Title

Modern Materials and Technology in Service of the Carillon

A THESIS SUBMITTED TO THE FACULTY OF THE KONINKLIJKE BEIAARDSCHOOL ‘JEF DENYN’

KEIRAN CANTILINA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE END EXAM AND DIPLOMA

ADVISOR: KOEN VAN ASSCHE

MAY 2019

Copyright Page

© 2019

Keiran Cantilina

ALL RIGHTS RESERVED

Acknowledgements

Many thanks to my advisor Koen Van Assche for supporting and promoting this project. Without the nearly unlimited access he provided to the carillon of Lier, this project would have been impossible. I’m also thankful for his advice, hands-on help, and cups of tea which made the cold, greasy parts of this project much more bearable.

Thanks to Eddy Mariën for his support and ideas during the early phases of this project, to Koen Cosaert for interpersonal and life advice, and to Hanna Lin for convincing me that switching from OpenSCAD to Onshape would save my sanity.

Thanks to Alexius Jullien and Henricus Joltrain for creating a fascinating machine 300 years ago. It’s been an amazing opportunity studying their handiwork.

Last but not least, I’m immensely thankful for the support of my wonderful wife Ren. Thanks for having my back.

Dedication

This thesis is dedicated to my wonderful wife Ren, who has my sincere and everlasting appreciation. Her encouragement and support has made this thesis possible without the loss of my sanity. I am thankful for her immense patience through everything from clothes covered in 18th-century machine grease to 3D-printer noises in the wee hours of the morning.

Abstract

With the exception of incremental refinements, the carillon has experienced few changes over the last 300 years. Perhaps the time is ripe to try something new! This thesis explores ways modern materials and technology can be used to benefit the carillon using a project as an example. The project demonstrates the use of 3D printed materials and modern computational tools to aid calibration of a historical automatic playing mechanism.

Table of Contents

[List of Figures vi](#_Toc7056259)

[Introduction 1](#_Toc7056260)

[Background 2](#_Toc7056261)

[Methods 3](#_Toc7056262)

[Methods I: Project Setup 3](#_Toc7056263)

[Methods II: Development of Calibration Protocol 6](#_Toc7056264)

[Methods III: Implementation of Perfect 3D Printed Pins 7](#_Toc7056265)

[Methods IV: Implementation of Calibration Tune 10](#_Toc7056266)

[Methods V: Implementation of Software 12](#_Toc7056267)

[Methods VI: Putting It All Together 14](#_Toc7056268)

[Results and Data 15](#_Toc7056269)

[Analysis and Discussion 20](#_Toc7056270)

[Part I: Performance of 3D Printed Pins 20](#_Toc7056271)

[Part II: Analysis of Calibration Data 25](#_Toc7056272)

[Appendix 32](#_Toc7056273)

[Pictures 32](#_Toc7056274)

[Project Github Repository 34](#_Toc7056275)

[Autospeelwerk Calibration Analysis Program 34](#_Toc7056276)

[Calibration Tune Musescore file 37](#_Toc7056277)

[Calibration Tune ABC text file 37](#_Toc7056278)

[Calibration Recording 38](#_Toc7056279)

[3D Printed Pins CAD drawings 38](#_Toc7056280)

[Drum measurement CAD drawings 38](#_Toc7056281)

# List of Figures

[Figure 1: Measured drum dimensions. 3](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056638)

[Figure 2: Pin 5 average dimensions 4](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056639)

[Figure 3: FEA analysis of first prototype pin 9](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056640)

[Figure 4: New pin design 9](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056641)

[Figure 5: FEA analysis of new pin design 9](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056642)

[Figure 6: Clipping pin design 10](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056643)

[Figure 7: Human-readable form of the tune 11](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056644)

[Figure 8: Program function diagram 13](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056645)

[Figure 9: Example of program output 13](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056646)

[Figure 10: Time between measures over the course of the calibration tune 15](#_Toc7056647)

[Figure 11: Histogram showing distribution of rhythmic error 15](#_Toc7056648)

[Figure 12: Error over the course of the calibration tune 16](#_Toc7056649)

[Figure 13: Calibration plot of error over carillon range 16](#_Toc7056650)

[Figure 14: Time between measures over the course of the first 40 notes of the calibration tune 17](#_Toc7056651)

[Figure 15: Histogram showing distribution of error over the first 40 notes of the calibration tune 17](#_Toc7056652)

[Figure 16: Error over the course of the first 40 notes of the calibration tune 18](#_Toc7056653)

[Figure 17: Calibration plot of error over carillon range, using only data from the first 40 notes of the calibration recording (missing values removed) 18](#_Toc7056654)

[Figure 18: Figure 12 with approx. the first 40 notes of the calibration tune circled 19](#_Toc7056655)

[Figure 19: 2nd Pin #5 (left) and Pin #1 designs 20](file:///C:\Users\Keiran\Desktop\Carillon%20Thesis%20Composition\Carillon%20autospeelwerk%20project\Writeup\BEIAARDSCHOOL%20THESIS%20DRAFT.docx#_Toc7056656)

[Figure 20: Illustration of multiple regression lines used to model drum speed error 28](#_Toc7056657)

[Figure 21: Blue lines are examples of residuals that show the true rhythmic error due to adjustment 29](#_Toc7056658)

[Figure 22: St. Gummarus Autospeelwerk 32](#_Toc7056659)

[Figure 23: The pinning process 32](#_Toc7056660)

[Figure 24: Cable adjustment screws (left) and view of a clipping pin from inside the drum (right) 33](#_Toc7056661)

[Figure 25: Close-up view of pins 33](#_Toc7056662)

[Figure 26: Sketchbook 34](#_Toc7056663)

# Introduction

Given its grand size, superior listening range, and the unique beauty of its sound, the carillon is arguably the king of instruments. With its origin in the 16th century, the carillon predates even the piano. Among keyboard instruments frequently played today, the carillon is preceded in history only by the pipe organ, harpsichord, and clavichord. However, unlike the piano and pipe organ, the design of the modern carillon is substantially similar to a carillon one might find in the 1700s. With the exception of incremental refinements in tuning (which serve to ease the ear of the listener) and modernizations of the transmission design (which serve to ease the arm of the carillonist), the carillon has experienced few changes over the last 300 years. Perhaps the time is ripe to try something new!

This thesis explores ways modern materials and technology can be used to benefit the carillon. Described herein is a project demonstrating the use of 3D printed materials and modern computational tools to aid calibration of a historical autospeelwerk (automatic playing mechanism), followed by a discussion evaluating the results and proposing improvements.

# Background

Historical mechanical playing drums often have a reputation for playing with rhythmic inconsistencies. In some severe cases, imprecisions in the mechanism are often sufficiently large as to render the programmed tune nearly unrecognizable. These imprecisions are due to the (by today’s standards) crude manufacturing methods and limited precision of measurement available to craftsmen in the past. Every component in the autospeel system is imprecise in a slightly different way, and therefore every note in the tune sounds with a slightly different rhythmic error. Luckily, many historical drums were built with adjustment mechanisms designed to allow a degree of manual error correction. Unfortunately, because there are so many sources of error all mixed together in the autospeelwerk, it is difficult to perceive the effect of any small change made using the adjustment system. Thus, typical use of these adjustment mechanisms has required trial-and-error processes that are exceedingly tedious for carillonists (and for the involuntary audience living close to the carillon).

This project demonstrates a strategy to improve adjustment of the autospeel mechanism to make better use of the carillonist’s time. The main method consists of accurately characterizing the errors inherent in the autospeel system by isolating (as much as possible) the sources of error individually. This then allows the carillonist to make informed adjustments to the system, or at the very least enables a detailed understanding of any uncorrectable sources of rhythmic error.

# Methods

## Methods I: Project Setup

Because the autospeelwerk of the Sint-Romboutstoren in Mechelen was under restoration during the research period of this thesis, the autospeelwerk of the Sint-Gummarustoren in Lier was used as a model and experimental platform instead (see pictures in Appendix).

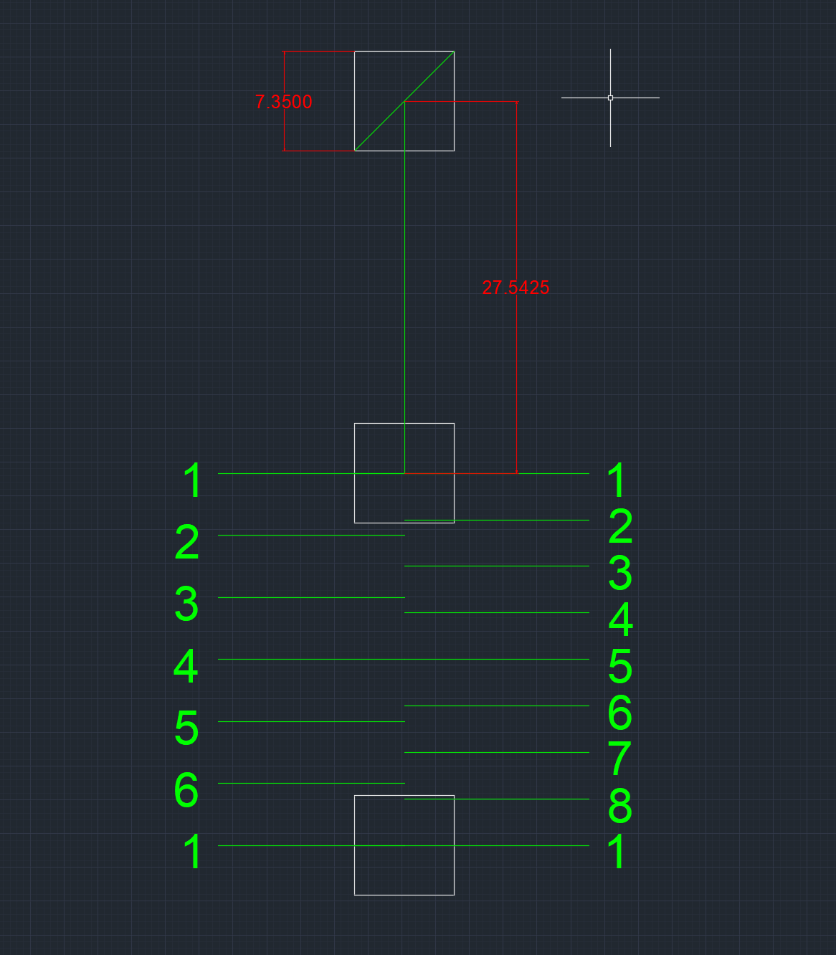
 The project began with detailed measurements made of the autospeelwerk drum and pins, followed by close observation of the mechanism in order to understand how it works.

Figure : Measured drum dimensions.

The drum was found to have 148 pin sockets per rotation. Since the drum in Lier is a “jumping drum”, the effective number of usable drum measures is doubled (296). The average space between drum measures and the average socket dimensions were also found (see Figure 1. White boxes are the holes in the drum used as sockets for the pins). The extant pins used for the drum come in 8 sizes, which means each drum measure can be divided into 8th notes (see green markings on the right side of Figure 1).

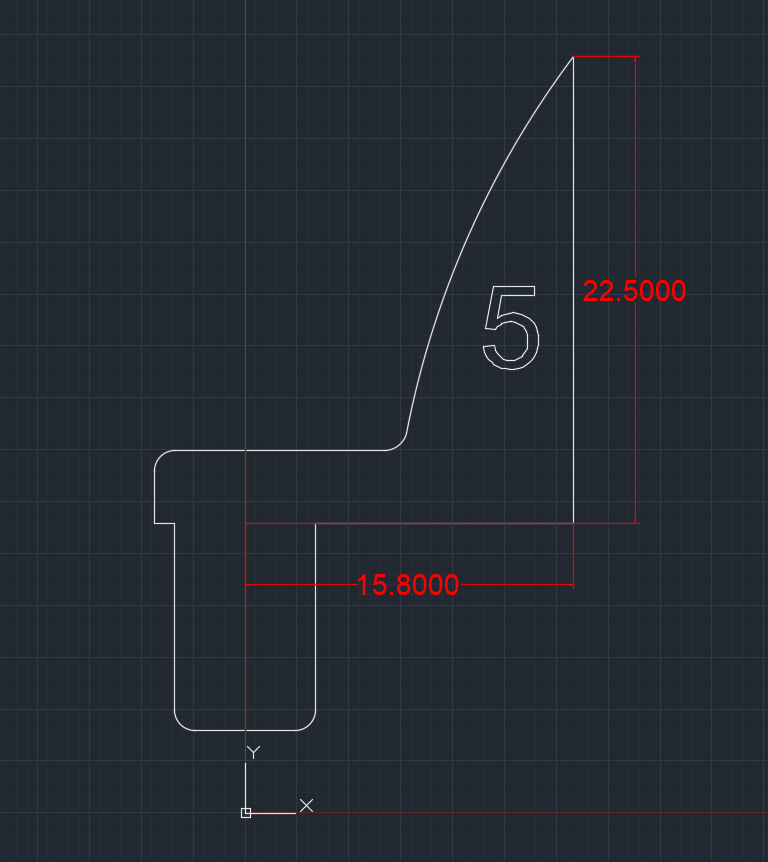
 The pins were also measured to establish average dimensions and to get an idea of dimensional variance. It was found that the pin dimensions can vary by more than 10%, and so it was assumed that this variation was a major contributor to the overall rhythmic error of the system.

Figure : Pin 5 average dimensions

As an aside from the tedium of measurement-taking, an attempt was made to estimate the power consumption of the autospeelwerk. It was found that the weight powering the system is 496 kg, and the weight falls about 2 meters during a complete rotation of the drum. This leaves 9728.2 Joules of energy for the drum to operate over one rotation. If we estimate that the drum takes about 5 minutes to complete one rotation, this means that the autospeelwerk consumes something in the vicinity of 30 Watts (0.04 horsepower) of mechanical power while running. This is quite amazing for machinery designed and built in the 18th century! For comparison, a Schulmerich Mark V Electric Carillon from the 1960s consumed about 1300 Watts (1.74 horsepower) of electrical power when running[[1]](#footnote-1). Although comparing the two systems is very much an apples vs. oranges sort of false equivalency, one can claim that the Lier autospeelwerk probably consumes less power while running.

After this close examination of the mechanism, it was decided that the two greatest sources of rhythmic error were:

* Inconsistent length of the transmission cables leading from the “pin-reading” levers to the bell hammers
* Inconsistent pin dimensions

Some other (probably lesser) sources of error:

* Bell hammer leaf spring tension and position
* Imperfect roundness of the drum
* Imperfect weight distribution of the drum (caused by inconsistent thickness of the drum wall)
* Inconsistent roundness and/or lubrication of gears in the drum geartrain
* Bearing friction
* Damage and wear to various parts of the drum mechanism

Given limitations of project resources, it was decided that the scope of the project would address the first source of error (transmission cables) because this part of the mechanism is the only part provided with manual adjustment screws. The calibration protocol would try to isolate this error from the other sources of error in the drum system.

## Methods II: Development of Calibration Protocol

Since the target of calibration was the transmission length, a method had to be developed to precisely quantify the error in this parameter with as little interference as possible from the other sources of error. The biggest potential source of confusion was probably the inconsistent historical pins. The obvious solution here was to not use the historical pins at all, and instead use specially made pins with much more perfect dimensions.

With these special pins, most of the autospeelwerk’s rhythmic error would be from inconsistencies in the transmission length. If the drum were to be set with a song (a calibration tune) that uses every bell (and thus every transmission cable) at least once, it should be possible to compare the expected timing/rhythm of each note with the actual timing played by the drum. The difference between the observed and expected timing values would yield the adjustment error for each transmission cable.

However, it would be quite difficult to measure this timing error by ear without having to play the calibration tune over and over again. Instead, a computer can be used to listen to a recording of the drum playing the calibration tune, compare observed and expected note rhythms, and output the error of each bell.

In summary, the calibration method would consist of these steps:

* Design a calibration tune that uses every note. This tune would preferably sound musical so as not to annoy the local audience.
* Make special, very precise pins
* Set a calibration tune on the drum using the special pins
* Record the drum playing the calibration tune
* Write and run code to have a computer compare the calibration tune score to the recording
* Computer outputs timing error of each bell in milliseconds
* Transmission cables can be adjusted based on outputted data

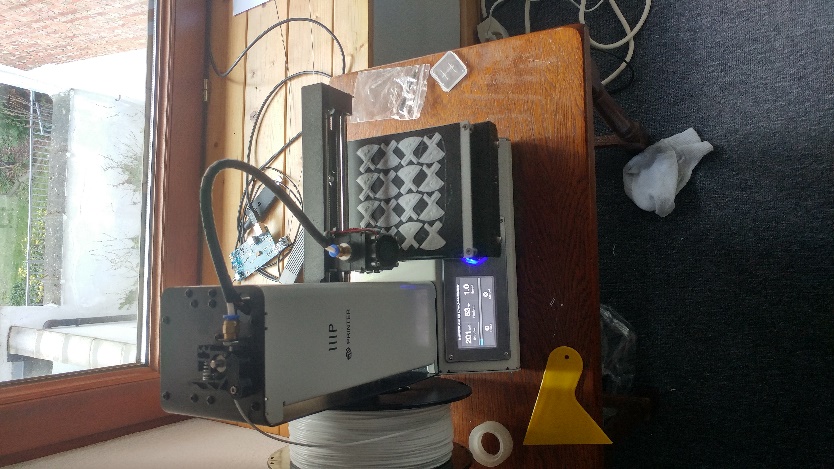
The following sections describe implementations of the above steps.

## Methods III: Implementation of Perfect 3D Printed Pins

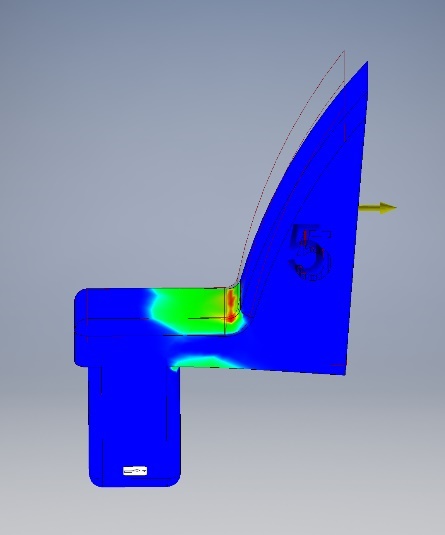
The simplest solution for manufacturing “perfect” pins was found to be 3D printing.

3D printing is a modern manufacturing technique that excels at creating cheap, relatively precise objects in a short period of time. While there are many different kinds of 3D printers available today, the type chosen for this project was a Fused Deposition Modeling (FDM) printer. This type of printer works by melting plastic and laying it down in thin layers to build up an object. With a desktop 3D printer, it is possible to create many plastic pins with identical dimensions!

The recent explosion in popularity of desktop 3D printers means that fully-assembled FDM printers can now be had for $150USD and below, with a plethora of free software available on the internet for supporting the use of these printers. For this project, a Monoprice Select Mini V2 was chosen because of its low price, heated printing bed (important for when printing small parts), high printing resolution, and ease of use.

While other cheaper printers with potentially higher print quality exist and can be ordered from China, the MP Select Mini was chosen because, unlike some of those other printers, it does not have a reputation for spontaneously catching fire[[2]](#footnote-2). It was decided that paying a bit more for something safer was worthwhile, especially considering that the printer was to be located on rented premises.

The prototype design of the 3D printed pins conformed fairly closely to the design of the historical pins. The profile of a #5 pin (see Fig. 2) was imported into Onshape[[3]](#footnote-3) 3D CAD software and extruded to a thickness of 7mm. The resulting part was exported as an STL mesh, sliced and compiled to gcode using Cura[[4]](#footnote-4) slicing software, and printed on the MP Select Mini using PLA plastic on the default draft print settings with 20% infill. The print took about 15 minutes to complete.

This prototype pin was then brought to Lier and installed in the drum as a test. The pin was attached to the drum by press-fit[[5]](#footnote-5) and corresponded to the the bass E on the carillon. After a week, it was found that the pin had broken under the load of playing the low E for so long. ☹

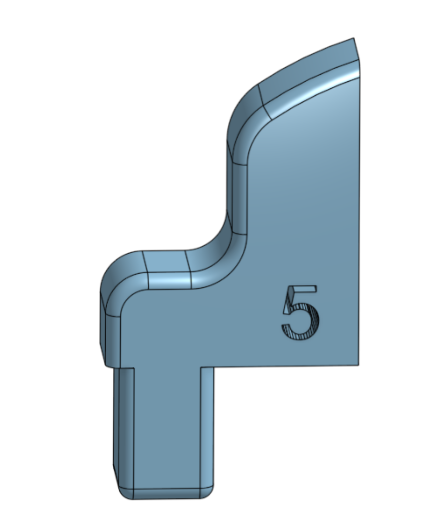
Although discouraging, the failure of the first prototype pin laid the groundwork for a better design. The old design was simulated using Autodesk Inventor’s FEA toolkit to find stress points. Problems were found where the head of the pin meets the body, as can be seen in Figure 3. This simulation matches well with the observed failure point of the broken pin. In addition, it was determined that 20% infill was insufficient and that at least 50% infill was necessary for the pin to withstand forces from the drum.

Figure : FEA analysis of first prototype pin

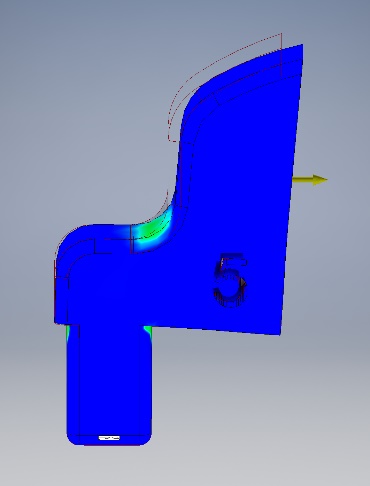
A new design was created which avoided the flaws of the old design. Rounded edges were used to avoid concentrating forces at weak points, and the body and head of the pin was made thicker. Using the same concepts, a #1 pin was also designed. Both pins still used the press-fit attachment method. Although these new pins took longer to print, they proved to be very hardy.

Figure : New pin design

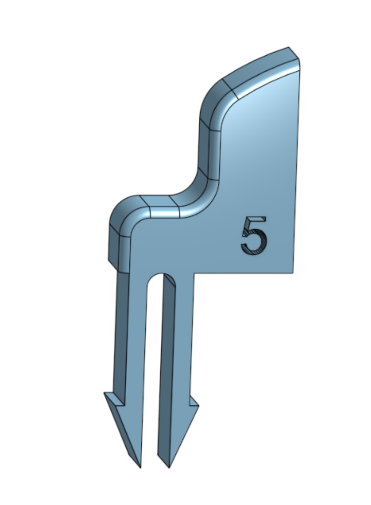
60 #5 pins were printed at 100% infill in batches of 20 each, and 64 #1 pins were printed at 100% infill in batches of 16 each. All prints were done with white Hatchbox PLA extruded at 200°C with a bed temperature of 50°C. The printer took about 5 to 6 hours to print each batch, for a total printing time of about 35 hours distributed over 2 weeks. Although the printing could have been accomplished faster if it was allowed to proceed overnight, this was not done in order to reduce fire safety risks. All printed parts were checked with calipers and post-processed to a final part tolerance of ±0.1mm.

Figure : FEA analysis of new pin design

Towards the end of the project, a third pin design was developed with a clip attachment mechanism. Because the entire autospeelwerk is covered in a thin (and sometimes thick) layer of grease, the press-fit pins were sometimes popping out of their sockets. This new design was somewhat successful; problems mainly arose out of the inconsistent thickness of the drum.

Figure : Clipping pin design

## Methods IV: Implementation of Calibration Tune

The original idea for composing the calibration tune was to provide a computer with the range of the carillon and then to have the computer generate the composition algorithmically. While this does seem to be possible by using Musescore’s Javascript plugin creation capabilities[[6]](#footnote-6), the quickly approaching project deadline made it clear that computer generated composition was a bit outside the scope of this project.

Instead, the calibration tune was composed by hand in Musescore using a checklist to ensure every note was used[[7]](#footnote-7). With #1 and #5 pins available, the composition uses two half notes per drum measure. Every note has the same rhythm, in order to simplify things for the eventual signal processing computations. The composition also begins with a 4 measure clock synchronization pattern (meant to help the computer “lock on” to the tune when listening to the recording) which turned out not to be necessary in the end. There are a total of 102 notes in the calibration tune, which comes out to 51 drum measures. Although on the drum each note is a half note, in Musescore every note was assigned its own measure (again, in order to simplify machine-readability of the score).

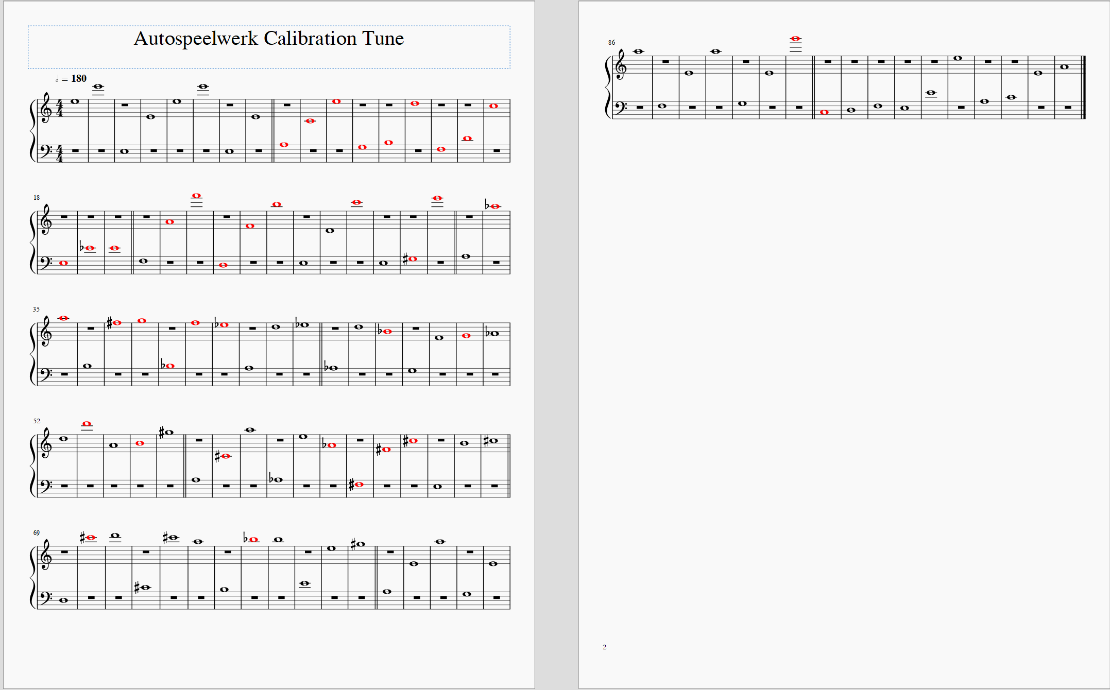
The score was exported as a pdf for human reading, and as a MusicXML file converted to a MusicABC-formatted text file[[8]](#footnote-8) for the computer to read.

Figure : Human-readable form of the tune

## Methods V: Implementation of Software

One of the most challenging parts of this project was designing and implementing the signal processing program for analyzing the recording of the calibration tune. The program needed to be able to do several things:

* Parse a recording of the calibration tune while ignoring noise and false signals (birds, construction noise, etc.)
* Parse the written score
* Set expectations for the recording based on the written score
* Compare expectations vs. the recording
* Output results of the comparison in an easily readable way
* The ability to be implemented despite the programmer’s limited coding ability or experience with signal processing

The code written was mostly successful at achieving these requirements. It was particularly difficult to implement false signal rejection, so instead manual editing of the recording was used to remove sounds that were confusing the program. The code was written in the programming language R because of the author’s familiarity with it, and because it supports many signal processing and data analysis packages. A copy of the code with comments may be found in the Appendix.

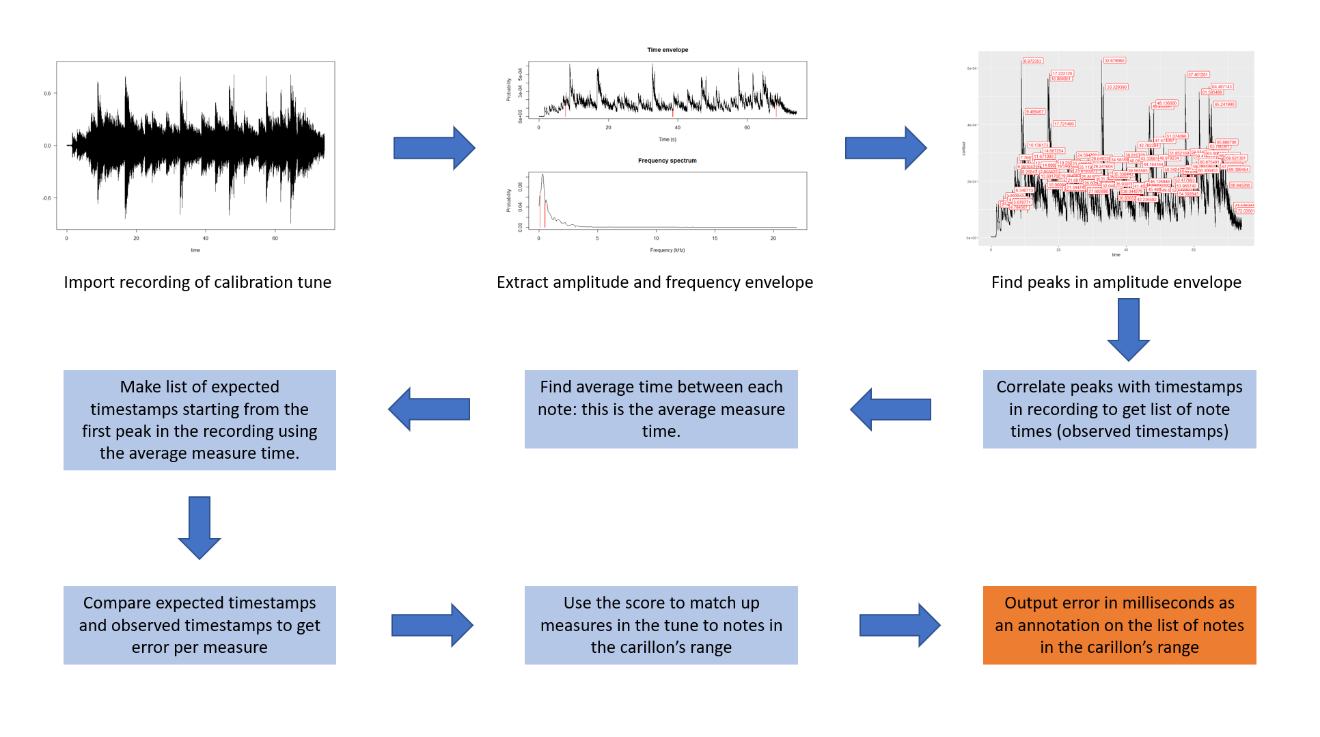
 What follows is a diagram outlining the function of the program.

Figure : Program function diagram

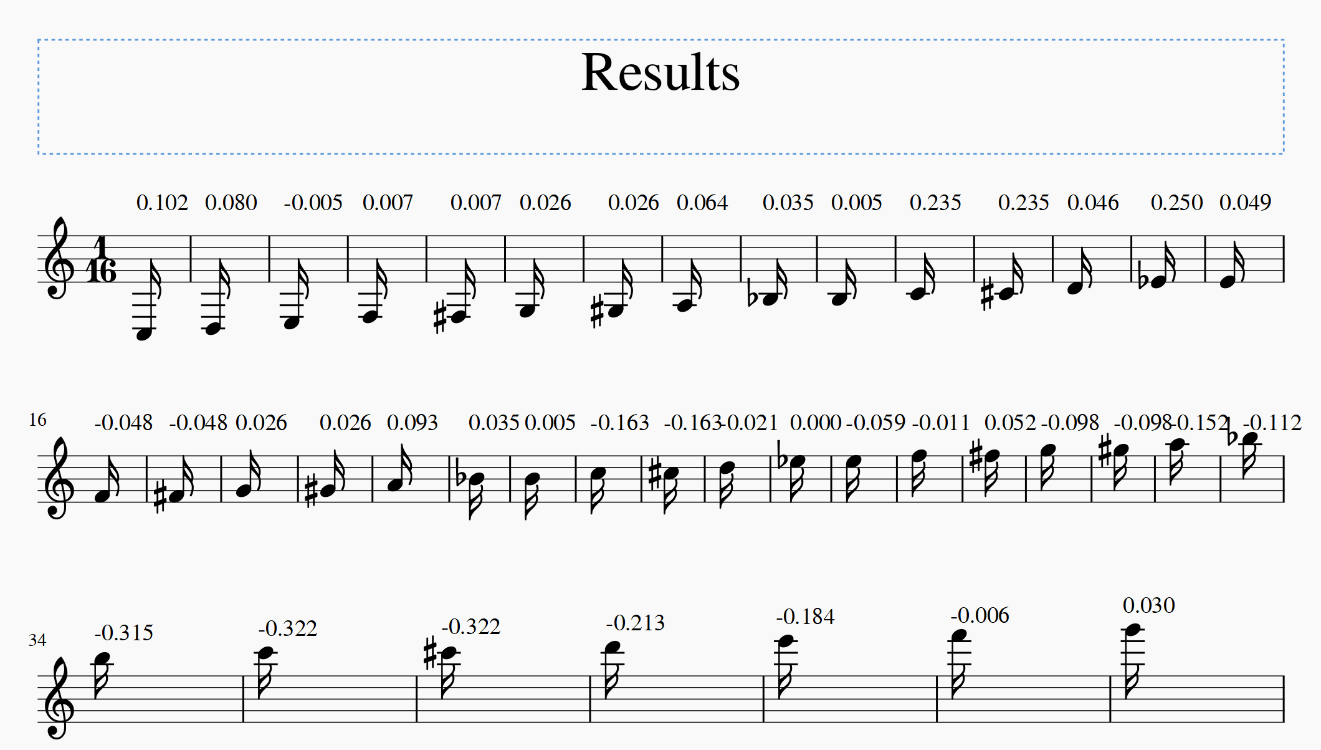


Figure : Example of program output

## Methods VI: Putting It All Together

Execution of the project went mostly as planned. The drum was set with the pins for the calibration tune over the course of a weekend. Several recordings were then made of the tune over the course of a week and a half to fine-tune recording parameters and to ensure that the drum was playing consistently. After manual cleaning of the recording (due mostly to pigeon noises and double-playing hammers), the program was able to successfully extract the calibration tune from the recording and process it. Data produced from the program was then statistically analyzed to test validity of the calibration method and to gain insight concerning the sources of error in the autospeel mechanism.

# Results and Data

Data generated from the calibration program is below in graphical form.

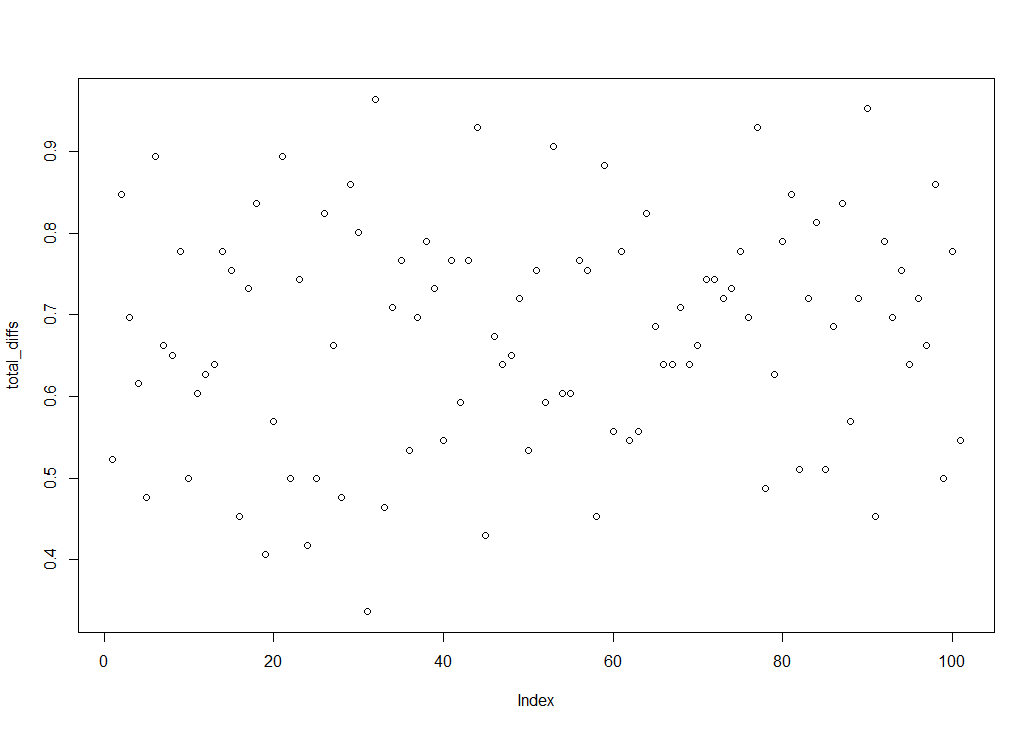


Figure : Time between measures over the course of the calibration tune

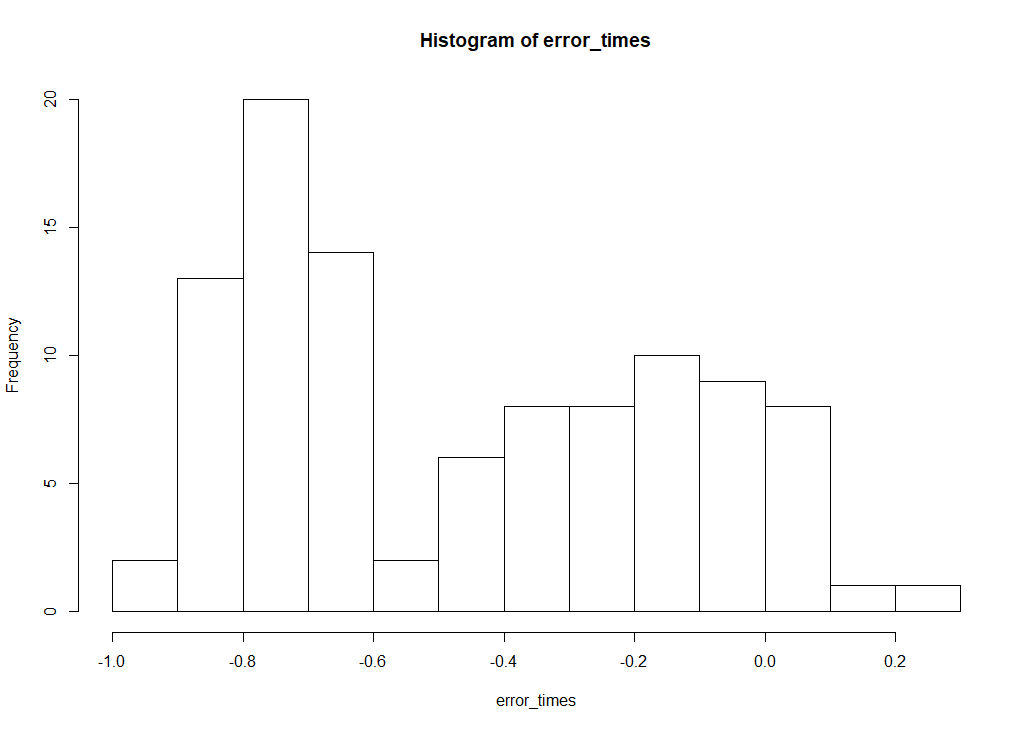


Figure : Histogram showing distribution of rhythmic error

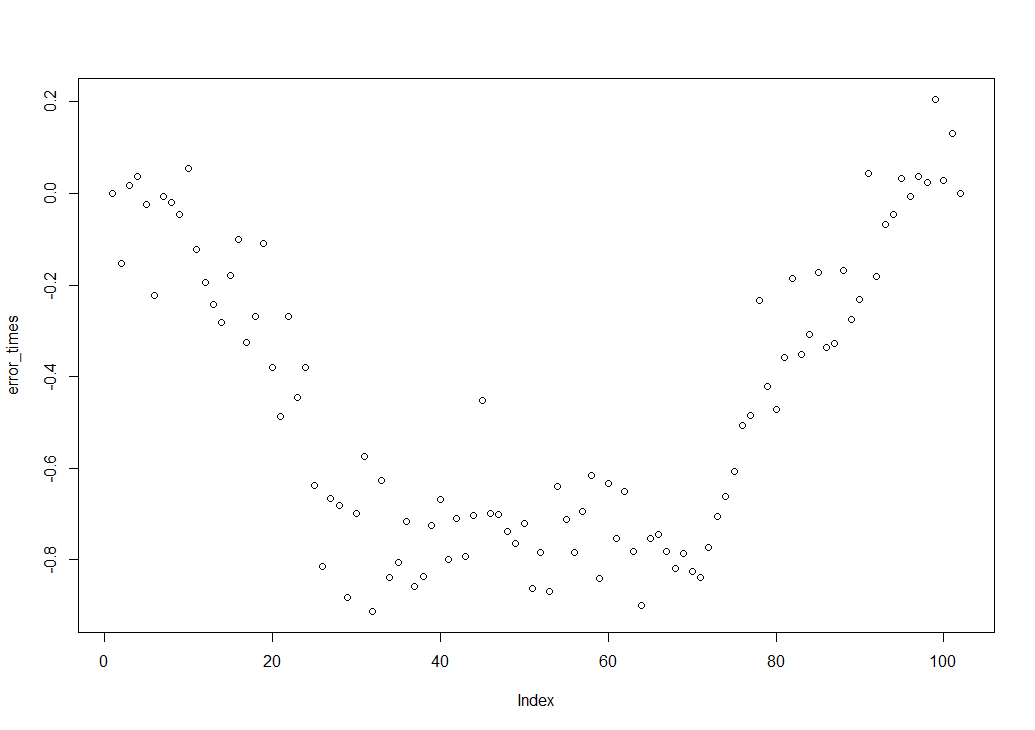


Figure : Error over the course of the calibration tune



Figure : Calibration plot of error over carillon range

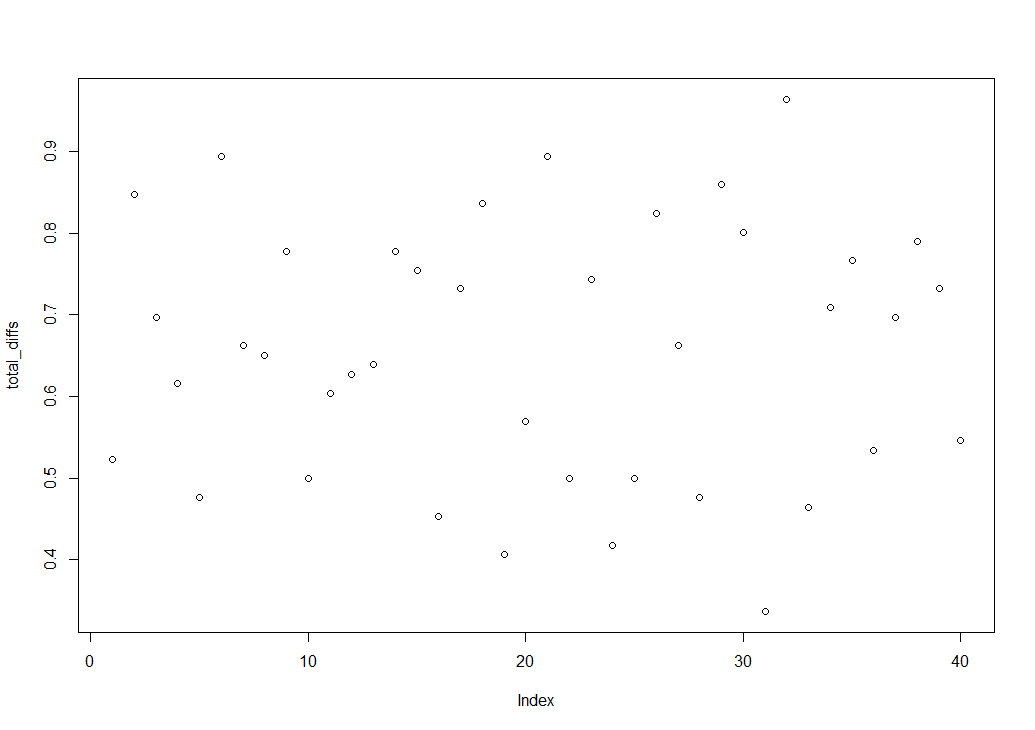


Figure : Time between measures over the course of the first 40 notes of the calibration tune

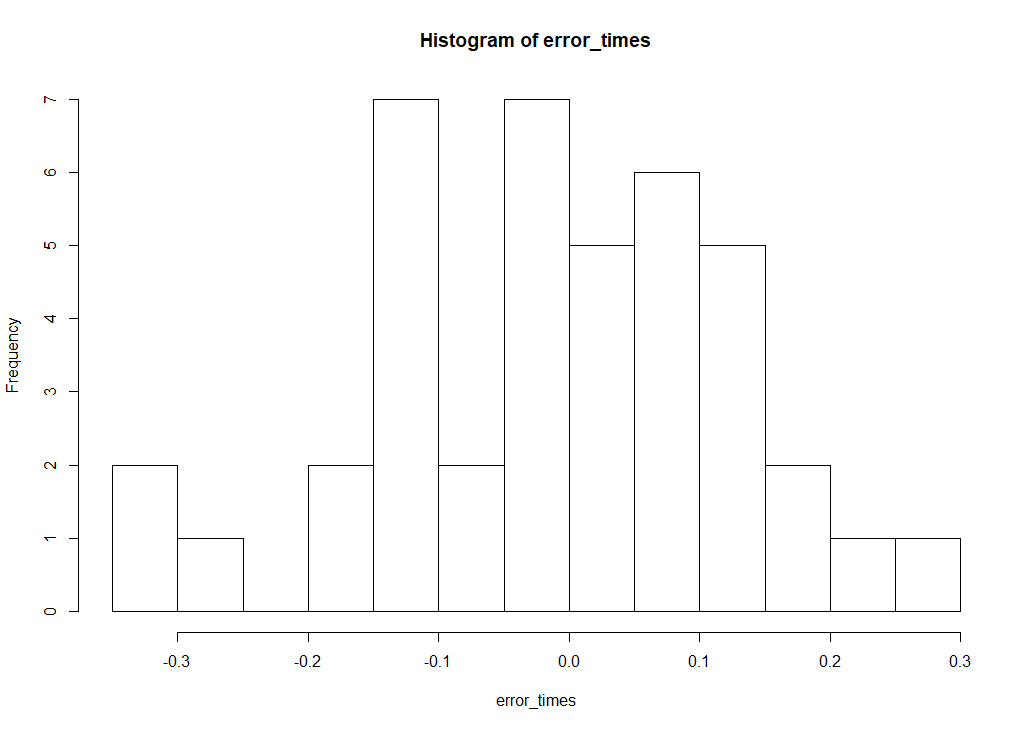


Figure : Histogram showing distribution of error over the first 40 notes of the calibration tune



Figure : Error over the course of the first 40 notes of the calibration tune

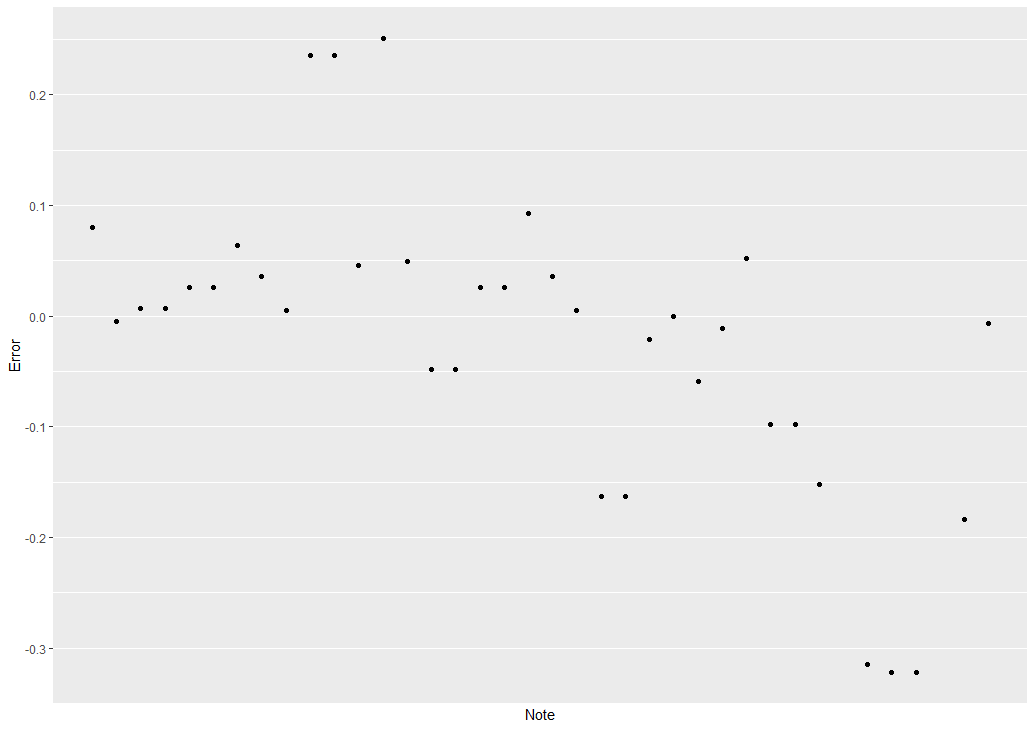


Figure : Calibration plot of error over carillon range, using only data from the first 40 notes of the calibration recording (missing values removed)

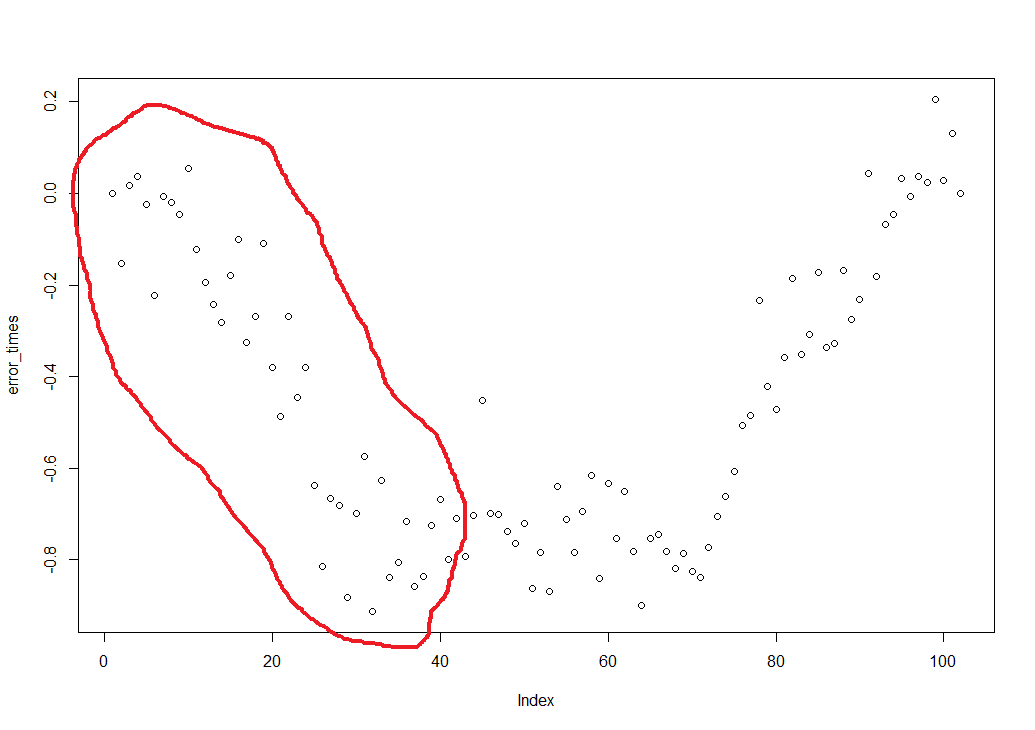


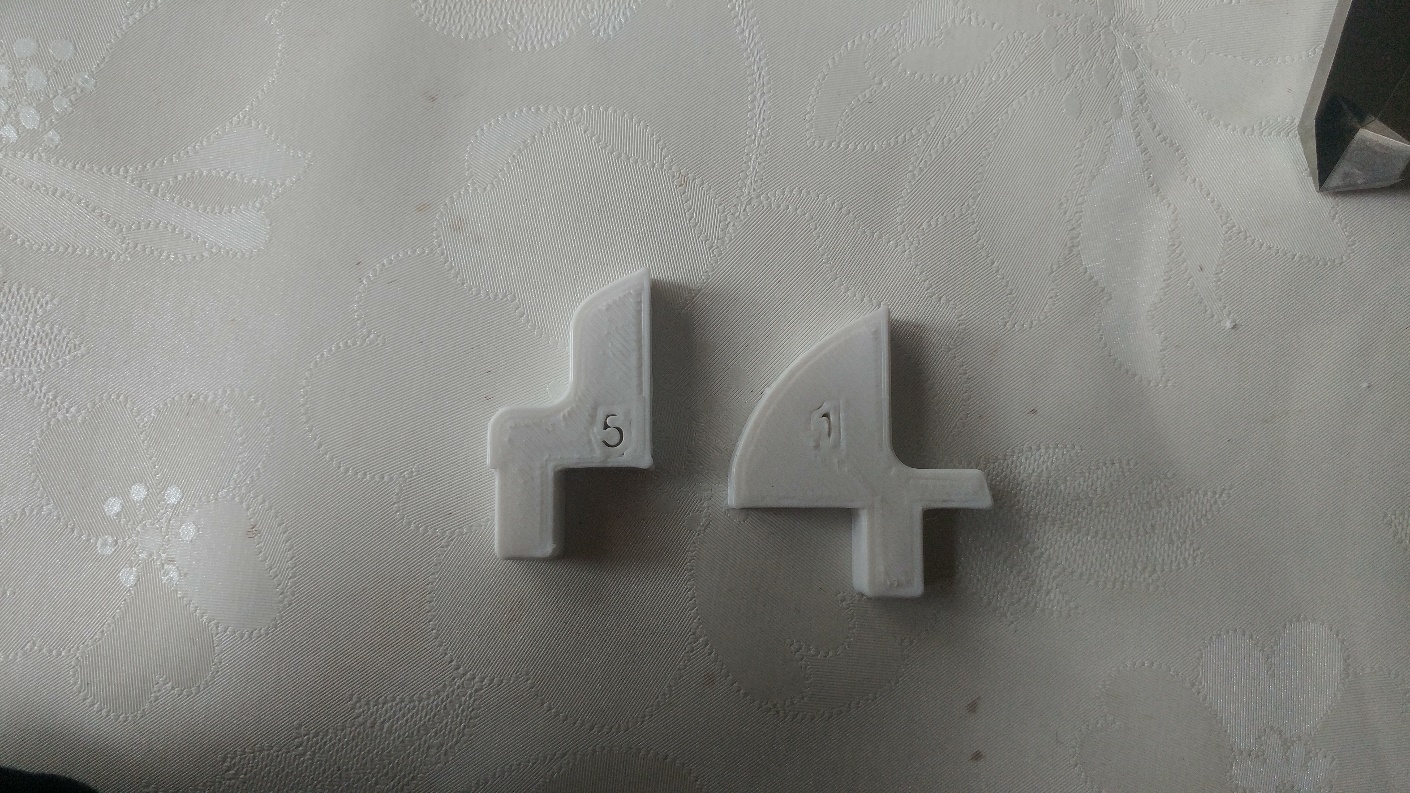
Figure : Figure 12 with approx. the first 40 notes of the calibration tune circled

Raw Data output from the calibration program can be found at the Github link listed in the Appendix.

# Analysis and Discussion

This analysis and discussion of results is divided into two parts. The first part covers the project’s use of 3D printed pins and designs (and potential future improvements), and the second part covers analysis of the data produced by the calibration algorithm (along with potential improvements to the code).

## Part I: Performance of 3D Printed Pins

In total, four different pin designs were created and 3D printed. The first design imitated the design of the historical #5 pins (see Methods: Figure 2). Although functional, it was deemed inadequate for use in the playing drum of a heavy carillon. FEA mechanical load stress analysis indicated that the sharp angles and thin support structures of the design led to failure of the inner edge of the pin between the body and head. In addition, the low infill percentage probably was a contributing factor. It is possible that use of a different infill pattern (or infill printed orthogonal to the direction of expected force) or different printing material might improve performance of this design; however, there is no incentive to pursue research in this direction.

The second design was roughly based off of the dimensions of the historical #5 pin, and was an improvement of the previous design. The root of the design consisted of creating the pin head profile as a 90 degree circular segment (pizza slice) with the back end cut out to provide a surface for hammering the pin into the drum. All edges not expected to be in contact with the print bed were smoothed with 2mm radius fillets, and the transition between the head and body of the pin was reinforced with a 5mm radius fillet to more evenly distribute the force of the drum against the pin without causing a stress point. The third design was a #1 pin created using the same ideas used for the second design. When printed at 100% infill, these second and third designs solved the problems of the first prototype and were used in most of the calibration work.

Figure : 2nd Pin #5 (left) and Pin #1 designs

However, they were not without flaws. Since the pins were printed in PLA (which has a relatively low melting point), the surface in contact with the print bed had a tendency to deform because of the heat of the printing bed. If bed heating was not used, the prints tended to peel away from the bed causing problems with the printing process. The deformed edges, when sufficiently severe, had to be filed or sanded to keep the pins within dimensional tolerances. The solution to this problem would have been filleting all edges of the pins in the design stage, including the edges in contact with the print bed.

An additional design flaw was a failure to take into account the shape of the drum. Since the historical pins have a flat base resting against the drum, the pins designed for 3D printing also were drawn with flat bases. However, the drum is round. Flat surfaces do not sit flush against a round surface. While this mismatch probably didn’t cause any severe problems, it looked ugly and possibly contributed slightly to the problem described below. Again, this problem could be fixed in the future by modifying the design. The base of the pins can be designed as an arch with a radius corresponding to the outer radius of the drum.

The last major flaw with the designs were that they continued to use the “press-fit” method to stay attached to the drum. PLA is slightly flexible, and so for the most part the pins would stay in their sockets through friction once forced in. However, because the sockets in the drum are handmade, some sockets had beveled (ie. angled, not flat) faces which tended to squeeze the pin out of the hole. In addition, the fact that the pin base is a flat surface sitting against the round drum meant a bit of extra leverage which converted the playing force against the pin head to a bit of force directed towards pulling the pin out. An attempt was made to temporarily fix the problem by wrapping the pin posts in paper masking tape, but this was only a superficial solution. An attempt at a solution for this problem was made in the last pin design.

In the last pin design (Methods: Figure 6), the second design was modified so that the pin could be clipped into the drum. This was an attempt to avoid the problems with the press-fit system used in previous designs. While the clipping idea would be perfect in an ideal world, in reality the fact that the drum is handmade caused problems again. It turns out that the wall thickness of the drum is not consistent. In some places, the drum is thicker than in other places. This means that the clipping pins are too long (letting the pins wobble up and down) in some places and too short (preventing the pins from properly clipping) in other places. Some experimentation would be required to solve this problem, but it is possible that designing the clipping end with a flared end[[9]](#footnote-9) could work.

All pin designs after the first prototype held up well to repeated use in the drum. There was a minor amount of pin wear that was reduced by applying a good amount of grease to the mechanism. Future 3D printed pins would benefit from using slightly more expensive but hardier plastic filament materials instead of the discount PLA filament used in this project. Some suggestions include ABS (the plastic used in Lego), or PETG (the plastic used for high pressure soda and sparkling water bottles). However, these materials exhibit higher thermal expansion than PLA, which means some experimentation would be required to obtain accurate printing dimensions.

Some general design principles for creating 3D printed pins can be inferred from the results of this project.

* Consider the effects of distortion caused by print bed heating
* Fillet all edges to prevent concentration of stress forces
* Print at 100% infill
* Print with layers orthogonal to the expected loading direction
* Carefully characterize variation in the drum before designing pins
* Clean pin sockets to get rid of grease so that the pins don’t pop out

Although finding the optimal design for 3D printed pins was outside the scope of this project, some conjectures can be made about the nature of such a design.

Given all of the trouble getting the pins to stay in the drum, it would probably be best to use a screw and nut system just like the historical pins. A threaded system can adapt well to variation in the historical mechanism while simultaneously providing reliable fixturing force. While it is certainly possible to 3D print threaded objects, it is often more practical to add threaded components to printed parts during the printing process or shortly afterwards. Here are some possibilities:

* Inserting threaded rod into a hole designed in the stem of the pin (this method has been used to create new #8 pins for the St. Rombouts drum in Mechelen, Belgium).
* Inserting a nut into a space designed into the pin stem (this common 3D printing technique is called “captured nut” and is considered more reliable than using threaded rod)
* Inserting the head of a bolt into a space designed into the pin stem (“captured screw”, essentially the complement of “captured nut”)
* Using a heated press to apply brass threaded inserts into the pin stem (more reliable than captured nut, but can be difficult to implement)

The author considers the “captured screw” method to hold the most potential for future designs.

Lastly, given the ability to design arbitrary pin shapes and sizes for 3D printing, it would be quite easy for special pins (ie. double or triple pins) or pins in other time signatures to be designed for the autospeel system. It would be quite interesting to see music in 6/8 or even 5/4 pinned for the drum.

## Part II: Analysis of Calibration Data

The calibration strategy was devised under several assumptions:

* Error in the mechanism is random and is distributed normally
* Suboptimal cable adjustment is the most significant source of error
* Non-uniform pin dimensions is the second most significant source of error
* Other sources of error in the mechanism are negligible compared to poor adjustment and pins

The calibration recording analysis program functions as expected, which is great. Data output by the program reveals several interesting things that challenge the assumptions under which the calibration strategy is devised. If the initial assumptions were true, we would have expected to see:

1. Variation in time between measures is random (ie. no apparent patterns)
2. Variation in rhythmic error is random (ie. no apparent patterns)
3. Distribution of error times and time between measures is normally distributed (that is, a histogram of the errors would look like a standard normal distribution: “bell curve”)

Given point 1 above, we expected that Figure 10 would show points randomly distributed around a mean, and Figure 10 appears to match with expectations. The first surprise comes in Figure 11; the histogram shows that the rhythmic error is definitely not distributed in a bell curve. We see a sort of bimodal distribution, and thus our expectation 3 listed above is not met. Clues as to why this is can be found in Figure 12. We expected Figure 12 to look very similar to Figure 10; however these two charts look nothing alike. The data in Figure 12 show a clear pattern of increasing error until a plateau about 40% through the calibration tune, followed by a pattern of decreasing error all the way to the end of the calibration tune.

Given that we know the calibration program calculates error using the average drum measure time, Figure 12 shows us that notes in the recording are at first playing increasingly faster than the average drum measure time, followed by a period of consistent speed, and lastly a period of notes playing increasingly slower. We can infer from this information that the drum does not turn at a consistent speed: it speeds up at the beginning of the tune, stays consistent for a while, then slows down at the end of the tune. This explains our bimodal error distribution: the first hump is the collection of too-fast notes when the drum is speeding up, and the second hump is the too-slow notes when the drum is slowing down. Some possible reasons as to why the drum does not rotate consistently:

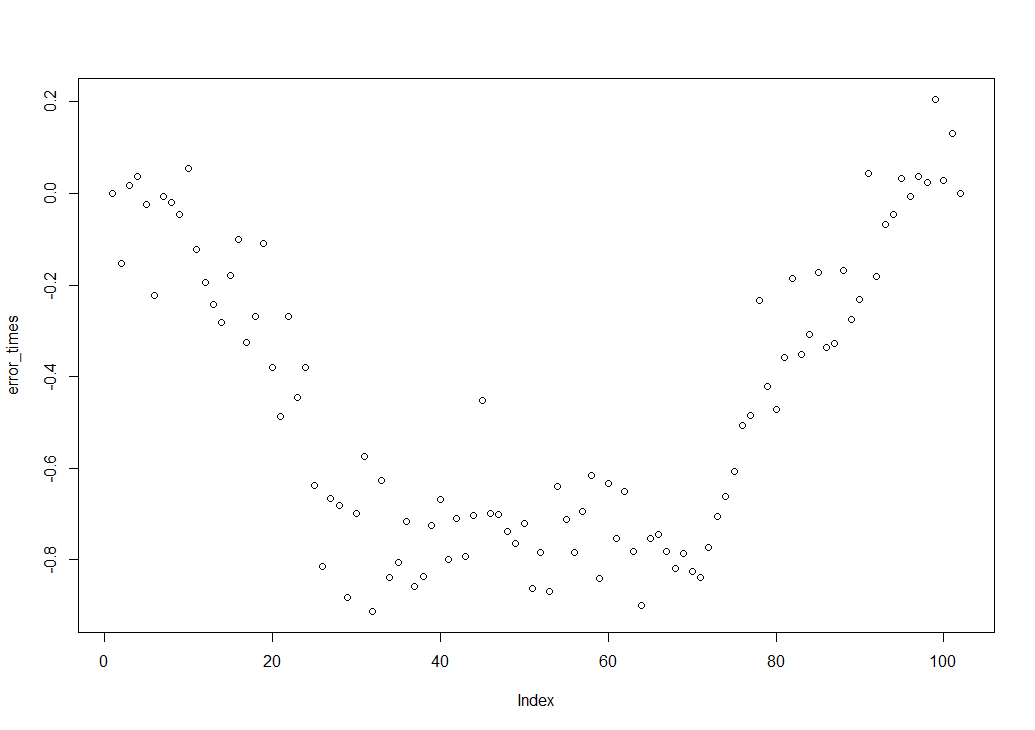
* The drum may not be perfectly round
* The drum may be heavier on one side than the other (due to nonuniform thickness)
* The weight drive geartrain may not be uniformly lubricated
* The weight drive geartrain may be worn or damaged asymmetrically
* The drum’s speed governor may be set outside of tolerable limits

It is unknown which of these factors is the cause of the drum’s inconsistent speed; it is possible that all factors may play a role.

Unrelatedly, we see in Figure 13 (which is just figure 12 ordered by note from lowest to highest) that there appears to be a decreasing trend in the rhythmic errors of the notes of the carillon. Although this trend wasn’t verified as statistically significant (especially given the confounding variable of drum speed discussed above), we can surmise that the lower notes of the carillon may be more error prone than the higher notes. This might have something to do with the decreasing weight of the hammers over the range of the carillon.

The appearance of the drum speed inconsistency was an unwelcome surprise which violated all assumptions made when formulating the calibration method: error in the mechanism is not completely random and suboptimal cable adjustment is not the most significant source of error in the autospeel system. However, it is thankfully possible to use statistical techniques to work around this confounding variable.

Because the drum speed anomaly occurs over the course of the entire calibration tune instead of on a note by note basis, it is possible to use a set of linear formulae (regression lines) to model the drum’s slowing down and speeding up. The difference between each note’s measured error and the predicted error for that note based only on the drum speed model (the “residual” of the error) yields the error we care about: error due to poor adjustment of the cables. This concept is shown more clearly in visual form below.



Line 3

Line 2

Line 1

Figure : Illustration of multiple regression lines used to model drum speed error

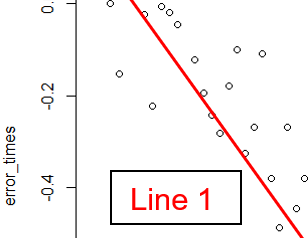


Figure : Blue lines are examples of residuals that show the true rhythmic error due to adjustment

As an illustration, Figure 20 shows three hypothetical linear regression lines that together cover the extent of the calibration song. These lines are equations in the form y = mx + b, where m is the slope and b is the y-intercept. Figure 21 is an inset of Figure 20, and shows a representation of the residuals of the total errors. These “residuals” are a close approximation to errors from non-optimum cable adjustment.

While this method works on a logical and mathematical level, the difficulty lies in automating the process; that is, integrating the process of calculating the regression lines and extracting the residuals into the calibration program. The calibration program is written in the programming language R which does have excellent support for statistical analysis (including the ability to calculate regression lines given a set of data). Unfortunately, the author of this thesis does not have sufficient coding ability to figure out how to get the program to understand how to split up the data into multiple regression analyses. As of now, a human must tell the computer how to subdivide the data. There is perhaps potential in using iterative Taylor Series approximations for nonlinear regression analysis (which would allow the computer to approximate the drum speed with a single equation), but implementation of this method is clearly far beyond the scope of this project.

Alternatively, Figures 14 through 18 show the output of a much more approximate but simpler method. The program is told to only look at a relatively linear segment of data at one time (in this case, calibration tune notes 1 to 40, see circled points in Figure 18). The average drum measure time is again used as a standard, and the program calculates note errors using the same method as the original algorithm. This is essentially equivalent to using a linear regression line with a slope of 0. The results are acceptable. Figures 14 and 16 look similar as expected, and the histogram in Figure 15 shows a distribution that looks far more like the expected normal distribution bell curve. Error values output by the program are within reason and are useful for making informed adjustments of the autospeelwerk cable lengths.

Below is a list of possible future improvements to the program:

* Multivariate analysis
* Automated segmented linear regression
* Nonlinear regression analysis
* Pitch-based note recognition using Fast Fourier Transform (instead of only detecting amplitude envelope peaks)
* Automated noise removal/rejection
* Detection of double notes (from hammer springs being too loose)
* Conversion of timing error to cable length error using hammer weight and hammer resting angle table (ie. “A#1 requires adjustment by +2.53 mm”)
* Integration of cable adjustment screw pitch table so that the program can give adjustment instructions in terms of screw rotations (ie. “A#1 requires adjustment by 1.3 clockwise rotations of the adjustment screw”)

# 

# Appendix

## Pictures

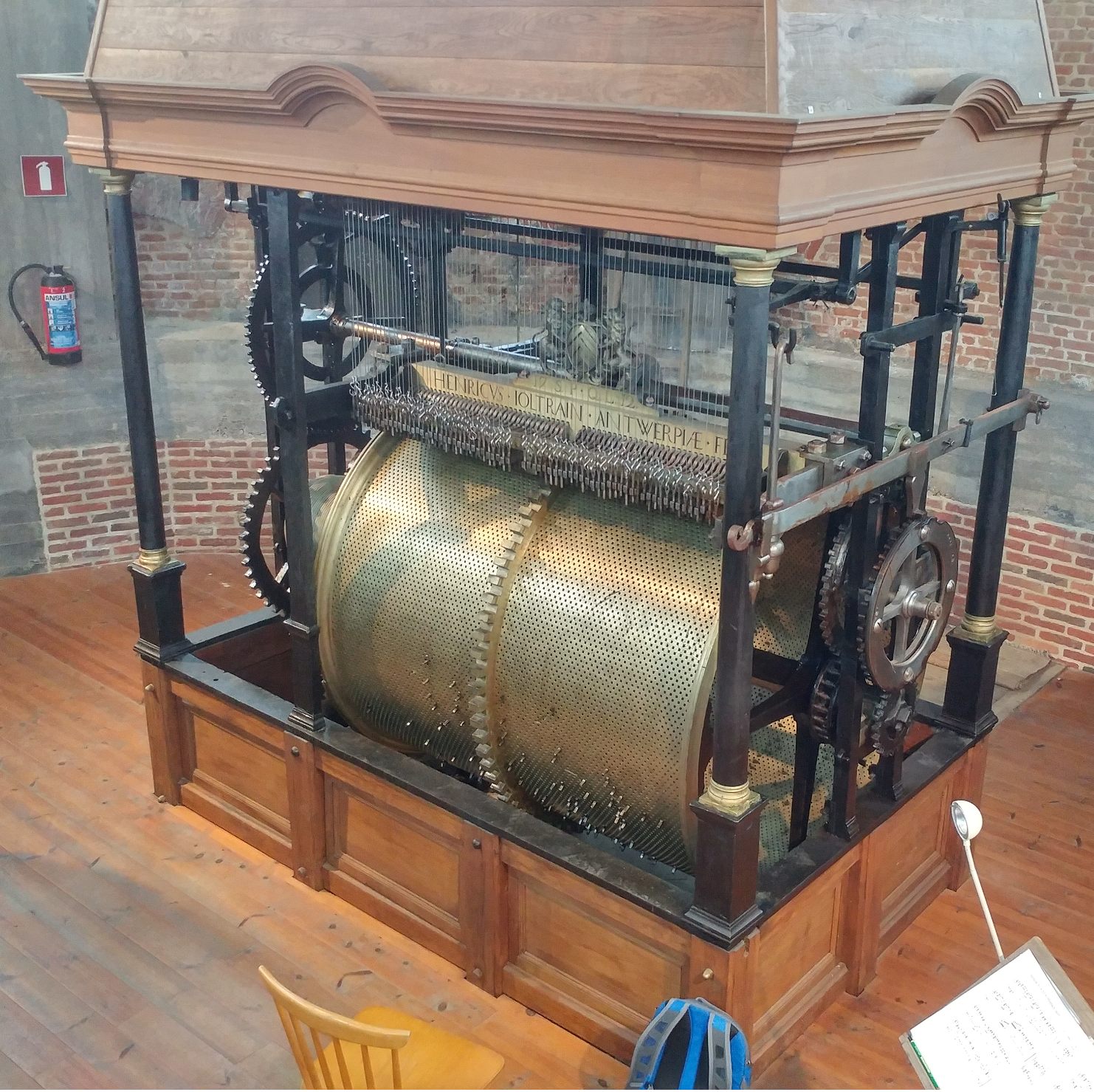


Figure : St. Gummarus Autospeelwerk

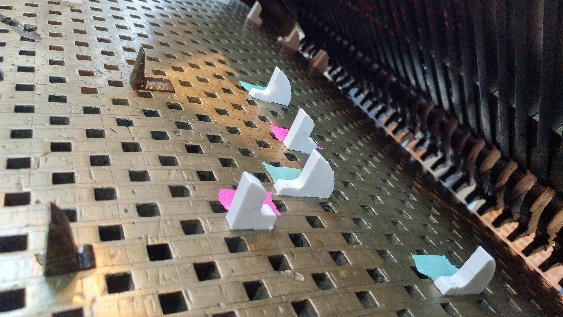


Figure : The pinning process



Figure : Cable adjustment screws (left) and view of a clipping pin from inside the drum (right)

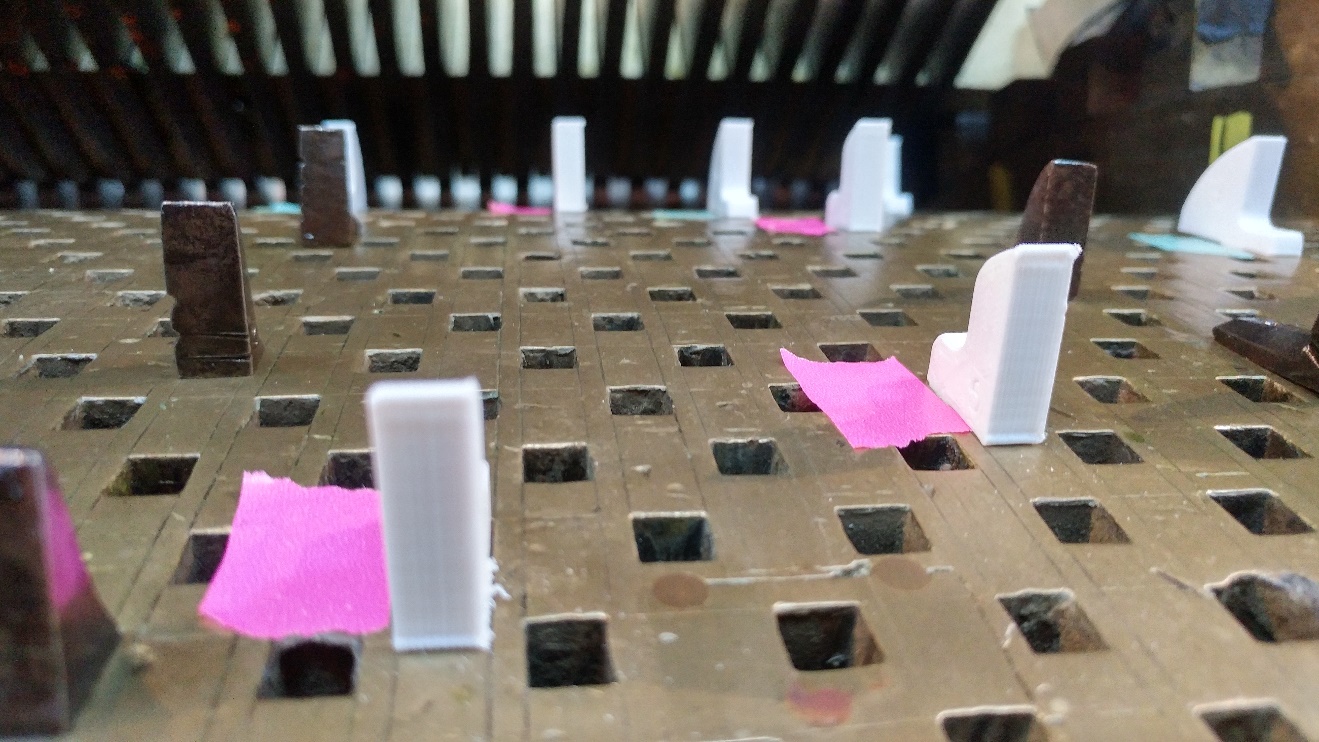


Figure : Close-up view of pins

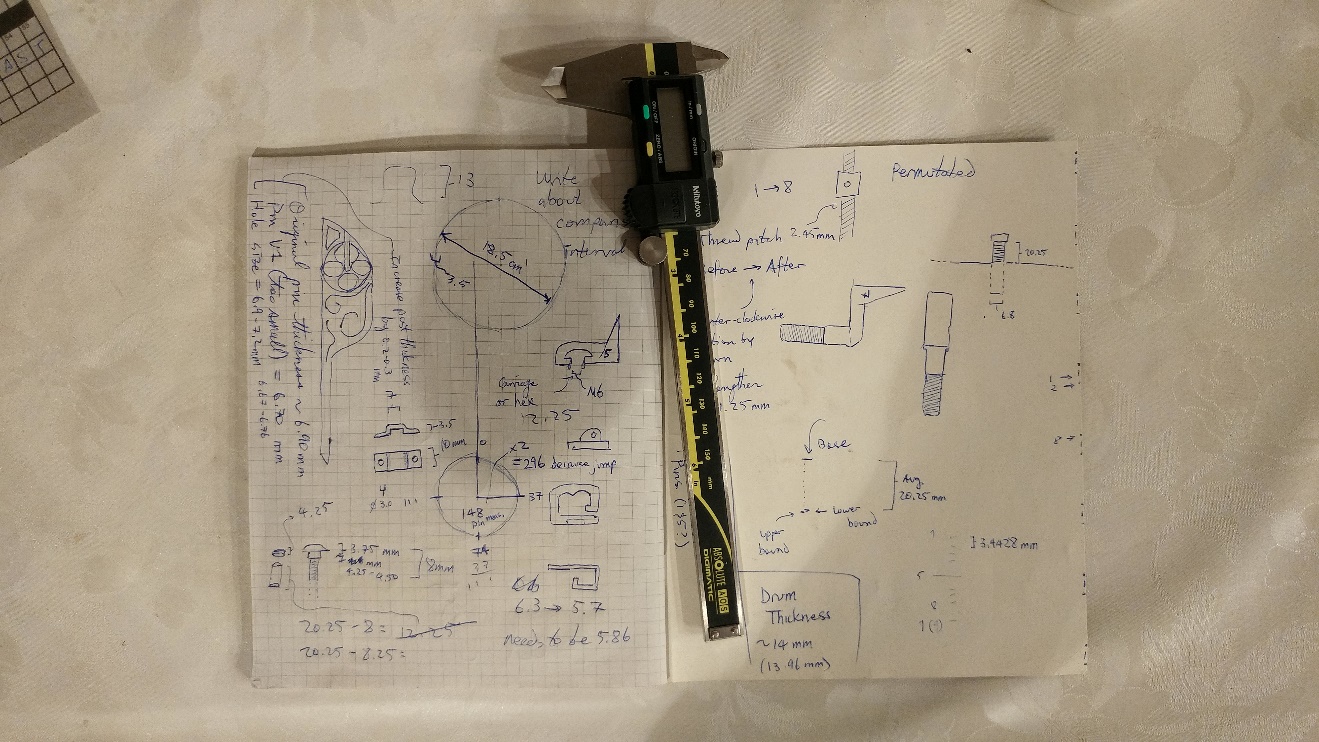


Figure : Sketchbook

## Project Github Repository

<https://github.com/KeiranCantilina/Carillon_Autospeelwerk_Calibration>

Most project files and programs can be found in this repository.

## Autospeelwerk Calibration Analysis Program

This program can also be downloaded at <https://github.com/KeiranCantilina/Carillon_Autospeelwerk_Calibration/blob/master/calibration_algorithm.R>

It requires a working installation of R to run. The author recommends the use of a development environment such as RStudio.

## Carillon Autospeelwerk Calibration Error Quantifier

## Note: For the process to work well, please make sure the recording does not contain loud noises other than bells, and that there

## is several seconds of silence before and after the calibration tune.

library**(**signal**)**

library**(**fftw**)**

library**(**seewave**)**

library**(**tuneR**)**

library**(**audio**)**

library**(**reshape2**)**

library**(**ggplot2**)**

library**(**ggpmisc**)**

library**(**splus2R**)**

library**(**gginnards**)**

recording\_path **<-** "C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\Calibration recording 4.wav"

calibration\_tune\_path **<-** "C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\Calibration Tune.abc"

autospeelwerk\_range\_path **<-** "C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\Checklist.abc"

output\_path **<-** "C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\abc2xml\_218\\results.abc"

debug\_output **<-** "C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\abc2xml\_218\\debug.csv"

## Number of notes in the clock sync header of the calibration tune

number\_clocks **<-** 8

## Number of notes in the total calibration tune

number\_notes **<-** 102

## Number of notes available in the range of the autospeelwerk (not including repeated notes)

number\_autospeelwerk\_range **<-** 40

## Thread pitch of adjustment screws (in mm/rotation). Put NA if output in milliseconds delay is desired

thread\_pitch **<-** **NA**

## Adjustment length delay constant (in seconds delay per mm of adjustment). Default is NA. This is a crude linearization.

adjust\_const **<-** **NA**

## Import recording of calibration tune

recording **<-** readWave**(**recording\_path**)**

## Import calibration tune ABC file

calibration\_tune\_file **<-** read.delim**(**calibration\_tune\_path**)**

calibration\_tune\_raw **<-** as.character**(**calibration\_tune\_file**[**9,**])**

calibration\_tune\_raw **<-** gsub**(**" | ",'\n', calibration\_tune\_raw, fixed**=TRUE)**

calibration\_tune **<-** read.table**(**text**=**calibration\_tune\_raw,fill **=** **TRUE**, header **=** **FALSE**, stringsAsFactors **=** **FALSE)**

## Import autospeelwerk note range ABC file

autospeelwerk\_range\_file **<-** read.delim**(**autospeelwerk\_range\_path, stringsAsFactors **=** **FALSE)**

autospeelwerk\_range\_raw **<-** as.character**(**autospeelwerk\_range\_file**[**8,**])**

autospeelwerk\_range\_raw **<-** gsub**(**" | ",'\n', autospeelwerk\_range\_raw, fixed**=TRUE)**

autospeelwerk\_range **<-** read.table**(**text**=**autospeelwerk\_range\_raw,fill **=** **TRUE**, header **=** **FALSE**, stringsAsFactors **=** **FALSE)**

## Extract filtetred amplitude envelope (contour)

contour **<-** acoustat**(**recording**)**

## Visualize contour

contour\_graph **<-** contour**$**time.contour

## Reformat contour as timestamp/amplitude dataframe

contour\_dataframe **<-** as.data.frame**(**contour\_graph**)**

## Plot contour and compute peaks

peaks\_ggplot **<-** ggplot**(**contour\_dataframe, aes**(**time,contour**))+** geom\_line**()** **+** stat\_peaks**(**label.fmt **=** "%0.6f", ignore\_threshold**=** 0.15, colour **=** "red", span **=** 50, strict**=TRUE**, geom**=**"label", hjust **=** **-**0.1**)**

peaks\_ggplot

## extract peaks from ggplot object

peaks\_ggplot\_data **<-** ggplot\_build**(**peaks\_ggplot**)**

peak\_times **<-** as.numeric**(**peaks\_ggplot\_data**[[**"data"**]][[**2**]][[**"label"**]])**

## Sort timestamps from first to last, keeping only the correct number of peaks (just in case)

peak\_times **<-** sort**(**peak\_times**)**

peak\_times **<-** peak\_times**[**1**:**number\_notes**]**

## Extract clock sync from data (should be first 8 notes)

clock\_times **<-** peak\_times**[**1**:**number\_clocks**]**

## Extract reference drum measure duration from clock sync

clock\_diffs **<-** diff**(**clock\_times**)**

average\_measure\_time **<-** mean**(**clock\_diffs**)**

## Just for fun, find average measure over entire recording (might be more accurate)

total\_diffs **<-** diff**(**peak\_times**)**

average\_total\_diffs **<-** mean**(**total\_diffs**)**

## Build reference vector (ref\_times) of expected peak timestamps based on the drum measure times, using first clock peak as starting time

origin\_time **<-** peak\_times**[**1**]**

ref\_times **<-** c**(**origin\_time**)**

**for** **(**i **in** 1**:(**number\_notes**-**1**)){**

ref\_times **<-** append**(**ref\_times, origin\_time**+(**average\_total\_diffs**\***i**))** ## Try subbing average total vs. average clock diffs

**}**

## Compare ref (ref\_times) to dataset (peak\_times) and ouptut diff vector (observed-expected).

## Positive errors are notes playing too late; negative errors are notes playing too early.

error\_times **<-** peak\_times**-**ref\_times

plot**(**error\_times**)**

hist**(**error\_times, breaks**=**10**)**

## Pair error with notes in cal tune

calibration\_results **<-** as.data.frame**(**calibration\_tune**[**1**:**number\_notes,**])**

colnames**(**calibration\_results**)** **<-** "Notes"

calibration\_results**$**Error **<-** error\_times

## Compute error per note

## use lookup table and grep to pair errors (averaged) with notes

collated\_results **<-** as.data.frame**(**autospeelwerk\_range**[**1**:**number\_autospeelwerk\_range,**])**

colnames**(**collated\_results**)** **<-** "Notes"

collated\_results**$**Error **<-** **NA**

**for** **(**j **in** 1**:**number\_autospeelwerk\_range**){**

grep\_indices **<-** grep**(**autospeelwerk\_range**[**j,1**]**,calibration\_results**$**Notes**)**

grep\_mean **<-** mean**(**calibration\_results**$**Error**[**grep\_indices**[**1**:**length**(**grep\_indices**)]])**

collated\_results**$**Error**[**j**]** **<-** grep\_mean

**}**

## Output as ABC file with annotations over notes (^"text")

results\_ABC\_file **<-** data.frame**(**autospeelwerk\_range\_file**$**X.1, stringsAsFactors **=** **FALSE)**

temp\_paste\_vector\_1 **<-** c**()**

**for** **(**k **in** 1**:**length**(**collated\_results**$**Notes**)){**

temp\_paste\_vector\_1**[**k**]** **<-** paste**(**'@"', sprintf**(**"%0.3f",collated\_results**$**Error**[**k**])**,'"', collated\_results**$**Notes**[**k**]**,sep **=** ""**)**

**}**

temp\_paste\_vector\_2 **<-** paste**(**temp\_paste\_vector\_1, collapse**=**" | "**)**

results\_ABC\_file**[**8,**]** **<-**temp\_paste\_vector\_2

results\_ABC\_file**[**1,**]** **<-** "T:Results"

write.table**(**results\_ABC\_file, output\_path, quote**=FALSE**, row.names **=** **FALSE**, col.names **=** **FALSE**, sep**=**"\n"**)**

## Convert ABC File to musicXML file, leave in results folder

## This section is done by calling a Windows batch program. This will need to be changed for other platforms. Set the “command” variable to the path of the abc to xml conversion program.

command **<-** "\"C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\abc2xml\_218\\abc2xml.exe\" \"C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\abc2xml\_218\\results.abc\" -o \"C:\\Users\\Keiran\\Desktop\\Carillon Thesis Composition\\Carillon autospeelwerk project\\Results\" -z replace"

system**(**command**)**

plot**(**total\_diffs**)**

## Debug/Raw Data output

write.csv**(**peak\_times, debug\_output**)**

####################

## Calibration Tune Musescore file

<https://github.com/KeiranCantilina/Carillon_Autospeelwerk_Calibration/blob/master/Calibration%20Tune.mscz>

## Calibration Tune ABC text file

X:1

T:Calibration Tune

L:1/1

Q:1/2=180

M:4/4

I:linebreak $

K:C

V:1 treble nm="Piano" snm="Pno."

V:1

e | e' | E, | E | e | e' | E, | E | A, | C | e | G, | B, | d | F, | D | c | E, | \_E | E | F, | A | f' | D, | F | b | E, | D | c' | E, | ^G, | e' | A, | \_a | a | B, | ^f | g | \_B, | f | \_e | A, | d | \_e | \_A, | d | \_B | G, | F | G | \_A | d | d' | A | B | ^g | A, | ^C | a | \_A, | e | \_A | ^F, | ^F | ^c | E, | B | ^c | D, | ^c' | d' | ^C | ^c' | a | B, | \_b | b | E | e | ^g | A, | E | a | G, | E | a | F, | E | a | G, | E | g' | C, | D, | F, | E, | E | e | A, | C | E | A | %102

## Calibration Recording

Please visit <https://soundcloud.com/keiran-cantilina/calibration-recording-4> to access the recording.

## 3D Printed Pins CAD drawings

Historical #5 pin 2D CAD drawing: <https://cad.onshape.com/documents/2741f5c0ccd39309527bc4d3/w/e6a419d372b98bde99d54b69/e/997e4240abbd3541f24412fb>

Prototype #5 pin 3D CAD drawing: <https://cad.onshape.com/documents/03de121fbbb6eeb9e1cafd16/w/b137daeaf8fc937561b92eff/e/68efe126dd9e349dbeb74ea9>

Press-fit #5 pin 3D CAD drawing: <https://cad.onshape.com/documents/3f4e96adedb08f0047702275/w/754525a131bf4347794c6ea1/e/41f3891dda9f1be648de8abb> (see version history)

Press-fit #1 pin 3D CAD drawing: <https://cad.onshape.com/documents/6c8170c9167b03adadc3b7c6/w/85af93b2c181e65f32399e0f/e/606ce7e02229b5d4fc1aee87>

Clip-fit #5 pin 3D CAD drawing: <https://cad.onshape.com/documents/3f4e96adedb08f0047702275/w/754525a131bf4347794c6ea1/e/41f3891dda9f1be648de8abb>

NOTE: STL files for 3D printing can be directly exported from the Onshape drawing documents.

## Drum measurement CAD drawings

<https://cad.onshape.com/documents/f348bc71120be64cca27a18e/w/6a0672c22cc1f92aad7e6b5a/e/e7212d5f32697354f854a6cf>

Hardcopy or offline copies of the data in this Appendix can be requested from the author at keirancantilina@gmail.com.

1. Information from <https://antiquitymusic.com/schulmerich-americana-keyboard-carillon> [↑](#footnote-ref-1)
2. Do a Google search for “Anet A8 fire.” The results are scary. [↑](#footnote-ref-2)
3. <https://www.onshape.com/> [↑](#footnote-ref-3)
4. <https://ultimaker.com/en/products/ultimaker-cura-software> [↑](#footnote-ref-4)
5. The pin post was slightly larger than the socket on the drum, and it was hammered until it fit [↑](#footnote-ref-5)
6. <https://musescore.org/en/plugin-development/introduction> [↑](#footnote-ref-6)
7. Composition can also be seen in the Appendix [↑](#footnote-ref-7)
8. See MusicABC file in the Appendix [↑](#footnote-ref-8)
9. Similar to a ["molly bolt"](https://en.wikipedia.org/wiki/Molly_(fastener)) [↑](#footnote-ref-9)