Historic Calculator Project

California Polytechnic State University

Mechanical Engineering Senior Project

Final Report

Fall 2011

Sponsor

Deutsches Museum, Munich, Germany

Team

Will Harris

Dan Marioni

Kevin Webb

Chad Williams

Team email: deutschescalpoly@gmail.com

Advisors

Sarah Harding: sthardin@calpoly.edu

Dr. Frank Owen: fowen@calpoly.edu

STATEMENT OF DISCLAIMER

Unless otherwise stated, the software provided is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. THERE IS NO WARRANTY FOR THE SOFTWARE, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS PROVIDE THE SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE SOFTWARE TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

EXECUTIVE SUMMARY

The Deutsches Museum of Munich, Germany requested add-on components to their mechanical calculator exhibit. The end-goal is to produce an interactive 3D model in software to demonstrate the functions of the Braun-Vayringe Machine (BVM). There will be four students working on this project at California Polytechnic State University, San Luis Obispo: two mechanical engineers (Dan Marioni and Will Harris) and two computer engineers (Chad Williams and Kevin Webb).

This project involves the coordination of three facilities; the Deutsches Museum, and the Mechanical and Computer Science Departments of Cal Poly. The name of the project is known as the Historic Calculator Project (HCP), because of the coordinated effort to revisit a piece of history using modern day technology. The following report explains the project from two perspectives, therefore each section is divided into two parts: Mechanical Engineers (ME SIDE) and Computer Engineers (CPE SIDE).

INTRODUCTION

THE MUSEUM

The Deutsches Museum is located in Munich Germany and prides itself on tracking the development of technology. The museum was founded in 1903 and has been steadfast in its promise to document, educate, and research technology from aeronautical to computer science achievements throughout history.

The museum has worked with Cal Poly since 2006. The first project was working with the Frauenkirche Turmuhr, an old historic clock located in the Frauenkirche church in downtown Munich. Up to 2008, the Turmuhr project has had two different teams associated with Cal Poly working on modeling and created simulations of the working inner components of the clock. The work put into the Turmuhr has also helped with modeling the turning mechanism in the Piedras Blancas lighthouse in Cambria, California.

There is great interest by both Cal Poly and the Deutsches Museum to continue working together. The museum is very interested in creating more interactive exhibits to make the museum more exciting and increase the learning experience. The museum has many projects that offer an opportunity for international experience with historic and technological intrigue.

THE MACHINE

In 1724, Anton Braun, a German mechanic from Baden-Württemberg, Germany, was appointed as the optician of the imperial court in Vienna, Austria. Shortly before his death in 1728, he had been designing a machine in order to aid the court in their calculations. The machine that he thought of came from the ideas set previously by instrument maker, physicist, and mathematician Jacob Leupold, who died in 1727. Leupold's design was never produced, due to his early passing. A machine was produced based on these designs by a mechanic named Phillippe Vayringe who built the Braun design in 1737, hence the name Braun-Vayringe [1].

The designed that inspired this machine by Leupold is based on a circular design. Numbers (1-9 and 0) would be entered into the central wheels, with one full revolution of the crank these numbers were then transferred to the outer dials (Figure 1). The most outer "result" mechanism has 11 wheels as shown in Figure 2. These wheels have a smaller inner wheel and larger outer wheel that display addition/subtraction results.

Figure 1: The Deutsches Museum replica model with a transparent top to show the rotating gears and number planes.

The design that Leupold made was not directly implemented into the Braun-Vayringe, but there are similarities between the basic function and carrying mechanism. Originally, there were only 6 inner wheels and nine on the outer ring, yet the Braun Vayringe has 7 and 11 respectively. These designs went on to further calculating machines that grew in complexity.

The Deutsches Museum has selected the BVM partially because there is not extensive research done on this machine. The work done on the calculator is the foundation for further efforts which may potentially include technical papers and a completely interactive touch screen exhibit.

SPONSOR REQUESTED NEEDS

ME SIDE

The ultimate goal for the MEs is to model the BVM using SolidWorks CAD. There are limited technical specifications for modeling the BVM. A large portion of the BVM project will include coordinated research with the museum's *Urmachermeister* (Master Watch and Clockmaker), Thomas Rebenyi, who will act as the team's technical supervisor. He will deliver pictures and video of the disassembled machine to the mechanical engineers to explain how the machine works. The pictures will be turned into CAD models then given to the CPEs to be implemented into their software.

There will also be weekly meeting via Skype with Mr. Rebenyi in order to keep track of the project progress and answer questions about the machine which may be difficult to explain through emails and still images. This process will be accompanied by Dr. Owen for additional help with German-English translation if needed.

CPE SIDE

Our sponsor has a working replica of the BVM that needs to be modeled to show museum patrons how the calculator internally works and functions. The sponsor requested a standalone Windows 7 application that renders the model and allows users to interact with the software using a touch screen monitor. The program can run on separate Windows 7 computers simultaneously if multiple users want to use the software. The program will launch full screen and only accept user input from the touch screen. This way, museum patrons cannot access the underlying operating system underneath the application. The GUI (Graphical User Interface) will be modeled with a black theme with no museum logos. The majority of the screen will be used to display the model and allow users to zoom and rotate around the model. The other portion will be used for button layouts to switch user options.

FORMAL PROBLEM DESCRIPTION

ME SIDE

Using SolidWorks for CAD software, coordinate with the Deutsches Museum to translate the physical pieces of the Braun-Vayringe Calculator into a complete 3D assembly.

CPE SIDE

Produce and electronically deliver via drop box a standalone .exe application that allows users to zoom and rotate around the BVM model showing its internal functions.

PROJECT MANAGEMENT

The HCP has divided the responsibilities of this project such that everyone will be contributing an equal amount of work and responsibility.

Table 1: Project roles for each team member.

Position	**Requirements of Position**	**Completed by**
Documentation	One ME and one CPE will focus together on weekly reports and documentation requirements	Kevin Webb Dan Marioni
Research on Calculators	Both MEs will coordinate with the sponsor to learn about the BVM machine from photos and measurements.	Dan Marioni Will Harris
SolidWorks Modeling	MEs will produce 3D models of the parts required for the BVM.	Dan Marioni Will Harris
Software Design and Programming	CPEs will combine efforts to create the application prototype, test, and finalize the application	Kevin Webb Chad Williams
3D Model Importer	Importing and converting SW models into usable ones that can be used by the XNA framework	Chad Williams
Shipping / Building / Recommending PC Hardware	Test a PC with touch-screen ability, and load it with the 3D interactive application produced for the museum	Kevin Webb

BACKGROUND

The Deutsches Museum has two models of the BVM, the original machine shown in Figure 2 and a replica that was built as a side project some time ago. The museum does not want to risk damaging the original machine; therefore this project will utilize the replica. There are slight differences between the two models, such as artistic details in the casing and number dials that will be missing from our CAD model. However, the museum is fine with these differences, because the goal is to facilitate explanation of the machine's function. Fine detail in the CAD models will be a future goal of the museum.

The BVM is truly a piece of history and one of a kind. There are other "mechanical calculators" that have accomplished similar tasks. The best examples of these are also located in the Deutsches Museum, some examples are shown below in Table 3.

Figure 2: The original Braun-Vayringe machine.

Table 2: Similar historic calculators to the Braun-Vayringe Machine are shown below.

			
Name	Leibniz Stepped Rekoner	Hahn Calculator	Pasaline Calculator
Year	est. 1668	1774	est. 1650
Capabilities	addition, subtraction, multiplication, square roots	addition, subtraction, multiplication	addition, subtraction, multiplication
Function	via repeated addtion	via repeated addtion	via repeated addtion
Source	http://www.oobject.com/category/mechanical-marvel-calculators/		http://www.computermuseum.li/Testpage/Pascaline-Calculator-1642.htm

DESIGN DEVELOPMENT

ME SIDE

Concepts

The design goals of this project are to take an existing product and make a 3D model of it; therefore, there are no design concepts necessary for the calculator. Again the main goal is to create a solid model that will aid in teaching the function for the museum. The SolidWorks model should completely represent all of the parts and dimensions that are given to the team. The parts will be fully assembled to show the BVM as accurately as possible to the actual calculator.

Design Process

In order to efficiently model the BVM, the team will create a parts list of documented pieces from the museum. The calculator is divided up into three major sections: 100, 200 and 300; these numbers represent the inner, middle and outer portions of the calculator respectively (Figure 3). See the Appendix for the complete parts list and pictures.

The 100 assembly is known as the drive for the calculator, because it includes the crank. The 200 assembly is known as the input, because it's where the operators are input to the calculator one the inner dials. The 300 assembly is known as the output, because the output is read on the outer dials.

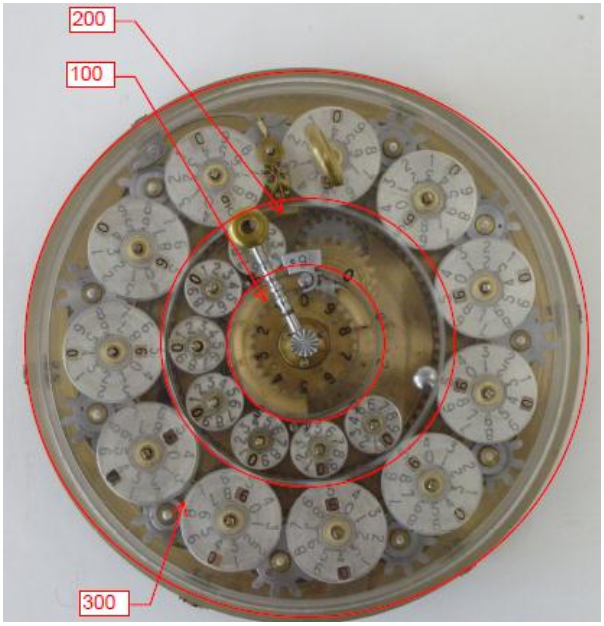


Figure 3: The 100, 200 and 300 divisions are shown above.

Dimensions of the calculator are received from the Deutsches Museum electronically either through email directly or off of the dropbox.com account set up for the project. Questions about dimensions, function or other are asked directly via email, or documented and handled on weekly meeting via Skype. As parts are completely modeled, their part name matches directly to the part number on the parts list to avoid any confusion. The parts list was created as a Google Document, such that each member working on the model had access to the parts list and knew which part still needed to be made (Figure 4). Parts are modeled and then uploaded to the Dropbox in order to keep all the files in one central place that any member can download from and work.

Braun Vayringe Parts List		
Part Range	**Description**	
100-199	Inner Circle	Crank, first gears
200-299	Second Circle	Inputs
300-399	Thrid Circle	Outputs
NOTES:	number increases with level - top to bottom. Part dwgs must be done. BLANK means that part has not been started yet. gears are based off of part 109	
Legend	**ip**	in progress (NAME)
	y	yes-complete
		needs dimensions

Figure 4: the legend for the parts list shown how each part is documented and progress is recorded.

CPE SIDE

Software is different from many other products in that it can be delivered and produced using no materials. Therefore, the design process is substantially different than that of a material process. Initial design decisions, from a software perspective, revolve around:

- What existing code and libraries are available
- Software licensing costs
- What language the code is produced in
- What development environment should be used
- How the end result is tested

Given that the museum has deployed a Microsoft Windows 7 environment, we knew right away that Linux software and other open source operating system graphics libraries were out of the question. Windows development tools and libraries are much easier to work with than many other open-source alternatives, so we decided to have our application be Windows-only, using the technologies described later in this report.

Visual Studio 2010 is a development environment designed to produce anything from desktop applications to websites on Windows-based desktops and servers. It supports designing user interfaces (UIs) in a framework known as WPF (Windows Presentation Foundation), as well as supporting C#; these two factors were major reasons for choosing Visual Studio.

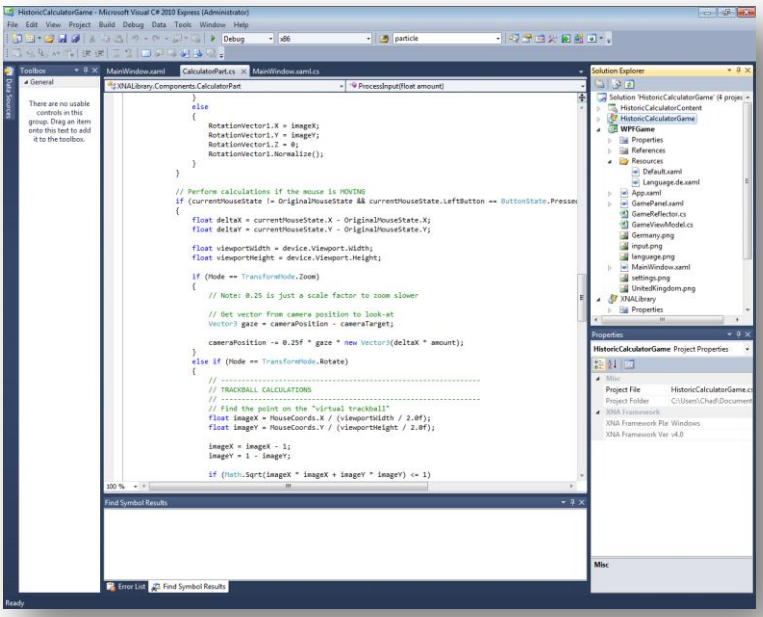


Figure 5: The Visual Studio development environment

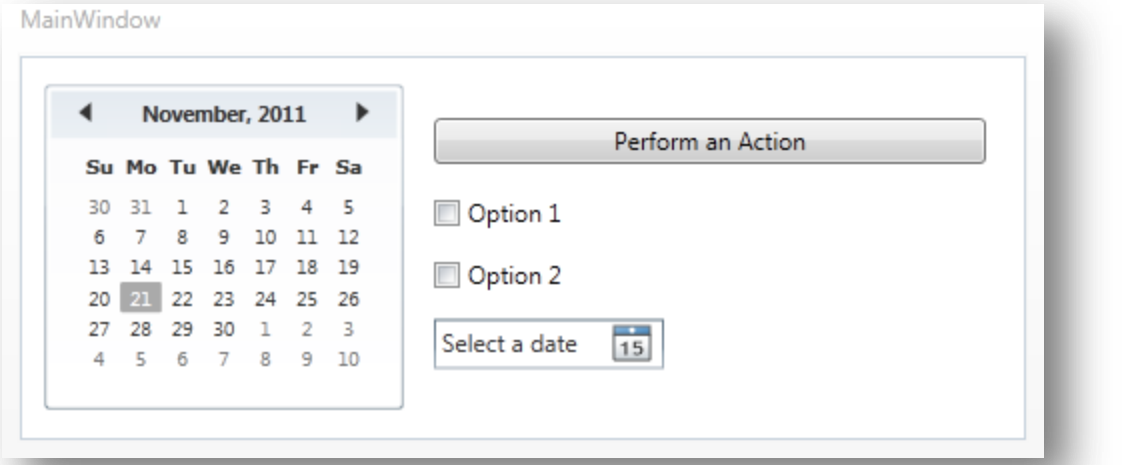


Figure 6: Simple WFP application example

WPF allows programmers to create User Interfaces (UIs) that are internally handled via *events*. It provides a framework that allows the programmer to design a clean UI, and provide code that the events will execute via a *code behind.* For example, clicking on the “Rotate” button will trigger the code that switches to rotation mode internally, code which is found in the code-behind and triggered by the button’s *click* event.

Given that we are using Visual Studio, the best option for a programming language will be C#. This language allows the program access to unique features in the included .NET framework.

The .NET framework provides a rich set of classes that abstract away the low-level details of programming. Most importantly, using the .NET framework will allow us to reduce development time.

The last design decision would be the selection of the graphics framework that will show the model to the user. Games (like Halo and Counter Strike) have game engines that render the 3D scene presented to the user. The components of this 3D scene can include models, lights, cameras, skyboxes, and more. Microsoft's XNA framework was chosen to be the 3D engine backing the kiosk application due to its ability to work side-by-side with WPF, its collection of graphics-related classes, and its ability to be used with C# (our chosen programming language), and its integration into Visual Studio. In total, the project uses the following:

- C# (using .NET) as our development language
- Visual Studio for our development environment
- XNA to render our graphics
- WPF to draw the user interface

Figure 7: Windows, XNA, and Visual Studio Development Logos

Coding Methodologies

The final design will incorporate the C# language, which is an object-oriented language. This is different than a language like standard C, which has no support for classes (and therefore no capacity for inheritance, and other object-oriented features). Languages like C rely primarily on function calls for program structure:

For example, let's assume three function calls A, B, and C:

```
A()

B()

C()
```

All three functions will be executed in order. Our final design will incorporate the object-oriented features in C#. The program will be broken up into classes that allow for easy code modification in the future.

An example of a class, in pseudo-code, is shown below:

```
Class Point

{
```

```
        float X, float Y, float Z

        A()

        B()

        C()

    }
```

In order to call any method A, B, or C, we would have to declare an *instance* of the Point class, and then call the methods on that object. This approach may add some complexity but offers long term benefits such as code maintainability, readability, and more. Object-oriented methods are more maintainable because the objects are self-contained via *encapsulation*. In addition, the code is scalable as we can add additional functionality to an object, or update an existing algorithm, by only changing the object; existing code that uses the objects would not need to be changed.

Work Division

In order to divide the programming work between two people, it is necessary that both students have access to updated code when a change has been made. Updates can be obtained through what's known as a *code repository*. Code can be uploaded to the repository server so others have access to the updated code immediately. There are many benefits to using a source-control server/repository. Benefits include the ability to perform *diffs* (seeing what has changed in between updates), a log of what changes have been made, the ability to allow multiple people to work on the same code base simultaneously, and the ability to revert to old versions of the code if necessary (if, for example, an update introduced a regression, or bug, into the codebase). For this project the SVN (Subversion) system was chosen due to its wide adoption and the wealth of available SVN clients available for Windows, most notably Tortoise SVN.

Figure 8: Repository Client Used to Share Code

Reference Designs

Inspiration for this project came from the "Southwest" baggage terminal kiosk found at most airports, where passengers check baggage in using a touchscreen terminal. This application employs large, easy-to-use buttons and a clean user interface for maximum touch-screen usability; these ideas were incorporated into the final design.

FINAL DESIGN

ME SIDE

The final design of the BVM includes a large assembly of parts that make up subassemblies. The machine is split into 4 main subassemblies; the inner, middle, outer, and housing assemblies. These subassemblies were modeled progressively from the inner part of the machine to the outer part. The inner assembly fits into the middle assembly, and the middle assembly fits into the outer assembly. The inner, middle, and outer assemblies all fit into the housing assembly making the entire machine.

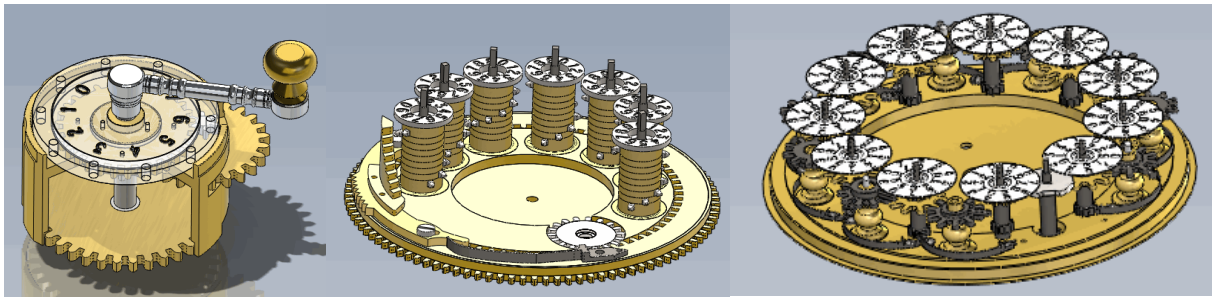


Figure 9: Inner, middle, outer assemblies of the BVM (left to right). These assemblies make up the internal working components of the machine.

In order to keep track of all the parts that make up the BVM, the parts list was used. With the use of the parts list, it is easy see the status of each individual part, and work independently on parts that have not been modeled yet.

When all subassemblies are completed, they can be assembled to form the complete machine. This final assembly will include all parts, and will be a virtual model of the BVM. With this model, the museum can create simulations, animations, internal views, exploded views, and even replicate parts of the machine. This model can be used however the museum wishes in order to educate on the function of the BVM.

The final design includes the complete model in addition to all parts associated with the model. These parts can be seen in the Appendix. The model can be taken apart to view subassemblies, and can be taken apart further to view all parts in the machine. The graphical representation and display of the machine can be presented in numerous ways, and can be customized with the creativity of the museum. The final design does not include a working simulation of the model. A simulation of the model can be performed in the future if the user wishes, however the time and computing power necessary for a working simulation made this goal secondary to the primary goal of completing a solid model of the machine.

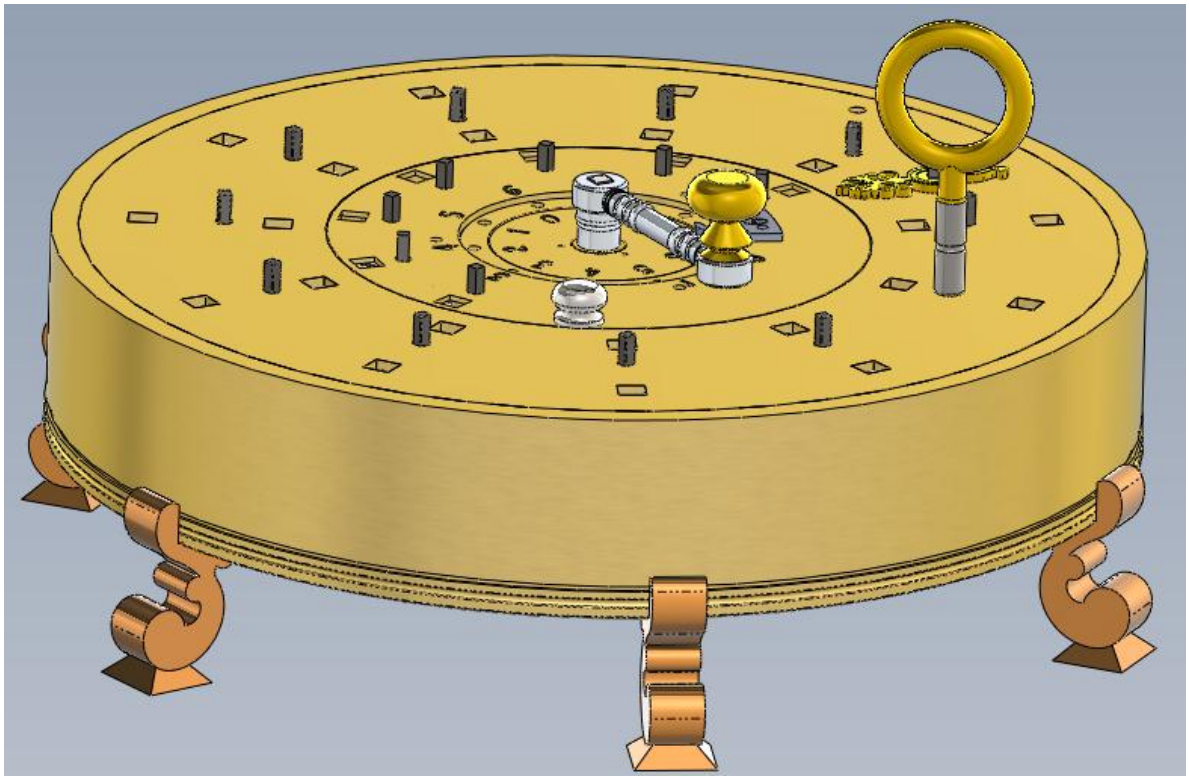


Figure 10: Final model of the BVM composed of all parts and subassemblies.

CPE SIDE

The final design will display language and sub-assembly options for the entire BVM model. Zoom and rotation features are available to allow the user to view all areas of the model. Rotation is “natural” according to the user’s touch input; no sliders are used. Similarly, zoom is implementing by swiping from left to right (zoom in), or right to left (zoom out). The language for all titles and captions will be changed by hitting the appropriate flag button at the top-left. Additionally, the program will auto-detect the locale used on the computer it is running on, so the language will (in most circumstances) be correct when the application is first started.

The user will be able to view the entire assembly as one “overall” unit, the exploded view of the entire model, and the individual assemblies that are contained in the entire model (each assembly has an exploded view and an “overall” view). Each sub-assembly will have a caption associated with it so the user viewing the sub-assembly is given information about how it works, and its role in the overall functionality of the calculator.

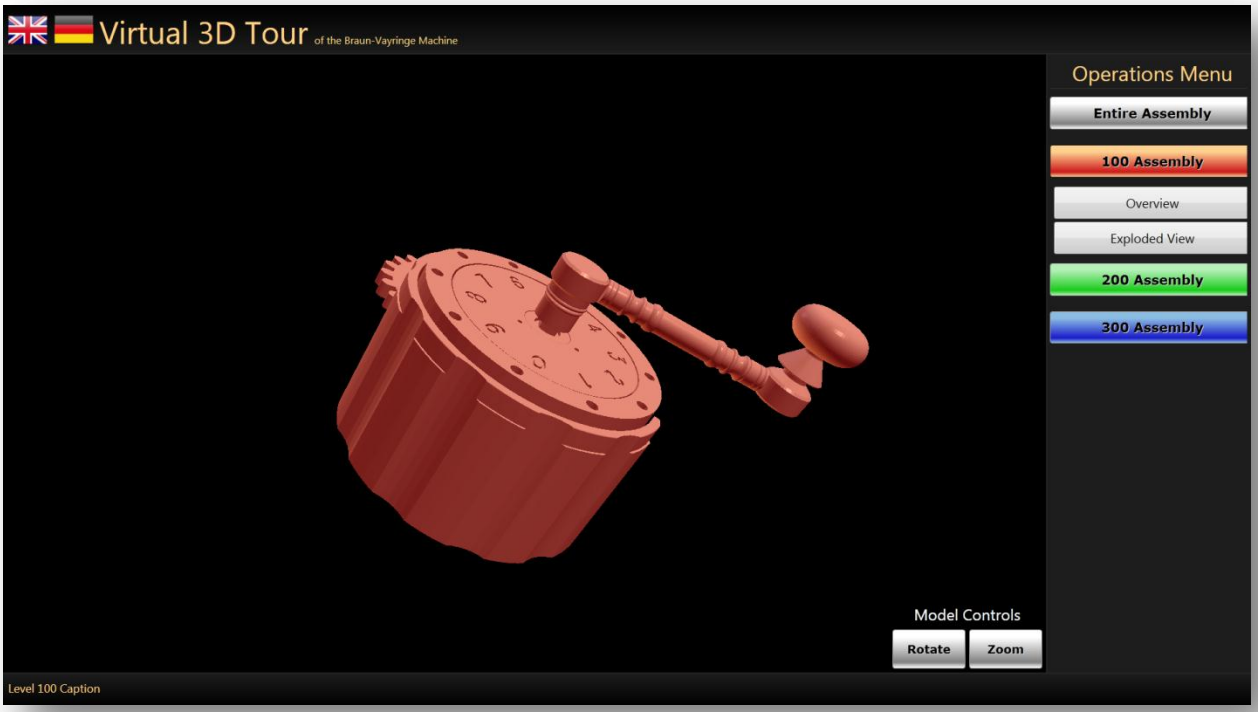


Figure 11: The final graphical environment for the models to be shown.

An interesting challenge that accompanied the user's interaction with the model was the way in which the model rotated while in *rotation mode.* Several different interaction methods were tested, and the final method was, by far, the most intuitive. The first design used "sliders" that could be dragged up-and-down (for rotation about the X axis) and left-and-right (for rotation about the Y axis). This worked, but was somewhat clunky (it restricted rotation to one axis at a time) and unintuitive. With the proliferation of touch-screen applications, another solution was needed. The second iteration involved, again, rotation about the individual X and Y axes, but instead of using sliders, the user would simply drag their finger vertically or horizontally to rotate about X or Y, respectively. This design eliminated the clunky sliders, but still had the problem of an unintuitive way of rotating an object.

The final design uses what is called a "virtual trackball" to model rotations. The virtual trackball superimposes a sphere over the entire 3D scene, and when a user touches the screen, the point at which they touch is projected on to this 3D sphere. When the user drags to rotate, a *rotation axis* is calculated, and a rotation matrix is created which encodes this rotation axis, as well as the rotation amount. This method eliminates sliders and single-axis rotation, and instead allows the user to rotate the object in a very natural way, as if they were actually touching it. For a full discussion of this technique, see the paper "Virtual Trackballs Revisited" [1].

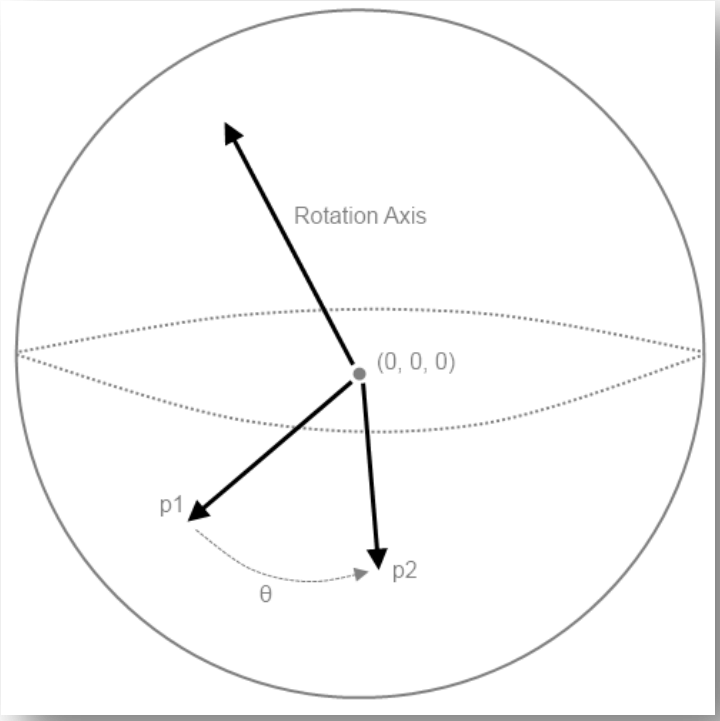


Figure 12: Trackball rotation scheme showing where the user clicked (p1) and dragged to (p2), and the corresponding rotation axis that is calculated

Another important feature integrated into the program is its ability to easily change between different languages. International flags were used as buttons to allow the user to switch between languages intuitively. In order to facilitate the ability to have the language changing while the program is running, all WPF controls that contain translatable text (buttons, text boxes, etc) are dynamically bound at runtime to a back-end XAML file that contains translations. This XAML file can be opened up in any text editor, and edited to change the translations. There is a separate XAML file for each language that is to be used in the program (in this case, just English and German are provided). Because the translations are stored in a text file, no modification of the C# code is needed in order to change the wording of any of the UI elements. Below is an example of all it takes to change the wording of the menu's title:

```
<system:String x:Key="MenuTitle">Operations Menu</system:String>
```

…to…

```
<system:String x:Key="MenuTitle">New Operations Menu Title</system:String>
```

In order for the models to display inside of the 3D environment, they had to be converted to the FBX format. FBX is a standard file format used in the 3D and visual effects industry, and is also one of the two file formats that XNA supports for importing 3D models. SolidWorks cannot export this format, so the open-source 3D program *Blender* was used to convert the STL format that SolidWorks exports into FBX format. Finally, *Autodesk Maya* was used to further fine-tune the model exported by Blender (such as moving the assembly's pivot point to the center of the object, and performing polygon reductions).

Figure 13: Export process to bring a SolidWorks model into XNA

In addition, Maya was used to artificially change the shading model inside of XNA from smooth-shaded to flat-shaded. The most modern version of XNA, version 4, no longer supports fixed-function graphics features, and as such, flat shading is not an option in XNA unless a custom shader is created. To easily emulate flat-shading, the “Set to Face” command was used inside of the Normals menu. This sets the normals of each vertex on a polygon to the normal of the polygon. Each vertex still has a normal, but those normals are now set to the polygon’s normal, effectively emulating flat shading. Flat shading was used to provide a more mechanical appearance, in addition to vastly improving the appearance of the models. Furthermore, the specular highlights give a more metallic appearance in the flat-shaded models due to the high polygon count of each assembly. A comparison of flat and smooth shading is given below. An unfortunate side effect of using flat-shading is that the models look more faceted, but this downside is mostly mitigated by the high polygon count present in the assemblies.

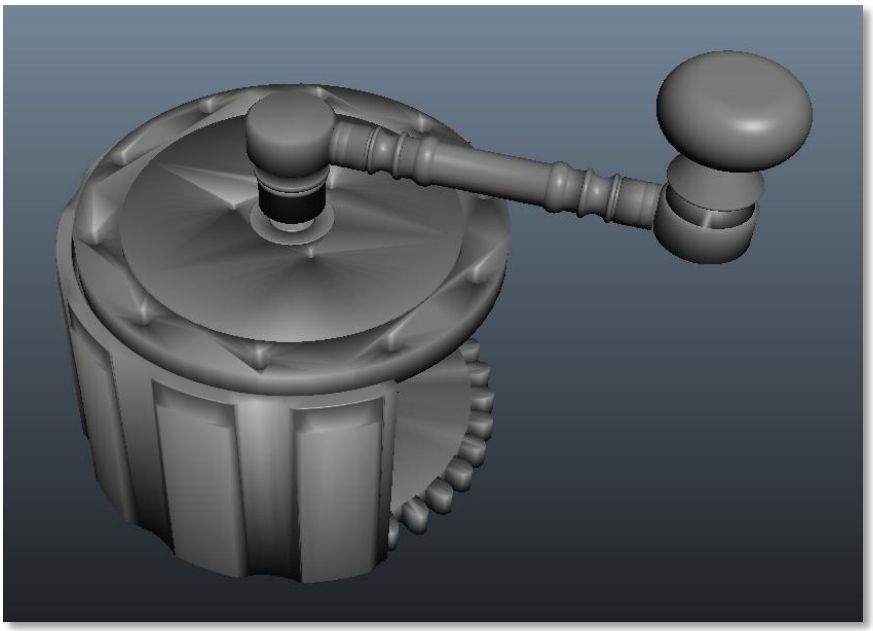


Figure 14: Smooth shading of an assembly (shown inside of Maya)

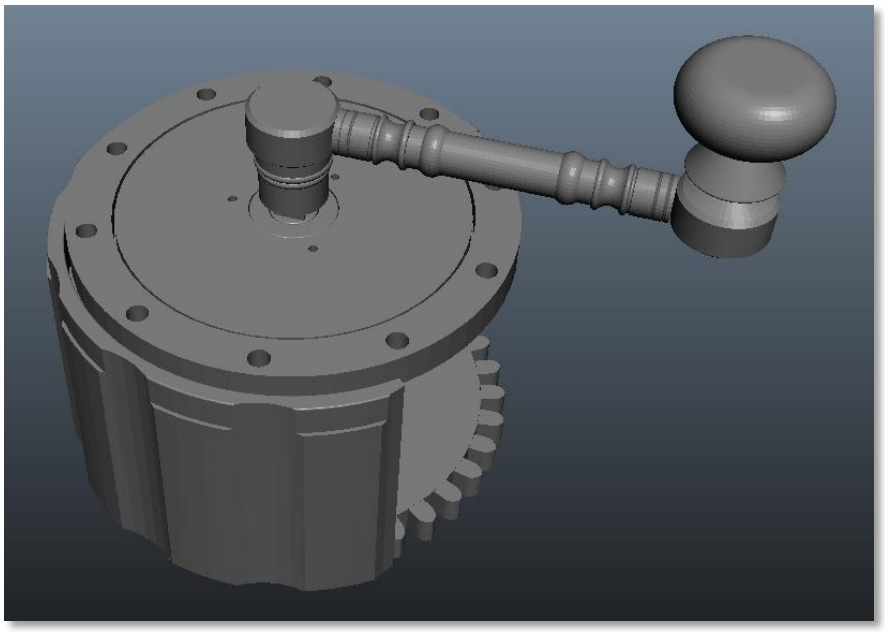


Figure 15: Flat shading of an assembly (shown inside of Maya)

A final concern involved in the design of the program was how responsive the application was. Higher levels of realism can be captured when there are more polygons in the assemblies, but this comes with slower interaction speed due to the graphics card having to perform more calculations, among other things. A compromise was reached by keeping most of the polygons in the models that were exported from SolidWorks, but polygon reductions were performed in certain areas (reduction done in Maya). Because of the high polygon count, modern hardware is recommended for running the application. See the below section, “Hardware Requirements,” for the recommended hardware specification.

Table 3: Output from Maya showing the polygon count of one of the assemblies. This assembly contains over 800,000 triangles, which is very high in today's standards.

Verts:	400919	0	0
Edges:	1202126	0	0
Faces:	800804	0	0
Tris:	800804	0	0
UVs:	0	0	0

Mode of Delivery

The code will be compiled into a non-debug EXE executable that can launch on any Windows 7 computer (provided the support libraries, mentioned below, have been installed). The files will be uploaded to a shared network drive, *Dropbox.com*. The museum can log in to *Dropbox.com* with the provided account details and download the program with the installation guide shown below.

Hardware Requirements

Due to the high polygon count present in the assemblies, it is recommended that the program be run on a modern computer. We recommend that the program be run on a computer that is equipped with a recent dual-core processor or higher (e.g., an Intel Core-2 Duo), 4GB or more of ram, and a dedicated graphics processor (at minimum, a NVIDIA 8800). Although this hardware is modern, it does not mean it has to cost a lot. 4GB ram can be easily obtained for $30, the NVIDIA 8800 for less than $150, and the processor for around $120.

HISTORIC CALCULATOR APPLICATION INSTALL INSTRUCTIONS

Before running the program, install the following software:

Microsoft .NET Framework 4.0:

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992&displaylang=en

Microsoft XNA Framework 4.0:

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=a88c6dec-aeae-42cd-a108-d35c013c3b97

Once these two frameworks have been installed successfully without error, run the following:

HistoricCalculatorGame\WPFGame\bin\Release\WPFGame.exe

Running this EXE should begin the program in full-screen mode. To exit the application, press the ESCAPE key on your keyboard.

PRODUCT REALIZATION

The solid models created of the BVM represent the measurements and dimensions given for each part, however the assemblies need some more work. The museum is impressed by our model even though there are a couple improvements to be made. For example, part 200 the "nine tooth cylinder" is a very complex part to model (Figure 16). The part had to be made as a separate assembly such that lines were visible in the model that do not exist on the actual part. Corrections like this are small, but are very important to the museum for two reasons: aesthetic value and having CAD models that are true representations of the machine.

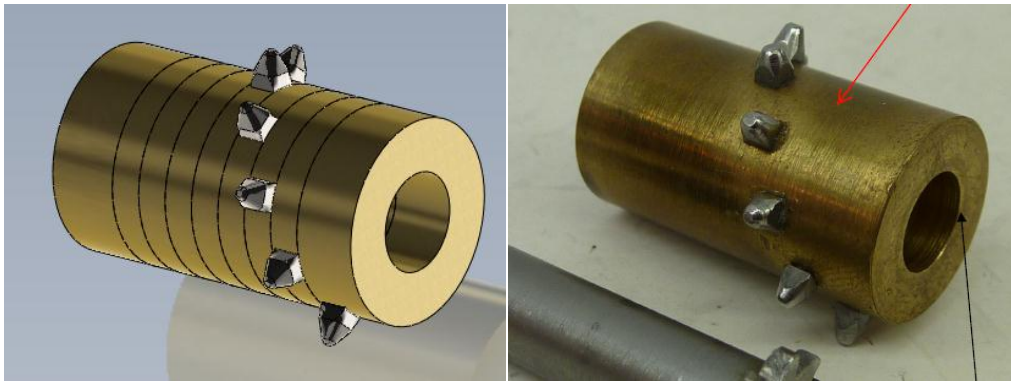


Figure 16: The lines on the modeled part (LEFT) show the slight difference between this SolidWorks model and the actual part.

The museum would eventually like to have a model that is fully defined however, such that every mate in SolidWorks is completely accurate to the original BVM. We had hoped that some motion studies could be made beyond exploded views. Further time could be put into creating interference detection in SolidWorks, as well as gear mates, and possibly spring mates. Research into these motion studies could aid in the further software development for the museum's future exhibits.

VERIFICATION OF DESIGN

ME SIDE

The SolidWorks model is verified in two ways, via the approval of Mr. Rebenyi and the software developed by the CPEs. The museum is satisfied and the software had no issues with the .STL files created.

CPE SIDE

The final software product must be verified that it indeed meets all sponsor requirements and the requirements added by the group. The program must run on the Windows environment at any time, and for any amount of time, without running into problems (crashes, exceptions, etc). The application must also be tested to run on the minimum recommended hardware (see the "Hardware Requirements" section) without encountering issues.

The first test in software testing is commonly referred to as a functional test. This includes testing the software on a compatible Windows platform and running the software with a certified Windows touch-screen monitor (the HP L2105tm was used for testing). The software was launched and all user interactions were tested to be bug-free. In addition, all language options were tested to ensure that the user interface would be truly multi-language.

The quality of the product was tested under nonfunctional testing. Responsiveness was tested in all use cases. If algorithms were found to be non-responsive, then the underlying code and algorithm were optimized. Bad responsiveness can ruin a 3D-based application, so ensuring that all code was well-optimized was a priority of the project.

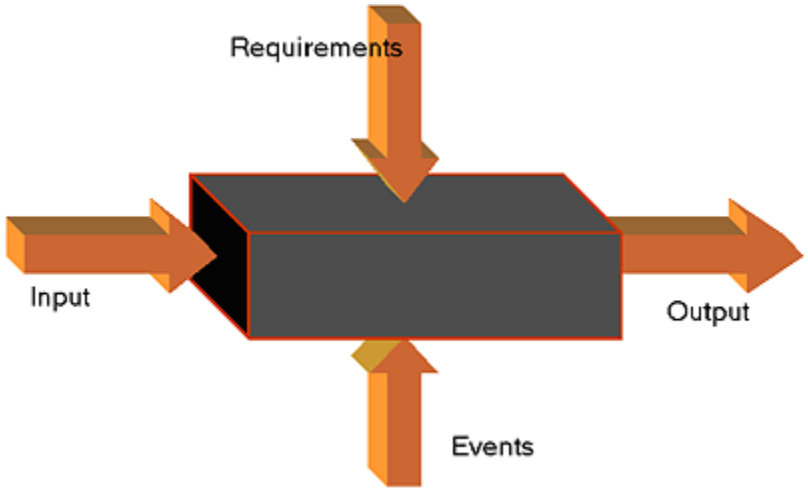


Figure 17: White Box Functional Testing

Furthermore, the application was tested using black-box testing. White-box testing was not used in our circumstance primarily because we are not providing any API's (Application Programming Interfaces) that will be available for use; black-box testing was sufficient for this program. Black-box testing is a type of testing that has the end-goal of asking the question "does this software meet the required specifications?" The final program was compared to the given application specifications, and all of the program features relating to the specifications were tested to be bug-free. Edge-cases (circumstances that rarely occur) were also tested for in this stage.

Security for this program was not of immediate concern as this application does not connect to any network nor does it receive any input except from the touch-screen interface. The museum will have to ensure that the user can only access the program through the touch-screen. The user should not have any access to the keyboard or he/she could gain access to the underlying Windows operating system. The application was tested, so that once launched, the touch-screen interface would not allow access to the operating system in any way.

CONCLUSIONS

ME PERSPECTIVE

The Historic Calculator Project has been a great opportunity for the ME team. Having undergraduate experience working on an international, multidisciplinary project is priceless. The CPEs have given us some insight into software generation and the museum has given us valuable historic knowledge. Being a part of the Deutsches Museum project is rewarding also, because the team members can go to the museum and see what they have become a part of. We hope that the Historic Calculator Project will aid the museum to create a more interactive learning environment.

The BVM model is truly impressive but there are some improvements and potential changes that could be made.

- Verify the assembly by checking interference on gears.
- Be able to turn the crank and have the other gears rotate appropriately.
- Apply images to the surfaces of the BVM that show the intricate detail of the original machine.
- Utilize the model to position the staircase in the best position to eliminate binding and improve operation of the museum replica.

CPE PERSPECTIVE

This project was a first that bridged our experiences to other engineers outside of the electrical and software arena. The major obstacle was dividing this project equally yet allowing both teams to work autonomously. By dividing the work between 1) the SolidWorks modeling, and 2) the program coding, both parties maximize their abilities and contributions to the project. Our final software proved to be interactive for users to rotate and zoom in on the models, but the following upgrades could improve the experience for museum visitors.

- The program could color-code the different parts of each assembly to highlight important areas of each model. This would make the captions even more useful, as something like “the *green* gear turns the *red* gear which does…” could be said.
- The program could employ narratives, rather than captions, to guide the user through the functionality of each sub-assembly.
- The program could support multi-touch finger gestures, rather than single-finger gestures, to support more intuitive interactions (like “pinch to zoom”).
- Graphical features, such as reflection, could be employed to make the environment in which the models are displayed in more immersive and believable. Shaders (implemented in the HLSL shader language) which provide a metallic characteristic to the assemblies would also be a nice touch.
- Create an animated model that executes mathematical operations just like the real BVM.
- Use vocals along with text in the software to explain the models and inner workings of the BVM.

References:

[1] “History of Computers and Computing” Mechanical Calculators, 18th Century. Georgi Dalakov. Feb. 3, 2011. <http://historycomputer.com/Mechanical Calculators/18thCentury>

[2] Henriksen, K., J. Sporring, and K. Hornbaek. "Virtual Trackballs Revisited." *IEEE Transactions on Visualization and Computer Graphics* 10.2 (2004): 206-16. Print.

Braun Vayringe Parts List

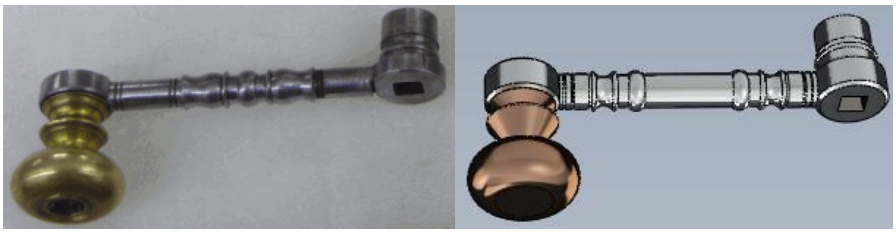
Part Range	Description	
100-199	Inner Circle	Crank, first gears
200-299	Second Circle	Inputs
300-399	Thrid Circle	Outputs
NOTES:	number increases with level - top to bottom. Part dwgs must be done. BLANK means that part has not been started yet. gears are based off of part 109	
Legend	**ip**	in progress (NAME)
	y	yes-complete
		needs dimensions

Part #	PART NAME (Description)	SolidWorks	Appendix	Part DWG	Dimensions	Notes
100	crank	y (D)			Picture 1	
101	center shaft	Y (W)			Picture 2	
102	inner dial	ip (D)			d=10,1_D=39,8_t=2,5	needs numbers
103	pin	y (D)			d=2,4x3,6_D=5x6	
104	pin holder	y (D)			D=50,5_D=40,4x2,8_d=19,3_t=4	
105	first gear	y (D)			z=30_ Dk=38,7_df=34,3_d=18,8_t=3,6	
106	small gray gear	y (D)			z=6_Dk=9,7_df=5,2_t=3,15_Ø4,4x3,1_□3,96x7,25	
107	small gear pin	Y (W)			Ø2,5x6,85_Ø3x3	
108	housing	y (W)			see BVM PICS	Changed slot from 37.8 to 38.8 to make gear fit
109	overhanging gear	y (D)			z=24_Dk=38,25_df=32,40_t=3,1_□4,0	fixed and uploaded to parts
110	bottom gear	y (D)			z=36_ Dk=52,47_df=46,33_d=8,0_t=3,6	
100 Assembly	100 Assembly	ip(W)				Needs some work (gear meshing etc.)
200	9 Tooth Cylinder	y (W)	x	y (W)		spiral teeth -sent TR
201	number dial	ip (D)	x			needs graphic numbers
202	indexer	y (D)	x	y (D)		jpg done - sent TR
203	9T base	y (W)	x			
210	inner base	y (D)	x	y (D)		need more pics, sent to TR, needs screws, coil springs, mounting holes
211	stair driving gear	y (D)	x		Dk=26,94_df=20,78_ t=3,7_ z=18_ (hole Ø5,0_ for screwhead Ø6,5x1,2_other side for position Ø12,0x1,46)	
212	index spring	y (TR)	x			(TR) I had to draw it in SW, because I want to make them a new for the BVM in the Museum
213	screw for driving gear	y (TR)	x			~~wrote TR about this one~~
214	212 ratcheting block	y (W)	x			
220	Stair Case	y(D)	x			
221	stair case holder	y(W)	x			

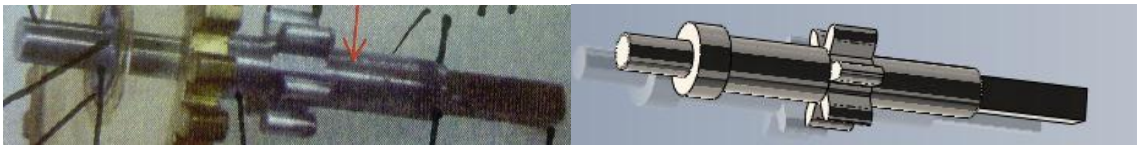
213	screw for driving gear	y (TR)	x			~~wrote TR about this one~~
214	212 ratcheting block	y (W)	x			
220	Stair Case	y(D)	x			
221	stair case holder	y(W)	x			
222	set screw	y(D)	x		Ø8,62x2,35_Ø4,73x3_M3x4	no threads
223	Lever Spring	y(D)	x			(TR) can you take the measurements out of the picture?
224	Tolerance Pin	y(D)	x		Ø6,15x1,9_Ø3,48x3_M3x4	
225	ring gear	y(D)	x			measurements in pictures
230	dial face	TR				
200 Assembly	200 Assembly	y(W)	o			
300	Outside Dial	y(D)	x			needs numbers - is there a square cut in the bottom?
301	outside dial Shaft	y(D)	x			unsure about the overall length of the shaft and position of taper.
301_11	outside dial Shaft special	y(D)	x			why the offset??
302	ODD Dual Gear	W	x		z=10_Dk=16_df=11_t=2,5	
303	EVEN Dual Gear	W	x		z=10_Dk=27_df=18_t=2,0	
305	inner base turning shaft	TR	x			
306	shaft position disc	TR	x			
307	hand	TR	x			
308	disc spring	TR	x			
310	ODD Carryover Column	y(D)	x			
311	EVEN Carryover Column	y(D)	x			
312_1	Carry over gear	y(D)	x		Dk=27,0_df=17,9_z=10_t=2,3 carryover-tooth:R=17,0x2,3_D=13,0_d=7,0_t=1,2	
312_2	Carry over gear	y(D)	x			
312_3	Carry over gear	y(D)	x			
312_4	Carry over gear	y(D)	x			
312_5	Carry over gear	y(D)	x			
312_6	Carry over gear	y(D)	x			
312_7	Carry over gear	y(D)	x			
312_8	Carry over gear	y(D)	x			
312_9	Carry over gear	y(D)	x			
312_10	Carry over gear	y(D)	x			
313	Outer linear Springs	y(D)	x			
314	Set Screws	y(D)	x			
316	ratchet block	y(D)	x			
317	ratchet screw	y(D)	x			
350	Main base plate	TR	x			
351	main base ring	TR	x			
352	Foot	TR	x			
360	dial ring	TR	x			
370	dial face	TR	x			
373	Key	DM	x			

APPENDIX: PARTS LIST

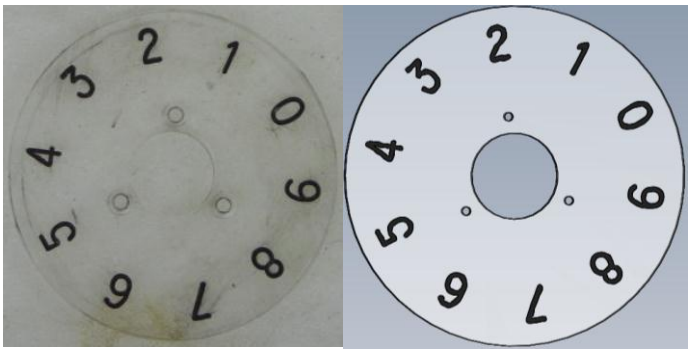
PART 100



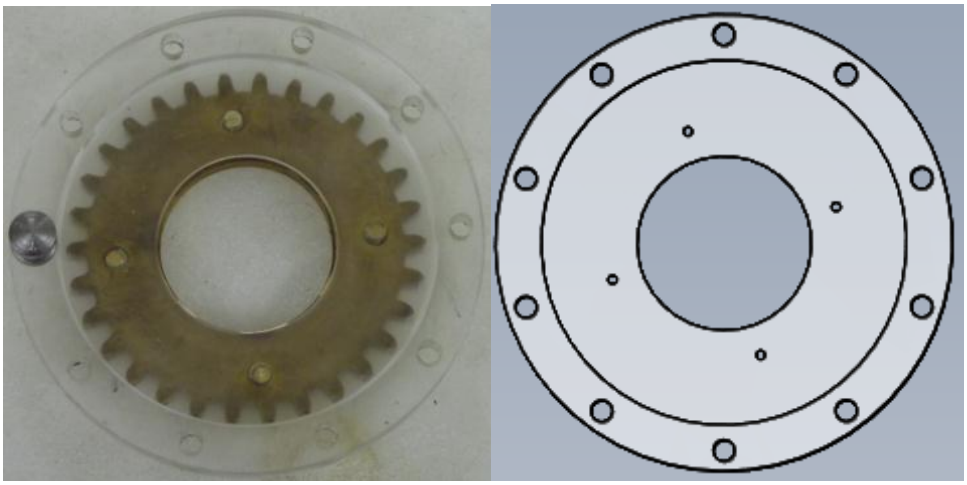
PART 101



PART 102

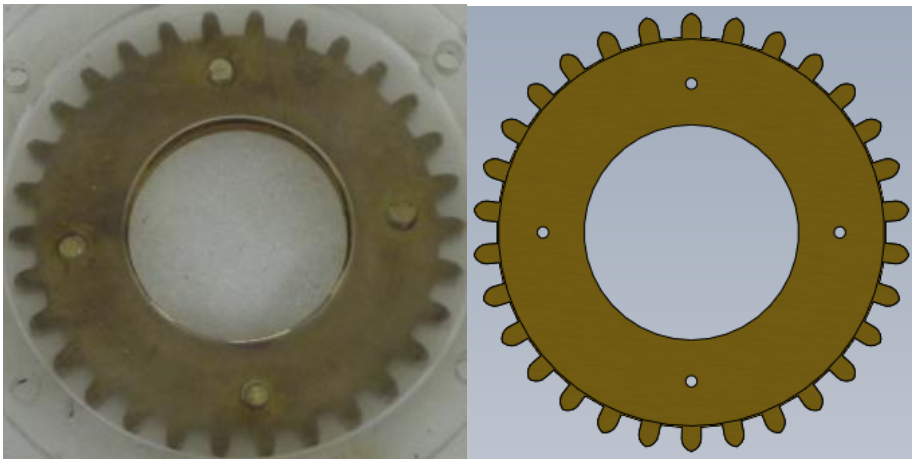


PART 104

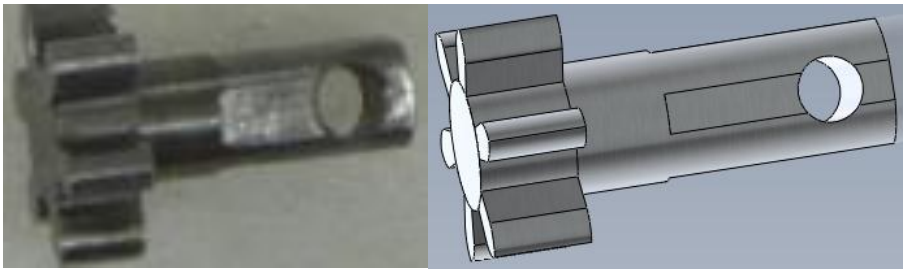


APPENDIX: PARTS LIST

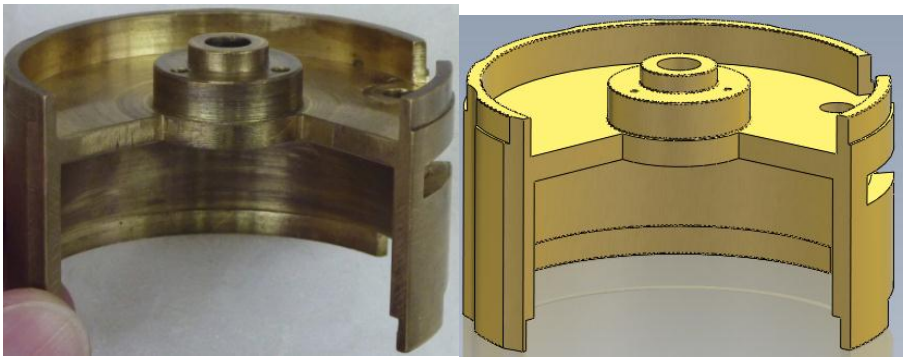
PART 105



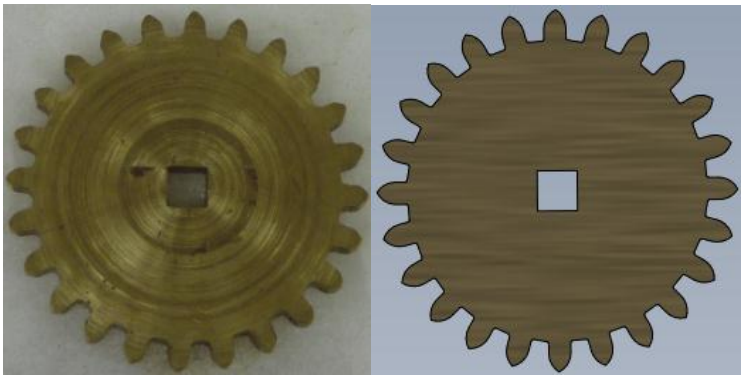
PART 106



PART 108

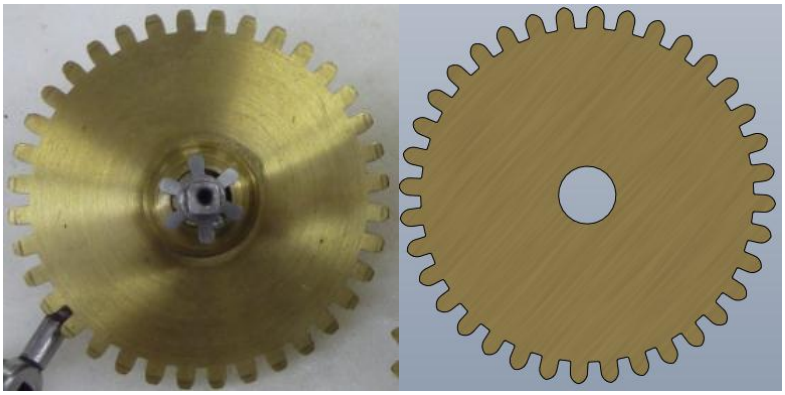


PART 109

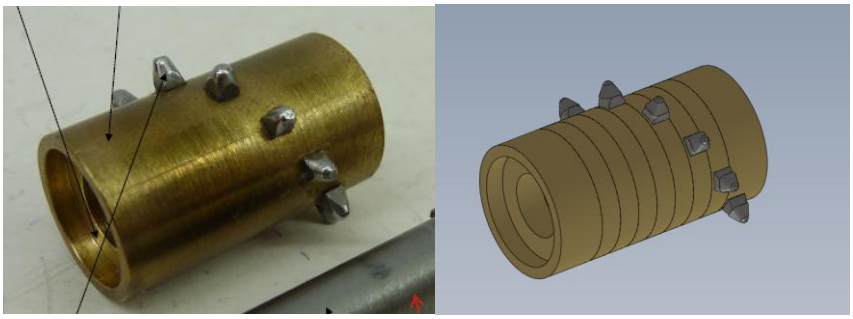


APPENDIX: PARTS LIST

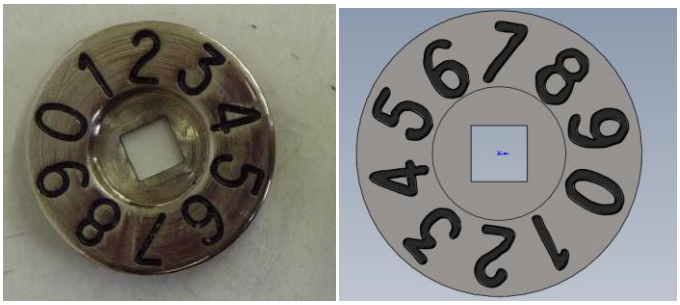
PART 110



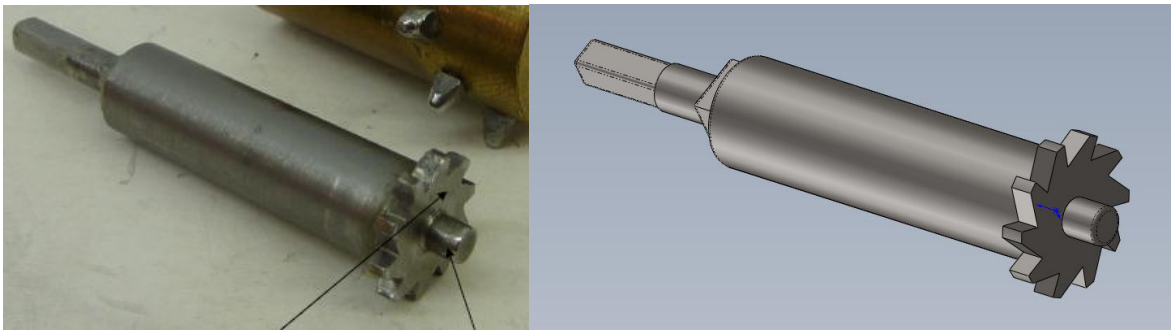
PART 200



PART 201

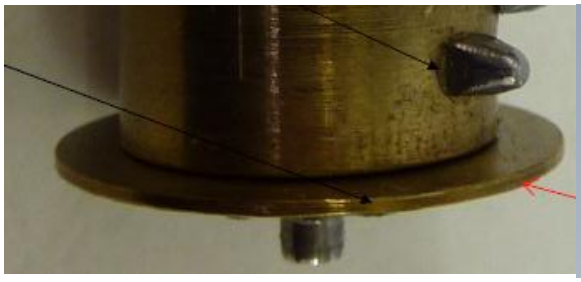
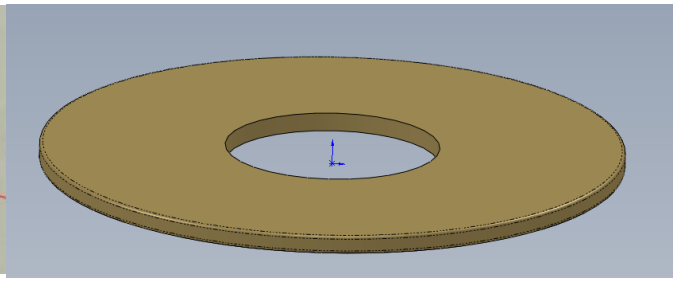


PART 202

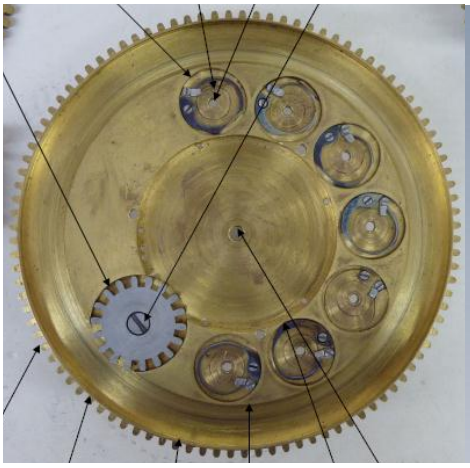
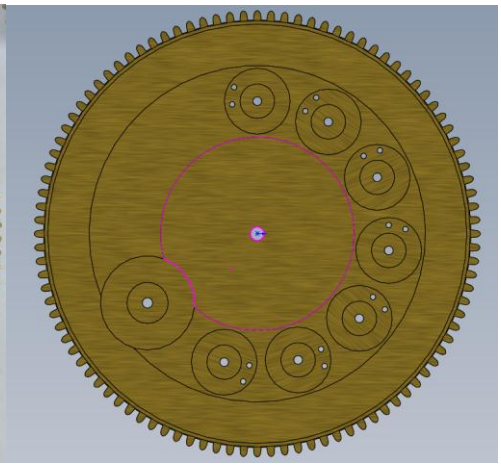


APPENDIX: PARTS LIST

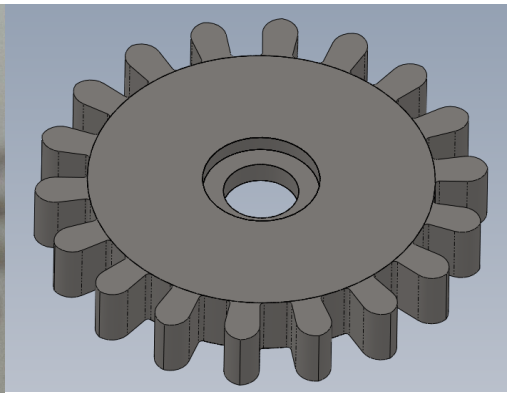
PART 203

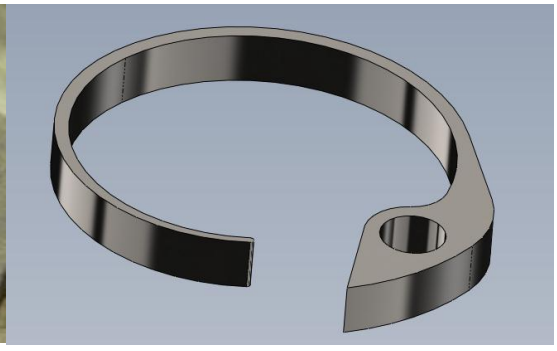
PART 210

PART 211

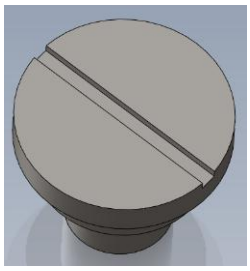



PART 212

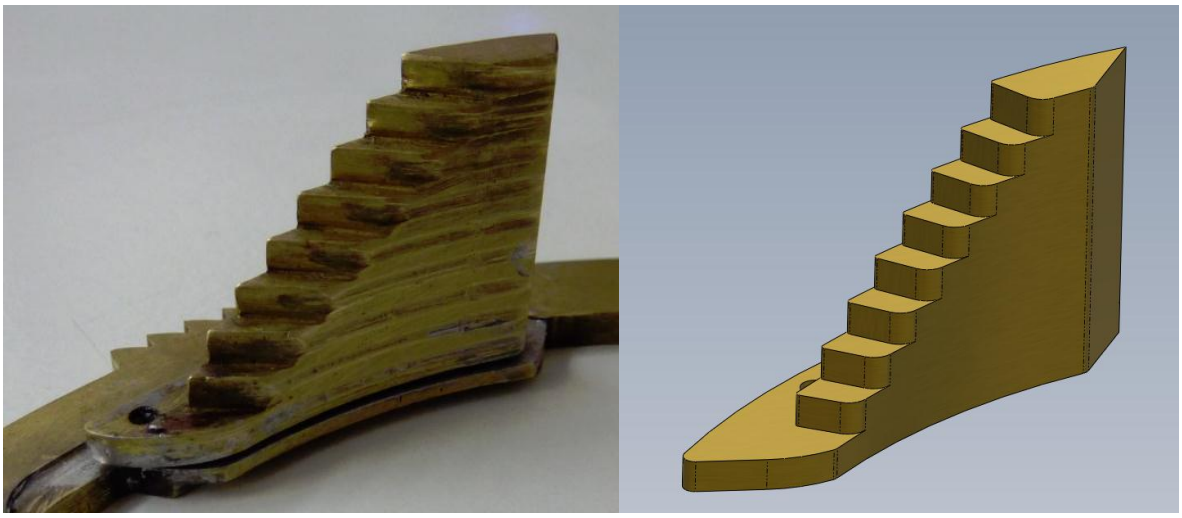
APPENDIX: PARTS LIST

PART 213

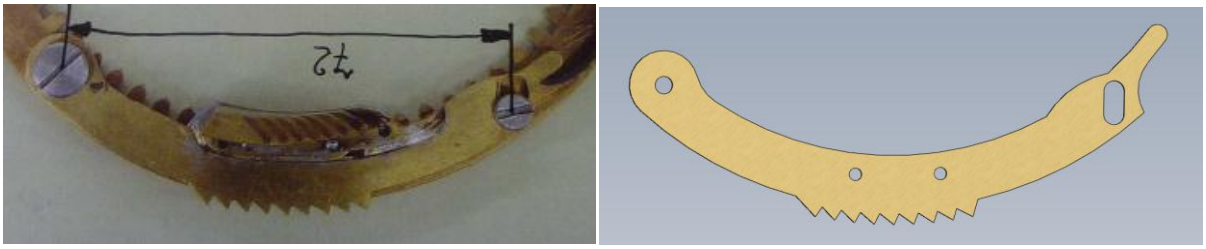



PART 214

PART 220

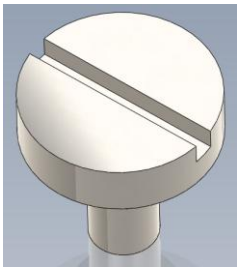


PART 221

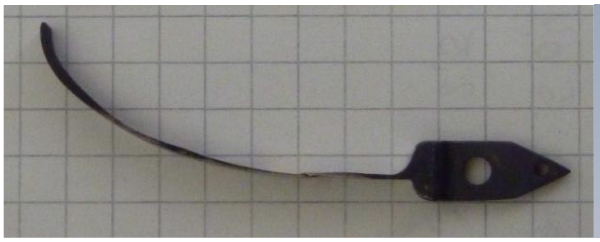
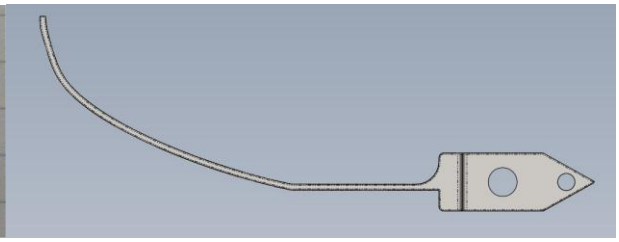


APPENDIX: PARTS LIST

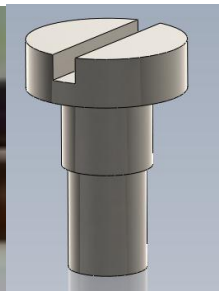
PART 222

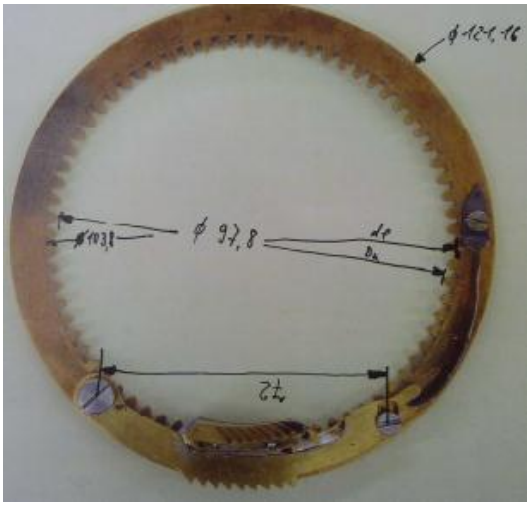
PART 223

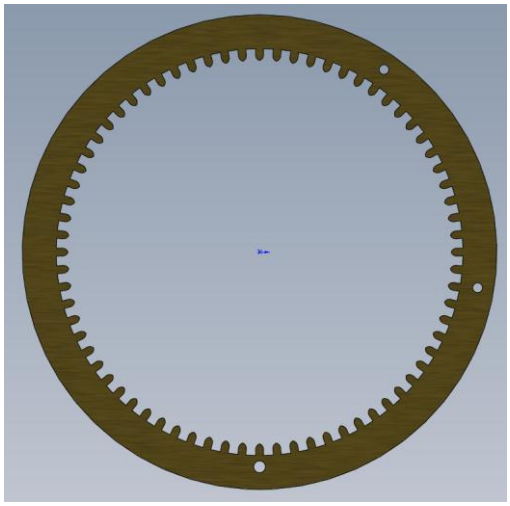



PART 224

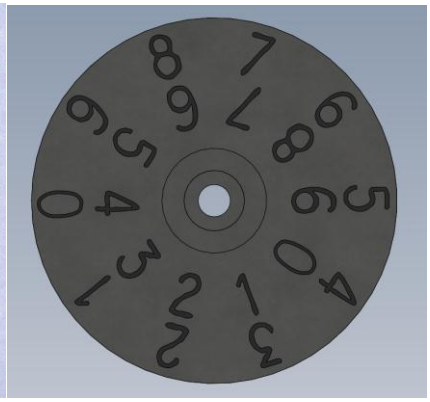
PART 225



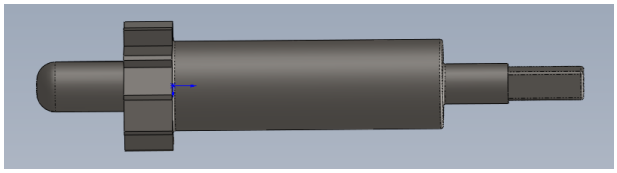


APPENDIX: PARTS LIST

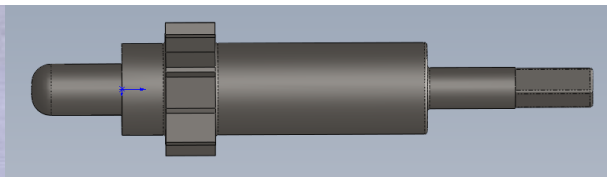
PART 300

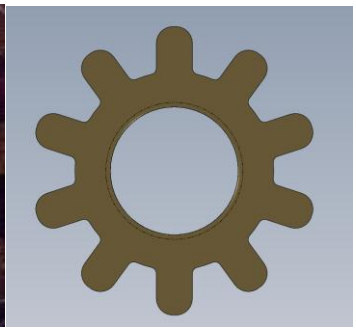
PART 301

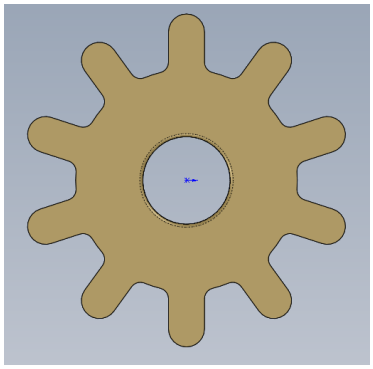
PART 301_11

PART 302

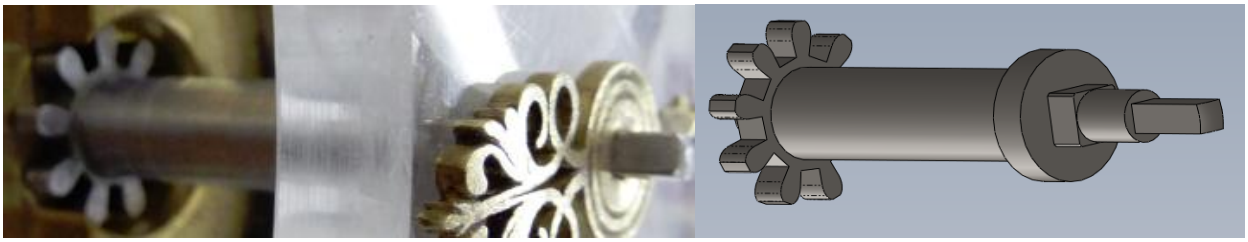



PART 303

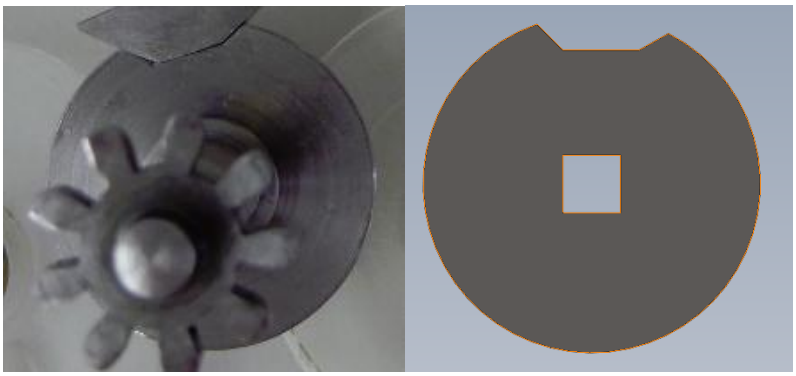



APPENDIX: PARTS LIST

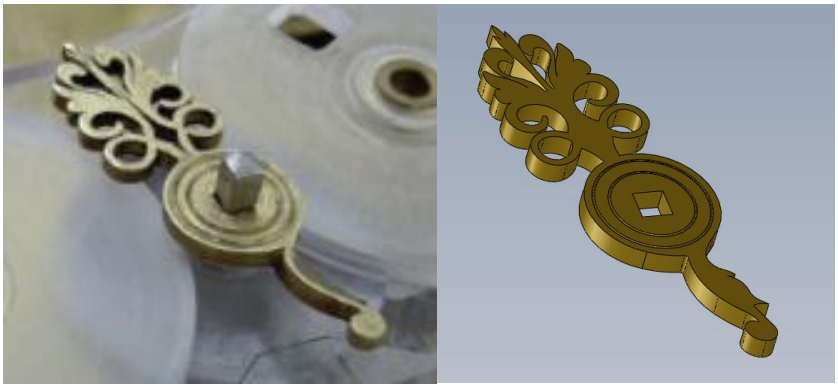
PART 305



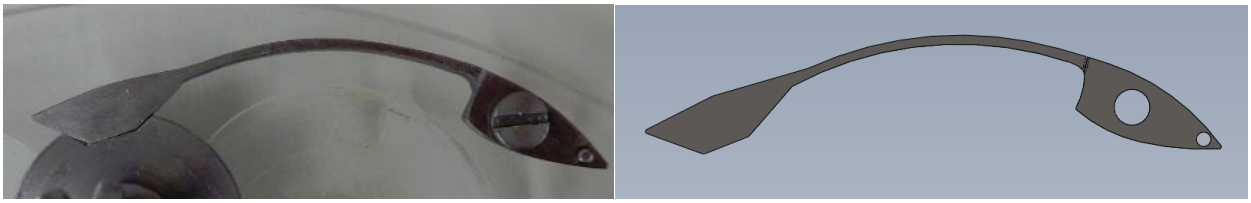
PART 306



PART 307

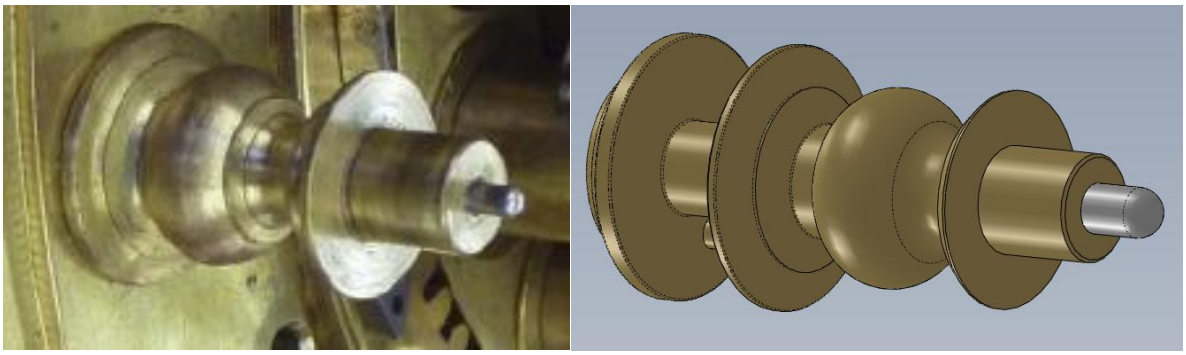


PART 308

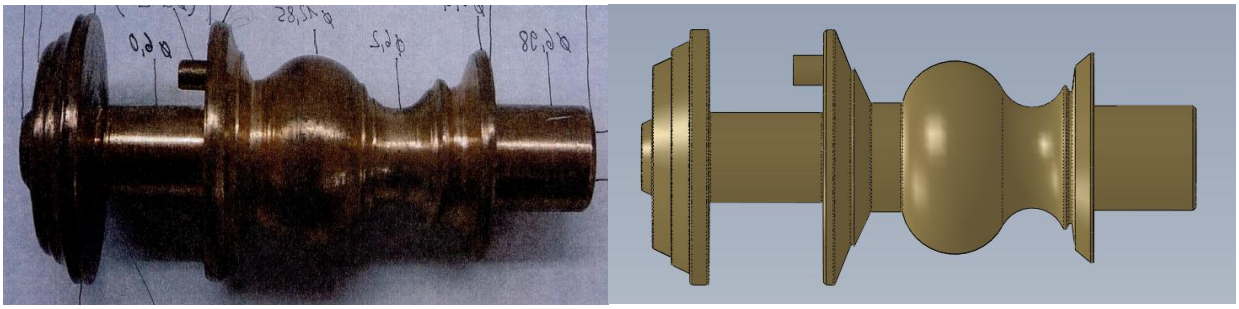


APPENDIX: PARTS LIST

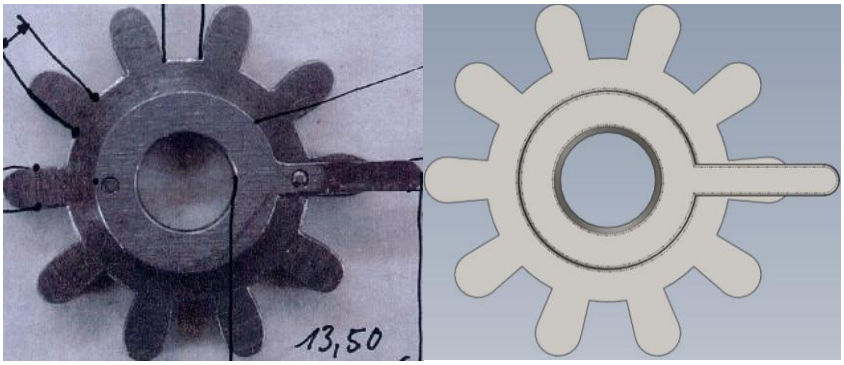
PART 310



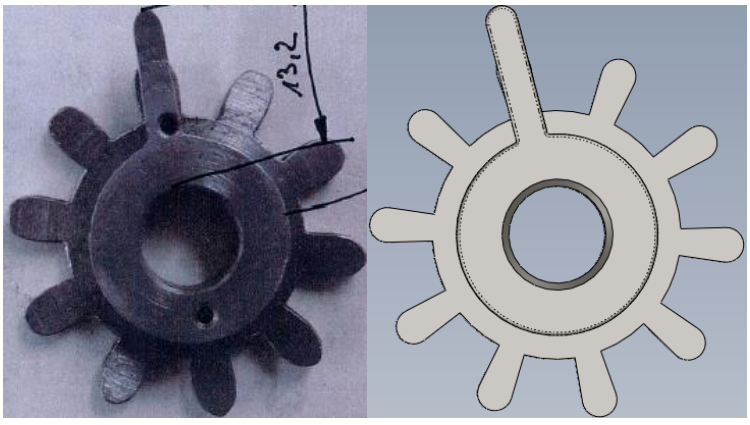
PART 311



PART 312_1

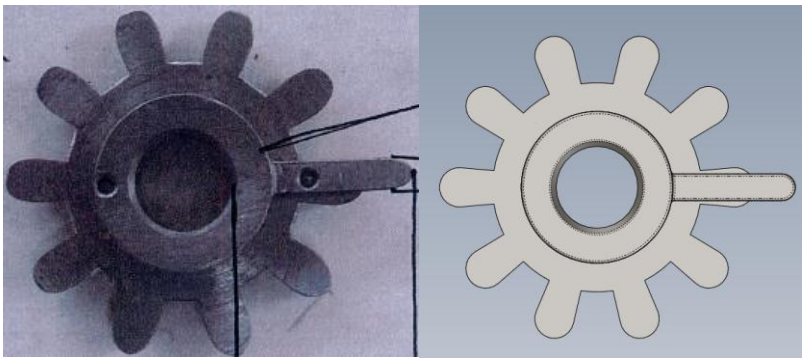


PART 312_2

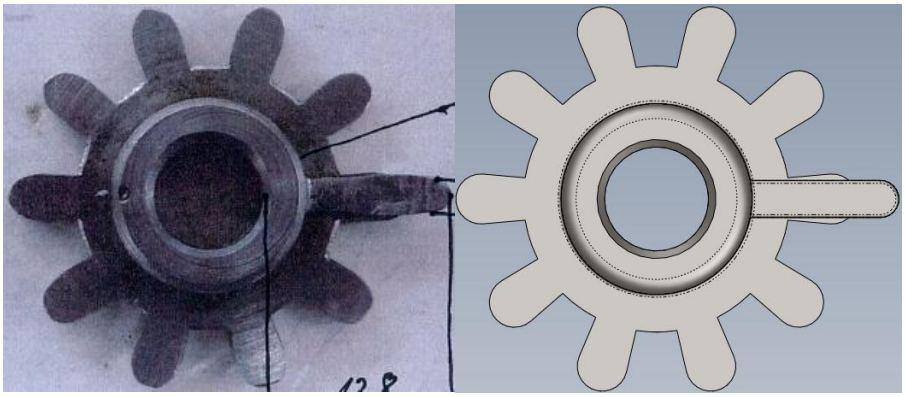


APPENDIX: PARTS LIST

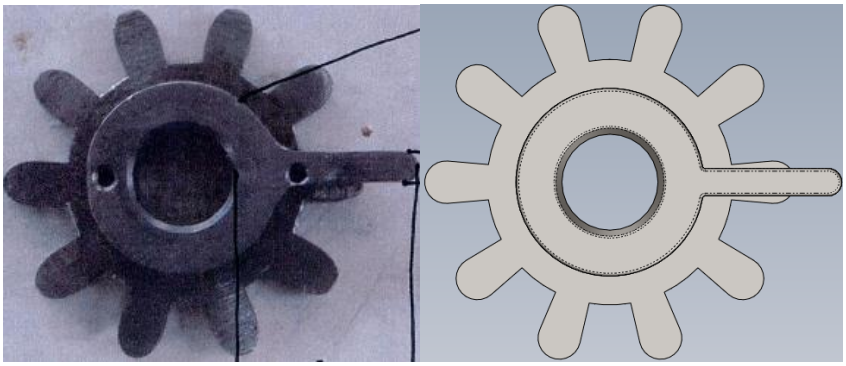
PART 312_3



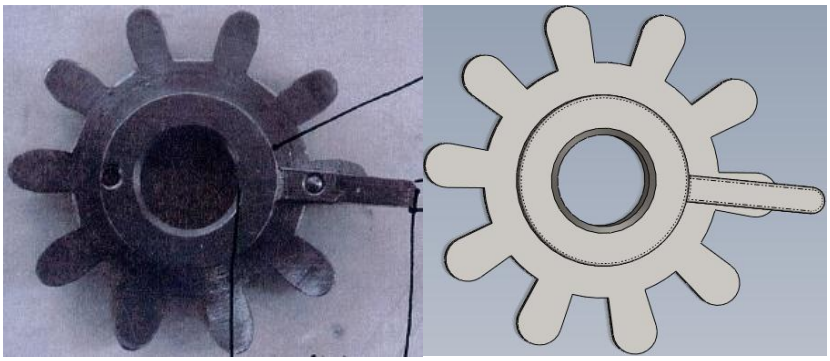
PART 312_4



PART 312_5

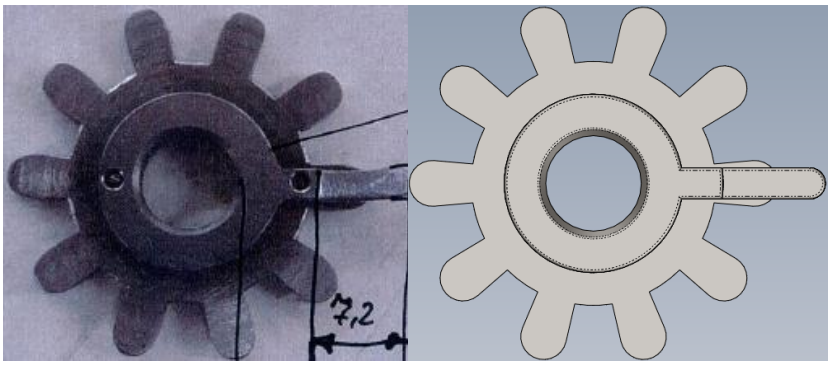


PART 312_6

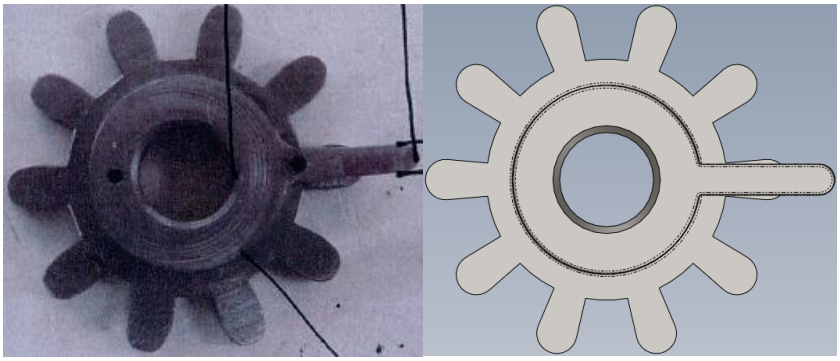


APPENDIX: PARTS LIST

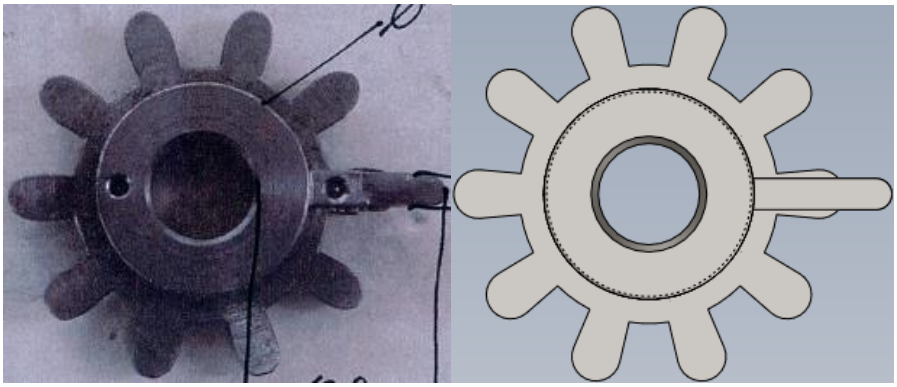
PART 312_7



PART 312_8

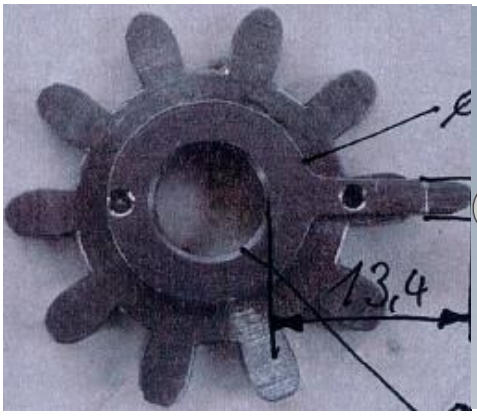


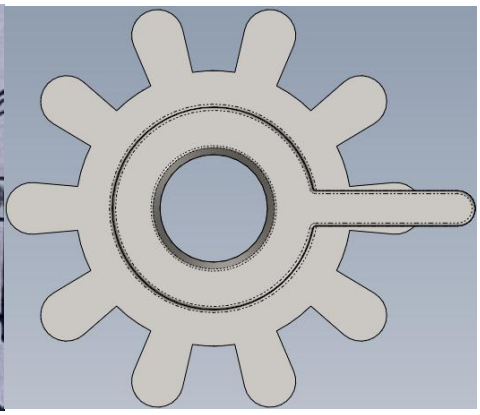
PART 312_9



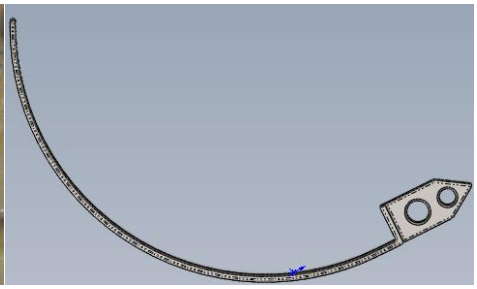
APPENDIX: PARTS LIST

PART 312_10

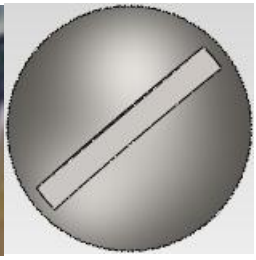




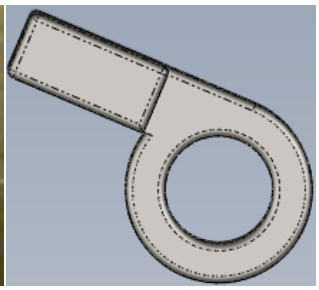
PART 313

PART 314

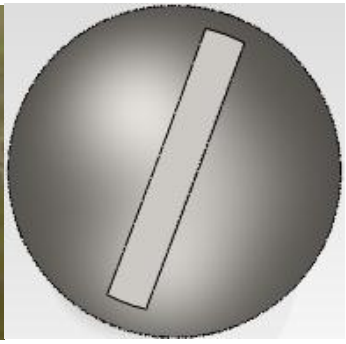



PART 316

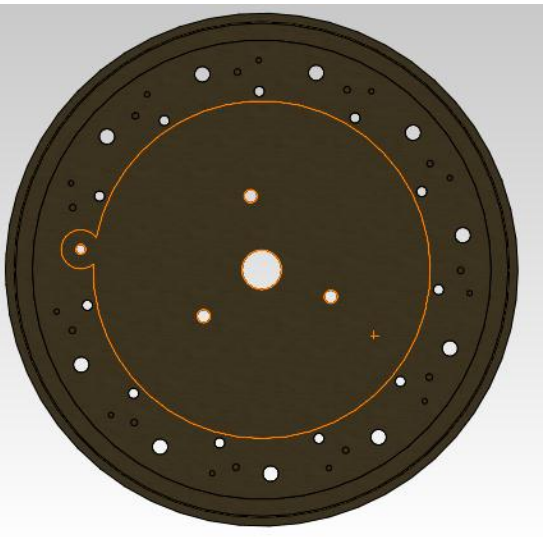



APPENDIX: PARTS LIST

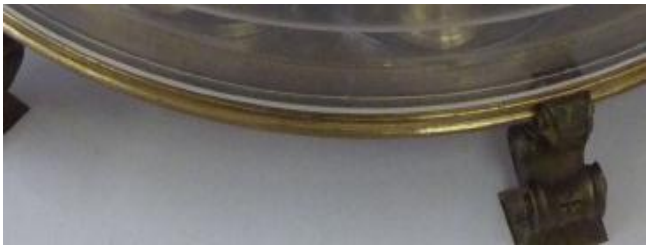
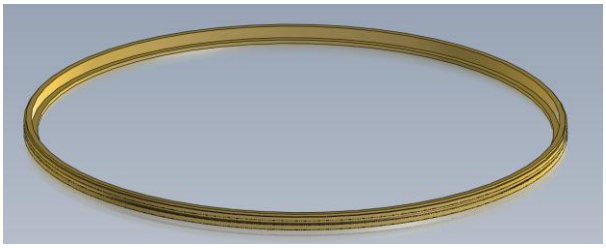
PART 317

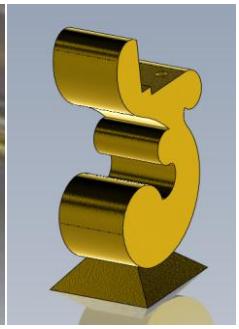
PART 350

PART 351

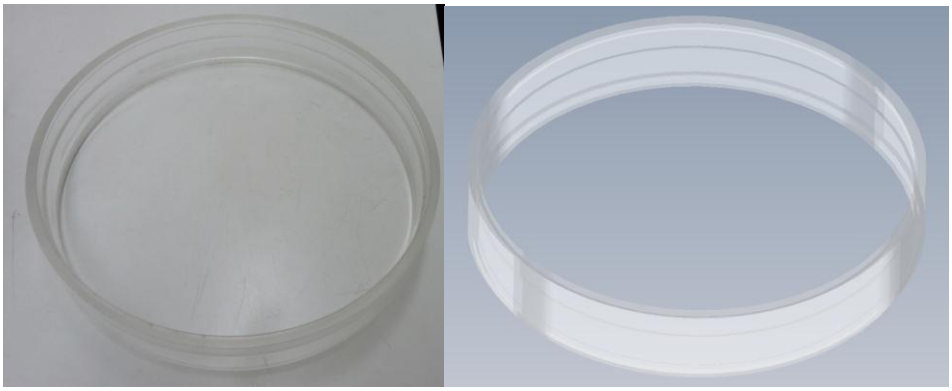



PART 352

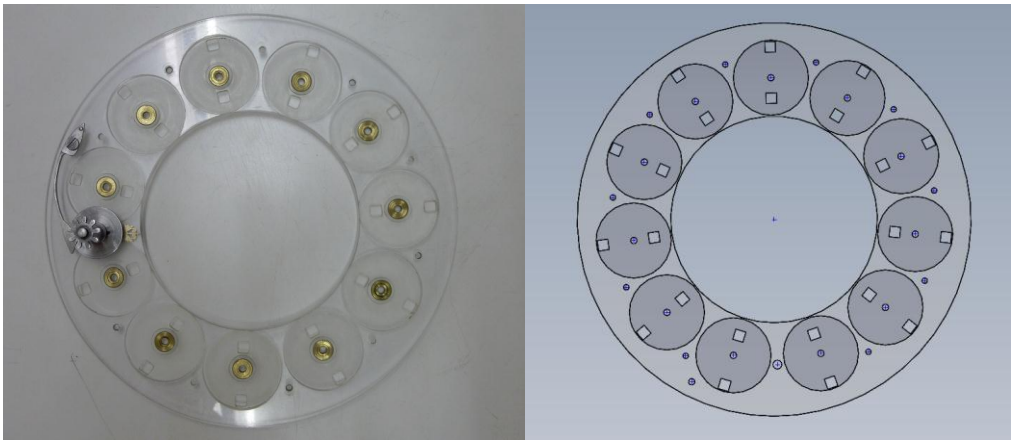



APPENDIX: PARTS LIST

PART 360



PART 370



PART 373