Remote Positioning of a Multidirectional Camera

Linus Wannebro

Victor Ohlson

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6126
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2021

Abstract

In today's world of uncertainties, video surveillance is widely used to make the people of our society feel safe and protected. As more cameras are installed, the time of installation and ease of use play a critical role when customers decide which camera solution to select.

One way to limit the number of cameras needed to cover multiple directions and large areas is to install a multidirectional camera. A multidirectional camera is a complete camera solution where multiple camera heads are incorporated in the same chassis and the most common type of multidirectional camera has only one IP-address which further favors usability. Additionally, to limit the time of installation and allow for adjustments during the camera's lifetime it is beneficial to incorporate a remote positioning functionality to the camera.

The objective of this thesis is to develop and produce a working prototype of a multidirectional camera, which consists of multiple camera modules, each with the capabilities of movement in four directions, pan, twist, tilt, and rotate which all can be controlled remotely. This was made possible by making an entirely new mechanical platform, PCB, and software for controlling the camera heads individually.

The concept development phase was dominated by investigating which type of transmissions and actuators were best suited to fulfill the needs requested from the company. After deciding which concepts to further develop, these concepts were realized and produced through 3D printing. The prototype was then tested in order to verify that the needs were satisfied.

Acknowledgements

We would like to express our gratitude to everyone who has contributed in every way to this master's thesis.

To our academic supervisor Anders Robertsson, thank you for believing in us and for the continuous support during the entirety of this thesis. Without you the result we have performed would not have been the same.

To our academic examiner Karl-Erik Årzén, thank you for stepping in when you were needed the most. Without your help, the report would not have been finished in time.

To our industrial supervisors Malin and Rasmus, thank you for guiding us through every step in the development of the prototype and for sharing your knowledge with us. This thesis would not have been possible to finish without you.

To our many colleagues at the mechanical department of MCP, thank you for your input, especially during the brainstorming session of the concept development. If it weren't for you, our concepts would have been half as innovative and many times duller.

To our many colleagues at the electrical department of MCP, thank you for your incredible perseverance to continuously helping us mechanical engineers with the simplest of electrical problems and for keeping us from getting electrocuted. Without you we probably would have died.

To our manager at MCP Magnus, thank you for making this master thesis possible. Without your proposition of this thesis, we would not have regarded the master thesis as the most enjoyable part of our education. Thanks to your generosity, we were able to do everything to envision our goal of the thesis.

To our product manager Olof, thank you for your continuous support and direct feedback on our solution. Also, thank you for believing in us even though many of our demos flopped. Without you, the prototype would not have even been close to completed.

To our families and friends, thank you for supporting us throughout the entirety of our education. Sometimes it is easy to think that "*it is never too late to give up"* so thank you for that you will "*never gonna give us up*".

Table of contents

1. Introduction

This chapter will give a brief introduction to the background, goals, and incitement of this master's thesis as well as the methods used during the project. Due to secrecy, the corporation, which this master thesis has been conducted in collaboration with, will not be named, only referred to as "the company". Some illustrations and component names have been altered in order to not reveal any sensitive information or subcontractors of the company.

1.1 Background

Multidirectional cameras are a popular product within the surveillance camera segment. The definition of a multidirectional camera is a camera which houses two or more camera sensors incorporated in one camera housing, see Figure 1 for an example. This gives the user the possibility to monitor multiple directions simultaneously with one camera, since the individual camera sensors can be positioned independently of each other. This provides up to 360° coverage through separate video streams while also letting the user adjust both the viewing angle and level of zoom for each camera sensor. This makes multidirectional cameras flexible since the user does not need to compromise what to capture.
The camera can be set up to capture both overview images and images with more detail at the same time. One common area of use, where multidirectional cameras can be found advantageous compared to traditional single sensor cameras, are four-way intersections in corridors, see Figure 2. In places like this one multidirectional camera can be utilized to replace the need of several traditional cameras.

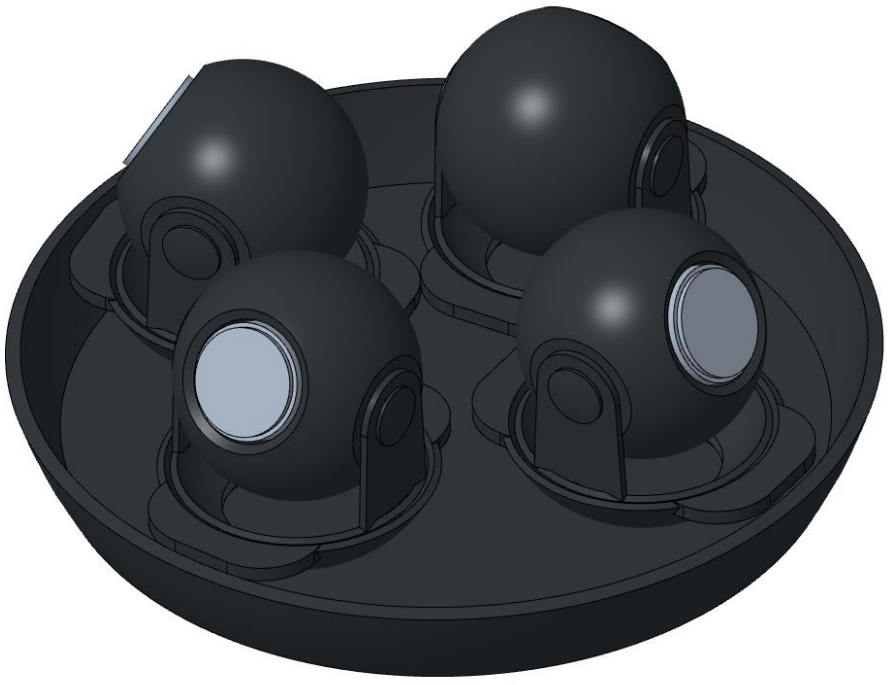


Figure 1: Illustration of a general multidirectional camera with four camera heads.

Figure 2: Illustration of the field-of-view of a multidirectional camera. The blue indicates coverage provided by just one multidirectional camera in one of many use cases.

Today, most multidirectional cameras are designed to be adjusted manually when the camera is installed. The viewing angles of the sensor modules are locked in place when the camera is mounted. Any corrections needed after that point will require someone to physically re-do the installation procedure, which can be rather difficult, since most of the cameras are deliberately mounted in places hard to reach, for safety reasons. This can lead to expensive and time-consuming installations and re-adjustments of the camera sensors.

Because of above mentioned reasons, ease of installation and redirection are important factors when developing a multidirectional camera. One potential way to limit the installation time and allow for adjustments during the camera's lifetime is to make it possible to adjust the viewing direction of the camera sensors remotely.

1.2 Mission Statement

The objective of this master thesis is to design a working prototype of a multidirectional camera with the capabilities of remotely controlling the movement pan, twist, tilt and rotate, all specified further in the theory section. It is not necessary for the movements to be done more than a couple of times since the remote positioning is only for installation purposes and not for continuous use. This was done using methods of product development, with an initial concept generation and selection and then further iterative concept development. The concept was then realized using CAD-software and 3D printing. Much emphasis was placed on the choice of actuators and the transmission of torque from the actuators to the movement of the camera module.

1.3 Assumptions and delimitations

The problem description of this project is broad, and some assumptions and delimitations have therefore been made to be able to arrive at a final concept within the limited time of this Master Thesis. Due to this, new experimental actuation methods will not be investigated in depth during the concept generation. Focus will be on developing a working prototype. As a result, little regard will be taken to the manufacturability or cost during the initial development phase of this master's thesis. It will however be addressed in the discussion.

To make the final prototype as close to a real product as possible, it was decided that it would be designed around a camera sensor currently used at the company. This would allow an image feed to be incorporated into the prototype using an existing PCB and provides some base dimensions to relate to when developing the mechanical design. The concept was to be as small as possible, while still being robust enough to handle normal use conditions.

The general functionality of the prototype was to be the same as current multidirectional cameras on the market. This means a camera body, with four individual cameras, that all have the movements pan, twist, tilt and rotate. More specific metrics will be presented in section 3.2 Metrics. All the specifications for this project was derived in collaboration with stakeholders at the company.

1.4 Individual Contributions

There has not been a distinct division of work in this thesis; the main part of this thesis as well as the report have been done in joint effort by Linus Wannebro and Victor Ohlson. Linus Wannebro has been in charge of the overall mechanical design and Victor Ohlson had the main responsibility of the electrical and software design.

1.5 The Product Development Process

The development process in this thesis will follow the general product development process described in Ulrich & Eppinger Product Design and Development. The process consists of six phases, as shown in Figure 3, which begins in the planning phase and generally ends with a product launch, at which time the product becomes available for purchase in the marketplace. Each phase can be described as follows:

Phase 0: Planning
The planning phase is often referred to as "phase zero" as it precedes the project approval and the actual start of the product development process. It starts with identification of opportunities of the project which later results in the project mission statement, which specifies the business goals, the target market, the key assumptions, and constraints. (Ulrich & Eppinger, 2016)

Phase 1: Concept Development
In the first phase of the actual product development process, the needs of the target market are identified, alternative product concepts are generated and evaluated, which implies that one or more concepts are selected for further development and testing. A concept is a description of the form, function and the

features of a product and is usually paired with certain specifications, an analysis of competitive products, and an economic justification of the project. (Ulrich & Eppinger, 2016)

Phase 2: System-Level Design

This design phase includes the definition of the product architecture, decomposition of the product into subsystems and components, and preliminary design of key components. Later parts of the product development process such as plans for the production systems and final assembly are usually defined here as well. The output of this phase is often a geometric representation, or at least a layout, of the product, a functional specification of each of the product's subsystems, and a preliminary process flow diagram for the final assembly process. (Ulrich & Eppinger, 2016)

Phase 3: Detail Design

In the detail design phase, all of the specifications, material choices and tolerances are set. Identification of standard parts which are to be bought from suppliers is also done here. The plan for the process, which includes tooling etc., is designed for each part within the production system. The goal of the process is to complete the control documentation for the project - the drawings or computer files describing each part of the product in order for it to be manufacturable as well as a list of specifications for the purchased parts. The key issues of this phase are material selection, production cost, and the performance of the product. (Ulrich & Eppinger, 2016)

Phase 4: Testing and Refinement

The testing and refinement phase includes the construction and testing of multiple pre-production versions of the product. Prototypes are also constructed, called Alpha prototypes, which have the same geometry and tolerances as the product intended for extensive production. These prototypes are built in order to verify that the product works as intended and will satisfy the customer needs. Beta prototypes are then built using the same production methods as the final product but are not assembled with the intended methods. The principle of these prototypes is to show the reliability and performance of the final product. (Ulrich & Eppinger, 2016)

Phase 5: Production Ramp-Up

In the final phase of the product development process, the product is made using the intended production system. This is done to prepare the workforce and to work out any of the remaining problems in the production process. The products produced in this phase are sometimes delivered to customers and are evaluated to find any remaining defects. The transition between production ramp-up phase and the continuous production phase is usually gradual. It is during this stage of the product development process that the product is launched which is usually followed by a project review. This review includes the assessment of the project both from a commercial point of view as well as technical to identify ways of improvement of the development process for future projects. (Ulrich & Eppinger, 2016)

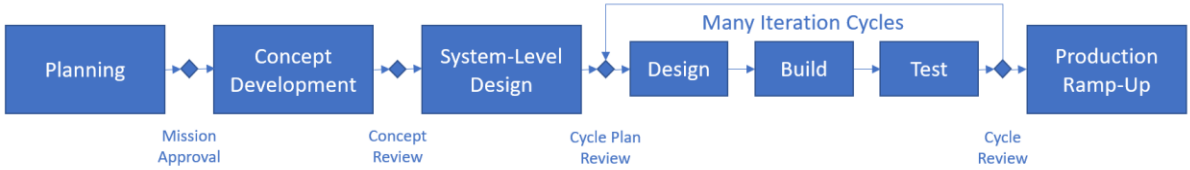


Figure 3: Ulrich & Eppinger's product development process. Reproduced from: (Ulrich & Eppinger, 2016)

As the goals of this project is to develop and build a working prototype, the project will only cover up to the third phase, Detail Design, with some aspect of Phase 4. Since the goal of the thesis is to produce a working prototype through 3D printing, most of the benchmarking tests will not be applicable to the prototype due to rigidity reasons. The overall methodology described above has been used as a tool but is modified to fit the project.

2. Theory

This chapter will give an introduction to the many fields this project touches on. Since this project includes everything from mechanical transmissions to electrical components and computer communication, it is not assumed that the reader has prior knowledge in all of these fields. The purpose here is to give some background information to the decisions that are made later in the development process.

2.1 Camera movements

In order to get clarity in the movements of the camera often mentioned in this report, this section aims to illustrate the movements in respect to the camera axes. This will be done using a general multidirectional camera with four camera heads, which will be the starting point of the project.

Pan

The pan movement is the movement where each camera module is rotated around the center axis of the entire multidirectional camera which allows for 360-degree movement around the center axis, see Figure 4.

Figure 4: Illustration of the pan movement on the multidirectional camera.

Twist

The twist movement is the same movement as the pan, but the rotation is done around the center axis of the camera module instead of the entire camera. This movement is done in order to be able to set focus on the same subject with two different camera modules, see Figure 5

Figure 5: Illustration of the twist movement on the multidirectional camera module.

Tilt

The tilt movement is done by rotating the camera module around the green axis shown in Figure 6.

Figure 6: Illustration of the tilt movement on the multidirectional camera module.

Rotate

The rotate movement is done by rotating the camera module around the axis shown in Figure 7. The feature marked in grey will then rotate independently from the surrounding black feature. This movement has many areas of use. The first is to correct for any horizontal misalignments when the camera is first mounted. The second is to be able to change orientation of the camera frame from landscape to portrait.

Figure 7: Illustration of the rotate movement on the multidirectional camera module.

2.2 Construction in plastics

Since most parts produced in this project will be constructed in plastics this section will show how components should be designed in order to be strong enough to withstand loads. Also, different manufacturing methods for the prototype will be presented. For assembly of components in plastics, snap fits are commonly used. This method of fastening will also be thoroughly explained.

2.2.1 Injection molding

The injection molding process is a high speed, automated process that is used to produce plastic parts and is suitable if the part has complex geometries. The process is versatile since it can produce either small or large components using virtually any plastic material. Though, it is of utmost importance that the design of the component that will be produced has the right geometric features in order to be manufacturable as well as tooling requirements and cost. (Malloy, 2010)

In addition, the part designer must consider that the properties of the molded plastic part will be greatly affected by factors such as the tool design and processing conditions. In order to develop a properly designed component, the plastic part designer, the mold designer, the material supplier, and the process engineer must all work together to make the component both moldable and fully functional. (Malloy, 2010)

The actual manufacturing of the injection molding process is as follows. Thermoset or thermoplastic material in granular form is fed into a heating barrel through a hopper. In order to get the plastic in a manageable state the plastic is heated to a predetermined temperature and is driven by a screw through a gate and into the mold. As the mold is fitted, the screw will stay in place in order to apply appropriate pressure for the duration of the cooling time. When the plastic has set, the screw is withdrawn, the mold is opened, and the part is ejected. (Creative Mechanisms, 2017)

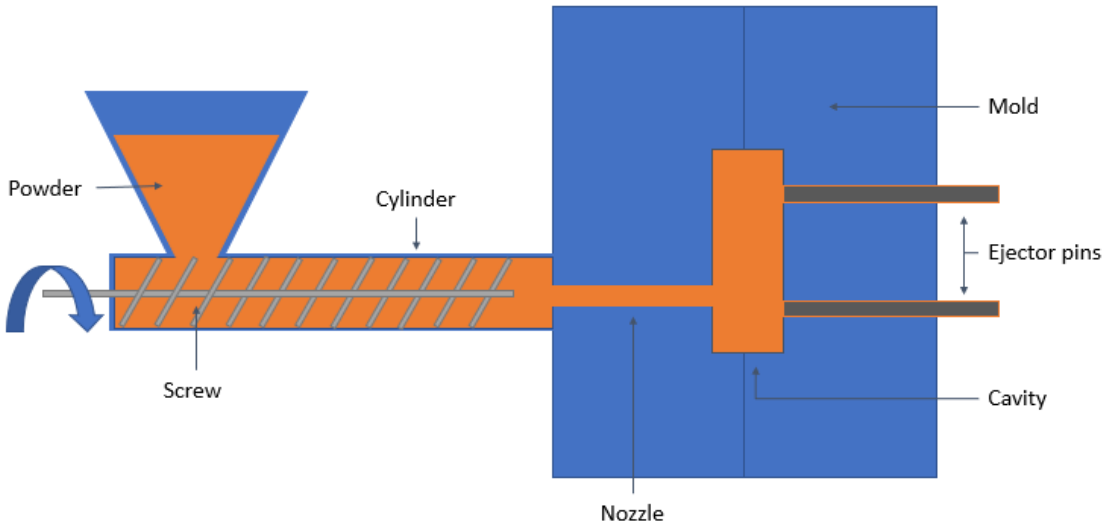


Figure 8: Illustration of a typical injection molding process. Reproduced from: (Creative Mechanisms, 2017)

The following design principles must be ensued in order to make a plastic component that is manufactured by injection molding.

Draft angles

Since the injection mold process requires for the two parts of the mold to be separated, draft angles or tapers of the part that is to be constructed are required. When parts are produced the cavity is stationary while the core is attached to the moving plate. Usually, the plastic part tends to stay with the core when the molds are separated because of the shrinkage which causes friction forces between the plastic component and the plates. (DSM, n.d.)

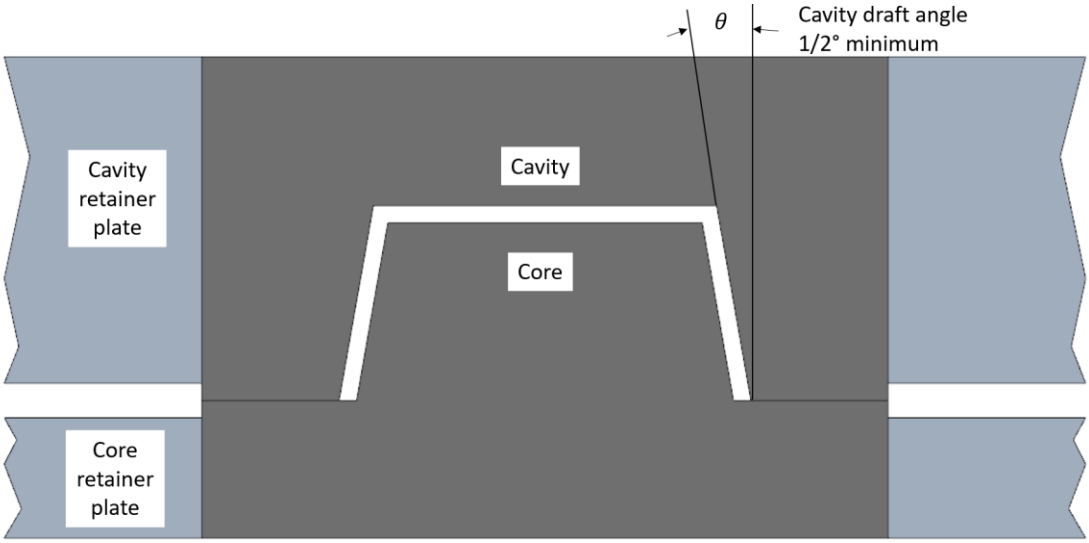


Figure 9: Cavity draft angles are used to facilitate release from the cavity and the tool opens while core draft angles are added to facilitate release from the core during ejection. Reproduced from: (Malloy, 2010, p. 90)

Radii and chamfers

When designing injection molded parts, sharp edges and corners should be avoided. Sharp edges can lead to high molded-in stresses, poor flow characteristics, reduced mechanical properties, increased tool wear and surface appearance problems. (DSM, n.d.)

By applying radii and chamfers, many advantages follow such as less warpage, less flow resistance and easier filling, reduced stress concentration, less notch sensitivity, more uniform cooling, and better appearance. (DSM, n.d.)

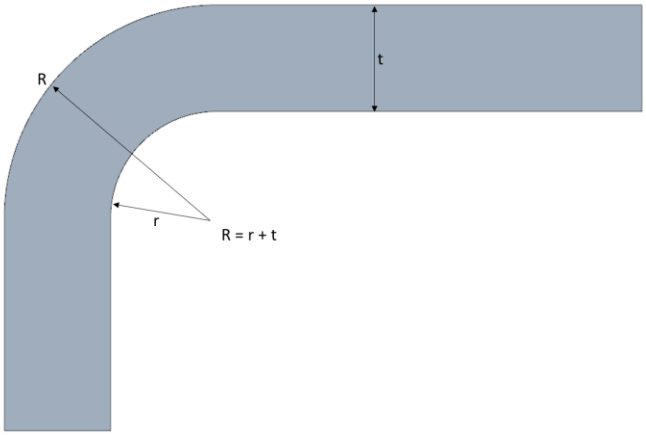


Figure 10: Illustration of corner radius. Reproduced from: (DSM, n.d.)

Undercuts

Ideally, the plastic part that is constructed by injection molding should be designed in order to be ejected from the mold with any special actions such as, angle pins, collapsible cores, and unscrewing mechanisms. The special mold actions can make the part more expensive since every action requires intervention in the mold. (Malloy, 2010)

For an example, it can be seen in Figure 11 that in order to make the hole in the back wall, some sort of side action is required since the mold movement is in the z direction. If the part was designed as the features next to the hole, there is no need for any side action which will reduce the cost of the component. It can also be seen in Figure 11 that the undercut associated with the snap beam can be resolved without any side action in form of a lifter by forming it with a shut-off. (Malloy, 2010)

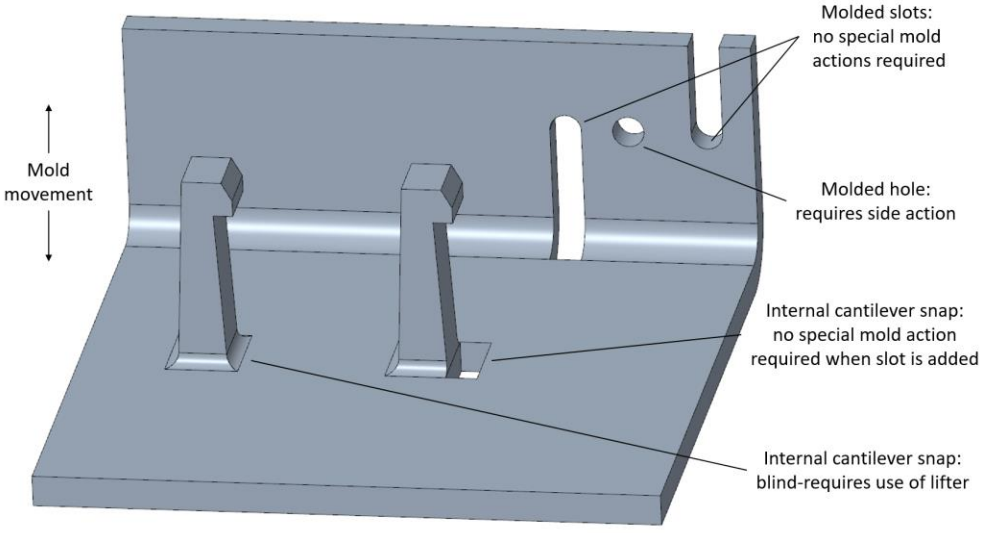


Figure 11: Illustration of multiple features that do or do not require special actions during ejection form the mold. Reproduced from: (Malloy, 2010, p. 101)

Wall thickness

The injection molding process is a sequential, semi-continuous process with several phases as the material is filling the mold. The packing phase begins once the material has floated into the extremities of the cavity. (Brockotter, n.d.)

Parts with large wall thickness are the hardest to cool and pack since thicker sections require more time to cool and higher pressure to pack. If there is a mixture of thicker and thinner sections, there is a need for gating into the thicker sections since it enables for separate packing of the different sections. The different cooling and packing requirements of different thickness sections lead to shrinkage-related internal stresses which is undesirable. Therefore, it is important to design plastic parts with constant wall thickness for constructability reasons. (Brockotter, n.d.)

2.2.2 3D printing

To verify the construction and functions of the different concepts in the iterative phase of the master thesis as well as the final concept, 3D printing is the most suitable option of verifying the design and functionalities of the prototype. The overall design philosophy is the same for 3D printing as for injection molding but since the components that are to be produced through printing generally are more fragile and do not have the same characteristics as parts made through injection molding, some modifications have to be done to the components that are produced during this thesis.

Design Considerations

The 3D printing process builds parts by applying layers upon layers of material. Every layer must be deposited onto another layer which means that every layer must be printed over some underline material.

Overhangs are parts of the model which are either partially supported by the layer below or not supported at all. There is a limit on the overhang angle every printer can produce without the need of support material, which is material that is only used during the printing of the model and later removed after the entire model is fully constructed. In most types of 3D printers, such as Fused Deposition Molding (FDM) and Stereolithography (SLA), this angle is approximately 45 degrees (Brockotter, n.d.). Support material can be used as the component is constructed but it is good practice to avoid since usually the surface of which support material is constructed upon has a rougher finish.

Wall Thickness is another important consideration to have in mind when designing a 3D printed component. Every 3D printing process can produce features accurately that are thin up to a certain point. It is usually necessary to add thickness to the components, especially when printing a scale model of a component. Wall thickness of at least 0.8 mm can be printed successfully with all 3D printing processes. (Brockotter, n.d.)

When the material is processed in the printer, it is first heated up then solidified which means that the material undergoes physical change. The heating and cooling process of the material may cause the material to warp during the process. (Brockotter, n.d.)

Some surfaces are more prone to warping than others. Large, flat surfaces are more easily warped as the material settles. Warping can be avoided by using correct machine calibration and having adequate adhesion between the component and the print bed. Usually, the most secure way of avoiding warping is to design parts without large flat surfaces and by adding rounded corners to the component. (Brockotter, n.d.)

Materials

Polylactic acid (PLA) is commonly used as material in 3D printers. The material is rigid, has good layer bonding, and is ideal when printing larger components since the material has little tendency to warp or deform during printing. (Horne & Hausman, n.d.)

However, the material has drawbacks. PLA is not as impact resistant as some other plastic materials which may cause cracks if subjected to excessive loads or knocks. Also, because of the materials' low-temperature glass transition point, it can creep slightly after it has been produced. (Horne & Hausman, n.d.)

Acrylonitrile butadiene styrene (ABS), is also a common choice for functional parts. Though, because of the specific requirements of a tempered printing bed, the list of printers that can print in this material is short and is therefore not used as often. (Horne & Hausman, n.d.)

Layer Height

As stated before, the component that is 3D printed consists of multiple layers of printed material and it is of importance that the height of each layer is correct. The layer height determines the appearance of the finished object and has correlations to the components ability to withstand forces and impacts. The relationship of each layers' height that can be used with a certain nozzle size is directly dependent on the width of the printed plastic line and the bonding strength of each layer. (Horne & Hausman, n.d.)

The time of the printing process is also in direct proportion to the layer height since the nozzle travels at constant speeds. If there is need for an impact resistant part and there is a shortage of time of producing, a layer height of 50 percent of the nozzle size contributes to a strong part that is quick to print. It is to be said that the finish of the produced part is worsened by an increase in layer height. (Horne & Hausman, n.d.)

Infill

To limit the time of the printing process, it is most common that the component that is to be printed has an infill grade of less than one, meaning that the object is constructed by a honeycomb-like structure. Infill level is the relative amount of material within a structure which is not air. For example, even though the perimeter of an object is 0.5 mm and the infill level are 25 percent, the object is still strong. If the component is decorative, infill levels can be as low as five or even zero percent. (Horne & Hausman, n.d.)

2.2.3 Snap fits

The principle of a snap fit is to push a projection on one part past an obstruction on a mating part. This is made possible by the elasticity of plastics. Generally, one part is rigid while the other is flexible. Depending on the design, the snap can either be permanent or releasable. (Tres, 2017)

The most common type of snap fits is the cantilever snap fit, which is also known as snap hook, catch spring, spur, or lug, and is shown in Figure 12. The joint is assembled by applying a force to bring the parts together. When the parts meet, joining is opposed by an interference between the parts. As the applied force is sufficiently large, an elastic deflection that is equal to the interference allows for a fit between the parts. The design of the snap-fit should be that the plastic deflection is fully or partially released once the snap has passed the interference points have past and the parts have come together. (Tres, 2017)

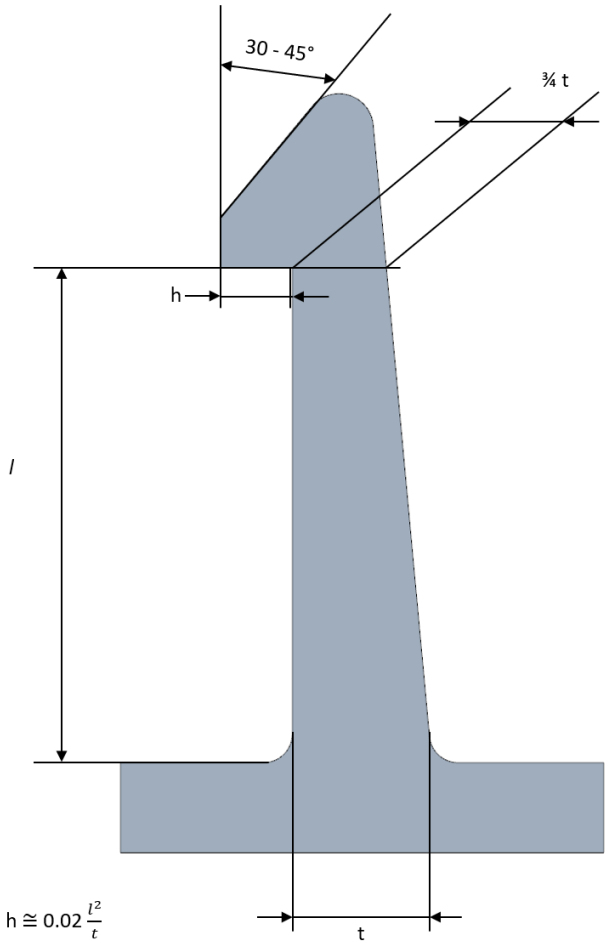


Figure 12: Simple illustration of a cantilever snap with relative dimensions of features in relation to the thicknesses. Reproduced from: (Haddad, n.d.)

The snap-fit can be described as an elegant way of fitting two parts together since it is efficient, virtually free of charge since it does not require any other fastening components, and 'green'. The fitting mechanism can be designed as an integral part of the component without causing any increase in cycling

time at the expense of a small amount of increase in mold cost and maintenance. Another benefit is that the assembly process can be done without any use of tooling and components such as screws, clips, or adhesives. (Tres, 2017)

The design of the snap is crucial to maintain a good fit as well as making sure that the two components are fastened without breaking the snap. Figure 12 shows the relationship between the length of the snap head, the length of the shaft and the protrusion of the head itself. It is also important to make the head of the snap angled for gradual load of the snap joint. (Haddad, n.d.)

2.3 Transmissions

In order to transfer the mechanical energy from the output axis of an actuator to actual movement, some sort of torque transferring method must be used. The meaning of a transmission is something that transmits movement between different places within a construction (Vedmar, 2012). Since the torque needed to move the camera in every direction is relatively large, especially when considering the limited space for the motors available within the structure, some sort of gear setup must be used. This section aims to explain the principle of different torque transferring methods and gears as well as the pros and cons of said methods.

2.3.1 Cogwheels

A transmission usually consists of two wheels which are somewhat connected to each other. If there is a need for different speeds of the wheels in order to get more torque, the two wheels must have different sizes. (Vedmar, 2012)

When the need for torque is greater than the friction force between two wheels, a gear transmission is used. This means that the wheels are fitted with cogs where the cogs have contact with each other at any given time. Due to the cogs, movement and force will be transmitted without slippage between the wheels. This does not mean that there is no slip in a transmission with cogwheels, as this would mean that the transmission is in practice loss free in its transferal of energy, which is impossible. (Vedmar, 2012)

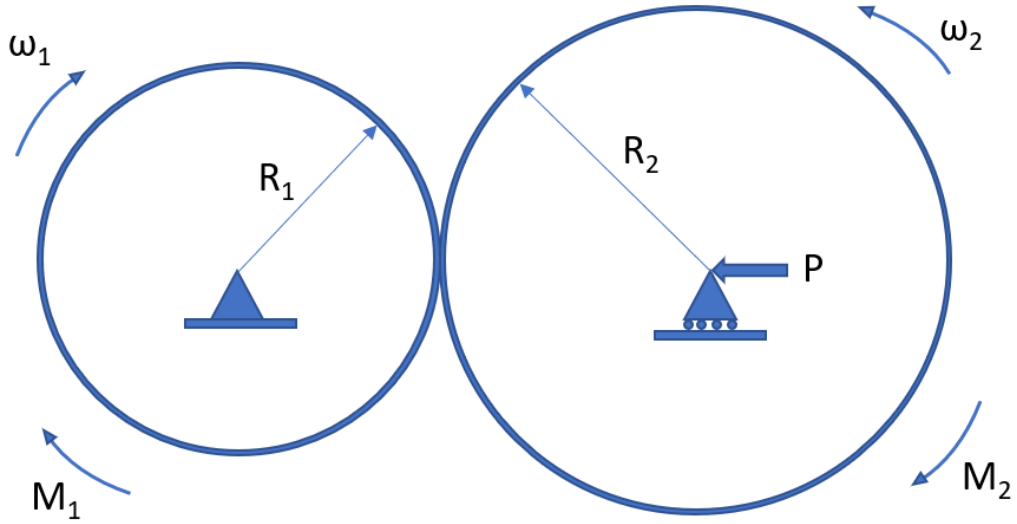


Figure 13: Simple illustration of a general torque transfer transmission, wheel 1 to the left, wheel 2 to the right. Reproduced from: (Vedmar, 2012)

In order to get a proper understanding of why a transmission with cogs is preferable, a comparison with a transmission solely based on friction will be done. In Figure 13 above, two cylindrical wheels, each held

in place in the center of the wheel, are pushed against each other by the force P. As one of the wheels are driven with the angular velocity ω_1, the other wheel will rotate with the angular velocity of ω_2 in the other direction. In order to drive the first wheel, a motor must be fitted which outputs the torque M_1. In this application the other wheel is loaded with the torque M_2. If there is no slippage between the wheels, the peripheral speed at the contact point between the wheels will be $\omega_1 R_1 = \omega_2 R_2$ which gives the speed ratio:

$$U = \frac{\omega_1}{\omega_2} = \frac{R_2}{R_1} \tag{1}$$

Speed ratio, or later gear ratio, is the characteristic feature of a transmission. This is the feature which allows for the conversion of movement and forces between different parts. The drawback of the transmission in Figure 13 above is that there is high risk of slippage between the wheels. The friction force in the contact point is $F = \mu P$, where μ is the coefficient of friction in the contact point between the wheels, which gives:

$$\mu = \frac{M_2}{P\,R_2} \tag{2}$$

If the wheel that is to be driven is loaded with a torque M_2 that is greater than the available friction force, there will be slippage between the wheels which means that the speed ratio will no longer be constant which is undesirable. To counteract this, the applied force P could be increased but this might cause damage to the wheels themselves. Another way is to change the material of the wheels so that the friction coefficient is greater which as seen in Formula 2 increases the torque M_2. (Vedmar, 2012)

Spur gear

In order to solve the risk of slippage, the wheels can be combined with cogs as seen in Figure 14 below which shows two cogwheels where cogwheel 1 has $z_1 = 12$ cogs and cogwheel 2 has $z_2 = 18$ cogs. As the wheels are loaded, the contact force between the wheels will push the wheels apart which means that the wheels will have to be mounted firmly at a fixed distance. Compared to the previous transmission, there is no need for an external force P. Each wheel has the same number of cogs as gaps and when one wheel is turned one wheel's cog will push into the other's gap. (Vedmar, 2012)

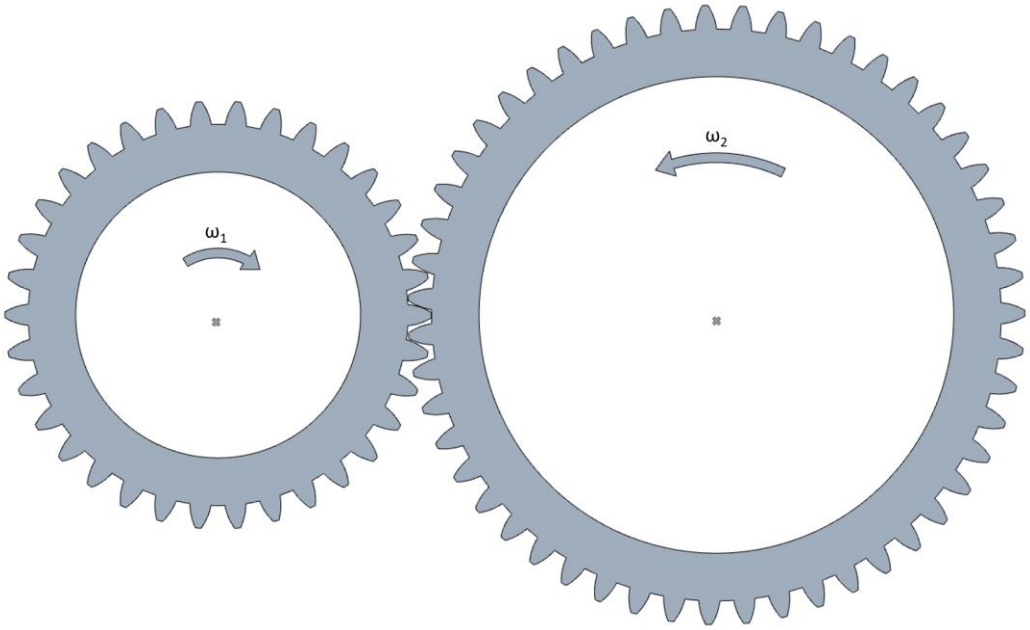


Figure 14: Simple illustration of an external spur gear transmission, wheel 1 to the left, wheel 2 on the right. Reproduced from: (Vedmar, 2012)

Since the two wheels do not have the same number of cogs, as wheel 1 has turned one lap the other will have been turned $\frac{z_1}{z_2}$ laps. The gear ratio between the wheels will therefore be:

$$U = \frac{\omega_1}{\omega_2} = \frac{z_2}{z_1} \tag{3}$$

Internal gear

An internal gear consists of two cogwheels where the larger has the shape of a ring and the smaller is an ordinary cogwheel. If a comparison can be made between the spur gear and the internal is that both center of the cogwheels are on the same side of the contact point which means that the direction of the driving and driven cogwheels are the same. The gear ratio of an internal gear is the same as for the spur gear,

$$U = \frac{\omega_1}{\omega_2} = \frac{z_2}{z_1} \tag{4}$$

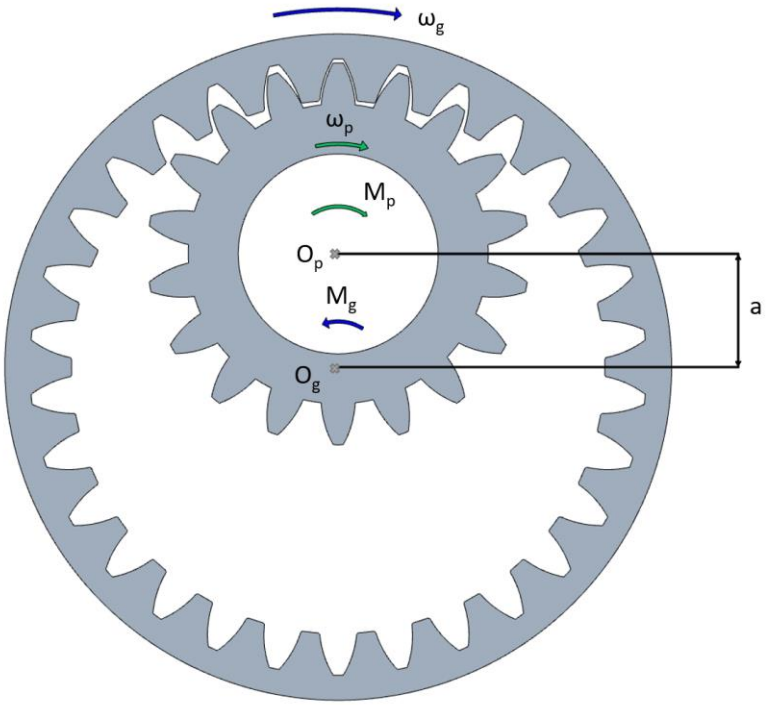


Figure 15: Internal gear illustration, arrows marked in blue refers to the big cog, green for small. Reproduced from: (Vedmar, 2012)

As seen in Figure 15, the cogwheels are fitted at a fixed distance a.

Bevel gear

If the axes of the two cogwheels are not parallel but have an angle offset, a bevel gear is a way of transmitting movement and force between the wheels. Most applications of bevel gears are for axes that are perpendicular to each other. (Childs, 2019)

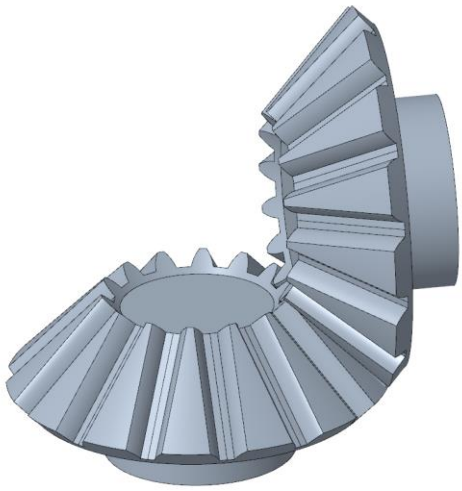


Figure 16: Illustration of a bevel gear. Reproduced from: (Kohara Gear Industry, n.d.)

Since the working principle of a bevel gear does not part from the previously mentioned principles of cogwheels, there is no need for further explanation of the working principle.

The gear ratio of a bevel gear is the same as for the spur gear,

$$U = \frac{\omega_1}{\omega_2} = \frac{z_2}{z_1} \tag{5}$$

Worm gear

The worm gear is a form of screw transmission, where the thread rises, which is perpendicular to the rotational axis, is suitable for transmission of movement and force in this direction.
(Vedmar, 2012)

A worm gear consists of a screw (also called worm) and a cogwheel, see Figure 17. The screw can have multiple entries, called multi-start worm. This means that each thread rise s becomes multiplied by the amount of a multitude $1 / g$. This implies that the total rise of the worm gear becomes $\frac{s}{g}$. When the worm is rotated one lap, the transmitted axial movement is equal to one rise of the worm. As for multi-start worms, the amount of rotations needed in order to achieve one rise of axial movement is $1 / g$ laps. As the wheel turns it is connected to the screw, it also turns but related to the number of cogs on the wheel $1 / z$ laps. (Vedmar, 2012)

The gear ratio of a worm gear can therefore be calculated using the rotational speed of the worm and the wheel, φ and ω.

$$\frac{\varphi}{\omega} = \frac{\frac{1}{g}}{\frac{1}{z}} = \frac{z}{g} \tag{6}$$

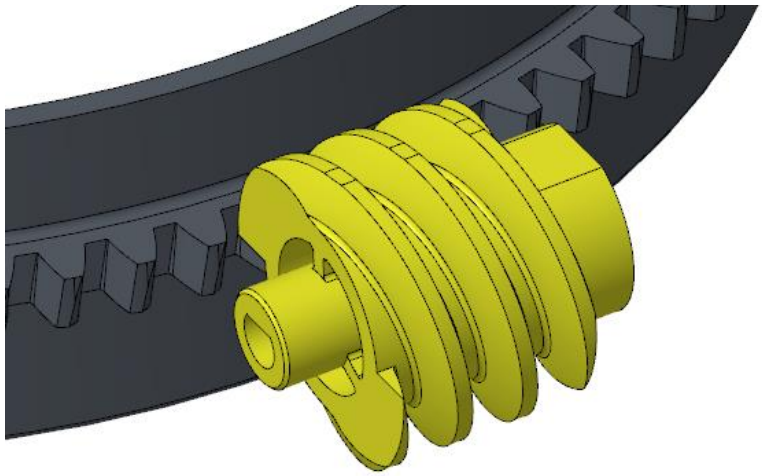


Figure 17: Illustration of a worm gear. Reproduced from: (Bunnell & Najia, 2020)

2.3.2 Belts

In a belt transmission system, movement and torque is transmitted between two circular wheels which are connected by a belt which encloses the wheels. By friction between the wheels and the belt, forces can be transmitted which means that the belt is a mean of transportation between the wheels. Belt transmission is an alternative to the spur gear but requires more space, has less efficiency and, since there is slippage between the wheels and the belt, there is no geometrically consistent gear ratio. On the other hand, since the belt is somewhat elastic the drive is more silent than a conventional spur gear and the belt enables for connection between axes that are further apart. (Vedmar, 2012)

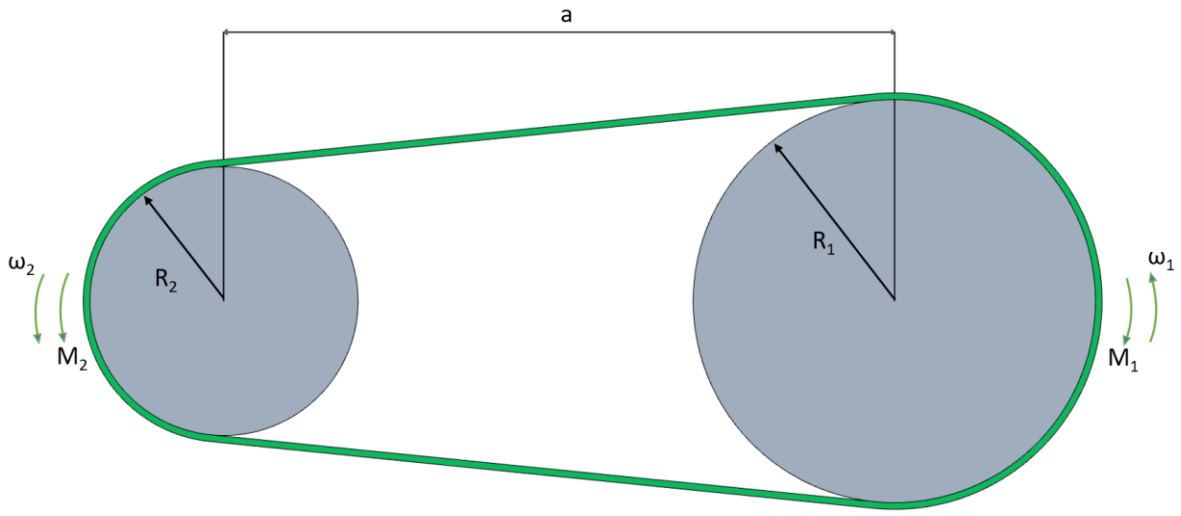


Figure 18: Simple illustration of a belt transmission, wheel 1 to the right, wheel 2 on the left. Reproduced from: (Vedmar, 2012)

There are fundamentally three different types of belts that can be used in a belt transmission: flat belts, V-belts, and toothed belts. As stated before, there is a risk of slippage when using belts, especially flat belts. In this regard the toothed belt is superior since the risk of slippage is non-existent. (Vedmar, 2012)

A flat belt transmission consists of a rectangular shaped belt, if seen as a cross section, and the forces are transmitted through friction between the belt and the wheels along the contact surfaces which means that pressure has to be applied in order for the friction force to be able to transmit the torque necessary. As seen in Figure 18 above, torque is transmitted from the smaller wheel to the larger through a flat belt. If there is no slippage in the drive, the belt achieves the speed of $\omega_2 R_2$ which means that the larger wheel gets a rotational speed of

$$\omega_1 = \frac{\omega_2 R_2}{R_1}$$

The speed ratio can therefore be calculated as

$$U = \frac{\omega_2}{\omega_1} = \frac{R_1}{R_2}$$

If the friction force of a flat belt is not enough to transmit the torque a V-belt can be used instead. The V-belt differs from the flat belt as it is wedge-shaped in cross section, see Figure 19. This means that the pressure at the contact surface is not only directed in the radial direction. If the radial component of the pressure in a V-belt is as large as the pressure in a flat belt the resulting pressure will be larger with a V-belt. (Vedmar, 2012)

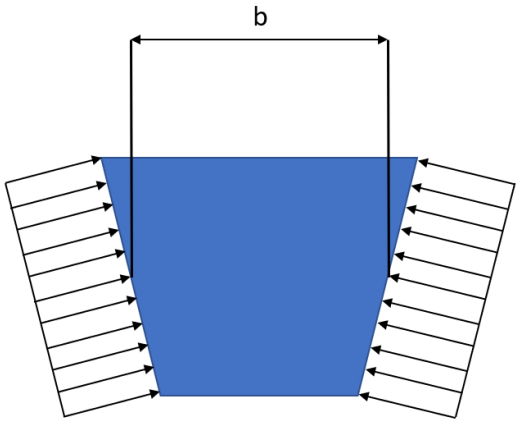


Figure 19: Cross section of a V-belt. Reproduced from: (Vedmar, 2012)

A toothed belt is fundamentally a flat belt that has been fitted with cogs. Subsequently the wheels are also fitted with matching cogs. The clear advantage of using a toothed belt is that there is no risk of slippage between the belt and the wheels which means that the gear ratio can be said to be consistent. Another

advantage is that there is no need for an external force to keep friction between the belt and the wheels which means that the storing of the wheels is at lower stress. (Vedmar, 2012)

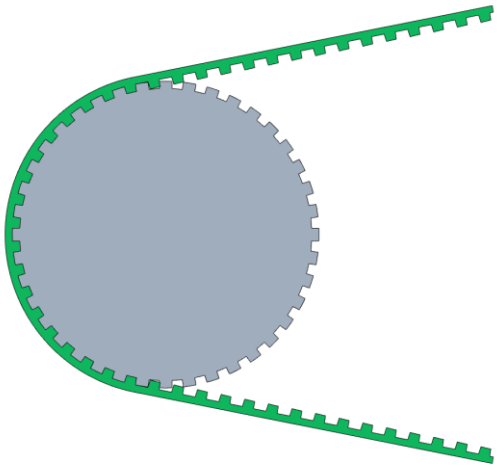


Figure 20: Simple illustration of a toothed belt transmission. Reproduced from: (Vedmar, 2012)

2.4 Actuation

This chapter will describe the theory behind all the actuation methods considered during this project, with focus on their respective advantages and disadvantages in different applications.

Generally, an actuator can be described as a component within a system or machine that is responsible for moving a mechanism, for example allowing a camera to move in different directions.

The subject of actuation is vast, with loads of different methods and power sources used in all sorts of machines and appliances. The type of actuation used in a product is selected based on the application needs and its requirements. Deciding factors can be anything from size, torque, speed, cost, reliability, accuracy, and power consumption.

In this master's thesis, only electrical actuation was considered since the camera is already connected to a power outlet. When it comes to converting electrical power to mechanical movements there are two main alternatives, linear and rotational movement. These movements can in turn be achieved with different technologies giving them different advantages or disadvantages.

2.4.1 Rotary Actuator

Rotary actuators convert the chosen power source to a rotational movement or/and torque. They can operate continuously or by fixed angular steps depending on their construction. They have no limit to their range, meaning they can rotate endlessly in either direction. When it comes to electrical rotational actuators the main alternatives are stepper motors and DC motors.

Stepper motor

The stepper motor is one of the most widely used actuators in this category. It uses a magnetic field to generate rotational torque. Figure 21 shows the general working principle of the stepper motor. It consists of a core, called a rotor, which can be either a permanent magnet or iron core. This core is surrounded by stator windings which generate a magnetic field then it is powered. By switching the power to the different stator windings, the rotor can be moved accordingly. (Fiore, n.d.) (Ida, 2014)

The performance of the stepper motor varies based on its internal construction. They are categorized by the type of rotor they use. But performance can also vary depending on the design of the rotor and the number of stator windings.

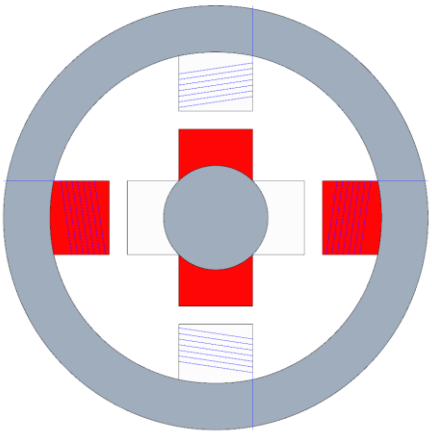


Figure 21: Cross section of a simple stepper motor.

There are the three main types of stepper motors:

1. Permanent magnet stepper motor (PMSM)
2. Variable reluctance motor (VRM)
3. Hybrid stepper motor (HSM)

The shared characteristics between these types are that they all use an electrically generated magnetic field to make rotational steps. For every electrical pulse one step will be made by the rotor. The number of steps in one revolution depends on how many stator windings, also called poles, the stator has. More poles will lead to higher precision. For example, a stator consisting of n poles with x rotor poles will need $n \times x$ electrical pulses to complete one full rotation. This in turn will result in a step length of $\frac{360}{n \times x}$ degrees. This makes the stepper motor useful in applications where low-level position control is desirable since it does not require any encoder or feedback to the system. (Fiore, n.d.)

Permanent magnet stepper motor (PMSM)
In this configuration a permanent magnet is used as the rotor. When subjected to the magnetic field generated by the stator windings it will align with the poles generating torque and ultimately movement. It will, due to this construction, also generate detent torque, meaning it will resist any change of position without any of the coils being active. This can prove useful in applications where it is desirable to lock the motor in place. One drawback with this is that it also has to overcome the threshold torque in the application it is used in to be able to move at all. Another drawback is that it needs to be connected to power at all time to keep its detent torque and thereby also its position. (Fiore, n.d.)

Variable reluctance motor (VRM)
In this configuration an iron core is used instead of a permanent magnet. The principle behind this is that the minimum reluctance occurs then the distance between the rotor and stator is at its smallest. The rotor will therefore follow the active stator pole allowing for higher speeds and resolution than for the PMSM but with lower torque and no detent torque. (Fiore, n.d.)

Hybrid stepper motor (HSM)

This configuration uses a combination of the two above. It uses both permanent magnet and reluctance to achieve better performance overall. This construction has the advantages of both the VRM and PMSM but with greater complexity and at a higher cost. (Fiore, n.d.)

DC motors

Just as stepper motors, DC motors are widely used in all sorts of electronic devices. Their working principle is, just like the stepper motor, based on magnetic fields but with different characteristics. The stator is a permanent magnet or an electromagnet with stationary field windings. The rotor, called armature, contains armature windings which are electrified during use. In the DC motor the pole switching happens in the armature windings with the help of a commutator, see Figure 22, that automatically changes the electrified coils during rotation. This creates a continuous motion where the speed is proportional to the voltage supplied. This makes the DC motor useful in high speed application with a simple Pulse Width Modulation (PWM) signal. On the other hand, it lacks the incremental steps that the stepper motor has and therefore needs an encoder or position feedback to be able to have any sort of position control. Due to this continuous motion, it does not generate any detent torque meaning it requires some sort of locking mechanism while the motor is inactive. The high speed of the motor can be converted to torque with a gear, allowing for high torque output at a cost of size. (Circuit globe, n.d.) (Ida, 2014)

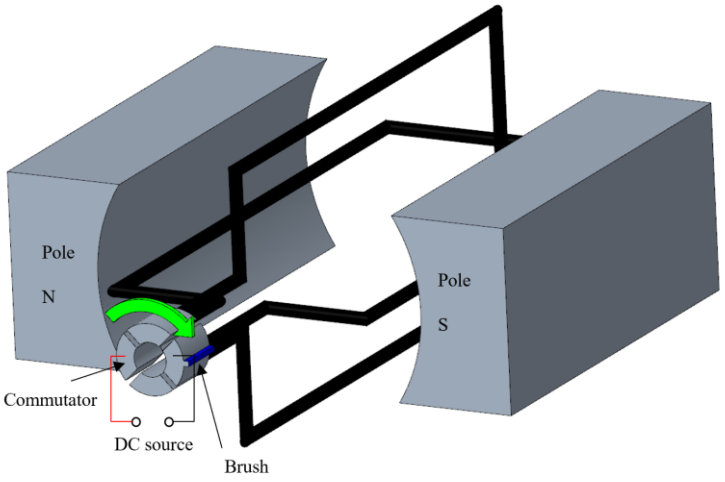


Figure 22: Illustration of the working principle of a simple DC motor. Reproduced from: (Ida, 2014)

2.4.2 Linear Actuator

Conventional electric motors are typically suited for rotational motion. In some applications it may, however, be more desirable with a linear motion to fulfil the application needs. The linear motion can be converted from a rotary motor with a combination of different gears and/or belts. This is however not space efficient and will not be included in this chapter.

Direct linear actuator converts the chosen power source to a linear motion. They generally have a rather small working range and have varying characteristics when it comes to torque and direction depending on the technology behind the actuator. The most common electrical linear actuators are voice coils and solenoids.

Voice coil

The voice coil actuator consists of a permanent magnet at its core surrounded by an electrified coil. The current flowing through the coil will interact with the magnetic field of the permanent magnet and push the shaft in either direction depending on the direction of the current, it is bidirectional. Since it is driven radially from the current in the coil, the force generated on the shaft will be proportional throughout the whole stroke. This characteristic makes them perfect for applications that need position control. Just like the DC motor which works on the same principle, the voice coil will be fully disengaged if the power is turned off. (Ida, 2014) (H2W Technologies, 2015)

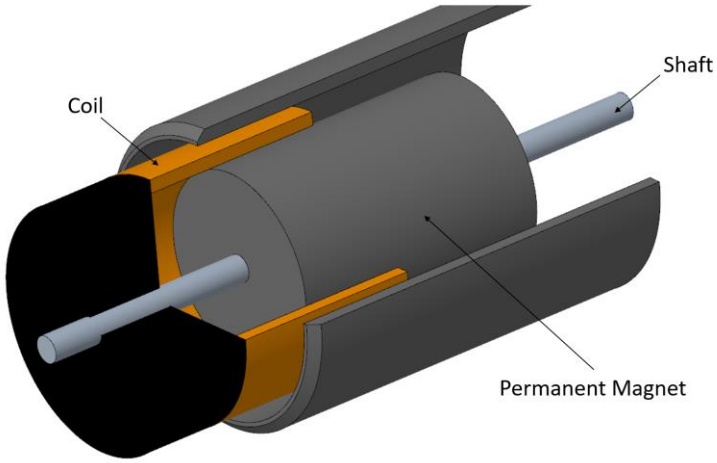


Figure 23: Cross section of a voice coil motor. Reproduced from: (H2W Technologies, 2015)

Solenoid

The solenoid actuator consists of a coil, magnetic end plate, spring, and a steel slug, see Figure 24. As the coil is electrified, it will create an electromagnetic field that will attract the magnetic end plate driving the steel slug forward. The stroke of the slug will have high initial force which will drop as it travels to its end point. When the coil is not electrified the force will drop to zero and the spring will pull the slug back to its starting point. This type of actuation method is typically used in on/off applications such as latches or valves. (Ida, 2014) (H2W Technologies, 2015)

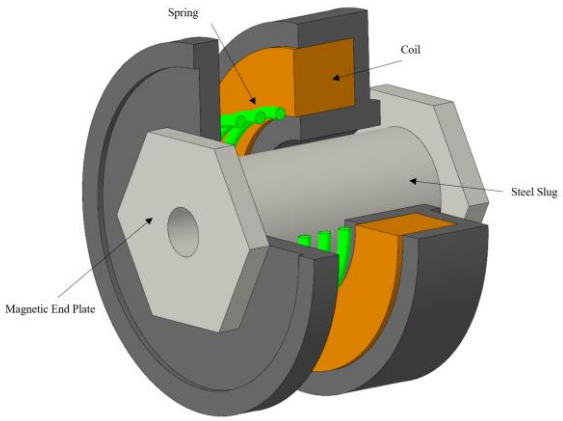


Figure 24: Cross section of a solenoid motor. Reproduced from: (H2W Technologies, 2015)

2.5 Electrical components

This chapter will explain the use cases and basic functionality of some important electrical components for controlling devices, particularly small motors in embedded systems.

To successfully control electric components in an embedded system, such as a camera, some sort of electronic circuit is essential. This is usually solved by using a tailor-made Printed Circuit Board (PCB) with the functionalities needed for the specific product. This is advantageous for several reasons when creating a finalized product. They are cheap, small, and customizable for their purpose which makes it easy to configure the measurements and size of the footprint to be integrated into the product.

In the prototyping stage of product development, creating a PCB is not necessarily desirable. A PCB is optimized for its purpose making it hard to do rapid prototyping or tests before the design is set. In these cases, when the electrical design is still under development, it is often easier to create the electrical system using integrated prototype circuits. This makes the circuit highly customizable during the development process. The only drawback being that the size of the prototype board is not representable for the footprint of a PCB that will be included in a final prototype, which will require some mechanical redesigns to be implemented at a later stage.

When it comes to creating prototype circuits for embedded systems the most common starting point is to use some sort of microcontroller.

2.5.1 Microcontroller

A microcontroller is an integrated circuit containing a digital processor and other useful peripherals, such as memory, programmable inputs/outputs, internal clock, and communication channels. The integrated functions vary based on brand and product type, but the general description is that a microcontroller is a computer chip or the brain of an embedded system.

Having all the necessary components integrated into one board decreases the complexity for the developer wanting to create a controllable system. It also allows the user to easily integrate desired functions or add-on boards without having to worry about compatibility.

Then it comes to the programming environment, microcontrollers support most binary languages such as C, Python, and Java. Some languages are more common due to the computational limits of small microcontrollers. Most common is C, which has the advantage of large open source firmware libraries, easily accessible online.

More complex microcontrollers allow for more advanced programs with multiple processing cores allowing for concurrent programs. This is of great use in server-based applications or complex embedded systems. For simple applications, a sequential single core microcontroller is often sufficient. (Hunter, 2020)

2.5.2 Transistors

Transistors are an essential core device in modern electronics. They allow the user to amplify or switch on/off a circuit much like a physical switch, with the advantage of being controlled digitally. They can be used separately or in combination with other devices on integrated circuits creating controllable electrical systems.

There are various types of transistors on the market, based on different technologies, with different advantages. The two main types are Bi-polar Junction Transistors (BJT) and Field Effect Transistors (FET), with a lot of subtypes and variations. They are both constructed using a semiconductor, which is a material that conducts current when some conditions are met, typically a voltage threshold. This condition is utilized to switch on or off the transistor junction and allow current to flow through.

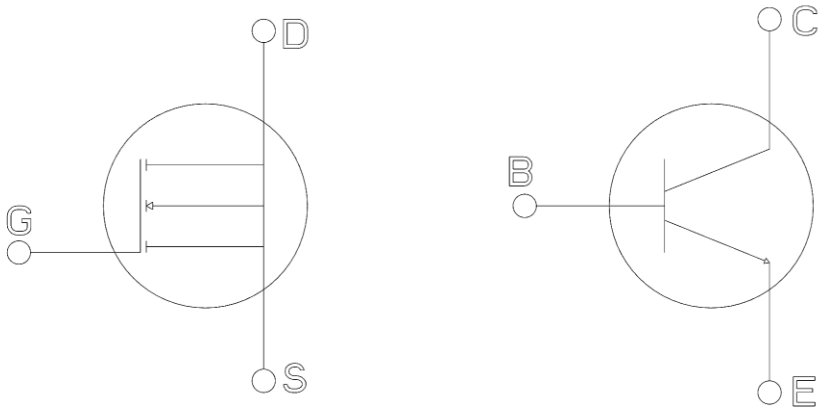


Figure 25: Principle schematic of a general FET (left) and BJT transistor (right).

In Figure 25, a simple schematic of the two different transistor types is shown. As can be seen both have three terminals. The FET uses gate, source and drain, where the gate is voltage driven and can be driven

on low or high depending on type to allow current to flow from source to drain. The BJT transistor on the other hand has a base, emitter, and collector. The general functionality is the same but the BTJ is driven by current on the base. The current allowed to flow through from the emitter to the collector is a multiple of the current on the base, known as “gain”. (Agarwai, n.d.)

Both transistors have unique characteristics and there is no obvious better option. It all depends on application and what is the goal. As mentioned previously the FET is voltage driven which makes it useful in low current applications, such as for example microcontrollers where the maximum current output is limited. The BTJ is current driven and less complex in its structure. This makes it good for simple circuits where the current drain of the component is predictable, such as a Light Emitting Diode (LED). It also works well as an amplifier since the transistor gain can be utilized easily. (James & Granath, 2020)

2.5.3 H-bridge

In general, a H-bridge is a rather simple circuit, It utilizes four transistors to alternate the direction of the current flowing through the load, this can be seen in Figure 26, where a motor is connected to the circuit.

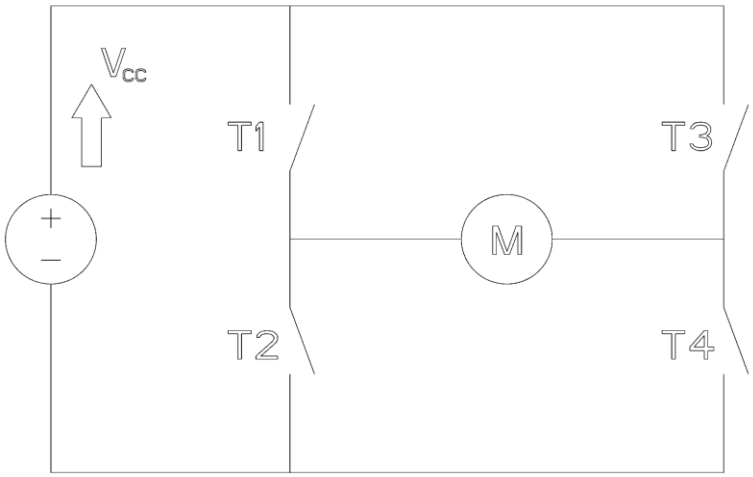


Figure 26: Working principle schematic of a H-bridge.

An actual H-bridge will include several diodes to ensure that the current does not flow the wrong way, but the working principle can be illustrated easier with only the transistors. So, by switching T1 and T4 the current goes from left to right through the motor and when switching T3 and T2 the current goes the opposite direction. This allows for bi-directional drive, often desirable in motor applications. (Modular Circuits, n.d.)

2.6 Communication

Implementing a remote-control functionality requires a communication channel between the user application and the microcontroller of the device. There are a lot of different alternatives to choose from

with different advantages in terms of functionality, reliability, and ease of implementation. In this project, I2C and Ethernet communication were used since these methods are used frequently in embedded systems.

2.6.1 Ethernet

Ethernet is a commonly used technology for communication between two or more devices over a network. It has been around for a long time and is popular for its speed, reliability, and flexibility. It can be set up over both a Local Area Network (LAN) or a Wide Area Network (WAN). It uses pre-determined protocols and set of rules for communication. Sending and receiving messages requires the devices to be using the same Ethernet protocol language to understand each other. The communication is bi-directional but does not necessarily require any acknowledgment in-between the devices. (Weslay, Irei, & Burke, 2020)

Using Ethernet communication in prototyping requires that the microcontroller has Ethernet functionally incorporated. Most microcontroller either have Ethernet capability on them or the possibility for an Ethernet extension. Implementing the communication protocols can be done with already existing libraries for the given microcontroller.

One common protocol used in microcontrollers is UDP. It utilizes the Ethernet IP layer when sending and receiving messages over ports without any direct connection between the host and receiver. Because of this, it is easy to implement and set up. On the other hand, there is no guarantee of message delivery and no way of tracking messages since there is no connection or acknowledgment between the devices communicating. Using this communication for prototyping can be advantageous even with its flaws since it is simple, and the application device can be set to listen to incoming messages on the given network port.

2.6.2 I2C

Inter-Integrated circuit, or I2C, is a common communication protocol suitable for communication between on-board peripherals in embedded systems. It uses a master/slave structure as shown in Figure 27. The master works as a communication hub and can be connected to several slaves. The advantage of using this type of communication compared to just connecting wires to the master is that all the slaves can be connected to the same communication line. This means that only two communication wires are needed, the communication line (SDA) and a clock line (SCL), independent on how many slaves are connected to the master. This can decrease the number of cables needed in applications that uses many peripherals. (Liu, Bao, Meng, Xu, & Liao, 2019)

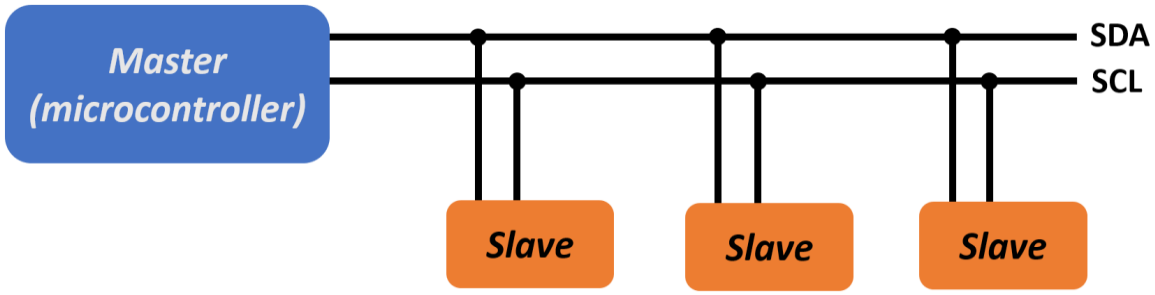


Figure 27: Example of I2C communication. Reproduced from: (Electronics Hub, 2018)

The communication works by sending individual bits through the SDA line. A high voltage represents a 1 and low voltage represents a 0. The line is naturally pulled high to the same voltage as the supply. The master can then control the voltage on this line, by pulling it down to zero. The slave can read the SDA line based on the clock sampling time set by the SCL line. (Electronics Hub, 2018)

The expected sequence, set by the I2C protocol, is described below and shown in Figure 28:

1. The master pulls the line low to inform the slaves to listening for a message.
2. The master sends the address of the desired slave together with a bit indicating to the slave if the master wants to transmit or receive data.
3. The slave with the corresponding address sends an acknowledgment bit to indicate that it understood and is ready to communicate.
4. The slave/master then proceed to send 8-bits of data depending on the READ/WRITE bit and the receiver answers with an acknowledgment bit (ACK).
5. When all strings of data have been sent, the master ends the communication with a STOP-bit.

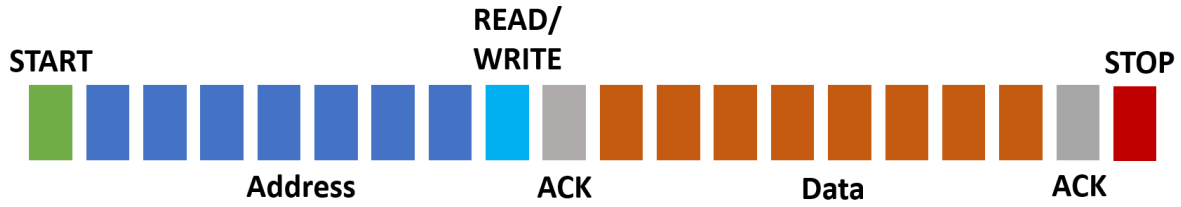


Figure 28: Illustration of the I2C communication protocol. Reproduced from: (Electronics Hub, 2018)

3. System Specifications

3.1 Customer Needs

To be able to set specifications for a product, customer needs are to be determined. The customer in this case is not the final customer of the product but instead product specialists and managers at the company. These needs were chosen by conducting interviews with both supervisors and product specialists as well as studying similar products with similar criteria of robustness, simplicity, cost etcetera.

Since the goal of this project is to produce a working prototype and not a final product, some of the criteria like robustness cannot be verified through testing. As these criteria will not be addressed in this report, they may be applicable for future work.

The identified customer needs are shown in Table 1 and are graded in order of importance. All the needs are graded on a scale of 1-3 where 3 is of highest priority, 2 is lower and 1 is of lowest priority.

Table 1: Customer needs with comments and graded importance.

No.	Need	Comment	Imp.
1	The solution should be flexible	Pan, tilt, rotate and twist	3
2	The solution should be move or drift when exposed to vibrations		3
3	The solution has a low cost		1
4	The solution should be faster to install than by hand		3
5	The solution should be compact		3
6	The solution should be powered by the existing camera hardware		2
7	The solution should be intuitive to use	Controllability	2
8	The solution should be easy to assemble		2

3.2 Metrics

When the needs of the projects have been established, the next step in the development process, according to Ullrich and Eppinger, is then to translate these into metrics. It may be necessary to translate one need into multiple metrics and the goal is to translate the needs into quantifiable specifications in order to create precise targets for the development process. All the identified metrics are shown in Table 2, with corresponding units and optimal and acceptable values. The relationship between customer needs and metrics are shown in Table 3.

Table 2: Identified metrics from customer needs.

No.	Need no.	Metric	Imp.	Unit	Optimal	Acceptable
1	1	Each module should be able to move in pan, twist, tilt and rotate.	3	Nr.	4	4
2	1	Range of motion pan	3	°	360	360
3	1	Range of motion twist	3	°	±25	±20
4	1	Range of motion tilt	3	°	95	95
5	1	Range of motion rotate	3	°	100	95
6	2	Pass vibration test	3	Bin.	Y	Y
7	3, 8	Cost	1	kr	-	-
8	4, 7	Installation time	3	s	-	-
9	5	Outer diameter of concept	3	mm	200	300
10	5	Height of concept	1	mm	100	200
11	6	Max. power	2	mW	1000	800

Table 3: The Need-Metrics matrix, an illustration between the relationship between the identified customer needs and the metrics.

	Metric	Need	1	2	3	4	5	6	7	8
			The solution should be flexible	The solution should be not be altered when exposed to vibrations	The solution has a low cost	The solution should be faster to install than by hand	The solution should be compact	The solution should be powered by the existing camera hardware	The solution should be intuitive to use	The solution should be easy to assemble
1	Each module should be able to move in pan, twist, tilt and rotate		●							
2	Range of motion pan		●							
3	Range of motion twist		●							
4	Range of motion tilt		●							
5	Range of motion rotate		●							
6	Pass vibration test			●						
7	Cost				●					●
8	Installation time					●			●	
9	Outer diameter of concept						●			
10	Height of concept						●			
11	Max. power							●		

4. Concept Generation

In this chapter we will present the initial mechanical and electrical concepts for this project. For the movements rotate, tilt and twist, the emphasis in the concept is on the general placement of the motor and the type of actuator which means that a general camera sensor and module is used in order to get a sense of the relative placement of the concept contra the camera module. The pan movement has little to no real connection to the camera module itself which means that the concepts will be displayed separately.

As for the electrical concepts, the different actuator types are presented with data in order to be able to perform the concept scoring in the next chapter.

4.1 Mechanical concepts

During the concept generation process several possible solutions for the mechanical construction have been created. All the different idees will be presented for the individual movements.

4.1.1 Rotate

As stated in Section 1.3, the foundation of the master thesis is the current camera sensor which is the starting point for the concept generation. There are two approaches to fitting the driving motor of the rotate movement, either on the inside of the sensor housing, or on the outside. If fitted on the outside of the camera housing, the size of the housing itself will be smaller but the position of the motor will be further from the rotational axis of the movement. This will lead to an overall larger package which is undesirable.

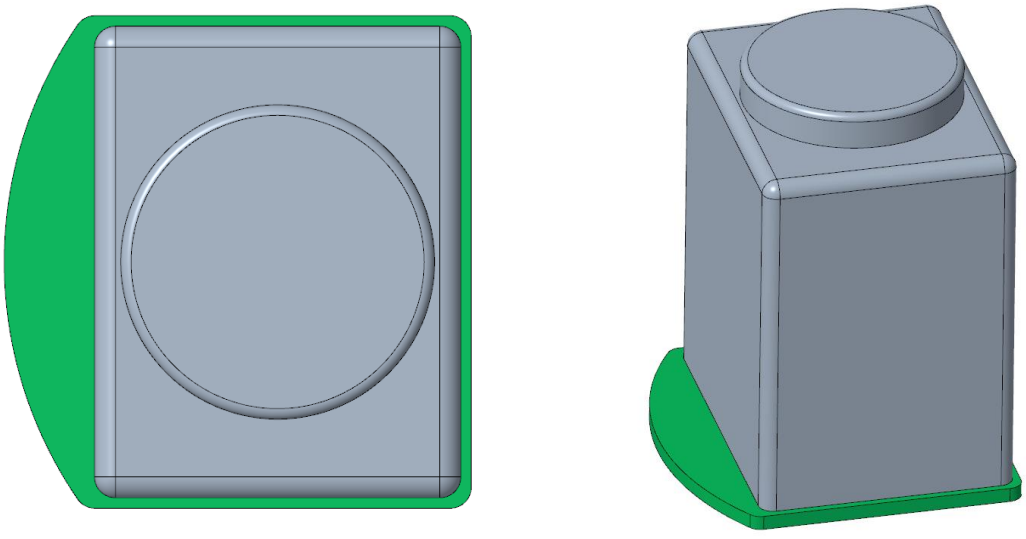


Figure 29: Mock-up illustration of the camera sensor used in the master thesis.

As seen in Figure 29, the shape of the sensor is rectangular. If considering the PCB in the bottom, there is more space to the right of the sensor since the contact of the PCB is to the left. There were no limitations to the design aspect of the camera housing itself apart from the size which should be as small as possible. When examining cameras in the multidirectional segment, most of the products had circular shapes on its sensor housings which was chosen as the design of the prototype as well. These two decisions led to the conclusion that the best fitting of the motor was on the inside of the camera housing as close as possible to the right of the sensor in Figure 29.

Top drive

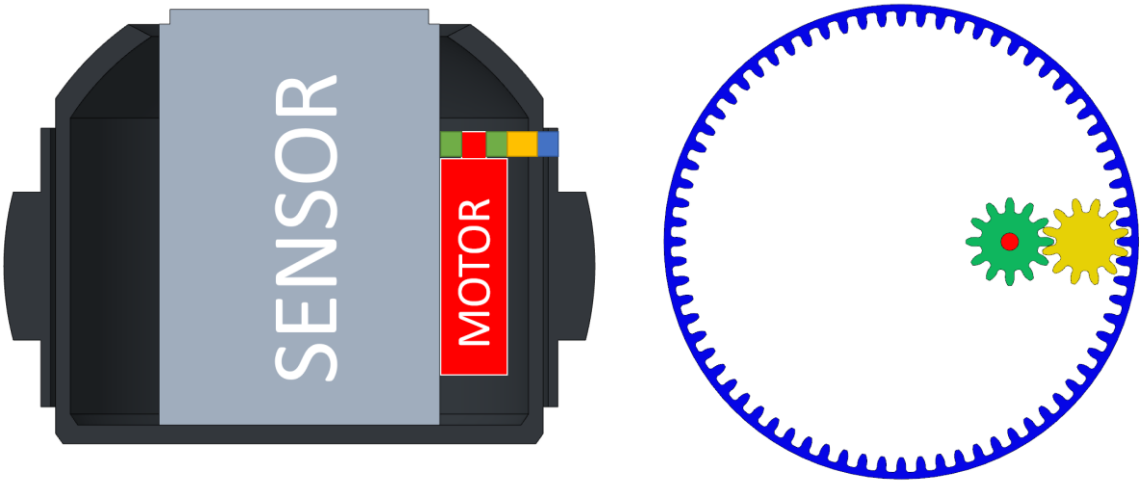


Figure 30: Illustration of the concept Top drive. The cogwheels are displayed in the same color in both figures.

This concept has the motor fitted vertically with the cogwheel placed upwards. The different parts can therefore be driven by fitting a larger outer cogwheel marked in blue in Figure 30. Due to the position of the sensor, the dimensions of the cogwheel fitted to the motor cannot exceed the outer dimension of the motor, which means that there must be some sort of an intermediate cogwheel between the driving and driven components.

Bottom drive

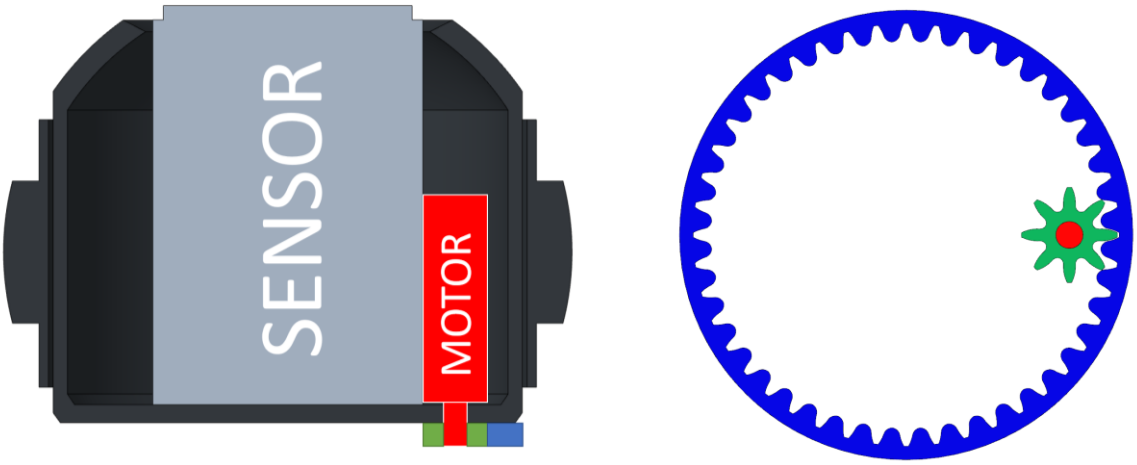


Figure 31: Illustration of the concept Bottom drive. The cogwheels are displayed in the same color in both figures.

This concept has the motor fitted in the same place as the previous concept but in the other direction with the cogwheel placed in the bottom of the sensor package. To make this concept possible, the shaft of the motor must pass through the bottom of the sensor package in order to be able to connect with the driven cogwheel placed below the sensor. As seen in Figure 31, there is no need for an intermediate cogwheel in order to drive this concept.

4.1.2 Tilt

The different concepts for the tilt movement focus on the placement of the motor which drives said axis. As the entire tilt package will have to be rotated by both the twist and pan movement, it is essential that the construction remains narrow radially. As for the tilt movement, there are also two different approaches. One is to place the motor horizontally and the other is to place the motor vertically.

Belt drive with horizontal placement

The main idea of this concept is that the motor package is placed below the camera module which enables a compact concept, especially radially. Consequently, the entire structure will be significantly taller, see Figure 32. The belt makes the placement of the motor flexible, but it must be said that the ideal placement of the motor is parallel to the axis it revolves, as it will be rotated by the twist movement. This is since its distance from the center of the camera module is the smallest directly below the axis.

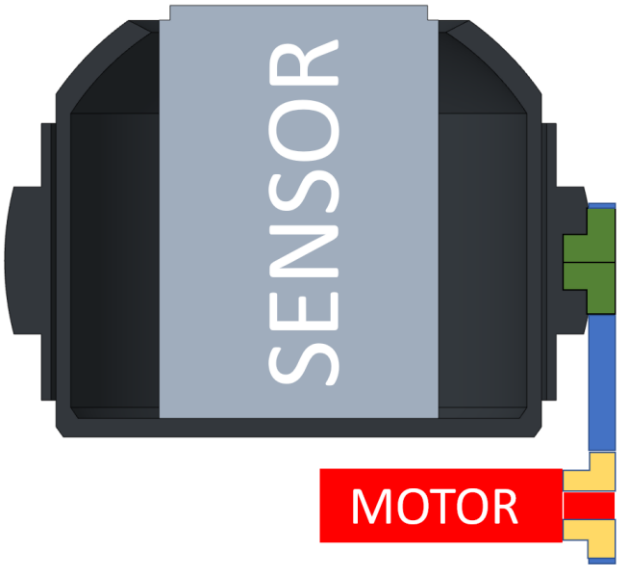


Figure 32: Illustration of the concept Belt drive with horizontal placement.

Bevel gear with vertical placement

Another way of placing the motor is vertically with a bevel gear with a 45-degree angle to transfer the torque to the axis which will be perpendicular to the shaft. As Figure 33 below shows, when the motor is placed vertically the entire construction will become taller by a substantial amount as well as wider. One difficulty with fitting the motor vertically is that the esthetical aspect might suffer since it may be difficult for the module to remain symmetrical without adding additional bulk.

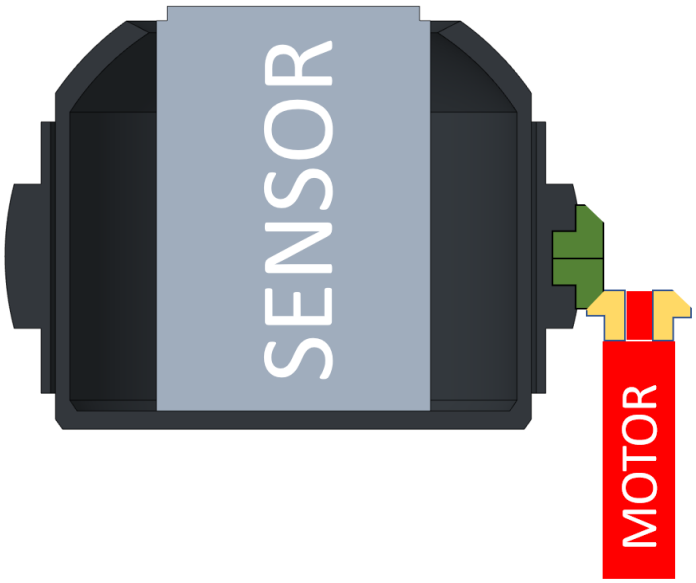


Figure 33: Illustration of the concept Bevel gear with vertical placement.

4.1.3 Twist

The concepts for the twist movement focus on the placement of the motor just like the tilt movement, but with some considerations to the chosen tilt concept and how they could be combined, as the space for the twist motor is highly dependent on where the tilt motor is fitted, see Figure 34.

Cogwheel with vertically placed motor

This concept uses a motor that is fitted vertically on the side of the sensor package and drives a cogwheel that is fitted below. This solution can be paired with the concept "Bevel Gear with vertical placement" for tilt which means that both motors are fitted in the same way but on each side of the module. This would make the solution symmetrical but considerably wider.

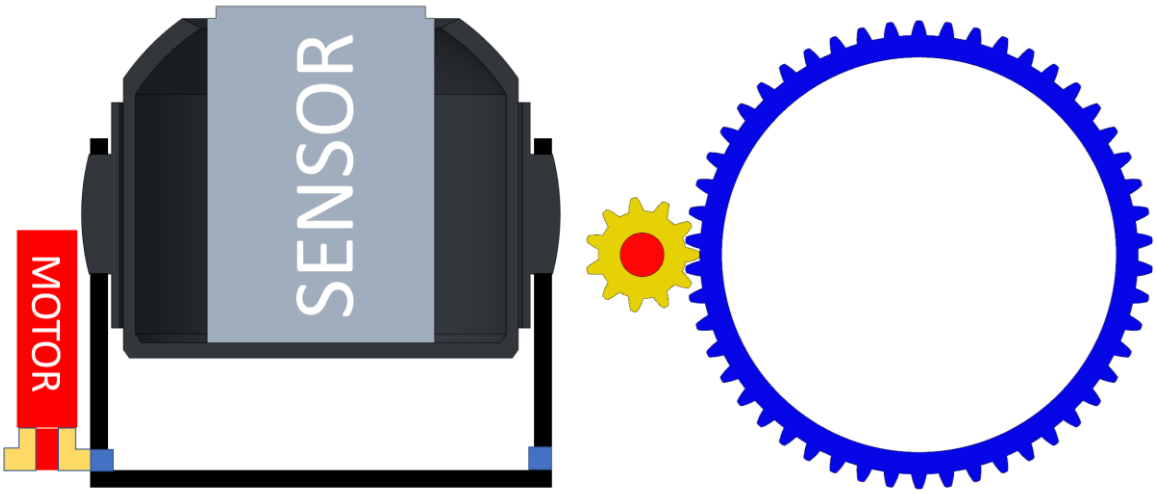


Figure 34: Illustration of the concept Cogwheel with vertically placed motor.

Horizontally placed motor

This concept places the motor below the module instead of to the side as in the previous concept. This makes the whole construction taller but not as wide. As for the two concepts mentioned before, this can be paired with the concept for tilt, “Belt drive with horizontal placement”, which places both motors directly below the module.

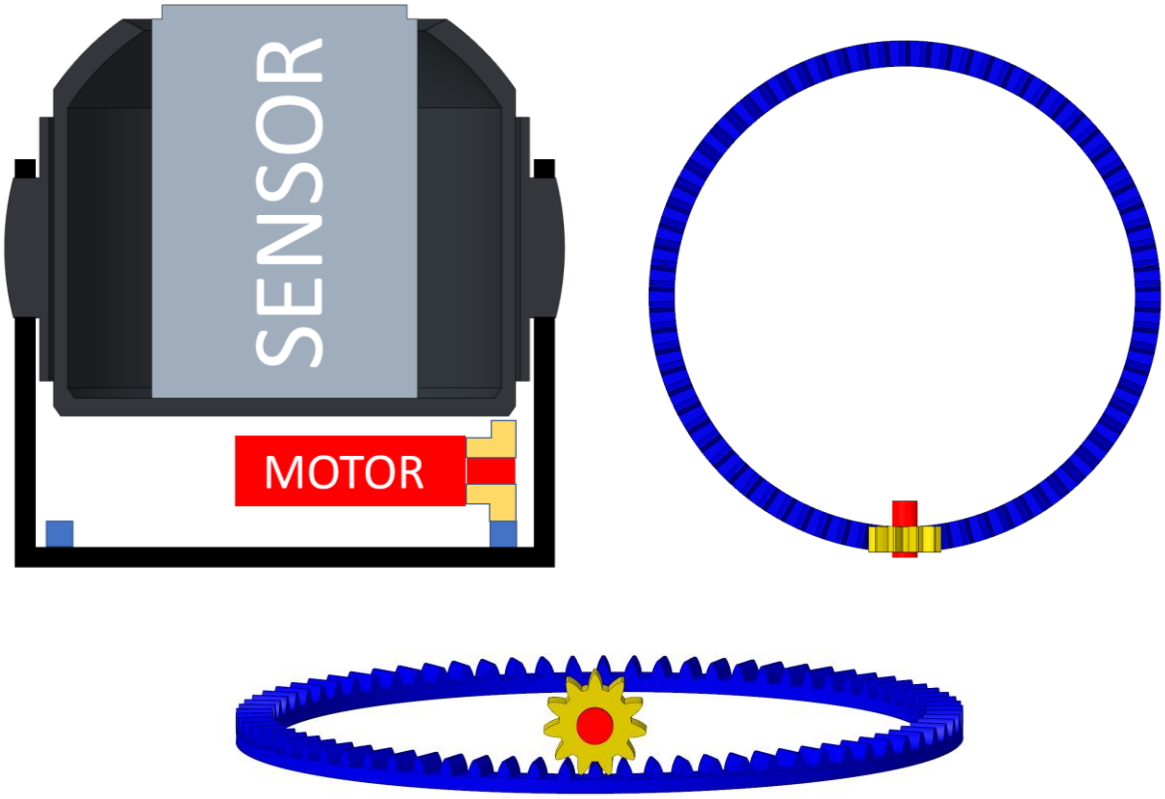


Figure 35: Illustration of the concept Horizontally placed motor.

4.1.4 Pan

A brainstorming session with the mechanical department at the company for the pan movement was conducted. The session gave a lot of possible solutions. The idees can be categorized into subgroups with similar working principles and will be presented below. It is the working principle of the concept that is shown in this chapter, not the physical application of it. Therefore, the camera modules are not placed in the concepts.

4.1.4.1 Single-Pan Solutions

The first subcategory is called Single-pan solutions and will solve the pan movement by using one motor for each camera module.

Large Cog

This concept uses the pan track of the multidirectional camera, with each module driven by a spur gear powered by an actuator. The gear could either be placed as in Figure 36, as an internal gear, or it could be placed as an outer gear or even placed vertically instead of horizontally, as in Figure 37. Each of the concept variations will be evaluated separately to be able to choose the best suited solution.

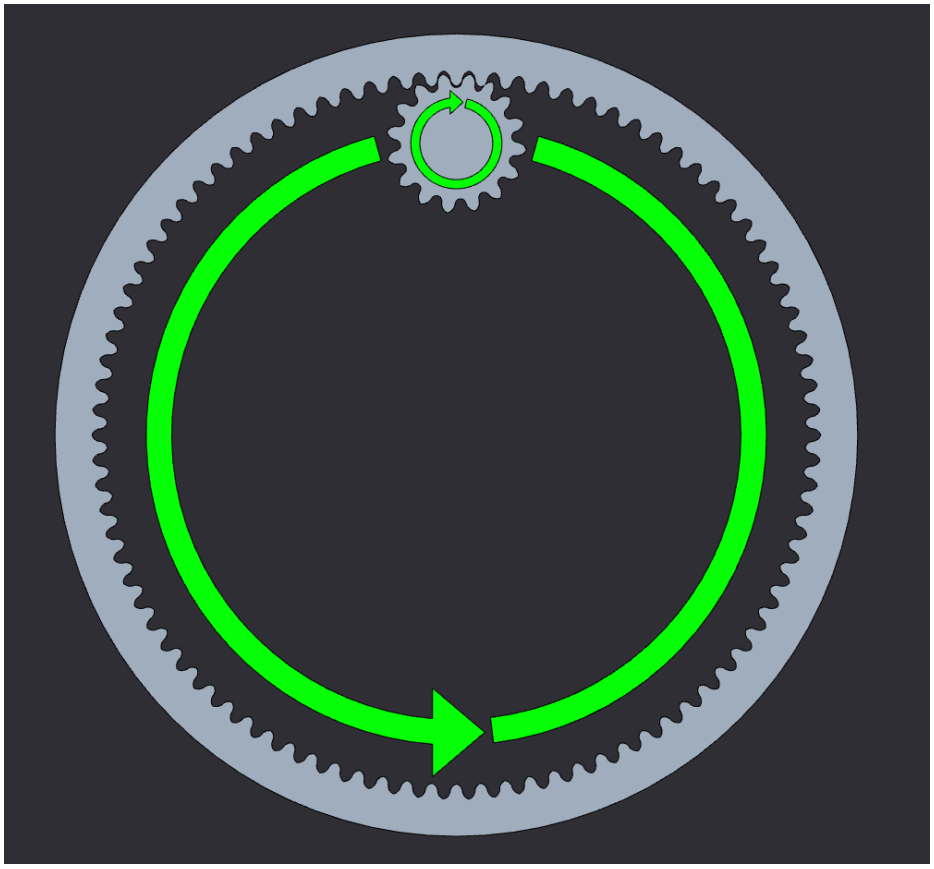


Figure 36: Illustration of the Large Cog Concept with a vertical placement.

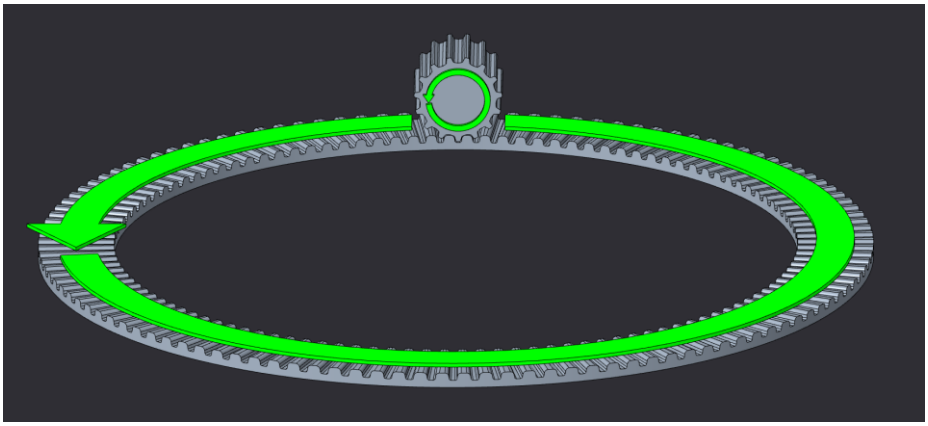


Figure 37: Illustration of the Large Cog concept with a horizontal placement.

4.1.4.2 Multi-Pan Solutions

The second subcategory uses a single motor in order to move all modules. Some sort of linear actuation is used to gain separate control of each module.

Locking with actuator

In this concept, each of the modules are placed with a linear actuator which is moved linearly in order to lock with a centrally placed rotating component. The geometry of the component that is placed in the middle does not necessarily need to be a cogwheel, but it is preferable because of its ability to lock the modules in place giving high transfer of torque.

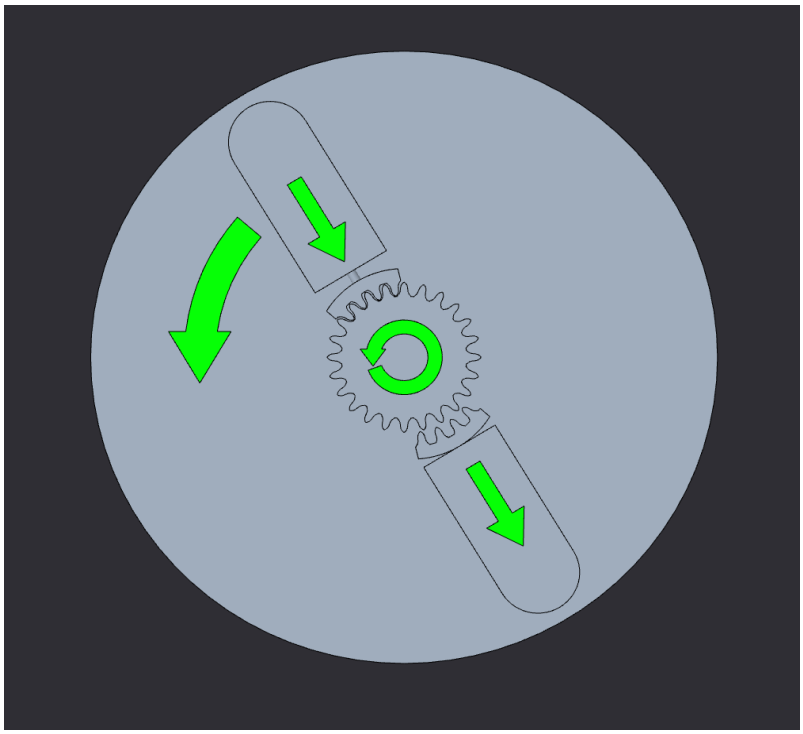


Figure 38: Illustration of the concept Locking with actuator.

Simple four step gear - Horizontal

This concept uses a single driven cogwheel that is placed in the center of the structure. The first iteration has a vertically placed cogwheel that can be moved radially in order to engage or disengage each cogwheel connected to the arm which in turn has a camera module fitted, see Figure 39. Depending on the placement of the center cogwheel, each cogwheel fitted with a module can be turned.

All the cogwheels must be synchronized in order for the transition between the driven states to be performed.

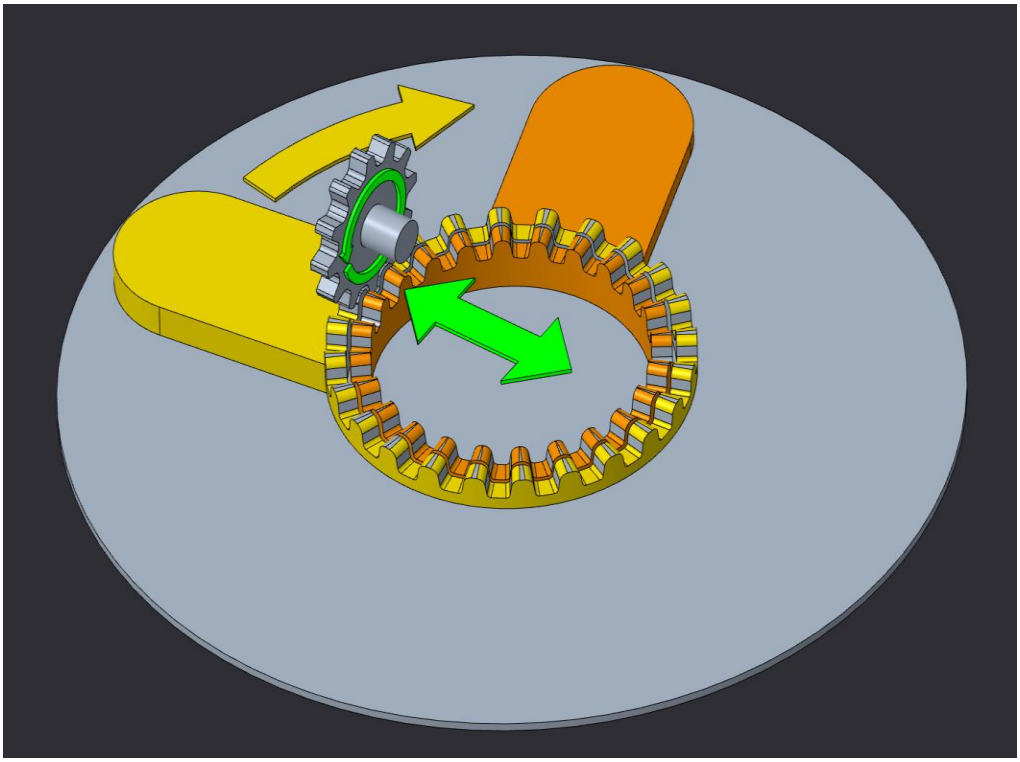


Figure 39: Illustration of the concept Simple four gear – Horizontal with two arms. Here, the yellow cogwheel is driven. As the driving cogwheel is moved inwards, the yellow cogwheel is disconnected, and the orange cogwheel is engaged.

Simple four step gear - Vertical

This iteration, of the same concept as above, uses the same principle, but the center cogwheel is placed horizontally and can be moved vertically in order to engage the arms with the camera modules, see Figure 40. All the cogwheels have to be synchronized in order for the transition between the driven states to be performed.

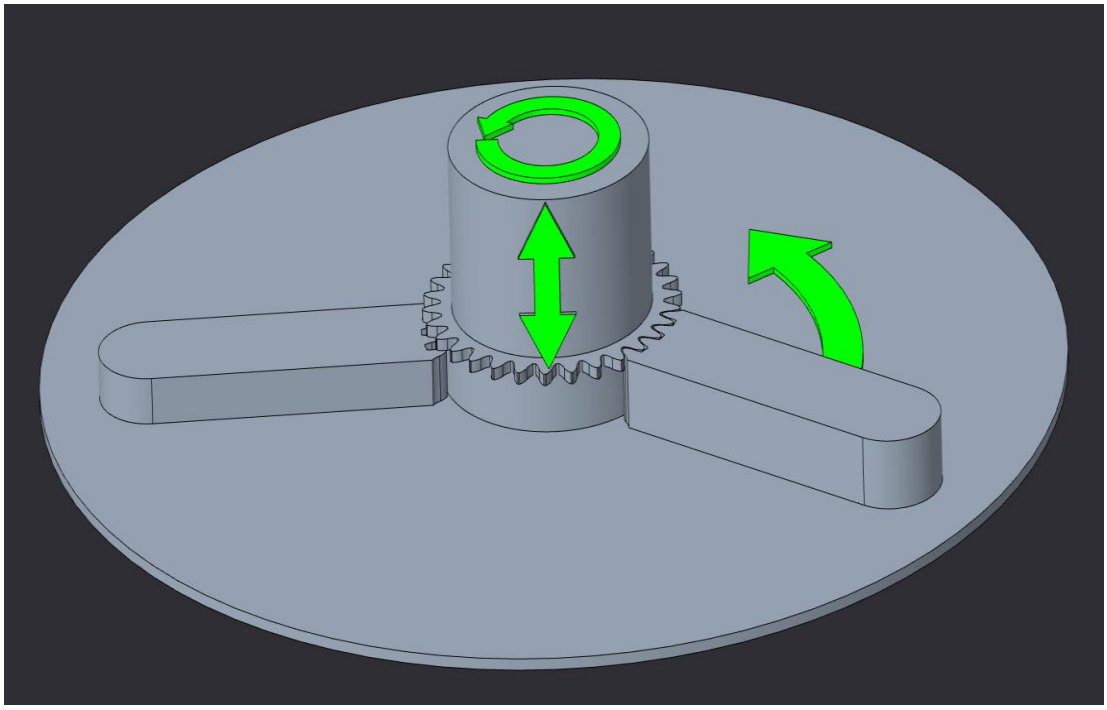


Figure 40: Illustration of concept Simple four step gear – Vertical with two arms.

Lateral complex gear - 16 modes

This concept is a more complex version of the previous one. The principle is the same as the horizontally placed single cogwheel that drives the arms but instead of just having the ability of controlling one arm at the time, this concept makes it possible to control all combinations of the arms at the same time with the restriction of only being able to drive them in the same direction.

This is made possible by a totem pole like axis in the middle which in Figure 41 only has four modes but can be fitted with more if the design is compact enough. If it is possible to fit 16 modes all the different arms can be moved in combination with each other in the same direction.

In Figure 41 the yellow arm is connected to the centrally placed gear. If the arm is lowered one mode, the orange arm will be connected to the centrally placed gear and vice versa. To make this possible, all the arms must be synchronized for the transition between the modes.

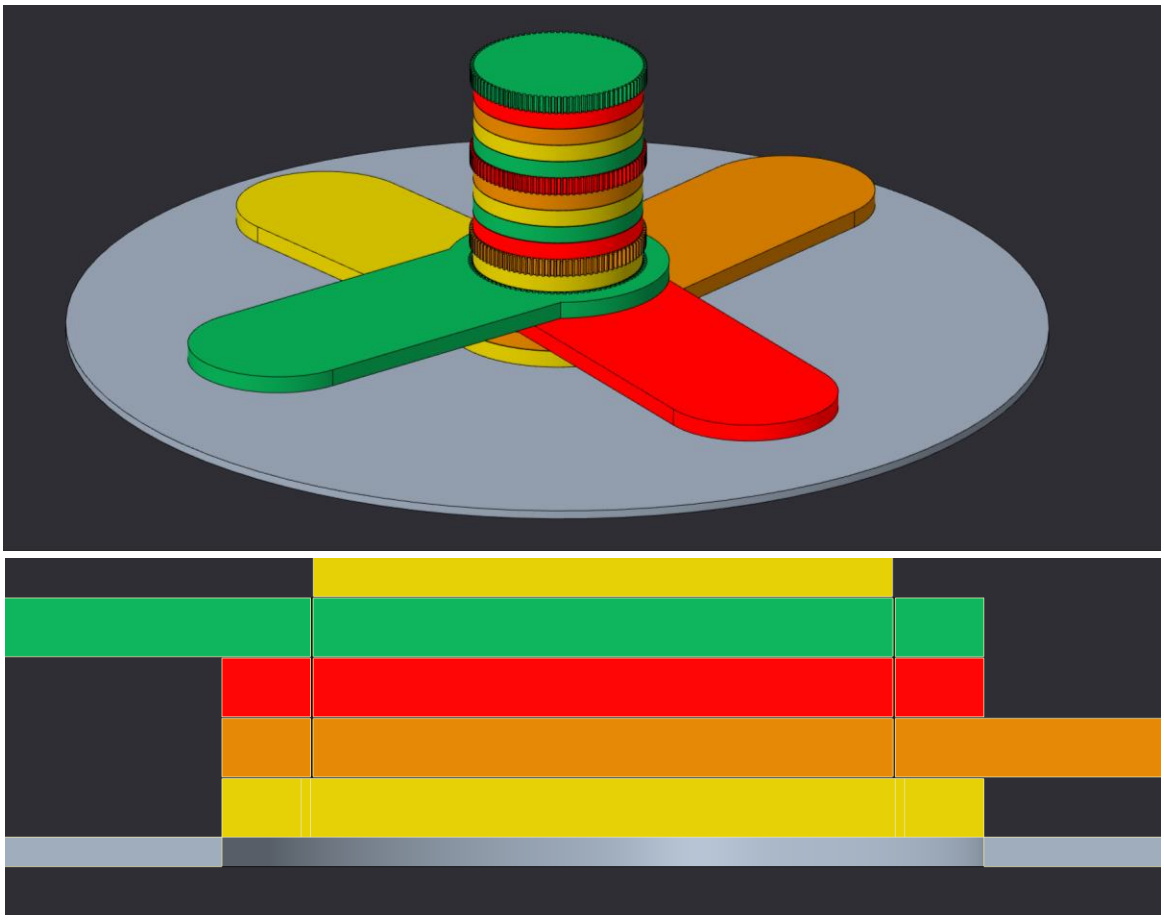


Figure 41: Illustration of concept Lateral complex gear – 16 modes with four modes, one for each arm. The lower illustration shows connection between the yellow cogwheels.

4.2 Actuation concepts

This sub-chapter refers to the actual actuation methods that could be used to achieve the mechanical movements in the previous chapter. These concepts are mainly focused on the torque, size and locking ability of the actuators.

Stepper Motor

The rotational movements in the mechanical concepts could be solved with a stepper motor. This would allow for simple position control and since the stepper motor locks itself and does not need a high gear ratio to lock any unwanted movement, during use. On the other hand, to ensure that the motor holds its position even when it is not powered it need to be combined with some sort of external locking ability, for example a linear actuator, which can be a disadvantage.

Inspiration on stepper motors has been taken from motors used in current cameras at the company, one example of this is stepper motor B. Using an already tested motor can be advantageous, since it is easily accessible for testing. Two extra motors have been listed below as extra alternatives for benchmarking,

both of the same brand. Most motors in this category have the same specifications with some variation in size and output torque.

Stepper A is of the same type as the one used at the company but with smaller dimensions and less torque output. Stepper C has a different configuration with smaller diameter and length that might be advantageous if the dimensions better fit the space available in our chosen mechanical concept. All stepper motors are benchmarked in Table 4.

Table 4: Benchmarking of stepper motors

Actuator	Stepper A	Stepper B	Stepper C
Diameter [mm]	15.0	20.0	10.0
Length without shaft [mm]	12.0	15.5	10.7
Resolution [step/rotation]	20	20	20
torque [mNm]	2.5 - 0.5	5.0 - 1.5	2.0 - 0.5

Geared Continuous DC Motors

This solution uses a DC motor with an integrated gear. This allows the motor to generate more torque than the stepper, at the cost of rotational speed. This combination of DC motor and integrated transmission is something that the company has used before in similar applications. The DC motor currently used is DC motor A. This motor has a rather high gear ratio which also makes it a bit larger than other alternatives on the market. It is possible, depending on the space available and torque needed, to choose a smaller motor with lower gear ratio and less output torque. The advantage with using the motor with high gear ratio is that the holding torque of the DC motor is proportional to the gear ratio since the holding torque comes from the friction between the cogs in the integrated gear. All the DC motors are benchmarked in Table 5**.**

Table 5: Benchmarking of DC motors

Actuator	DC motor A	DC motor B	DC motor C
Power	0.35W (3V)	0.29W (3V)	0.46W (6V)
Height [mm]	10	10	10
Width [mm]	12	12	12
Length without shaft [mm]	28.5	25.2	25.2
Speed [rotations/min]	11	54	176
Max torque [mNm]	272.6	34.5	19.8
Gear ratio	1000	298	102

Linear actuators

From the mechanical concepts, the linear actuator only acts as a locking mechanism in combination with a rotational actuator. Since this is the use case, a solenoid would be the preferred choice of linear actuator because it is designed specifically for on/off purposes.

No initial benchmarking of solenoids will be done since the available solenoids on the market mostly differ in size. It will therefore be investigated at a later stage, if it is needed, based on available space in the mechanical design as well as which mechanical concept is selected.

5. Concept Scoring

In this chapter, all the mechanical design concepts will be scored in order to determine which one will fulfill the product specifications best. Since the mechanical concepts are divided into four movements, there will be four winners, which will all be combined into a fully functioning concept. Also, the different actuators will be scored and ranked as one category in order to select the best actuator regardless of type.

The scoring was performed with weighted criteria, which were selected in collaboration with both supervisors and other stakeholders at the company in order to determine the criteria in the most suitable way possible. The weights were set in an ascending order from 1, which is least important, to 5 which is most important. The concepts were then scored on a scale from 1 to 5, where 1 is least satisfactory and 5 is most. The scores and weights were then multiplied and summed to get the result of the concept scoring.

To get the most comparable scoring result, each team member was first tasked with scoring the concepts individually so that the result could then be compared and discussed within the team. After the best concepts were chosen, the result was then presented to the supervisors as well as product managers to validate the result.

After the scoring was done, the winning concepts were further developed, and the result will be presented later in the report. For the electrical actuator, the most suitable option is chosen based on the space in the module for the specific motion. This is based on requirement that each motor in the actuation concept section has the needed torque in order to drive each movement. Due to implementation and cost advantages one type of rotational motor is chosen for all four movements.

Table 6: Concept scoring matrix categorized by every movement. Winners in every subcategory are shown in bold

Mechanical concept evaluation												
Concepts		**Rotate**		**Tilt**		**Twist**		**Pan**				
		Top drive	Bottom drive	Belt drive with horizontal placement	Bevel gear with vertical placement	Cogwheel with vertically placed motor	Horizontally placed motor	Large cog	Locking with actuator	Simple four step gear - Horizontal	Simple four step gear - Vertical	Lateral complex gear - 16 modes
Criteria	Weight											
Complexity	5	2	4	5	1	2	3	5	3	3	3	1
Height	5	5	5	5	1	1	5	3	2	3	2	1
Width	3	5	5	3	5	4	3	3	3	3	3	3
Aesthetics	3	3	3	3	4	3	3	4	4	3	3	3
Manufacturability	3	4	3	2	4	3	2	5	4	3	2	1
Cost	1	2	2	3	2	3	2	5	4	3	3	1
Ease of installation	3	2	4	4	3	2	4	5	4	2	3	2
Robustness	3	3	3	4	3	3	4	4	3	2	2	2
Cable management	2	4	3	3	3	3	3	3	3	3	4	4
Range of motion	5	5	5	5	5	5	5	5	3	3	3	3
Vibration robustness	5	5	5	4	5	5	5	5	5	5	5	5
Locking ability	5	5	5	4	5	4	5	5	4	3	3	3
Summation		171	182	172	166	153	174	180	138	134	157	106
Rank		2	**1**	**1**	2	2	**1**	**1**	3	4	2	5

As seen in Table 6, the chosen sub concepts are “Bottom drive” for roll, “Horizontal placement” for tilt, “Horizontally placed motor” for twist and “Large cog” for pan. Since all the parts will be 3D printed, they will have some geometries which will require support material. These surfaces will not be as smooth as if they were produced by other methods such as injection molding which might lead to higher friction in between moving parts. Therefore, the torque criteria of the rotational actuator will be of high importance.

Table 7: Actuator scoring matrix. Scoring is made across both categories of motors.

Actuator scoring matrix							
Actuator		Stepper A	Stepper B	Stepper C	DC motor A	DC motor B	DC motor C
Criteria	Weight						
Height	3	2	1	4	5	5	5
Width	5	2	1	4	2	2	2
Torque	5	1	1	1	5	2	1
Cost	1	3	3	3	5	5	5
Locking ability	5	1	1	1	4	2	1
Summation		29	21	45	75	50	40
Rank		5	6	3	1	2	4

As seen in Table 7, one important criterion for the actuation was the locking ability of the motors. The stepper motors all received a 1 because they cannot hold their position if they are not powered, which might create problems if there are for example a power shortage. Using an additional locking mechanism for every stepper motor would make the size and complexity of the concept undesirably much larger and is therefore not considered an alternative.

The size is of the actuator is not important for the functionality of the movement but still of high importance for the final implementation of the product, just like for the mechanical construction. The size of the motor will have a large effect on the possibilities for mechanical design.

Since none of the chosen mechanical concepts included a linear actuator, no further investigation into the solenoids are needed. For the scoring of the rotational actuators it can be seen in Table 7, that DC motor A won by a large margin with high scores in all of the most important criterions. This motor was therefore used as reference when designing the mechanical concepts.

6. Concept realization and presentation

In this chapter, the final mechanical, electrical and software design will be presented thoroughly. In order to illustrate some of the mechanical design assemblies, the color of the parts have been modified and cross sections are used to further explain the most complex assemblies together with highlighted sections. An illustration of the entire assembly is shown in Figure 42.

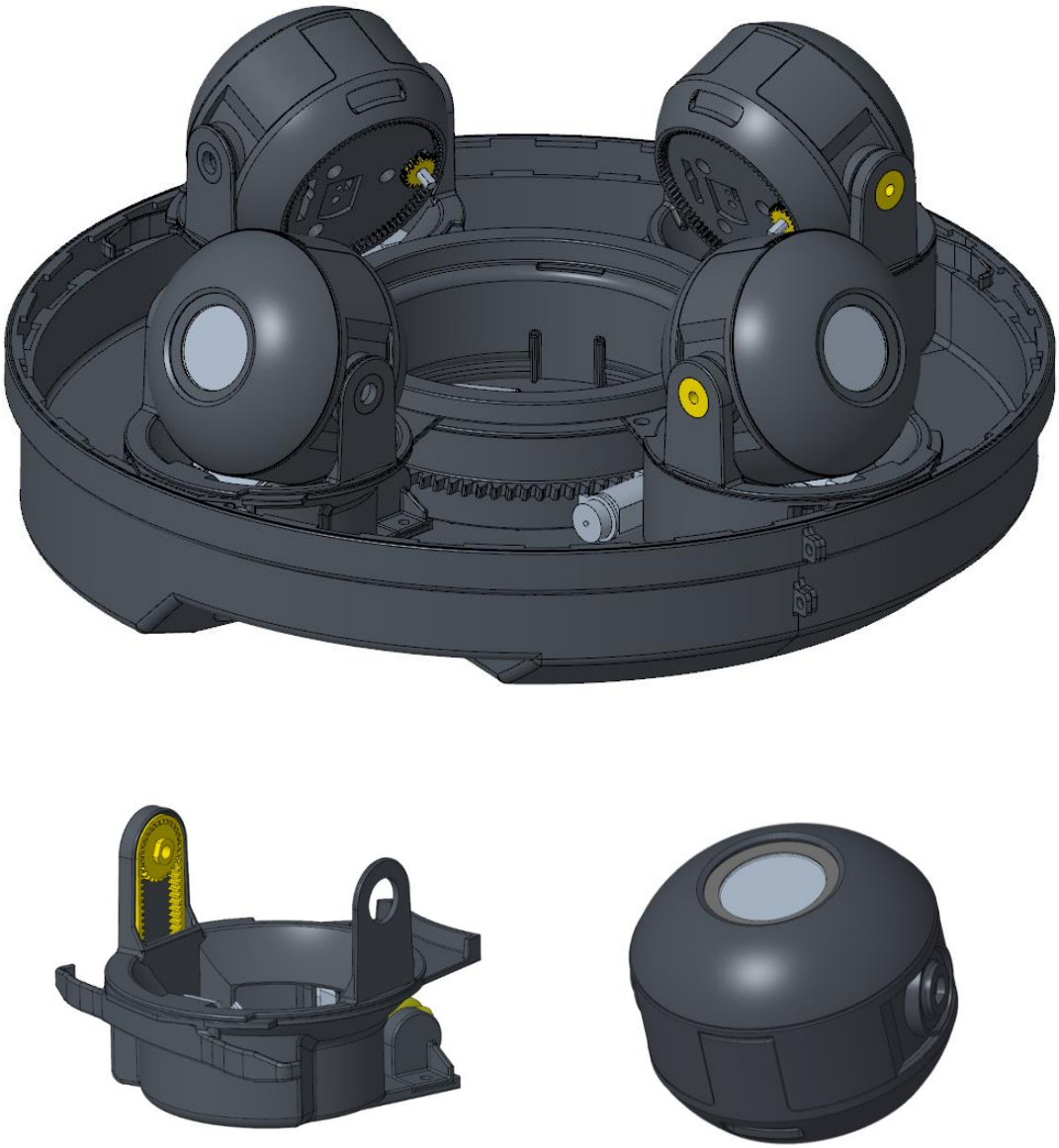


Figure 42: Illustrations of the entire prototype (upper), bottom part of the camera module (bottom left), and the camera housing (bottom right).

6.1 Rotate

The concept chosen for the rotate movement is “Bottom drive”. As can be seen in Figure 43, the motor was placed as close as possible to the sensor in order to achieve the smallest outer housing of the camera module.

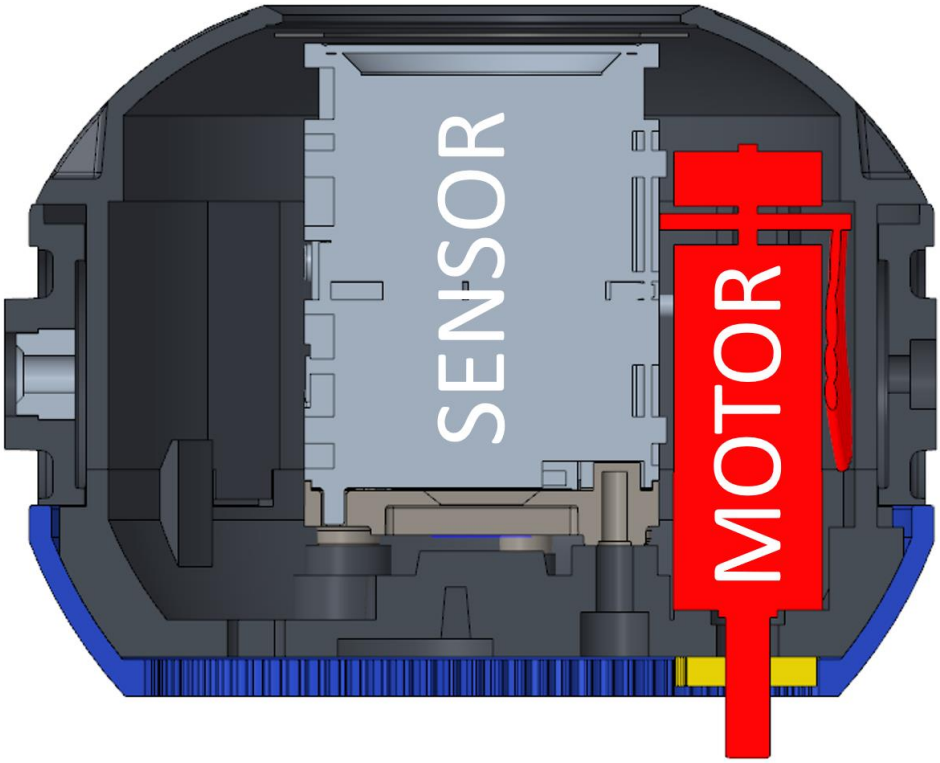


Figure 43: Cross section of the camera housing module with sensor and motor.

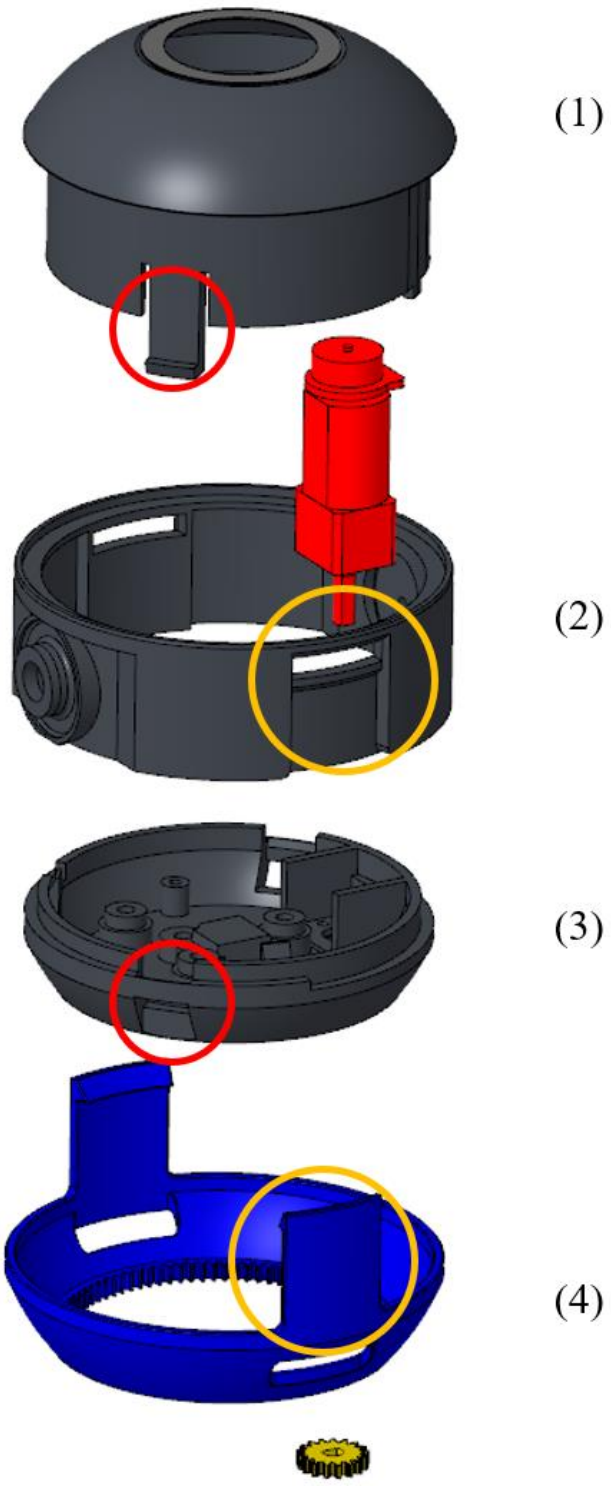


Figure 44: Exploded view of the camera module. Every part is numbered for ease of referencing. The highlighted sections illustrate the two different snap pairs.

The basic functionality of the concept is a sandwich design, where Parts 1 and 3, shown in Figure 44, are fastened with two cantilever snaps and Parts 2 and 4 are fastened with the same principle. This facilitates for movement between the two part pairings as the pair can move relative to each other because of the outer cogwheel placed in the bottom of Part 4 in Figure 44. The motor, which is fastened to the bottom of Part 3 drives the movement. Due to patent application reasons, the fixation of the cogwheel cannot be visualized and is therefore suppressed from the illustration.

6.2 Tilt

As for tilt, the concept chosen was to use a belt drive fitted to the chosen DC motor. In order to get a solution as compact as possible the motors for tilt, twist and pan were placed in the same plane which will be further explained in responding subsections. The camera module is fixated by two screws at the highlighted sections in Figure 45, and is therefore rotated as the belt is driven.

The choice of belt was crucial to make the tilt movement as slim as possible. The belt was ordered from a supplier which is used frequently at the company. Since the motors produce a high amount of torque, the decision was made that the belt should not be as small as feasibly possible, but rather wide enough for the torque transfer to be made possible without damaging the cogwheels.

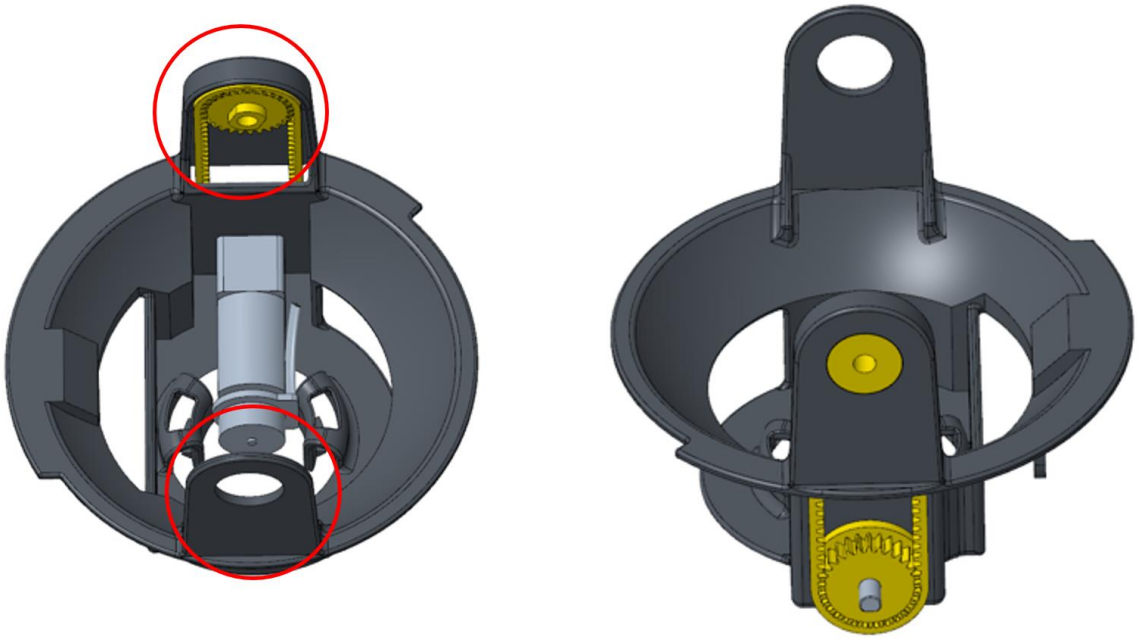


Figure 45: Motor for twist fitted in the center of the module which is connected to a belt drive to the upper cogwheel. The highlighted section shows where the camera module is fixated.

6.3 Twist

In order to achieve the twist movement, the bottom part of the camera module had to be built using two parts which can move in relation to each other. The upper one is displayed in Figure 45 and the lower one below in Figure 46. The motors for tilt and twist are placed in the same plane, illustrated in Figure 47. In Figure 48, two additional cantilever snaps can be seen in the middle of the bottom of the camera module which allows for secure fastening of the two parts while still making the twist movement feasible. To guide the twist movement, both parts have spherical surfaces which are highlighted in red in Figure 48.

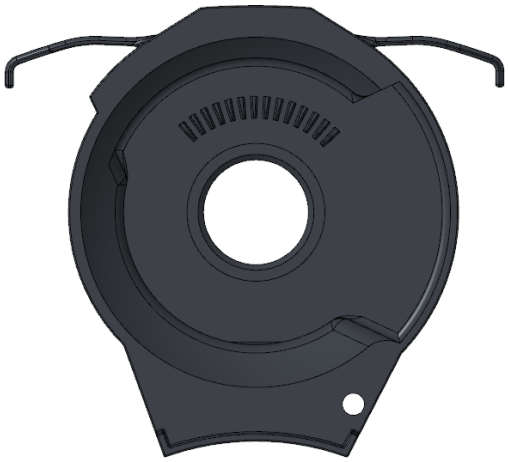


Figure 46: Lower part of the bottom of the camera module. The cog which makes the torque transfer possible can be seen in the bottom of the part.

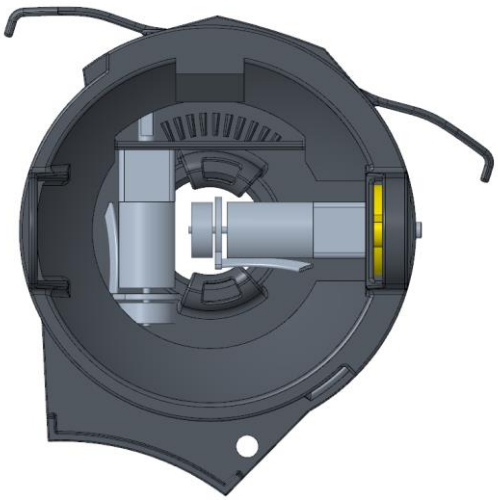


Figure 47: The bottom of the camera module as seen from above with the motor for twist to the left. The cog which is fitted to the lowest part in the bottom, connects to the cogwheel on the twist motor.

Due to a patent application, the transmission between the motor for twist and the cog in the lower part of the bottom of the module cannot be revealed and is therefore suppressed in the illustration.

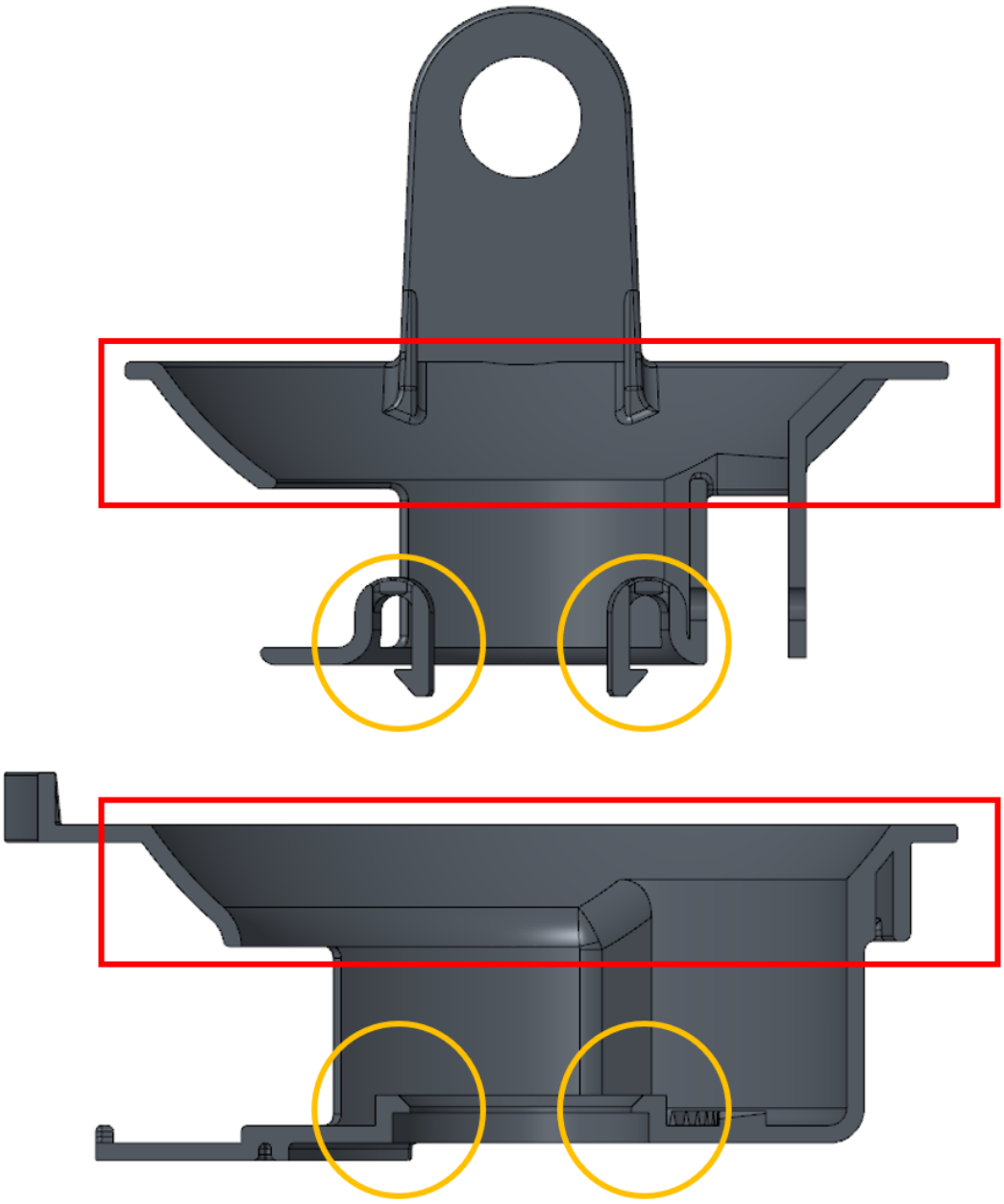


Figure 48: Cross section of the bottom of the camera module. The red highlight is the two spherical surfaces where the two parts have contact. The yellow highlight is the two cantilever snaps with corresponding features in the bottom which fasten the two parts.

6.4 Pan

The concept chosen for the pan movement was called large cog which fundamentally means that each module travels along its path by individually driving a cogwheel which is connected to a large cogwheel placed in the center of the entire camera. As stated before, all the motors for tilt, twist and pan were placed in the same plane which meant that the design of the cogwheel had to differ from what was presented in the initial concept. A worm gear was fitted to the motor which made it possible to place the motor horizontally. As can be seen in the next subsection, the worm gear is the best solution for this application since the pan movement is the most torque demanding movement as the entire module must be moved.

As for all the other motors, the motor was secured to the structure by simply fastening it with a set of screws. Since the transmission built into the DC motor made the torque substantially larger than necessary it was essential that each motor was properly fastened. When testing an early iteration of the concept it was discovered that the torque from the motor was too large for the 3D printed prototype which caused it to flex which led the worm gear to be dislocated from the large cogwheel. In order to fix this, a plastic component was added at the end of the worm gear in order to fixate it properly, see Figure 50.

Another important aspect of the pan movement was the placement of the large cogwheel in the middle, which each module uses for the movement, see Figure 49. The parts that made this possible will be presented below in the chassis section.

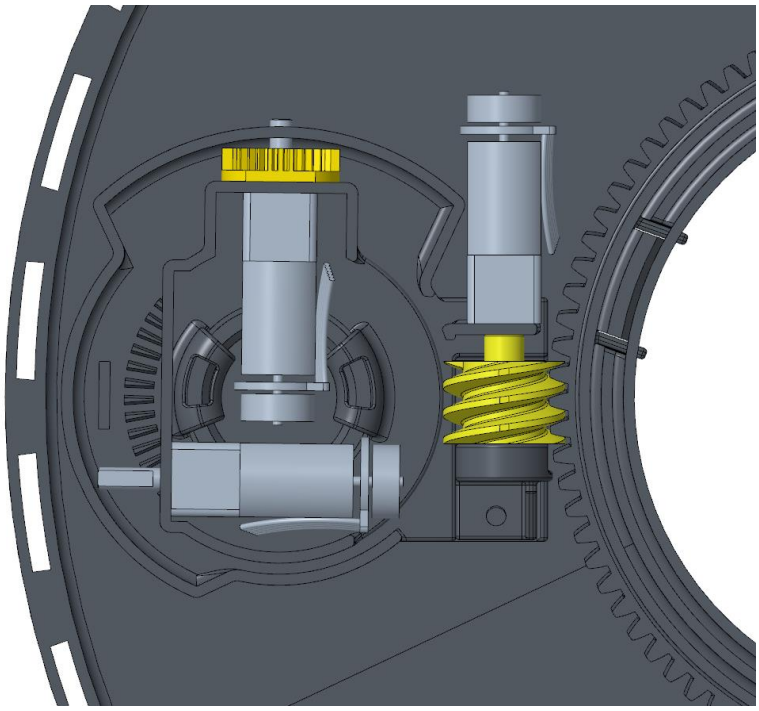


Figure 49: Cross section which shows the motor for the pan movement with the worm gear.

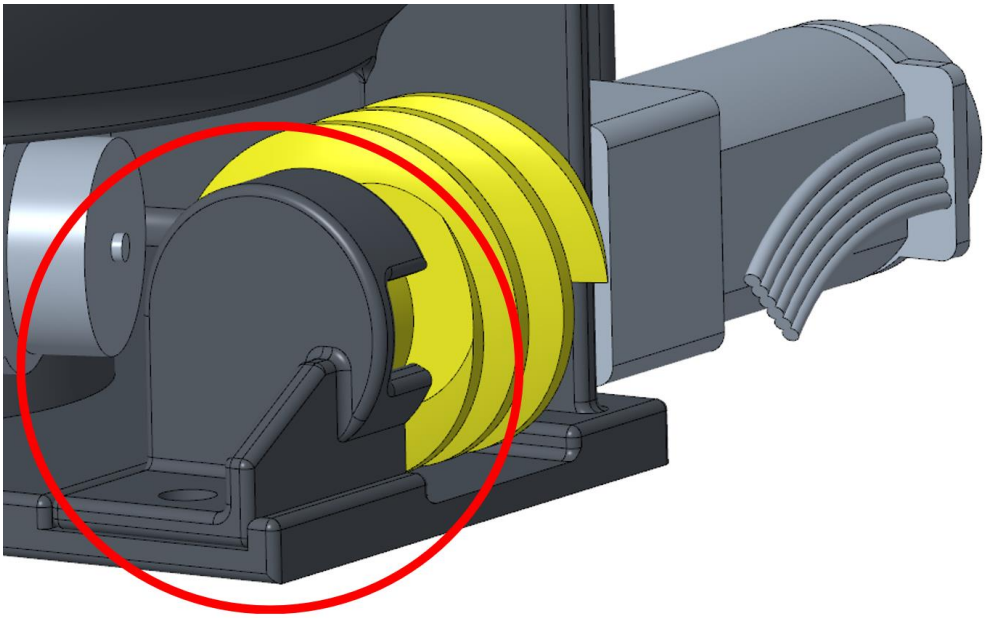


Figure 50: The small part introduced in order to secure the worm gear in the pan movement. A small cut-out had to be introduced to remove the risk of interfering with the larger cogwheel.

6.5 Gear ratios

In every movement there is some sort of torque transferring measure which in most cases involves gear ratios. As seen in the theory section of the torque transferring methods, the gear ratio decides the speed of the rotation which is essential to the concept since the position of the motors are time dependent. This will be explained further in the Arduino section of the report.

The gear ratios in the concepts are summarized in Table 8 below.

Table 8: Gear ratios of all the movements.

Movement	Pan	Twist	Tilt	Rotate
Gear ratio	44	3.25	1	5.30

6.6 Chassis

In order to make each camera module able to move along the pan track a chassis had to be constructed. Apart from holding the modules in place, the chassis was used to hold the PCB for the video streams in place.

The center part of the chassis, marked as (2) in the figure below, holds the large cogwheel which is used for the pan movement. To impede rotation, eight features were added, seen in the yellow highlights on the middle part in Figure 51, with corresponding cuts in the bottom and top parts which fixates the part.

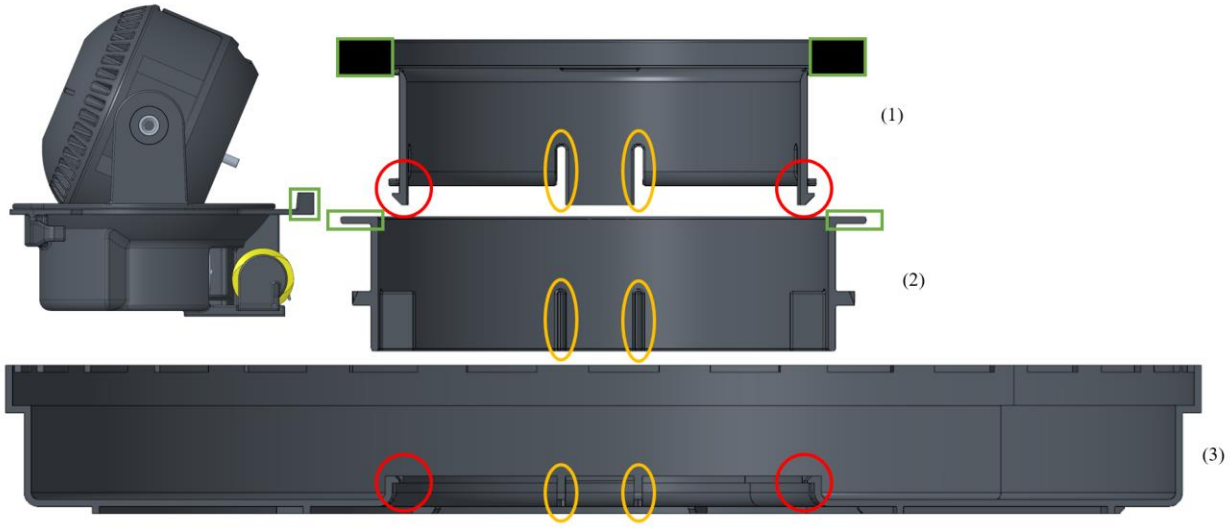


Figure 51: Cross section of the chassis. The red highlighted sections show the cantilever snaps with corresponding features in the bottom. The yellow highlights illustrate the guiding feature which keeps the three components from rotating in relation to each other. The green highlights show the contact surfaces on the chassis and the camera module.

The part that is held in place by another part, marked as (1) in the figure, which is snapped in place by four snaps which hold the entire center puck in place in the x, y, and z direction, see the red highlights in Figure 51. The same part which holds the cogwheel also supports the inner edge of the camera module in the z direction as well as guiding the modules along its path, see the green highlights. Due to patent application reasons, this feature cannot be shown as an illustration and is therefore blacked out in the figure.

Since the camera can be placed with the cameras pointed downwards, the outer part of the chassis had to secure each module in the z direction. This was made possible by adding small ridges with cutouts under for manufacturability, which can be seen in the outer part of the chassis in Figure 52. Each module has wings fitted on the outer side which function is both to prevent the modules from falling when placed upside down as well as creating radial force which keeps the worm gear connected to the centrally placed cogwheel, displayed in Figure 52.

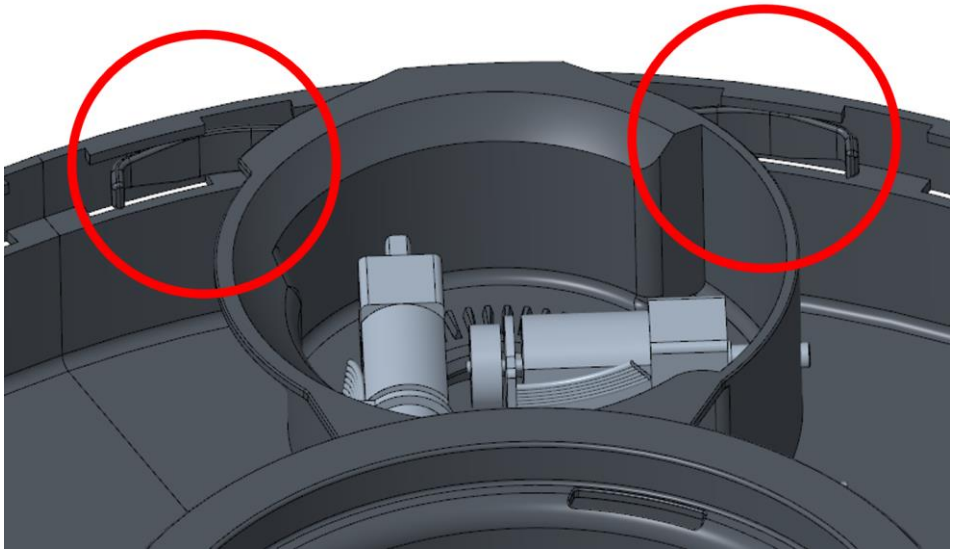


Figure 52: Illustration of the wings which secure the modules in the z direction.

To prevent contact between the modules and the PCB which had to be placed somewhere in the structure, another part was added to the chassis. This part has no function but to hold the PCB in place and to route the cables for easy cable management. The part is shown in Figure 53. To make it possible to print the parts in the 3D printer at the company, both parts that together make up the chassis had to be split in the middle. It is not intended for these parts to be split in the final prototype.

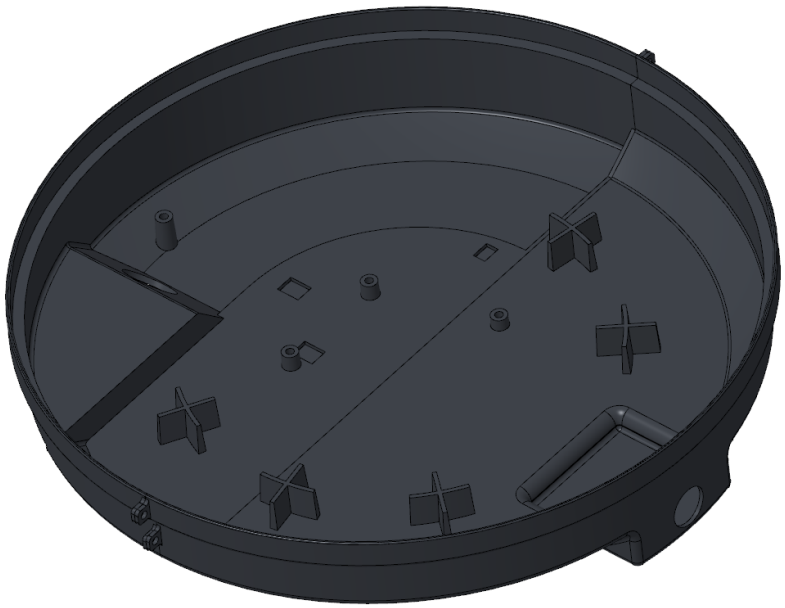


Figure 53: Bottom part of the entire prototype which houses the PCB for the video feed.

6.7 Circuit board

In this chapter, the prototype circuit board, created to control the mechanical prototype, will be presented together with explanations to the design decisions that have been made during the development process.

Initially, an old PCB, which was compatible with the motors used, was utilized to verify the motion of motors. It worked well for testing a single motor, but when the number of motors increased, the need for a more tailor-made control system arose to be able to control all the motors with one interface.

A circuit board was therefore developed to be able to control all the motors used in the physical prototype. An Arduino was used as the main controller for this together with an Ethernet Shield for communication between the computer and the Arduino. Ethernet was chosen over serial communication to make the control system more realistic, since the sensors communicate over Ethernet. To allow for bi-directional rotation for every motor, 16 H-bridges were used. This was controlled through I2C communication since it decreases the number of wires needed in the circuit from 16 to 4 per module. Some problems related to the addressing of the H-bridges arose during the development of the board which led to a need for more components such as the I2C buses, P-MOSFET, and a level shifter, seen in Figure 54. This will be explained further under Section 6.7.2.

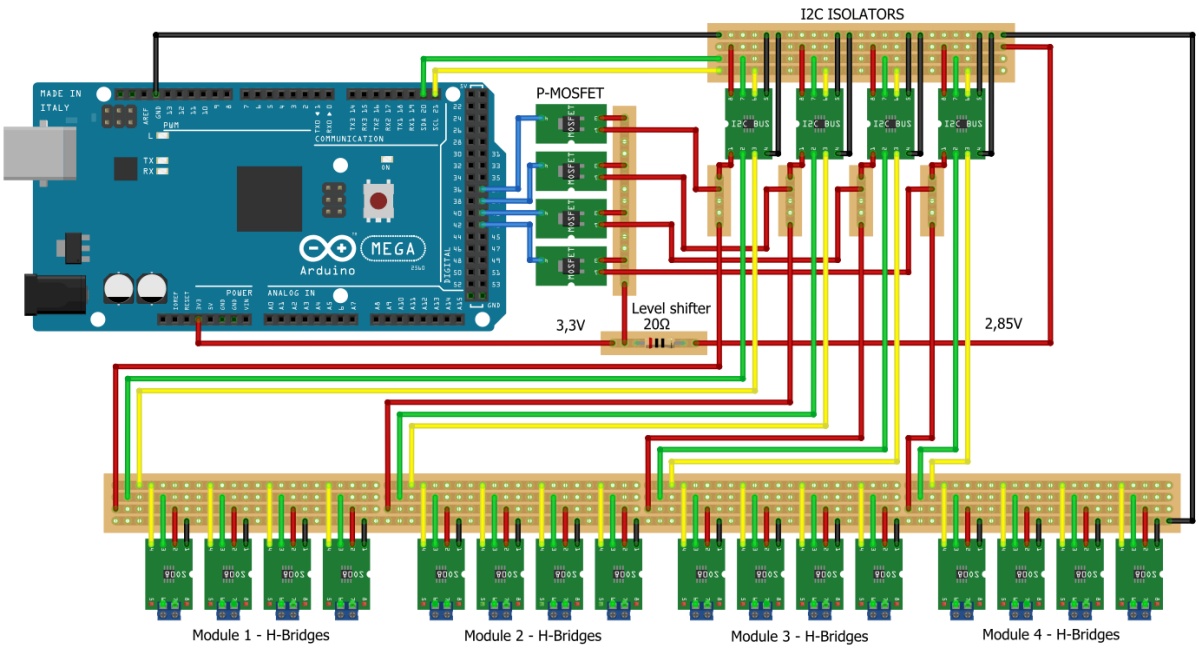


Figure 54: Schematic of the final prototype board implementation.

Figure 55 shows the final prototype board. All the components were integrated into a 3D printed chassis to make the wiring easier and more secure to take away any risk of bugs related to tangled or disconnected wires. Power was initially supplied by an external power supply so that the Arduino was not at risk of being destroyed due to high currents. After testing, it was discovered that the Arduino would be able to power the motors by itself, if driven sequentially. This means that the user can connect it to either

an external power supply or through the Arduino depending on use case. The wires can be connected to the H-bridges in the board from the outside as seen in the highlighted red section in Figure 55.

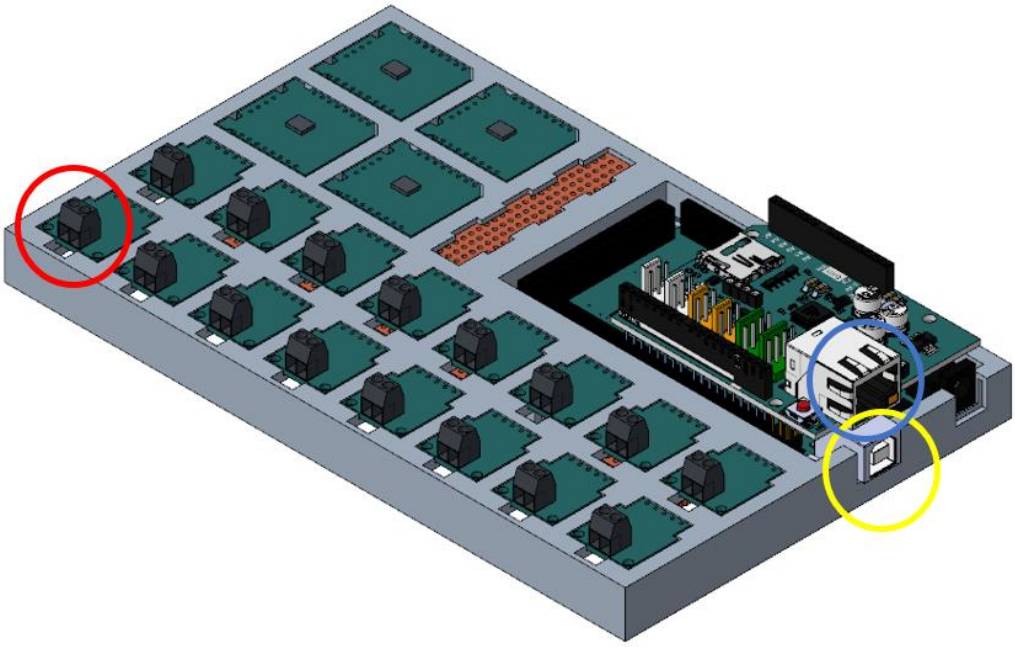


Figure 55: 3D model of the finished prototype board. Connections for one of the 16 H-bridges, highlighted in red, the power supply in yellow, and Ethernet in blue.

6.7.1 Arduino and Ethernet Shield

An Arduino Mega 2560 was used together with an Arduino Ethernet Shield 2. This board had all the integrated peripherals needed, such as I2C and Ethernet communication. A smaller Arduino could have been chosen, since most of the of the I/O ports were not used. But since the prototype circuit board would not be integrated into the camera structure there was no need to make it as small as possible. It was considered better to have a flexible board with a lot of peripherals for different solutions. The peripherals used in this project was SDA and SCL for the I2C communication, the Ethernet port for Ethernet communication and four I/O ports to switch the MOSFETs.

6.7.2 Components

The H-bridges were ordered as integrated circuits to make the soldering process easier. The chips on these circuit boards were of the same type as H-bridges used in combination with the chosen motor in other projects at the company. This ensured that the components would work well together. The only disadvantage with the chips was that they only had four unique I2C addresses available, where 16 would be needed to drive the motors individually. This was solved using four I2C isolators, called buses, and four MOSFET transistors, see Figure 54, to switch the power and subsequently the communication lines to the motors. The software written only allowed one bus to be active at a time which gave the combinatorial commands listed in Table 9.

Table 9: Combination table of motor drive with Active I2C busses and H-bridges addresses.

States	I2C adr. (0xC0)	I2C adr. (0xC2)	I2C adr. (0xC6)	Rotate adr. (0xC8)
I2C bus 1	Pan module 1	Twist module 1	Tilt module 1	Rotate module 1
I2C bus 2	Pan module 2	Twist module 2	Tilt module 2	Rotate module 2
I2C bus 3	Pan module 3	Twist module 3	Tilt module 3	Rotate module 3
I2C bus 4	Pan module 4	Twist module 4	Tilt module 4	Rotate module 4

The I2C buses were controlled through the microcontroller by switching an I/O pin connected to a MOSFET transistor that would connect or disconnect the supply voltage to the specified bus and associated H-bridges. In this way all four motions could be controlled for the four modules. To make it intuitive and more user friendly the four H-bridges connected to a bus were grouped as a module with pan, twist, tilt and rotate.

The level shifter was put in place to allow the Arduino to switch the communication lines in time for the bus to read the message. In testing, it worked when connecting up to 3 buses on the Arduino, but when connecting the fourth, no communication could be transmitted. This was due to the internal resistance of the buses that accumulated on the SDA line increasing the time it took for the Arduino to pull the line low. Since the internal resistance on the prototype boards could not be changed, the only way to save time when pulling the line low was to lower the reference voltage. This was tested with an adjustable power supply where the breaking point was discovered to be 3V. To have some margin of error the 3.3V supply was shifted down to 2.85V.

Since the motors were rated for 3.3V, the lower side of the I2C bus, which was connected to the H-bridges, still had to be connected to 3.3V. Generally, I2C communication should be transmitted on the same voltage to work properly, but since the bit change threshold is at 70% of the supply voltage everything down to 3.31V would be considered a high bit by the bus, allowing it to translate 2.85V bits to 3.3V bits.

The power switching to the buses created a lot of problems in the initial stages of the development. Due to the interconnection between the supply voltage and the SDA/SCL line, there was a big risk of backwards current if the lines were not correctly isolated. This would lead to burnt transistors and disbursed voltages over all the lines making transmission impossible. The solution to this was to connect the lower side of the I2C bus and the associated H-bridges to the same transistor. By doing this, it was ensured that the communication line would be low until the power was switched on and pulled it high to allow communication from the Arduino to come through.

6.8 Software implementation

In this chapter, the software of the microcontroller, and Graphical User Interface (GUI) as well as the communication between the different programs will be presented. Since all the code will be implemented in the Appendix, this section acts as an explanation of the functions of each program.

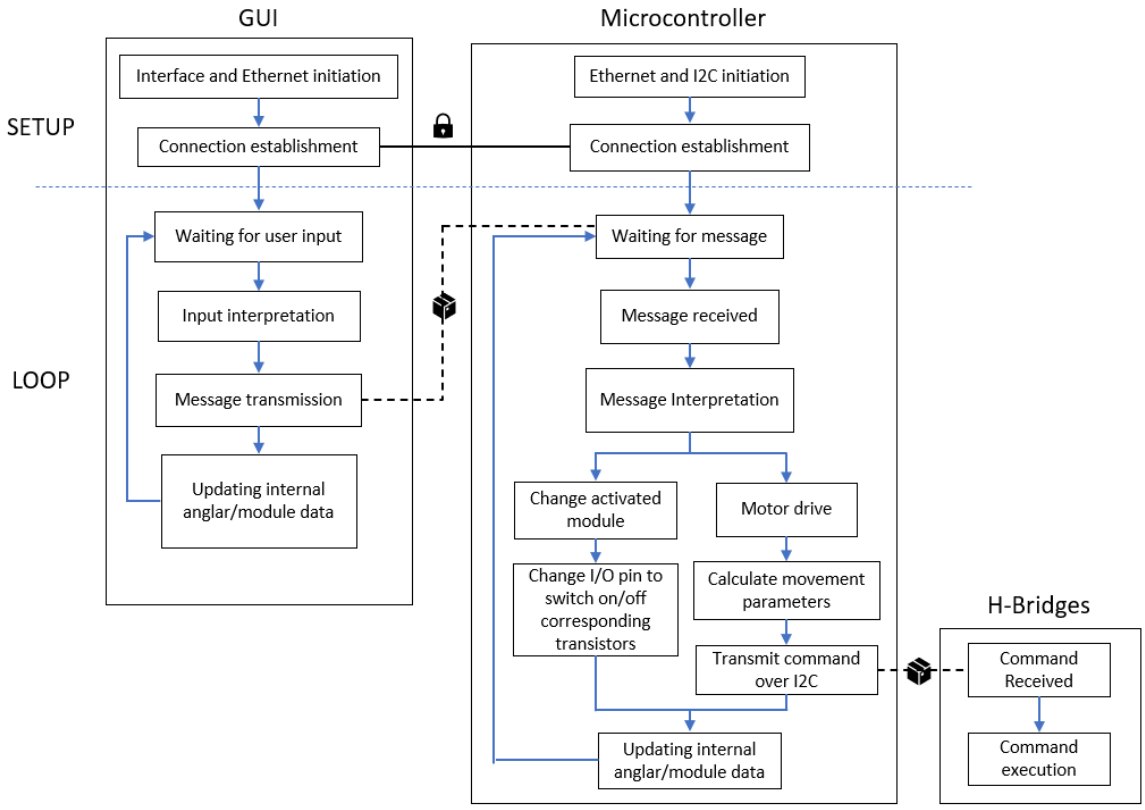


Figure 56: Flowchart of the software implementation used in the GUI and Microcontroller.

When the GUI application is started on the computer, it initiates its setup protocol. If the microcontroller is connected correctly it will try to make a connection. When the connection between the GUI and the microcontroller is established the program is ready to be used. The simplified sequence of events can be seen in, Figure 56, under "LOOP". Initially nothing will happen, The GUI is waiting for user input while the microcontroller waits for a message from the GUI.

As the user gives an input to the program it will first interpret the input, put it in a User Datagram Protocol (UDP) packet and send it to the microcontroller. The microcontroller will detect that a message has been received and read it. It will interpret what the user requested, either change module or drive a motor and then act accordingly. If the message was to switch modules, the controller just switches the desired I/O pin connected to the transistors. If the message was to drive a specific motor, the controller will first calculate the change in angle and time needed for the movement based on the gear ratio and then transmit it over I2C to the correct H-bridge for execution. All the updated variables will be saved locally on both sides before the program loops back to the start, ready for new inputs.

6.8.1 Interface

The user only interacts with the program through the interface and the input that can be sent to the microcontroller is limited to the inputs integrated into the interface. When the application is started the interface looks like the illustration in Figure 57.

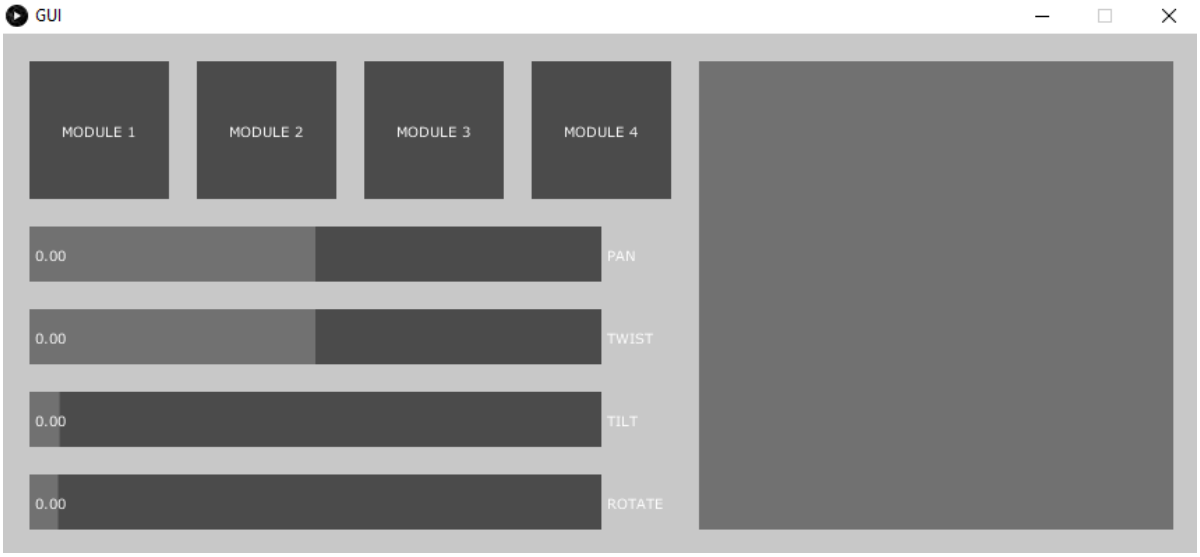


Figure 57: The GUI when booted up. No module is selected, and all movements are at the default position.

The interface allows the user to control each of the 16 movements in total, with an easy overview of the movements as well as the current position of each of the motors. The communication method makes it possible to send messages between the Arduino and the GUI and it was therefore decided to add a window that displayed simple feedback messages to the user. This was also utilized as a debugging tool during the development process.

Figure 58: When the module is selected, the button is highlighted, and confirmation text is shown in the window.

The top four boxes can be toggled on or off in order to select which module the user wants to control. When the box is pressed, it changes color to red which indicates that that module is now activated and can be controlled. At the same time, a confirmation message is shown in the window to the right. A module can be deactivated by either toggling the box which is activated or by selecting and activating another module. An idle module is always shown in grey. When no module is activated no values are sent to the Arduino nor updated on the GUI side.

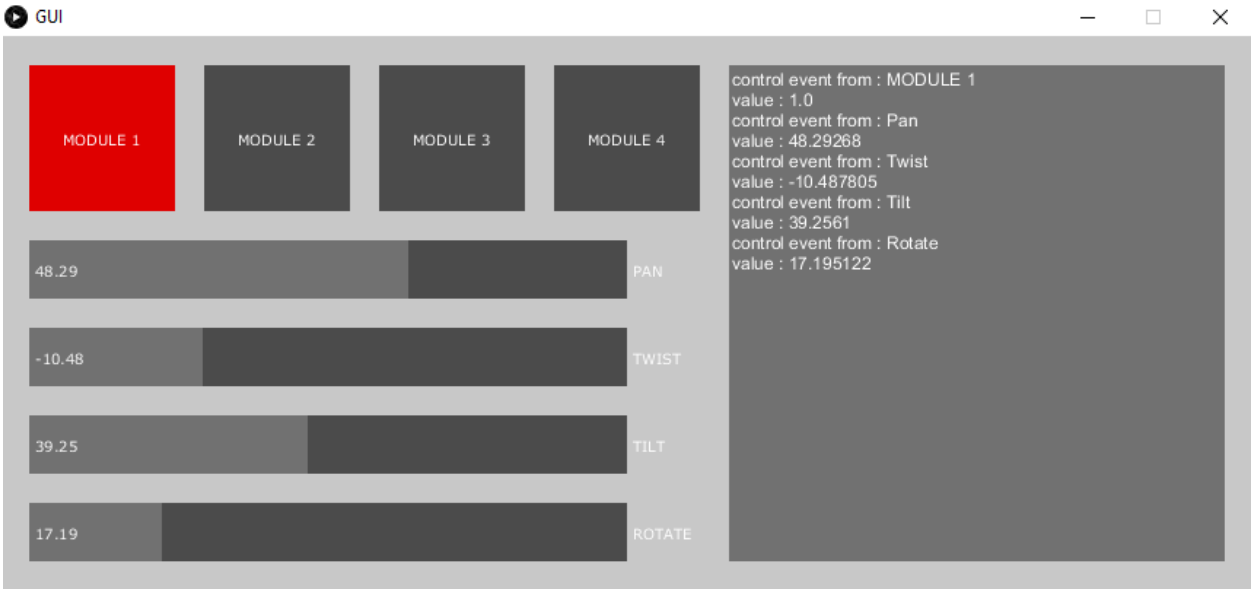


Figure 59: The sliders are modified and updated; confirmation text is shown in the window.

When the module is selected and activated, the four motions of pan, twist, tilt and rotate can be controlled. Local values are stored in both the Arduino and the GUI program which keeps track of the current position of the motors. When the slider for a motor is moved, the values in the slider to the left are changed and stored on both sides. When one module is activated and has had its positions changed and another module is selected, the sliders are updated to the local values stored which makes the toggling experience fluid and correct, see Figure 59.

6.8.2 Communication between GUI and Arduino

As stated previously, the communication between the GUI and Arduino is done over Ethernet through UDP. Each packet which is sent from the GUI has a predefined format illustrated in Figure 60 below.

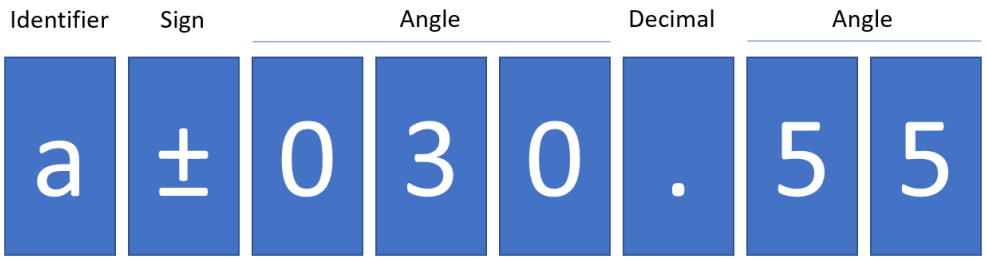


Figure 60. Illustration of the packet sent from the GUI to the Arduino.

The first bit contains an identifier which refers to which action is sent from the GUI. This action can either be to activate a module which means that the packet contains '1', '2', '3' or '4' for each module, or a longer packet for each motion. It was decided that the identifier 'a' corresponds to pan, 'b' twist, 'c' tilt and 'd' rotate.

When the packet is read, and the content starts with a number, nothing is done but activation or deactivation of the corresponding module. A local Boolean on each side of the communication keeps track of the toggling of each module. If the packet is read and the first character is a letter, then the next character is read which contains which sign the inputted angle has, since most of the movements have a range which overlaps zero. The next three characters are then read to determine the amount of degrees the motor is to be turned. The characters are multiplied with its corresponding power of ten to be then added together.

When the next character is read, which will contain the decimal point, the reading is interrupted. This is because it was decided that, due to the lack of resolution, decimals would be excluded from the angles.

When the entire packet has been read, the motor which is to be driven is activated, the current position of the motor is read, the new angle is calculated by comparing the old angle to the new and the time of which the motor should be driven is calculated and in which direction. The Arduino then waits for another packet.

7. Discussion

It can be seen in the concept scoring matrix that all concepts that included some sort of gear synchronization and linear actuation scored poorly which can be explained by the sheer complexity of engaging, disengaging and synchronizing gears. Since the goal of the master thesis was to produce a working prototype that fulfills the specifications stated in Chapter 3, it was decided that the metric complexity was to be a deciding factor when weighing the concepts. It can be discussed that some concepts were removed even though great promise of their working principle was shown, but as previously stated the final prototype was to be constructed.

The general design philosophy which has been followed in this thesis has been to create a working prototype which will fulfill all the specifications stated in Chapter 3. This approach has meant that most of the components that have been created are modified in such way that even though the components are made through 3D printing, the specifications of are still met. Therefore, most components must be redesigned for the processes of which the real product will be produced by.

One solution that has not been investigated in this thesis is the positioning of the camera modules. It was discussed early in the project whether it was necessary to fit sensors that were able to detect when the motors have driven the module close to its max position. It was decided that, since the remote drive of the entire camera is for installation purposes only, this will introduce unnecessary and undesired complexity and cost to the prototype.

To solve this in an alternative way it was decided that the overcurrent detectors on the H-bridges would be used in combination with the mechanical stops for each motion. As the limit on the overcurrent detectors was fixed it meant that the combination of brittle plastic components and over-dimensioned motors resulted in that the motors tore the prototype apart before the current passed its limit. Even though this method could not be tested and realized, it should work in a final prototype where the over-current detectors limit could be altered and where the parts are produced through injection molding and much more durable.

The final solution is simplified but functional. As stated before, each motor must be connected to the prototype PCB which has been constructed in the project rather than small PCBs on each camera module. This means that two cables must be used for each motor just for the drive. The motors that are used in the prototype are fitted with encoders which can be used to determine how many rotations the motors have turned during each movement, but since the amount of cables will be substantially increased, it was decided that the motors will be ‘blind’ which means that their position will be controlled by the time of each movement, i.e., in open loop. The position of each motor will be saved after the movement has been made which means that there will be some uncertainty of the exact position of each motor.

When discussing the specifications of the twist movement, it was concluded that each module was to be able to rotate enough for two modules’ images to overlap. Since each sensor has a field of view which is rather wide, the prototype constructed has a rather big overlap of the pictures when the modules are placed as close as possible and when the twist movement is fully utilized. When comparing the prototype to multidirectional cameras on the market, the footprint of the prototype is larger because of this reason. It

can be discussed whether the twist movement specification must be as large as stated in the specification list since it has a great influence on the outer dimensions of the prototype.

As the prototype was benchmarked, it did not resist vibrations as well as was thought. It was concluded that this was because of the manufacturing process of which the cogwheels were made by, 3D printing. Since the material is rather brittle and fragile, after using the motors for some time, the cogwheels were worn down which created backlash in the transmission of each movement. As the cogwheels will not be constructed of 3D printed plastic in the final prototype, it was decided that vibration test could not be conducted on the prototype.

Due to company secrecy, the overall dimensions of the prototype cannot be presented as well as the exact amount of movement of every direction. Without disclosing this, it can be said that the requirements were met. After producing all components and assembly of all the parts it was concluded that the prototype fulfills all the specifications stated in Chapter 3. System Specifications, apart from the vibrations test criteria which could not be performed.

It can be discussed whether the pan movement is fast enough, but since the movement are for installation purposes it should be sufficient. If not, an alternative could be to solve it by using a worm gear with more entries making the movement faster.

The electrical implementations caused a lot of problems during the development. In hindsight, the electrical system could have been solved in a simpler way to avoid some of the switching problems that occurred, with dispersed voltages over the communication line and burnt transistors. The lack of addresses on the H-bridges could have been solved with an I2C multiplexer, which is a device designed to handle slaves with the same address, essentially giving them new addresses from the Arduinos' perspective. This would have completely removed the need for power switching allowing the communications lines to be pulled high at all time. The reason for not switching over to a multiplexer was that it would still be a problem with the internal resistance on the multiplexer and since the initial concept worked with some minor changes to the voltage levels, it was decided that it would be sufficient enough since it is a prototype board.

8. Conclusion

The project was successful in creating a working prototype of a multidirectional camera with remote functionality for pan, twist, tilt, rotate. The functionality shows great promise for future development, even though some aspects, discussed in the discussion chapter, needs to be investigated further for optimal design. Some key aspects for further work are presented below:

General design of all components to fit injection molding instead of 3D printing
As stated in the previous chapter, all the components that have been designed in this thesis have been constructed through 3D printing which is known will not be the manufacturing process of which the final prototype will be manufactured by.

Use and design cogwheels which are not 3D printed
To be able to do many iterations during a limited amount of time, most of the cogwheels that have been created in this thesis have been designed to be producible through 3D printing.

Design and build small PCBs which are mounted on each module
Since all of the electrical components which are used in this master thesis are either built-on prototype cards or fully external, as the Arduino, the size of the components meant that a separate PCB was created instead of using the main PCB of the camera streams. It has been stated in the previous chapter that it would be beneficial for the H-bridges and the I2C buses to be located on each camera module instead of on the larger PCB which would lessen the amount of cabling in the camera.

Investigate if it is possible to fit smaller motors
The reasoning behind the choice of motors was mainly to get enough torque to be able to drive each motion even though each part has been made through 3D printing. In the final prototype, the amount of torque needed will be substantially lower which means that smaller motors can be used instead of the over-dimensioned ones which are used in the prototype. This might mean that the final prototype will be much smaller than the current prototype, manufactured in 3D-printed plastic.

Cable management
There has been some thought put into the cable management of the camera, especially since there are a large amount of motors fitted in the camera. Though, as previously stated, designing and fitting of a smaller PCB on each module means that the cabling would be entirely different on the final prototype. Therefore, there must be much work put into the cable management, mostly because there might be a risk of pinching as the modules travels along the pan track.

Further testing

As stated in the previous chapter, vibrations test could not be performed. To make sure that the prototype can withstand the vibrations needed, cogwheels and all other parts should be manufactured in another material than the plastic used in this prototype.

Cost analysis

Since the camera has been fitted with no more than 16 motors, a cost analysis of the prototype is essential. This has not been done in the master thesis but might be crucial to determine if the product is worth investing more into.

Bibliography

Agarwai, T. (n.d.). *Different Types of Transistors and Their Functions*. Retrieved 04 08, 2021, from Elprocus: https://www.elprocus.com/different-types-of-transistors-and-their-functions/

Brockotter, R. (n.d.). *Key design considerations for 3D printing*. Retrieved 04 08, 2021, from 3D HUBS: https://www.3dhubs.com/knowledge-base/key-design-considerations-3d-printing/

Bunnell, B., & Najia, S. (2020, July 13). *Skill Builder: Learn The Types Of Gears*. Retrieved 04 08, 2021 from Make: Community: https://makezine.com/2020/07/13/skill-builder-learn-the-types-of-gears/

Childs, P. (2019). *Mechanical Design Engineering Handbook.* Butterworth-Heinemann.

Circuit globe. (n.d.). *Working Principle of a DC Motor*. Retrieved 04 08, 2021, from Circuit globe: https://circuitglobe.com/working-principle-of-a-dc-motor.html

Creative Mechanisms. (2017, 03 15). *3 Basic Steps of the injection Molding Process*. Retrieved 04 08, 2021, from Creative mechanisms: https://www.creativemechanisms.com/blog/basic-steps-of-injection-molding-process

DSM. (n.d.). *DSM*. Retrieved 04 08, 2021, from Radii and chamfers: https://www.dsm.com/engineering-materials/en_US/design-guide/design-guidelines/radii-and-chamfers.html

Electronics Hub. (2018, 02 02). *Basics of I2C Communication | Hardware, Data Transfer, Configuration*. Retrieved 04 08, 2021, from Electronics Hub: https://www.electronicshub.org/basics-i2c-communication/#Data_Transfer_Protocol

Fiore, C. (n.d.). *Stepper Motors Basics: Types, Uses, and Working Principles*. Retrieved 04 08, 2021, from MPS Electronics: https://www.monolithicpower.com/en/stepper-motors-basics-types-uses

H2W Technologies. (2015, 10 22). *Voice Coil Actuators vs. Solenoids: What is the difference?* Retrieved 04 08, 2021, from H2Wtech: https://www.h2wtech.com/blog/voice-coil-actuators-vs-solenoids?gclid=CjwKCAiAq8f-BRBtEiwAGr3DgekZ735KuJuCyLbI7_WpbgJ4IUxJxbHEuoMBXYkvPE0er9tQ4wx_HBoCX8cQAvD_BwE

Haddad, E. (n.d.). *DFA - Simple Mechanical Assembly Techniques for Plastic Parts*. Retrieved April 8, 2021, from Aire Plastics: https://www.aireplastics.com/dfa-design-for-assembly/

Horne, R., & Hausman, K. K. (n.d.). *Designing Parts for 3D Printing*. Retrieved April 8, 2021, from Dummies: https://www.dummies.com/computers/pcs/printers/designing-parts-3d-printing/

Hunter, K. (2020, 08). *How to Prototype Electronics - Arduino vs PCB*. Retrieved 04 08, 2021, from Microtype: https://www.microtype.io/how-to-prototype-electronics/

Ida, N. (2014). *Sensors, Actuators, and their Interface - A Multidisciplinary Introduction.* New Jersey: SciTech Publishing.

James, L., & Granath, E. (2020, 02 27). *BASIC KNOWLEDGE - MOSFET VS. BJT*. Retrieved 04 08, 2021, from Power & Beyond: https://www.power-and-beyond.com/whats-the-difference-between-mosfet-and-bjt-a-909006/

Kohara Gear Industry. (n.d.). *Type of Gears.* Retrieved April 8, 2021, from KHK STOCK GEARS: https://khkgears.net/new/gear_knowledge/introduction_to_gears/types_of_gears.html

Liu, C., Bao, X., Meng, Q., Xu, C., & Liao, T. (2019). A Flexible Hardware Architecture for Slave Device of I2C Bus. *International Conference on Electronic Engineering and Informatics.* Wuhan. Retrieved 04 08, 2021

Malloy, R. (2010). *Plastic Part Design for Injection Molding : An Introduction.* Munich: Carl Hanser Verlag.

Modular Circuits. (n.d.). *H-Bridge - the Basics*. Retrieved 04 08, 2021, from Modular Circuits: https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/

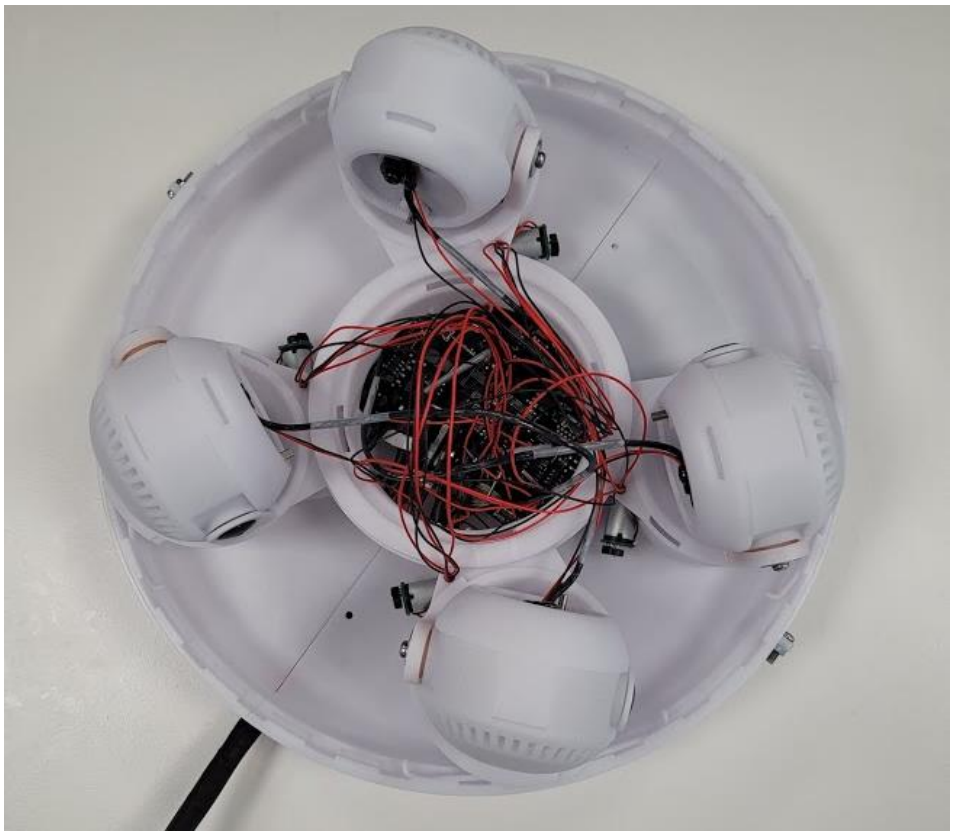
Tres, P. A. (2017). *Designing Plastic Parts for Assembly.* Munich: Hanser Publications.

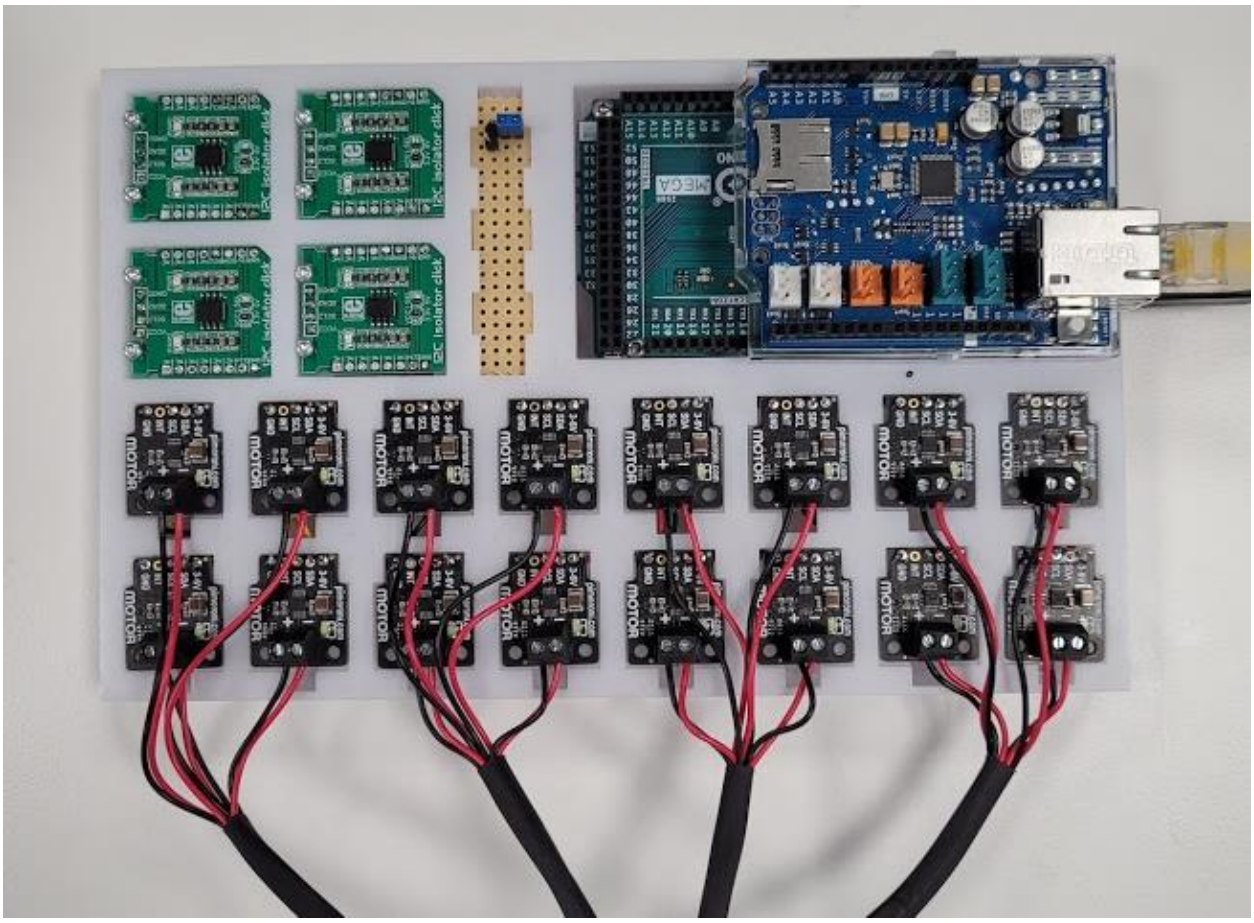
Ulrich, K. T., & Eppinger, S. D. (2016). *Product Design and Development, Sixth Edition.* New York: McGraw-Hill Education.

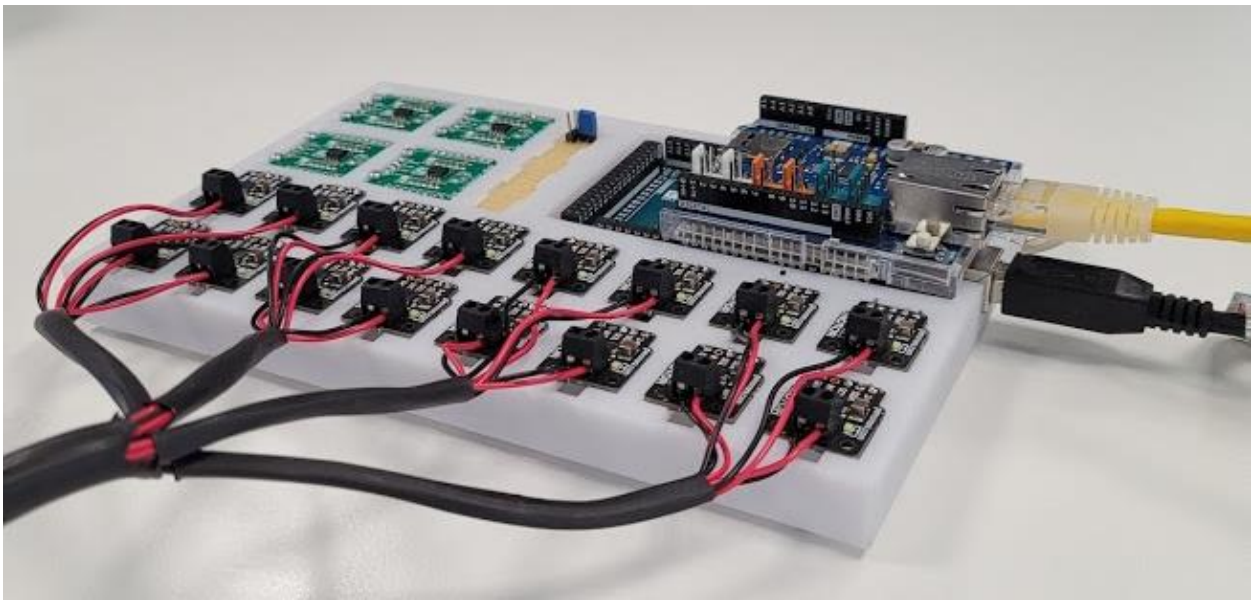
Vedmar, L. (2012). *Maskinelement - Transmissioner.* Lund: KFS i Lund AB.

Weslay, C., Irei, A., & Burke, J. (2020, 03). *Ethernet*. Retrieved 04 08, 2021, from SeachNetwork TechTarget: https://searchnetworking.techtarget.com/definition/Ethernet

Appendix A – 3D printed prototype




I2C isolator click
MEGA
motor



Appendix B – Arduino Program

```
#include <SparkFunMiniMoto.h>
#include <EthernetUdp.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xA8, 0x61, 0x0A, 0xAe, 0x65, 0xE2
};
IPAddress ip(192, 168, 0, 2);

unsigned int localPort = 8888;      // local port to listen on

// buffers for receiving and sending data
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  // buffer to hold incoming
packet,

// An EthernetUDP instance to let us send and receive packets over UDP
EthernetUDP Udp;

MiniMoto motor1(0xC0); // A1 = 0,     A0 = 0
MiniMoto motor2(0xC2); // A1 = 0,     A0 = Open
MiniMoto motor3(0xC6); // A1 = Open,  A0 = 0
MiniMoto motor4(0xC8); // A1 = Open,  A0 = Open

boolean m1;
boolean m2;
boolean m3;
boolean m4;

int panSpeed = 67;
int twistSpeed = 20;
int tiltSpeed = 15;
int rotateSpeed = 67;

int panDelay = 150;
int twistDelay = 40;
int tiltDelay = 15;
int rotateDelay = 25;

int angle;
int newAngle;
int sign;

int currentAnglePan1;
int currentAngleTwist1;
int currentAngleTilt1 = 90;
int currentAngleRotate1;

int currentAnglePan2;
int currentAngleTwist2;
int currentAngleTilt2 = 90;
int currentAngleRotate2;

int currentAnglePan3;
```

```
int currentAngleTwist3;
int currentAngleTilt3 = 90;
int currentAngleRotate3;

int currentAnglePan4;
int currentAngleTwist4;
int currentAngleTilt4 = 90;
int currentAngleRotate4;

#define module1Pin 33
#define module2Pin  41
#define module3Pin  45
#define module4Pin  49


void setup() {
  pinMode(module1Pin, OUTPUT);
  pinMode(module2Pin, OUTPUT);
  pinMode(module3Pin, OUTPUT);
  pinMode(module4Pin, OUTPUT);
  delay(200);
  digitalWrite(module1Pin, HIGH);
  delay(200);
  digitalWrite(module2Pin, HIGH);
  delay(200);
  digitalWrite(module3Pin, HIGH);
  delay(200);
  digitalWrite(module4Pin, HIGH);

  // You can use Ethernet.init(pin) to configure the CS pin
  Ethernet.init(10);  // Most Arduino shields

  // start the Ethernet
  Ethernet.begin(mac, ip);

  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  Serial.print("Serial setup");
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  //Check for Ethernet hardware present
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without
hardware. :(");
    while (true) {
      delay(1); // do nothing, no point running without Ethernet hardware
    }
  }
  // start UDP
  Udp.begin(localPort);
}

int readAngle() {
  int sign;
  if (packetBuffer[1] == '+') {
    sign = 1;
```

```
  }
  else {
    sign = -1;
  }
  int i = 2;
  int angle = 0;
  int value = 0;
  while (packetBuffer[i] != '.') {
    value = packetBuffer[i] - '0';
    angle += value * pow(10, 4 - i);
    i++;
  }
  return sign * angle;
}

int checkSign(int angle) {
  if (newAngle > angle) {
    return 1;
  }
  else {
    return -1;
  }
}
void reads() {
  boolean message = false;
  while (!message) {
    // if there's data available, read a packet
    int packetSize = Udp.parsePacket();
    if (packetSize) {
      Serial.print("Received packet of size ");
      Serial.println(packetSize);
      Serial.print("From ");
      IPAddress remote = Udp.remoteIP();
      for (int i = 0; i < 4; i++) {
        Serial.print(remote[i], DEC);
        if (i < 3) {
          Serial.print(".");
        }
      }
      Serial.print(", port ");
      Serial.println(Udp.remotePort());

      // read the packet into packetBuffer
      Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
      Serial.print("Contents: ");
      Serial.println(packetBuffer);
      message = true;
    }
  }
}
void loop() {
  Serial.println("Starting Loop");
  reads();
  if (packetBuffer[0] == '1') {
    m1 = !m1;
    if (m1) {
      digitalWrite(module1Pin, LOW);
```

```
    m2 = false;
    digitalWrite(module2Pin, HIGH);
    m3 = false;
    digitalWrite(module3Pin, HIGH);
    m4 = false;
    digitalWrite(module4Pin, HIGH);
    Serial.println("Module 1 activated");
  }
  else {
    Serial.println("Module 1 deactivated");
    digitalWrite(module1Pin, HIGH);
  }
}
if (packetBuffer[0] == '2') {
  m2 = !m2;
  if (m2) {
    digitalWrite(module2Pin, LOW);

    m1 = false;
    digitalWrite(module1Pin, HIGH);
    m3 = false;
    digitalWrite(module3Pin, HIGH);
    m4 = false;
    digitalWrite(module4Pin, HIGH);
    Serial.println("Module 2 activated");
  }
  else {
    Serial.println("Module 2 deactivated");
    digitalWrite(module2Pin, HIGH);
  }

}
if (packetBuffer[0] == '3') {
  m3 = !m3;
  if (m3) {
    digitalWrite(module3Pin, LOW);

    m1 = false;
    digitalWrite(module1Pin, HIGH);
    m2 = false;
    digitalWrite(module2Pin, HIGH);
    m4 = false;
    digitalWrite(module4Pin, HIGH);
    Serial.println("Module 3 activated");
  }
  else {
    Serial.println("Module 3 deactivated");
    digitalWrite(module3Pin, HIGH);
  }

}
if (packetBuffer[0] == '4') {
  m4 = !m4;
  if (m4) {
    digitalWrite(module4Pin, LOW);
```

```
    m1 = false;
    digitalWrite(module1Pin, HIGH);
    m2 = false;
    digitalWrite(module2Pin, HIGH);
    m3 = false;
    digitalWrite(module3Pin, HIGH);
    Serial.println("Module 4 activated");
  }
  else {
    Serial.println("Module 4 deactivated");
    digitalWrite(module4Pin, HIGH);
  }
}
if (packetBuffer[0] == 'a') {
  newAngle = readAngle();

  if (m1) {
    sign = checkSign(currentAnglePan1) * -1;
    angle = newAngle - currentAnglePan1;
    int tempDelayInt = abs(angle * panDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor1.drive(sign * panSpeed);
    delay(tempDelay);
    motor1.stop();
    currentAnglePan1 = newAngle;
  }
  if (m2) {
    sign = checkSign(currentAnglePan2) * -1;
    angle = newAngle - currentAnglePan2;
    int tempDelayInt = abs(angle * panDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor1.drive(sign * panSpeed);
    delay(tempDelay);
    motor1.stop();
    currentAnglePan2 = newAngle;
  }
  if (m3) {
    sign = checkSign(currentAnglePan3) * -1;
    angle = newAngle - currentAnglePan3;
    int tempDelayInt = abs(angle * panDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor1.drive(sign * panSpeed);
    delay(tempDelay);
    motor1.stop();
    currentAnglePan3 = newAngle;
  }
  if (m4) {
    sign = checkSign(currentAnglePan4) * -1;
    angle = newAngle - currentAnglePan4;
    int tempDelayInt = abs(angle * panDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor1.drive(sign * panSpeed);
    delay(tempDelay);
    motor1.stop();
    currentAnglePan4 = newAngle;
  }
```

```
}
if (packetBuffer[0] == 'b') {
  newAngle = readAngle();

  if (m1) {
    sign = checkSign(currentAngleTwist1) * -1;
    angle = newAngle - currentAngleTwist1;
    int tempDelayInt = abs(angle * twistDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor2.drive(sign * twistSpeed);
    delay(tempDelay);
    motor2.stop();
    currentAngleTwist1 = newAngle;
  }
  if (m2) {
    sign = checkSign(currentAngleTwist2 * -1);
    angle = newAngle - currentAngleTwist2;
    int tempDelayInt = abs(angle * twistDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor2.drive(sign * twistSpeed);
    delay(tempDelay);
    motor2.stop();
    currentAngleTwist2 = newAngle;
  }
  if (m3) {
    sign = checkSign(currentAngleTwist3) * -1;
    angle = newAngle - currentAngleTwist3;
    int tempDelayInt = abs(angle * twistDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor2.drive(sign * twistSpeed);
    delay(tempDelay);
    motor2.stop();
    currentAngleTwist3 = newAngle;
  }
  if (m4) {
    sign = checkSign(currentAngleTwist4) * -1;
    angle = newAngle - currentAngleTwist4;
    int tempDelayInt = abs(angle * twistDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor2.drive(sign * twistSpeed);
    delay(tempDelay);
    motor2.stop();
    currentAngleTwist4 = newAngle;
  }
}
if (packetBuffer[0] == 'c') {
  newAngle = readAngle();

  if (m1) {
    sign = checkSign(currentAngleTilt1) * -1;
    angle = newAngle - currentAngleTilt1;
    int tempDelayInt = abs(angle * tiltDelay);
    unsigned long tempDelay = (unsigned long) tempDelayInt;
    motor3.drive(sign * tiltSpeed);
    delay(tempDelay);
    motor3.stop();
    currentAngleTilt1 = newAngle;
```

```
    }
    if (m2) {
      sign = checkSign(currentAngleTilt2)* -1;
      angle = newAngle - currentAngleTilt2;
      int tempDelayInt = abs(angle * tiltDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor3.drive(sign * tiltSpeed);
      delay(tempDelay);
      motor3.stop();
      currentAngleTilt2 = newAngle;
    }
    if (m3) {
      sign = checkSign(currentAngleTilt3)* -1;
      angle = newAngle - currentAngleTilt3;
      int tempDelayInt = abs(angle * tiltDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor3.drive(sign * tiltSpeed);
      delay(tempDelay);
      motor3.stop();
      currentAngleTilt3 = newAngle;
    }
    if (m4) {
      sign = checkSign(currentAngleTilt4)* -1;
      angle = newAngle - currentAngleTilt4;
      int tempDelayInt = abs(angle * tiltDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor3.drive(sign * tiltSpeed);
      delay(tempDelay);
      motor3.stop();
      currentAngleTilt4 = newAngle;
    }
  }
  if (packetBuffer[0] == 'd') {
    newAngle = readAngle();

    if (m1) {
      sign = checkSign(currentAngleRotate1)* -1;
      angle = newAngle - currentAngleRotate1;
      int tempDelayInt = abs(angle * rotateDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor4.drive(sign * rotateSpeed);
      delay(tempDelay);
      motor4.stop();
      currentAngleRotate1 = newAngle;
    }
    if (m2) {
      sign = checkSign(currentAngleRotate2)* -1;
      angle = newAngle - currentAngleRotate2;
      int tempDelayInt = abs(angle * rotateDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor4.drive(sign * rotateSpeed);
      delay(tempDelay);
      motor4.stop();
      currentAngleRotate2 = newAngle;
    }
    if (m3) {
      sign = checkSign(currentAngleRotate3)* -1;
```

```
      angle = newAngle - currentAngleRotate3;
      int tempDelayInt = abs(angle * rotateDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor4.drive(sign * rotateSpeed);
      delay(tempDelay);
      motor4.stop();
      currentAngleRotate3 = newAngle;
    }
    if (m4) {
      sign = checkSign(currentAngleRotate4)* -1;
      angle = newAngle - currentAngleRotate4;
      int tempDelayInt = abs(angle * rotateDelay);
      unsigned long tempDelay = (unsigned long) tempDelayInt;
      motor4.drive(sign * rotateSpeed);
      delay(tempDelay);
      motor4.stop();
      currentAngleRotate4 = newAngle;
    }
  }
}
```

Appendix C – GUI program

The full processing GUI program is divided into three files named GUI, Communication and Methods. GUI is the main program which uses the methods in Communication and Methods. The code has been structured this way to make it easier to follow.

GUI

```
// ControlP5 Example 2 : Style UI elements and setValue
// ControlP5 by Andreas Schlegel :
// http://www.sojamo.de/libraries/controlP5/

import controlP5.*;
import hypermedia.net.*;

//DECLARE
UDP udp;  // define the UDP object
ControlP5 controlP5;
Textarea txt;
boolean pressed1;
boolean pressed2;
boolean pressed3;
boolean pressed4;

float currentAnglePan1;
float currentAngleTwist1;
float currentAngleTilt1 = 90;
float currentAngleRotate1;

float currentAnglePan2;
float currentAngleTwist2;
float currentAngleTilt2 = 90;
float currentAngleRotate2;

float currentAnglePan3;
float currentAngleTwist3;
float currentAngleTilt3 = 90;
float currentAngleRotate3;

float currentAnglePan4;
float currentAngleTwist4;
float currentAngleTilt4 = 90;
float currentAngleRotate4;

//Program Initiation
void setup() {
  size(860, 380);
  smooth();
  controlP5 = new ControlP5(this);
  InitCom();

  controlP5.setBroadcast(false);
  designSetup();
  Buttons();
```

```
  Sliders();
  textField();
  draw();
  controlP5.setBroadcast(true);
  udp.listen( true );
}


//EventHandler for controlsystem
void controlEvent(ControlEvent theEvent) {

  if (theEvent.isController()) {

    txt.append("control event from : "+theEvent.getController().getName() +
&amp;apos;\n&amp;apos;);
    txt.append("value : "+theEvent.getController().getValue() +
&amp;apos;\n&amp;apos;);
    Float value = new Float(theEvent.getController().getValue());
    String sign;

    if (value < 0) {
      sign = "-";
      value = -1 * value;
    } else {
      sign = "+";
    }
    String message = value.toString();

    // Chose MODULE to control
    if (theEvent.getController().getName()=="MODULE 1") {
      Send(message);
      pressed1 = !pressed1;
      controlP5.setBroadcast(false);
      controlP5.getController("Pan").setValue(currentAnglePan1);
      controlP5.getController("Twist").setValue(currentAngleTwist1);
      controlP5.getController("Tilt").setValue(currentAngleTilt1);
      controlP5.getController("Rotate").setValue(currentAngleRotate1);
      controlP5.setBroadcast(true);
      if (pressed1) {
        theEvent.getController().setColorBackground(0xffDF0000);
        controlP5.getController("MODULE 2").setColorBackground(0xff4B4B4B);
        controlP5.getController("MODULE 3").setColorBackground(0xff4B4B4B);
        controlP5.getController("MODULE 4").setColorBackground(0xff4B4B4B);
        pressed2 = false;
        pressed3 = false;
        pressed4 = false;
      } else {
        theEvent.getController().setColorBackground(0xff4B4B4B);
      }
    }

    if (theEvent.getController().getName()=="MODULE 2") {
      Send(message);
      pressed2 = !pressed2;
      controlP5.setBroadcast(false);
      controlP5.getController("Pan").setValue(currentAnglePan2);
      controlP5.getController("Twist").setValue(currentAngleTwist2);
```

```
  controlP5.getController("Tilt").setValue(currentAngleTilt2);
  controlP5.getController("Rotate").setValue(currentAngleRotate2);
  controlP5.setBroadcast(true);
  if (pressed2) {
    theEvent.getController().setColorBackground(0xffDF0000);
    controlP5.getController("MODULE 1").setColorBackground(0xff4B4B4B);
    controlP5.getController("MODULE 3").setColorBackground(0xff4B4B4B);
    controlP5.getController("MODULE 4").setColorBackground(0xff4B4B4B);
    pressed1 = false;
    pressed3 = false;
    pressed4 = false;
  } else {
    theEvent.getController().setColorBackground(0xff4B4B4B);
  }
}

if (theEvent.getController().getName()=="MODULE 3") {
  Send(message);
  pressed3 = !pressed3;
  controlP5.setBroadcast(false);
  controlP5.getController("Pan").setValue(currentAnglePan3);
  controlP5.getController("Twist").setValue(currentAngleTwist3);
  controlP5.getController("Tilt").setValue(currentAngleTilt3);
  controlP5.getController("Rotate").setValue(currentAngleRotate3);
  controlP5.setBroadcast(true);
  if (pressed3) {
    theEvent.getController().setColorBackground(0xffDF0000);
    controlP5.getController("MODULE 1").setColorBackground(0xff4B4B4B);
    controlP5.getController("MODULE 2").setColorBackground(0xff4B4B4B);
    controlP5.getController("MODULE 4").setColorBackground(0xff4B4B4B);
    pressed1 = false;
    pressed2 = false;
    pressed4 = false;
  } else {
    theEvent.getController().setColorBackground(0xff4B4B4B);
  }
}

if (theEvent.getController().getName()=="MODULE 4") {
  Send(message);
  pressed4 = !pressed4;
  controlP5.setBroadcast(false);
  controlP5.getController("Pan").setValue(currentAnglePan4);
  controlP5.getController("Twist").setValue(currentAngleTwist4);
  controlP5.getController("Tilt").setValue(currentAngleTilt4);
  controlP5.getController("Rotate").setValue(currentAngleRotate4);
  controlP5.setBroadcast(true);
  if (pressed4) {
    theEvent.getController().setColorBackground(0xffDF0000);
    controlP5.getController("MODULE 1").setColorBackground(0xff4B4B4B);
    controlP5.getController("MODULE 2").setColorBackground(0xff4B4B4B);
    controlP5.getController("MODULE 3").setColorBackground(0xff4B4B4B);
    pressed1 = false;
    pressed2 = false;
    pressed3 = false;
  } else {
    theEvent.getController().setColorBackground(0xff4B4B4B);
```

```
  }
}


if (value < 100) {
  message = "0" + message;
}

if (value < 10) {
  message = "0" + message;
}

//Pan/Twist/Tilt/Rotate
if (theEvent.getController().getName()=="Pan") {
  Send("a" + sign + message);
  if (pressed1) {
    currentAnglePan1 = theEvent.getController().getValue() ;
  }
  if (pressed2) {
    currentAnglePan2 = theEvent.getController().getValue() ;
  }
  if (pressed3) {
    currentAnglePan3 = theEvent.getController().getValue() ;
  }
  if (pressed4) {
    currentAnglePan4 = theEvent.getController().getValue() ;
  }
}

// clicking on MOTOR 2 sets toggle1 value to 0 (false)
if (theEvent.getController().getName()=="Twist") {
  Send("b" + sign + message);
  if (pressed1) {
    currentAngleTwist1 = theEvent.getController().getValue() ;
  }
  if (pressed2) {
    currentAngleTwist2 = theEvent.getController().getValue() ;
  }
  if (pressed3) {
    currentAngleTwist3 = theEvent.getController().getValue() ;
  }
  if (pressed4) {
    currentAngleTwist4 = theEvent.getController().getValue() ;
  }
}

// clicking on MOTOR 3 sets toggle1 value to 0 (false)
if (theEvent.getController().getName()=="Tilt") {
  Send("c" + sign + message);
  if (pressed1) {
    currentAngleTilt1 = theEvent.getController().getValue() ;
  }
  if (pressed2) {
    currentAngleTilt2 = theEvent.getController().getValue() ;
  }
  if (pressed3) {
    currentAngleTilt3 = theEvent.getController().getValue() ;
```

```
      }
      if (pressed4) {
        currentAngleTilt4 = theEvent.getController().getValue() ;
      }
    }

    // clicking on MOTOR 4 sets toggle1 value to 0 (false)
    if (theEvent.getController().getName()=="Rotate") {
      Send("d" + sign + message);
      if (pressed1) {
        currentAngleRotate1 = theEvent.getController().getValue() ;
      }
      if (pressed2) {
        currentAngleRotate2 = theEvent.getController().getValue() ;
      }
      if (pressed3) {
        currentAngleRotate3 = theEvent.getController().getValue() ;
      }
      if (pressed4) {
        currentAngleRotate4 = theEvent.getController().getValue() ;
      }
    }
  }
}
```

Communication

```
void  InitCom() {
  udp = new UDP( this, 6000 );  // create a new datagram connection on port
6000
  //udp.log( true );     // <-- printout the connection activity
  udp.listen( true );           // and wait for incoming message
}

void Send(String message) {
  String ip       = "192.168.0.2";  // the remote IP address
  int port        = 8888;    // the destination port

  udp.send(message, ip, port );   // the message to send
  println(message);
}

void receive( byte[] data ) {       // <-- default handler
  //void receive( byte[] data, String ip, int port ) {  // <-- extended
handler

  for (int i=0; i < data.length; i++)
    print(char(data[i]));
    println();

  data = subset(data, 0, data.length);
  String error = new String( data );
  txt.append(error + &apos;\n&apos;);


}
```

Methods

```
void designSetup() {

  // change the default font to Verdana
  PFont p = createFont("Verdana", 10);
  ControlFont font = new ControlFont(p);

  // change the original colors

  controlP5.setColorForeground(0xff717171);
  controlP5.setColorBackground(0xff4B4B4B);
  controlP5.setFont(font);
  controlP5.setColorActive(0xffDF0000);
}

void Buttons() {
  //Placement parameters
  int height = 100;      //Height of button
  int width = 100;      //Width of button
  int space = 20;        //Space between buttons
  int yplacement = 20;  //y-axis placment on application grid
  int xplacement = 0;    //x-axis placement on application grid

  //Motor Buttons
  controlP5.addButton("MODULE 1")
    .setPosition(xplacement+space, yplacement)
    .setSize(width, height)
    .setColorForeground(0xffDF0000)
    .setValue((float)1);
  controlP5.addButton("MODULE 2")
    .setPosition(xplacement+width + 2*space, yplacement)
    .setSize(width, height)
    .setColorForeground(0xffDF0000)
    .setValue((float)2);
  controlP5.addButton("MODULE 3")
    .setPosition(xplacement+2*width + 3*space, yplacement)
    .setSize(width, height)
    .setColorForeground(0xffDF0000)
    .setValue((float)3);
  controlP5.addButton("MODULE 4")
    .setPosition(xplacement+3*width + 4*space, yplacement)
    .setSize(width, height)
    .setColorForeground(0xffDF0000)
    .setValue((float)4);
}
void Sliders() {
  //Placement parameters
  int height = 40;      //Height of button
  int width = 410;      //Width of button
  int space = 20;        //Space between buttons
  int yplacement = 120;  //y-axis placment on application grid
  int xplacement = 20;    //x-axis placement on application grid

  //Range values
  float maxPan = 180;
```

```
  float minPan = -maxPan;
  float maxTwist = 25;
  float minTwist = -maxTwist;
  float maxTilt = 95;
  float minTilt = -5;
  float maxRotate= 95;
  float minRotate = -5;

  //Pan/Twist/Tilt/Rotate
  controlP5.addSlider("Pan")
    .setTriggerEvent(Slider.RELEASE)
    .setPosition(xplacement, yplacement+space)
    .setSize(width, height)
    .setRange(minPan, maxPan);
  controlP5.addSlider("Twist")
    .setTriggerEvent(Slider.RELEASE)
    .setPosition(xplacement, yplacement+height + 2*space)
    .setSize(width, height)
    .setRange(minTwist, maxTwist);
  controlP5.addSlider("Tilt")
    .setTriggerEvent(Slider.RELEASE)
    .setPosition(xplacement, yplacement+2*height + 3*space)
    .setSize(width, height)
    .setValue(90)
    .setRange(minTilt, maxTilt);
  controlP5.addSlider("Rotate")
    .setTriggerEvent(Slider.RELEASE)
    .setPosition(xplacement, yplacement+3*height + 4*space)
    .setSize(width, height)
    .setRange(minRotate, maxRotate);
}
Textarea textField() {
  txt = controlP5.addTextarea("txt")
    .setPosition(500, 20)
    .setSize(340, 340)
    .setFont(createFont("arial", 12))
    .setLineHeight(14)
    .setColor(color(255))
    .setColorBackground(color(113, 113, 113))
    .setColorForeground(color(255, 100));
  return txt;
}

void draw() {
  background(200); // background black
}
```

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2021

Document Number
TFRT-6126

Author(s)
Linus Wannebro
Victor Ohlson

Supervisor
Malin Klein, Axis, Sweden
Rasmus Axenram, Axis, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Remote Positioning of a Multidirectional Camera

Abstract

In today's world of uncertainties, video surveillance is widely used to make the people of our society feel safe and protected. As more cameras are installed, the time of installation and ease of use play a critical role when customers decide which camera solution to select.

One way to limit the number of cameras needed to cover multiple directions and large areas is to install a multidirectional camera. A multidirectional camera is a complete camera solution where multiple camera heads are incorporated in the same chassis and the most common type of multidirectional camera has only one IP-address which further favors usability. Additionally, to limit the time of installation and allow for adjustments during the camera's lifetime it is beneficial to incorporate a remote positioning functionality to the camera.

The objective of this thesis is to develop and produce a working prototype of a multidirectional camera, which consists of multiple camera modules, each with the capabilities of movement in four directions, pan, twist, tilt, and rotate which all can be controlled remotely. This was made possible by making an entirely new mechanical platform, PCB, and software for controlling the camera heads individually.

The concept development phase was dominated by investigating which type of transmissions and actuators were best suited to fulfill the needs requested from the company. After deciding which concepts to further develop, these concepts were realized and produced through 3D printing. The prototype was then tested in order to verify that the needs were satisfied.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-91

Recipient's notes

Security classification