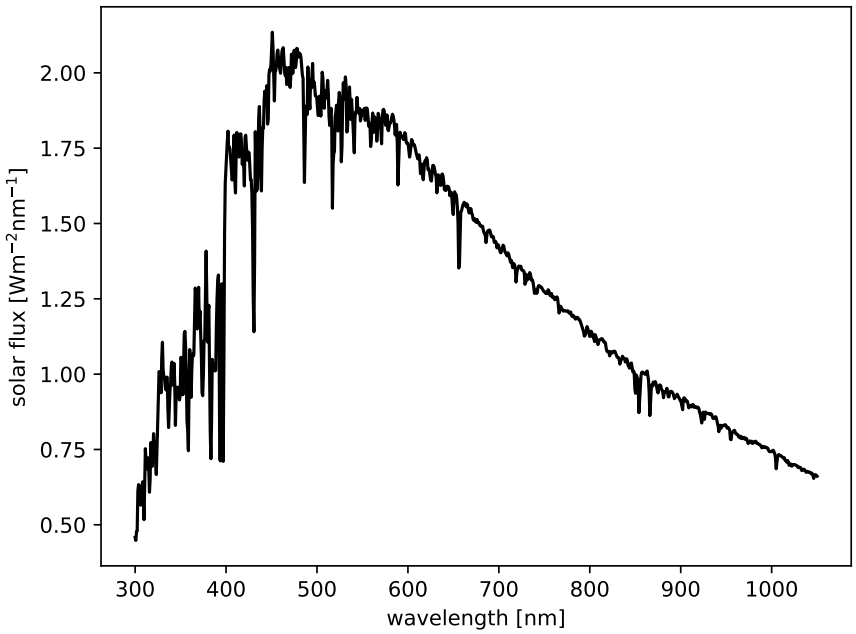


Figure E.3: The solar flux used in the calculations. [NREL, 2021]

parameter	value	unit	description
f	0.880	m	focal length
A_a	0.0573	m^2	aperture area
A_p	4.9×10^{-11}	m^2	pixel area
Ω	A_p/f^2	Sr	solid angle of one pixel
N	4	-	number of mirrors
G	3.5	e^-/DN	sensor gain
t_{exp}	> 0.22	ms	exposure time
r_h	0.85 - 1.35	A.U.	distance to Sun

Table E.3: List of CoCa parameters. [Meier et al., 2021]

CoCa

UNIVERSITÄT BERN

Camera (CoCa)

COMET-UBE-COC-TN-006

Issue 0 Revision 1

Open Loop Controller

Mandatory Cover Page Attributes:

Document Title:	Open Loop Controller
Origin Name:	Physikalisches Institut, Universität Bern, Linus Leo Stöckli
Date of Issue:	27. Aug. 2021
WBS code:	N/A
Package code:	N/A

CoCa Cover Page Attributes:

Old document reference:	
Restrictions:	None
(additional attributes)	

CoCa

UNIVERSITÄT BERN

Distribution List

Public	[ ]
CoCa (Through Project Manager)	[X]
Industry: specify in detail (Company Name, point of contact)	[ ]

Approval Sheet

	Name	Signature	Date
Prepared by:	Linus Leo Stöckli		28.8.2021
Verified by:	Claudio Zimmermann	Claudio Zimmermann Digital unterschrieben von Claudio Zimmermann DN: cn=Claudio Zimmermann, o=Uni Bern, ou=Space Research, email=claudio.zimmermann@space.unibe.ch, c=CH Datum: 2021.08.30 08:43:36 +02'00'	
Approved by:	Thomas Beck	Digitally signed by Thomas Beck Date: 2021.08.30 08:49:14 +02'00'	
Approved by:	Nick Thomas		

Document Change Record

Issue	Rev.	Date	Pages affected	Modification	DCR	Initials
0	1		all	(new document)	N/A	NME

DCR = document change request number

Table of Contents

List of documents

Applicable documents:	
AD1	

Reference documents:	
RD-1	**Filter Wheel Open-Loop Confidence Test (COMET-UBE-COC-TN-002)**
RD-2	**Implementation of the NIM Shutter Mechanism Controller (JUI-UBE-PEP-DD-005)**
RD-3	**Radiation-Tolerant ProASIC3 Low Power Spaceflight Flash FPGAs with Flash*Freeze Technology**
RD-4	**JUI-UBE-PEP-DW-007, i1.1 Schematics WP170207A_NIM-Ctrl**
RD-5	**Spartan-3E FPGA Starter Kit Board User Guide (Xilinx UG230)**
RD-6	**Open-Loop Control of BLDC-Motor for Space Mass Spectrometer (Master Thesis, J. Wüthrich)**
RD-7	**Faulhaber Brushless DC-Servomotor 1226…B Datasheet**
RD-8	**Faulhaber Planetary-Gear 12/4 Datasheet**
RD-9	**Precision Point, Third Order Point-To-Point Motion Profile, JPE**

1 General

1.1 Scope of the Document

The scope of this TN is to document the development of the FPGA design for phase A/B.

1.2 Abbreviations and Acronyms

For general abbreviation, refer to CoCa's List of Abbreviations.

ADC – Analog Digital Converter
BLDC – Brushless Direct Current Motor
DN – Digital Numbers
DPM – Digital Processing Module
FPGA – Field Programmable Gate Array
LUT – Look Up Table
LVDS – Low Voltage Differential Signaling
NIM – Neutral gas and Ion Mass spectrometer
PLB – Processor Local Bus
PMSM – Permanent Magnet Sync Motor
PWM – Pulse Width Modulation
SoC – System on Chip
UART – Universal Asynchronous Receiver Transmitter
VHDL – Very High-Speed Integrated Circuit Hardware Description Language

2 Introduction

This document focuses on the implementation of an open loop controller for the filter wheel on the Comet Camera (CoCa) on the Comet Interceptor mission. Open loop has been chosen instead of closed loop due to simplicity while still being accurate enough to fulfil the requirements.

2.1 Functional Requirements

The top-level requirement is to move the filter wheel with variable acceleration and velocities. The following conditions must be fulfilled:

- The acceleration must not exceed 6000 rad/s^2 and the velocity must not exceed 5000 rpm.
- The FPGA must move the filter wheel 35°, 70° and 105° to switch between filters. All movements must be completed within 0.5 seconds.
- The accuracy must lie within +/- 0.19°.
- Power must be variable.
- The system should be based on the NIM project and must thus be built around a BLDC from Faulhaber motor of type No: 1226A012B 124 16:1

A top view of the filter wheel is shown in Figure 1. Certain parameters such as Hall sensor placement and backlash are different for the prototype than for the flight model.

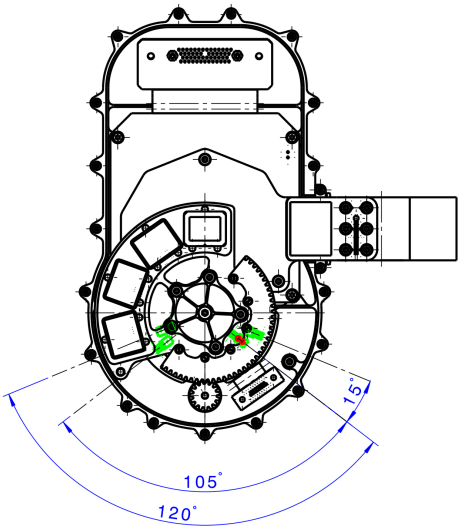


Figure 1: Top view of the filter wheel flight model.

2.2 Hardware requirement

The electronics hardware that controls the motor is the breadboard used by the NIM project.

3 Setup

The System is developed on the shutter controller of the NIM project. To control the filter wheel to an accuracy of 0.19°, multiple error contributions must be taken into consideration. The main contributions are backlash, end switch detection and arithmetic errors.

3.1 Mechanics

A similar, but more powerful, BLDC motor as used in the NIM project has been chosen. The motor and planetary gearbox assembly is ordered from Faulhaber (No: 1226A012B 124 16:1) [RD-7, RD-8]. The reduction is 16:1, a second gear stage with a reduction ratio of 5:1 is attached to the planetary gearbox, yielding a total gear reduction of 80:1. The motor has an outer diameter of 12mm, maximum torque of 2.6 mNm and an operational voltage of 12V. Technically this is a PMSM motor, which is controlled by three sinusoidal curves. There are 3 analog hall sensors with 120° dephasing built into the round motor PCB. This analog 3 phase sinus signal is then converted by the controller into the absolute digital position. The controller to readout the built in Hall-Effect Sensor Position is the MCBL3002 S RS, Art. No: 6500.01486. The encoder is accurate to 360/3000° which corresponds to 0.0015° on the filter wheel. A MAP4000 potentiometer is attached to the filter wheel to measure the position of the filter wheel itself. The potentiometer is accurate to +/- 0.01°. The backlash is only visible on the potentiometer.

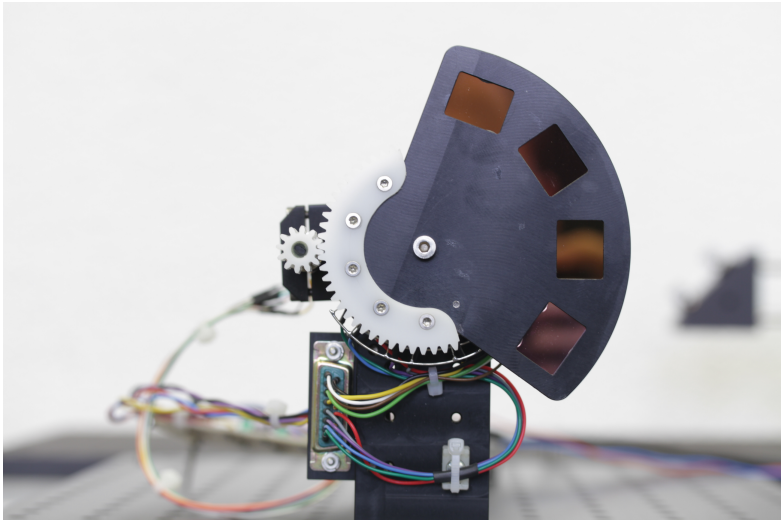


Figure 2: The filter wheel prototype with the Faulhaber motor connected to it.

3.2 Electronics

The electronics hardware of the prototype is the same as used in the NIM project. Because NIM only uses two end-switches, the prototype for the filter-wheel only uses two of the four hall sensors. The emergency stop end-switches are unconnected, only the homing hall switches are used.

3.3 Code

The code can be found on the Git-Repository:

https://spacegit.unibe.ch/cometinterceptor/coca/coca-motor-ctrl-test

4 Motion Profiles

To move the shutter, the NIM software used a trapezoidal velocity profile as shown in Figure 4. This will however introduce a lot of vibrations since the jerk is in theory a delta peak, in practice the jerk will certainly be finite, but still high, which can lead to faster ware on the structure.

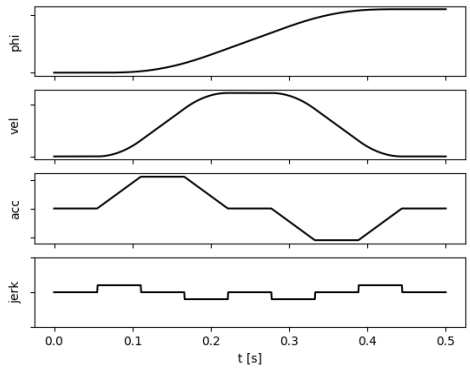


Figure 3: A general third order profile.

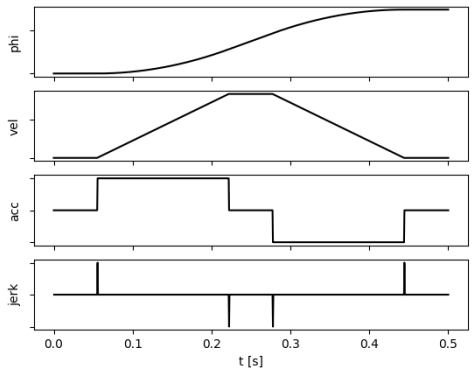


Figure 4: A general second order profile.

To mitigate this, a third order (s-curve velocity) motion profile was chosen, which leads to a low finite jerk value. Since the slope for the acceleration can be chosen, the jerk can be directly controlled for each movement. This reduces vibrations and ware on the structure but requires a more advanced control system. For simplicity, the profiles are symmetrical.

Third order (S-curve)	
+ less jerk/vibrations + less wear/tear	- more complex - slower
Second order (trapezoidal)	
- more jerk/vibrations - more wear/tear	+ simple + faster

The calculation of the different parameters that define a third order profile is given in the following equations [RD-9] and the parameters are shown in Figure 5.

$$t_1 = 0$$

$$t_2 = \frac{a_{\max}}{m} = a_{\max} \cdot \text{accel\_step}$$

$$t_3 = \frac{v_{\max}}{a_{\max}}$$

$$t_4 = t_1 + t_2 + t_3$$

$$t_5 = \frac{\varphi_{\max}}{v_{\max}}$$

$$t_6 = t_5 + t_2$$

$$t_7 = t_5 + t_3$$

$$t_8 = t_5 + t_4$$

Where m is the slope of the acceleration, $a_{\max}$ is the maximum acceleration, $v_{\max}$ is the maximum velocity $\varphi_{\max}$ the target angle.

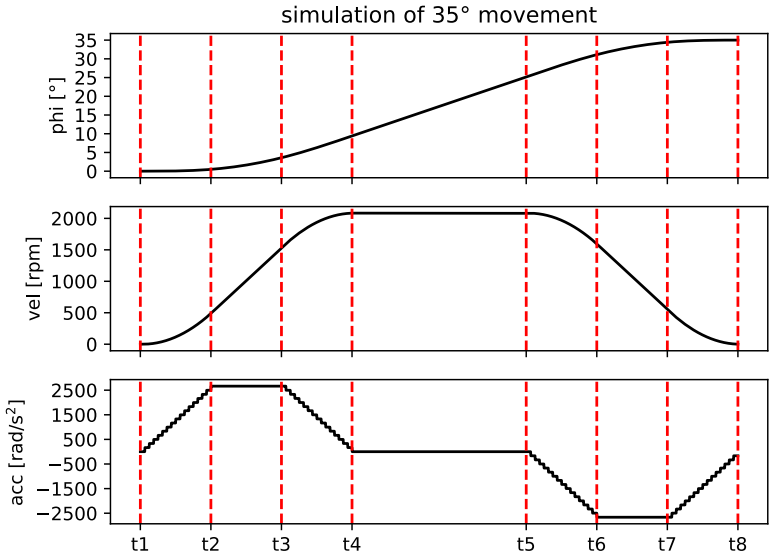


Figure 5: A third order profile simulation for a 35°-degree movement on the prototype with maximum acceleration $a_{\max} = 2600\ \mathrm{rad/s^2}$ *and maximum velocity* $v_{\max} = 2000\ rpm.$

Since this calculation requires division and multiplication, it is offloaded from the FPGA and an updatable look up table with the required parameters is located on the DPM.

To mitigate arithmetic errors, the time parameters t_i and the acceleration timer need to be integers. For that to be the case, only certain maximum velocities and accelerations are allowed. The generation of the look up table is described in section 4.2 .

Since the movement is defined by integers only, no arithmetic errors can build up, the error on the output solely depends on the accuracy of the sine table. For the prototype a sine-table of length 360 has been chosen. This can likely be reduced for the flight model later. The length must be dividable by three to generate the offset for the three phases.

The parameters that define a single movement are given in Table 1.

parameter	*description*	*bit size*
t1, t2, t3, …, t8	Time parameters	8x 17
power	Power during movement (0 to 255)	8
power_idle	Hold Power during IDLE (0 to 255)	8
accel_step	Acceleration steps, inverse of the slope of the acceleration	17
direction bit	0 or 1 to set the direction	1
ramp	Ramp up time (0 to 63)	6
movement bit	0 or 1 to trigger movement	1

Table 1: Parameters that define a movement. A total of 177 bits needs to be set in the registers for one movement. Ramp parameter may be excluded and hard coded in the future.

4.1 Homing

In homing mode, the statemachine will not transition from t_4 to t_5 unless the hall sensor is reached. Thus, the filter-wheel will travel the remaining distance after hitting the end-stop. As seen in Figure 1, the hall sensors are mounted in the centers of the filters, however measurements have shown [RD-1] that the hall sensors are activated at 1.55mm away. The hall sensors are mounted at a radius of 22.5 mm. This means, the remaining distance that the filter-wheel must travel is given by

$$\varphi = \frac{1.55}{22.5} \cdot \frac{180°}{\pi} = 3.95°$$

The parameters for the movement need to be adjusted such that the filter-wheel travels 3.95° between t_5 and t_8. This has been done by trial and error and not fully tested on the prototype, since the positioning of the hall-sensors is not equivalent to the final flight model.

4.2 Look-Up-Table

The look up table is constructed using the "**simulation.py**" python script. Inside this script, the function "**get_possible_params**" from "**integer_params.py**" is called, which returns a 3-dimensional structure containing all possible acceleration, velocity and acceleration-step values. Depending on the requirement for the movement (fast/slow acceleration, fast/slow speed) we have to manual select one of each.

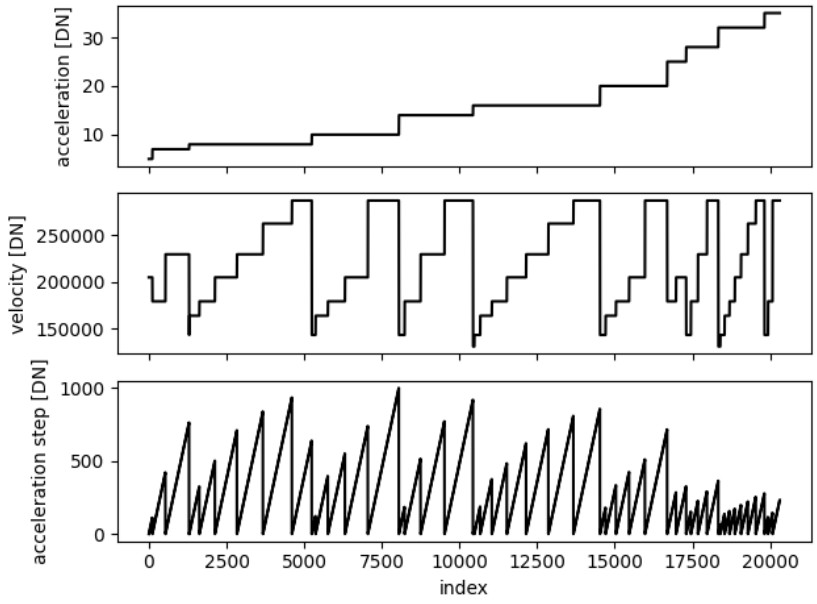


Figure 6: All parameter possibilities for acceleration, velocity and acceleration step in digital numbers (DN). This output is produced by the get_config() function in the simulation.py script.

To select a combination for the simulation, one must choose an index. This can either be done by looking at the graph or by sorting the values, thus limiting the possibilities, as shown in Figure 7. In general, the **accel_step** is the inverse of the slope for the acceleration, hence

a higher **accel_step** value will lead to a flatter slope. The higher the acceleration value, the steeper the velocity slope.

```
fastest_v = np.max(v)
a = a[v == fastest_v]
accel_steps = accel_steps[v == fastest_v]

seq = a.argsort()
a = a[seq]
accel_steps = accel_steps[seq]

if amax is None: amax = a[index]
if vmax is None: vmax = fastest_v
if accel_step is None: accel_step = accel_steps[index]
```

Figure 7: Simple code segment to select the fastest velocity and then sort the accelerations. In the end an index = 1 and accel_step = 600 were chosen, to select a combination that is suitable for a 35° movement.

The selected values will then be used to construct all the required parameters as described in Table 1. A simulation will then run with these parameters, eventually a plot of position, velocity and acceleration is saved. The parameters are saved in the look-up-table LUT.json, which is used to control the hardware.
As mentioned in section 4.1 Homing, the parameters for homing are chosen such that the overshoot equals 3.95°. There are many possible solutions, we fixed a low velocity and from there on started adjusting the acceleration and acceleration step parameters through trial and error.

5 FPGA-Architecture

The prototype is built around a Spartan-3E development board. The code is written in VHDL using the Xilinx ISE 14.7 software package. The system is based on the NIM shutter controller [RD-6], which was also created using ISE 14.7, thus we also use a Microblaze softcore processor and a PLB bus to connect it to the statemachine entity. The CoCa controller is lighter since it does not use the ADCs on the motor ctrl breadboard.

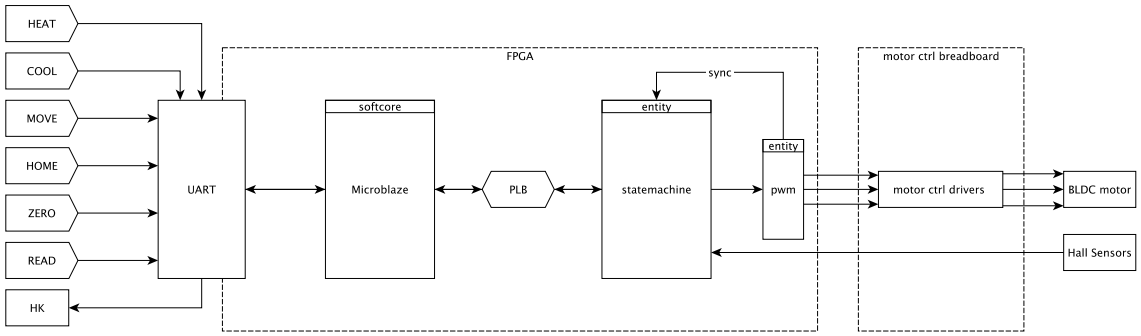


Figure 8: High level block diagram of the setup.

5.1 Statemachine entity

The architecture is built around the two-process-method with a record container. We have the clocked process **pc_fsm** that updates the **pres** record with the **nxt** record. The combinatorial process **p_fsm** contains the state-machine and is synced to the **pwm entity** as shown in Figure 9. Minor adaptations had to be made to the **pwm entity**[1], because ISE 14.7 does not support the latest VHDL standard. A sync output has also been implemented, which is pulled high for one clock cycle, during every PWM cycle. Thus, the state-machine will increase the timestamp at each PWM cycle. The state-machine is then updated according to Figure 10.

[1] https://spacegit.unibe.ch/cometinterceptor/coca/coca-motor-ctrl-test/-/tree/master/src/_local/pwm

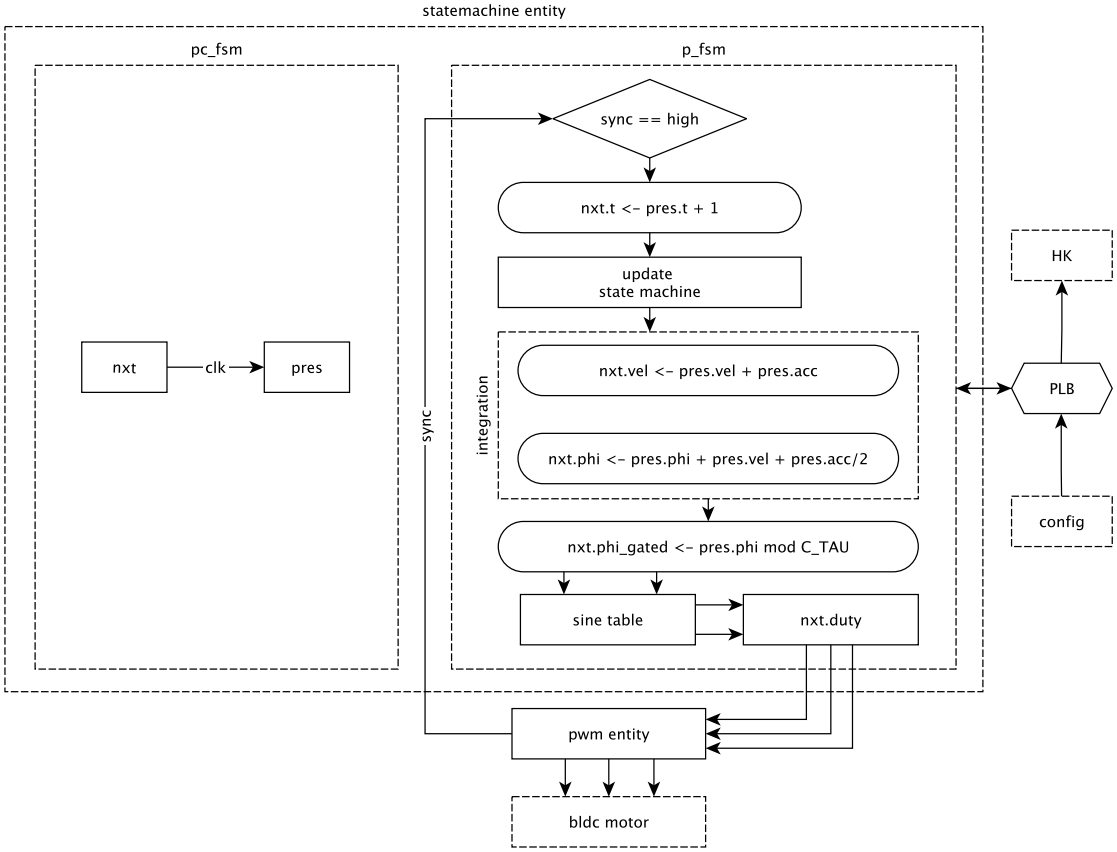


Figure 9: Simplified diagram of the statemachine entity (statemachine.vhd). The flow diagram for updating the state machine is shown in Figure 10.

The then integrated angle φ_1 is gated to prevent an over-indexing of the sine-table, this is essentially just a modulo calculation which had to be implemented with case statements because the modulo operator was not available in Xilinx ISE 14.7. Another angle φ_2, which is shifted by $360/3$ is calculated and the sine value is calculated for both.

$$\varphi_1 = \varphi_1 \bmod 360$$
$$\varphi_2 = (\varphi_1 + 360/3) \bmod 360$$

The duty cycles are then calculated using the set power P according to:

$$\mathrm{PWM}_1 = \frac{T_{\mathrm{PWM}}}{2}(1 + P \cdot \sin(\varphi_1))$$
$$\mathrm{PWM}_2 = \frac{T_{\mathrm{PWM}}}{2}(1 + P \cdot \sin(\varphi_2))$$

For the prototype with 50 MHz clock frequency the PWM frequency is set to 200 kHz and thus $T_{PWM} = 250$ times a clock period.

The duty cycle of the third phase is calculated using

$$\mathrm{PWM}_3 = \frac{T_{\mathrm{PWM}}}{2}(1 - P \cdot \sin(\varphi_1) - P \cdot \sin(\varphi_2))$$

The sine values are taken from a table, which is initialised as a constant with length 360. No further tests have been conducted to deduce what the minimum functional length is, but it is assumed that the length can be reduced to conserve memory while having little to no influence on the overall accuracy.

$$\mathrm{sin\_table}[i] = PWM_{\mathrm{width}} \sin\left(\frac{2\pi}{360} \cdot i\right)$$

As an alternative, the CORDIC algorithm has been analysed but the sine-table approach was chosen due to its simplicity and because enough memory is available on the FPGA.

The **pwm entity** uses the three duty cycle values to update the PWM at the next PWM cycle. The three sine modulated PWM signals are then amplified by Power MOSFET push-pull stages. A low-pass LC filter to demodulate the PWM signal into pure sine signals then filters each signal (see [RD-4] for schematics). The resolution of the system is determined by the length of the sine table:

$$\mathrm{resolution} = \frac{360°}{360 \cdot 80} = 0.0125°$$

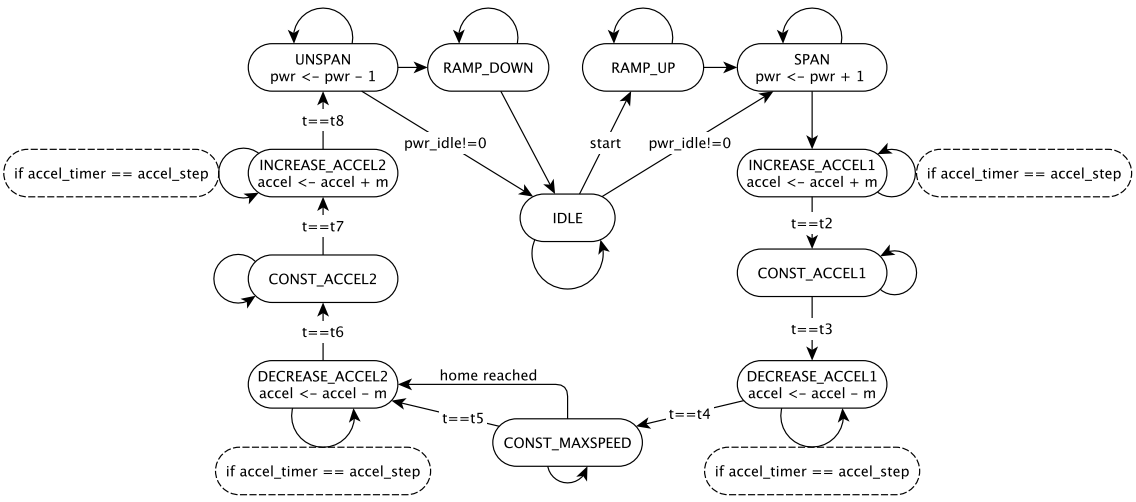


Figure 10: Simplified block diagram of the state-machine for a movement.

A closer look at the state-machine during a single movement is shown in Figure 10. Depending on the configurations, the phases need to be first ramped up and spanned up before entering INCREAS_ACCEL1 state. There the acceleration value will only get updated if the **accel_timer**, which increases at the same rate as the timestamp *t*, has reached the **accel_step** value. If the timestamp has reached t_1, it will continue to the next state and so on. It is important to note that when in homing mode, the state-machine remains in CONST_MAXSPEED state until either a timeout is reached, or a hall-sensor is reached. If any movement is taking longer than 2 seconds, it will automatically revert into IDLE mode and set the status flag to timeout.

5.2 Registers

The FPGA registers are described in Table 2. All registers are 32 bits wide, but not always all 32 bits are used. Bit 0 is LSB and bit 31 is MSB.

The address of a given register is calculated as *REG_ADDR = BASE_ADDR + 4*REG*. The Microblaze uses Byte addressing, while every register is 4 Byte wide, hence the factor 4.

REG	*name*	*value*	*type*	*content*	*bit size*	*description*
0	new_cmd	0 or 1	WT	[0]	1	Trigger new command
	direction	0 or 1, 0 for clockwise rotation	RW	[1]	1	Set the direction of the movement
	home	0 or 1	RW	[2]	1	Set the home mode
	zero	0 or 1	RW	[3]	1	Set current position to zero
	hall_config_mask	e.g. 0b0110	RW	[7:4]	4	Mask to select hall sensors
	heat_up	0 or 1	RW	[8]	1	Set the heat-up mode
	cool_down	0 or 1	RW	[9]	1	Set the cool-down mode
	ramp	0 to 63	RW	[15:10]	6	Ramp up time of the three phases from 0 to 6V
	power_idle	0 to 255	RW	[23:16]	8	Power idle value
	power	0 to 255	RW	[31:24]	8	Power value
1	idle_led	0 if busy	R	[0]	1	Busy flag
		reserved		[4:1]		-
	status	0 to 5	R	[7:5]	3	Status value

2	t1	0 to 131071	RW	[16:0]	17	t_1
3	t2	0 to 131071	RW	[16:0]	17	t_2
4	t3	0 to 131071	RW	[16:0]	17	t_3
5	t4	0 to 131071	RW	[16:0]	17	t_4
6	t5	0 to 131071	RW	[16:0]	17	t_5
7	t6	0 to 131071	RW	[16:0]	17	t_6
8	t7	0 to 131071	RW	[16:0]	17	t_7
9	t8	0 to 131071	RW	[16:0]	17	t_8
10	accel_step	0 to 131071	RW	[16:0]	17	accel_step
11						
12		*deprecated*				
13						
14	turns	-2147483648 to +2147483648	R	[31:0]	32	Full turns since 0
15	angle	0 to +1509949439	R	[31:0]	32	Angle of not completed turn
16		*deprecated*				
17	end_switches	e.g. 0b1000	R	[3:0]	4	Raw hall sensor value

Table 2: FPGA registers of the Open Loop Controller. Bits [4:1] of register 1 contain the debug LED states. Registers 11, 12, 13 and 16 are deprecated and not used anymore. WT stands for write-trigger, R stands for read-only and RW stands for read-write.

The Registers are handled in the PIF[2] (processor interface package) which defines the signals between the core and the processor interface.

5.3 ModelSim Simulation

To simulate the design on a computer before implementing the design on the FPGA, the ModelSim environment has been chosen. The testbench[3] is also written in VHDL and will run the **statemachine** and **pwm** entities on the computer as a simulation. All signals are visualized

[2] https://spacegit.unibe.ch/cometinterceptor/coca/coca-motor-ctrl-test/-/tree/master/src/_local/statemachine/src/pif_src

[3] https://spacegit.unibe.ch/cometinterceptor/coca/coca-motor-ctrl-test/-/tree/master/src/_local/statemachine/tb

in a wave-viewer and can be exported as a text file to analyse further. The testbench does not include the microblaze softcore and thus all registers need to be set in the testbench itself.

The simulations have proven that there are no arithmetic errors and the integrated position counter will reach the expected values.

The simulation was needed to verify the timing diagram shown in Appendix C: Timing Diagram.

5.4 Timing Constraints

A timing diagram can be seen in the Appendix in Figure 16. It is clear to see that all state changes are synced to the PWM sync signal. Each signal must be updated within 1/200kHz = 5 us, for that we need to set timing constraints:

```
TIMESPEC TS_MULTICYLCE_DURATION = PERIOD 5 us;

TIMESPEC TS_MULTICYCLE_1 = FROM FFS("i_core/pres.state_*") TO
FFS("i_core/pres.timestamp_*") TS_MULTICYLCE_DURATION;

TIMESPEC TS_MULTICYCLE_2 = FROM FFS("i_core/pres.timestamp_*") TO
FFS("i_core/pres.accel_timer_*") TS_MULTICYLCE_DURATION;

TIMESPEC TS_MULTICYCLE_3 = FROM FFS("i_core/pres.accel_timer_*") TO
FFS("i_core/pres.accel_*") TS_MULTICYLCE_DURATION;

TIMESPEC TS_MULTICYCLE_4 = FROM FFS("i_core/pres.accel_*") TO
FFS("i_core/pres.vel_*") TS_MULTICYLCE_DURATION;

TIMESPEC TS_MULTICYCLE_5 = FROM FFS("i_core/pres.vel_*") TO
FFS("i_core/pres.phi_gated_*") TS_MULTICYLCE_DURATION;

TIMESPEC TS_MULTICYCLE_6 = FROM FFS("i_core/pres.phi_gated_*") TO
FFS("i_core/pres.duty_?_i*") TS_MULTICYLCE_DURATION; # duty <- v_phi <- phi_gated

TIMESPEC TS_MULTICYCLE_7 = FROM FFS("i_core/pres.accel_*") TO
FFS("i_core/pres.duty_?_i*") TS_MULTICYLCE_DURATION; # duty <- v_phi <- accel

TIMESPEC TS_MULTICYCLE_8 = FROM FFS("i_core/pres.vel_*") TO
FFS("i_core/pres.duty_?_i*") TS_MULTICYLCE_DURATION; # duty <- v_phi <- vel

TIMESPEC TS_MULTICYCLE_9 = FROM FFS("i_core/pres.duty_?_i*") TO
FFS("i_core/pres.duty_?*") TS_MULTICYLCE_DURATION;

TIMESPEC TS_MULTICYCLE_10 = FROM FFS("i_core/pres.state_*") TO
FFS("i_core/pres.duty_?*") TS_MULTICYLCE_DURATION;  # state IDLE means duty <-
zero_DURATION;
```

Since **phi_gated, accel** and **vel** are needed to calculate the variable **v_phi**, and **v_phi** is used to calculate the new signal **duty**, all three need to be constraint.
The last constraint is not used during the movements, but in IDLE state where the condition (state = IDLE) sets the duty values to C_ZERO_DUTY.

It is evident that the movement is delayed by 7 PWM-cycles. This is only relevant in the homing mode, where the deceleration only starts $7 \cdot 5\,\mathrm{us}$ after the hall sensor signal is recognized. Thus, the filter wheel will overshoot the zero position by 0.015°. This can be compensated for when choosing the homing parameters described in subsection 4.1 .
This overshoot has not been compensated in the test on the prototype.

5.5 Resource Usage and Connections

The resources used are shown in Table 3 and Table 4.

Device	*Package*	*Sine Table Length*	*Slice*	*Slice-Flip-Flop*	*LUT*	*Multiplier*
xc3s500e	fg320	360	470 [19%]	905 [5%]	1447 [15%]	2 [10%]

Table 3: Resource usage of the Xilinx Spartan-3E prototype. Only statemachine.vhd has been synthesized.

Device	*Package*	*Sine Table Length*	*D-Flip-Flops [statemachine]*	*D-Flip-Flops [pwm]*
RT3PE600L	484 CCGA	360	440 [3.2%]	40 [0.3%]
RT3PE600L	484 CCGA	300	441 [3.2%]	40 [0.3%]
RT3PE600L	484 CCGA	258	441 [3.2%]	40 [0.3%]

Table 4: Resource usage of the Microsemi flight model for different sine table lengths. The sine table length does only have an influence on the RAM blocks, not on the flip-flops. Note that a different package was used for these estimations, since the code needs to be optimized (for now, all signals are mapped to ports, thus increasing the I/O usage) to fit into the actual flight package [RD-3]. The resource usage of the statemachine and pwm entities are shown.

The necessary peripherals are connected to the Hirose FX2 connector as shown in Table 5.

FPGA Pin	*FX2 Pin*	*Board Signal*	*Purpose*
B11	A36	bldc_ctrl_0_pwm_a_pin	Motor PWM A
A11	A37	bldc_ctrl_0_pwm_b_pin	Motor PWM B
G9	A39	bldc_ctrl_0_pwm_c_pin	Motor PWM C
E13	A34	bldc_ctrl_0_end_switches_pin[2]	Homing Sensor 1
C4	A35	bldc_ctrl_0_end_switches_pin[1]	Homing Sensor 2
E9	A19	bldc_ctrl_0_end_switches_pin[3]	Stop Sensor 1
C11	A21	bldc_ctrl_0_end_switches_pin[0]	Stop Sensor 2

Table 5: Connections to the hardware board [RD-2].

The prototype only used the two inner homing hall sensors, the system is ready to be connected to the two outer emergency-stop hall-sensors.

5.6 Possible Adaptations

5.6.1 Parameter Adaptations

The prototype runs at a clock frequency of 50 MHz. To change **clock frequency**, one must adapt:

- GC_CLOCK_FREQUENCY_HZ in the statemachine entity (statemachine.vhd line 46)

The prototype runs the PWM at a frequency of 200 kHz. The PWM frequency can be chosen freely but the timing constraints must be met. There must be enough clock cycles to complete all demanded calculations in one PWM cycle. To change **PWM frequency**, one must adapt:

- GC_PWM_TARGET_FREQUENCY_HZ in the statemachine entity (statemachine.vhd line 47)
- GC_PWM_PERIOD in the statemachine entity (statemachine.vhd line54)
- self.pwm_freq in the CoCa constructor (control.py line 18)
- PWM_FREQ in the simulation (simulation.py line 21)
- The look-up-table (LUT.json) needs to be recalculated for all movements.

The prototype uses a sine table of length 360, this could be reduced. As mentioned before, the resolution is usually determined by the sine table length – however, if one scaled down position change $\Delta\varphi$ is larger than 1, at least one step of the sine table is skipped and thus the resolution would be determined by the position counter. For the resolution to be determined by the sine table length the following condition must be fulfilled:

$$\frac{\Delta\varphi}{2^{22}} < 1$$

The maximum position change is equal to the maximum velocity $v = 5000$ rpm in DN:

$$v = 523.6\frac{\text{rad}}{\text{s}} \cdot \frac{360 \cdot 2^{22}}{2\pi\ \text{rad} \cdot 200000\ \text{Hz}}$$

Which is true for the parameters selected for the prototype

$$523.6 \cdot \frac{360}{2\pi \cdot 200000} = 0.15 < 1$$

We generalize this expression:

$$\text{sine\_table\_length} < \frac{2\pi \cdot \text{pwm\_freq}}{523.6} \approx 2400$$

This poses an upper limit for the sine table length. The lower limit is determined by the resolution of the system, which must be finer than 0.19° as stated in the requirements.

$$\frac{360°}{80 \cdot \text{sine\_table\_length}} \ll 0.19°$$

Thus, the lower limit becomes:

$$\text{sin\_table\_length} \gg \frac{360}{0.19 \cdot 80} \approx 24$$

It is recommended that the sine_table_length is much larger than the lower limit.

The sine table length must be dividable by three.

To change the **sine table length**, one must adapt:

- GC_SINE_TBL_LENGTH in the statemachine entity (statemachine.vhd line 52)
- self.sin_table_len in the CoCa constructor (control.py line 17)
- sin_table_len in the simulation (simulation.py line 22)
- The look-up-table (LUT.json) needs to be recalculated for all movements.

The prototype uses a scaling parameter of 22 Bits. This means that the calculated angle value will be divided by 2^{22} and then used to index the sine table. As can be seen in the equations above, the scaling parameter has no direct impact on the resolution – its purpose is to increase the number of possible discrete acceleration values. For the prototype configuration a lower limit of 18 has been determined, where there is only one possible acceleration value, thus creating a rectangular acceleration profile. A higher scaling parameter leads to a smoother motion profile. The scaling parameter cannot be larger than 31. To adjust this **scaling parameter**, adaptations need to be done in:

- GC_SCALE_INT in the statemachine entity (statemachine.vhd line 57)
- self.scale_int in the CoCa constructor (control.py line 19)
- norm_scale_int in the simulation (simulation.py line 23)
- The look-up-table (LUT.json) needs to be recalculated for all movements.

The timeout is set to 2 seconds. To change the **time-out** one needs to adapt:

- C_TIMEOUT in the statemachine entity (statemachine.vhd line 82)

5.6.2 Control Adaptations

It is planned that the parameter calculation is off-loaded, and all parameters given in Table 1 are sent over LVDS to the FPGA. Alternatively, the calculation can either be implemented on the FPGA, but note that $a_{\max}$ and $v_{\max}$ need to be a power of 2, such that the divisions (shown in Section 4) can be integrated as bit shifts. This would reduce the possibilities of $a_{\max}$ and $v_{\max}$. The adaptation for that on the simulation side can be done in:

- get_possible_params in the simulation (integer_params.py line 51)

Further adaptations in the VHDL code as well as in the C-code and Python-code for communication would be necessary.

parameter	*description*	*bit size*
target	Target angle	32
a_max	Maximum acceleration	32
v_max	Maximum velocity	32
power	Power during movement (0 to 255)	8
power_idle	Hold Power during IDLE (0 to 255)	8
accel_step	Acceleration steps	17

Table 6: Parameters required for a movement, given that the time parameters are calculated on-board.

For each of the 5 movements (35°, 70°, 105°, homing and blacklash) an updateable look-up-table containing the parameters from Table 6 can be stored on the FPGA itself, such that the DPM only needs to send the parameters shown in Table 7.

parameter	*description*	*bit size*
target_position	Target position	3?
direction	Direction bit	1

Table 7: Parameters required to be sent over LVDS to the FPGA for a nominal movement, given that the look up tables are stored on-board. It is not yet clear what the bit size of the target_position parameter would be.

The look-up tables should be updateable, such that we can later on change to a slow velocity and high power if, for example, a mechanical issue is detected during the mission.

5.7 Microblaze and UART interface

The prototype is controlled by the microblaze softcore processor which is running on the FPGA as a System-on-Chip (SoC). was created with the *Create or Import Peripheral...* wizard of the Xilinx Platform Studio with a PLB bus. This wizard automatically created the necessary directory structure as well as the top-level **statemachine** entity plus an example **user_logic** entity. Contrary to the NIM controller, it was not configured with an interrupt input, since the ADC housekeeping is not implemented. The C-code running on the microblaze is managing the UART connection to the python application. The microblaze is connected over the PLB to the XPS UART Lite IP block. Because the compiled C-code needs to fit into the onboard flash memory, we cannot use heavy libraries such as stdio. The **SerialIO** struct manages the communication and the **config_t** struct contains commands. It reads the commands over UART and write them into the corresponding FPGA registers. The commands start with a keyword and the different arguments are separated by spaces.

The configuration of the UART is

- **Baudrate**: 9600
- **Bytesize:** 8 bit
- **Partiy bit:** None
- **Stop bits:** 1
- **Flow control:** None

A softcore processor will not be implemented in the flight model, it is planned that the DPM will read from and write to the FPGA registers described in Table 2 using an LVDS.

6 Behavior in Idle State

To power the motor, three sine waves must be applied to the three phases. Since we are only working with positive voltage supplies, the sine waves are not centered around 0, but around *Vcc/2* = 6V. When the system is powered off, the voltage on the three phases is 0. The simplest approach is to ramp up the three phases with a timer until they reach their individual initial voltage. However, since the voltages are not separated by 120° degrees during this sequence, a current is flowing, and the motor might jump. This can be seen in Figure 11.

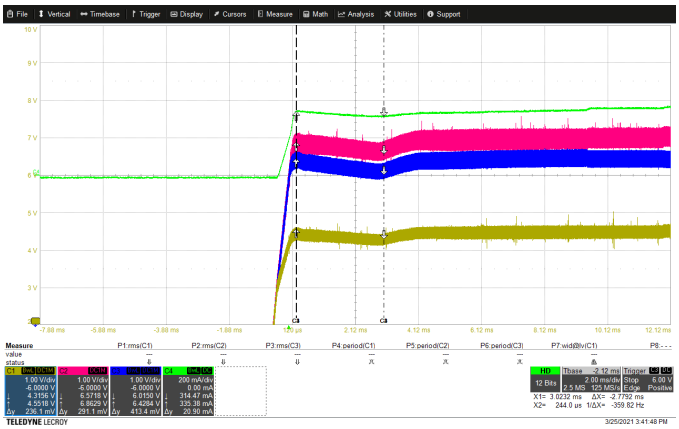


Figure 11: After ramping up the three phases, the motor jumps into position, which is visible in the voltage drop shortly after reaching the targeted voltage on all three phases.

Thus, we need to ramp up the three phases simultaneously to 6V and from there span up the three individual initial voltages. The same procedure will be applied after the movement to un-span and ramp down the voltages, this can be seen in Figure 12.

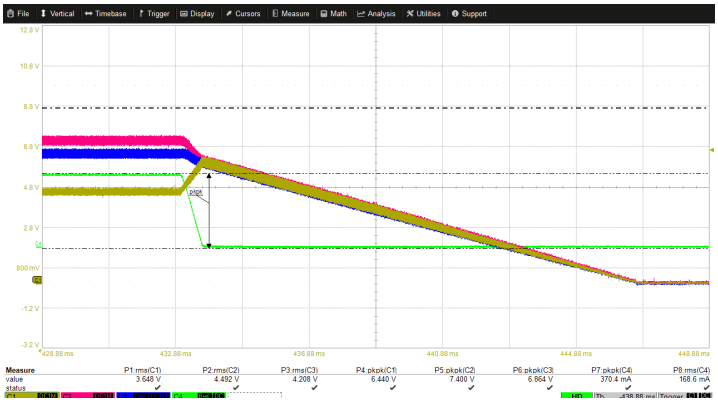


Figure 12: The three phases are un-spanned to 6V and from there on ramped down to 0V.

We noticed that when ramping down between movements, the motor experiences a slight jump when ramping up again, which can be seen as a slight voltage and current jump in Figure

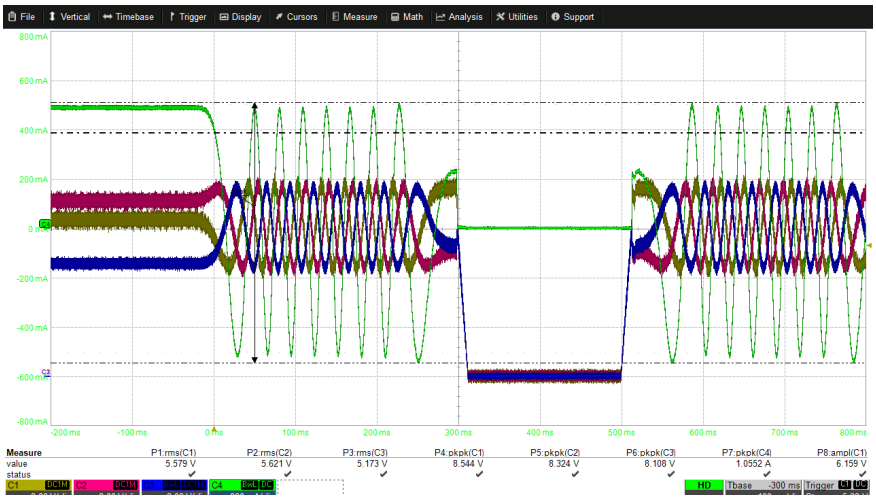


Figure 13: Two consecutive movements, with idle power set to 0. Thus, the voltages are ramped down in IDLE state. After ramping up, the motor will jump back into position which is visible as a short voltage drop.

13 shortly after the 500 ms mark. This results in a jump of 30-50 increments of the motor (max 0.06° on the Filter-Wheel).

These jumps have previously been believed to be due to the motor hitting the gears because of backlash [RD-1]. However, up on further inspection, it is suspected that as the motor looses torque when ramping down, the elasticity of the gears (planetary gearbox) pushes it back. When ramping up again, the motor jumps back into the previous position. It is thus recommended to not ramp down to 0 V but leaving the motor at an idle power, thus providing

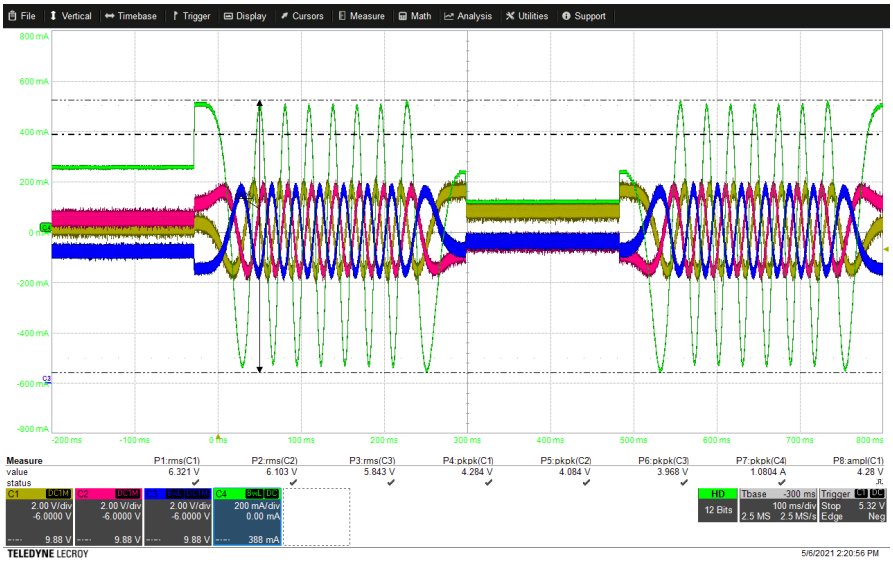


Figure 14: Two consecutive movements with the idle power value being set at 63 out of 255. A current of around 100mA is flowing during IDLE state.

holding torque, between movements. This method will require current to flow in idle state, thus using power and generating heat. It will however also prevent the filter wheel from being turned by external forces. The idle power can however be limited much further than shown in Figure 14. We have noticed that if all three phases are put to 6V in IDLE state, hence no current is flowing, the jump is not visible. Further tests have shown that this jump solely occurs when the voltage is ramped down faster than 35 ms as shown in Figure 15. The slow ramp up and down takes too much time to fit into the 0.5s window for a 105 degree movement including backlash compensation, we thus recommend to remain the system at idle power during movements. To ensure accuracy of the position, a homing manoeuver is recommended after ramping down and it is recommended to apply a holding torque (minimal current > 0 mA) in IDLE state between movements.

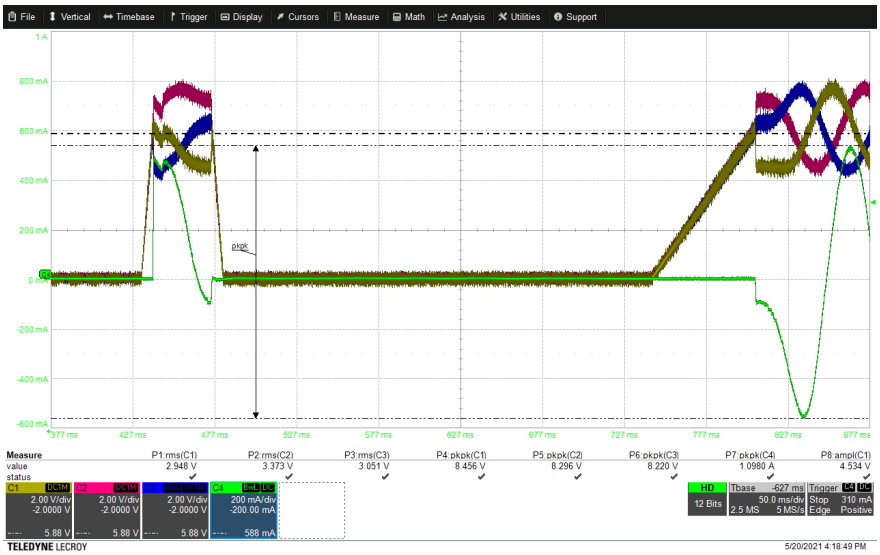


Figure 15: The jump in the three voltages is visible if the ramping up and down is faster than 35 ms (at the 440 ms mark). With a slower ramping, the jump is not visible (at the 800 ms mark).

All measurements were taken with a LeCroy HDO6104A-MS 1GHz bandwith, 4-Channels with 12-Bit-A/D Converter. The used passive voltage probes are: 1:10, 500MHz, 10MOhm and the current probe is: CP031A, 100MHz 30Arms.

7 Backlash Compensation

In the confidence test, the backlash of the prototype has been measured to be 0.98°. However, backlash of the actual flight model will likely be smaller. The software for the prototype cannot generate a 0.98° movement without generating arithmetic errors, thus a backlash compensation of 1° has been chosen. Other possible options are for example 0.8°, 0.9° 0.97° 0.99°.

Backlash needs to be compensated only when we change the direction of the rotation. Before a direction change, an additional movement of 1° is commanded to compensate the backlash. After this movement, all the gears should touch but the filter wheel should not have moved yet. The fastest configuration for a 1° movement will take 0.038 seconds. A 105° movement will take 0.422 seconds thus making it possible to move 105° and compensate the backlash afterwards within the required 0.5 seconds.

8 Control Software

The UART port of the FPGA can be connected to a computer through a Serial-to-USB adapter. The python script “control.py” will then establish a connection to the board and commands can be sent and received. A detailed documentation of the control software can be found in Appendix A: Software Documentation.

command	*parameters*	*return*	*description*
conf	hall_config_mask	“[OK] CONF”	Set hall config
home	power, power_idle, accel_step, direction, ramp, t1, t2, t3, t4, t5, t6, t7, t8	“[OK] HOME” “[OK] target reached!”	Commence homing
move	power, power_idle, accel_step, direction, ramp, t1, t2, t3, t4, t5, t6, t7, t8	“[OK] MOVE” “[OK] target reached!”	Commence movement
heat	power_idle, ramp	“[OK] HEATUP” “[OK] target reached!”	Ramps up the phases without turning the rotor (increases heating of the motor)
cool	power_idle, ramp	“[OK] COOLDOWN” “[OK] target reached!”	Ramps down the phases (decreases heating of the motor)
zero	-	“[OK] ZERO”	Set turns register to 0
read	-	“[OK] READ”	Passively readout registers

Table 8: Available commands to send over UART.

The parameters sent in the command structure in Table 8 are written into the registers as described in Table 2.

After all necessary commands have been received, the housekeeping data about the current state can be read out. The documentation is shown in Table 9.

message	*value*	*description*
STATUS	0 – IDLE, 1 – HOME, 2 – MOVE, 3 – EMG STOP LEFT, 4 – EMG STOP RIGHT, 5 - TIMEOUT	Status register
LIMITS	0 – No Sensor, 1 – EMG LEFT, 2 – HOME LEFT, 4 – HOME RIGHT, 8 – EMG RIGHT	Hall Sensor readout
ANGLE	0 to +1509949439	Angle of not fully completed turn
TURNS	-2147483648 to +2147483648	Number of completed turns since last 0

Table 9: Housekeeping (HK) data that is sent back over UART.

The filter wheel prototype is controlled over the UART port. The python script “control.py” will send the parameters shown in table Table 8. The parameters describe the velocity, acceleration, distance, and torque of a rotation. There are additional parameters that set the ramping up down (hold-torque) of the motor. The different possible accelerations and velocities for different distances can be determined using the python script “simulation.py”. The parameters **t_i**, **vmax**, **amax**, **xmax** (target position) need to be of integer type to mitigate rounding errors. This is discussed in section 4.2 and the functions are documented in Appendix A2: Predefined Simulation Functions.

9 Conclusion

The tests on the prototype have shown that all the requirements from section 2.1 have been met. A movement of 105° (filter 1 to filter 4 or vice versa) can be performed within 0.422 seconds with a maximum angular velocity of 4883 rpm and a maximum angular acceleration of +/- 5326 rad/s^2. Additionally, backlash can be compensated within 0.037 seconds with a maximum angular velocity of 651 rpm and a maximum angular acceleration of +/- 5326 rad/s^2. Both movements together can be performed within the required 0.5 seconds and do not exceed the velocity and acceleration limits of the motor. It follows that the smaller 35° and 70° movements can also be performed without violating the requirements.

The accuracy of the movement has been measured on the prototype using a MAP4000 potentiometer directly attached to the filter wheel. The potentiometer accuracy is not entirely symmetrical, as it appears that the resistance is not truly linear. This explains why the accuracy is larger on one side than on the other, and thus larger for movements that go across both sides such as 70° and 105°.

The measured accuracies are shown Table 10. They been extracted from Table 11, Table 12, Table 13 and Table 14. For 35°- and 105°-degree movements, we see a slight overshoot, but for 70° we are slightly lower than the target. As mentioned before, the potentiometer to measure the angle is not truly linear and might deviate for different angles. Additionally, from Table 12 we see that when we exclude the backlash compensation, we travel around 34.321° +/- 0.006° which indicates an estimated backlash of approximately 0.7°, this might explain the overshoot since we use a backlash compensation of 1° in the other measurements.

The resolution is determined by the length of the sine table as described in subsection 5.1 for a sine table length of 360, the resolution equals 0.0125° and is independent of the movement.

Movement	*Calculated resolution*	*Standard deviation*	*Backlash compensation*
35°	0.0125°	0.005°	Yes
35°	0.0125°	0.006°	No
70°	0.0125°	0.005°	Yes
105°	0.0125°	0.006°	Yes

Table 10: Measured accuracies of the prototype. The backlash compensation does not have an influence on the standard deviation here. If we do not include backlash compensation, the target position will be off by the backlash-distance.

10 Outlook

Further long-term tests need to be conducted to verify the functionality of the system. The prototype also needs to be tested with gears that are closer to the flight model. The hall sensor placement must also be adapted to the flight model layout for further testing. The code needs to be further modified to fit onto the Microsemi RT3PE600L chosen for the flight model. LVDS needs to be implemented to replace the Microblaze – UART interface. ADC readout should also be implemented to get HK data about motor temperature, phase voltages and currents.
There is no failure catch implemented in the FPGA. It will likely hang and require a reset if invalid t-parameters are sent. This problem should be addressed in the future.

Detailed possible adaptations are further explained in section 5.6

Appendix A: Software Documentation

To perform tests on the filter-wheel hardware, the python script “coca_test.py”[4] can be used. It imports the Controller class from the “control.py”[5] file, which will establish a connection to the FPGA using the UART serial protocol. The code is written and tested in Python 3.8 and uses the following dependencies: **pyserial**, **numpy** and **matplotlib**.

In the following subsections, the individual predefined functions to control the hardware and to perform simulations are documented.

An example of the initialization is shown in the following code section. At first the serial port is initialized using the pyserial library. After the Controller object is created, we set the hall configuration to 0b0110 (homing sensors enabled, emergency stop disabled). We then perform a homing maneuver followed by zeroing the system.

The Python Script will save a log file for each test in the “python/logs” folder. The filename is set to the date and time of the beginning of the test.

[4] https://spacegit.unibe.ch/cometinterceptor/coca/coca-motor-ctrl-test/-/blob/master/python/coca_test.py

[5] https://spacegit.unibe.ch/cometinterceptor/coca/coca-motor-ctrl-test/-/blob/master/python/control.py

```
if __name__ == "__main__":

    file = open("conf.json")
    conf = json.load(file)

    try:
        serial_port = serial.Serial(conf["serial_adapter"], 9600,
timeout=2)
    except serial.serialutil.SerialException as e:
        print(e)
        print("no serial connection found...")
        serial_port = None
    CoCa = Controller(serial_port, verbose=True)
    if serial_port is None:
        CoCa.logging.error("no serial adapter found! exiting...")
        exit()

    # mandatory! set hall mask at first
    #    0b1111 - all hall sensors are enabled
    #    0b0110 - only homing sensors are used, emergency stop disabled
    CoCa.set_hall(0b0110)

    # very important to home at first! choose a direction 1 or -1
    CoCa.home(direction=1, power_idle=63, ramp=55)

    # zero this home position!
    CoCa.zero()

    ########################################################
    ####  after this you are free to do what you want!  ####
    ########################################################
    CoCa.move(35, power=127, power_idle=0, ramp=55, backlash=1)
    CoCa.move(-35, power=127, power_idle=0, ramp=55)
```

A1: Predefined Control-Functions

This is the documentation of the functions defined in “control.py” and “coca_test.py”.

Constructor

To initialize the CoCa interface, the constructor needs to be called with a PySerial object. This returns the CoCa-object which can be used to send movements. It loads the configurations created by “simulation.py” from the “LUT.json” look up table.

parameter	*default value*	*description*
ser	-	PySerial object for the UART-connection
verbose	False	Print more information

set_hall

This function can be called to configure which hall sensors are used.

parameter	*default value*	*description*

hall_conf	0b0110	Hall-sensor mask

The mask is constructed as follows: the two outer bits are the emergency stop sensors, the two inner are the homing sensors. You should only choose between the following configurations: 0b0110 or 0b1111 (e.g. enable or disable the emergency stop sensors).

home

A simple homing command can be commanded by calling the “home” function. This function can be called with the following parameters:

parameter	*default value*	*description*
direction	1	Direction of the homing
power	127	Power value for movements (0 to 255)
power_idle	63	Power value in IDLE (0 to 255)
ramp	55	Counter for ramp up/ span time of the phases
backlash	1	Backlash to be compensated after the movement. Angle in [°]. If 0, no backlash compensation is executed.

Backlash compensation should always be executed since the next movement will be in the other direction.

zero

The zero function does not take any arguments and will set the reference value in the CoCa object on the python side as well as reset the turns register on the FPGA.

move

A simple movement command can be commanded by calling the “move” function. This function can be called with the following parameters:

parameter	*default value*	*description*
target	-	angle in [°]
power	127	Power value for movements (0 to 255)
power_idle	63	Power value in IDLE (0 to 255)
ramp	55	Counter for ramp up/ span time of the phases
backlash	0	Backlash to be compensated after the movement. Angle in [°]. If 0, no backlash compensation is executed.

Backlash compensation should only be executed if the next movement will be in the other direction.

single_move

A single movement command can be commanded by calling the “single_move” function. This function can be called with the following parameters:

parameter	*default value*	*description*
target	-	angle in [°]
direction	1	Direction of the movement
power	127	Power value for movements (0 to 255)
power_idle	63	Power value in IDLE (0 to 255)
ramp	55	Counter for ramp up/ span time of the phases

It only differs from the move function in that it will not do backlash compensation.

readout

The readout function does not take any arguments, but it will print out the housekeeping data such as current position, status etc. This will also be saved to the log file.

heat_up

To heat up the motor (span up the three phases) without initiating a movement, the heat_up function can be called.

parameter	*default value*	*description*
power	63	Power value for movements (0 to 255)
ramp	55	Counter for ramp up/ span time of the phases

cool_down

To cool down the motor (unspan up the three phases) without initiating a movement, the cool_down function can be called.

parameter	*default value*	*description*
power_idle	0	Power value for movements (0 to 255)
ramp	55	Counter for ramp up/ span time of the phases

power_test

This function is already implemented to do a simple power test.

parameter	*default value*	*description*
ctrl	-	CoCa object
angle	-	Distance of the movements
start_power	255	Power value for movements (0 to 255)
end_power	0	Power value in IDLE (0 to 255)
power_steps	10	
ramp	55	Counter for ramp up/ span time of the phases

It will do back and forth movements while decreasing the power on each movement. The goal is to see what the lowest power value is for which the motor is still able to rotate the filter wheel.

step_through_filters

This function is already implemented to do a simple demonstration of all filter positions.

parameter	*default value*	*description*
ctrl	-	CoCa object
n	10	Amount of total runs
power	127	Power value for movements (0 to 255)
power_idle	63	Power value in IDLE (0 to 255)
ramp	55	Counter for ramp up/ span time of the phases

This will do 35° steps from one end to the other n times.

A2: Predefined Simulation Functions

This is the documentation of the functions defined in “simulation.py” and “integer_params.py”.

get_config

This function returns and saves the config dictionary for a movement. One can override amax, vmax and accel_step manually or use the values that are selected with the index out of all possibilities. Internally *get_possible_params* is called and one possibility must be chosen. This function will create Figure 6 where all possible parameters for acceleration, velocity and acceleration step are displayed. One can either override these values upon calling *get_config* or the combinations can be chosen internally with the index.

parameter	*default value*	*description*
target_phi	-	Target distance
index	0	Index of the possibilities
update_file	False	Update LUT.json with this configuration
amax	None	Override amax
vmax	None	Override vmax
accel_step	None	Override accel_step

get_possible_params

This function returns all possible velocity, acceleration and accel_step parameters for the given distance.

parameter	*default value*	*description*
target_phi	35	Target distance
si_units	False	Returned values in SI units or Digital Numbers (DN)
time_scale	200e3	PWM frequency

run

This function performs a simulation with a given config-dictionary. The config dictionary can either be obtained by first calling *get_config* or by selecting one configuration from the *LUT.json. To show/save a plot, one must call plt.show() or plt.savefig(filename) afterwards.*

parameter	*default value*	*description*
conf	-	Config dictionary
target_phi	35	Distance of the movement, used for the title of the plot

Appendix B: Tests

Start angle [°] +/- 0.01°	End angle [°] +/- 0.01°	Distance moved [°] +/- 0.01°
0.00	35.05	35.05
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.05	35.04
35.05	0.01	35.04
0.01	35.04	35.03
35.04	0.01	35.03
0.01	35.04	35.03

Table 11: 35° Movements including backlash compensation resulted in an average distance moved of 35.037° +/- 0.005°.

Start angle [°] +/- 0.01°	End angle [°] +/- 0.01°	Distance moved [°] +/- 0.01°
0.00	35.11	35.11
35.11	0.82	34.29
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.10	34.28
35.10	0.82	34.28
0.82	35.09	34.27
35.09	0.82	34.27
0.82	35.09	34.27
35.09	0.82	34.27
0.82	35.09	34.27
35.09	0.82	34.27
0.82	35.09	34.27

Table 12: 35° Movements without backlash compensation resulted in an average distance moved of 34.321° +/- 0.006°.

Start angle [°] +/- 0.01°	End angle [°] +/- 0.01°	Distance moved [°] +/- 0.01°
0.00	69.95	69.95
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.00	69.95
0.00	69.95	69.95
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.00	69.95
0.00	69.95	69.95
69.95	0.00	69.95
0.00	69.95	69.95
69.95	0.01	69.94
0.01	69.95	69.94
69.95	0.00	69.95

Table 13: 70° Movements including backlash compensation resulted in an average distance moved of 69.944° +/- 0.005°.

Start angle [°] +/- 0.01°	End angle [°] +/- 0.01°	Distance moved [°] +/- 0.01°
0.00	105.20	105.20
105.20	0.01	105.19
0.01	105.21	105.20
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.20	105.20
105.20	0.00	105.20
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21
105.21	0.00	105.21
0.00	105.21	105.21

Table 14: 105° Movements including backlash compensation resulted in an average distance moved of 105.207° +/- 0.006°.

# movement	Target angle [°] +/- 0.01°
20	0.01
40	0.01
60	0.00
80	0.01
100	0.00
120	0.00
140	0.00
160	0.00
180	0.00
200	0.01
220	0.01
240	0.01
260	0.02
280	0.02
300	0.02
320	0.02
340	0.02
360	0.03
380	0.03
400	0.03
420	0.03
440	0.03
460	0.04
480	0.03
500	0.03

Table 15: Extended run test of 105° back and forth where the target distance is noted down every 20 runs. Since the calculated angle on the FPGA does stay constant after each pair of movement, the minor drift is assumed to be caused by uneven mounting of the structure or deviation of the potentiometer.

# movement	Target angle [°] +/- 0.01°	Encoder +/- 2 counts
50	0.01	35855
100	0.01	35855
150	0.02	35854
200	0.02	35854
250	0.02	35854
300	0.02	35853
350	0.02	35853
400	0.02	35854
450	0.02	35853
500	0.02	35854
550	0.02	35853
600	0.02	35854
650	0.02	35853
700	0.02	35855
750	0.02	35853
800	0.02	35854
850	0.02	35854
900	0.02	35853
950	0.03	35853
1000	0.03	35853

Table 16: Extended run test of 105° back and forth where the target distance is noted down every 50 runs. The calculated angle on the FPGA remained constant. Additionally, the encoder position on the motor was measured. The measured 0.03° returned to 0.01° after 20 minutes of no movement. It is assumed that the drift was a result of a rise in temperature of the potentiometer.

Appendix C: Timing Diagram

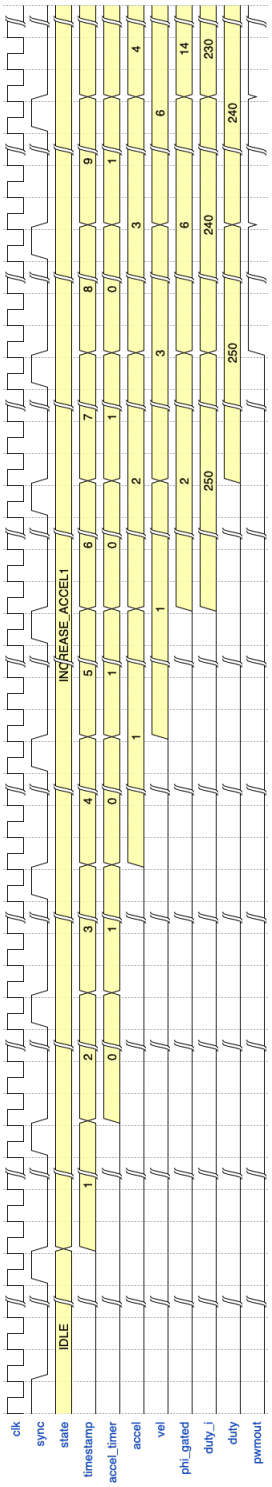


Figure 16: Timing diagram of the statemachine entity.

Appendix D: Working with Xilinx ISE 14.7

Xilinx ISE 14.7 was installed on Red Hat Linux 6 Enterprise Edition (RHEL 6) on a virtual machine. We tried installing it on Ubuntu and CentOS but parts of Xilinx ISE 14.7 were not working (XPS). The following commands are required to get ISE 14.7 to work after installation:

```
su
usermod -aG wheel *username*
usermod -aG root *username*
nano /etc/sudoers          # uncomment the wheel part
su *user*
sudo whoami

install fxload

cd Downloads
tar -xf Xilinx.tar
cd Xilinx
sudo ./xsetup
#select cable drivers!

#Try making this change to the file $TEMPNODE -> $tempnode:
/opt/Xilinx/14.7/ise/bin/lin64/xusbdfwu.rules

#install 32 bit glibc
wget http://rpmfind.net/linux/centos/7.9.2009/os/x86_64/Packages/glibc-
2.17-317.el7.i686.rpm
sudo rpm -ivh glibc-2.17-317.el7.i686.rpm --nodeps --replacefiles

#install zlib:
wget https://rpmfind.net/linux/centos/7.9.2009/os/x86_64/Packages/zlib-
1.2.7-18.el7.i686.rpm

sudo rpm -ivh zlib-1.2.7-18.el7.i686.rpm
```

If place&route exits with the following error:

"Pl_Uap_Flow1FitterRuleFastFeedbacks: bad index to sec_nodes array":

Go to "Desing Goals & Strategies" and select the option "Timing Performance".

To launch SDK in Xilinx ISE 14.7 the simple command “xsdk” may result in an error due to a java-issue, to mitigate this, one needs to enter the following command in the TCL console:

```
xsdk -vmargs -Dorg.eclipse.swt.internal.gtk.cairoGraphics="false"
```

When connecting a USB cable after a reboot, one might have to set chmod -x 777 for the usb driver file, otherwise connection to the USB driver will fail.

Declaration of consent

on the basis of Article 30 of the RSL Phil.-nat. 18

Name/First Name:	Linus Leo Stöckli
Registration Number:	15-706-518
Study program:	Physics

Bachelor ☐ Master ☑ Dissertation ☐

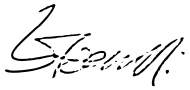
Title of the thesis:	Investigations into the Filter-Wheel of the CoCa Experiment on the Comet Interceptor Mission
Supervisor:	Prof. Dr. Nicolas Thomas

I declare herewith that this thesis is my own work and that I have not used any sources other than those stated. I have indicated the adoption of quotations as well as thoughts taken from other authors as such in the thesis. I am aware that the Senate pursuant to Article 36 paragraph 1 litera r of the University Act of 5 September, 1996 is authorized to revoke the title awarded on the basis of this thesis.

For the purposes of evaluation and verification of compliance with the declaration of originality and the regulations governing plagiarism, I hereby grant the University of Bern the right to process my personal data and to perform the acts of use this requires, in particular, to reproduce the written thesis and to store it permanently in a database, and to use said database, or to make said database available, to enable comparison with future theses submitted by others.

Zürich, 30.9.2021

Place/Date



Signature