The Coordination Table: Augmented Furniture to Read Rapport in Dyadic Interaction

Nicholas Masso
Honors Thesis in Psychology and Interactive Media Studies
Miami University
Oxford, OH
April 2011
Abstract

In the field of augmented furniture, many pieces of furniture use technology to comment on human interaction. One piece, Nikolovska’s Conversation Table, uses microphone and LEDs to comment on dominance in conversation. This paper develops this notion by incorporating psychological research to create a furniture set that can read subtleties in dyadic interaction. This Coordination Table is a table and two chairs that use sensors to read the posture of each person. A connected laptop analyzes this data in realtime to calculate how similar the users’ posture is. According to psychological research, people with similar posture, especially over time, tend to have higher rapport. The Coordination Table uses its data to estimate the rapport currently experienced by the dyad.

This paper outlines the software and technology used to construct a prototype Coordination Table. The table was also informally tested to prove that it could detect differences in posture due to higher or lower rapport. Although it has not been evaluated to work exactly as expected, it was shown to detect differences between users engaging in different activities. The author also discusses potential applications for the Coordination Table technology, including its usefulness for psychological research, its potential utility in performance art, and its potential feasibility as a commercial product.
Acknowledgements

This project could never have happened without the generous support of my advisors Jacob Tonski and Leonard “Jay” Smart, as well as my reader John Humphries. Thank you for being both excited for my project and critical of it as well.

I owe an enormous thanks to Liz Miller, for her patience and willingness to help me with testing, writing, and most of all my emotional health. I could not have completed this project without your support.

Special thanks also to Spencer Pate, Dana Cretu, Emma Cooper and Lauren Romano for helping me with final testing.

Thanks also to the Office for the Advancement of Research and Scholarship for their financial support of my project via the Undergraduate Research Award. I hope my projects can help continue Miami University’s commitment to research.

Lastly, I thank my parents and brother for their continued support of my unorthodox approach to education and my odd projects. There are so many small and big things you’ve done that have gotten me to where I am today.
Table of Contents

Introduction ........................................................................................................................................ 1
Design .............................................................................................................................................. 7
Evaluation ....................................................................................................................................... 19
Future Development ....................................................................................................................... 22
Conclusion ....................................................................................................................................... 28
References ....................................................................................................................................... 29
Appendix A: Software Source Code ............................................................................................... 31
List of Figures

Figure 1. FFT Comparison ................................................................. 12
Figure 2. Incoming Data Monitor ..................................................... 15
Figure 3. Rapport Graph ................................................................. 16
Figure 4. Comparison of cross-correlation results by condition ............. 21
List of Illustrations

Illustration 1. Ikea Robotics ................................................................. 1
Illustration 2. The Conversation Table ................................................ 2
Illustration 3. Load sensor ................................................................. 7
Illustration 4. Circuitry for one chair .................................................. 8
Illustration 5. Chair with circuity housing ......................................... 9
Illustration 6. Sensor housing design ............................................... 10
Illustration 7. Mahogany and pine .................................................. 10
Illustration 8. Line Output Examples ............................................... 17
Illustration 9. The Coordination Table ............................................ 18
Introduction

Augmented Furniture

We humans have a unique relationship to furniture. Pieces of furniture are some of the only material items that are made to physically complement us, without overwhelming or controlling us. It is because of this symbiosis that we can view furniture as both a place for ourselves, as well as a person in its own right. Old leather armchairs remind us of grandfathers, while love seats conjure images of couples cuddling together. Furniture has been around about as long as humans, but like everything else in today’s increasingly interactive world, furniture is getting an upgrade. This resulting new field, augmented furniture, presents exciting opportunities to integrate interaction design into mundane life.

A great example of interactive furniture is Adam Lassy’s Ikea Robotics, in which he uses electronics to animate Ikea furniture, imbuing them with personality (Kuang, 2010). His chairs rush forward in anticipation whenever someone is near and his tables will desert you unless you constantly press a button on its surface to let it know that you still care. An exploratory study of augmented furniture reveals that there are many ways that furniture can be interactive (Ackerman & Nikolovska, 2009). Ackerman and Nikolovska identify three dimensions by which augmented furniture can be organized: functional, relational, and poetic. The present project focuses on the poetic axis, which describes the different characterizations furniture can be designed to portray. The character of this

Illustration 1. Ikea Robotics
furniture is most evident in small moments, where the object captures the user’s attention through unexpected behavior that reveals an inner character. Furniture within the mediator area of this axis embodies “…the psychoanalyst’s eye…if one pays attention, this table reveals otherwise unspoken aspects of the dynamics of interaction between its users.” (p 162, Nikolovska and Ackerman, 2009).

A potent example within this area is Nikolovska’s *Conversation Table*, a two person table which uses LEDs to indicate who is talking (Nikolovska, 2006). Between the two ends of the table are a row of LEDs. When one person talks, the LEDs animate a moving point that continuously travels from the speaker to the listener. “The goal of the project was to offer a commentary on power dynamics as they occur during conversations” (p 56, Nikolovska, 2006). Nikolovska documents the sudden hyper self-awareness that people experienced in interactions where the table was turned on in the middle of a conversation. Suddenly confronted by the undeniable representation of their speaking patterns, they couldn’t help but interact with the table as much as they did with each other. Nikolovska’s creation responds to human behavior and creates a conversation around its reactions.

Illustration 2. The *Conversation Table*

However, the *Conversation Table* relies on a simplistic interpretation of human interaction. First, it only responds to talking. No other factors of the interaction are incorporated in the table’s interpretation of the conversation. Second, the table employs a binary view of human interaction. At the *Conversation Table*, you are either dominating
the conversation or you are being dominated in the conversation. There are no degrees of power. Third, the Conversation Table doesn’t respond to anything the users can’t experience directly. The table is clearly an engaging commentator on conversation, but it doesn’t comment on anything the participants can’t already experience directly. These aspects of the table limit it’s ability to understand human interaction and by extension its ability to contribute to the conversation. Much like a good psychoanalyst, mediating furniture should understand the complexities of human interaction, its nuanced nature, and be able to contribute something to the conversation that neither party could have readily noticed. Mediating furniture should be able to communicate a depth of analysis.

Rapport

To understand how this can be accomplished, an investigation of psychological research in nonverbal and verbal behavior is warranted. The first issue is determining which general indicator the furniture should monitor. There is a growing body of research within psychology that relates various social cues to a measure of rapport, which “reflects the combined effects of the interactants’ mutual attention, mutual positivity, and coordination. Increased rapport should be indicative of better functioning dyads or, at the very least, more positive relationships.” (pg 454, Grahe & Sherman, 2007).

Vocal Characteristics and Rapport

Researchers have found that many vocal characteristics of speech correlate with rapport between speakers. As people engage in conversation, their manner of speaking
converges and becomes more similar. Natale has found a link between speakers’ vocal intensity over time as compared to change in their fellow converser (1975). In a conversation between a confederate and a participant, participants would change their vocal intensity to match that of their speaking partner. Even if the confederate changed their vocal intensity in the middle of the conversation, participants would adjust their intensity to match. Additionally, vocal intensity was found to converge more and more over time, showing that convergence of vocal intensity is also a function of intimacy of the speakers. Similar evidence has been found for speaking rate in the context of interviews (Street, 1984), as well as the pausing frequency between two adults in conversation (Capella and Planalp, 1981). All of this research examines dyadic interaction, where there are only two people involved. LaFrance observed group dynamics, connecting speech to posture (1982). After observing a teacher and her class, LaFrance found that students who found their teacher to be an engaging speaker were significantly more likely to also mirror their teacher’s posture. Posture’s relation to rapport is another well-developed area of research.

Posture and Rapport

Posture is an especially appropriate field for augmented furniture to draw from. Furniture is intrinsically linked to posture, as it is tied to how people physically relate to their furniture. Furthermore, posture is actually a dynamic process where being completely static makes basic tasks (like visual inspections and breathing) more difficult (Shockley, Santana & Fowler, 2003). Small (and at times larger) swaying is an adaptive mechanism for the human body. Thus, posture has the potential to provide dynamic, live
data that reflects subtle aspects of interpersonal interaction. There is also a growing body of research relating postural mimicry to interpersonal interaction, creating a rich resource to draw from for guidance in this project (for a review, see Chartrand & van Baaren, 2009).

Some of the first research that relates nonverbal cues to rapport comes from Robert Scheflen (1964). He was the first to argue that people within a group who have similar opinions tend to adopt similar postures. He identified two different kinds of posture sharing: congruency and mirroring. Both involve imitating another person’s posture, but congruency means that the individual adopts another’s posture exactly, while mirroring means they do the same, but mirror the posture, such that their left side is the same as their partner’s right side (see figure 1). LaFrance tested these notions in the same study mentioned above (1982). The relationship between postural congruency and rapport was slightly positive, though nonsignificant. However, LaFrance found a strong positive correlation between students’ postural mirroring of their teacher and their level of rapport with their teacher. This finding is best explained by Bavelas et al:

“the primary function of motor mimicry must be communicative and that any relationship to vicarious processes is secondary. A similar analysis of other nonverbal behaviors may well reveal that they are also expressions to another person rather than expressions of intrapsychic states.” (pg 275, 1988).

People unconsciously use motor mimicry to tell others how they feel about the relationship. On some level, a choice is made to show their evaluations of others.

Other researchers have studied postural synchrony in dyadic conversations. Charney recorded postural information from therapy sessions and compared it with the
verbal content of their conversation to see what kinds of language were used in tandem with different levels of postural synchrony (1966). He found that when clients’ posture was unlike their therapist’s posture, they rarely mentioned others, including the therapist, and were often more ambiguous in their speech. But, when they mirrored the therapist’s posture, they actively mentioned others, talked about the therapist-client relationship, and was clearer in thought and speech. Postural mirroring was found to be correlated with effective therapeutic interaction, another form of rapport. Bernieri et al. found significant correlational between postural synchrony and resolution of arguments (1996). Participants were placed in pairs and asked to complete two tasks together while being recorded. The first task was competitive (a debate) and the second was cooperative (planning a vacation). In both tasks, there was a significant positive correlation ($r = .35, p < .05$ and $r = .5, p < .001$, respectively) between postural synchrony of the pair and their ability to make consensual judgments. In a wide range of contexts, postural synchrony is a reliable predictor for rapport in dyads.

With these findings in mind, it is the author’s intent to create a furniture set of a table and two chairs that uses integrated sensors to monitor vocal and postural characteristics that have been linked to rapport. From these, it will calculate a general level of rapport between the users in real time. It will be called the *Coordination Table*, after the experiment used to test the table’s accuracy.
Design

The Coordination Table is a combination of furniture design, circuit design, and software design. The Coordination Table has embedded sensors that report to microcontrollers, one set per chair. The microcontrollers send data to a laptop that analyzes the data and displays the results on screen. The laptop software was developed in the Processing IDE, while the Firmata library was used to interface that software with the Arduino microcontrollers. Approximately every 1/22 of a second, the software read the integrated sensors and performed its analysis. The realtime nature of this solution makes it different than anything I was able to find in prior research. All methodologies in past psychological research performed data analysis post-hoc, usually on video tape recordings or raw positional data stored from sensors.

Electronics

The sensors include two load sensors in each chair and two microphones mounted in the table. The load sensors are mounted beneath the two back feet of each chair. Initially, I tried mounting the sensors between the seat of the chair and the legs, but the seat tended to flex under the sitter’s weight, taking too much load off of the sensors. Placing them under the feet (and letting the seat lay flat on the chair’s structure) made sure they received the sitter’s weight. It is also more comfortable—the slight incline from the sensors is much more noticeable when mounted directly beneath the chair set. A microphone is embedded in

Illustration 3. Load sensor
each side of the table where a person would sit. The microphones are Pyle-Pro PLM3 mono-directional lavaliers connected directly to the computer and were interfaced via the Minim library for Processing. They were chosen for their small form factor, making them easy to embed, as well as their limited pickup range, which ensured that it would only pickup sounds that were part of the conversation and hardly any ambient noise. They are also mono-directional, meaning that they only pick up sound from one general direction, which allows each microphone to only record the sounds of one person.

The load sensors are 50 kg load sensors from an unknown manufacturer connected to an Adafruit BoArduino microcontroller. They required the use of amplifiers to create a signal range detectable by the microcontroller. This is because the load sensor is a strain gauge, meaning that it relies on the slight bending of its metal clip to create a change in resistance. Under a full load, the gauge only produces a voltage drop of about 5 mV, which is the smallest amount that the arduino microcontroller can detect. This makes the load sensor behave like a digital on-off switch, which doesn't provide much information, especially when we need to sense subtler shifts in weight. Operational amplifiers can take a small difference in voltage and amplify it into large difference, which gives the microcontroller much better resolution in the weight sensor information. I used the INA125, an operational amplifier from Burr-Brown, in a non-inverting difference amplifier.
A difference amplifier compares two incoming voltages and amplifies their difference. This is accomplished with positive feedback, where current is run from the amplifier’s output back to the positive input with a resistor called a gain resistor. The resistance of gain resistor determines the gain of the amplifier, or by what factor it amplifies the difference voltage. As more electricity flows out of the amplifier, more is fed back into the positive input, thereby making the difference larger and larger. It amplified the load sensor’s range to about 3.5 V, providing a scale with a decent resolution that the arduino can use. The arduino reads these analog load sensors and sends that information to the Processing sketch on the laptop via USB.

Illustration 5. Chair with circuitry housing

Furniture and Sensor Housing

A table and two chairs were chosen to encourage dyadic interaction. Chairs afford a resting place for people to linger, while the table affords a space in which interaction can happen. Two chairs can be placed in any number of configurations, but a table and two chairs only work a few different ways, all of which place sitters within an optimal distance to hold a conversation.
I designed housings to hold the pressure sensors in place and hide them from view. When having a natural conversation is an integral part of this project’s concept, making the technology blend into the its context is as important as the technology itself. Since the best place for the sensors was not beneath the chair’s seat, I decided to abandon the idea of totally integrated sensors and develop a more modular design that fit around each of the chair’s feet. I began thinking of the sensor housings as boots for the chair, and the remnants of a boot buckle and tongue can still be found in the model to the right. However, I still used the IKEA chair as a guide for the design, for practicality’s sake and to have some constraints for the design. I chose pine and mahogany wood (see illustration 7) for the design to match the pine chairs and provide a nice duotone to differentiate the boot from its chair. The load sensor is attached to the bottom on the inside of the chair boot, the boot guiding the chair leg to rest squarely on it. The cut out section allows the chair feet crossmember to fit and the thin vertical cut to the base allows the load sensor’s wires to connect to the BoArduino. This same piece is used for the back two feet of the chair, while shorter placeholders are used on the front to keep the legs on equally high grounding. This also ensures that the front
two legs do not bear the majority of the weight.

Software

The basic software architecture is composed of two objects, and a few global functions that handle the order of analyses and output. The objects are of a class Person, written to collect, store and analyze the data for each person at the table. There are two Persons in this version of the program for two people: personL and personR. Each chair object contains all of the pressure and audio data for its user and the indicators calculated from that data. The program runs a collect, store and analyze cycle about 22 times per second. Each cycle includes sensor reading and three stages of analysis for each chair. Each Person produces a momentary rapport rating, which are factored into a global rapport level with an easing function. This stage method allows the sensor reading to be as simultaneous as possible and allows the Persons’ separate analyses to reference values from each other’s earlier analyses from within the cycle. I will describe this cycle, but those looking for a more rigorous account can see the source code in Appendix A.

Stage 1

Every cycle begins with personR and personL taking sensor readings from their respective load sensors and microphones, and storing them. Because the load sensors can be rather noisy, the method actually takes two readings every cycle and averages them together. The software stores the most recent 128 sensor readings for
each sensor, which means that it stores about the last 6 seconds of data. The Persons use a two-dimensional array, making it easy to expand it to accommodate extra sensors in the future. After storing the data, the software analyzes each pressure sensor’s array of data with a Fast Fourier Transform (FFT) from the Minim library. The Fast Fourier Transform transforms the time series data into frequency data. It determines how much spectral energy falls within certain frequencies of oscillation, and creates an array of values that measure how much energy is within each area, or bin. For an example, see figure 1, where there are two graphs of sensor data with its FFT spectrum graphed over it. Sensor one has fast oscillations, there is a red spike higher in the FFT data. Sensor two has a slow oscillation, so the spike in the FFT array is much lower. The FFT data allow us to understand at what speed people are shifting weight. Low frequencies of shifting produce low values in the FFT, while high frequencies produce high values.

Next, the software calculates a couple different values from the sensor and FFT data. First, it calculates the total between each chair’s sensors (totalLR), giving an indication of the total amount of weight on the chair. This value is only used to calculate rapport indirectly. Second, it calculates what percentage of that total weight is placed on the left sensor (propL). Absolute values could not be used because both speakers
would have to have the same weight to ever be found as having rapport by the
software. This concern shaped many of the indicators chosen for the software. Using
absolute numbers would mean that someone weighing 200 pounds could never have
indicators similar to someone weighing 100 pounds, and vice versa. I use percentages
because they are purely relative and therefore can find similarities between people of
different weights. Third, each Person calculates the change between the current read
value and the preceding read value for propL and totalLR and stores them in
propLDer and totalLRDer, respectively. Fourth, each Person finds the FFT band
with the highest value, but every FFT band that is not above a noise threshold (about
50) is clamped to 0 to protect them from noise.

Stage 2

This stage is where each Person determines when their data are most similar.

For each of the values calculated in the previous stage by its opposite, each Person
searches its stored array of values over the last two seconds to find the closest match.
For example, PersonL searches in its data buffers to find the value that is closest to the
value just calculated by PersonR. It stores the index values for where that data is
located. It compares the current value of the partner to every value it has had for the
last two second and remembers at which point they were closest. This was inspired by
a close read of the methodologies used in past research (LaFrance, 1986; Bernieri et al,
1996; Shockley et al, 2003). In LaFrance’s studies, human observers coded posture at
regular intervals, first observing a single person, then scanning the room to see how
many others had similar characteristics. This process undoubtedly took at least a
second or two to accomplish. Therefore, it made sense to work the same method into
the software, since much of the research it draws from employed similar methods.
Additionally, the software comparisons measure along the range of identical to
completely different. Past encoding methods couldn’t measure the degree of similarity in
posture as precisely as a set of sensors can. In order for this project to accurately read
human behavior, it has to have some flexibility built in to its reading. Similar
mathematical methods are employed at a more sophisticated level by Shockley et al
(2003). They utilize magnetic tracking devices that read positional data at 60 Hz,
analyze that data with a method that looks for curve similarities over time. This process
involves looking for delayed similarities in the data. If one person leans to the left, and
the other mirrors that movement half a second later, that should be recognized.
Furthermore, Dr. Tanya Chartrand’s research lab employs a three-to-five second rule
when they analyze videos for mimicry (personal communication, 2011). If a similar
behavior occurs within three to five seconds or less of another behavior, it is coded as
mimicry. In order to capture this aspect of mirroring, I created this second stage of
analysis to tease out the similarities that a strict analysis wouldn’t find. So, each chair
knows at which points in time it was most similar to the other chair on each factor within
the last couple seconds.

For audio, the Persons compare microphone levels separately to determine
which person is speaking. Whichever side has higher levels is assumed to be speaking,
unless the highest level is below a noise threshold. In that case, the software considers
the users to be in a moment of silence. Whenever the current speaker is not the same
as the speaker of last cycle (including silence as a third speaker) the function calculates
the time spent by the last speaker speaking (or of neither speaker speaking) and stores it in an array. Then, it calculates the average length of speech for each Person, and the average length of silence, as well as the total deviance from each data set's respective mean. This invariance of speech and sound is used to compare matching in the length of silences between speakers (response latency) as well as rate of speech.

Stage 3

The third stage of analysis is a comparison of those values found to be most similar over the past two seconds. The difference between these values are computed and factored into a total rating of rapport at that moment. This function maps the absolute value of every difference to a 100 point range. Then, each of these are multiplied by some fraction and added together. These fraction multipliers all add up to 1. This is because the final sum of factors for each Person must fall within a 100 point range, just like the individual differences. For example, if every individual difference was at the top of the 100 point range, the only way to factor those together so that they do not exceed the range limit is to multiply them by a set of factors that equal 1. These factors become weightings that represent the importance of individual indicators to the rapport rating. Differences

Figure 2. Incoming Data Monitor
Left side is one chair, right is the other. Top graph shows overall rapport ratings
calculated from the FFT are weighted more because it captures so many aspects of movement. If two people are swaying at a similar frequency, that actually encompasses other factors that are also computed, like changes in weight distribution and the rate of that change. At the end of this stage, each Person has computed a summary rapport rating for itself, falling within a 100 point range.

Audio is computed differently. The total deviance values are mapped into a 100 point range. The absolute difference between the average speech length for personL and personR is found and mapped as well.

Stage 4

The fourth and final stage of analysis calculates the global rapport level for the interaction. It is not contained within the Person objects—it is a global function. It first averages the summary rapport rating from each chair object together, as well as the audio rapport rating. Then it factors this average into the global rapport level with an easing function:

\[
\text{rapportLevel}[0] = \text{rapportLevel}[1]*0.97 + \text{rapportMoment}[0]*.03;
\]
Each cycle, the global rapport level is only influenced a slight amount by the momentary rapport rating. However, since this cycle is completed twenty-two times a second, the global rapport level is actually fairly responsive to sustained changes in the momentary rapport rating. This final value is the culmination of the software’s computation. But the question remains as to what the Coordination Table should do with this computation.

Output

A simple interaction method was developed to demonstrate the Coordination Table’s functionality. The table was programmed to act as a quiet listener who entered the conversation only intermittently. During conversation, the laptop screen remained blank. After 30-45 seconds, it would slowly fade in a graphic of straight or erratic lines. The line composition was based on the average rapport rating found by the table during that 30-45 seconds of conversation. High ratings of rapport produce straight horizontal lines, suggesting the harmony found in the interaction. Low ratings of rapport produce erratic zig-zags, suggesting a jarring sense of conflict in the interaction. After the graphic faded in, the users could press a button on the keyboard to reset the table and restart the cycle.

This method of interaction was chosen first and
foremost designed to keep the table from becoming the center of attention in interaction. In preliminary testing, we noticed that people would change how they sat drastically if they could see it affecting on screen graphics in realtime. Even something as simple and uninspired as a realtime bar graph almost became a game for users. Nikolovska found similar results in her user evaluations of the Conversation Table (2006). Participants conversation became entirely about the table, forgetting to discuss almost anything they had been discussing before the table was turned on. The simplistic output of lines was chosen to keep the table from being too explicit in its output. The lines were still heavily influenced by the ratings of rapport, but they don’t judge the interaction, nor do they label it. Instead, they provide a visual reflection of the interaction to make users reflect on it themselves.

Admittedly, this interaction is fairly simple and doesn’t take advantage of the Coordination Table’s full potential. Issues in and ideas for further development will be discussed later.

Illustration 9. The Coordination Table
Evaluation

An informal experiment was run to test the ability of the table to read levels of rapport. The methodology of a similar study done by Kevin Shockley was adapted slightly for this purpose (2005). Two participants who were friends of the author volunteered to participate in the study. First, the participants were thanked for their time, asked to sit at the Coordination Table, and told they would be asked to solve a series of visual puzzles. “Can you spot the differences?” puzzles were used. They consist of two very similar graphics that people must compare to find subtle differences. First, each participant was given their own puzzle and told to find as many differences as they could within two minutes, working by themselves. After this task, participants were each given half of the visual puzzle. They were told that they were not allowed to look at their partner’s piece and instead had to talk to each other to find the differences in the same time allotment. The second task required participants to engage and coordinate with one another to solve the task, leading to higher levels of rapport as compared to the first puzzle, where they were not allowed to interact.

During each of these two puzzles, the Coordination Table logged raw pressure data from the participants. Microphone data was not used for testing due to technical challenges in signal amplification. The key indicators used in the software were reconstructed on the data, generating values for every data point. A cross-correlation was used to analyze how the indicators from each person compared to the indicators of their partner. For example, indicator one from person one was compared to indicator one from person two, but it was not compared to indicator two or three of person two.
lag of 22 was used because this would be equal to about a second of data. This allowed the analysis to find how similar indicators varied together, even if they had delayed correlation. I would then compare the highest correlation found and at what lag it was found between the two different conditions. I predicted that indicators would have higher correlations at lower lag times during the cooperation task than compared to the independent task.

However, the results of this analysis did not support this hypothesis. The cross correlation results showed the opposite. The independent task had higher correlations with lower lag times than the coordination task. Figure 4 shows the analysis results for one of the indicators, the proportion of left weight of the total weight on the chair. The other two indicators had very similar results.

Although these results do not support the hypothesis they actually still support the validity of the Coordination Table's ability to detect changes in behavior. The analysis shows that there is a definitive difference between the indicators in one condition as compared to the same indicators in the other. This difference could only have been caused by the experimental manipulation. Therefore, the Coordination Table is able to produce different data in response to different user behavior, showing that it can perform its fundamental function. The reversal of the hypothesis in the results can be conclusively explained. With only a single pair testing the table, any individual differences present could have changed the predicted behavior.
Figure 4. Comparison of cross-correlation results by condition
Future Development

In its current state, the *Coordination Table* is far from complete. This project is interdisciplinary, which means that it borrows pieces from the agendas and aims of the different disciplines it spans. Any one of these agendas can influence the further development. I will identify the potential paths that are most compelling and outline the first steps in following that path. The only core technical challenge that remains unsolved is the microphone amplification, so this is one obvious issue to be developed in the future. These three possible directions address how different target audiences and contexts would influence the development of the *Coordination Table*.

Research Tool

As the sensory array in the *Coordination Table* was developed from psychological research, it is natural to apply it to psychological research. This notion of usefulness in research was further confirmed by posture researcher Dr. Tanya Chartrand’s independent assessment of the technologies (personal communication, 2011). The first step in this development path is the optimization of technology. Better strain gauges and amplifiers should be implemented. The strain gauges need to contain full Wheatstone Bridges to account for temperature drifting. A strain gauge that doesn’t contain a Wheatstone bridge requires a separate resistor set that must be tuned to the zero-point of the strain gauge. This zero-point drifts as the temperature of the sensor changes, sometimes requiring frequent and relatively drastic adjustments. A Wheatstone bridge integrates that function into the sensor itself and tunes itself, making it far more reliable, accurate, and user friendly. Two more strain gauges should also be
added for each chair to capture all of the possible data. The system described in this paper can only guess at forward lean based on previous pressure put on the rear sensors. This is a fairly invalid and unreliable method, prone to misreads of the situation. A strain gauge beneath each chair would add precision to the sensing of weight and its distribution. The amplifiers used only amplified the strain gauge to a range of about 3.5V. With a true rail-to-rail input and output operational amplifier, that range would stretch to 5V, giving the sensors much better resolution.

Second, the technology would have to be better integrated into the furniture. The laptop would have to be removed from the system, at least physically. The laptop was included primarily to perform calculations on the sensor readings in realtime. However, this would not be necessary for psychology researchers. Psychological research methodologies only require raw data from the sensors for later analysis (Chartrand, personal communication, 2011). There are two different approached to replacing the laptop. The first is the use of wireless communication, via either WiFi or Bluetooth technology. The Arduino microcontroller would stream its data live to a desktop computer within range. The second option is to use a data logger to record data to a removable storage device like an SD card or USB drive. In fact, Adafruit Industries sell a data logging shield for Arduinos that would fulfill this function. As for physical embedding of pressure sensors, the best method would be to place them directly into the base of the chair feet. A hollow cavity with an attachment point for the sensor in its roof would almost completely hide the sensors. The microcontroller would be housed with a battery power source in a wooden box to match that of the chair and attached somewhere beneath the seat, out of sight. The biggest challenge would be hiding the wiring.
between the sensors and the microcontroller. They would either be discreetly run on the inside of the chair legs, painted to blend in with the chair wood, or they could be embedded in channels cut into the wood and covered with wood putty. However, it's important to remember that the more embedded the sensors are, the harder it will be to replace components or perform other maintenance.

By taking the steps outlined above, the technology behind the Coordination Table could become a powerful tool for psychological research. First, it is more accurate and precise than methodologies where people encode videos of participants. Concerns about inter-rater reliability are removed and the problem of observer fatigue is avoided. Second, it is cheaper to produce than magnetic position tracking devices ( ). The vast majority of psychological research on posture and rapport has focused on free standing participants (e.g., Bernieri et al, 1996, Shockley et al, 2003, and Grahe & Sherman, 2007). A well-made, less expensive research tool to record motion might enable more researchers to tackle this problem.

**Consumer Product**

The Coordination Table could be developed into a consumer product designed to facilitate interaction by helping people understand each other and connect. The technology developed in thus far could be modified to provide each user with a visualization of their partner’s data. Users would be able to pick up on subtle nonverbal signals that they might not normally notice, especially when their partner verbal signals are do not match. They would then have the ability to ask their partner about this reading to see if they were wanting to talk about it. People would become more
emotionally attuned to one another, and thus better able to manage their relationship with the other person.

Realizing this design would require the technology to become better hidden, as described above. Except that a central computer for realtime data analysis would have to be embedded in the table. The microcontrollers would use bluetooth technology to transmit data to the laptop and it would reveal the results of its analysis to each of the users in some manner. This manner is a point of a great many possibilities that would require some prototyping and evaluating to find the best method. The results could be sent to a smartphone or tablet application. They could be sent to a wrist module with a screen, like a wrist watch. They could also be projected behind the head of each participant or they could be delivered through shielded screens protruding from the table at an angle. The indicator would need to be designed as well. Ideally it would express changes in user’s emotional or mental state easily read from far away by user’s with little experience with the table. It would have to be intuitive to read. An organic form that changed in color and shape might fulfill these criteria.

A product like this would be best suited for friends and family that were already comfortable around each other. It would help them to improve their relationship and reach a new level of interpersonal understanding. It might not be appropriate for acquaintances because it does not give the user any control over the data that their partner can see. Agreeing to use the table with another person would be a signal of trust, since the user would become vulnerable to that person. Acquaintances are not likely to exhibit that level of trust. Even if they were to agree to use the table together, the awareness of being watched might make them attempt to self-monitor excessively
to hide certain aspects of their personality. Acquaintances are also not likely to feel comfortable asking one another about deeper feelings and states of mind. It might also have useful applications in therapy for individuals and couples and so might be interesting to therapists and psychiatrists.

Performance Art

The *Coordination Table*’s technology could also be leveraged to create performance art that is heavily interactive. A modified version of the *Coordination Table* similar to the consumer product described above, with fully embedded technology, except that the central laptop would be held in another room. The microcontrollers would transmit data via bluetooth. The table would sense and compute data during the entire interaction of two people. At the end of their interaction, it would produce a final average level of rapport during the interaction and that would be delivered to the users. This delivery, as well as the table itself would be fully integrated into an everyday experience. For example, the table could be used in a restaurant and the final average rapport level would be delivered on the check. Or perhaps it might be at a coffee shop, and the final average would be shown at the bottom of the coffee cup. Whatever method is implemented, the experience should surprise and intrigue users. Whether or not the table is trusted as a true scientific authority, it can still be compelling. Fortune tellers and TV psychics are compelling to many people, despite a complete lack of scientific support to their methods.

This development path is not completely different from the consumer product described above. An experience as unusual and compelling as this might be
marketable. Restaurants, resorts, amusement parks might be interested in offering a unique interpersonal experience to their clientele, who might be willing to pay for it. The biggest challenge marketability would be helping people understand something they have probably never experienced anything like before.
Conclusion

The *Coordination Table* is in many ways a response to and refinement of Nikolovska’s *Conversation Table*. The furniture performs the same basic function, but in a more sophisticated way by incorporating multiple indicators of behavior based on psychological research. Although The *Coordination Table* did not behave as predicted in testing, it was still able to detect differences in levels of coordination.

The *Coordination Table* has many potential applications and avenues for growth in the future. With little further development and just a few tweaks, it could be used by psychologists to further study posture’s relation to human interaction. It could also be incorporated into performance art or developed into a range of consumer products designed to help people connect with each other better.

The *Coordination Table* is an example of the power and intelligence that can be woven into everyday objects with technology and a grounding in past research. It pushes the boundaries of what our furniture understands about us and what it should do about it. Hopefully, this work can motivate further exploration of interaction paradigms for augmented furniture.
References

Ackerman, E., & Nikolovska, L. (2009). Exploratory design, augmented furniture: On the importance of objects' presence. In P. Dillenbourg, J. Huang & M. Cherubini (Eds.), *Interactive artifacts and furniture supporting collaborative work and learning* (pp. 147-166) Springer Science.


Appendix A: Software Source Code

```java
import ddf.minim.*;
import ddf.minim.signals.*;
import ddf.minim.analysis.*;
import ddf.minim.effects.*;
import cc.arduino.*;
import processing.serial.*;

Arduino arduinoL;
Arduino arduinoR;
Minim minim;
AudioInput mic;
Person personL;
Person personR;
PrintWriter outL, outR;
DataDisplay[] sensorDisplays;
color fftColor, pressureValsColor, currentValColor, micColor;
PGraphics lines;

int lastReadTime;
int readsPerSec = 22;
float[] rapportLevel, rapportMoment;
float audioRapportMoment;
float testVal = 0;
int lastSpeaker, newSpeaker, currentSpeaker, pastSpeaker;
float[][] utterances;
float[][] latencies;
float[] currentSpeakerMicTotals;
float avgMicTotals;
it utterancesCutoff;
it lastUtteranceStart;
float[] recordedRapports, recordedRapportMoments;
boolean recordRapport = false, recordData = false, showFinalImage = false, showGraphs = false;
it time, n; // ref values
float avgRIntensity, avgLIntensity, averageLUtteranceLength, averageRUtteranceLength,
    latenciesAvg, latenciesSSQ;
float tintVal = 0, backgroundVal;

void setup() {
    size(800, 1000);
    fftColor = color(200, 0, 10, 150);
    pressureValsColor = color(216, 105, 0, 100);
    currentValColor = color(5, 60, 200);
    micColor = color(20, 180, 20);

```
rapportLevel = new float[256];
rapportMoment = new float[256];
for (int i = 0; i < rapportLevel.length; i++) {
    rapportLevel[i] = 0;
    rapportMoment[i] = 0;
}

// println(Serial.list());
arduinoL = new Arduino(this, Arduino.list()[0], 57600);
arduinoR = new Arduino(this, Arduino.list()[2], 57600);
for (int i = 0; i < 4; i++) {
    arduinoL.pinMode(i, arduinoL.INPUT);
    arduinoR.pinMode(i, arduinoR.INPUT);
}

minim = new Minim(this);
ic = minim.getLineIn(Minim.STEREO, 512);
rectMode(CENTER);

personL = new Person(0);
personR = new Person(1);
sensorDisplays = new DataDisplay[4];
// setupX = -width/2;
for (int i = 0; i < sensorDisplays.length; i++) {
    sensorDisplays[i] = new DataDisplay();
    setupCounter++;
    setupIndex *= -1;
}

utterances = new float[audioAnalysisScope][3]; // [0] length [1] timestamp
latencies = new float[audioAnalysisScope][2]; // length and timestamp
recordedRapports = new float[0];
recordedRapportMoments = new float[0];
currentSpeakerMicTotals = new float[0];

outL = createWriter("leftData.txt");
outR = createWriter("rightData.txt");

void draw() {
    if (showFinalImage == true && 
        90 - abs(map(cos(backgroundVal), -1, 1, 40, 200)) < .5) {
        tintVal += .25;
        background(90);
        tint(255, tintVal);
        image(lines, width/2, height/2);
    }
    else {
        backgroundVal += .005;
        background(map(cos(backgroundVal), -1, 1, 40, 200));
        background(90);
        time = millis();
        if (recordRapport && time - recordStartTime > presetRecordLength) {
            endRapportRecording();
        }
    }
}
if(time - lastReadTime >= (1000/readsPerSec)-5) {
    println(time - lastReadTime);
    lastReadTime = time;
    personL.readSensors();
    personR.readSensors();
    if(recordData) {
        recordDataLine();
    }
    //    println(time - lastReadTime);
}
else {
    println(time - lastReadTime);
}

if(showGraphs) {
    for (int i = 0; i < sensorDisplays.length; i++) {
        sensorDisplays[i].drawGraphs();
    }
    rapportGraph();
    //    micIndicator();
    //    output();
}

void readMic() {
    personL.micTotal = 0;
    personR.micTotal = 0;
    for (int i = 1; i < mic.mix.size()-1; i++) {
        personL.micTotal += abs(mic.left.get(i));
        personR.micTotal += abs(mic.right.get(i));
    }
}
if(personL.micTotal >= personR.micTotal) {
    newSpeaker = 0;
    currentSpeakerMicTotals = append(currentSpeakerMicTotals, personL.micTotal);
} else if(personR.micTotal >= personL.micTotal) {
    newSpeaker = 1;
    currentSpeakerMicTotals = append(currentSpeakerMicTotals, personR.micTotal);
}
if(abs(personL.micTotal+personR.micTotal) < 20) {
    newSpeaker = 2;
}
println(abs(personL.micTotal+personR.micTotal));
// println(newSpeaker);
if(newSpeaker != currentSpeaker) {
    if(currentSpeakerMicTotals.length > 0) {
        avgMicTotals = 0;
        for(int i = 1; i < currentSpeakerMicTotals.length; i++) { // 0 is the value for the new speaker
            avgMicTotals+=currentSpeakerMicTotals[i];
        }
    } else {
        avgMicTotals = 0;
        println("avgMicTotals set to 0");
    }
    if(currentSpeaker==0) {
        arrayCopy(personL.utterances, 0, personL.utterances, 1,
        personL.utterances.length-1);
        personL.utterances[0][0] = time - personL.lastUtteranceStart;
        personL.utterances[0][1] = time;
        personL.utterances[0][2] = avgMicTotals;
    } else if(currentSpeaker==1) {
        arrayCopy(personR.utterances, 0, personR.utterances, 1,
        personR.utterances.length-1);
        personR.utterances[0][0] = time - personR.lastUtteranceStart;
        personR.utterances[0][1] = time;
        personR.utterances[0][2] = avgMicTotals;
    } else if(currentSpeaker==2) {
        arrayCopy(utterances, 0, utterances, 1, utterances.length-1);
        utterances[0][0] = time - lastUtteranceStart;
        utterances[0][1] = time;
        utterances[0][2] = utterances[1][1] - utterances[1][1];
        // need to know time since last silence, not the length of the silence
    }
    if(newSpeaker==0) {
        personL.lastUtteranceStart = time;
else if(newSpeaker==1) {
    personR.lastUtteranceStart = time;
}
else if(newSpeaker==2) {
    lastUtteranceStart = time;
}
float savedVal = currentSpeakerMicTotals[0];
currentSpeakerMicTotals = new float[1];
currentSpeakerMicTotals[0] = savedVal;
if(newSpeaker != pastSpeaker && currentSpeaker==0) {
    latencies[0][0] = utterances[0][2];
    latencies[0][1] = time;
}
pastSpeaker = currentSpeaker;
currentSpeaker = newSpeaker;
//new speaker is set at the beginning of the function
}

void analyzeMicData() {
    audioRapportMoment= 0;
    //look for difference between average L utterance and average R utterance
    //FUTURE FEATURE: Calculate variance of each item from the mean of all
    //use total variance adjusted by differ means to see what the disimilarity really is
    averageLUtteranceLength = 0;
    n = 0;
    for(int i = 0; i < personL.utterances.length; i++) {
        if(time-personL.utterances[i][1] < secondsPrior*1000) {
            averageLUtteranceLength+=personL.utterances[i][0];
            n++;
        }
    }
    averageLUtteranceLength/=n;
    averageRUtteranceLength = 0;
    n = 0;
    for(int i = 0; i < personR.utterances.length; i++) {
        if(time-personR.utterances[i][1] < secondsPrior*1000) {
            averageRUtteranceLength+=personR.utterances[i][0];
            n++;
        }
    }
    averageRUtteranceLength/=n;
    float utteranceLengthSimilarity = abs(averageLUtteranceLength-
averageRUtteranceLength);
    audioRapportMoment+=map(utteranceLengthSimilarity, 0,(averageLUtteranceLength
+averageRUtteranceLength)/2,
-50,50)*.2;
    //vocal intensity(volume)
avgLIntensity = 0;
for(int i = 0; i < personL.utterances.length; i++) {
  avgLIntensity+=personL.utterances[i][2];
}
avgLIntensity/=personL.utterances.length;

avgRIntensity = 0;
for(int i = 0; i < personR.utterances.length; i++) {
  avgRIntensity+=personR.utterances[i][2];
}
avgRIntensity/=personR.utterances.length;

audioRapportMoment+= map(abs(avgRIntensity-avgLIntensity),0,abs(avgRIntensity +avgLIntensity),50,-50)*.4;

//latencies
latenciesAvg = 0;
latenciesSSQ = 0;
n = 0;
for(int i = 0; i< latencies.length; i++) {
  if(time - latencies[i][1] < secondsPrior*1000) { //FIX COMPARISON VALUE
    latenciesAvg+=latencies[i][0];
    n++;
  }
}
latenciesAvg/=n;

for(int i = 0; i < latencies.length; i++) {
  if(time - latencies[i][1] < secondsPrior*1000) {
    latenciesSSQ+=sq(latenciesAvg - latencies[i][0]);
  }
}
audioRapportMoment+=map(latenciesSSQ, 0,sq(latenciesAvg*n), 50,-50)*.2;

//see how regular pause frequency is
float averageSilenceInter = 0;
float silenceSSQ = 0;
for(int i = 0; i < utterances.length; i++) {
  averageSilenceInter+=utterances[i][2];
}
for(int i = 0; i < utterances.length; i++) {
  silenceSSQ+=sq(averageSilenceInter-utterances[i][2]);
}
audioRapportMoment+=map(silenceSSQ,0,sq(averageSilenceInter*utterances.length), -50,50)*.2;
println("arM: " + audioRapportMoment);
}

void compareData() {
  arrayCopy(rapportMoment, 0, rapportMoment, 1, rapportMoment.length-1);
  rapportMoment[0] = (personL.rapportMoment + personR.rapportMoment +
  audioRapportMoment)/3;
  arrayCopy(rapportLevel, 0, rapportLevel, 1, rapportLevel.length-1);
rapportLevel[0] = rapportLevel[1]*0.97 + 
rapportMoment[0]*.03;

if (Float.isNaN(rapportLevel[0])) {
    println("not a numbah");
    rapportLevel[0] = 0;
}

if(recordRapport) {
    recordedRapports = append(recordedRapports, rapportLevel[0]);
    recordedRapportMoments = append(recordedRapportMoments, rapportMoment[0]);
} // println("Level: " + rapportLevel[0]);

void rapportGraph() {
    noStroke();
    for(int i = 0; i < rapportLevel.length; i++) {
        fill(0,180,10,100);
        rect(i*2.5,200-map(rapportMoment[i],-50,50,0,200),i*2.5+2.5,200);
        fill(0,20,180,100);
        rect(i*2.5,200-map(rapportLevel[i],-50,50,0,200),i*2.5+2.5,200);
    }
}

void micIndicator() {
    ellipseMode(CENTER);
    noStroke();
    fill(75,200);
    if(pastSpeaker==0) {
        ellipse(0+50,200,50,50);
    } else if(pastSpeaker==1) {
        ellipse(width-50,200,50,50);
    } else if(pastSpeaker==2) {
        ellipse(width/2,200,50,50);
    }
    fill(150,200);
    if(currentSpeaker==0) {
        ellipse(0+50,200,50,50);
    } else if(currentSpeaker==1) {
        ellipse(width-50,200,50,50);
    } else if(currentSpeaker==2) {
        ellipse(width/2,200,50,50);
    }
    fill(225,200);
    if(newSpeaker==0) {
        ellipse(0+50,200,50,50);
    }
else if(newSpeaker==1) {
    ellipse(width-50,200,50,50);
}
else if(newSpeaker==2) {
    ellipse(width/2,200,50,50);
}
}

void recordDataLine() {
    outL.println(personL.directionalVals[0][0] + "," + personL.directionalVals[1][0]+ "," + personL.micTotal);
    outR.println(personR.directionalVals[0][0] + "," + personR.directionalVals[1][0]+ "," + personR.micTotal);
}

void finalImage(float input) {
    println("finalImage called " + input);
    lines = createGraphics(360,360,P2D);

    int lineX=40, lineY=40;
    int pointNum = 200;
    int zigWidth = 20;

    lines.beginDraw();
    lines.background(255);
    lines.smooth();
    lines.noFill();
    lines.strokeWeight(2);
    lines.strokeCap(SQUARE);
    float zigHeight = map(input, -40,30, 18,0);
    float randomRange = map(input, -40,30, 8,0);
    // float r[] = new float[pointNum];
    // for(int i = 0; i < r.length; i++) {
    //   r[i] = random(-randomRange,randomRange);
    // } 
    for(int i = 0; i < pointNum; i++) {
        println(i);
        lines.line(lineX-zigWidth,lineY+zigHeight+random(-randomRange,randomRange),
        lineX,lineY-zigHeight+random(-randomRange,randomRange));
        //   line(lineX-zigWidth,lineY+zigHeight+r[i],
        //   lineX,lineY-zigHeight+r[i+1]);
        zigHeight*=-1;
        lineX+=zigWidth;
        if(lineX > lines.width-20) {
            lineX=40;
            lineY+=40;
            //   zigHeight*=-1;
        }
        if(lineY >= lines.height-30) {
            i=pointNum;
            // return;
        }
    }
}
lines.endDraw();
showFinalImage = true;
println("should I show final image: "+showFinalImage);

imageMode(CENTER);
}

void output() {
background(255);
fill(map(rapportLevel[0],-50,50,0,255));
noStroke();
rectMode(CENTER);
for (int i = 0; i < 5; i++) {
    rect((i+1)*width/7,height/2,100,100);
}
}

void stop() {
    mic.close();
    minim.stop();
    super.stop();
}

int setupCounter = 0;
int setupX = 0;
int setupY = -200;
int setupIndex = -1;
inHi = 850, inLo = 600;
float r;
int recordStartTime, presetRecordLength;

class DataDisplay{
    int x,y;
in w = 400;
in h = 400;
in index;
boolean side;
Person person;

    DataDisplay() {
        setupY+=h;
        if (setupY >= height) {
            setupY=200;
            setupX=w;
        }
x=setupX;
y=setupY;
if (setupCounter < 2) {
    side = false; //left person
person = personL;
}
else {
    side = true; // right person
    person = personR;
}
if (setupIndex == -1) {
    index = 0;
}
else if (setupIndex == 1) {
    index = 1;
}
println(x + " " + y);
println("side is " + side);
println("index = " + index);
}

void drawGraphs() {
    // draw currentVal first
    noStroke();
    fill(currentValColor);
    rectMode(CORNERS);
    r = map(person.pressureVals[index][0], inLo, inHi, 0, h);
    rect(x, y+h-r, x+10, y+h);

    stroke(pressureValsColor);
    noFill();
    strokeWeight(3);
    for(int i = 0; i < person.pressureVals[index].length; i++) {
        line(x+i*3, y+h, x+i*3, y+h - map(person.pressureVals[index][i], inLo, inHi, 0, h));
    }

    stroke(fftColor);
    strokeWeight(6);
    for(int i = 0; i < person.ffts[index].specSize(); i++) {
        // draw the line for frequency band i, scaling it by 4 so we can see it a bit better
        line(x+i*6, y+h, x+i*6, y+h - person.ffts[index].getBand(i)/12);
    }

    stroke(micColor);
    strokeWeight(10);
    line(x+w, y+h, x+w, y+h - (map(person.micTotal, 0, 1024, 0, h)));

    strokeWeight(15);
    if(index==0) {
        line(x+w-20, y+h, x+w-20, y+h - (map(avgLIntensity, 0, 1024, 0, h)));
    }
    if(index==1) {
        line(x+w-20, y+h, x+w-20, y+h - (map(avgRIntensity, 0, 1024, 0, h)));
    }
void keyReleased() {
    if (key=='g') {
        if(showGraphs) {
            showGraphs=false;
        } else if(showGraphs==false) {
            showGraphs=true;
        }
    }
    if (key=='r') {
        recordRapport = true;
        recordStartTime = millis();
        presetRecordLength = round(random(23000,30000));
        loop();
        showFinalImage=false;
        tintVal=0;
    }
    if (key=='d') {
        recordData = true;
    }
    if(key=='e') {
        if(recordRapport) {
            endRapportRecording();
        }
    }
    if(recordData) {
        noLoop();
        outL.flush();
        outR.flush();
        outL.close();
        outR.close();
        exit();
    }
}

void endRapportRecording() {
    // noLoop();
    PGraphics finalGraphic = createGraphics
        (recordedRapportMoments.length*3,300,P2D);
    finalGraphic.beginDraw();
    finalGraphic.background(255);
    finalGraphic.noStroke();
    finalGraphic.rectMode(CORNERS);
    float totalAverageRapport = 0;
    for(int i = 0; i < recordedRapports.length; i++) {
        totalAverageRapport+=recordedRapports[i];
        finalGraphic.fill(0,180,10,100);
        finalGraphic.rect(i*3,250-map(recordedRapportMoments[i],-50,50,0,200),
                          (i*3)-3,250);
    }
class Person {

    Arduino arduino;
    Person partner;
    FFT[] ffts;
    int pressurePins[];
    float[][] pressureVals;
    float[][] directionalVals;
    // float[] audioVals;
    float[] propL, propF;
    float[] propLDer, propFDer;
    float[] totalLR, totalLRDer;
    float micTotal;
    float[][] utterances;
    int utterancesCutoff;
    int lastUtteranceStart;
    float readVal;
    int[] highestBand;
    float[] highestBandVal;
    float[] avgBandVal, hiAvgBandDisparity;
    int closestPropLIndex, closestPropLDerIndex, closestTotalLRDerIndex;
    int[] closestFFTIndex;
    boolean side; //true=left, false=right
    float currentLoDiff, rapportMoment;

    Person(int num) {
        pressurePins = new int[2];
        // for(int i = 0; i < 2; i++) {
        //    pressurePins[i] = i;
        //}
        pressurePins[0] = 0;
        pressurePins[1] = 1;

        if(num==0) {
            side=true;
            arduino = arduinoL;
        }
    }

    int arrayLength = 64;
    int secondsPrior = 4;
    int postAnalysisScope = 63; //readsPerSec*secondsPrior;
    int audioAnalysisScope = readsPerSec*secondsPrior;
}

Coordination Table  

| Masso | 42 |
partner = personR;
if(num==1) {
    side=false;
    arduino = arduinoR;
    partner = personL;
}

for(int i = 0; i < pressurePins.length; i++) {
    arduino.pinMode(pressurePins[i], Arduino.INPUT);
}
pressureVals = new float[pressurePins.length][arrayLength];
directionalVals = new float[2][arrayLength];
// 0=left,1=right,2=front,3=back
// audioVals = new float[arrayLength];
ffts = new FFT[pressurePins.length];
for(int i = 0; i < ffts.length; i++) {
    ffts[i] = new FFT(pressureVals[0].length, readsPerSec);
}
closestFFTIndex = new int[ffts.length];

propL = new float[arrayLength];
// propF = new float[arrayLength];
propLDer = new float[arrayLength-1];
// propFDer = new float[127];
totalLR = new float[arrayLength];
totalLRDer = new float[arrayLength-1];
highestBand = new int[pressurePins.length][arrayLength];
highestBandVal = new float[pressurePins.length][arrayLength];
avgBandVal = new float[pressurePins.length];
hiAvgBandDisparity = new float[pressurePins.length];
    avgLoudness
}

void partnerTest() {
    if(side) {
        partner = personL;
    }
    else if(side==false) {
        partner = personR;
    }
    println(partner);
    println(partner.side);
}

void readSensors() {
    // read arduino sensors and store values in arrays
    // the idea here is to have an array for each sensor, all contained in an array (of arrays)
    // the first element of each array is the last read value, and the last is the least recent value
//so, we copy each array into itself, but one element later,
//then write the new value in the first element
for(int i = 0; i < pressurePins.length; i++) {
    readVal = arduino.analogRead(pressurePins[i]);
    readVal += arduino.analogRead(pressurePins[i]);
    readVal /= 2;  //those last three lines hopefully mitigate noise
    // println(readVal);
    arrayCopy(pressureVals[i], 0, pressureVals[i], 1, pressureVals[i].length-1);
    pressureVals[i][0] = readVal;
}

//adding the read vals into left, right, front, and back arrays for easy analysis
for(int i = 0; i < ffts.length; i++) {
    ffts[i].forward(directionalVals[i]);
}

void selfAnalyze() {
    // creates measures that can be compared between people's sensors
    arrayCopy(totalLR[0], 0, totalLR[0], 1, totalLR.length-1);
    totalLR[0] = directionalVals[1][0] + directionalVals[0][0];
    arrayCopy(totalLRDer[0], 0, totalLRDer[0], 1, totalLRDer.length-1);
    totalLRDer[0] = (totalLR[0] - totalLR[1])/totalLR[1];

    // then calculates a ratio of left to right
    arrayCopy(propL[0], 0, propL[0], 1, propL.length-1);
    propL[0] = directionalVals[0][0]/totalLR[0];
    // println(this + " propL = " + propL[0]);

    // calculate change in proportions
    arrayCopy(propLDer[0], 0, propLDer[0], 1, propLDer.length-1);
    propLDer[0] = propL[0] - propL[1];

    // find highest frequency
    for(int i = 0; i < ffts.length; i++) {
        arrayCopy(highestBand[i], 0, highestBand[i], 1, highestBand[i].length-1);
    }
}
arrayCopy(highestBandVal[i], 0, highestBandVal[i], 1, highestBandVal[i].length-1);

highestBand[i][0] = 0;

for(int j = 1; j < ffts[i].specSize(); j++) { // don't look at 0, it's always high
    if(ffts[i].getBand(j) < 75) { // clamping down low, noisy signals
        ffts[i].setBand(j, 0);
    }
    avgBandVal[i] += ffts[i].getBand(j);

    if(ffts[i].getBand(j) > highestBand[i][0]) {
        highestBand[i][0] = j;
        highestBandVal[i][0] = ffts[i].getBand(j);
    }
}

hiAvgBandDisparity[i] = ffts[i].getBand(highestBand[i][0])/avgBandVal[i];

// need to find an appropriate value for if statement
// println disparity vals to find something that works
//    if (hiAvgBandDisparity[i] < 1) {
//        highestBand[i][0] = 0;
//        highestBandVal[i][0] = 0;
//    }
// }

// println(this + " selfAnalyse results: ");
// println(totalLR[0] + " " + totalLRDer[0] + " " + propL[0] + " " + propLDer[0] + " " + highestBand[0][0] + " " + highestBand[1][0]);

void postAnalysis() {

    // find its own closest value to the other person's
    // FUTURE DEV: run one single i loop for everything (but fft) and use separate currentLoDiffs
    // then calculate which i value has the lowest total difference -- pick the lowest index val for
    // comparison. Adding some flexibility would be good too, but that may be a headache waiting to happen

    if(side) {
        partner = personR;
    } else if(side == false) {
        partner = personL;
    }

    // println("TEST VAL " + partner.propL[0]);

    for(int i = 0; i < ffts.length; i++) {
        currentLoDiff = 1000;
        for(int j = 0; j < ffts[i].specSize(); j++) {
            if(abs(partner.highestBandVal[ffts.length-i-1][0] - ffts[ffts.length-i-1].getBand(j)) < currentLoDiff) {
                closestFFTIndex[i] = j;
                currentLoDiff = abs(partner.highestBandVal[ffts.length-i-1][0] - ffts[i].getBand(j));
            }
        }
    }
}
currentLoDiff=1000;
for(int i = 0; i < postAnalysisScope; i++) {
    if(abs((1-partner.propL[0]) - propL[i]) < currentLoDiff) {
        closestPropLIndex = i;
        currentLoDiff = abs(partner.propL[0] - propL[i]);
    }
}

currentLoDiff=1000;
for(int i = 0; i < postAnalysisScope; i++) {
    if(abs((-partner.propLDer[0]) - propLDer[i]) < currentLoDiff) {
        closestPropLDerIndex = i;
        currentLoDiff = abs(partner.propLDer[0] - propLDer[i]);
    }
}

currentLoDiff=1000;
for(int i = 0; i < postAnalysisScope; i++) {
    if(abs(partner.totalLRDer[0] - totalLRDer[i]) < currentLoDiff) {
        closestTotalLRDerIndex = i;
        currentLoDiff = abs(partner.totalLRDer[0] - totalLRDer[i]);
    }
}

// println(this + " postAnalysis results:");
// println(closestFFTIndex[0] + " " + closestFFTIndex[1] + " " +
// closestPropLIndex + " " + closestPropLDerIndex + " " + closestTotalLRDerIndex);

void calculateRapport() {
    // how similar am I to my partner?
    rapportMoment = 0;

    float mirroringLRDiff = abs(propL[0] - (1-(partner.propL
        [partner.closestPropLIndex])));
    // println("LRDiff = " + mirroringLRDiff);
    float mirroringLRLevel = map(mirroringLRDiff, 0.016, .05, 50, -50);
    // println("LRLevel = " + mirroringLRLevel);
    rapportMoment += mirroringLRLevel*.2;
    float mirroringLRDerDiff = abs(propLDer[0] - -partner.propLDer
        [partner.closestPropLDerIndex]);
    rapportMoment += map(mirroringLRDerDiff, propL[1]/5,propL[1]/2, -50,50)*.2;
    float mirroringTotalLRDiff = abs(totalLRDer[0] - partner.totalLRDer
        [partner.closestTotalLRDerIndex]);
    rapportMoment += map(mirroringTotalLRDiff, totalLR[0]/5, totalLR[0]/2, 50, -50)*.2;

    // compare highest fftbands
    for (int i = 0; i < personL.ffts.length; i++) {
        float hiBandDiff = abs(highestBand[i][0]-partner.highestBand[i]
            [partner.closestFFTIndex[i]]);
        //
float hiBandMagDiff = abs(highestBandVal[i][0] - partner.highestBandVal[i][0]);
rapportMoment += (map(hiBandDiff, 4, personL.ffts[i].specSize() / 3, 50, -50) +
    map(hiBandMagDiff, 60, 400, 50, -50)) / 2 * 0.20;
}

if (Float.isNaN(rapportMoment)) {
    println("not a numbah");
    rapportMoment = 0;
}
//    println("rM: " + rapportMoment);
}