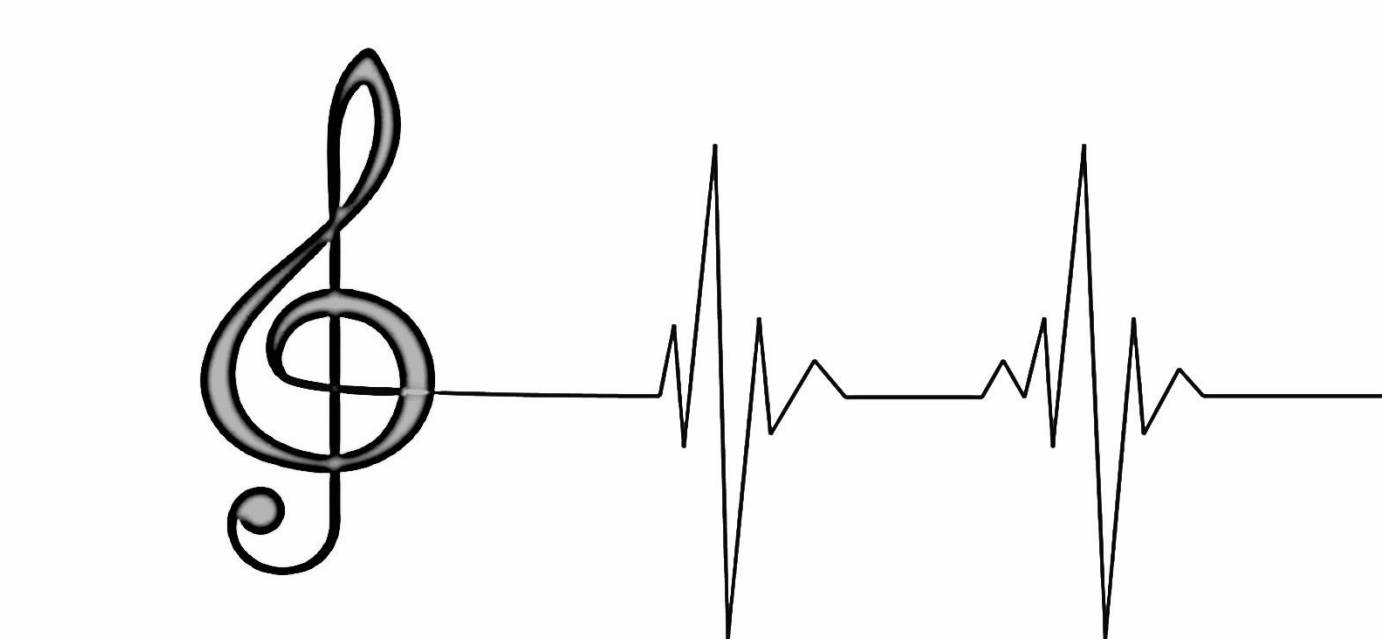
Visualizing the physiological synchronization of musicians



15/02/2019

Jochem Postmes
s1594761
BSc Creative Technology
University of Twente

Supervisors:
Dr. Angelika Mader
Dr. Ir. Erik Faber

Abstract

The aim of this research project is to investigate the physiological synchronization between musicians that make music together, for the following purpose: if this synchronization is present and can be feedbacked, it can improve the musicians' experience of playing together. This leads to a research question: how to design a system that enhances the physiological synchronization of musicians making music together through the use of a visualization? Based on a literature study and ideation, the main design idea is to measure the heart rate variability (HRV) of musicians making music together, and use the synchronization in this value between them to make a visualization. This concept is tested by experimenting with different prototypes for a feedback visualization system. The results of these experiments show that the heart rate variability of musicians is indeed a variable which can be used to feedback the amount of physiological synchronization. However, the visualization that is used does not have a clear impact on the musicians' experience of playing. Based on the results, this research project gives recommendations of how to measure the HRV and feedback it in an unobtrusive way. It provides other recommendations for designing a system which visualizes the physiological synchronization of musicians as well.

Acknowledgements

First off, the author would like to thank their supervisors for the large amount of support and availability throughout the long process that this graduation project was, as well as for providing the initial concept of investigating the topic. Furthermore, the author would like to thank the singers from Vocal Group Musilon that helped out, for their patience and active participation in my series of experiments. Also a word of thanks to the experts that were interviewed, who showed a lot of interest in the project. Lastly, a word of thanks to the author's friends and family for the support and motivation boosts that they provided when needed.

Index

Introduction

Whether someone is playing in a band, orchestra, ensemble or taking part in another musical setup: making music together is one of many ways of synchronizing the actions of multiple individuals. This synchronization is usually reached through directly visible or audible signals; for example a conductor's visual instructions (Luck, 2008), percussion- or bass-induced rhythm and the simultaneous breathing of singers or wind instruments. However, there are more factors that play a role in the process of synchronizing which could be used to enhance musicians' experience further. Heart rate variability, for example, is known to be an indicator of emotional measure and interpersonal connections (McCraty, 2017; Müller & Lindenberger, 2011). Moreover, there are other physiological signals that have an influence on these variables as well, such as breathing rate and skin conductivity (Bradley & Lang, 2000; Ouwerkerk, 2010).

An opportunity arises when the synchronization can be enhanced even further. If a matching of feelings and/or physiological signals exists during play, feeding this back to the musicians might improve their experience of playing together, and possibly increase the level of synchronization further. Subsequently, visualizing this to their audience will create an extra layer to the experience of watching a musical performance. Thus, the main goal is to create an enhanced musical experience for both the musicians and their audience.

The goal of this research is to create a system which visualizes the amount of physiological synchronizations between musicians playing together to strengthen their feeling of unity and improve their experience. The main research question can be formulated as follows:

> *How to design a system that enhances the physiological synchronization of musicians making music together through the use of a visualization?* [RQ]

To find an answer to this question, the following questions must be answered in turn:

> *How do musicians synchronize physiologically while making music together?* [Q1]
> *What defines physiological synchronization between human beings?* [Q2]
> *How can this synchronization be measured and quantified?* [Q3]
> *What are effective ways of visualizing synchronization?* [Q4]
> *How can intrusiveness of the visualization be minimized?* [Q5]
> *What is the impact of visualizing this on the musicians' synchronization?* [Q6]

In this report the following aspects will be addressed, thereby answering the subquestions in the process. Firstly, a literature research will be done on synchronization and measurement of physiological signals. This includes a literature study on physiological synchronization, measuring and quantification of synchronization and emotions. Furthermore, effective ways of visualizing music and emotions are discussed. The conclusions from this research are then translated into a prototype design via ideation and specification phases. This prototype will be tested and improved through a few iterations in a realization process, in order to reach a conclusion and evaluate the final design. In the end, recommendations will be made for future work.

Method

For this research, the Creative Technology design method is applied. This method was developed by Mader and Eggink (2014). It consists of four main phases: ideation, specification, realization and evaluation. During each phase the possibility for diverging ideas and iteration exists, while at the end of each phase the concept should converge again before proceeding to the next phase.

In this chapter, various design methods are explained which will be used throughout this research project. They form a toolkit that can be used to apply the Creative Technology design method. These methods will be combined with a literature research and a state of the art research to realize the ideation and specification of the desired design. A prototype will be designed which is then tested and iterated upon in the realization phase, after which it is evaluated by the designer.

Expert interviews

In the ideation process, several expert interviews will be held. An expert interview is a structured interview with an experienced individual, whom has more knowledge of the field the designer is working in. Findings from an interview such as this one will help the designer in their understanding of the subject they are investigating.

Scenario-based design

To define the possible end users of the system, as well as get an image of the system's requirements, scenario-based design will be used. This includes user personas, user scenarios and use scenarios. Sketching a situation for a possible deployment of the system helps to guide the designer in realizing the intended use experience, as explained by Carrol (1999).

MoSCoW analysis

The MoSCoW analysis[1] is a way of determining the final system requirements. It consists of listing every requirement for the system and dividing them into four categories. These are requirements which the system:

Must have: Requirements that are an absolute necessity.
Should have: Requirements that are important but not absolutely necessary.
Could have: Requirements that will benefit the system, but that will only be included if time and resources permit it.
Won't have (this time): Requirements that will not be implemented in the current system, but might be useful to implement in the future.

[1] https://en.wikipedia.org/wiki/MoSCoW_method

Background

To construct a system which visualizes the physiological synchronization of musicians, there are a few aspects that have to be investigated. To find out more about the state of the art, a literature review is done. This literature research focuses on a few of the research questions that were stated in the introduction. The main topics investigated will be the definition of physiological synchronization, the requirements for measuring and quantifying this synchronization, the connection between this and measuring emotion in a physiological way, and visualization using colours. Combining the findings from these researched topics with the rest of the state-of-the-art research will result in a conclusion with recommendations for designing the desired system.

Measuring physiological synchronization

For this research it is important to define physiological synchronization and what it is influenced by. To measure this synchronization as accurately as possible, it is important to define which bodily signals display effects of synchronization. Furthermore, the goal is to minimize factors that have a negative influence on the level of synchronization. An often used variable that can supply a measure of synchronization between multiple individuals is heart rate variability (HRV). McCraty (2017) states HRV as the primary method to measure physiological or personal coherence. Due to the heart's own magnetic field, it has a 'messenger' kind of influence on others in close proximity (within a few metres). The research done by Leskowitz (2008) supports these statements. HRV relates to the breathing rhythm as well; the combination between the two is called respiratory sinus arrhythmia, and in this process one influences the other, as explained by Yasuma and Hayano (2004). Therefore, both variables can be regarded as important indicators of the level of synchronization.

To clarify, these variables will be explained a bit further (see also Figure 1). Heart rate variability is the difference in time between each individual heartbeat, or in other words, the standard deviation of the inter-beat intervals (IBI). For more information and details about HRV, see Acharya, Joseph, Kannathal, Min, and Suri (2007). A high HRV is an indicator of rest and feeling at ease, whereas a constant IBI, or low HRV, indicates large amount of stress or being unhealthy. Respiratory sinus arrythmia is the process where the IBI increases on inspiration and decreases on expiration, mostly observed when a subject is at rest, as explained by Hirsch and Bishop (1981).

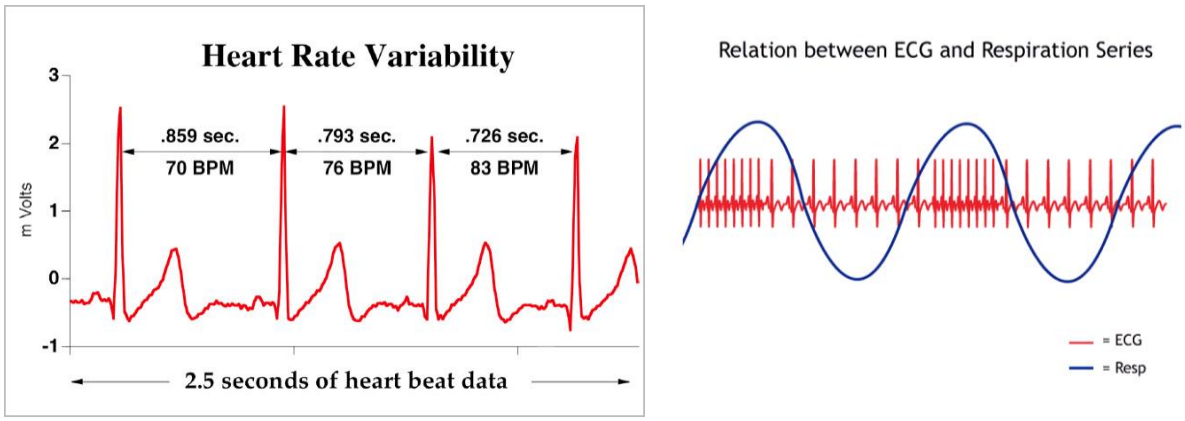


Figure 1: Left, a graph that shows HRV as being a difference in IBI. Right, a graphical representation of the RSA.[2]

[2] Image sources: https://dailyhealthpoints.com/2016/12/01/managing-your-physical-condition-with-heart-rate-variability/ and https://support.mindwaretech.com/2017/09/all-about-hrv-part-4-respiratory-sinus-arrhythmia/

As breathing often occurs at the same time in groups of singers or wind instrument players, this has an effect on the HRV as well, and thus on the overall feeling of synchronicity. Vickhoff et al. (2013) prove that this is indeed the case. Their experiments calculate the RSA and HRV of a group of choir singers for songs that have both simultaneous and non-simultaneous breathing. Their results show that synchronization levels are indeed higher in songs with a simultaneous breathing pattern, however, they also show that synchronization is present in both cases. Müller and Lindenberger (2011) did a similar experiment, confirming the presence of synchronization in HRV and respiration rhythm between choir members. Moreover, they found that there was more synchronization with songs that were sung in unison, as compared to songs with different parts. This shows that HRV and breathing are influenced by synchronization between singers and influenced the most when a similar part is sung.

Another aspect of the human body that has a link with synchronization between people is the brain. As discussed earlier (see *Synchronization in practice*), there exists a feeling of anticipation between persons that tells them when the other makes its next move, similar to how a herd knows when to move away from danger when one animal senses it. This happens partially via brain waves, as shown by Rizzolatti and Sinigaglia (2010). Furthermore, Barnett and Cerf (2017) show that brain waves synchronize between cinema visitors watching the same movie. Another brain wave synchronization process, this time between guitar players playing together has been shown by Lindenberger, Li, Gruber, and Müller (2009). However, because of the complexity of these signals, they will not be discussed further here.

Looking at the rhythm matching between musicians, Hennig (2014) shows that this form of synchronization goes deeper than just audible rhythm synchronization. The pianists that participate in this study show that there exists a collective rhythm memory while playing together, which is based upon the time played together, going back several minutes. The study indicates that the human brain plays a significant role in this specific synchronization process, in a similar way as the studies discussed earlier.

Measuring and quantifying emotions in synchronization

Apart from defining the amount of physiological synchronization, measuring whether the musicians perceive the same emotions and visualizing this can also support the concept of synchronization, as a musician's physiological state is heavily affected by the way they are feeling at that moment and how they experience a song they are playing. There are various available ways of measuring perceived emotions. Among others, heart rate (HR), heart rate variability (HRV), respiration rate and galvanic skin response (GSR; also named skin conductivity) are named frequently (Bradley & Lang, 2000; Healey, 2000; Mauss, Levenson, McCarter, Wilhelm, & Gross, 2005; Vickhoff et al., 2013). Ouwerkerk (2010) names these and a few other methods as well, such as facial expression and body temperature. However, those variables are more difficult to measure accurately, and easily influenced by other factors. Therefore, they will not be focused on further in this research, and thus the focus will be on HR, HRV, respiration and GSR, since they are easier to measure.

The most common method of quantifying emotions is the Two-Dimensional Emotion Space (2DES). The reliability of this method was validated by Schubert (1999). As clearly described by Russell (1989), the method is based on combining two variables: the intensity/arousal (usually scaled from sleepy to arousal) and the valence (usually scaled from happy to sad) of the emotion (see Figure 2[3]). This system can be linked easily to the biological signals that are being measured, as can be seen

[3] Image source: https://www.researchgate.net/figure/Example-of-emotions-plotted-in-a-two-dimensional-emotion-space-2-DES-9-p-86_fig3_7021104

from the work of Ouwerkerk (2010). A matching emotion between subjects, defined as a combination of these two variables, could indicate a measure of synchronization. Ouwerkerk (2010) states that breathing rhythm variation and HRV affect both arousal and valence, whereas heart rate and GSR affect just the arousal. Similarly, Mauss and Robinson (2009) give insight into which behaviour and signals influence arousal and/or valence. Measuring HRV and respiration rate can thus give a more varied insight in the subject's emotions that HR and GSR measurements, based on the use of the 2D Emotion Space to quantify the emotions.

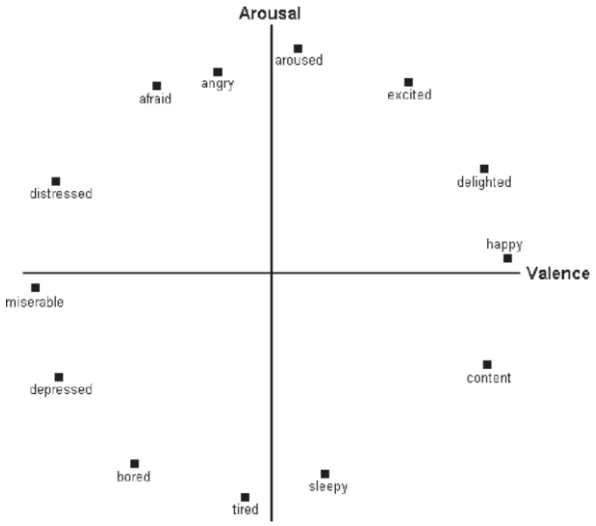


Figure 2: Examples of several emotions plotted in the 2DES

Other variables influencing measurements

There are various variables that have an influence on the level of synchronization between people in close proximity, which must be taken into account when measuring synchronization. Timofejeva et al. (2017) show that the earth's magnetic field has a synchronizing influence on the heart rate variability. however, more importantly they prove the significant influence of stress and the quality of interpersonal relationships on the HRV. This is supported by McCraty (2017), who states that social discord has a negative influence, whereas family members and couples generally synchronize more easily. Keeping stress level at a minimum and working with people having good interpersonal relationship should thus help in increasing the level of synchronization.

What kind of measurement equipment is used can influence the measurement results further. An overview of the aspects of various sensing methods is discussed by Ouwerkerk (2010). They state that if the measuring method is too intrusive, it will influence the emotions the subject feels, and therefore the measurement itself. It will also place the subject under a larger amount of stress. From this follows the conclusion that the sensors used should be as unnoticeable and non-intrusive as possible.

When measuring emotions, there are a few constraints as well. In the study of Healey (2000), it becomes clear that both GSR measurement and heart rate measurement are heavily affected by even slight amounts of physical activity. Since making music involves significant physical activity, data derived from the heart rate and GSR measurements will likely not be reliable enough. On the other hand, HRV measurements are heavily influenced by stress, as discussed before. Taelman, Vandeput, Spaepen, and Van Huffel (2009) confirm this. Furthermore, Iwanaga, Kobayashi, and Kawasaki (2005) show that different types of music have an influence on the HRV of listeners. On the other hand, according to Van Dyck et al. (2017), HR is influenced by listening to music as well, although the tempo of music does not have a direct effect. Conclusively, when performing measurements GSR and HR data will be less relevant, and the fact that HR and HRV will be influenced by the music that is being listened to should be kept in mind.

Visualizing synchronization and emotion with colour

A simple and effective way of making an effective visualization is using colour. Naz and Epps (2004) show that colours are rich in symbolism, based on which hue is used. For synchronization, this could mean choosing a colour which symbolizes unity and harmony will incite the desired association within the user. As can be seen from their research, this symbolism is highly subjective and varies

per person. Furthermore, the associations with different colours vary per society and culture. Therefore, it might be difficult to find one colour which symbolizes harmony for all participants in a study. In contrast, d'Andrade and Egan (1974) discuss that colour-emotion association is not culture-based. Furthermore, their research shows that the most influential effect on arousal and emotional association is reached by varying the colour's saturation, and not the hue. This is confirmed by Mikellides (1990). Less saturated colours will create less emotional response in a viewer.

Conclusion of the literature review

One goal of this research is to find out how physiological synchronization of musicians can be measured and quantified. From the research into the subject of physiological synchronization can be seen that that both heart rate variability (HRV) and respiratory sinus arrhythmia (RSA) are important variables for measuring synchronization, answering [Q2]. Furthermore, there are various measurement conditions that are useful to enhance the synchronization: singing or playing the same part, minimizing stress levels and testing with subjects that have solid interpersonal relationships. To keep stress at a minimum, the sensors used for measurements should be non-intrusive.

To quantify emotions, the Two-Dimensional Emotion Space can be used, scaling emotions based on valence and arousal. Four common physiological variables for measuring and quantifying emotions are respiration rate, heart rate variability (HRV), heart rate (HR) and galvanic skin response (GSR). Since the latter two are heavily affected by physical activity, they will not be reliable when measuring musicians actively making music. HRV and HR are both known to be influenced by music, which must be kept in mind. Furthermore, both the valence and arousal data needed can be derived from either the HRV or respiration rate, whereas HR and GSR only give a measure of arousal. This combined gives a preliminary answer to [Q3]

For visualizing the synchronization, a type of colour visualization can be used, where variations of saturation are used to incite different emotions or associations. Less saturated colours incite less arousal and emotional response. This partially answers [Q4].

A list of requirements for measuring and quantifying synchronization and emotions has been established in this research. However, there are several aspects and methods that have not been investigated. For further research, it would be good to look into ways to visualize musicians' emotions and synchronization back to them and their audience. This includes existing ways of visualizing music, but also non-intrusive ways of visualizing emotions. Furthermore, not all possible physiological signals that might play a role in human synchronization have been discussed. For example, what happens in the brain while playing together could provide valuable insights for this research, and there are other similar signals as well.

Ideation and State of the Art

In the ideation phase, the topic will be explored further using various tools, forming a broad basis on which the system can be built. To further solidify the concept, an image has to be formed of the scope of this project. This image will be obtained through a brainstorm session, moodboard, and a stakeholder analysis. The main state of the art research consists of expert interviews and observations from similar projects. User personas and a user scenario complete this overview. Combining the findings from both the literature review and the ideation process will result in recommendations for the next stage of the research and specifications for the system design.

Stakeholders

To start the ideation process, the stakeholders of this project are defined. The main stakeholders that can benefit from this research are the musicians using the system. Feedbacking the level of synchronization to them can improve their experience of playing, as well as improve the group harmony between them. If this is drawn a little further, the musicians' audience can be seen as stakeholders as well. Including the amount of synchronization in the light show at a performance, for example, can heighten the impact of a concert and improve the audience's experience of listening and seeing an artist perform.

Lastly, another stakeholder are the researchers investigating this topic at the University of Twente, or more specifically the Human Media Interaction group. If any important findings result from the experiments of this project, they might provide a starting point for continuing research into the subject, or help other investigation topics to progress further.

Project scope, brainstorm and moodboard

To narrow down the number of researchable topics, and trying to capture the scope of this research, a brainstorm was held. This brainstorm session was an individual session, done while keeping the literature research and found stakeholders in mind. The result is shown in Figure 4. After the brainstorm, the determined topics were made more tangible through the construction of a moodboard. This moodboard is found in Figure 3.

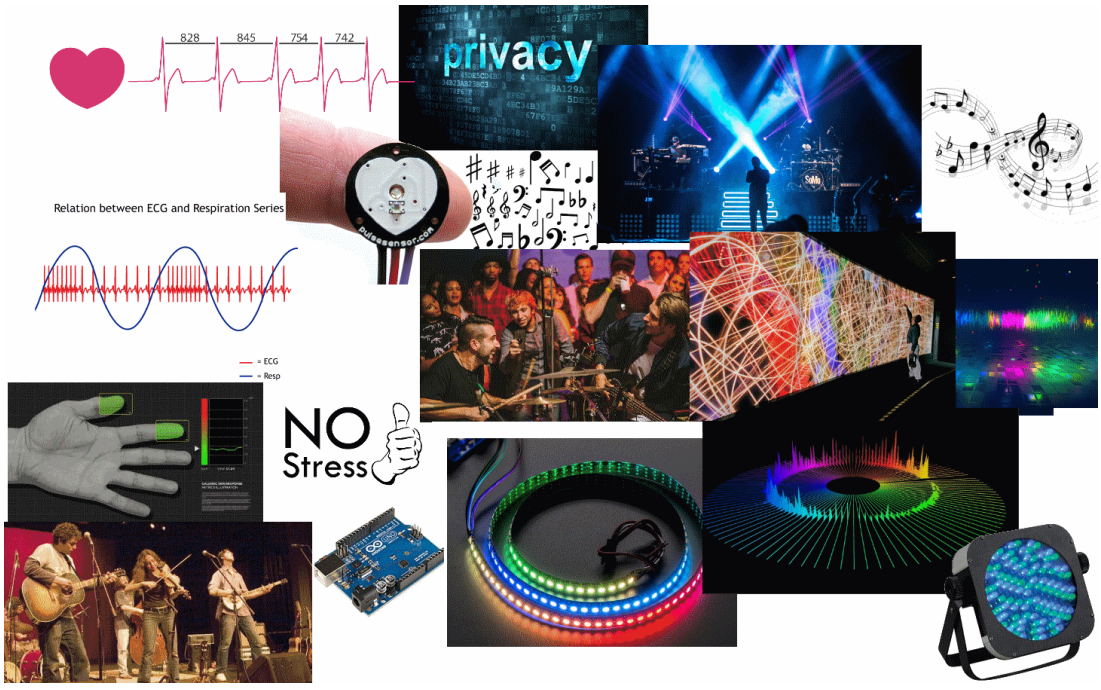


Figure 3: Moodboard showing the scope of the topic

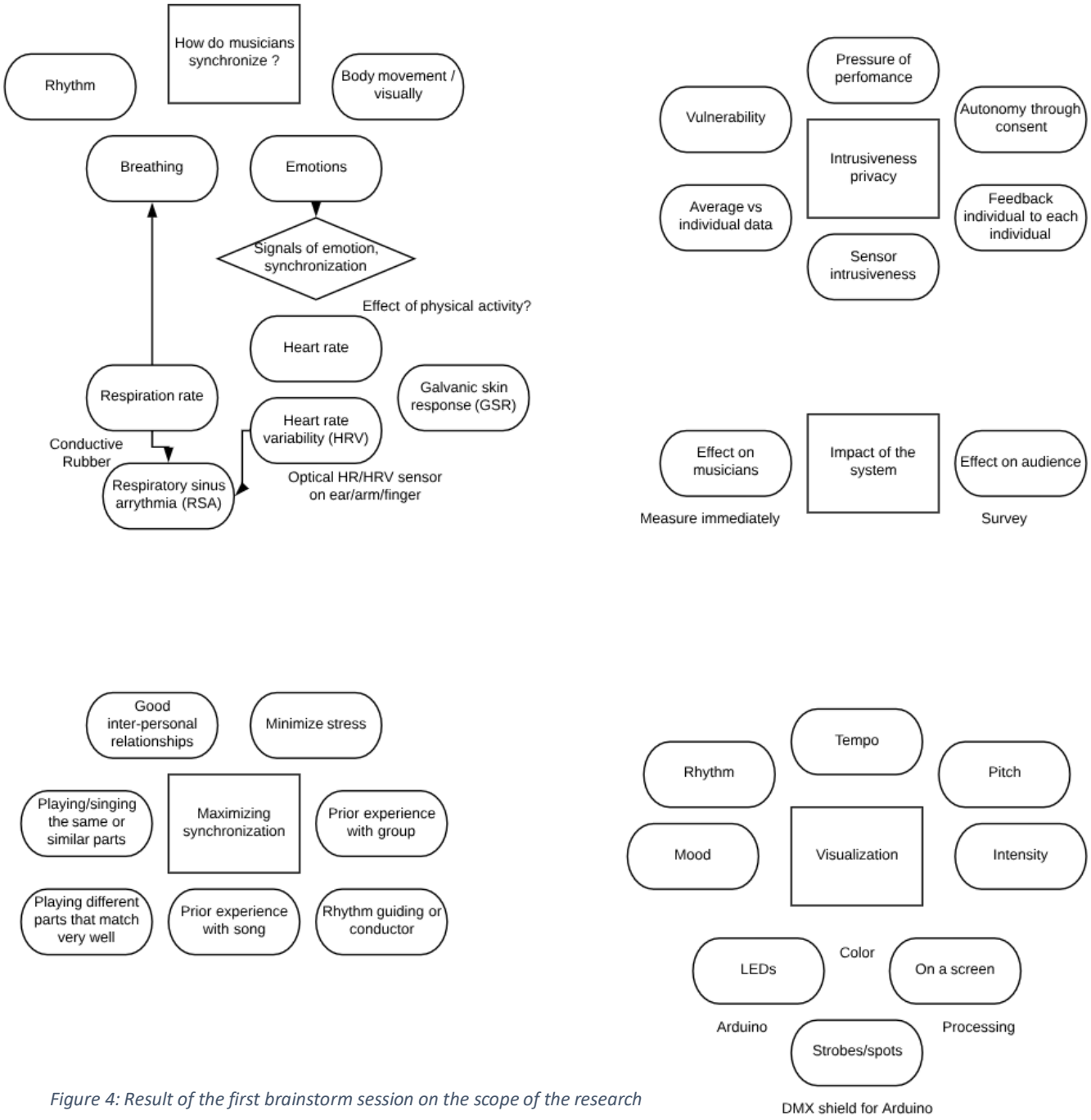


Figure 4: Result of the first brainstorm session on the scope of the research

As can be derived from the brainstorm results in Figure 4, the project was defined to have five core goals (shown as square blocks): measuring synchronization, maximizing synchronization, visualization, minimizing intrusiveness and having an impact. The subtopics and tools mentioned around each of these goals (shown as oval and diamond blocks) illustrate possible ways of achieving them that arose during the brainstorm session.

Next, to perform an ideation process about the topics introduced by the brainstorm and moodboard, five interviews are held. These interviews are integrated within a state of the art research.

Synchronization in practice

Looking at synchronization between musicians, something that immediately comes to mind is rhythm. Rhythm is a core element of most kinds of music and is usually a way to connect different players in a piece of music. This connection is usually achieved through percussion, bass, or in larger groups of musicians, a conductor or band leader, who gives the group visual cues for, among others, the common rhythm. Other types of synchronization include collective breathing (especially for singers and brass or wind players), and, as some people argue, some sort of energy which exists between people that are in sync with each other.

Interview with Gregor Mayrhofer

In an interview with conductor Gregor Mayrhofer[4] (*Appendix B – Interview*), he defines synchronization in an orchestra to be a combination of three things. Firstly, it is the musicians' imagination of what a piece should sound like by looking at it and by receiving initial instructions from the conductor. This also includes a memory of what tempo the musicians have been playing in before. Secondly, the musicians watching of the conductor and each other plays a role in finding the tempo. Lastly, it is the listening to each other to feel the anticipation of where and when the next note should be played. Mayrhofer stresses that it could be the musicians' collective breathing that plays an important role in this anticipation. Finally, he states that this anticipation is where the energy between people could play a role, although he believes that the three aspects listed above are far more relevant and, more importantly, easier to measure.

To find out more about the synchronization of an orchestra, Mayrhofer conducted experiments with both professional and amateur orchestras and ensembles, where the musicians had to close their eyes while playing, after receiving initial instructions and a tempo from the conductor. The result was that, sometimes after trying a few times, the musicians were able to play together without much trouble by just listening to each other. This indicates that they were able to determine when the next note should be played, based on listening to each other's anticipation and breath.

When asked about what the best way is for telling the musicians whether they are in sync with each other, Mayrhofer states that visual stimuli might not be the best way of showing them, since it might distract the musicians from listening to each other, which is probably more important for the overall synchronization process as described before.

CLASSSH

To add to the subject of distraction by visualization, the principle stated by Mayrhofer can be compared to a project called CLASSSH[5], a concert where classical music and pop culture comes together. It uses video mapping to show projections while an orchestra is playing. This video projection is based on the sound intensity in the concert venue. Since it just follows sound intensity and not the rhythm and mood of the music, it could feel out of place and might even distract the viewer from looking at the orchestra.

[4] Gregor Mayrhofer is a German conductor and composer. See also http://www.gregor-a-mayrhofer.de
[5] CLASSSH: https://classsh.nl/

Measuring physiological synchronization

Interview with Kirstin Neidlinger

To see the impact of breathing and other physiological signals, it is important to get an idea of the necessary requirements for measuring them. To gather practical information on the subject, an expert interview was held with Kirstin Neidlinger, founder of SENSOREE[6]. She did a similar project on physiological synchronization, namely on visualizing the connection between two trapeze artists. The initial concept of this project was based on the belief that two people in close proximity start to synchronize their heartbeat or heart rate variation by a certain amount. Among others, the research institute of the Heartmath company[7] has done research on this.

The SENSOREE project with the trapeze artists consisted of two wearable systems with sensors and visual actuators, based on the Arduino platform[8], connected via X-Bees. The signal that was being measured was the heart rate variability (HRV), the variation in time between each heartbeat. This system was used in a fashion show, where the two artists were sitting closely together wearing the system. It was a two-way visualization: the other person's signal was displayed on their chest, and their own signal was displayed on their hat. In this experiment, a certain synchronization between the two artists was visible, just by having them sit closely together without moving. A notable comment from Neidlinger on measuring in ways like this, is that it should be as non-intrusive as possible, lest the stress from being measured will impact the measurements too much. For heart rate, this could mean using a pulse oximeter sensor – a type of heart rate sensor that uses infrared to detect the subject's pulse in a less intrusive way than by using an electrocardiogram (ECG).

Lastly, Neidlinger mentioned another way of measuring a person's emotion and thus maybe synchronization as well: facial expressions. Measuring these can be achieved through the use of a high-definition camera in combination with classification algorithms. An application for this in music can be found in Sight Machine[9], a project that uses artificial intelligence to add another dimension to a musical performance. However, due to the complexity of the necessary algorithms, using video to identify facial expressions will not be pursued further in this research.

Interview with Emiel Harmsen

Furthermore, an interview was held with Emiel Harmsen, a master student in Human Media Interaction at the University of Twente. He developed a board (for the SENSOREE company) which can be used to combine different biodata to quantify the emotions of the user. The reason for this interview was to find out whether the board is suitable for use with this project, and whether he had useful findings about this field of research.

His design, the Escalade, incorporates the measurement of various physiological signals into a single device. Heart rate is measured via ECG, after which the signal is filtered to output the heart rate, from which later on the heart rate variation (HRV) is calculated. The system uses a strap of conductive rubber to measure the respiration rate, and electrodes to measure galvanic skin response (GSR), also called electrodermal activity (EDA). These electrodes were designed specifically for measuring the EDA. Testing the system, it was possible to receive clear heart rate data (and thus also clear HRV data) and clear respiration data. Measuring EDA proved a little more difficult, although he does give recommendations on how to improve the results for this.

---

[6] SENSOREE is a company that focuses on bio-responsive design & wearable technology: http://sensoree.com/
[7] Heartmath research institute: https://www.heartmath.org/research/
[8] Arduino platform: https://www.arduino.cc/
[9] https://www.wired.com/2017/04/unsettling-performance-showed-world-ais-eyes/amp

One interesting thing that he discusses as well is the link between heart rate and breathing: breathing can be extrapolated from heart rhythm, since the heart rhythm is different when the subject is breathing out or breathing in. This is due to the respiratory sinus arrythmia (RSA), which has been discussed in the literature review as well. Moreover, from these two interviews it becomes clear that synchronization shows in the same physiological signals as emotions.

Lucem Sanae – a similar project

Interview with Jur van Geel

Subsequently, Jur van Geel was interviewed. He is a Creative Technology student from the University of Twente, who led a group project which has many similarities with this research (Data Physicalization module 8 – Lucem Sanae). Their project had as main goal the visualization of data for the Orkest van het Oosten[10]. Based on interviews with the orchestra, they decided to make it a form of art, instead of a function visualization. The idea was to animate a concert venue based on biodata of the orchestra members.

If this would have been done in a functional way, the musicians' physiological signals, such as heart rate, heart rate variation, blood pressure, eye movement, breathing and body movement could have been applied to support the musicians, giving them feedback that could help them in improving their playing. However, the musicians stated that they wanted this data only to be feedbacked to themselves, so that it was not possible for other people to view and analyse their data and tell them what they are doing right or wrong. Visualizing this could have been interesting to the musicians themselves but not specifically to their audience. Since the orchestra wanted to make their performances more attractive to their audience, this option was not explored further. What is important to note about this, is the fact that musicians were not willing to display their biological data publicly. The musicians should give consent to their data being measured and shown, and the display of data should happen in an anonymous way.

In terms of sensors, their final system used infrared pulse sensors for detecting heart rate, two accelerometers connected to the chest for measuring respiration and a glove with another accelerometer to measure arm movement (specifically for string players). The heart rate sensors used were non-intrusive and gave good results, however, the method for measuring breathing was disturbed by the musicians' physical movements. Their suggestion is to use a strap of conductive material connected to the diaphragm instead. Lastly, they wanted to use video analysis as well, however, this proved too complicated for a larger group of musicians.

For visualizing their data, they used strobe lights and spots, controlled by the DMX shield for Arduino[11]. This proved to be a simple way of adding colour to their visualization. They also used a beamer to project computer-generated visualizations. They stress that colour is a nice way of contributing to something and works well for visualization, whereas most other ways of visualization become intrusive rather quickly.

The visualization contributed to the experience of the musicians in a positive way. Although some musicians were slightly sceptical and nervous beforehand, this tension contributed to the group dynamics, since they all felt the same way. They enjoyed making music while using the system. Although the audience was small when the system was used, they enjoyed the experience of watching a musical performance in this way as well.

[10] Orkest van het Oosten, an Enschede-based concert orchestra: https://www.ovho.nl/
[11] About the DMX shield for Arduino: https://playground.arduino.cc/DMX/DMXShield

Privacy and vulnerability

Interview with Birna van Riemsdijk

From an interview with Birna van Riemsdijk[12], a few things arose that must be kept in mind when designing this system. Apart from the benefits, such as possible enhancement of synchronization between musicians and the audience's experience of a performance, the system contains a few downsides as well. Most importantly, biodata is personal data. Showing this data to an audience can in any way be seen as a loss of privacy, depending on whether or not they are aware and consensual on the display of their data, in a way as argued by Lynch (2016). Musicians make themselves vulnerable by opening up a part of themselves, as was confirmed by the musicians participating in the Creative Technology project mentioned earlier. The question here is, is this a part of themselves that they do not want to show? The data itself can be protected by not storing it locally or uploading it to a cloud, but showing it alone is also a form of 'giving it away'. If a viewer wants to cause any of the musicians harm, in for example a social manner, the system might allow them to do this (confirming vulnerability as defined by Coeckelbergh (2013)). Furthermore, when showing the individual data of each musician, it might raise their stress levels significantly because of this earlier mentioned intrusiveness. Stress will have a negative effect on the results, and it is possible that, because of this stress, the goal of the system cannot be reached.

There are a few possible solutions to these problems. One obvious solution is giving the musicians the right to choose, only letting those who approve of the showing of their data use the system. Lynch (2016) argues that in this way the value of autonomy is upheld, which can be seen as an important aspect of privacy, as well as freedom, as argued by Rössler (2018). Apart from this, the decision can be made to only use the system when musicians are playing together and not in a public setting. This decreases the number of people who have access to the personal data by a significant amount. Moreover, the visualisation can be manipulated in such a way that it serves its purpose without showing any identifiable data. A final, more rigorous option would be to make sure that, in the final system, no individual personal data is shown, only an average or an image of the group as a whole. This can be effective: showing a team-based visualization of work floor habits helped raise overall productivity in a project by Thales Research & Technology in Delft[13].

When comparing these options, a few questions arise. Even if musicians give consent on the visualizing of their data, does making music with their data on display not still make them feel too vulnerable and raise stress levels because of this? Furthermore, keeping the system in a setting with just the musicians will only help if the musicians know each other really well and have played together many times before. In this setting competition plays a large role and being the one not enough in sync with the rest can have a negative effect on that person's feelings and stress levels. Lastly, by showing only group data or averaged data, the system might not have the desired effect, with no clear output of how in sync every participant is. In the end, there might be a certain trade-off between how much privacy is lost and how effective the system will work.

Taking into account the vulnerability, loss of privacy and factors of stress mentioned earlier, it appears as if the downsides of these factors weigh heavier than the positive effect of the system. Given that in any case (either through stress levels or through not enough information), there is a possibility that the system will not work as desired, it is more important that the intrinsic values of privacy and autonomy are upheld. Therefore, the suggestion is to incorporate the system in such a way that no personal data is shown to either the musicians or their audience, limiting the display to

[12] M. Birna van Rijmsdijk is an associate professor in Intimate Computing at Delft University of Technology
[13] http://www.commit-nl.nl/news/reduce-stress-levels-and-increase-productivity

a result based on a combination of data from the whole group or abstracted data so that the personal data is not distinguishable anymore.

Personas and user scenarios

The following personas illustrate possible users of the system. For each persona, a user scenario is given which illustrates in what context the user would use the system. These personas describe fictional characters, not connected with any existing individuals.

Persona of John – band member of the Groovy Boots

John (Figure 5) is 25 years old and works at a large consultancy company. He just finished his Master's degree in business economics and now works full-time. He lives alone, plays guitar, loves listening to rock music and enjoys social engagements. One of his ways of finding this social engagement is his amateur band, the Groovy Boots, which consists of a group of friends from his student days. Every Thursday night the group comes together to have a jam session and rehearse a few songs. Their music already sounds quite nice, but John thinks they could sound better if they listen more to each other and try to get more in "the flow". John dreams of the day when he and his friends can play on a bigger stage, with a large enthusiastic audience.

Figure 5: John

User scenario of John

When the band is discussing their next repertoire, John suggests the use of a system which gives the band feedback on how well they synchronize with each other. His band members agree on trying out this system. While playing, the band members receive visual feedback on how well they align, based on their physiology. After the system helps them to get more and more in sync, the members start to enjoy playing much more. John is already thinking about whether this system can also be used in one of their performances, heightening their audience's experience of the show.

Persona of Richard – conductor of the national philharmonic orchestra

Richard (Figure 6) has been an orchestra conductor for the majority of his life. At 48 years old, he still enjoys having a large group of musicians harmonize under his directions. His work is his life, not having done much else. He has a wife and two daughters, and all three love playing the cello. Although Richard's education in music was a long time ago, he is still eager to find new ways to make his conducting more effective and enjoyable. He knows a lot about the factors that make the musicians in the orchestra synchronize with each other, and would like to find some way to make these factors tangible.

Figure 6: Richard

User scenario of Richard

Asking a group of researchers from the local university to assist him in his search for harmony, Richard agrees to the implementation of their suggestions into the orchestra rehearsal room. Richard now gets visual feedback on how well the orchestra synchronizes physiologically while playing as a group. The orchestra members also get feedback, but this time individual, to see how well they fit in with the rest of the group. Richard now knows in which songs the musicians have trouble aligning, so that he knows where he has to invest extra rehearsal time. Also, his orchestra members now know when to listen better to the rest of the orchestra, based on visual cues from the system. Richard appreciates the way that new technologies are improving his rehearsal quality.

Ideation results

Several preliminary conclusions can be drawn from the interviews and other ideation up until this point. Summarizing, the most important points are as follows. Listening, visual cues, and breathing are among the most important factors in synchronizing musicians. Furthermore, the heart rate variability of people being closely together synchronizes. Galvanic skin response could also be a measure of synchronization but is harder to measure than heart rate and respiration rate. The signals that indicate synchronization are the same signals that are affected by emotions. Non-intrusive and effective ways of measuring physiological signals include infrared pulse heartbeat sensors for measuring heart rate and conductive rubber straps for respiration. These findings complete the answer to [Q2] and [Q3].

Musicians should give consent to their data being used and how their data is visualized. To minimize stress and pressure, group data or abstracted data can be shown instead of personal data. These suggestions answer [Q5]. In visualizing synchronization, the visualization should not distract either the musicians or their audience.

Conclusively, a list of recommendations for measuring and quantifying physiological synchronization of musicians, has been established with this research, answering the first few research questions stated in the beginning to a sufficient extent. This includes recommendations for a visualization of the physiological data. The result can be seen in Table 1. By combining these recommendations, a system can be made which can be used to measure if musicians synchronize, and whether feeding back this synchronization to them helps in increasing the overall feeling of coherence.

Measurement methods	**Conditions**	**Privacy/vulnerability**	**Visualization**
Heart rate variability	Minimized stress	Only group data	Colour (saturation)
Optical pulse sensors			
	Good relations	User consent of	Arduino with DMX
Respiration rate		visualizing their data	
Conductive rubber straps	Well-known songs		Screen
Unobtrusive sensing	No distractions		LEDs

Table 1: Recommendations for the desired system, based on the results of the ideation phase

Specification

In the specification phase, the concept from the ideation phase and background research is solidified into requirements for designing the system. Starting point for the specification are the results of the ideation process (Table 1), summarized as the following recommendations:

Measuring the subjects' HRV and respiration rate and using them to determine the RSA will suffice for the scope of this research. The sensors used should be as unobtrusive as possible.

As for measurement conditions, the subjects tested on should have good interpersonal relations, experience a minimum amount of stress and play similar, well-known parts to maximize the amount of synchronization. Furthermore, they should consent to the use of their data and the way that it is displayed. Possible sensors that can be used are pulse ear clip heartbeat sensors to detect heart rate and HRV, and conductive rubber straps to measure respiration.

For visualizing music colours can be used that vary in saturation according to how synchronized the musicians are, with the saturation increasing with the synchronization. An example would be to use Arduino with a DMX shield to set up stage lighting in the space of the experiment, or to use a screen showing varying colour saturation.

Design requirements

The recommendations above can be combined in a list of design requirements for the system. This includes two iterations of a MoSCoW analysis, one regarding the system itself (Table 2), and one regarding measurement conditions (Table 3). The requirements will also be subdivided into functional and non-functional requirements.

System requirements

The system must have unobtrusive sensing methods, to reduce the chance of the musicians experiencing stress. Furthermore, the visualization should not distract the musicians, since being a distraction might reduce the feeling of being in sync if they cannot focus on their playing. It might cause stress as well, if the feedback causes the musicians to feel uncertain about them not being in sync as a group. As discussed before, the visualization should only show group data and no individual data to avoid a loss of privacy and the musicians feeling vulnerable.

For now, only HRV measurements are necessary, since it is expected that this will suffice for mapping the level of synchronization, partially because the breathing and HRV are a linked process. Therefore, the system won't have breathing measurements, but it must have HRV measurements. The individual data resulting from the measurements could be sent over a Wi-Fi wireless connection to a central point, so that the musicians are not constricted in their movement by wire connection between them.

The visualization will be kept simple in order not to make it intrusive, leading to the fact that the system could have a simple visualization using colour or light. The visualization won't have a visualization using a screen, or a visualization using DMX and stage lights, since these might distract the musicians more.

Must have	Should have	Could have	Won't have
Unobtrusive sensing	Show only group data	Visualization using colour	Breathing measurements
HRV measurements	Non-distracting visualization	Visualization using light	Visualization using a screen
		Wi-Fi data link	Arduino with DMX

Table 2: MoSCoW analysis of the system requirements

Measurement requirements

It is highly important that the users consent to the showing of their physiological data, even if only the group data is shown. Furthermore, stress levels have to be kept at a minimum since it has a large effect on the individuals' HRV. This includes external stress factors that have not been discussed yet, such as external stimuli.

To maximize the feeling of synchronization, the musicians should play well-known parts of music. They should have good interpersonal relationships between them, and to further add to this feeling, they could play similar parts of music.

Must have	Should have	Could have	Won't have
Users' consent of showing their biodata	Well-known parts of music	Similar parts of music	
Minimized stress levels	Good interpersonal relationships		

Table 3: MoSCoW analysis of the experiment requirements

Functional and non-functional requirements

The requirements listed above can be divided into functional and non-functional requirements. This is done in Table 4, not including the won't haves from the MosCoW analysis. Based on these requirements, a use scenario can be constructed.

Functional requirements	Non-functional requirements
HRV measurements	Unobtrusive sensing
Visualization showing only group data	Non-distracting visualization
Visualization using colour	Users' consent of showing their biodata
Visualization using light	Minimized stress levels
Wi-Fi data link	Well-known parts of music
	Good interpersonal relationships
	Similar parts of music

Table 4: List of functional and non-functional requirements of the system

Use scenario

This use scenario is a description of how the requirements are included in the system, forming an example of how the system could operate.

A band, consisting of six musicians, sits in a closed room with their instruments: two guitars, a bass, a keyboard, a saxophone and a drum kit. Each musician has their own simple heartbeat sensor, barely noticeable when playing. Their sensor is connected to a microprocessor they carry on their back in a backpack, so that the wires are not in the way. This microprocessor is connected to a Wi-Fi network and sends the musician's HRV data to a central data point. This central data point compares all the HRVs and uses the result in a visualization using coloured lights. When the musicians synchronize

their HRV, the visualization gives a clear feedback of only the group averaged data. This feedback does not distract the musicians from their playing.

The musicians have all signed a consent form in which they agree to the use and display of their data. They have played together many times before, know their own parts by heart and have good interpersonal connections between them.

Design choices

Based on these requirements and the use scenario, a design for testing a first prototype can be established (see Figure 7). Experiments to test this prototype will be conducted in a closed room, with little windows to avoid external distractions.

The experiments will be conducted with small groups of singers, since they do not move as much as various instrument players, making it easier to obtain accurate measurement results. Because of the little movement, an optical heartbeat sensor attached to their finger with a Velcro strap will suffice. Each singer will be connected to an individual Arduino microprocessor, which sends the resulting data to a central collection point via Wi-Fi wireless. From this central collection point, the visualization will be controlled. This visualization will be simple and non-distracting, showing only group data.

The singers will be asked to sing songs that they know well, and that they have sung together before. The individual parts of these songs should not differ from the rest too much. The singers will be sitting on chairs during the experiment, to reduce the chance of them becoming tired.

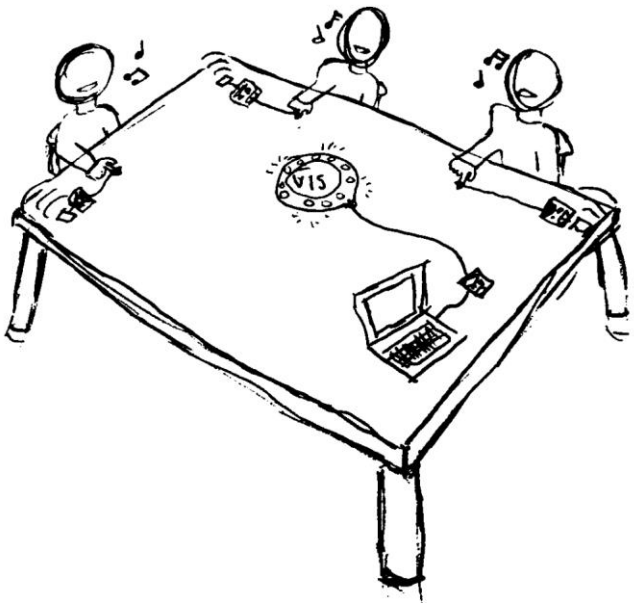


Figure 7: Sketch of the chosen design

This concludes the specification phase. With the design choices specified here, different prototypes can be designed, tested and iterated in the realization phase.

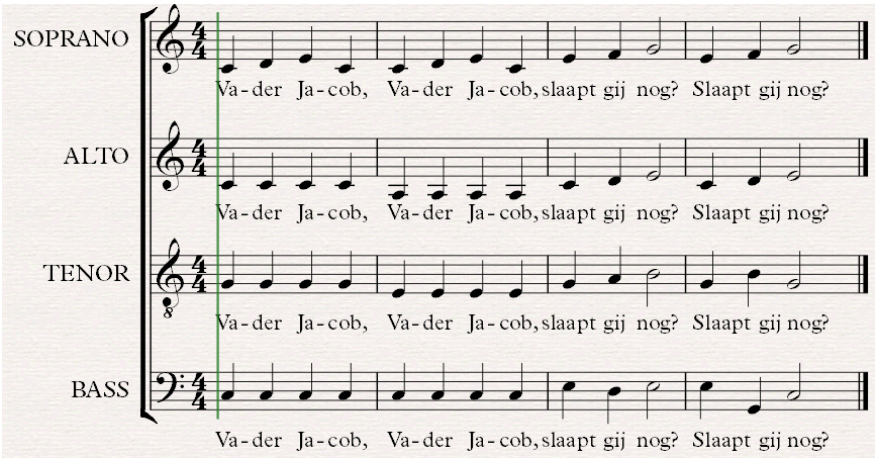


Figure 8: Example of a homophonic song: same rhythm for all voices and, in this case, varying notes

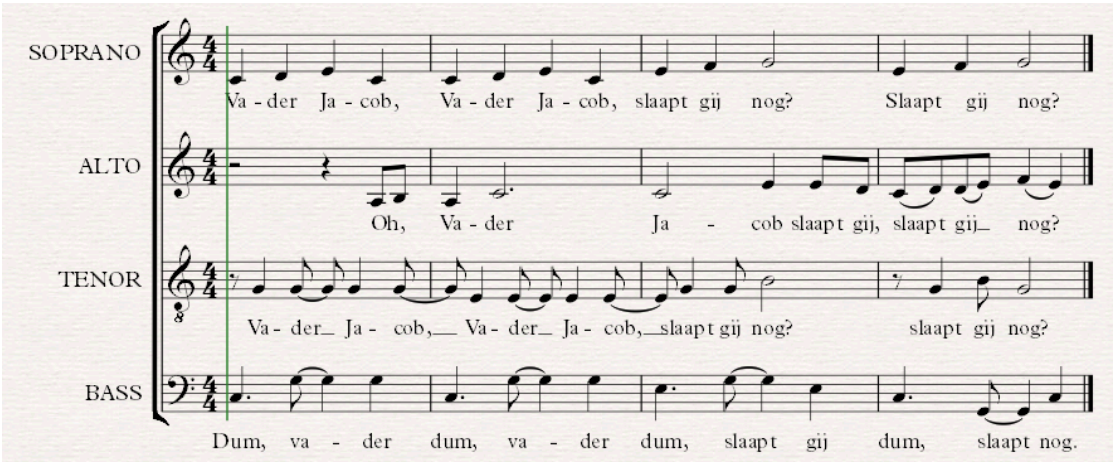


Figure 9: Example of a polyphonic song: different rhythms and different notes for all voices

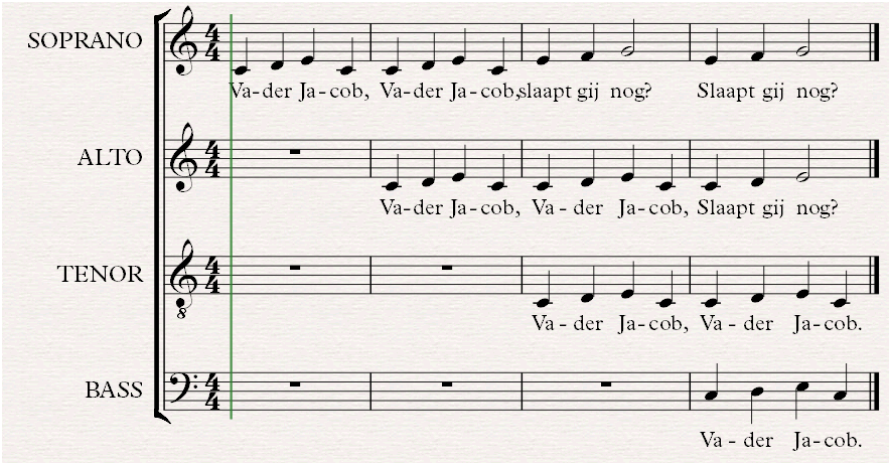


Figure 10: Example of a round: all voices sing the same line, but with a different starting point in time

Realization

In the realization phase, the requirements of the specification phase and the concepts from the ideation phase are combined into a series of prototype. Each prototype is built and tested with possible end users of the system, leading to changes which are then incorporated into a new iteration of the design. The findings from all (in this case three) iterations will be used in the answering of the research question stated at the beginning [RQ].

Iteration 1

In this first iteration, the heartbeat and heart rate variation (HRV) of three singers will be measured while they perform a few tasks. No visualization or any form of live feedback to the musicians will is used (yet). This experiment has a few goals. Firstly, the validity of the HRV measurement method has to be tested, as well as the calculations to obtain the HRV values. Secondly, the goal is to find out whether the singers indeed synchronize physiologically while singing. If this is the case, the values that are found can serve as a basis for threshold or trigger values for the visualization in the final system. Lastly, several practical aspects of the installation have to be tested, mainly concerning sensor stability and Wi-Fi reliability.

Experiment setup and conditions

Three singers (including the author) participated in this experiment, who were tasked with singing various songs in an a capella setting. During the singing, their heartbeat is constantly being measured, and from this their HRV is calculated. These singers have sung multiple times together before and have performed together as well. They have good interpersonal connections and are highly familiar with most of the songs that are sung. One song is more difficult and less known to them, and one other song is a round. See Table 5 for a full list of tasks performed during the experiment; six songs and one break. Before singing, the singers have had no vocal warming up, and they start each song in the right key through the use of a tuning fork. The experiment lasted for approximately 40 minutes.

Song	**Specifics**
Veerpont (I)	Really well-known song, singers have the same rhythm for most of the song (homophonic)
De Gezusters Karamazov	Really well-known song. One narrator, the rest sings different rhythms (polyphonic)
De Mens Is Slecht	Difficult song, less known, different rhythms (polyphonic)
The Woman of Ryde	Round: singers sing the same line but at different timings
Veerpont (II)	See Veerpont (I)
--- Break ---	--- Break ---
Dodenrit	Well known song. One narrator, the rest sings different rhythms (polyphonic)

Table 5: Experiment protocol and song specifics, iteration 1

In this context, a homophonic song (Figure 8) is a song in which the singers sing mostly the same rhythm, either singing the same notes or different notes. A polyphonic song (Figure 9) is a song in which the singers have both different notes and different rhythms. A round (Figure 10) is a song where the singers sing exactly the same, only they start at different times. In the research by Vickhoff et al. (2013), it showed that singers tended to synchronize less during polyphonic songs, and more during homophonic songs and the round. For this experiment, the outcome is expected to be the same.

At the end of the experiment, the singers were asked to fill in a short survey. In this survey they were asked to rate their feeling of synchronization during each stage of the experiment, as well as asked a few questions about their experience. This data can be compared to the actual HRV data to classify the results.

Hardware setup

The materials used in this experiment were (used in the setup as shown in Figure 12):

- Three Arduino Uno microcontroller boards
- Three Pulse Sensors
- Jumper wires (male-male and female-male)
- Breadboards
- Two esp8266 Wi-Fi modules (of which one was defected)
- Two plug and play battery packs for Arduino
- USB cable

The experiment uses three Pulse optical heartbeat sensors, each one connected to their own Arduino. The Pulse sensors are applied to the subjects' fingers with Velcro straps. The Arduinos are programmed to measure each inter-beat interval (IBI) of the test subject. For the code used in this experiment, see Appendix G – Code. These IBI values are then used to calculate an average and the standard deviation of the IBI. This standard deviation is assumed to roughly correspond with the actual HRV value. The program calculates the average and standard deviation of each five values that come in. These "sub-values" are then added to a sum of all averages or standard deviations respectively, and subsequently divided by the number of averages or standard deviations that were added to this number previously. After adding 100 averages or standard deviations, the totals reset and the process repeats. These totals are output each time they are updated. See also Figure 11 for a clarification of this process.

$$(Sub)average\ \mu = \frac{\sum_{n=0}^{n=4} IBI_n}{5} \qquad (Sub)\ standard\ deviation\ \sigma = \sqrt{\frac{\sum_{n=0}^{n=4}(IBI_n - \mu)^2}{5}}$$

$$\mu_{total} = \frac{\sum_0^n \mu_n}{n} \qquad \sigma_{total} = \frac{\sum_0^n \sigma_n}{n}$$

Here, the IBI is the inter-beat interval, μ is the average over the past 5 IBIs, σ is the standard deviation over the past 5 IBIs, μ_{total} represents the total average, from which the BPM can be calculated, and σ_{total} represents the total standard deviation, which is the HRV.

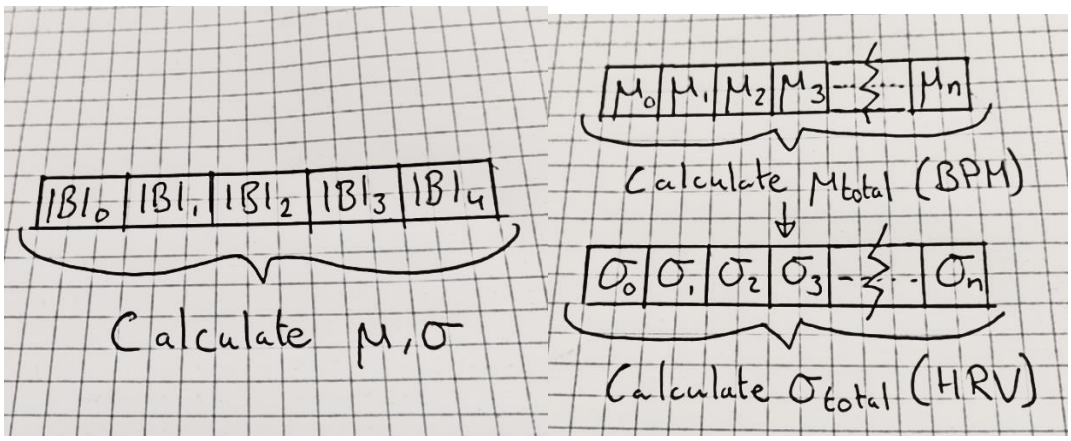


Figure 11: Clarification of the HRV calculation formulas and the method for data storage

Two of the Arduinos are connected to an esp8266 Wi-Fi module (see Figure 12). This module is used to connect to a Wi-Fi network, a mobile hotspot in this case. Each time the standard deviation total is updated, the Arduino tries to send this data to a Thingspeak channel via a HTTP GET request. Thingspeak[14] is an online platform for sensor analysis, which includes a live feed of incoming data. Each sensor has its own Thingspeak data field within the channel. The results can be downloaded as .csv files, which can then be used for further analysis. Since each singer had their sensor connected to a separate Arduino, they were less limited in their movement freedom.

The third Arduino is connected directly to a computer via USB serial. This is due to an unforeseen malfunction in the third esp8266 module, rendering it impossible to create a Wi-Fi setup for all three singers at the time of the experiment. The output values are sent to the serial monitor of the Arduino IDE (Integrated Development Environment), from which they are directly extracted and put into a .csv file.

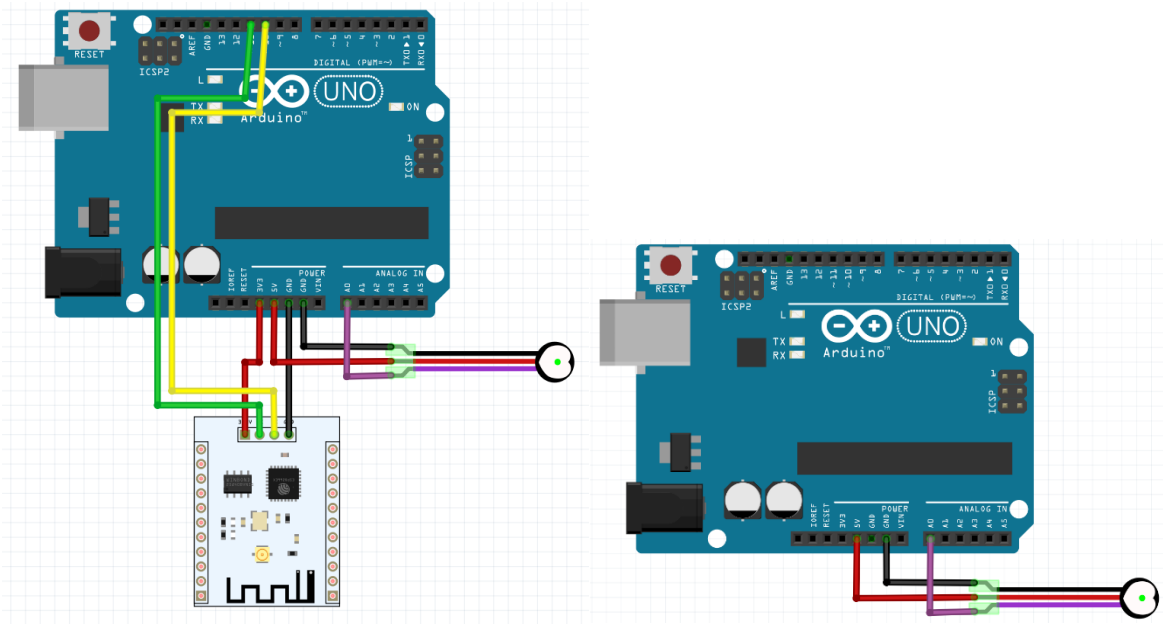


Figure 12: Frizing sketches of both the used Pulse sensor setups. Left, an Arduino connected to WiFi via an esp8266. Right, the Arduino connected via USB serial.

[14] Thingspeak is an online IoT platform developed by Mathworks (the developers of MATLAB): https://thingspeak.com/

Results and discussion iteration 1

The measurements that were sent over Wi-Fi gave a clear image of the HRV over time (see Figure 13, left). However, the GET request to update the feed only worked for a small number of samples, which is why the dataset has been limited to roughly one or two samples per minute, instead of recording data every five heartbeats.

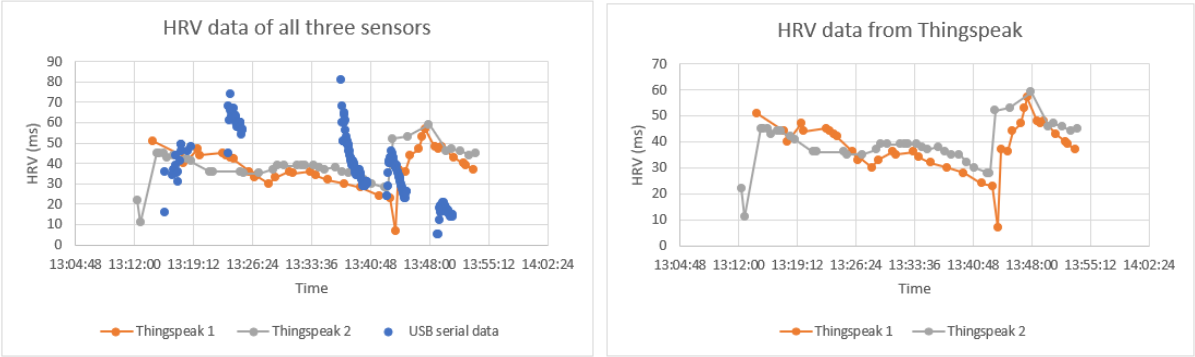


Figure 13: Left, results of all three sensors in one graph. Right, only the results from the two Thingspeak channels.

The Pulse sensor that was connected to the Arduino with USB serial connection did not function properly throughout the whole experiment. This resulted in missing data for parts of the experiment. Furthermore, the data that was obtained was unreliable to a certain extent, since the sensor seemed to detect heartbeats in between the actual beats sometimes. This resulted in a pulse that seemed twice as fast. In Figure 13, all three sensor results have been plotted in the left graph. When this data is compared to the right graph, where data from the third sensor has been left out, it can be seen that the values of the other two sensors appear to be much more realistic. Therefore, the results of the third sensor will not be included in the final discussion.

When looking at the data from the other two sensors (Figure 13 and Table 6), a few things can be noticed immediately. Both sensors seem to reach a minimum around 13:43, before rising again to a maximum right afterwards. The rise corresponds with the break, the period of rest in between songs. This suggests that on average, the HRV of the singers was higher during rest than it was during the singing (around 60 milliseconds at its peak). Furthermore, the HRV seemed the lowest while singing the song “Veerpont” for the second time, while this song was one of the best-known songs, and the singers rated it as one of the highest in terms of feeling synchronization. Possibly, the HRV decreased over time due to multiple songs being sung after each other, since the first song, “Veerpont”, and the song after the break, “Dodenrit” resulted in the highest average HRV. In general, when one of the two HRVs drops or rises, the other does so as well. This indicates some form of synchronization between the singers, both while singing and during rest.

Experiment phase	Sensor 1 average	Sensor 2 average	Average both sensors	Stdev both sensors	Average rating by singer
Veerpont (1)	45,50	43,63	44,56	1,33	2,00
De Gezusters	39,57	35,33	37,45	3,00	3,50
De Mens Is Slecht	34,67	38,50	36,58	2,71	2,50
The Woman of Ryde	34,00	38,25	36,13	3,01	4,00
Veerpont (2)	27,33	33,60	30,47	4,43	3,50
---Break---	40,64	44,00	42,32	2,38	3,50
Dodenrit	39,75	45,60	42,68	4,14	3,50

Table 6: Averaged HRV per experiment phase, and standard deviation between them compared to singer rating, iteration 1

In contrast to the expectations, there does not seem to be a distinguishable difference between a round and a regular song, nor between a homophonic song and a polyphonic song. When comparing the averages to the singers' rating of experience, no clear pattern can be seen (Figure 14, right graph). The singers rated their experience on a scale of 1-5, where a 5 means they really felt in sync, and a 1 means they did not feel in sync at all. However, when looking at how far the HRV of the two singers is apart by calculating the standard deviation, a pattern seems to emerge. When the standard deviation between the singers increases, their rating of experience roughly increases as well, as can be seen in Figure 14 (left graph).

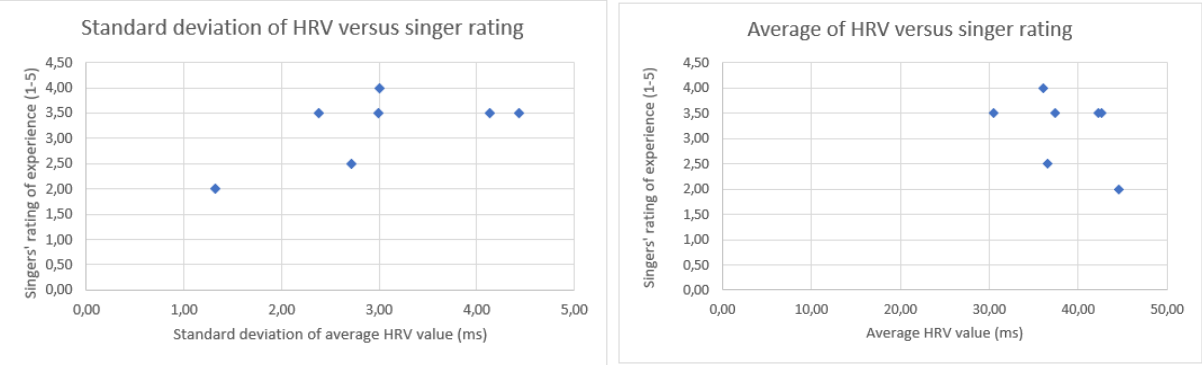


Figure 14: Singer rating per song compared to the difference in HRV between them (left) and the average HRV of the singers (right) . Graphs based on the data from Table 6.

Conclusion

Due to the small number of samples and test subjects, it is hard to draw solid conclusions based on these results. It is not possible yet to determine a clear value or threshold to base a visualization on. However, it can be seen that there is clearly a link between the HRV of the two singers. Furthermore, the HRV synchronizes both while singing and during rest, where it is higher on average during rest.

Finally, it is also important to note that the singers experienced none or little discomfort during the experiment. They were not obstructed by their wires in any way. This leads to the assumption that this way of measuring is, in this context, unobtrusive enough to obtain workable results. Moreover, since the Wi-Fi connection only limited the dataflow, wires should be used instead in the next iterations, as there is no direct constraint in limiting movement when measuring with singers.

Iteration 2

In the second iteration, a visualization will be added to the prototype. The main goal of this experiment is to find a threshold value to base this visualization on. To determine this value, the first measurements will be done without using the visualization. Other than that, the HRV determination method still has to be validated, since the results from the previous experiment were not satisfactory in this regard. Lastly, if the above two goals are reached, it will be possible to make a statement about the effect of the visualization, if there is a clear difference in values between measuring with and without the visualization.

Experiment setup and conditions

Five singers participated in this experiment who, again, were asked to sing multiple songs in an a capella setting. Their heartbeat is constantly measured, and from this, their HRV is calculated. The experiment consists of three rounds: one round of measuring without visualization, one round of measuring with a visualization that is manually triggered, and one round measuring with a visualization that is triggered based on the singers' HRV values. For a full layout of the experiment, see Table 7. The difficulty that is stated, is based on the singers' singing experience and their perception of how difficult a song is.

These singers have sung multiple times together as part of a choir and have performed together as well. They have experience singing a solo voice part, however, not necessarily in this group setting. They have good interpersonal connections and are highly familiar with all the songs that are sung. Before singing, the singers have had no vocal warming up, and they start each song in the right key through the use of a tuning fork. They receive visual tempo cues from a conductor. The experiment lasted for approximately one hour and 20 minutes. After the experiment, the musicians were again asked to fill in a survey about their experience.

Round 1 - Song	**Specifics**
Lean On Me (I)	Simple song, homophonic for most of the song
I Still Haven't Found What I'm Looking For	Difficult song, homophonic for most of the song, sometimes the lead stands out
Human	Medium difficulty, lead switches between voice groups, the rest of the voices have a homophonic accompaniment
--- Break ---	--- Break ---

Round 2 - Song	**Specifics**
Wonder Woman	Medium difficulty, polyphonic, the lead switches a lot
Oceaan	Medium difficulty, homophonic for most of the song
Nara	Medium difficulty, slow song, polyphonic, the lead switches a lot

Round 3 - Song	**Specifics**
Atlantis	Difficult song, lead switches between voice groups, the rest of the voices have a homophonic accompaniment
Wintersong	Medium difficulty, homophonic for most of the song, sometimes the lead stands out.
Lean On Me (II)	See Lean on Me (I)

Table 7: Experiment protocol and song specifics, iteration 2

The singers are given no prior instructions regarding the effect and operation of the visualization, to ensure that they are not biased or influenced beforehand. Other than that, the experiment is explained to them in its entirety.

Hardware setup

The materials used in this setup were as follows:

- One Arduino Uno microcontroller board
- Five Pulse Sensors
- One Adafruit NeoPixel LED strip, 5 metres long
- Four basic blue LEDs
- Four 1 kΩ resistors (for the LEDs)
- One 10 kΩ resistor (pull-up for the button)
- One 1 µF capacitor, non-polarized
- One basic pushbutton
- Jumper wires
- Four small breadboards
- USB cable

There are many similarities between this experiment and the previous ones. The sensing method and HRV calculation method is the same, although everything is controlled by a single Arduino Uno instead of multiple ones. Each sensor is linked with an LED, which blinks with the heartbeat of the user for debugging purposes. Schematics for this setup can be found in Figure 17 and Figure 18.

The visualization consists of a NeoPixel[15] LED strip, of which only the first 35 LEDs are used. When the visualization is on, a blue light goes slowly back and forth across the strip, with a fading tail behind it. When the visualization is off, the blue light stays on, but it stays in its current location and stops moving. The visualization can be turned on or off manually by pressing a pushbutton, or it can be controlled from the code, using a calculated threshold value to activate it. To dim the light intensity from the LED strip, a cylinder of regular paper is wrapped around the strip with a diameter of about 10 centimetres. The ends of the cylinder are concealed with a ball of bunched up paper. For an image of the visualization, see Figure 16.

As a threshold value, the standard deviation between the test group members is used, averaged over the past 30 seconds. This gives an image of how close the HRVs of the singers are together, and thus the level of synchronization. The upside of using a running average over the past 30 seconds, is that any jumps in HRV data will not cause drastic direct changes in the running average, causing the visualization to switch quickly between on/off or behave in any other undesired way. The standard deviation is calculated every second, and the average as well in the following way (where n is the singer and t is the time in seconds, see also Figure 15):

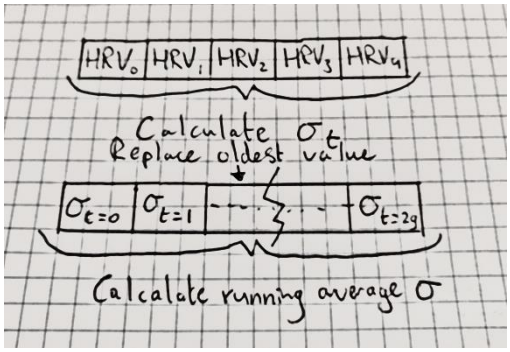


Figure 15: Clarification of the obtainment method for the running standard deviation in group HRV

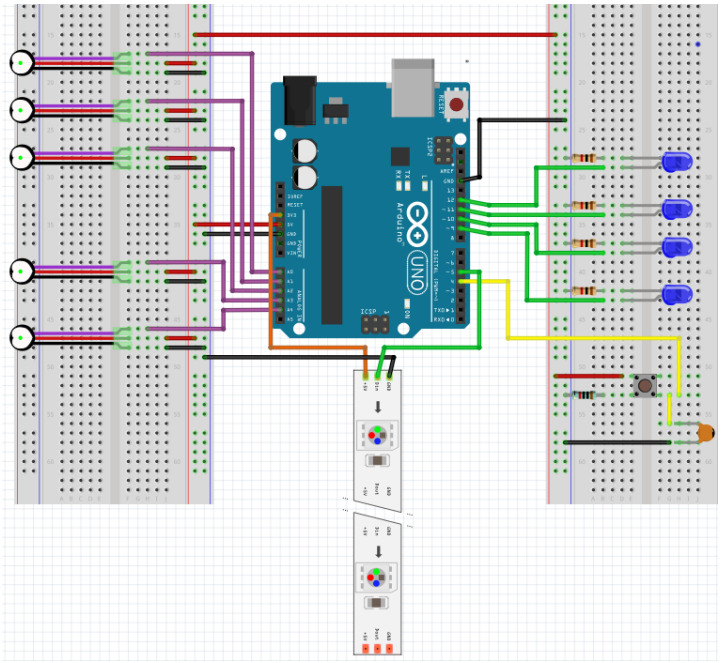
$$\sigma_t = \sqrt{\frac{\sum_{n=0}^{n=4}(HRV_n - \mu_{HRV})^2}{5}} \qquad Running\ average\ of\ \sigma = \frac{\sum_{t=0}^{t=29}\sigma_t}{30}$$

[15] https://learn.adafruit.com/adafruit-neopixel-uberguide/the-magic-of-neopixels

Here, σ_t is the standard deviation between all singers' HRVs at the second they are measured, and the running average of σ is the running average of the past thirty seconds' σ_t values.

If this running average drops below a fixed threshold, the visualization will turn on, and if it goes below this threshold, the visualization will turn off again. In the second round of the experiment, this turning on and off is triggered by pushing a button manually, when the observer of the experiment feels like the musicians are in sync.

The individual HRV values and this threshold variable are output via USB serial. On a computer, a Processing graph-writer sketch shows live graphs of this data and records the incoming values in a .csv file (see Appendix G – Code).



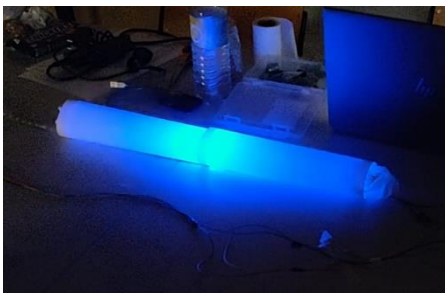


Figure 16: Image of the used visualization

Figure 17: Fritzing sketch of the Arduino setup, iteration 2 and 3, with five pulse sensors

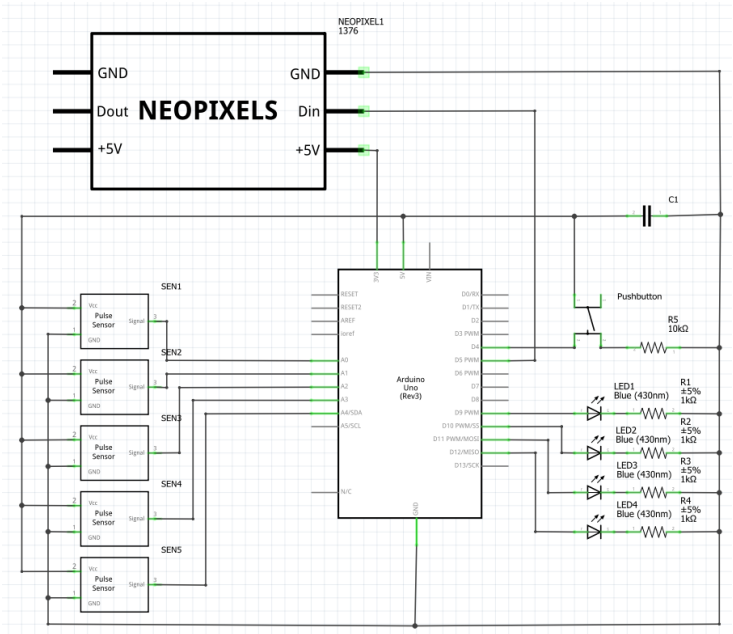


Figure 18: Electrical circuit diagram of the setup shown in Figure 17

Results and discussion iteration 2

After doing all rounds of the experiment, it became clear that the current HRV calculation method has some major flaws. The HRV is taken over 500 heartbeats before being reset, which makes it more and more accurate towards the point of being reset. However, after it resets, the data becomes highly inaccurate again, causing a sudden jump in the results. An example of this is shown in Figure 19. Looking at the overall overview of a round of measurements, it shows that this has a large effect on the rising and falling of the average standard deviation of the group (Figure 20). This inaccuracy makes it hard to do a proper analysis of the amount of synchronization, and unfortunately this means most of the data cannot be used properly. Also, since the measurement method was the same for iteration 1, those results are possibly inaccurate or unusable.

Figure 19: Resulting graphs from two songs measured with iteration 2. Sudden jumps in results are highlighted on the right.

Two thresholds were tried in the third round, but neither of them seemed accurate. The threshold was not set low enough, since the visualization kept moving as if the singers were in sync the whole time, although the data was inconclusive on this. Unfortunately, a part of the data of round 3 was lost, making it even harder to do a useful comparison between singer ratings of experience and data results. For a full overview of the group averages and standard deviation, see Table 8.

To see the effect of the visualization, however, round 2 contained a manually controlled visualization to see if it had any effect on the singers. Although the HRV measurements are not able to show much results due to the reasons mentioned above, the musicians were asked about the impact of the visualization in the Survey. The visualization scored a 2.2/5 on its impact in round 2, and a 2.6/5 on its impact in round 3. This means the musicians saw some difference between using a visualization and no visualization, albeit a small one. The visualization was not a distraction, which is good, scoring only a 1.6/5 on how much it distracted in both round 2 and 3.

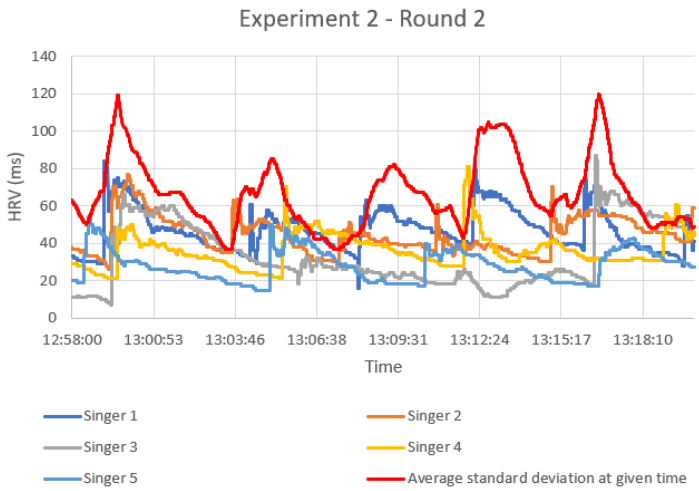


Figure 20: Resulting graph of the second round of measurements, iteration 2

Experiment phase	Average HRV of singers combined	Average standard deviation between singers	Average rating by singers / enjoyment
Lean on Me (I)	27.54	73.98	4.2
I Still Haven't Found	32.26	88.16	4.0
Human	33.93	65.36	4.0
---Break---	38.46	66.87	3.4
Average Round 1	**31.32**	**76.70**	**4.6**
Wonder Woman	37.22	57.59	3.6
Oceaan	34.67	72.88	4.8
Nara	41.59	67.35	4.0
---Break---	n/a	n/a	n/a
Average Round 2	**37.62**	**66.39**	**4.8**
Atlantis	31.80	66.48	3.0
Wintersong	n/a	n/a	4.6
Lean on Me (II)	n/a	n/a	3.8
Average Round 3	**n/a**	**n/a**	**4.6**

Table 8: Averaged HRV per experiment phase, and standard deviation between them compared to singer rating, iteration 2

Conclusion

Because of the major inaccuracies in the HRV obtainment method, a new HRV calculation method is needed. This method is thus not validated. The new method should use a running average to calculate over a sliding window of data. Other than that, the experiment setup was successful, so no large adjustments have to be made.

The threshold variable to trigger the visualization has to be chosen as a lower value. Furthermore, the musicians noticed the effect of the visualization, but in a minimal way. It was no distraction to them, leading to the conclusion that this visualization is unobtrusive and can be used in the third iteration again.

On the other hand, what can be learned from this experiment is that the sensor methods were experienced as unobtrusive, despite not using wireless communication. This leads to the conclusion that the method for obtaining measurements is a suitable one for this system.

Iteration 3

The third iteration is very similar to the second iteration. The main goal is still to find a variable to base a visualization on. Furthermore, the HRV measurement method still has to be validated, and in this experiment a new method will be tested. Again, if both these goals can be met, a conclusion can be drawn on how effective the visualization used in this experiment is.

Experiment setup and conditions

Before the actual experiment, the new HRV measurement method was tested by using the setup without musicians, measuring instead the HRV of a small group of people at rest, doing nothing more than talk to each other for around ten minutes. The resulting data had no jumps any more, and the values were as expected, without inexplainable behaviour. See Figure 21. The first few minutes of data (up until the red line) are inaccurate, since the system needs a few minutes of data to work, as is explained further ahead.

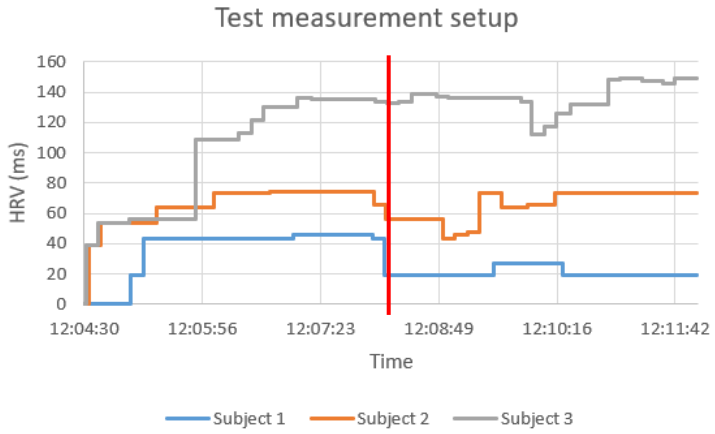


Figure 21: Testing the new HRV measurement method

Four singers participated in the actual testing of the third iteration. The test consisted of two rounds, one round measuring without the visualization, and one round measuring with the visualization. In the first round, a threshold variable will be chosen to trigger the visualization in the next round. An overview of the experiment can be found in Table 9. Before singing, the singers have a minimal vocal warming up in the form of singing a round together, and they start each song in the right key through the use of a tuning fork. They receive visual tempo cues from a conductor. The experiment lasted about an hour in total. After the experiment, the musicians were again asked to fill in a survey about their experience. Furthermore, they had no knowledge of what the visualization did (or of its effects) beforehand.

Round 1 - Song	**Specifics**
Woman of Ryde	Round, sung as a vocal warming up
Lean On Me (I)	Simple song, homophonic for most of the song
Human	Medium difficulty, lead switches between voice groups, the rest of the voices have a homophonic accompaniment
Wonder Woman (I)	Medium difficulty, polyphonic, the lead switches a lot
--- Break ---	--- Break ---

Round 2 - Song	Specifics
Nara	Medium difficulty, slow song, polyphonic, the lead switches a lot
Wonder Woman (II)	See Wonder Woman (I)
Lean On Me (II)	See Lean on Me (I)
--- Break ---	--- Break ---

Table 9: Experiment protocol and song specifics, iteration 3

Hardware setup

The experiment uses the exact same Hardware setup as in Iteration 2, only reduced to fit four singers instead of five (one less sensor, one less LED, and one less LED resistor). See Figure 17 and Figure 18 for the schematics.

To find a new, better, HRV obtainment method a few webpages[16,17] were consulted and compared with the literature discussed earlier in this project. Working with time-domain analysis methods is desired, to not increase complexity of the method too much, and allow for easy real-time analysis. The previous method used is also a time-domain method. Other possible methods include frequency-domain analysis and Poincare plots, however, they will be not be discussed further for the sake of simplicity. Farnsworth[17] discusses three accessible time-domain methods: RMSSD (Root Mean Square of Successive Differences), SDNN (Successive Differences of N-N intervals), and SDANN (Successive Differences of the Average of N-N intervals). RMSSD , SDNN and SDANN were tried out, and SDANN was found to be the best method to work with. These methods were also mentioned in the literature found in the background research. The method was implemented as follows: of every ten heartbeats, the average is taken. This average is stored in a data array, where it replaces the oldest value present there. The HRV is calculated as the standard deviation over the past 40 averages in this way, leading to a sliding window of data of the past five minutes. This is an approximation, as measuring 400 heartbeats at a normal heartrate of 80 bpm takes five minutes. The HRV is calculated every time a new average is found (every 10 heartbeats) See also Figure 22. The equations can be written down as follows:

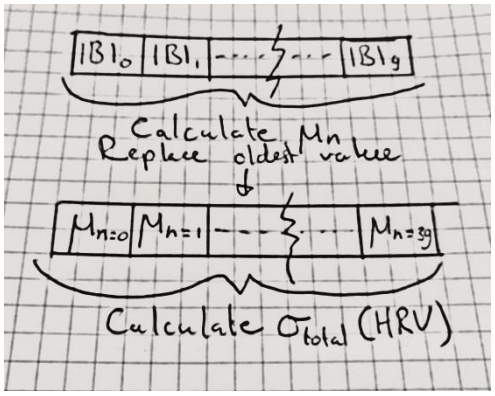


Figure 22: Clarification of the running average HRV obtainment method

$$\mu = \frac{\sum_{n=0}^{n=9} IBI_n}{10}$$

$$HRV = \sigma_{total} = \sqrt{\frac{\sum_{n=0}^{n=39}(\mu_n - \mu_{total})^2}{40}}$$

Here, the IBI is the inter-beat interval in milliseconds, μ is the average over 10 IBIs and the σ is the standard deviation of the past 40 averages; the HRV.

Subsequently, the visualization is controlled by the same mechanic as in the previous iteration. The manual override is not used, instead a threshold value is directly used after it has been determined through observing the results of the first round.

[16] https://www.hrv4training.com/blog/heart-rate-variability-a-primer

[17] https://imotions.com/blog/heart-rate-variability/

Results and discussion iteration 3

The resulting data of this experiment proved to be more reliable than that of iteration 1. The data contained no further inexplicable jumps, as can be seen in Figure 23. The first five minutes of data of each experiment can be disregarded, as the SDANN HRV obtainment method needs at least this much data to be accurate.

Figure 23: Total graph of results of round 2, iteration 3. Measurements with visualization on.

A good example of how the singers synchronize physiologically can be seen in Figure 24 (*SONG 1* in Figure 23). This song started heavily out of tune, but throughout the singing, the singers found each other and synchronized their sound again. This shows in the HRV of the singers, differing much at the start and the middle of the song, and becoming closer to each other near the end of the song. The average standard deviation between them also drops significantly near the end.

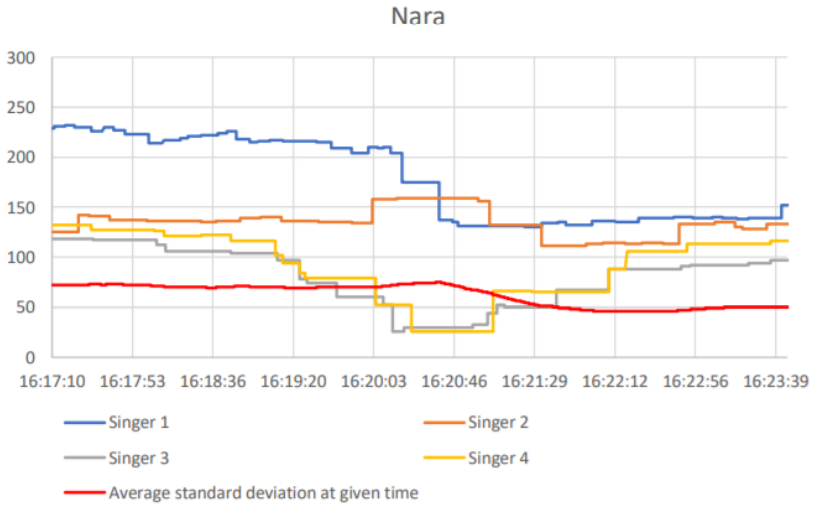


Figure 24: Results for Nara, the first song of round 2, iteration 3

The threshold for the visualization, as chosen after round 1, was a too low one. The chosen value was 37 ms, which the group bareley reached in the second round. This resulted in the visualization turning on just a few times, and then only briefly.

By comparing the difference between using a visualization and not using one, the effectiveness of using one can be determined. The graphs in Figure 25 show the same song (right one is *SONG 2* in Figure 23), sung both with and without using a visualization. It is hard to determine a clear difference between the two, which can be explained by a combination of things. It indicates that the visualization does not have an effect on the overall feeling of synchronization, which is confirmed by the singers' rating of the impact of the visualization (1.75/5). This might be caused by the fact that the visualization only turned on for a brief time, but also by the fact that the singers were not focussed on the visualization. They had no prior knowledge of how the visualization works, and concentrated mainly on their singing, each other, and occasionally, sheet music.

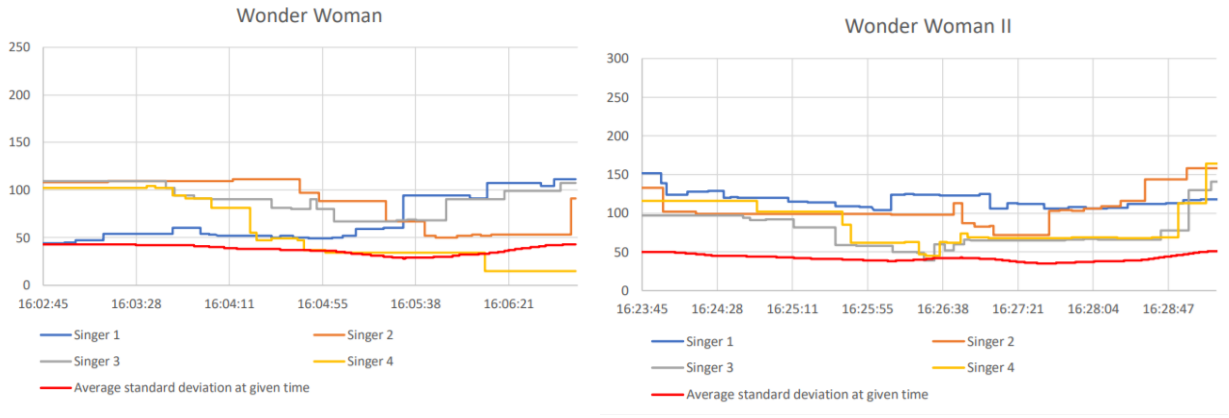


Figure 25: Results of a song from both rounds, left graph without visualization, right graph with visualization

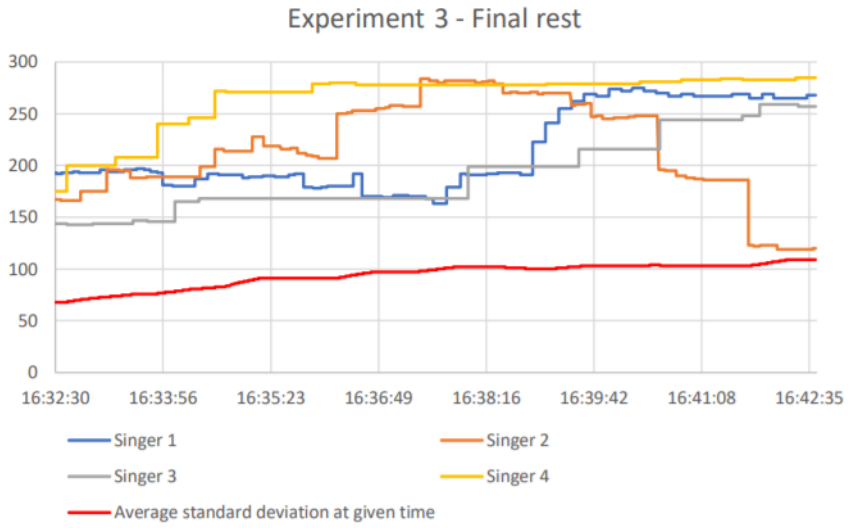


Figure 26: Results from the period of rest at the end the experiment with iteration 3

To see the difference between singing together and not singing together, the graph in Figure 26 (also corresponding with *REST* in Figure 23) can be observed. During this period, the musicians had no tasks to perform, but casually held a conversation between them. Clearly, the musicians' HRV rises because they are under a smaller amount of stress. On the other hand, their HRVs seem to be much further apart, which reflects in the running average, which only rises during the period of rest. This leads to the conclusion that while singing, the musicians experience more stress than while resting, but they synchronize while singing nonetheless, something they do not during rest. The difference can also be seen clearly in Table 10. The averaged values here were calculated similar to how they

were calculated in Iteration 1, taking the average of all of the singers' HRV values in a specific experiment phase, and taking the average of the running average standard deviation during this phase to find the values in the third column.

The measurement methods were, again, experienced as being unobtrusive.

Experiment phase	Average HRV of singers combined	Average standard deviation between singers	Average rating by singers / enjoyment
Woman of Ryde	89.80	48.15	n/a
Lean on Me (I)	87.70	63.28	3.50
Human	82.47	42.00	1.75
Wonder Woman (I)	75.64	37.54	4.00
---Break---	n/a	n/a	3.25
Average Round 1	**82.24**	**47.65**	**3.50**
Nara	120.83	62.25	4.00
Wonder Woman (II)	95.83	41.94	4.25
Lean on Me (II)	157.91	62.81	2.75
---Break---	222.06	94.11	n/a
Average Round 2	**160.54**	**70.65**	**4.00**

Table 10: Average HRV per experiment phase, and standard deviation between them compared to singer rating, iteration 3

Conclusion

To conclude, the musicians appeared to synchronize their HRVs while singing together. The running average standard deviation between them is a variable that can be used to trigger a visualization, although in this experiment the value was chosen too low (37 ms). It was hard to notice a difference between using a visualization and not using one, leading to the conclusion that the visualization was not a distraction, but had no effect on the feeling of synchronization either.

Lastly, in this case, using wires instead of wireless communication to get the data to a central point proved not to be an issue, but it is important to note that this may be an issue if the musicians have to move more, for example with guitar or piano players.

These conclusions will be further discussed in the next chapters of this report.

Evaluation and Discussion

There are a few aspects about this project that have to be evaluated. This includes a more elaborate discussion of the realization phase, but also a reflection on whether the system meets the requirements specified in the specification phase. Moreover, this chapter includes an observation of the possible societal impact of this research field.

First of all, the singers had no prior knowledge of what the visualization did. This was initially done to make sure it did not influence any measurements, but in the end it appeared that the visualization lost effectiveness due to its mechanics of the visualization being unclear. A better solution would have been to give the singers a few hints on how the system would operate, so that they could use the feedback to their advantage without having too much knowledge raise stress or influence them in any other way. As of now, the visualization was not a distraction, but the opposite: its effects went unnoticed by the musicians. Apart from the above, this might have been caused by the visualization's binary operation; an on/off system might not give a good representation of a process which occurs, in fact, in gradations. This could be solved by using a linear map of the group average instead of one threshold for this value.

Another point that has to be discussed includes the effect of potential outliers. With such a small group, one outlier has a large effect on the group average data. The group might appear to be out of sync, while in fact only one singer has a different HRV, with the rest still synchronized. To work around this effect, measurements could have been taken with a much larger group.

With the current system, musicians synchronize their HRV while singing together. In the case of singers, breathing often occurs at synchronized times due to the room to breathe within the music happening at the same time. Since the breathing and HRV are linked through the RSA (respiratory sinus arrhythmia), this might increase the synchronization effect of the HRV by a large amount. Possibly, the same happens if the measurements are done with brass- or wind players. To work around this, the experiment could be done with, for example, guitar or piano players. The downside of this is that other types of musicians usually are more physically active while playing, making it more difficult to obtain accurate measurements.

Functional requirements	Met	Non-functional requirements	Met
HRV measurements	X	Unobtrusive sensing	X
Visualization showing only group data	X	Non-distracting visualization	X
Visualization using colour		Users' consent of showing their biodata	X
Visualization using light	X	Minimized stress levels	X
Wi-Fi data link		Well-known parts of music	X
		Good interpersonal relationships	X
		Similar parts of music	

Table 11: List of the functional and non-functional requirements of the system, as well as an indication if they were met

Meeting the requirements

In Table 11 the functional and non-functional requirements of the system are recalled, as stated in the Specification phase. For each of these requirements, it is shown whether it has been met or not. As can be seen, there are only three requirements that have not been met. In the end, the used visualization did not include changes in colour hue or colour saturation. Also, no wireless data link was used between the individual musicians and the visualization, since this proved unnecessary in the setup as tested. Lastly, the musicians did not always sing similar parts of music, which may have impacted the amount of measured synchronization.

In contrast, the requirements that were met include all must haves from the MoSCoW analysis, meaning that the most important requirements have been met, leading to a functional system.

Future impact

This concludes the discussion on the system concerning its implementation now. When looking at the technology in a broader context, systems like these might influence a significant part of human life in the future. This section discusses the possible future impact of technology in this field.

Needless to say, physiological measurement is a field of science with much potential. By keeping track of one's physiology, it is possible to state something about that person's health, stress levels and even emotions. Tracking heart rate can be used to monitor sports' players performance and limits, leading to even more personalised training schedules and less performance injuries. This is already being done, for example as explained in the Guardian[18]. Moreover, elderly people that live by themselves can be monitored, so that when they fall or gain a serious injury in another way, help can be sent to them immediately. It is only natural to extend such a useful form of support to other fields as well, such as in this case the performing arts. It is highly possible that this technology will be widely accessible in the near future, especially when products like the Apple Watch[19] keep developing more. The current version of this product includes accurate ECG (electrocardiography) heart rate measurements, which in the past could only be obtained by visiting a hospital.

So, what fields are next? The measuring of physiology in music can be a gateway to using it in other performing arts, such as theatre or dance. Measuring the synchronization between yourself and other humans and identifying their emotions can add a whole new dimension to, for example the simple act of having a conversation with someone. Will it be possible to identify if someone is lying while talking to them? Will everyone's biodata be available whenever someone else requests it? As an employer, monitoring the physiology of your employees might lead to an increase in efficiency, since monitoring stress levels and emotions can give you an image of how to best engage your employees without overstressing them. On one hand, this seems like it could improve the way humans interact by a tremendous amount, but on the other hand, it means disregarding all forms of privacy. Looking at it from a societal point of view, it probably would not be accepted in the aforementioned form by current society standards and values of privacy. If this trend continues, however, showing a piece of personal biodata could become more and more normal. Look at the Apple Watch: it allows people to share their physiological data with their friends, to stimulate them to exercise in a competitive way. If this is already widely used, it makes one wonder if privacy values are maybe less applicable in this context. It is however, still highly important that users of technologies like this maintain their autonomy: they should be able to choose whether or not they want to share their physiological data. With the Apple Watch, this is the case with regard of sharing data with friends: the user chooses what their friends are allowed to see.

As this technology develops, it might become more and more accessible and applicable in more and more situations, as can already be seen in some products. Although this contributes a few benefits to society, its growth has to be carefully regulated to avoid the pitfalls of reduced privacy and loss of autonomy. Furthermore, if it takes on the form of general human enhancement, it should be accessible to all, to avoid discrimination based on economic status or nationality.

[18] https://www.theguardian.com/football/2015/aug/02/science-fine-tuning-elite-footballers
[19] https://venturebeat.com/2018/09/12/apple-watch-series-4-can-detect-falls-take-ecgs-and-lead-you-through-breathing-exercises/

Conclusion

The findings from this research can be combined into a conclusion, in an attempt to answer the research question: *how to design a system that enhances the physiological synchronization of musicians playing together through the use of a visualization?* [RQ]. With the necessary requirements for the system met, a few conclusions can be formed on the topics discussed in this report to answer the subquestions that were formed in the beginning. At the end of this chapter, the research question will be answered.

The subquestions were as follows:

> *How do musicians synchronize physiologically while making music together? [Q1]*
> *What defines physiological synchronization between human beings? [Q2]*
> *How can this synchronization be measured and quantified? [Q3]*
> *What are effective ways of visualizing synchronization? [Q4]*
> *How can intrusiveness of the visualization be minimized? [Q5]*
> *What is the impact of visualizing this on the musicians' synchronization? [Q6]*

From the literature research and the ideation, it showed that heart rate variability (HRV) and respiration rate are important variables for measuring synchronization. Furthermore, brain waves play a role in the synchronization of musicians as well. Galvanic skin response and heart rate are also variables that can be used, although they are harder to measure accurately if physical activity is involved. External factors that positively influence synchronization are good interpersonal relationships and minimized stress factors. Sensor methods should be unobtrusive. This answers [Q2] and [Q3].

Regarding visualization, the research and ideation show that the visualization should be non-distracting. An effective way of visualization is the display of a changing saturation of a single colour [Q4]. Furthermore, musicians should consent to the use of their personal data and how it is visualized. This, together with the non-distraction, ensures that the musicians are not put under a significant amount of stress [Q5].

Using the above findings, a system was developed. For this system three prototypes were iterated and tested with a series of experiments. These complete the answer to the research question.

The results from the experiments show that physiological synchronization occurs in singers' heart rate variability when singing together, making it a variable that can support a visualization on its own [Q1]. It is, in this case, not necessary to measure other physiological signals that could indicate a level of synchronization, such as breathing, heart rate or galvanic skin response. The variable used in this research, the running average of the standard deviation in HRV between singers can be used to base a visualization on, when physiological synchronization is what the system wants to feedback.

The visualization that was used in these experiments is experienced as an unobtrusive one. However, it was too unobtrusive, resulting in the singers not noticing what the visualization did and not making a clear difference in the amount of synchronization, making question [Q5] obsolete. The visualization did not have a measurable impact [Q6]. The chosen threshold value for the visualization, 37 milliseconds, was too low. The sensor methods were unobtrusive, but could have been made even less intrusive. This is necessary when measuring with musicians that have more physical activity when playing. Possible solutions include carrying the system in a backpack or sewing it into the musicians' clothing.

In conclusion, the design of a system that enhances the physiological synchronization of musicians making music together through the use of a visualization has been investigated. Such a system measures the HRV of musicians in a non-intrusive way and compares them to find the amount of synchronization between the group. This value is then used in a visualization to give feedback to the musicians. This visualization should not be a distraction and not put the musicians under any amount of stress, however, it should have a presence in the room and be clear in its feedback. Although this answers most of the research question [RQ], other types of visualizations, group compositions and measurement methods have to be investigated before a complete answer can be given. See also the Recommendations for Future Work.

Recommendations for Future Work

To conclude this report, there are a few recommendations that could be taken into account if this line of research is to be continued in the future. These recommendations follow from the conclusions and evaluation of the project, and some of them are necessary for drawing a final conclusion about the research topic.

Firstly, other kinds of musicians should participate in the experiments. Right now, the musicians that participated were all amateur musicians with musical experience. It would be interesting to see whether there is a difference between amateur and professional musicians, or between musicians with experience and musicians with no experience playing together. The goal of this is to see whether people that have played together for a multitude of times synchronize much more easily than people that have not. Furthermore, the experiment should be conducted with musicians that do not have synchronized breathing cues while playing, such as for example piano or guitar players, to find out if the synchronization in HRV takes place only if the musicians' breathing is timed together. Also, the group size should be increased to see the effect of outliers, in comparison to the large effect that one outlier can have in a small group.

Secondly, measurement variables that have not been explored yet include GSR, heart rate and brain waves. Although the others were ruled out earlier, brain waves have been shown to play an important role in the synchronization process of musicians. Due to limited resources, they were not investigated in this project, but cortical phase synchronization might prove useful to measure the amount of synchronization and perhaps base a visualization on.

Thirdly, only group data is being shown in the current design. A possibility would be to feedback individual data instead to see the actual effect of privacy intrusion. If this impact is not too large, individual feedback can be really useful. This holds for example in orchestras, where feedbacking which musicians do not synchronize with the group could help in improving the overall music quality. Of course this only holds as long as the musicians agree to this way of using their biodata.

Lastly, there are a few recommendations for designing a visualization that shows physiological synchronization between people. As the current system did not have a significant impact, other types of visualizations have to be tried out to find the possible effects that a visualization could have. Possible options include a change in colour, colour saturation, light intensity and/or movement speed. If the average synchronization in HRV is mapped to a linear scale, this can be used for a smooth transition scale for any of these variables instead of using a binary on/off threshold. Moreover, data from the music that is being played could be incorporated into the design to enhance the feeling of its integration within the context. Examples of this include showing the key, tempo, or mood of the music in the visualization as well. This could also assist in bringing the visualization more to the foreground.

Appendix A – References

Acharya, U. R., Joseph, K. P., Kannathal, N., Min, L. C., & Suri, J. S. (2007). Heart rate variability. In *Advances in cardiac signal processing* (pp. 121-165): Springer.

Barnett, S. B., & Cerf, M. (2017). A Ticket for Your Thoughts: Method for Predicting Content Recall and Sales Using Neural Similarity of Moviegoers. *Journal of Consumer Research, 44*(1), 160-181. doi:10.1093/jcr/ucw083

Bradley, M. M., & Lang, P. J. (2000). Measuring emotion: Behavior, feeling, and physiology. *Cognitive Neuroscience of Emotion, 25*, 49-59.

Carrol, J. M. (1999). *Five reasons for scenario-based design.* Paper presented at the Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on.

Coeckelbergh, M. (2013). Anthropology of Vulnerability. In *Human Being at Risk* (pp. 37-62): Springer.

d'Andrade, R., & Egan, M. (1974). The colors of emotion. *American ethnologist, 1*(1), 49-63. doi:10.1525/ae.1974.1.1.02a00030

Healey, J. A. (2000). *Wearable and automotive systems for affect recognition from physiology.* Massachusetts Institute of Technology, Retrieved from https://dspace.mit.edu/handle/1721.1/9067#files-area

Hennig, H. (2014). Synchronization in human musical rhythms and mutually interacting complex systems. *Proceedings of the National Academy of Sciences, 111*(36), 12974-12979. doi:10.1073/pnas.1324142111

Hirsch, J. A., & Bishop, B. (1981). Respiratory sinus arrhythmia in humans: how breathing pattern modulates heart rate. *American Journal of Physiology-Heart and Circulatory Physiology, 241*(4), H620-H629. doi:10.1152/ajpheart.1981.241.4.H620

Iwanaga, M., Kobayashi, A., & Kawasaki, C. (2005). Heart rate variability with repetitive exposure to music. *Biological psychology, 70*(1), 61-66. doi:10.1016/j.biopsycho.2004.11.015

Leskowitz, E. (2008). The influence of group heart rhythm on target subject physiology: Case report of a laboratory demonstration, and suggestions for further research. *Subtle Energies & Energy Medicine Journal Archives, 18*(3).

Lindenberger, U., Li, S.-C., Gruber, W., & Müller, V. (2009). Brains swinging in concert: cortical phase synchronization while playing guitar. *BMC Neuroscience, 10*(1), 22. doi:10.1186/1471-2202-10-22

Luck, G. a. N., Sol. (2008). An investigation of conductors' temporal gestures and conductor—musician synchronization, and a first experiment. *Psychology of Music, 36*(1), 81-99. doi:10.1177/0305735607080832

Lynch, M. P. (2016). Privacy and Autonomy. In *The internet of us: Knowing more and understanding less in the age of big data* (pp. 95-115): WW Norton & Company.

Mader, A., & Eggink, W. (2014). *A design process for creative technology.* Paper presented at the DS 78: Proceedings of the 16th International conference on Engineering and Product Design Education (E&PDE14), Design Education and Human Technology Relations, University of Twente, The Netherlands, 04-05.09. 2014.

Mauss, I. B., Levenson, R. W., McCarter, L., Wilhelm, F. H., & Gross, J. J. (2005). The tie that binds. Coherence among emotion experience, behavior, and physiology. *Emotion, 5*(2), 175. doi:10.1037/1528-3542.5.2.175

Mauss, I. B., & Robinson, M. D. (2009). Measures of emotion: A review. *Cognition and emotion, 23*(2), 209-237. doi:10.1080/02699930802204677

McCraty, R. (2017). New frontiers in heart rate variability and social coherence research: techniques, technologies, and implications for improving group dynamics and outcomes. *Frontiers in public health, 5*, 267. doi:10.3389/fpubh.2017.00267

Mikellides, B. (1990). Color and physiological arousal. *Journal of Architectural and Planning Research*, 13-20.

Müller, V., & Lindenberger, U. (2011). Cardiac and respiratory patterns synchronize between persons during choir singing. *PloS one, 6*(9). doi:10.1371/journal.pone.0024893

Naz, K., & Epps, H. (2004). Relationship between color and emotion: A study of college students. *College Student J, 38*(3), 396.

Ouwerkerk, M. (2010). Unobtrusive emotions sensing in daily life. In *Sensing Emotions* (Vol. 12, pp. 21-39). Dordrecht: Springer.

Rizzolatti, G., & Sinigaglia, C. (2010). The functional role of the parieto-frontal mirror circuit: interpretations and misinterpretations. *Nature Reviews Neuroscience, 11*, 264. doi:10.1038/nrn2805

Rössler, B. (2018). *The value of privacy*: John Wiley & Sons.

Russell, J. A. (1989). Measures of emotion. In *The measurement of emotions* (pp. 83-111): Elsevier.

Schubert, E. (1999). Measuring Emotion Continuously: Validity and Reliability of the Two-Dimensional Emotion-Space. *Australian Journal of Psychology, 51*(3), 154-165. doi:10.1080/00049539908255353

Taelman, J., Vandeput, S., Spaepen, A., & Van Huffel, S. (2009). *Influence of mental stress on heart rate and heart rate variability.* Paper presented at the 4th European conference of the international federation for medical and biological engineering.

Timofejeva, I., McCraty, R., Atkinson, M., Joffe, R., Vainoras, A., Alabdulgader, A. A., & Ragulskis, M. (2017). Identification of a group's physiological synchronization with earth's magnetic field. *International journal of environmental research and public health, 14*(9), 998. doi:10.3390/ijerph14090998

Van Dyck, E., Six, J., Soyer, E., Denys, M., Bardijn, I., & Leman, M. (2017). Adopting a music-to-heart rate alignment strategy to measure the impact of music and its tempo on human heart rate. *Musicae Scientiae, 21*(4), 390-404. doi:10.1177/1029864917700706

Vickhoff, B., Malmgren, H., Åström, R., Nyberg, G., Engvall, M., Snygg, J., . . . Jörnsten, R. (2013). Music structure determines heart rate variability of singers. *Frontiers in psychology, 4*, 334. doi:10.3389/fpsyg.2013.00334

Yasuma, F., & Hayano, J.-i. (2004). Respiratory sinus arrhythmia: why does the heartbeat synchronize with respiratory rhythm? *Chest, 125*(2), 683-690. doi:10.1378/chest.125.2.683

Appendix B – Interview

This is a transcription of the telephone interview with Gregor Mayrhofer.

Gregor (G): Hallo?

Jochem (J): Good afternoon! This is Jochem Postmes speaking, I'm calling for the interview that Angelika Mader had contact with you about.

G: Ah, yes! Hi, thank you for your call!

G: Maybe you can tell me a little bit once again about your project? Angelika wrote me a little bit, but I'm not sure whether I remember all the things right, about what you are looking for in your research, so can you give me an update?

J: Yes, of course! So what I am trying to do is measure whether musicians that play together actually synchronize physiologically as well, with their heartbeat and breathing for example, and if there is a way to visualize this. And that by visualizing this, it is possible to enhance this synchronization they have even more. That is basically the core of the project: I want to see if I can get musicians more in flow with each other, and I'm going to test it on smaller groups of musicians, but for an orchestra or ensemble it probably works the same way. And that's where the idea arose to ask you a few questions. Angelika also sent me an interview with you in a magazine, I don't remember which one, where you also talked about synchronization between orchestra members?

G: Ah yeah, about breath I think. They asked me, among others, about the tempo and beat in conducting classical music, and they were a bit surprised that I said it was much more about breathing together than looking (although of course you have to look at the conductor). You probably know this yourself: when you play optically, with just the conductor conducting, it will never be as good as when the group breathes together; when they try to find their breath and their tempo together. That is for us conductors always interesting to find out: what can we do so that a group finds one tempo together, that they play together, but in a philosophical sense too: how you find the psychological energy that makes people want to go in the same direction and give the same effort in playing together.

J: Do you maybe have an idea how the trade-off is: how much it is listening and adapting to that influences this, or how much it is other things as well? Do you notice this as a conductor?

G: Yes, I'm constantly trying to find out what this percentage is, and in a more scientific way, finding out what the deciding factor is. There are a lot of people who are only talking about the metaphysical secrets - magical energy between people - and it's nice to think about this, but it does not help you if it is not working in reality. Thus, we have to find out why exactly it is working or not working. I think it is basically three things working together. One thing is the basic tempo feeling of a musician, their inner imagination that sees the notes and the words 'allegro molto' (for example), and can decide what their idea or interpretation of the music is. We conductors have this very strongly. If we have a very convincing and strict inner concept of how the music should sound, how quick it should be, how the balance is, all these factors, then a lot of the tempo issues will clarify themselves. Somehow we are transmitting this idea non-verbally or subconsciously to the orchestra. This first factor is strong in musicians themselves, in a group of people, and it is very important and strong in us, conductors, but in composers too: the fact that you have a really clear musical imagination of how it should sound. Then, going through it chronologically, the second aspect is the optical aspect: giving an upbeat. Of course, in the orchestra everyone has formed their idea, and then the conductor comes in and shows optically (without singing for them) what the tempo should

be. In the opera, if you dive in a performance without rehearsal, the only way is to show in the upbeat the tempo and character that you imagine as clear as you can. This is the optical part, where musicians try to see two beats and then adapt their inner imagination to this. And then, lastly, comes the most important part for playing together: the listening. In a way, that's probably where the breath comes in and is so important. When musicians breathe together, it is a double thing: on one hand with a deep breath in your body, you synchronize the upbeat with the conductor, a preparation of what you are going to play next. And then, really important (all the good orchestras do this fantastically): you listen from the very first millisecond how the first note starts or how the first two notes start, and then this decides the tempo. As a musician or even conductor you have to be very reactive in this first moment, between what you assume to be the tempo and what you actually hear. Then you immediately adapt. This is also a huge difference between professional orchestras or ensembles and amateur ensembles. The amateur musicians can read the conductor's tempo exactly the same as the professional musicians, but they can be so busy with technical questions, that as soon as they start playing, they forget to listen very carefully to their neighbours and forget to adapt to what they hear. With very good orchestras you can sometimes have quite a bad conductor giving an unclear tempo, and they will play together anyways.

J: Yes, because they listen to each other.

G: Yes. I think that the truth is somewhere between these three elements. We have the inner imagination, the optical aspect of showing a tempo, by conducting or by breath, and then listening and immediately adapting to what is around you. And maybe there's a fourth one, which is tempo memory, being closely related to the first one, the inner imagination. When you rehearse a piece in a quicker tempo, and then do the performance in a slower tempo or the other way around, the professional musicians will be surprised or even have technical problems, because they saved an absolute tempo somewhere in the body. Maybe they can remember this even when given an unclear upbeat, resulting in the right tempo because they have been rehearsing it like that. So that is maybe the fourth aspect of playing together and synchronizing.

J: Yes, I read a paper that actually proved the existence of both short-term and long-term tempo memory in musicians.

G: I think the first aspect, the inner imagination, is a very long-term memory of how quick a tempo is, or how a melody line should sound. This is a thing that comes after years of making music: they start a piece they have never seen before, but they read the first notes and that the tempo is allegro or andante and then they create (without even playing anything) their idea of how quick this piece would sound. These are the important aspects which can analyse the concept of playing together (or not playing together) in a scientific way, besides having this magical energy between people. I did a few interesting experiments about this, both with amateur and professional orchestras, which could be interesting for your research.

I practiced very often with musicians, saying: "close your eyes so that you cannot see each other, and then try to play together" to a group of 30-40 people. That was very interesting, because in the beginning it did not work, but after having it done a few times sometimes it worked much better when people could not see anything. When they just took a breath together, and then played one pizzicato, they played at the same time. Of course it did not always work, sometimes more or less, but sometimes they played perfectly together, which is very hard to achieve in a normal rehearsal. I thought a lot about this: you could say that this is only the metaphysical brain energy between people, or you could analyse this in a more exact way: why did it work so well?

J: Yes, because it might also be the listening, for example. That this is present here, also for amateur musicians.

G: The listening for a tempo feeling does not really work if you have only one pizzicato. But of course, it's the breath you listen to. If people breathe together, they hear how their neighbour breathes, even if it is really soft. There is probably something in our brain that help us give an expectation of exactly when something is going to happen. Compare it to catching a ball that is falling down: somehow our brain tells us exactly where to put our hand to catch it. There is something similar about the pizzicato: when we breathe together and estimate each other's breaths, it gives a sort of climax, an expectation of a touching point, of where the note has to be played. People open their mind to feel this, and get a strong inner expectation of where your neighbour will play the note, not by seeing anyone giving you a tempo, but just by hearing the breath. Maybe sometimes not hearing the breath itself, but somehow hearing the movement of the body or movement of the clothes, or other very subtle things. This we tried out often, and it worked much better this way. Conducting works often much better when you try not to be super precise with the tempo and give really quick and clear beats, but by making much more smooth and round movements and define a very precise expectation of where you expect the sound to happen. Also, when the sound is a little bit late, you immediately adapt to it. I think it is probably caused this triangle of aspects that we talked about earlier.

When I was studying in New York, at Juilliard, we had conducting classes from Gilbert, the chief conductor of New York philharmonic. He is now the chief conductor of the NDR Elphilharmonie. He taught us not to be only precise in our conducting and tell people exactly when to play, but to focus the expectation of musicians towards a certain final point, and then as soon as the orchestra plays, to adapt to this. Because in practice, especially with professional orchestras, musicians play later than you conduct. Often you have to be exactly in between these two timings. You create an expectation of a tempo, and then they play later than you conduct, but if you react to it too much, it looks to them like you want them to play it slower, and if you don't react at all it will completely fall apart. You have to be somewhere in between.

J: Exactly, so you have to listen really well and anticipate how it will work out. Do you think that these factors you named earlier, that make musicians synchronize and play together, can be enhanced, by maybe giving them some visual feedback, audible feedback, or something like that? That in some way they see how well they are synchronized and by noticing this that they start to synchronize even more.

G: Hard to say, you will probably know this much better when you finish your research. But I can imagine that it works almost oppositely. When we see something it is very hard to stay focused on hearing enough. I think that both seeing and hearing often distract each other. It is hard for musicians to decide to focus on what they hear, or on what they see.

J: Then what if you use an audible way of showing them their synchronization?

G: Do you mean something like a clicktrack (metronome)? I have used these sometimes when rehearsing. If played to the whole orchestra, they do not really work. The conductor can use a clicktrack, either an audible or optical way of giving them the upbeat, but if used by the whole orchestra often, it does not work like it should, probably because of what I explained earlier. I think it is all about creating this expectation of where the notes should be. For example, take a singer that is singing a slow melody over fast notes played by an orchestra. The orchestra has to synchronize with the singer, even though they cannot hear when she will sing her next note. By listening and

feeling the anticipation before the singer's notes, they are able to synchronize anyways. I think our ears are stronger than our eyes when it comes to anticipating rhythm in this way. By the way, it is not entirely clear to me what you are going to measure (and why), can you tell me a bit more about that?

J: Yes, of course! The two bodily signals that I want to focus on in my measurements are breathing and heart rate variability, meaning the variation in time between the individual heartbeats. These two signals are linked as well, our heart rate is influenced by breathing in or out. I found some interesting things about these signals, mainly the heart rate variability. This signal is influenced by electromagnetic signals, so by for example, the earth's magnetic field, but also by other people's heartbeats. This would indicate that people being closely together also synchronize their heart rate variability. I want to find out if performing a synchronized task, such as making music, enhances this effect. I read some interesting research where they measured the heart rate variability of and breathing of choir singers singing various things together. It showed that the singers actually had more synchronized heart rate variability when singing together, and this increased even more if their parts were similar. It also made me wonder about the influence of breath on this. Of course singers and brass or wind players have to breathe at the same time when they sing/play the same part, but do a guitar player and piano player start to breathe at the same time as well when they play together?

G: I think that would be really interesting to find out! I would definitely like to hear the result of your research, if you find out anything about this.

J: I will! It was really interesting for me to hear your opinion about these things! Seeing as I do not have any more questions, I want to thank you for your time!

G: No problem! It was nice talking to you.

J: Bye!

G: Bye!

Appendix C – Consent form

Date: _____

Consent form

Measuring physiological synchronization of musicians

Investigator:
Jochem Postmes (s1594761)
Bachelor student Creative Technology
j.m.postmes@student.utwente.nl
+31 6 15680548

Graduation project supervisor:
Dr. Angelika Mader
Assistant Professor Human Media Interaction
a.h.mader@utwente.nl

Purpose of the experiment

The aim of this experiment is to test the visualization of physiological signals that synchronize between musicians making music together. Among others, the functioning and effectiveness of this visualization will be tested.

Involved procedures

After the experiment, you will be asked to complete a survey concerning your experience of the experiment. Any questions that you do not want to answer can be left blank.

During the study, you will be asked to perform a few pieces of music together with other participants, either singing or playing an instrument. While performing, your heart rate and heart rate variability (HRV) will be measured through an optical pulse sensor connected to your finger (with a Velcro strap) or earlobe (with a clip). Your heart rate variability is then compared to that of the other participants, to determine the amount of physiological synchronization between the group. In the second phase of the experiment, this is visualized to you and the other participants via a screen or other means of simplified visualization.

It is possible to withdraw from this experiment at any time.

Potential harms and discomforts

It is possible to experience slight discomfort from the sensors measuring your physiology. Please let the investigator know if you experience this, or if you want to withdraw. The same holds if you experience any form of stress during the experiment.

Potential benefits

There are no direct benefits to participating in this experiment. It might provide you with an insight into the amount of synchronization between you and the other participants while you make music together.

Confidentiality

All data gathered in this experiment will remain anonymous. The survey results will be anonymously kept for analysis, without any possibility to link data to individual participants. The results from your heart rate measurements will be kept anonymously, without any possibility to link data to individual participants, to be used for further analysis. Only the participants that make music with you will know that you participated in the experiment.

Participation and withdrawal

Participating in this experiment is a voluntary decision. If you decide to participate in the experiment, it is possible to withdraw at any time. It is also possible to request that your data will not be used, within one week of the experiment. If you do this, all data of your experiment will be removed. If you decide to withdraw, any data recorded on your experiment will be removed unless you decide otherwise.

Research results

If you would like to be notified of the results of this study, please contact Jochem Postmes (j.m.postmes@student.utwente.nl).

Contact Information for Questions about the research

For more information about the experiment or the research topic, please contact Jochem Postmes (j.m.postmes@student.utwente.nl)

Contact Information for Questions about Your Rights as a Research Participant

If you have questions about your rights as a research participant, or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee of the department of EEMCS, mw. J.M. Strootman-Baas, mail: ethics-comm-ewi@utwente.nl, tel. 053-489 6719.

Consent

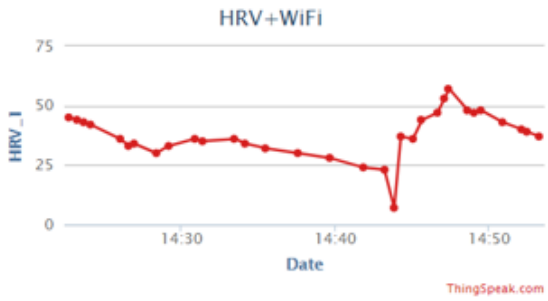
I have read the information in this consent form. I have had the opportunity to ask questions about my involvement in this experiment and inquire about additional details. I understand that if I agree to participate in this experiment, I may withdraw my participation at any time. I have been given a copy of this form. I agree to participate in this experiment with the other participants that have been introduced to me beforehand.

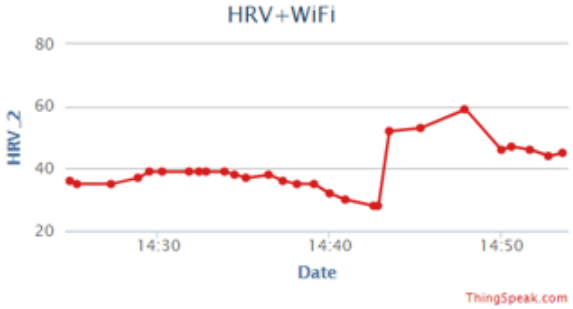
Name of participant: _____

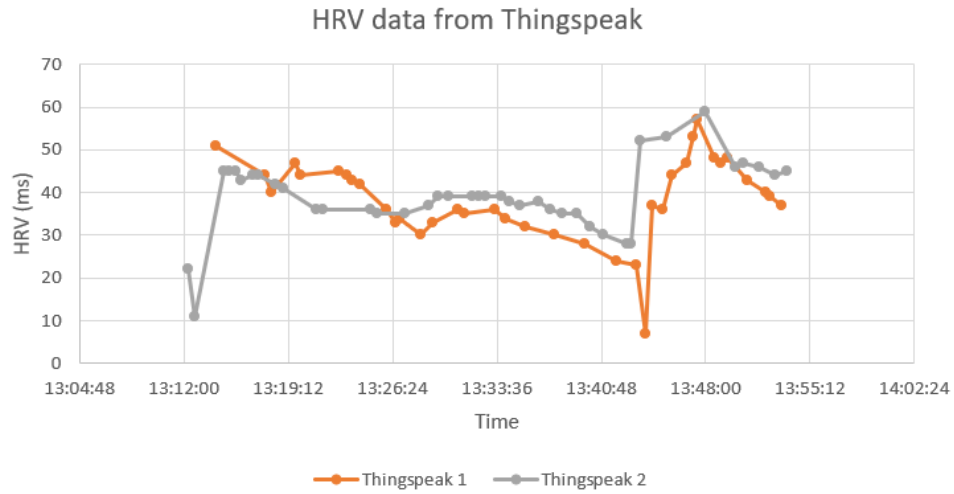
Participant signature: _____

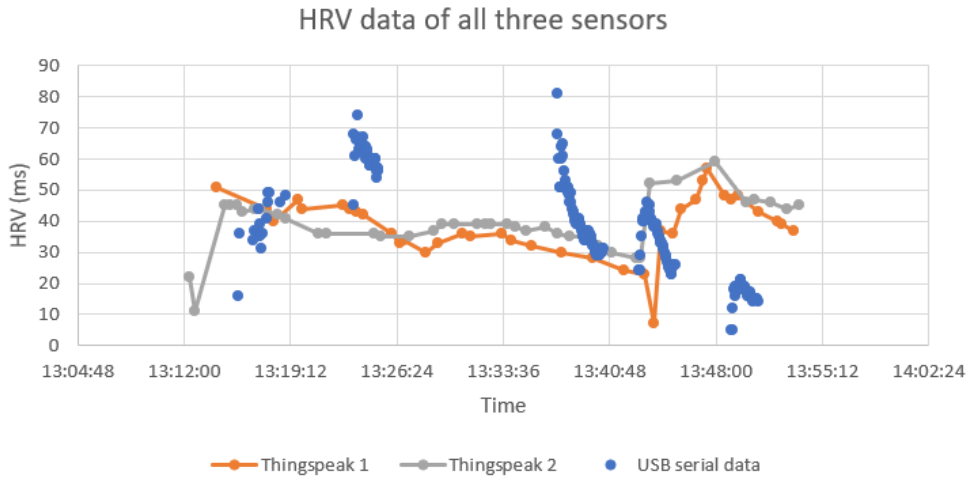
Appendix D – Full results iteration 1

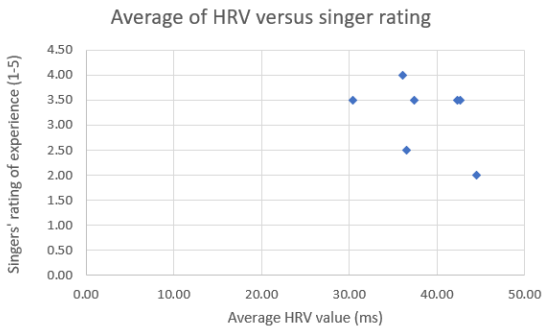
Graphs

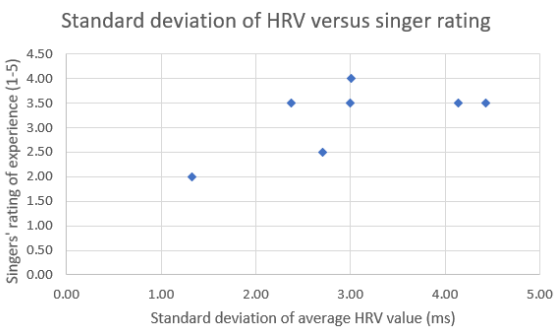












Survey

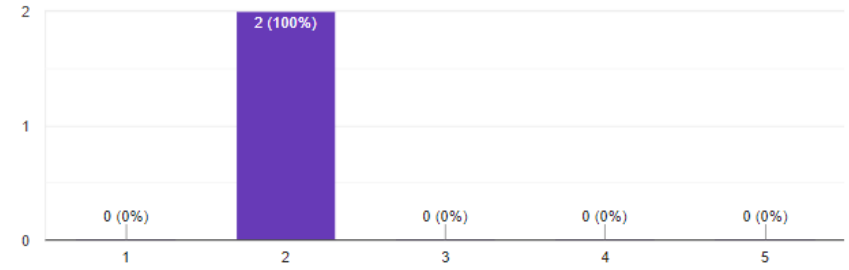
Did you experience any form of discomfort during the experiment? If so, please explain what kind of discomfort you experienced.

2 responses

No
My finger got a bit sweaty and itchy

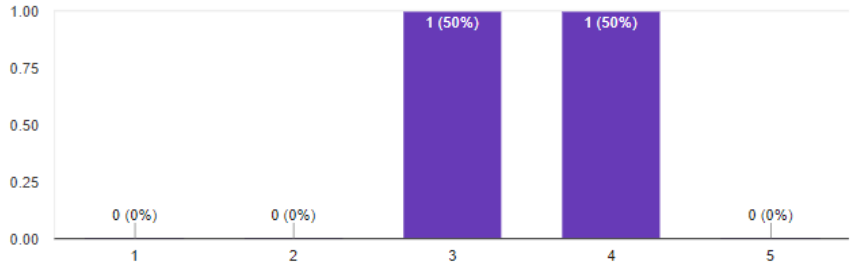
During the song 'Veerpont', how in sync did you feel with your fellow musicians?

2 responses



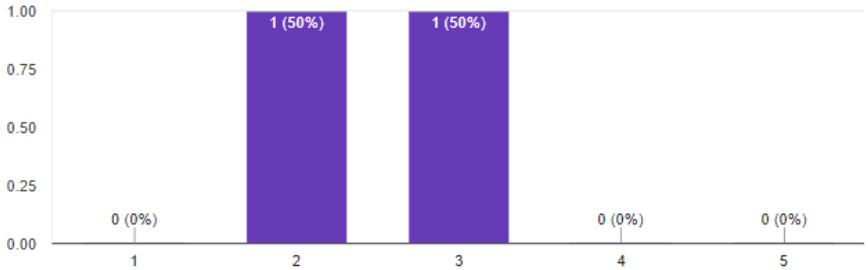
During the song 'De Gezusters Karamazov', how in sync did you feel with your fellow musicians?

2 responses



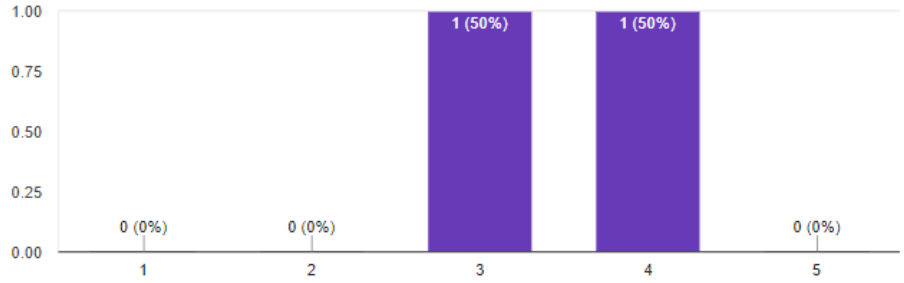
During the song 'De Mens Is Slecht', how in sync did you feel with your fellow musicians?

2 responses



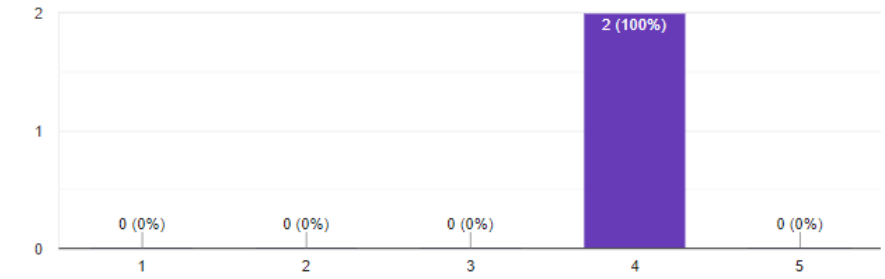
During the second time you sung the song 'Veerpont', how in sync did you feel with your fellow musicians?

2 responses



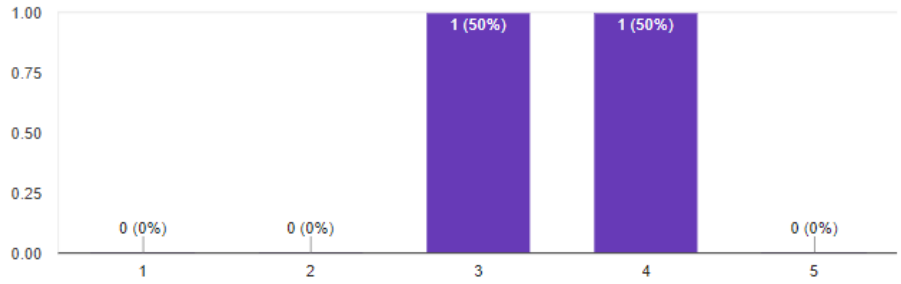
During the song 'The Woman of Ryde', how in sync did you feel with your fellow musicians?

2 responses



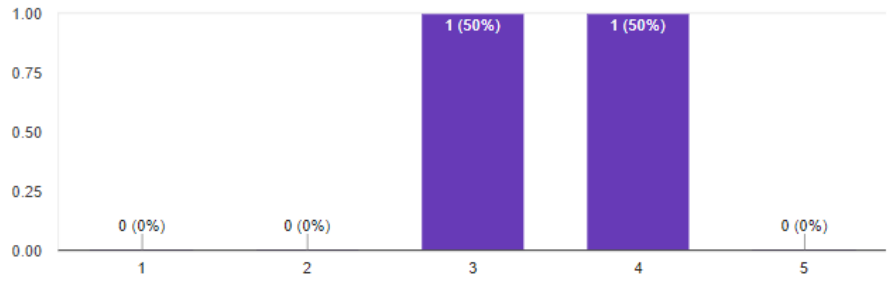
During the period of rest, how relaxed did you feel?

2 responses



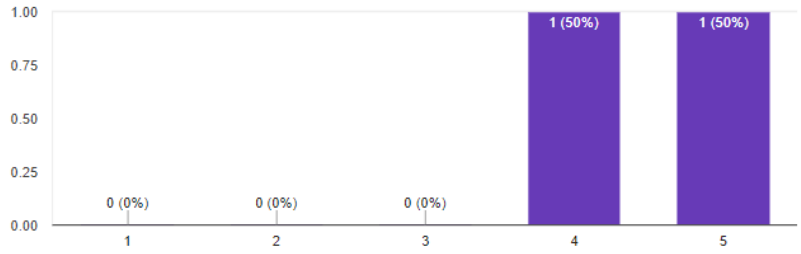
During the song 'Dodenrit', how in sync did you feel with your fellow musicians?

2 responses



How did you enjoy singing together with your fellow musicians in general?

2 responses



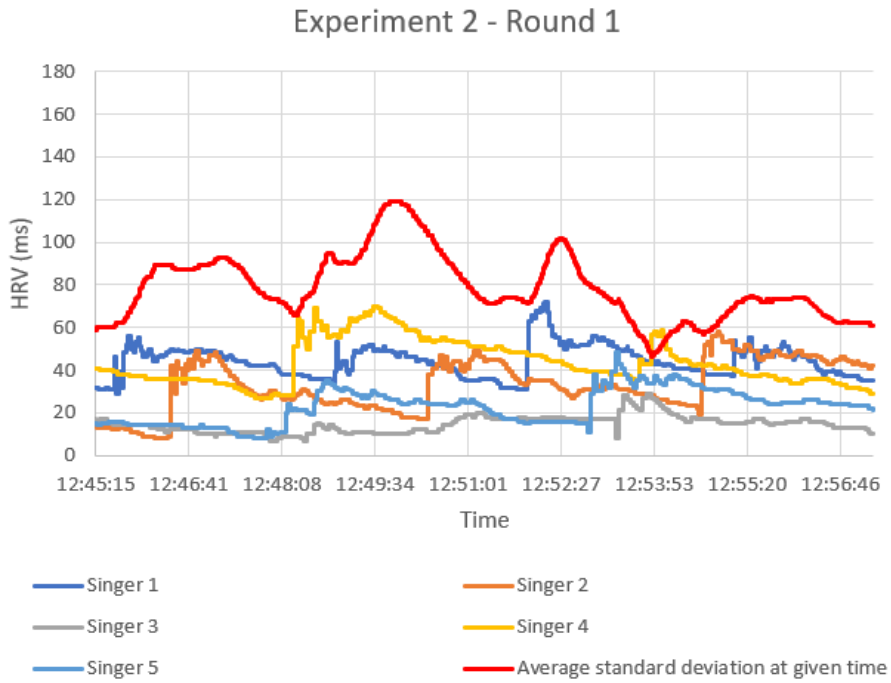
Do you have any other remarks and/or recommendations?

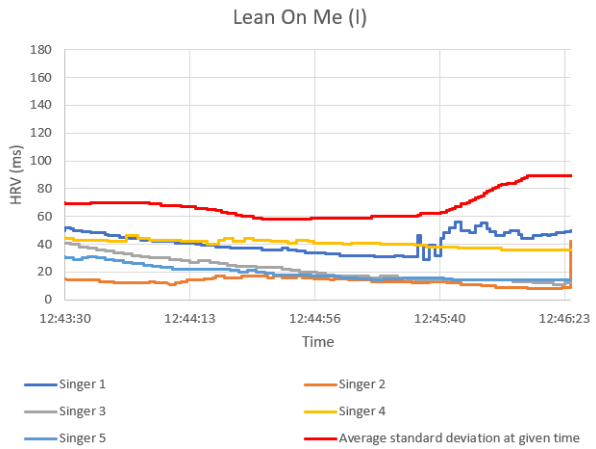
2 responses

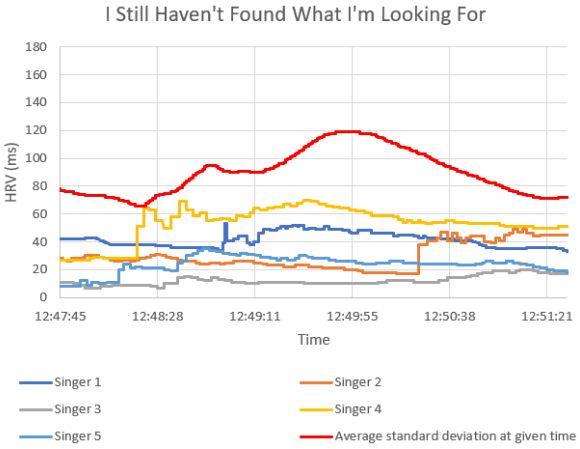
Good luck!
Not really

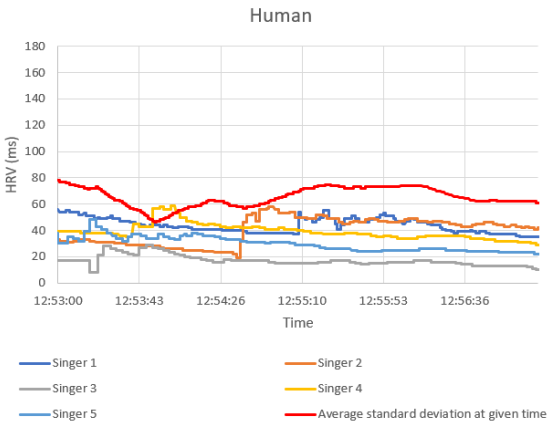
Appendix E – Full results iteration 2

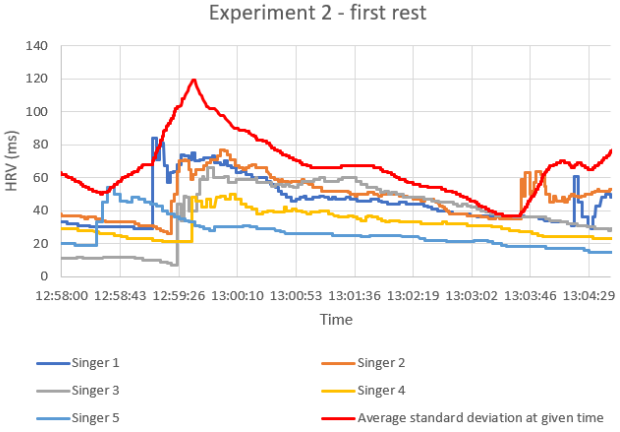
Graphs

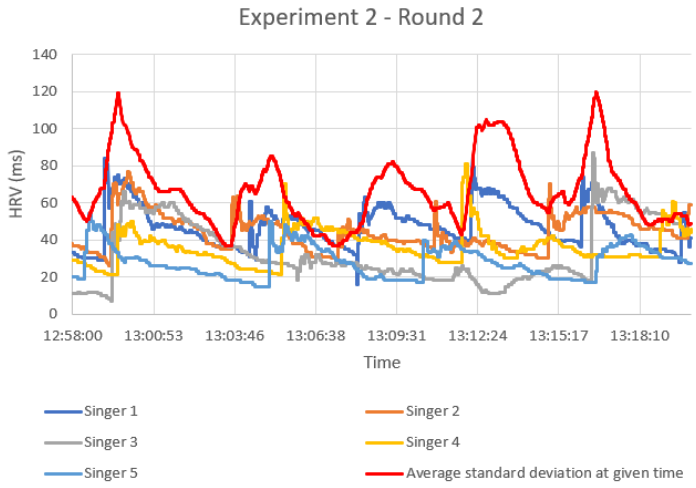


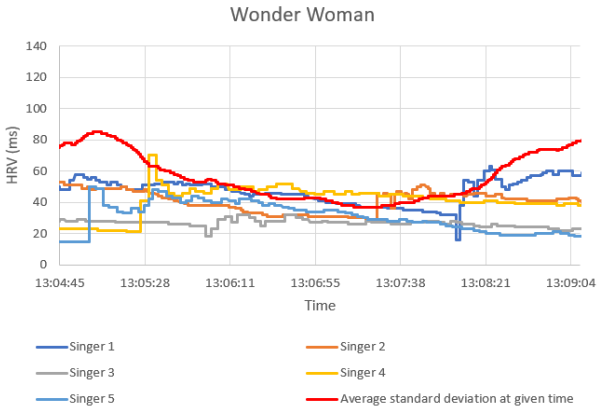


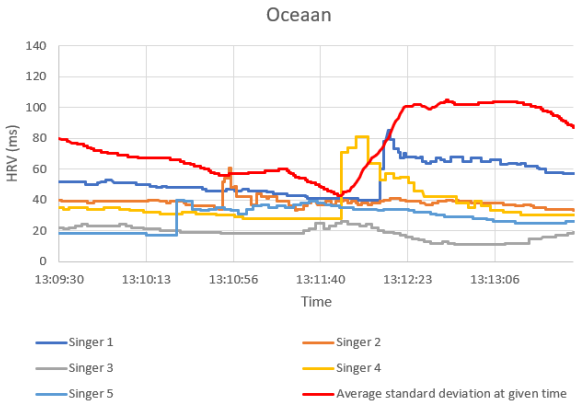


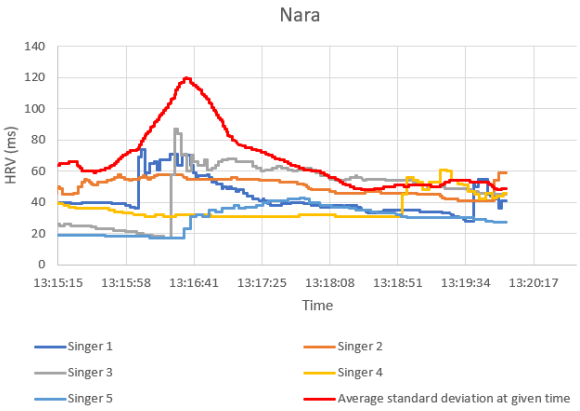


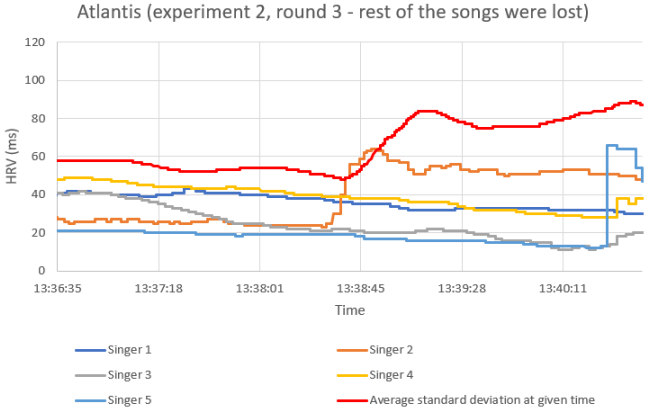



Experiment 2 - Round 2
HRV (ms)
140
120
100
80
60
40
20
0
12:58:00
13:00:53
13:03:46
13:06:38
13:09:31
13:12:24
13:15:17
13:18:10
Time
Singer 1
Singer 2
Singer 3
Singer 4
Singer 5
Average standard deviation at given time


Wonder Woman
HRV (ms)
140
120
100
80
60
40
20
0
13:04:45
13:05:28
13:06:11
13:06:55
13:07:38
13:08:21
13:09:04
Time
Singer 1
Singer 2
Singer 3
Singer 4
Singer 5
Average standard deviation at given time


Oceaan
HRV (ms)
140
120
100
80
60
40
20
0
13:09:30
13:10:13
13:10:56
13:11:40
13:12:23
13:13:06
Time
Singer 1
Singer 2
Singer 3
Singer 4
Singer 5
Average standard deviation at given time


Nara
HRV (ms)
140
120
100
80
60
40
20
0
13:15:15
13:15:58
13:16:41
13:17:25
13:18:08
13:18:51
13:19:34
13:20:17
Time
Singer 1
Singer 2
Singer 3
Singer 4
Singer 5
Average standard deviation at given time


Atlantis (experiment 2, round 3 - rest of the songs were lost)
HRV (ms)
120
100
80
60
40
20
0
13:36:35
13:37:18
13:38:01
13:38:45
13:39:28
13:40:11
Time
Singer 1
Singer 2
Singer 3
Singer 4
Singer 5
Average standard deviation at given time

Survey

Did you experience any form of discomfort during the experiment? If so, please explain what kind of discomfort you experienced.

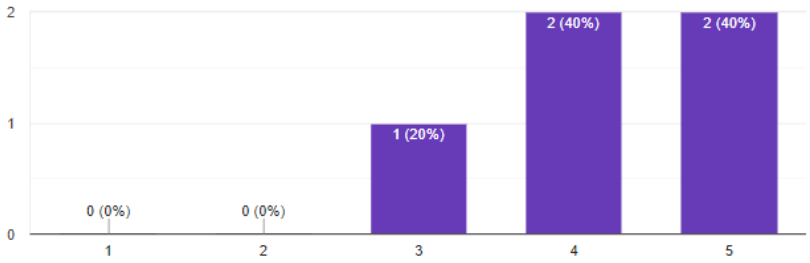
5 responses

No
None experienced
Ik heb de verkeerde arm uitgekozen voor de sensor
Being afraid that I would sing a wrong note, which is immediately audible in such a small group. Not completely being free with the hand, this was not too big of an issue, but it does keep your attention

Experience of singing, round 1

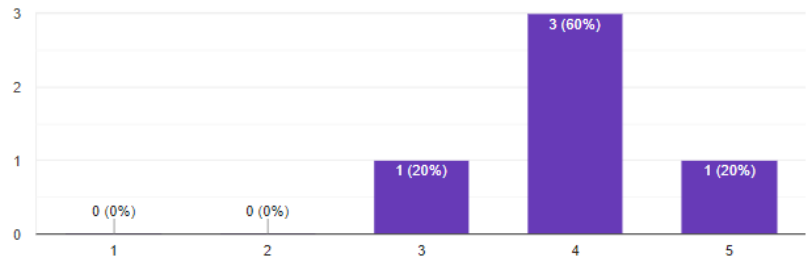
Lean On Me

5 responses



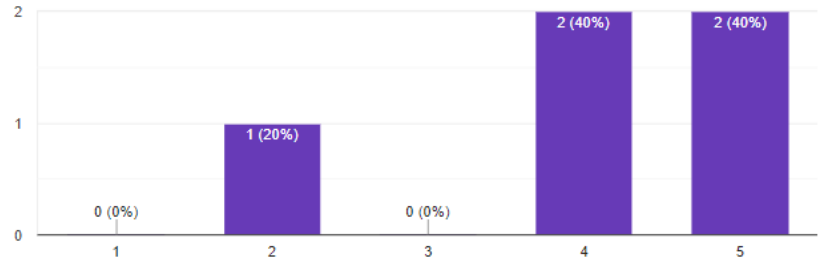
I Still Haven't Found What I'm Looking For

5 responses



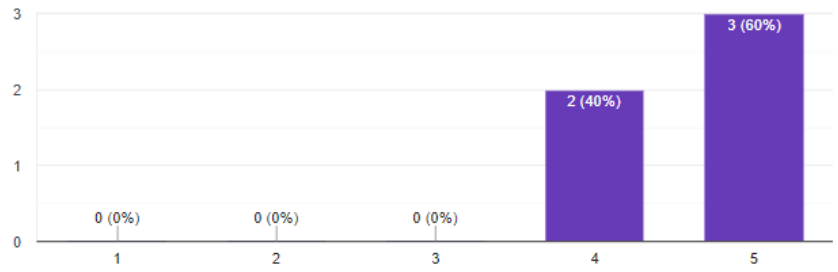
Human

5 responses



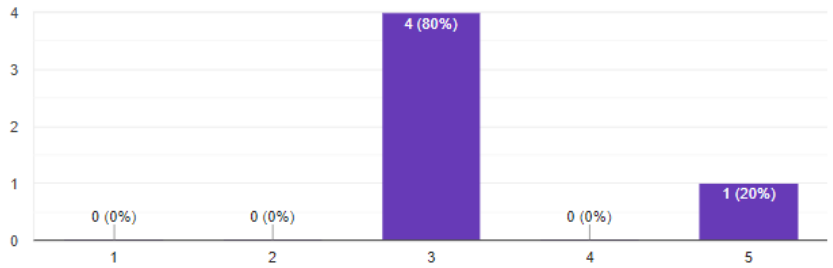
How did you enjoy singing together with your fellow musicians in round 1?

5 responses



During the period of rest between rounds, how relaxed did you feel?

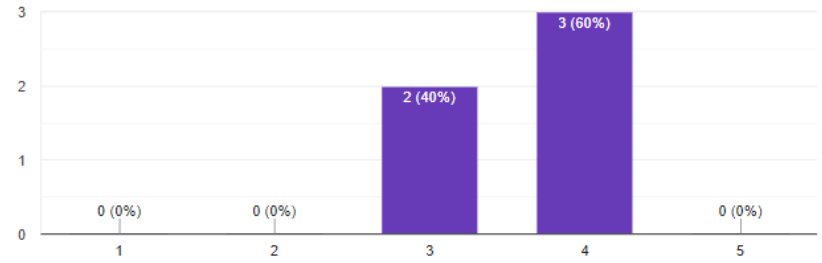
5 responses



Experience of singing, round 2

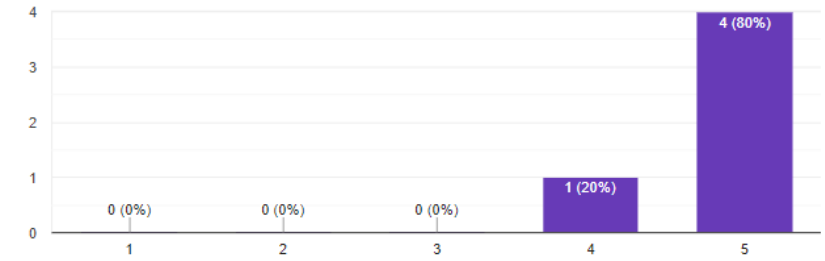
Wonder Woman

5 responses



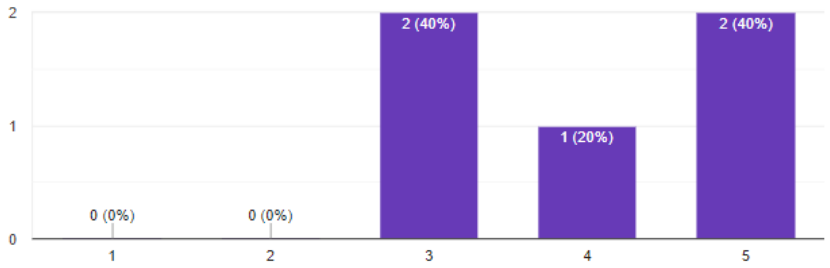
Oceaan

5 responses



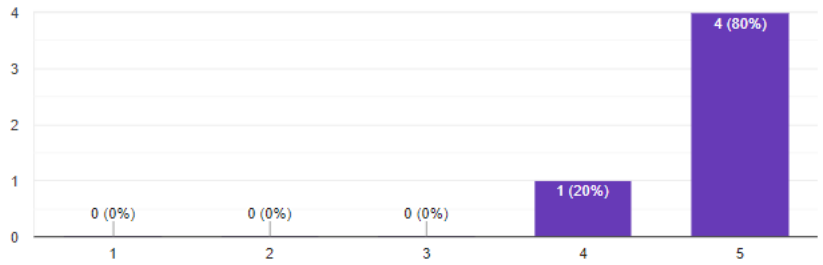
Nara

5 responses



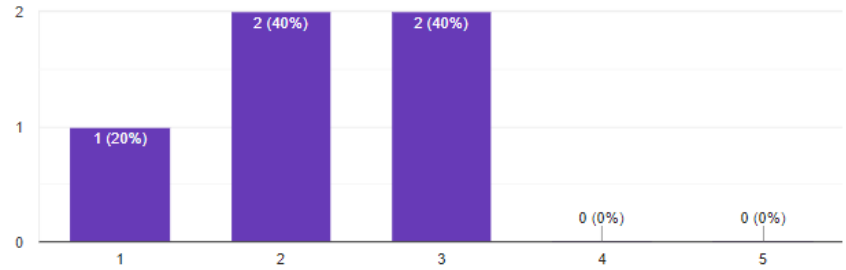
How did you enjoy singing together with your fellow musicians in round 2?

5 responses



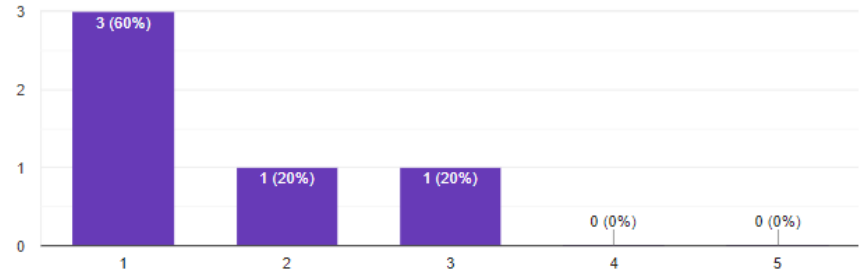
How much do you feel like the visualization contributed to your experience of singing?

5 responses



Did the visualization distract you from the singing, and if so, how much?

5 responses



Do you have any additional comments, remarks or recommendations?

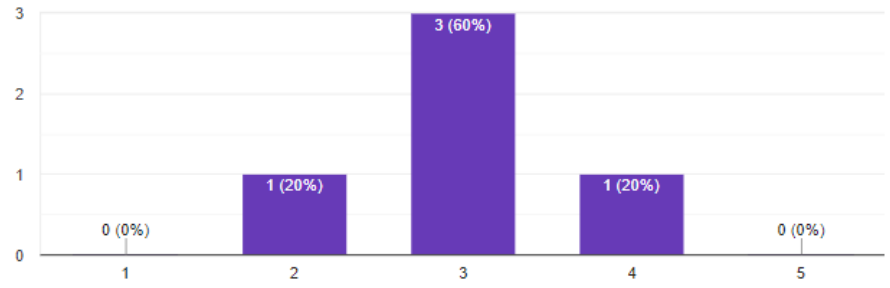
4 responses

When is a door not a door?
Misschien had ik meer op de visualisatie moeten letten
Singing is fun, and singing in the dark does make us (I feel) more synchronized
For the singing with the visualisation, it was less distracting than anticipated. But I think I listen best to the other voices when we are singing in the dark and actually have to listen to eachother and we can not look into our sheetmusic

Experience of singing, round 3

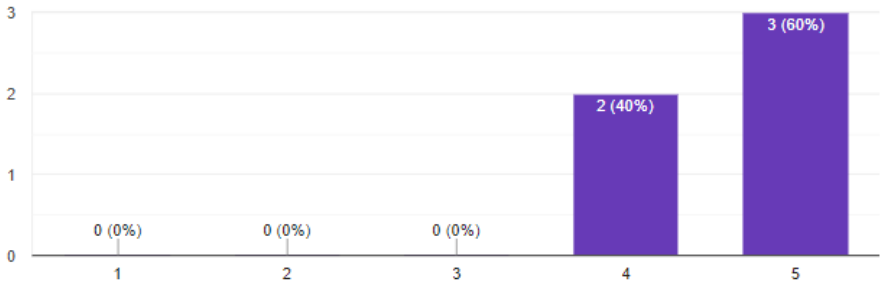
Atlantis

5 responses



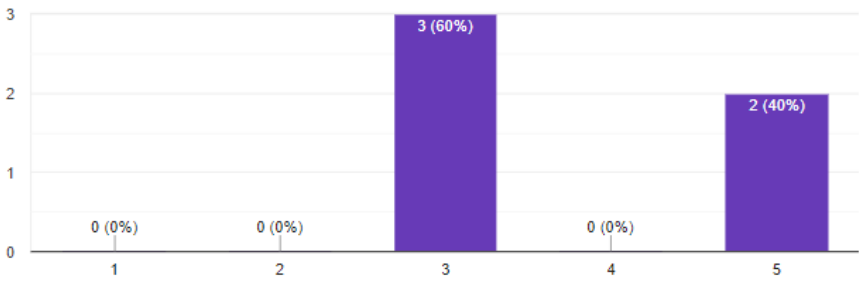
Wintersong

5 responses



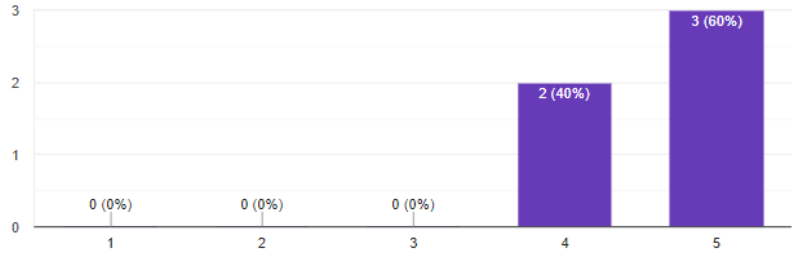
Lean On Me

5 responses



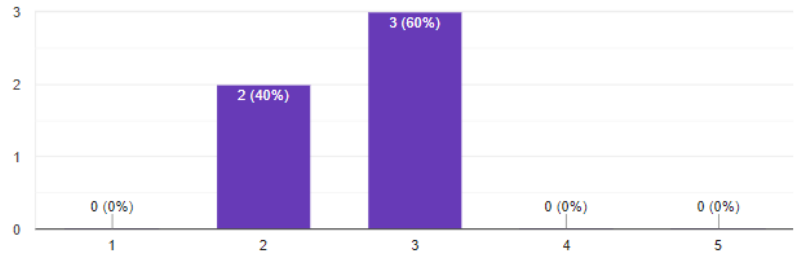
How did you enjoy singing together with your fellow musicians in round 3?

5 responses



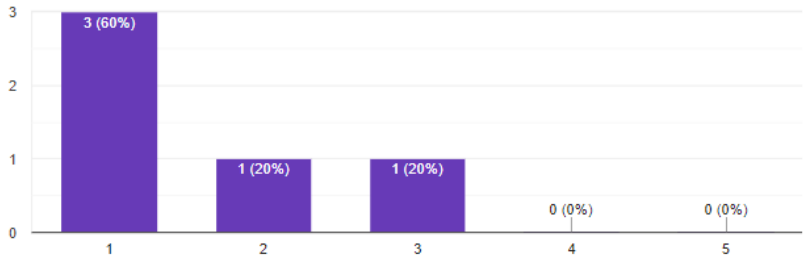
How much do you feel like the visualization contributed to your experience of singing?

5 responses



Did the visualization distract you from the singing, and if so, how much?

5 responses



Do you have any additional comments, remarks or recommendations?

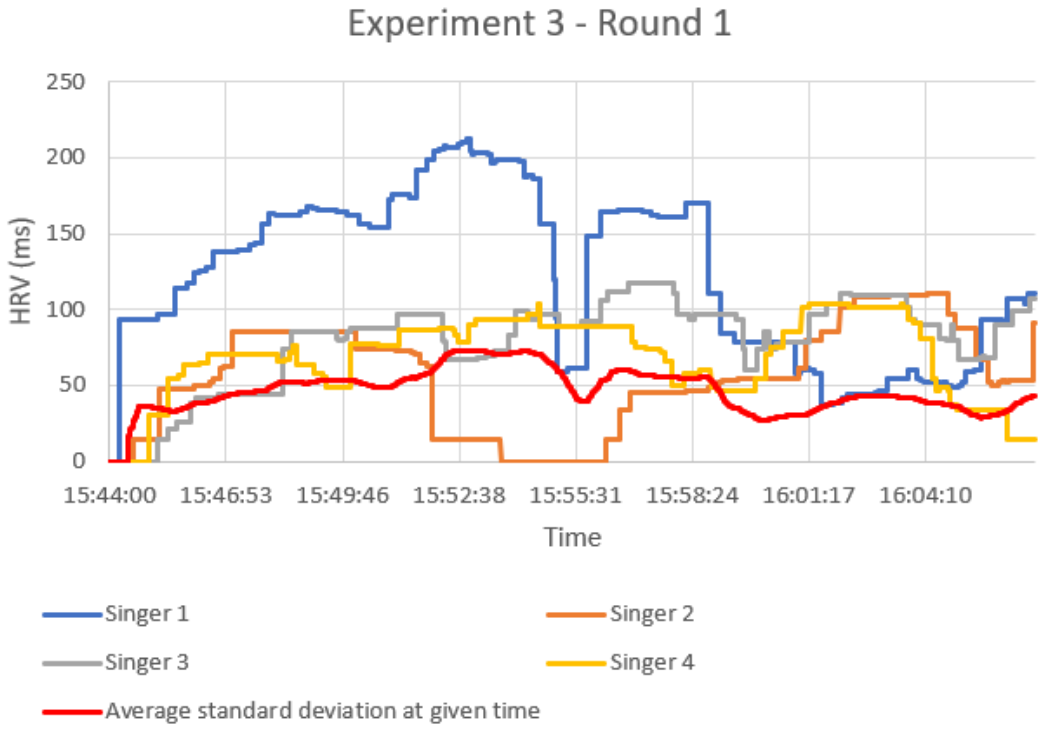
2 responses

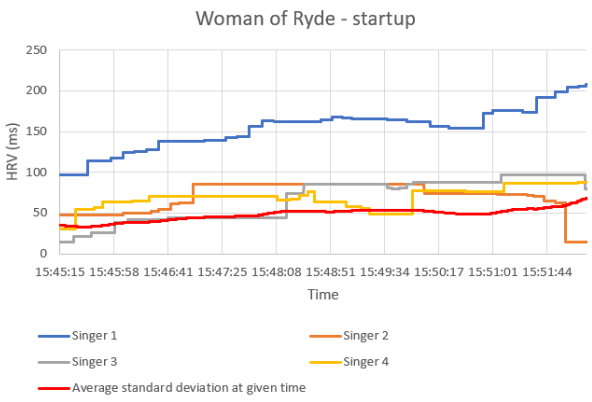
Just curious: what was different about the third round again?

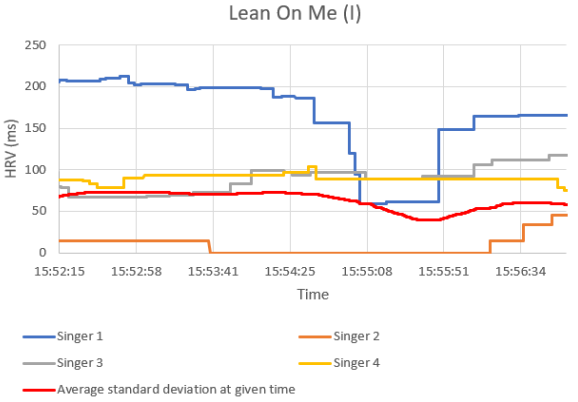
I would have liked in the third round to know what the idea was behind the lights. Now I saw a light going back and forth, but not knowing what to do with it. This made it difficult to retrieve information from it as a user.
the light was a nice neutral color which was not too distracting.

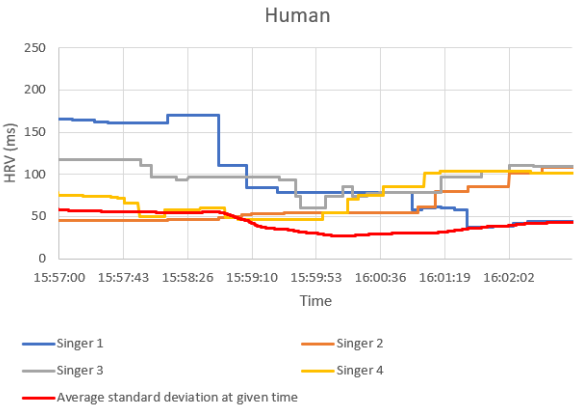
Appendix F – Full results iteration 3

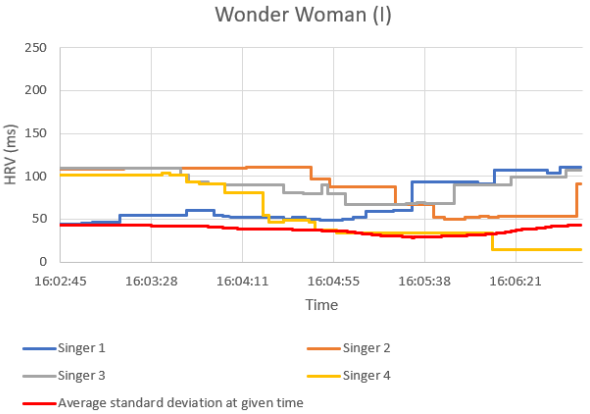
Graphs

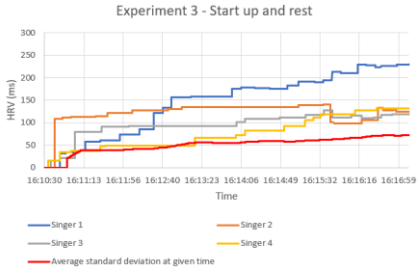


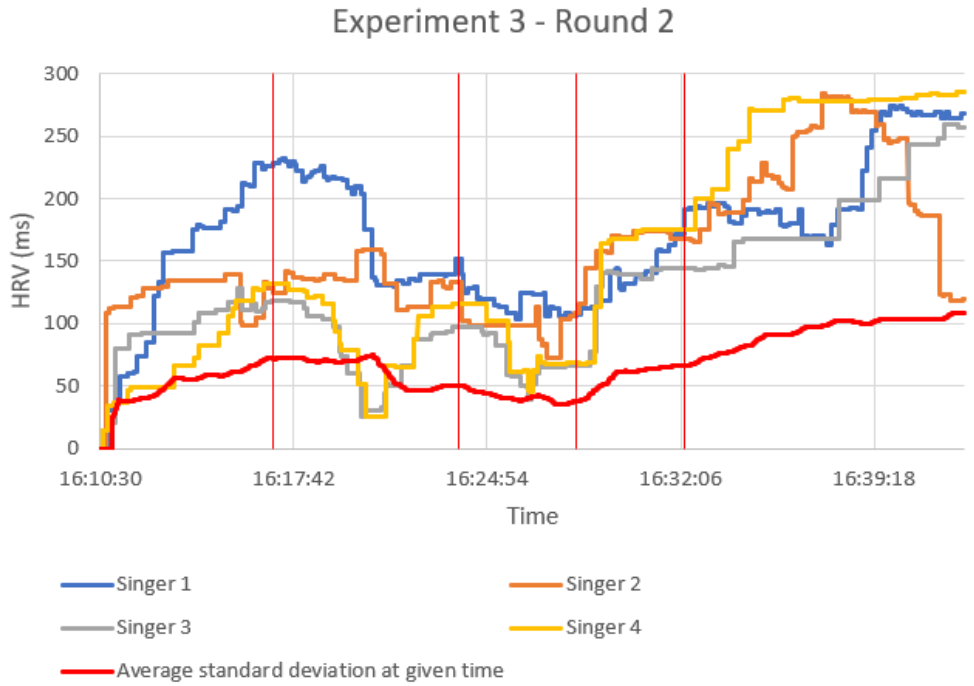


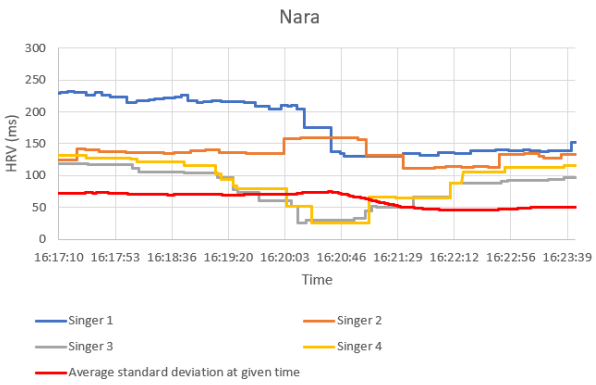


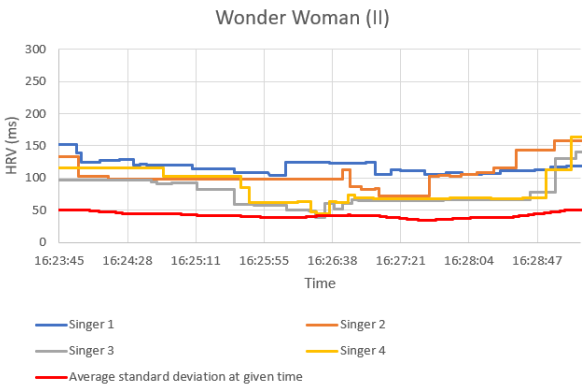


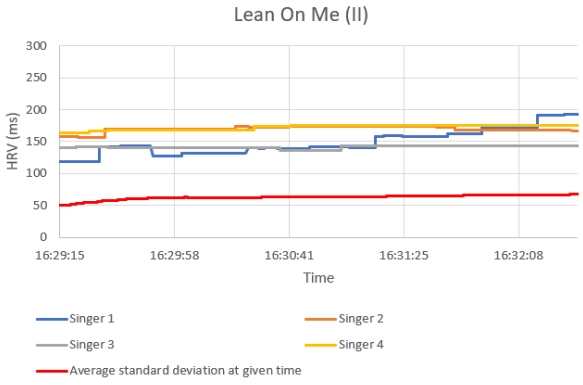


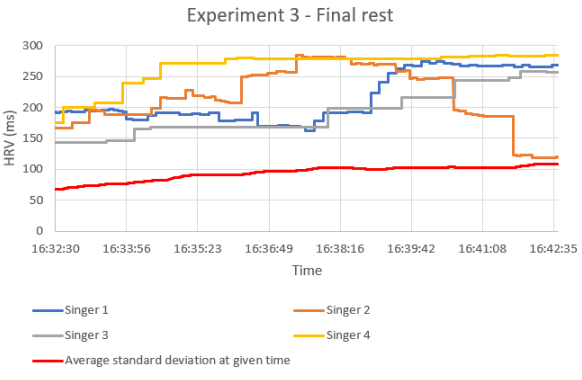

Experiment 3 - Start up and rest
HRV (ms)
Time
Singer 1
Singer 2
Singer 3
Singer 4
Average standard deviation at given time


Experiment 3 - Round 2
HRV (ms)
300
250
200
150
100
50
0
16:10:30
16:17:42
16:24:54
16:32:06
16:39:18
Time
Singer 1
Singer 2
Singer 3
Singer 4
Average standard deviation at given time


Nara
HRV (ms)
16:17:10
16:17:53
16:18:36
16:19:20
16:20:03
16:20:46
16:21:29
16:22:12
16:22:56
16:23:39
Time
Singer 1
Singer 2
Singer 3
Singer 4
Average standard deviation at given time


Wonder Woman (II)
HRV (ms)
16:23:45
16:24:28
16:25:11
16:25:55
16:26:38
16:27:21
16:28:04
16:28:47
Time
Singer 1
Singer 2
Singer 3
Singer 4
Average standard deviation at given time


Lean On Me (II)
HRV (ms)
16:29:15
16:29:58
16:30:41
16:31:25
16:32:08
Time
Singer 1
Singer 2
Singer 3
Singer 4
Average standard deviation at given time


Experiment 3 - Final rest
HRV (ms)
16:32:30
16:33:56
16:35:23
16:36:49
16:38:16
16:39:42
16:41:08
16:42:35
Time
Singer 1
Singer 2
Singer 3
Singer 4
Average standard deviation at given time

Survey

Did you experience any form of discomfort during the experiment? If so, please explain what kind of discomfort you experienced.

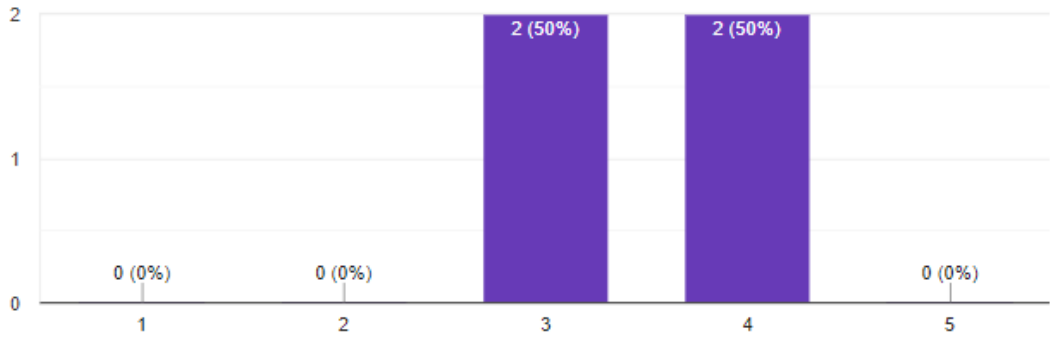
4 responses

We were off key often
Sweaty finger, and I got slightly dizzy during one song
Not really sometimes i was unsure of my notes which Made me slightly nervous
Discomfort no, more like being nervous, but that's also because I much rather sing with more people than alone

Experience of singing, round 1

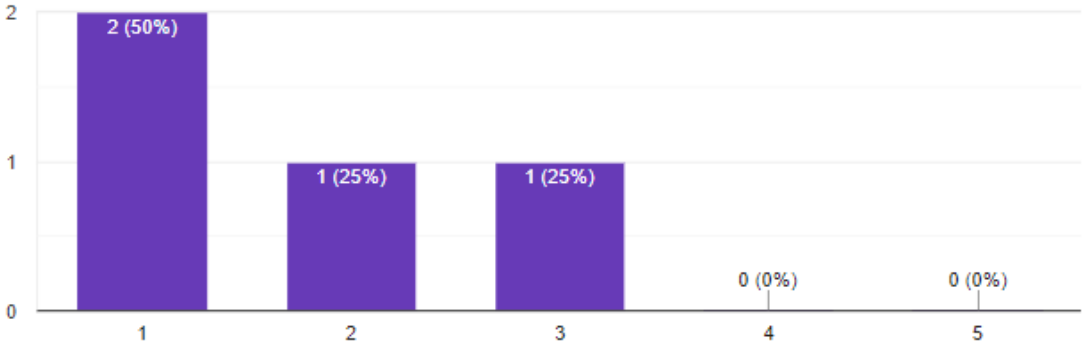
Lean On Me

4 responses



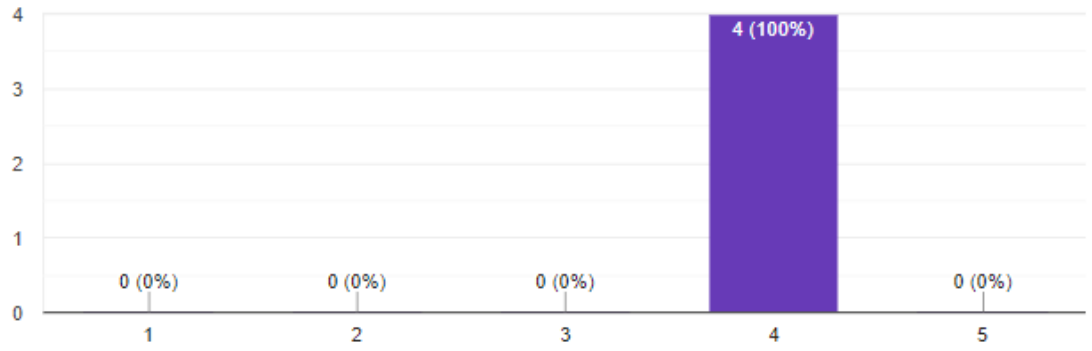
Human

4 responses



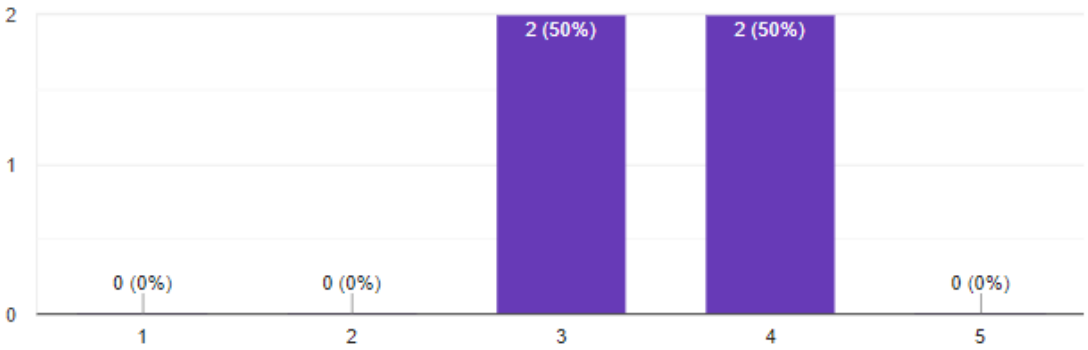
Wonder Woman

4 responses



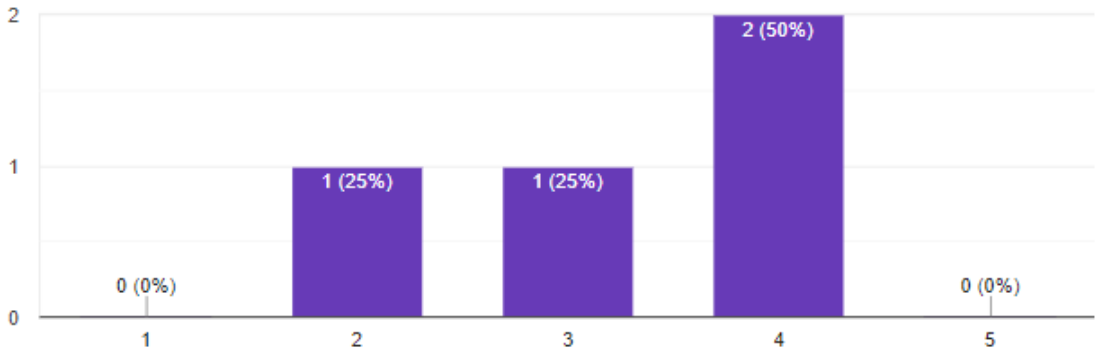
How did you enjoy singing together with your fellow musicians in round 1?

4 responses



During the period of rest between rounds, how relaxed did you feel?

4 responses



Do you have any comments on the singing in round 1? For example, any things that went wrong or right, and if so, when did they happen?

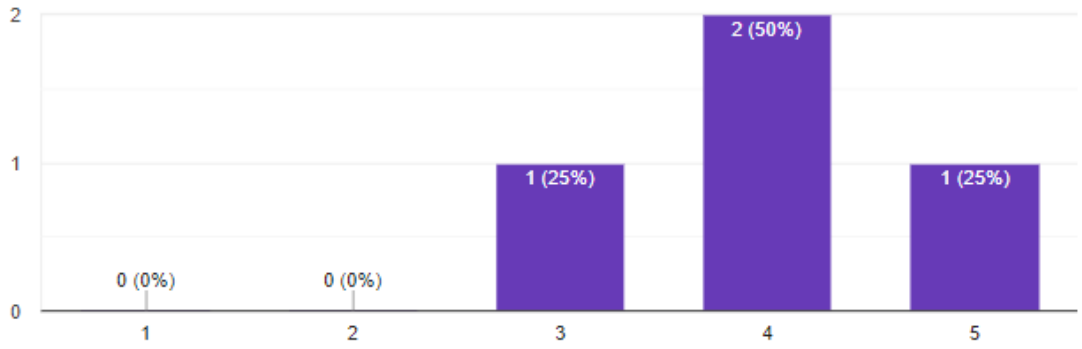
4 responses

We were off key often
Human start was pretty poor
Nope
Human didn't really go well, especially the middle part of the song, The first round of singing i felt like we still had to get used to singing together

Experience of singing, round 2

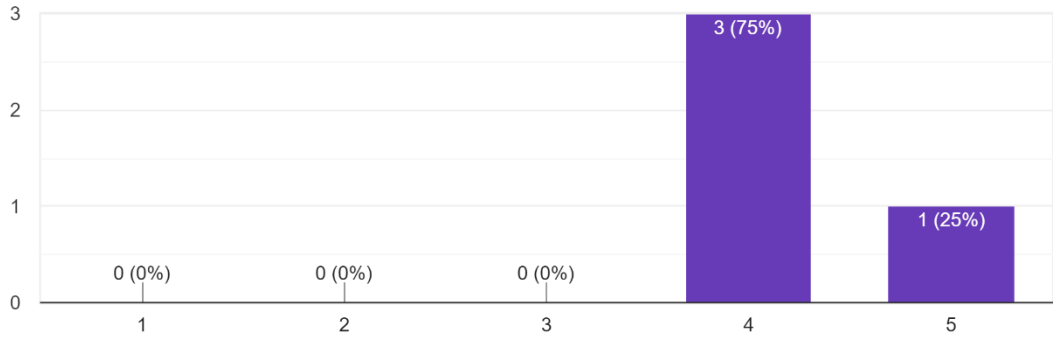
Nara

4 responses



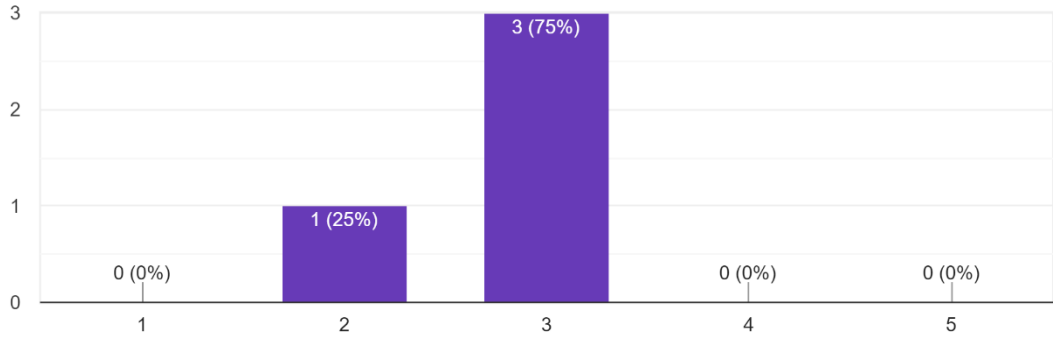
Wonder Woman

4 responses



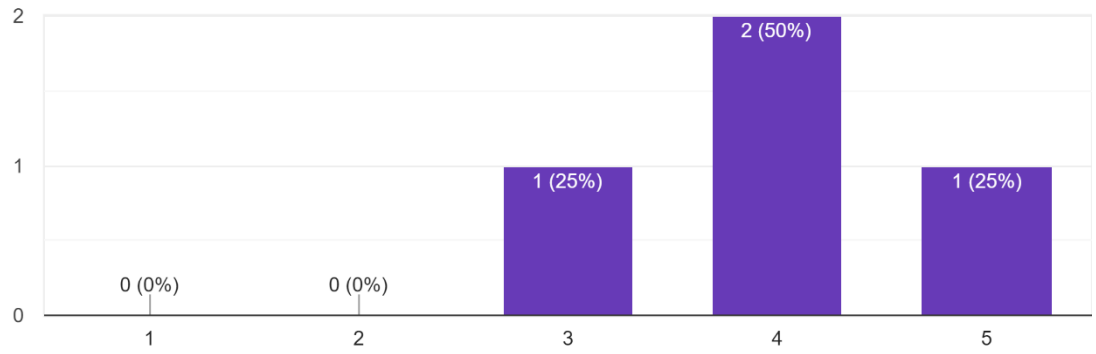
Lean On Me

4 responses



How did you enjoy singing together with your fellow musicians in round 2?

4 responses



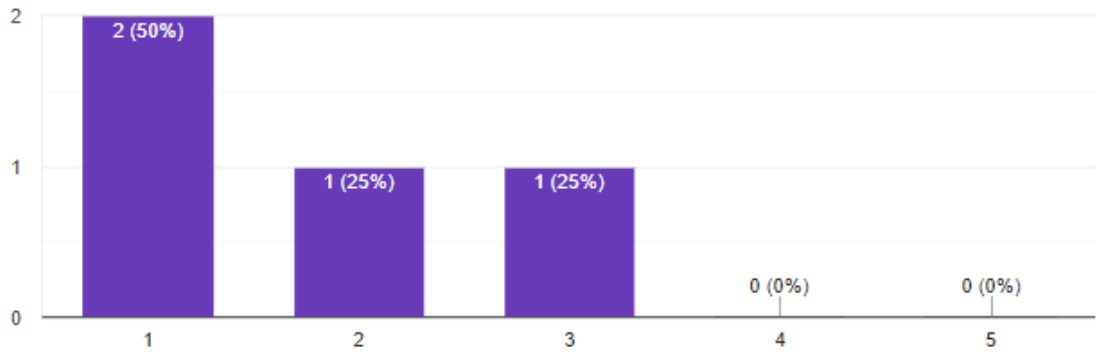
Do you have any comments on the singing in round 2? For example, any things that went wrong or right, and if so, when did they happen?

4 responses

It went a bit better
I got dizzy during Nara or Wonderwoman. During lean on me we were less in sync
Nope
The second round felt like it went a lot better than the previous round, mostly because we already sang together for a while and got used to singing together

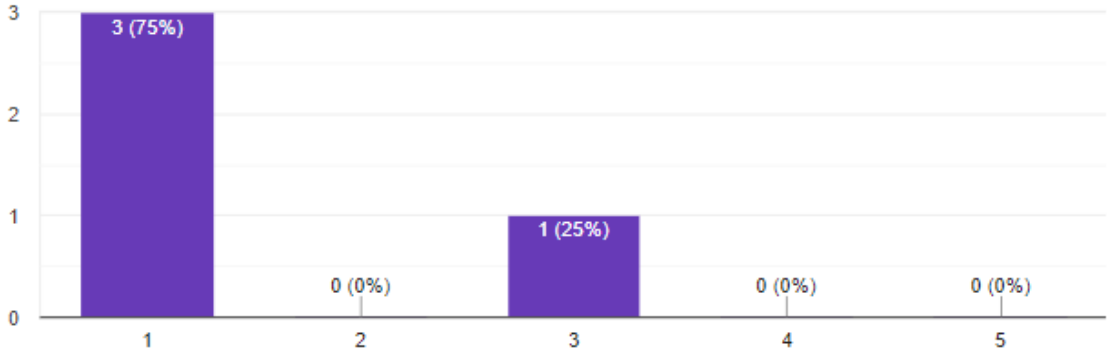
How much do you feel like the visualization contributed to your experience of singing?

4 responses



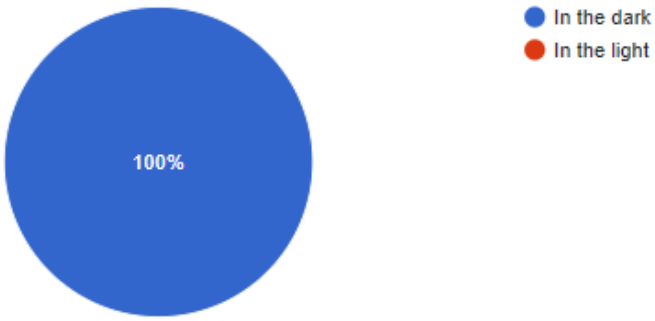
Did the visualization distract you from the singing, and if so, how much?

4 responses



Did you feel more in sync with your fellow singers when you were singing in the dark, or when you were singing in the light?

4 responses



Do you have any additional comments, remarks or recommendations?

2 responses

When it is a jar.

The hearbeat lights are distracting more than the other big LED visualization

Appendix G – Code

Arduino code iteration 1 – with WiFi

```
// Wifi serial code mainly based on code by Christopher Grant (from MEDIUM)
// https://medium.com/@cgrant/using-the-esp8266-wifi-module-with-arduino-uno-publishing-to-thingspeak-99fc77122e82
// Code partially based on the pulse sensor playground for Arduino
// See also: https://pulsesensor.com/

// ---- Software Serial setup for the esp8622 ---- //
#include <SoftwareSerial.h>
#define RX 10
#define TX 11
SoftwareSerial esp8266(RX, TX);

String AP = "XXXXXXXX"; // Insert network name
String PASS = "XXXXXXXX";// Insert network password
String API = "XXXXXXXXXXXXXXXX"; // Insert Thingspeak write API
String HOST = "api.thingspeak.com";
String PORT = "80";
String field1 = "field1";
String field2 = "field2";
int countTrueCommand;
int countTimeCommand;
boolean found = false;

// ---- Setup for the pulse sensor on this Arduino ---- //
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>

#define ARRAY_SIZE 5 // Send data every ... number of samples
#define AVERAGE_REFRESH 100 // refresh the averages every ... number of samples

const int PULSE_INPUT = A0; // A0 is the sensor pin
const int PULSE_BLINK = 13;  // Pin 13 is the on-board LED
const int PULSE_FADE = 5;
const int THRESHOLD = 500;   // Adjust this number to avoid noise when idle
const int OUTPUT_TYPE = SERIAL_PLOTTER;

PulseSensorPlayground pulseSensor;

int inputData[ARRAY_SIZE];
int arrayCounter = 0;
int averageCounter = 1; // divide by zero
int subAvg = 0;
int subStdev = 0;
int totAvg = 0; // placeholder value
int totStdev = 0;
boolean newAvg = false;

void setup() {
  Serial.begin(9600);
  esp8266.begin(115200);

  pulseSensor.analogInput(PULSE_INPUT);
  pulseSensor.blinkOnPulse(PULSE_BLINK);
  // pulseSensor.fadeOnPulse(PULSE_FADE); DO NOT USE PULSE FADE

  pulseSensor.setSerial(Serial);
  pulseSensor.setOutputType(OUTPUT_TYPE);
  pulseSensor.setThreshold(THRESHOLD);

  if (!pulseSensor.begin()) {
    /*
       PulseSensor initialization failed,
    */
    for (;;) {
```

```
      // Flash the led to show things didn't work.
      digitalWrite(PULSE_BLINK, LOW);
      delay(50);
      digitalWrite(PULSE_BLINK, HIGH);
      delay(50);
    }
  }

  int i;
  for (i = 0; i < ARRAY_SIZE; i++) {
    inputData[i] = 0; // fill the sensor array
  }

  sendCommand("AT", 5, "OK");
  sendCommand("AT+CWMODE=1", 5, "OK");
  sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 20, "OK"); // connect to
the network
}

void loop() {
  delay(20);

  refreshArray(inputData);

  if (newAvg) {
    int BPM = 60000 / totAvg; // calculate BPM
    String getDataHRV = "GET /update?api_key=" + API + "&" + field1 + "=" +
String(totStdev);
    String getDataBPM = "GET /update?api_key=" + API + "&" + field2 + "=" +
String(BPM); // currently not used

    // this code outputs the HRV to the given Thingspeak channel via WiFi
    sendCommand("AT+CIPMUX=1", 5, "OK");
    sendCommand("AT+CIPSTART=0,\"TCP\",\"" + HOST + "\"," + PORT, 15, "OK");
    sendCommand("AT+CIPSEND=0," + String(getDataHRV.length() + 4), 4, ">");
    esp8266.println(getDataHRV); delay(1500); countTrueCommand++; // print HRV
    sendCommand("AT+CIPCLOSE=0", 5, "OK");

    newAvg = false;
  }

}

int average(int Data[], int index) {
  // calculate the average of an array of ints.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    sum += Data[i];
  }
  int average = int(sum / ARRAY_SIZE);
  return average;
}

int stDeviation(int Data[], int index, int average) {
  // requires both a data array and the mean of this array as input.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    // sum of squared mean differences
    sum += (Data[i] - average) * (Data[i] - average);
  }
  float variance = sum / ARRAY_SIZE;
  int stdeviation = int(sqrt(variance));
  return stdeviation;
}

void refreshArray(int Data[]) {
```

```
  if (averageCounter < AVERAGE_REFRESH) {
    if (arrayCounter == ARRAY_SIZE) {
      int aver = average(Data, ARRAY_SIZE);
      int sdev = stDeviation(Data, ARRAY_SIZE, aver);
      // add current average and stdev to total values
      totAvg = (totAvg * (averageCounter - 1) + aver) / averageCounter;
      totStdev = (totStdev * (averageCounter - 1) + sdev) / averageCounter;

      arrayCounter = 0; // reset array counter
      averageCounter++; // increment number of averages
      newAvg = true; // tell the arduino to output data

    }
    if (pulseSensor.sawStartOfBeat()) {
      Data[arrayCounter] = pulseSensor.getInterBeatIntervalMs(); // get IBI
      arrayCounter++; // increment array index
    }

  } else {
    // refresh number of averages
    averageCounter = 2; // reset average counter but keep 1 set of data
  }
}

void sendCommand(String command, int maxTime, char readReply[]) {
  // send command function for the esp8266 wifi module
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while (countTimeCommand < maxTime) {
    esp8266.println(command); // AT + CIPSend
    if (esp8266.find(readReply)) { // get OK
      found = true;
      break;
    }
    countTimeCommand++;
  }
  if (found) {
    Serial.println("OYI"); // OK!
    countTrueCommand++;
    countTimeCommand = 0;
  }
  if (!found) {
    Serial.println("Fail"); // No OK...
    countTrueCommand = 0;
    countTimeCommand = 0;
  }
  found = false;
}
```

Arduino code iteration 1 – without WiFi

```
// Code partially based on the pulse sensor playground for Arduino
// See also: https://pulsesensor.com/

// ---- Setup for the pulse sensor on this Arduino ---- //
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>

#define ARRAY_SIZE 5 // Send data every ... number of samples
#define AVERAGE_REFRESH 100 // refresh the averages every ... number of samples

const int PULSE_INPUT = A0; // A0 is the sensor pin
const int PULSE_BLINK = 13;  // Pin 13 is the on-board LED
const int PULSE_FADE = 5;
const int THRESHOLD = 550;   // Adjust this number to avoid noise when idle
const int OUTPUT_TYPE = SERIAL_PLOTTER;

PulseSensorPlayground pulseSensor;

int inputData[ARRAY_SIZE];
int arrayCounter = 0;
int averageCounter = 1; // divide by zero
int subAvg = 0;
int subStdev = 0;
int totAvg = 0; // placeholder value
int totStdev = 0;
boolean newAvg = false;

void setup() {
  Serial.begin(9600);

  pulseSensor.analogInput(PULSE_INPUT);
  pulseSensor.blinkOnPulse(PULSE_BLINK);
  // pulseSensor.fadeOnPulse(PULSE_FADE); DO NOT USE PULSE FADE

  pulseSensor.setSerial(Serial);
  pulseSensor.setOutputType(OUTPUT_TYPE);
  pulseSensor.setThreshold(THRESHOLD);

  if (!pulseSensor.begin()) {
    /*
       PulseSensor initialization failed,
   */
    for (;;) {
      // Flash the led to show things didn't work.
      digitalWrite(PULSE_BLINK, LOW);
      delay(50);
      digitalWrite(PULSE_BLINK, HIGH);
      delay(50);
    }
  }

  int i;
  for (i = 0; i < ARRAY_SIZE; i++) {
    inputData[i] = 0; // fill the sensor array
  }
}

void loop() {
  delay(20);

  refreshArray(inputData);

  if (newAvg) {
    int BPM = 60000 / totAvg; // calculate BPM
    String output = ", " + String(totStdev) + ", " + String(BPM);
    unsigned long t = millis();
```

```
    unsigned int m = t / 60000;
    unsigned int s = t / 1000 - 60 * m;

    // serial print timestamp since start measurement and output HRV and BPM
    if (m < 10) {
      Serial.print('0');
    }
    Serial.print(m);
    Serial.print(':');
    if (s < 10) {
      Serial.print('0');
    }
    Serial.print(s);
    Serial.println(output);

    newAvg = false;
  }
}

int average(int Data[], int index) {
  // calculate the average of an array of ints.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    sum += Data[i];
  }
  int average = int(sum / ARRAY_SIZE);
  return average;
}

int stDeviation(int Data[], int index, int average) {
  // requires both a data array and the mean of this array as input.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    // sum of squared mean differences
    sum += (Data[i] - average) * (Data[i] - average);
  }
  float variance = sum / ARRAY_SIZE;
  int stdeviation = int(sqrt(variance));
  return stdeviation;
}

void refreshArray(int Data[]) {
  if (averageCounter < AVERAGE_REFRESH) {
    if (arrayCounter == ARRAY_SIZE) {
      int aver = average(Data, ARRAY_SIZE);
      int sdev = stDeviation(Data, ARRAY_SIZE, aver);
      // add current average and stdev to total values
      totAvg = (totAvg * (averageCounter - 1) + aver) / averageCounter;
      totStdev = (totStdev * (averageCounter - 1) + sdev) / averageCounter;

      arrayCounter = 0; // reset array counter
      averageCounter++; // increment number of averages
      newAvg = true; // tell the arduino to output data

    }
    if (pulseSensor.sawStartOfBeat()) {
      Data[arrayCounter] = pulseSensor.getInterBeatIntervalMs(); // get IBI
      arrayCounter++; // increment array index
    }
  } else {
    // refresh number of averages
    averageCounter = 2; // reset average counter but keep 1 set of data
  }
}
```

Arduino code iteration 2

```
// Sensor usage code partially based on the pulse sensor playground for Arduino
// See also: https://pulsesensor.com/

// NeoPixel code mainly copied and based on
// the multitasking with Arduino and NeoPixel tutorial by Bill Earl, Adafruit
// https://learn.adafruit.com/multi-tasking-the-arduino-part-3

#define NUM_GLOBAL_SENSORS 5 // number of sensors in the overall setup

// ---- Setup for the NeoPixel ---- //
#include <Adafruit_NeoPixel.h>

#define PIN 5 // pin the Neopixel is connected to
#define NUM_LEDS 35 // number of LEDs on the Neopixel

// Pattern types supported:
enum  pattern { NONE, SCANNER, FADE };
// Patern directions supported:
enum  direction { FORWARD, REVERSE };

// NeoPattern Class - derived from the Adafruit_NeoPixel class
// C++ class and function definitions to enable use of no loops
class NeoPatterns : public Adafruit_NeoPixel {
  public:
    // Member Variables:
    pattern  ActivePattern;  // which pattern is running
    direction Direction;     // direction to run the pattern

    unsigned long Interval;   // milliseconds between updates
    unsigned long lastUpdate; // last update of position

    uint32_t Color1, Color2;  // What colors are in use
    uint16_t TotalSteps;  // total number of steps in the pattern
    uint16_t Index;  // current step within the pattern

    void (*OnComplete)();  // Callback on completion of pattern

    // Constructor - calls base-class constructor to initialize strip
    NeoPatterns(uint16_t pixels, uint8_t pin, uint8_t type, void (*callback)())
      : Adafruit_NeoPixel(pixels, pin, type) {
      OnComplete = callback;
    }

    // Update the pattern
    void Update() {
      if ((millis() - lastUpdate) > Interval) {
        // time to update
        lastUpdate = millis();
        switch (ActivePattern) {
          case SCANNER:
            ScannerUpdate();
            break;
          case FADE:
            FadeUpdate();
            break;
          default:
            break;
        }
      }
    }
    // Increment the Index and reset at the end
    void Increment() {
      if (Direction == FORWARD) {
        Index++;
        if (Index >= TotalSteps) {
          Index = 0;
```

```
          if (OnComplete != NULL) {
            OnComplete(); // call the comlpetion callback
          }
        }
      } else {
        // Direction == REVERSE
        --Index;
        if (Index <= 0) {
          Index = TotalSteps - 1;
          if (OnComplete != NULL) {
            OnComplete(); // call the comlpetion callback
          }
        }
      }
    }
    // Reverse pattern direction
    void Reverse() {
      if (Direction == FORWARD) {
        Direction = REVERSE;
        Index = TotalSteps - 1;
      } else {
        Direction = FORWARD;
        Index = 0;
      }
    }
    // Initialize for a SCANNNER
    void Scanner(uint32_t color1, uint8_t interval) {
      ActivePattern = SCANNER;
      Interval = interval;
      TotalSteps = (numPixels() - 1) * 2;
      Color1 = color1;
      Index = 0;
    }
    // Update the Scanner Pattern
    void ScannerUpdate() {
      for (int i = 0; i < numPixels(); i++) {
        if (i == Index) {
          // Scan Pixel to the right
          setPixelColor(i, Color1);
          //setPixelColor(numPixels()-i,Color1);
        } else if (i == TotalSteps - Index) {
          // Scan Pixel to the left
          setPixelColor(i, Color1);
          //setPixelColor(numPixels()-i,Color1);
        } else {
          // Fading tail
          setPixelColor(i, DimColor(getPixelColor(i)));
          //setPixelColor(numPixels()-i, DimColor(getPixelColor(numPixels()-i)));
        }
      }
      show();
      Increment();
    }

    // Initialize for a Fade
    void Fade(uint32_t color1, uint32_t color2, uint16_t steps, uint8_t interval,
direction dir = FORWARD) {
      ActivePattern = FADE;
      Interval = interval;
      TotalSteps = steps;
      Color1 = color1;
      Color2 = color2;
      Index = 0;
      Direction = dir;
    }

    // Update the Fade Pattern
    void FadeUpdate() {
```

```
      // Calculate linear interpolation between Color1 and Color2
      // Optimise order of operations to minimize truncation error
      uint8_t red = ((Red(Color1) * (TotalSteps - Index)) + (Red(Color2) * Index))
/ TotalSteps;
      uint8_t green = ((Green(Color1) * (TotalSteps - Index)) + (Green(Color2) *
Index)) / TotalSteps;
      uint8_t blue = ((Blue(Color1) * (TotalSteps - Index)) + (Blue(Color2) *
Index)) / TotalSteps;

      ColorSet(Color(red, green, blue));
      show();
      Increment();
    }

    // Calculate 50% dimmed version of a color (used by ScannerUpdate)
    uint32_t DimColor(uint32_t color) {
      // Shift R, G and B components one bit to the right
      uint32_t dimColor = Color(Red(color) >> 1, Green(color) >> 1, Blue(color) >>
1);
      return dimColor;
    }

    // Set all pixels to a color (synchronously)
    void ColorSet(uint32_t color) {
      for (int i = 0; i < numPixels(); i++) {
        setPixelColor(i, color);
      }
      show();
    }

    // Returns the Red component of a 32-bit color
    uint8_t Red(uint32_t color) {
      return (color >> 16) & 0xFF;
    }

    // Returns the Green component of a 32-bit color
    uint8_t Green(uint32_t color) {
      return (color >> 8) & 0xFF;
    }

    // Returns the Blue component of a 32-bit color
    uint8_t Blue(uint32_t color) {
      return color & 0xFF;
    }

    // Input a value 0 to 255 to get a color value.
    // The colours are a transition r - g - b - back to r.
    uint32_t Wheel(byte WheelPos) {
      WheelPos = 255 - WheelPos;
      if (WheelPos < 85) {
        return Color(255 - WheelPos * 3, 0, WheelPos * 3);
      } else if (WheelPos < 170) {
        WheelPos -= 85;
        return Color(0, WheelPos * 3, 255 - WheelPos * 3);
      } else {
        WheelPos -= 170;
        return Color(WheelPos * 3, 255 - WheelPos * 3, 0);
      }
    }
};

void StripComplete();

NeoPatterns Strip(NUM_LEDS, PIN, NEO_GRB + NEO_KHZ800, &StripComplete);

// ---- Setup for the pulse sensor on this Arduino ---- //
#define USE_ARDUINO_INTERRUPTS true
```

```
#include <PulseSensorPlayground.h>
const int PULSE_SENSOR_COUNT = NUM_GLOBAL_SENSORS; // number of senors on this
arduino

#define ARRAY_SIZE 5 // Calculate new standard deviation every ... number of
samples
#define AVERAGE_REFRESH 100 // refresh the averages every ... number of samples

const int PULSE_INPUT0 = A0; // A0 is the sensor pin
const int PULSE_BLINK0 = 13; // LED on this pin blinks with heartbeat

const int PULSE_INPUT1 = A1;
const int PULSE_BLINK1 = 12;

const int PULSE_INPUT2 = A2;
const int PULSE_BLINK2 = 11;

const int PULSE_INPUT3 = A3;
const int PULSE_BLINK3 = 10;

const int PULSE_INPUT4 = A4;
const int PULSE_BLINK4 = 9;

const int THRESHOLD = 500;   // Adjust this number to avoid noise when idle
const int OUTPUT_TYPE = SERIAL_PLOTTER;

PulseSensorPlayground pulseSensor(PULSE_SENSOR_COUNT);

// no two-dimensional arrays, so define a separate one for each sensor
int inputData0[ARRAY_SIZE];
int inputData1[ARRAY_SIZE];
int inputData2[ARRAY_SIZE];
int inputData3[ARRAY_SIZE];
int inputData4[ARRAY_SIZE];

int outputData[NUM_GLOBAL_SENSORS];
boolean newData[NUM_GLOBAL_SENSORS];
char headers[NUM_GLOBAL_SENSORS];

int arrayCounter[PULSE_SENSOR_COUNT];
int averageCounter[PULSE_SENSOR_COUNT];
int totAvg[PULSE_SENSOR_COUNT];
int totStdev[PULSE_SENSOR_COUNT]; // Stores the HRV
boolean newAvg[PULSE_SENSOR_COUNT];

// button setup
int buttonState = 0;
#define BUTTON_PIN 4
boolean buttonPress = false;
boolean visOn = false;

// following variables are used for void compareData
#define COMPARE_NUM 30 // number of samples to compare the average standard
deviation on
unsigned long timer;
unsigned int timerThreshold = 1000; // how many ms between samples for caculating
the average standard deviation
int comparison[COMPARE_NUM];
int comparisonIndex = 0;
int averageThreshold = 60; // threshold for the visualization. Average standard
deviation over the last 30 seconds.

void setup() {
  Serial.begin(9600); // send data at 9600 baud

  pulseSensor.analogInput(PULSE_INPUT0, 0); // set sensor pin as input
  pulseSensor.blinkOnPulse(PULSE_BLINK0, 0); // set LED pin to blink with heartbeat
```

```
  pulseSensor.analogInput(PULSE_INPUT1, 1);
  pulseSensor.blinkOnPulse(PULSE_BLINK1, 1);

  pulseSensor.analogInput(PULSE_INPUT2, 2);
  pulseSensor.blinkOnPulse(PULSE_BLINK2, 2);

  pulseSensor.analogInput(PULSE_INPUT3, 3);
  pulseSensor.blinkOnPulse(PULSE_BLINK3, 3);

  pulseSensor.analogInput(PULSE_INPUT4, 4);
  pulseSensor.blinkOnPulse(PULSE_BLINK4, 4);

  pulseSensor.setSerial(Serial);
  pulseSensor.setOutputType(OUTPUT_TYPE);
  pulseSensor.setThreshold(THRESHOLD, 0);

  if (!pulseSensor.begin()) {
    // sketch does not work with this version of the Arduino (interrupts)
    for (;;) {
      // Flash the led to show things didn't work.
      digitalWrite(PULSE_BLINK0, LOW);
      delay(50);
      digitalWrite(PULSE_BLINK0, HIGH);
      delay(50);
    }
  }

  // a lot of variable value initializations below
  int i;
  for (i = 0; i < ARRAY_SIZE; i++) {
    inputData0[i] = 0; // fill the sensor array
    inputData1[i] = 0; // fill the sensor array
    inputData2[i] = 0; // fill the sensor array
    inputData3[i] = 0; // fill the sensor array
    inputData4[i] = 0; // fill the sensor array
  }
  int j;
  for (j = 0; j < NUM_GLOBAL_SENSORS; j++) {
    outputData[j] = 0; // fill the output array
    newData[j] = false;
    headers[j] = 'A' + j; // characters A and on for headers
  }
  int k;
  for (k = 0; k < PULSE_SENSOR_COUNT; k++) {
    arrayCounter[k] = 0;
    averageCounter[k] = 1; // avoid divide by 0
    totAvg[k] = 0;
    totStdev[k] = 0;
    newAvg[k] = false;
  }
  int l;
  for (l = 0; l < COMPARE_NUM; l++) {
    comparison[l] = 0;
  }

  Strip.begin(); // initialize NeoPixel
  Strip.ColorSet(Strip.Color(0, 0, 0)); // reset NeoPixel
  pinMode(BUTTON_PIN, INPUT);

  Strip.Scanner(Strip.Color(0, 30, 255), 75);
}

void loop() {
  // get new values for each sensor, and update HRVs accordingly
  refreshArray(inputData0, 0);
  refreshArray(inputData1, 1);
  refreshArray(inputData2, 2);
  refreshArray(inputData3, 3);
```

```
  refreshArray(inputData4, 4);

  int h;
  for (h = 0; h < PULSE_SENSOR_COUNT; h++) {
    if (newAvg[h]) {
      // triggers when a new HRV value is available
      outputData[h] = totStdev[h]; // store HRV for sensor on this Arduino
      newData[h] = true;
      newAvg[h] = false;
    }
  }

  int j;
  for (j = 0; j < NUM_GLOBAL_SENSORS; j++) {
    if (newData[j]) {
      Serial.print(headers[j]);
      Serial.println(outputData[j]); // output data to Serial for the Processing
graphwriter
      newData[j] = false;
    }
  }

  checkButton(); // See if the button state has changed
  compareData(); // Use for visualization based on amount of sync, define
averageThreshold beforehand

  if (visOn) {
    // start moving visualization
    Strip.Update();
  }
}

int average(int Data[], int index) {
  // calculate the average of an array of ints.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    sum += Data[i];
  }
  int average = int(sum / ARRAY_SIZE);
  return average;
}

int stDeviation(int Data[], int index, int average) {
  // requires both a data array and the mean of this array as input.
  // Use average() for this.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    // sum of squared mean differences
    sum += (Data[i] - average) * (Data[i] - average);
  }
  float variance = sum / ARRAY_SIZE;
  int stdeviation = int(sqrt(variance));
  return stdeviation;
}

void refreshArray(int Data[], int sensor_ID) {

  if (averageCounter[sensor_ID] < AVERAGE_REFRESH) {
    if (arrayCounter[sensor_ID] == ARRAY_SIZE) {
      int aver = average(Data, ARRAY_SIZE); // necessary for standard deviation
      int sdev = stDeviation(Data, ARRAY_SIZE, aver);
      // add current average and stdev to total values
      totAvg[sensor_ID] = (totAvg[sensor_ID] * (averageCounter[sensor_ID] - 1) +
aver) / averageCounter[sensor_ID];
      totStdev[sensor_ID] = (totStdev[sensor_ID] * (averageCounter[sensor_ID] - 1)
+ sdev) / averageCounter[sensor_ID];
```

```
      arrayCounter[sensor_ID] = 0; // reset array counter
      averageCounter[sensor_ID]++; // increment number of averages
      newAvg[sensor_ID] = true; // tell the arduino to output data

    }
    if (pulseSensor.sawStartOfBeat(sensor_ID)) {
      Data[arrayCounter[sensor_ID]] =
pulseSensor.getInterBeatIntervalMs(sensor_ID); // get IBI
      arrayCounter[sensor_ID]++; // increment array index
    }
  } else {
    // refresh number of averages
    averageCounter[sensor_ID] = 2; // reset average counter but keep 1 set of data
  }
}

void compareData() {
  if (millis() > timer + timerThreshold) {
    // compare data every second
    timer = millis();
    int i;
    int tempCompare[NUM_GLOBAL_SENSORS];
    int aver;
    for (i = 0; i < NUM_GLOBAL_SENSORS; i++) {
      tempCompare[i] = outputData[i];
    }
    aver = average(tempCompare, NUM_GLOBAL_SENSORS);
    comparison[comparisonIndex] = stDeviation(tempCompare, NUM_GLOBAL_SENSORS,
aver);
    if (comparisonIndex < COMPARE_NUM) {
      comparisonIndex++;
    } else {
      comparisonIndex = 0; // set index  to 0
    }

    if (millis() > (timerThreshold * COMPARE_NUM)) {
      // wait 30 seconds before visualizing anything
      int j;
      int prevCompared;
      int compared = average(comparison, COMPARE_NUM);
      Serial.print('F'); // header
      Serial.println(compared); // Serial print average stdev for analysis purposes
      if (compared < averageThreshold ) {
        visOn = true; // turn on the visualization, only use if not using button
      }
    }
  }
}

void checkButton() {
  // check and update button state and visualization state
  // to be used for manually activiting the visualization
  buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == HIGH && !buttonPress && !visOn) {
    visOn = true; // turn visualization on
  } else if (buttonState == HIGH && !buttonPress && visOn) {
    visOn = false; // turn visualization off
  }

  buttonPress = buttonState;
}

void StripComplete() {
  // To use for NeoPixel visualization; currently unused
  // Strip.Color1 = Strip.Color(0,0,0);
  // Strip.Reverse();
}
```

Processing graphwriter iteration 2 & 3

```
////////////////////////////////////////////////////////////////////////////
// Analog signal graph writer
// plots 6 analog signals, received at 9600Bps
// every value preceeded by A, B, C, D, E or F
//
// The screen can be paused using the spacebar
// pressing the space again resumes running
// 'c' clears the screen
// 'r' starts recording the values in a file
// 's' stops recording.
// everytime 'r' has been pressed a new
// file 'values1.txt' with consequetive number will be made
// ____________________________________________________________________________
// E.Dertien - sensors for Creative Technology - (cc) 2016
////////////////////////////////////////////////////////////////////////////
// Configure the following parameters before running the sketch:
int channels[] = {0,1,2,3,4,5};  // select the channels [0..5] to print, any
number, max 6
int PORTNUMBER = 0;          // select the correct portnumber from the printed list
int gridlines = 1;           // on/off for printing gridlines
//////////////////////////////
float scaling = 2;             // default: [0..1023] is mapped to [0..255]
import processing.serial.*;  // standard serial library
Serial port;                 // port object for serial communication
String buff = "";            // input buffer for serial data
int NEWLINE = '\n';          // terminator of the serial commands
char header[] = {  'A', 'B', 'C', 'D', 'E', 'F'}; // headers for the values
int linecolor[] = {0, 80, 120, 160, 200, 40};     // colors for the lines (HSB)
PFont fontA;                 // screen font
int value[] = new int[6];    // array of current received values
int diffValue[] = new int[6];// array of previously received values
PrintWriter output;          // use file output
String outputBuff = "";      // buffer for file output
int filenr;                  // current file (incremental while the sketch is
running)
int offset = 0;              // default: no offset in graph
float timescaling = 0.1;     // default: (1.0) = 25 sec time/per window
float amplitude = 1.0;       // default: (1.0) = 0..84 ms range per window
String mode="RUN";           // mode can be RUN, PAUSE or RECORD
float n=21;                  // reset cursor position to 21..

void setup() {
  size(533, 286);
  println("Available serial ports:");
  for (int i = 0; i<Serial.list ().length; i++) {
    print("[" + i + "] ");
    println(Serial.list()[i]);
  }
  port = new Serial(this, Serial.list()[PORTNUMBER], 9600);
  frameRate(20);  // delay of 50 ms, 20Hz update
  colorMode(HSB);
  fontA = loadFont("SansSerif-10.vlw");
  textFont(fontA, 10);
  background(255);
  drawscreen();
  port.write(1);  // sometimes necessary to get the serial communication
starting...
}
void draw() {
  while (port.available () > 0) {
    serialEvent(port.read()); // read data
  }
  outputBuff="" + millis();
  stroke(255);             // clear the space for the time
  fill(255);               // clear the space for the time
  rect(300, 0, 230, 12);  // clear the space for the time
```

```
  stroke(0);
  fill(0);
  text("time "+(millis()-500)/1000.0 +" s", 300, 12); // print actual time
  for(int i = 0; i<channels.length; i++){
    fill(linecolor[channels[i]], 255, 255);
    text("A" + channels[i] + " ",400+(i*20),12);
  }

  for (int z=0; z<channels.length; z++) {             // print the lines
    stroke(linecolor[channels[z]], 255, 255);
    line(n-timescaling, (height-12-offset)-
(amplitude*diffValue[channels[z]]/scaling), n, (height-12-offset)-
(amplitude*value[channels[z]]/scaling)); //draw line
    outputBuff += ("," + value[channels[z]]);
  }
  if (mode=="RECORD") {
    output.println(outputBuff);
    print(".");
  }
  if (mode!="PAUSE") n+=timescaling;
  if (n>width) {
    n=21;
    background(255); // clear screen
    drawscreen();
  }
  for (int z=0; z<6; z++) {
    diffValue[z]=value[z]; // store previous values
  }
}
void drawscreen() {
  stroke(255, 0, 0);
  line(20, height-11, width, height-11);
  line(20, 13, width, 13);
  line(20, 0, 20, height);
  fill(255, 0, 0);
  text("0.0", 2, height-7-offset);
  text(nf(250/amplitude, 1, 1), 2, (height/2+7)-offset);
  text(nf(500/amplitude, 1, 1), 2, 23-offset);
  text("amplitude x "+nf(amplitude, 1, 1), 50, 12);
  text("time x "+nf(timescaling, 1, 1), 150, 12);
  text("offset "+offset+ " px", 220, 12);
  text("0 (s)", 21, height-1);
  for (int n=1; n<6; n++) {
    text(nf((5*n)/timescaling, 0, 0), 100*n+21, height-1);
  }
  if (gridlines>0) {
    stroke(200);                                    // use grey lines
    for (float q=19; q<280; q+=25.5) {              // draw grid
      line(21, 0+q-offset, width, 0+q-offset);      // horizontal grid lines
    }
    for (int q=20; q<520; q+=20) {                  //vertical grid lines
      line(20+q, 14, 20+q, 274);
    }
  }
}
void serialEvent(int serial)
{
  try {                          // try-catch because of transmission errors
    if (serial != NEWLINE) {
      buff += char(serial);
    } else {
      println(buff);             // The first character tells us which axis this
value is for
      char c = buff.charAt(0);   // Remove it from the string
      buff = buff.substring(1); // Discard the carriage return at the end of the
buffer
      buff = buff.substring(0, buff.length()-1); // Parse the String into an
integer
```

```
      for (int z=0; z<6; z++) { // always all 6 values
        if (c == header[z]) {
          value[z] = Integer.parseInt(buff);
        }
      }
      buff = "";                // Clear the value of "buff"
    }
  }
  catch(Exception e) {
    println("no valid data");
  }
}
void keyPressed() {
  if (key==' ' && mode=="RUN") mode="PAUSE";
  else if (key==' ' && mode=="PAUSE") mode="RUN";
  if (key=='r' && mode!="RECORD") {
    mode="RECORD";
    print("Start recording...");
    filenr++;
    output = createWriter("values"+filenr+".txt");
  }
  if (key=='s' && mode=="RECORD") {
    mode="STOP";
    println("ready!");
    output.flush(); // Write the remaining data
    output.close(); // Finish the file
  }
  if (key=='c') {
    n=21;
    background(255);
  }
  if (key=='1' && amplitude>0.11) amplitude -=0.1;
  if (key=='2') amplitude +=0.1;
  if (key=='3' && timescaling > 0.11) timescaling -=0.1;
  if (key=='4') timescaling +=0.1;
  if (key=='5') offset -=1;
  if (key=='6') offset +=1;

  background(255); // clear screen
  drawscreen();    // redraw axes
}
```

Arduino code iteration 3

```
// Sensor usage code partially based on the pulse sensor playground for Arduino
// See also: https://pulsesensor.com/

// NeoPixel code mainly copied and based on
// the multitasking with Arduino and NeoPixel tutorial by Bill Earl, Adafruit
// https://learn.adafruit.com/multi-tasking-the-arduino-part-3

// HRV obtainment method derived from https://imotions.com/blog/heart-rate-
variability/ -> SDANN

#define NUM_GLOBAL_SENSORS 5 // number of sensors in the overall setup

// ---- Setup for the NeoPixel ---- //
#include <Adafruit_NeoPixel.h>

#define PIN 5 // pin the Neopixel is connected to
#define NUM_LEDS 35 // number of LEDs on the Neopixel

// Pattern types supported:
enum  pattern { NONE, SCANNER, FADE };
// Patern directions supported:
enum  direction { FORWARD, REVERSE };

// NeoPattern Class - derived from the Adafruit_NeoPixel class
// C++ class and function definitions to enable use of no loops
class NeoPatterns : public Adafruit_NeoPixel {
  public:
    // Member Variables:
    pattern  ActivePattern;  // which pattern is running
    direction Direction;     // direction to run the pattern

    unsigned long Interval;   // milliseconds between updates
    unsigned long lastUpdate; // last update of position

    uint32_t Color1, Color2;  // What colors are in use
    uint16_t TotalSteps;  // total number of steps in the pattern
    uint16_t Index;  // current step within the pattern

    void (*OnComplete)();  // Callback on completion of pattern

    // Constructor - calls base-class constructor to initialize strip
    NeoPatterns(uint16_t pixels, uint8_t pin, uint8_t type, void (*callback)())
      : Adafruit_NeoPixel(pixels, pin, type) {
      OnComplete = callback;
    }

    // Update the pattern
    void Update() {
      if ((millis() - lastUpdate) > Interval) {
        // time to update
        lastUpdate = millis();
        switch (ActivePattern) {
          case SCANNER:
            ScannerUpdate();
            break;
          case FADE:
            FadeUpdate();
            break;
          default:
            break;
        }
      }
    }

    // Increment the Index and reset at the end
    void Increment() {
```

```
      if (Direction == FORWARD) {
        Index++;
        if (Index >= TotalSteps) {
          Index = 0;
          if (OnComplete != NULL) {
            OnComplete(); // call the comlpetion callback
          }
        }
      } else {
        // Direction == REVERSE
        --Index;
        if (Index <= 0) {
          Index = TotalSteps - 1;
          if (OnComplete != NULL) {
            OnComplete(); // call the comlpetion callback
          }
        }
      }
    }

    // Reverse pattern direction
    void Reverse() {
      if (Direction == FORWARD) {
        Direction = REVERSE;
        Index = TotalSteps - 1;
      } else {
        Direction = FORWARD;
        Index = 0;
      }
    }

    // Initialize for a SCANNNER
    void Scanner(uint32_t color1, uint8_t interval) {
      ActivePattern = SCANNER;
      Interval = interval;
      TotalSteps = (numPixels() - 1) * 2;
      Color1 = color1;
      Index = 0;
    }

    // Update the Scanner Pattern
    void ScannerUpdate() {
      for (int i = 0; i < numPixels(); i++) {
        if (i == Index) {
          // Scan Pixel to the right
          setPixelColor(i, Color1);
          //setPixelColor(numPixels()-i,Color1);
        } else if (i == TotalSteps - Index) {
          // Scan Pixel to the left
          setPixelColor(i, Color1);
          //setPixelColor(numPixels()-i,Color1);
        } else {
          // Fading tail
          setPixelColor(i, DimColor(getPixelColor(i)));
          //setPixelColor(numPixels()-i, DimColor(getPixelColor(numPixels()-i)));
        }
      }
      show();
      Increment();
    }

    // Initialize for a Fade
    void Fade(uint32_t color1, uint32_t color2, uint16_t steps, uint8_t interval,
direction dir = FORWARD) {
      ActivePattern = FADE;
      Interval = interval;
      TotalSteps = steps;
      Color1 = color1;
```

```
      Color2 = color2;
      Index = 0;
      Direction = dir;
    }

    // Update the Fade Pattern
    void FadeUpdate() {
      // Calculate linear interpolation between Color1 and Color2
      // Optimise order of operations to minimize truncation error
      uint8_t red = ((Red(Color1) * (TotalSteps - Index)) + (Red(Color2) * Index))
/ TotalSteps;
      uint8_t green = ((Green(Color1) * (TotalSteps - Index)) + (Green(Color2) *
Index)) / TotalSteps;
      uint8_t blue = ((Blue(Color1) * (TotalSteps - Index)) + (Blue(Color2) *
Index)) / TotalSteps;

      ColorSet(Color(red, green, blue));
      show();
      Increment();
    }

    // Calculate 50% dimmed version of a color (used by ScannerUpdate)
    uint32_t DimColor(uint32_t color) {
      // Shift R, G and B components one bit to the right
      uint32_t dimColor = Color(Red(color) >> 1, Green(color) >> 1, Blue(color) >>
1);
      return dimColor;
    }

    // Set all pixels to a color (synchronously)
    void ColorSet(uint32_t color) {
      for (int i = 0; i < numPixels(); i++) {
        setPixelColor(i, color);
      }
      show();
    }

    // Returns the Red component of a 32-bit color
    uint8_t Red(uint32_t color) {
      return (color >> 16) & 0xFF;
    }

    // Returns the Green component of a 32-bit color
    uint8_t Green(uint32_t color) {
      return (color >> 8) & 0xFF;
    }

    // Returns the Blue component of a 32-bit color
    uint8_t Blue(uint32_t color) {
      return color & 0xFF;
    }

    // Input a value 0 to 255 to get a color value.
    // The colours are a transition r - g - b - back to r.
    uint32_t Wheel(byte WheelPos) {
      WheelPos = 255 - WheelPos;
      if (WheelPos < 85) {
        return Color(255 - WheelPos * 3, 0, WheelPos * 3);
      } else if (WheelPos < 170) {
        WheelPos -= 85;
        return Color(0, WheelPos * 3, 255 - WheelPos * 3);
      } else {
        WheelPos -= 170;
        return Color(WheelPos * 3, 255 - WheelPos * 3, 0);
      }
    }
};
```

```
void StripComplete();

NeoPatterns Strip(NUM_LEDS, PIN, NEO_GRB + NEO_KHZ800, &StripComplete);

// ---- Setup for the pulse sensor on this Arduino ---- //
#define USE_ARDUINO_INTERRUPTS true

#include <PulseSensorPlayground.h>
const int PULSE_SENSOR_COUNT = NUM_GLOBAL_SENSORS; // number of senors on this
arduino

#define ARRAY_SIZE 10 // Calculate a new average every ... number of samples
#define AVERAGE_REFRESH 40 // calculate the HRV over a number of ... averages
// HRV is thus over the past 400 heart beats
// gives ~5 minutes (if average 80bpm), necessary for SDANN

const int PULSE_INPUT0 = A0; // A0 is the sensor pin
const int PULSE_BLINK0 = 13; // LED on this pin blinks with heartbeat

const int PULSE_INPUT1 = A1;
const int PULSE_BLINK1 = 12;

const int PULSE_INPUT2 = A2;
const int PULSE_BLINK2 = 11;

const int PULSE_INPUT3 = A3;
const int PULSE_BLINK3 = 10;

const int PULSE_INPUT4 = A4;
const int PULSE_BLINK4 = 9;

const int THRESHOLD = 500;   // Adjust this number to avoid noise when idle
const int OUTPUT_TYPE = SERIAL_PLOTTER;

PulseSensorPlayground pulseSensor(PULSE_SENSOR_COUNT);

// no two-dimensional arrays, so define a separate one for each sensor
float inputData0[ARRAY_SIZE];
float inputData1[ARRAY_SIZE];
float inputData2[ARRAY_SIZE];
float inputData3[ARRAY_SIZE];
float inputData4[ARRAY_SIZE];

float averageData0[AVERAGE_REFRESH];
float averageData1[AVERAGE_REFRESH];
float averageData2[AVERAGE_REFRESH];
float averageData3[AVERAGE_REFRESH];
float averageData4[AVERAGE_REFRESH];

int outputData[NUM_GLOBAL_SENSORS];
boolean newData[NUM_GLOBAL_SENSORS];
char headers[NUM_GLOBAL_SENSORS];

int arrayCounter[PULSE_SENSOR_COUNT];
int averageCounter[PULSE_SENSOR_COUNT];
boolean newAvg[PULSE_SENSOR_COUNT];
boolean arrayFilled[PULSE_SENSOR_COUNT];

// following variables are used for void compareData
int buttonState = 0;
#define BUTTON_PIN 4
boolean buttonPress = false;
boolean visOn = false;

// used for void compareData
#define COMPARE_NUM 30 // number of samples to compare the average standard
deviation on
unsigned long timer;
```

```
unsigned int timerThreshold = 1000; // how many ms between samples for caculating
the average standard deviation
float comparison[COMPARE_NUM];
int comparisonIndex = 0;
int averageThreshold = 37; // threshold for the visualization. Average standard
deviation over the last 30 seconds.

void setup() {
  Serial.begin(9600); // send data at 9600 baud

  pulseSensor.analogInput(PULSE_INPUT0, 0); // set sensor pin as input
  pulseSensor.blinkOnPulse(PULSE_BLINK0, 0); // set LED pin to blink with heartbeat

  pulseSensor.analogInput(PULSE_INPUT1, 1);
  pulseSensor.blinkOnPulse(PULSE_BLINK1, 1);

  pulseSensor.analogInput(PULSE_INPUT2, 2);
  pulseSensor.blinkOnPulse(PULSE_BLINK2, 2);

  pulseSensor.analogInput(PULSE_INPUT3, 3);
  pulseSensor.blinkOnPulse(PULSE_BLINK3, 3);

  pulseSensor.analogInput(PULSE_INPUT4, 4);
  pulseSensor.blinkOnPulse(PULSE_BLINK4, 4);

  pulseSensor.setSerial(Serial);
  pulseSensor.setOutputType(OUTPUT_TYPE);
  pulseSensor.setThreshold(THRESHOLD, 0);

  if (!pulseSensor.begin()) {
    // sketch does not work with this version of the Arduino (interrupts)
    for (;;) {
      // Flash the led to show things didn't work.
      digitalWrite(PULSE_BLINK0, LOW);
      delay(50);
      digitalWrite(PULSE_BLINK0, HIGH);
      delay(50);
    }
  }

  // a lot of variable value initializations below

  int i;
  for (i = 0; i < ARRAY_SIZE; i++) {
    inputData0[i] = 0; // fill the sensor array
    inputData1[i] = 0; // fill the sensor array
    inputData2[i] = 0; // fill the sensor array
    inputData3[i] = 0; // fill the sensor array
    inputData4[i] = 0; // fill the sensor array
  }
  int j;
  for (j = 0; j < NUM_GLOBAL_SENSORS; j++) {
    outputData[j] = 0; // fill the output array
    newData[j] = false;
    headers[j] = 'A' + j; // characters A and on for headers
  }
  int k;
  for (k = 0; k < PULSE_SENSOR_COUNT; k++) {
    arrayCounter[k] = 0;
    averageCounter[k] = 0;
    newAvg[k] = false;
    arrayFilled[k] = false;
  }

  int l;
  for (l = 0; l < COMPARE_NUM; l++) {
    comparison[l] = 0;
  }
```

```
  int m;
  for (m = 0; m < AVERAGE_REFRESH; m++) {
    averageData0[m] = 0;
    averageData1[m] = 0;
    averageData2[m] = 0;
    averageData3[m] = 0;
    averageData4[m] = 0;
  }

    Strip.begin(); // initialize NeoPixel
    Strip.ColorSet(Strip.Color(0, 0, 0)); // reset NeoPixel
    pinMode(BUTTON_PIN, INPUT);

    Strip.Scanner(Strip.Color(0, 30, 255), 75);
}

void loop() {
  // get new values for each sensor
  refreshArray(inputData0, 0);
  refreshArray(inputData1, 1);
  refreshArray(inputData2, 2);
  refreshArray(inputData3, 3);
  refreshArray(inputData4, 4);

  getHRV(inputData0, averageData0, 0);
  getHRV(inputData1, averageData1, 1);
  getHRV(inputData2, averageData2, 2);
  getHRV(inputData3, averageData3, 3);
  getHRV(inputData4, averageData4, 4);

  int j;
  for (j = 0; j < NUM_GLOBAL_SENSORS; j++) {
    if (newAvg[j]) {
      Serial.print(headers[j]);
      Serial.println(outputData[j]); // output data to Serial for the Processing
graphwriter
      newAvg[j] = false;
    }
  }

  if (visOn) {
    // start moving visualization
    Strip.Update();
  }

  checkButton(); // See if the button state has changed
  compareData(); // Use for visualization based on amount of sync, define
averageThreshold beforehand

}

float average(float Data[], int index) {
  // calculate the average of an array of ints.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
    sum += Data[i];
  }
  float average = sum / index;
  return average;
}

float stDeviation(float Data[], int index, float average) {
  // requires both a data array and the mean of this array as input.
  int i;
  float sum = 0;
  for (i = 0; i < index; i++) {
```

```
    // sum of squared mean differences
    sum += (Data[i] - average) * (Data[i] - average);
  }
  float variance = sum / index;
  float stdeviation = sqrt(variance);
  return stdeviation;
}

void refreshArray(float Data[], int sensor_ID) {

  if (arrayCounter[sensor_ID] == ARRAY_SIZE) {
    // reset input array index, and calculate new average in getHRV()
    arrayCounter[sensor_ID] = 0;
    arrayFilled[sensor_ID] = true;
  }

  if (pulseSensor.sawStartOfBeat(sensor_ID)) {
    int datapoint = pulseSensor.getInterBeatIntervalMs(sensor_ID); // get IBI in
milliseconds
    Data[arrayCounter[sensor_ID]] = datapoint / 1000; // convert to seconds for
easier calculations, no variable overflow
    arrayCounter[sensor_ID]++; // increment input array index
  }
}

void getHRV(float Data[], float Average[], int sensor_ID) {
  if (arrayFilled[sensor_ID]) {
    if (averageCounter[sensor_ID] == AVERAGE_REFRESH) {
      // set index back to 0, to create sliding window
      averageCounter[sensor_ID] = 0;
    }

    float aver = average(Data, ARRAY_SIZE); //SDANN

    Average[averageCounter[sensor_ID]] = aver;

    float averaver = average(Average, AVERAGE_REFRESH); // average of averages,
necessary for the standard deviation
    float stdevnew = stDeviation(Average, AVERAGE_REFRESH, averaver); // get
standard deviation of the averages, which is the HRV

    outputData[sensor_ID] = int(stdevnew * 1000); // convert back to integers,
*1000 for graphwriter

    /* //RMSSD, possibly use instead of SDANN, seems a lot less consistent
      int sum = 0; // RMSSD
      for(int i=0;i<ARRAY_SIZE;i++) {
      if(i != ARRAY_SIZE - 1) {
        sum += (Data[i] - Data[i-1]) * (Data[i] - Data[i-1]);
      } else {
        sum += (Data[i] - Data[0]) * (Data[i] - Data[0]);
      }
      }
      outputData[sensor_ID] = sqrt(sum/ARRAY_SIZE);
      /*
      int aver = average(Data, ARRAY_SIZE);
      int sdev = stDeviation(Data, ARRAY_SIZE, aver);

      outputData[sensor_ID] = sdev;
    */
    averageCounter[sensor_ID]++;
    newAvg[sensor_ID] = true;
    arrayFilled[sensor_ID] = false;
  }
}

void compareData() {
  if (millis() > timer + timerThreshold) {
```

```
    // compare data every second
    timer = millis();
    int i;
    float tempCompare[NUM_GLOBAL_SENSORS];
    float aver;
    for (i = 0; i < NUM_GLOBAL_SENSORS; i++) {
      tempCompare[i] = outputData[i];
    }
    aver = average(tempCompare, NUM_GLOBAL_SENSORS);
    comparison[comparisonIndex] = stDeviation(tempCompare, NUM_GLOBAL_SENSORS,
aver);
    if (comparisonIndex < COMPARE_NUM) {
      comparisonIndex++;
    } else {
      comparisonIndex = 0; // set index to 0
    }

    if (millis() > (timerThreshold * COMPARE_NUM)) {
      // wait 30 seconds before visualizing anything
      int j;
      float prevCompared;
      float compareAvg = average(comparison, COMPARE_NUM);

      int compared = int(compareAvg);
      Serial.print('F');
      Serial.println(compared); // Serial print average stdev for analysis purposes
      if (compared < averageThreshold ) {
        visOn = true; // turn on the visualization, only use this if not using the
button
      } else {
        visOn = false; // turn off the visualization
      }
    }
  }
}

void checkButton() {
  // check and update button state and visualization state
  // to be used for manually activiting the visualization
  buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == HIGH && !buttonPress && !visOn) {
    visOn = true; // turn visualization on
  } else if (buttonState == HIGH && !buttonPress && visOn) {
    visOn = false; // turn visualization off
  }

  buttonPress = buttonState;
}

void StripComplete() {
  // To use for NeoPixel visualizations; currently unused
  // Strip.Color1 = Strip.Color(0,0,0);
  // Strip.Reverse();
}
```