Eindhoven University of Technology, Bachelor College
Major: Industrial Design
DPB100 - Project 1 Design
Sports
2015/2016 Semester B
B1

Co-Coach

Project coach: P.J.F. Peters and S.I. Lucas
Tutor: Emma Zoelen

T.P. Ernest	s154244
W. de Groot	s153481
W.W.J. Gelden	s158290
W.H.J Schaeffers	s151098

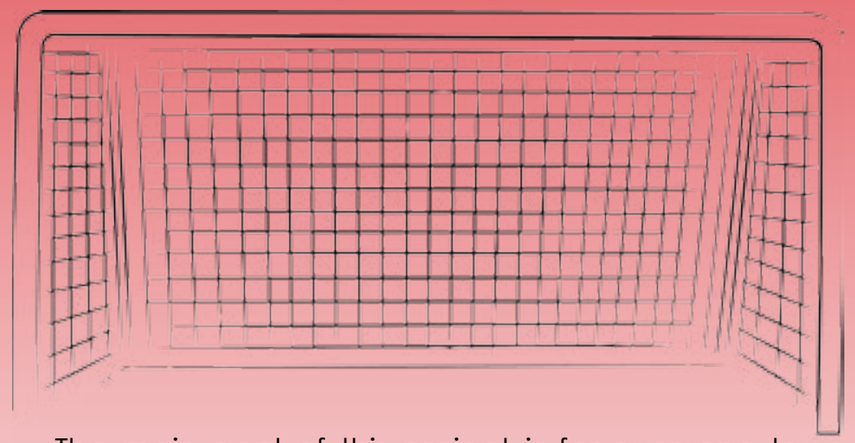
Table of Content

Introduction

This project on sports is about designing a product or service which can translate behavioural data to engage people in recreational sports in a social and playful way. In order to achieve this, several steps are needed within the design process. First, ideas must be generated, leading towards a concept which will be further worked out into a prototype. This prototype will be the starting point for the final design. In between these steps, the user is involved so we are able to iterate our design in such a way, that users actually want to use the product.
This project is not only about designing a product, but also about developing ourselves into becoming more experienced within all the aspects of design as a whole. Making progress in each of the expertise areas is essential in order to accomplish this development.

As a group, we have been busy with the abovementioned. We have designed the Co-Coach, an interactive data gathering device used by coaches to keep track of the events in a sports game in order to give feedback to the players. This will eventually result in a better overview for the coach as well as the players. The players can improve through the instructions of the coach after evaluation..

Project goal



The main goal of this project is for everyone to get familiar with the design process and develop the expertise areas. This is really important within the Industrial Design department because these aspects are of great significance for becoming the designer you want to be.
As for the project, we want to let teams perform better through reflecting upon previous matches. We want to enhance the quality of sports and as a result of that, we want people to get more

photo: clipart

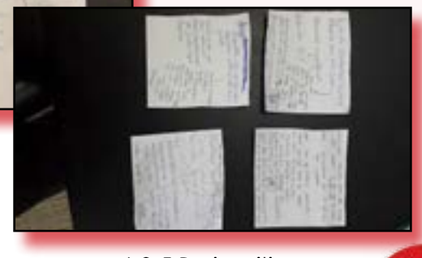
Process

In the first week of our project B1.2 we started with a new concept for us, called 'the pressure cooker'. The assignment for sports projects was to find a product which concerned the following question: 'How can systems translate behavioral data to engage people in recreational sports in a social and playful way?'.

Within one week we came up with some ideas, a final idea and a low fidelity prototype. During the pressure cooker we used multiple methods to come up with ideas. Starting with mindmapping, after this we proceeded to a method which one of our group members used during From Idea To Design and found very effective, this method is called **'Idea box' from thinkertoys by Michael Michalko (Michalko, M. (1991). Tinkertoys.** *(2nd ed.). United States: Ten Speed Press.). Unfortunately this method was not* working with our group so we tried the 6-3-5 Brainwriting brainstorm method. We changed it a bit to 4 participants 4 ideas and 5 minutes. Every time we past an idea trough the next person has to add a new feature to the idea. After finishing the 20 minutes we evaluated the ideas and discussed them. We all were most content about one concept.

Mindmapping

idea box



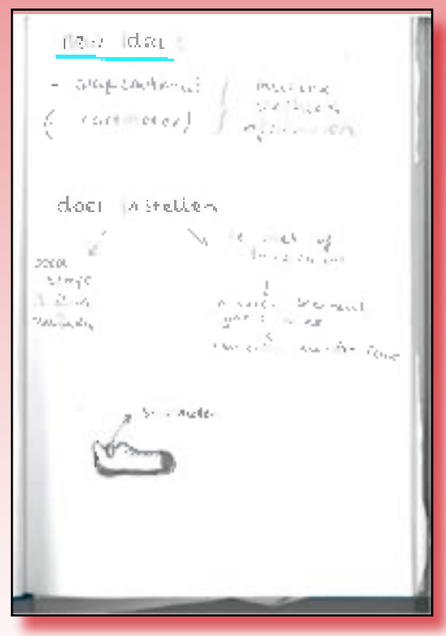
6-3-5 Brainwriting

This concept was the idea of interactive pylons and goals, for example to know when a ball has passed the goal line. We made a versions in which the pylons could move around when ordered to.
But we changed to pylons who could not move, this is because the trainer otherwise always has to have a controller to operate the pylons and thus could not focus on or join the team.
This was our *first small iteration.*
The low fidelity prototype was made and an enthusiastic pitch presented.

This lead to our *second iteration.* After presenting and searching on the internet, we found out that this concept already existed named 'The Smart Goals'. This brought us to the phase of conceptualization. We had to discuss and decide whether to continue with this concept and try to add or remove some features to improve the product, or to start again with generating new ideas. We decided, after doing some research on 'The Smart Goals', to change our concept and generate new ideas, because the already existing idea is quite well developed and we had no prominent options to add.

Interactive pylon

We decided to brainstorm by ourselves this time and this resulted in two new potential great ideas. The first idea was a device that keeps track of your walking speed and adjusts the music tempo to your walking rhythm. The user can set a goal to reach at the end of a jog session. Every time the user would run too slow, the music's tempo will decrease as well, making the user realise he has to speed up. This would work the other way around as well. If you would run too fast, the tempo would rise and the music will not sound as good as it is supposed to. The main idea was to regulate one's walking speed to be more constant rather than varying all the time and in this way reach your goal as planned.
The second idea was a device that would calm down tensed sporters before a match, to achieve better results. This would be done by placing the device in the changing room, where the players can get exercises to relax. Also, music and lighting would be involved. We discussed these ideas and asked for advice with our student coach.



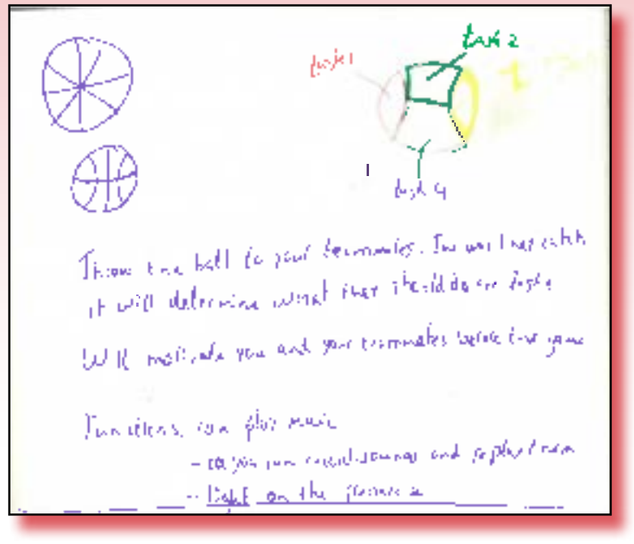
device that keeps track of walkoing speed

We decided to go with the second idea and called it ‘Sthriver’. We did a lot of research: How to change attitude towards the match; How to prevent stress; What kinds of stress exist; What colors are relaxing; If groups or individuals are better to work with when preventing stress. (references) The conclusions we got from this was, that stress is very common and can appear in many different forms. The most common form is to totally block, this is the form we want to prevent during matches. Another conclusion is that, because of the many different forms, preventing the whole group to stress out is almost impossible. Every player has his own issues to work on.
These conclusions are one of the reasons that lead to our *third iteration.* We changed our concept a bit from preventing stress to excite people for a match. The other reason was, because of the results from the questionnaire (link) we made. Many people had the opinion that stress was a good thing. Also, other sports project groups set the same thing about personalizing the device. The reason why we decided to excite every player to play the match is because it is very hard to design one device that can be used for every personal stress issue. In addition, cheering everyone up before the match starts is also a small way of stress releasing and gives a feeling of cohesion within the team players.

two different forms of prototypes of the first version of the Sthriver

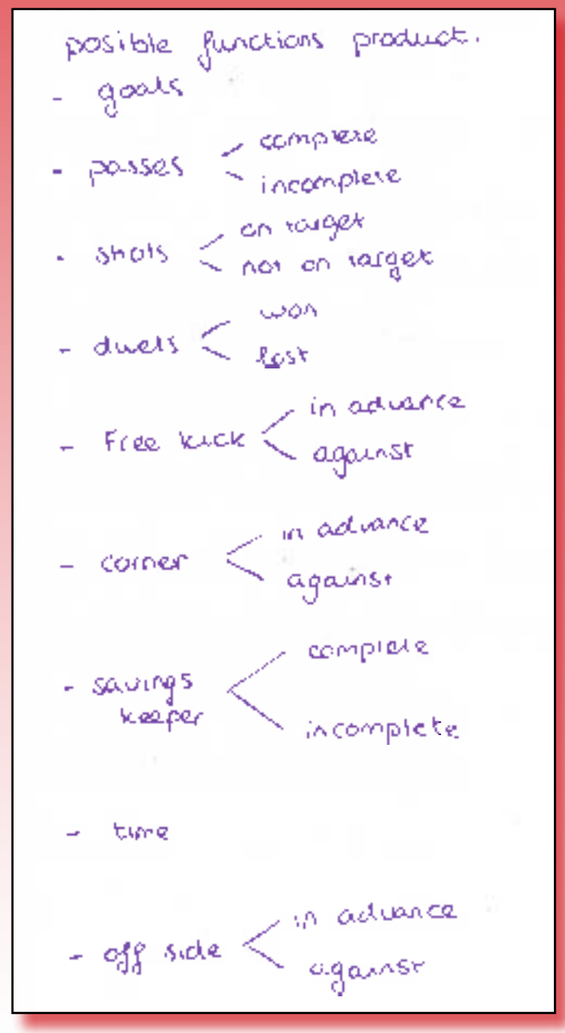
This also changed the form of our idea from a device implemented in the dressing room to an interactive ball, which was our *small fourth iteration.* The ball keeps the players warm and in a sportive mood. The interactivity also helps the players to cohere. We tried to come up with some interactive exercises to do, but we got stuck on this. We thought the exercises were to forced and may get boring after one season.

This caused the big defining change and our most important *fifth iteration*. Generating ideas for the ‘Sthriver’, we came up with a whole new concept by accident. Since we were still in the conceptualization phase and thus we thought it was accepted to still generating new ideas, we decided to go through with this idea.



The idea of a different form for the sthriver.

So came the **Co-Coach:** an interactive data gathering device to give feedback for the players. This idea is about tracking statistics during a sports match. These statistics can be used to reflect on the players' performances in order to enhance their gameplay. This is can be done by holding the Co-Coach during the match. Mostly the assisting coach will do this, but the coach can also do this himself/herself. Before the start of the game, the coach will decide what he wants to track during the match, this can be for example lost or won duels, passes, corners, etc. Every time one of these things happen, the coach has to push a button. The Co-Coach will translate the pushed button into a negative or positive outcome and will send this data to a laptop via Bluetooth. After the game the coach can see this data and discuss these results with his team through a graph. The coach can then decide to train on what needs improvement. This way the team will train more effective, resulting in a better overall performance during matches.

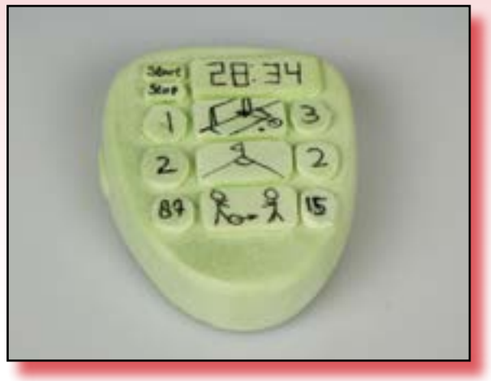


functiondescription Co-Coach

Co-Coach

First, we investigated what functions the device needs. We thought of a stopwatch, clock, four buttons to track and one button to reset. Then we thought of options to track during a match, this was very hard because it is different for every sport. So we decided that the coach could add what he wants to track in some kind of an app, which is connected to the Co-Coach. But evaluating every option, we came to the conclusion that the stopwatch and clock were too much and, could also be done by a mobile phone. So this left us with for buttons, an app and one reset button and thus our *sixth iteration.*

Then our seventh and eighth iteration came due to our user test. This showed that wanting to track four things and still focus on the game is a bit too much to ask from an amateur coach. So we changed from a device with many buttons and displays to something which is more active by movement detection. Pressing one button and moving the Co-Coach left (negative) or right (positive) immediately tracks date. This way coaches can focus on the match while using the co-coach. Also because of the same reason, we now only focused on one sport (soccer) and one selectable aspect of the match.



first design of the Co-Coach

secend design of the Co-Coach

Co-Coach

And the *last iteration* had to do with a more suitable look and comfortable form, which we asked feedback for during presentations. Also we went to vertigo to make multiple prototypes and ask around what people thought was the most comfortable. This resulted in a selection of two versions, one for bigger hands (men) and one for smaller hands (woman). To make the design look more sportive and combative we added a pattern and gave it a red color.

final design of the Co-Coach

Pattern and colour of the Co-Coach

Design process of the Co-Coach

Demo Days

Midterm demo-day

At the midterm demo-day we presented our first working prototype together with a design we wanted the final prototype to look like. The working prototype was a wooden box containing an Arduino and a gyroscope connected with a USB cable towards a laptop. The statistics were displayed in the serial port we opened on a laptop screen. Although, this was only a small beginning of our prototype, the concept was clear and we were given an opportunity to receive feedback. Our fellow students stated that the concept was clear after some explanation and could be useful for amateur teams. This was a great relief for us because now we were not the only ones who were confident about the concept, after changing it multiple times. Prior to the midterm demo-day, we did not really know what to expect and get out if it. We presented what we had and hoped for useful reactions, in order to proceed towards the final demo-day.

midterm demo day stand

Final demo-day

For the final demo-day, we set the goal for ourselves that we wanted to make a video to create a clear image of our concept within the context of a soccer game. Also, we wanted to make the prototype functional through buttons and wireless communication. We managed to do all these things and were satisfied with the end result. We presented a poster, video, demonstration of the prototype and the actual prototype as we had it in mind. This was all supported by a pitch we had prepared. Overall, we got good reactions and we want to highlight one of them.

final demo day stand

There was a man who was very interested in our concept. After we told him what the Co-Coach was about, he told us that he had been a coach as well for a very long time, and that he found our idea alluring. He told us a story of a Danish soccer team which used statistics to build a team. This team managed to promote 3 divisions within 3 years and eventually won the competition of the highest league in Denmark, all by analysing and using statistics. He also told us that he once made use of an application on his phone to gather data of a soccer match, but it did not work for him. He saw more opportunities in our design than in the app he used, which was really pleasant to hear. He even brought about an additive function for the Co-Coach; a voice recording function. He said he would find it useful to point out specific actions or events during a game, so he will not forget to talk about that specific thing after the match was done. Another thing he pointed out, was that this could also be useful in educating referees. He currently trained to-be referees and thought this had quite some opportunities within that context as well. It was a real pleasure to have talked with this man as he could imagine himself really using it.

the coach at demo day

User testing

User test 'Sthriver'

Our group went through several phases of user testing. For our first real concept, 'Sthriver', we set up a survey to get some first impressions on what people experience while they sport. The survey was focused on getting to know if people had a lot of tension before a match and if they had the feeling that this tension affected their performance badly during a match. In addition, we wanted to know if sporters use techniques already. If not, we wanted to know if they were open for the experience to try a device which could help them. Of course we already had an outline for a concept but we wanted to use some additional input from the actual users so we could take into account their opinions and inspirations. We chose for this specific method because we as a group wanted to have more insight in the needs of a sporter before elaborating our concept. If we had set up a more profound test, it would have taken a lot more time, as well as being of little use in a later stage of the whole process.

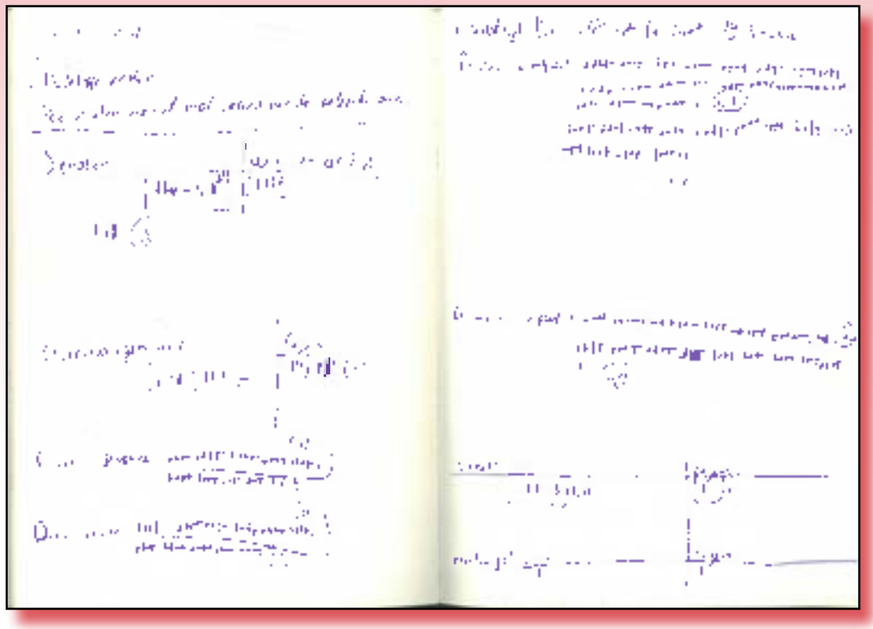
After making the survey, we carefully analyzed the results in order to find opportunities to design for. Our findings were helpful and sometimes unexpected. We found out that an amateur, professional, team player or individual player does not matter in how much tension a sporter feels before the game. Furthermore, every sporter feels some tension before a match, some more than others. In addition, almost everyone thinks that humor has a positive influence on stress release, but it has to be balanced. Something that surprised us, and we still cannot figure out why this is the case, is that sporters who feel lots of tension and feel like it has a negative influence on their performance, do not want to try a product which can help them release some stress.

The results of this questionnaire caused our group to make some important decisions about the concept. These decisions let to a whole new idea within the concept, which can be seen in the process of our project.

The survey can be found in the appendix.

User test Co-Coach

When we made the decision to continue with the concept of the Co-Coach, user testing had to be done. Before the decision was made to drop the concept of ‘Sthriver’ and continue making progress with Co-Coach, no formal way of user testing was done. However, we asked people in our surroundings if they saw potential in this concept and overall, the reactions were really positive. Our feeling was similar so we came up with a concept and then the following user test was done:
Ward visited a soccer game in his hometown in which he kept track of a lot of statistics during the game, including data such as passes complete/ incomplete, duels lost/won and a lot more. This was done with pen and paper, which Ward uses to keep track of everything he saw during the match. The complete findings can be seen in the table below. In the first half, the targeted coach’s team was ahead 1 to 0. In the second half, the opposing team scored, which made the score 1 – 1 which was also the end result of the game.



Keeping track of statistics by hand

Insights from the coach’s perspective

After the game, the results from the match were shown to the coach and he was then interviewed. A lot of valuable insights were gained following this interview. For example, the coach found these statistics really useful and immediately asked for a copy of the data gathered during the game. Also, he told the reason why he thought it is really useful, this is because he could analyze the statistics and evaluate these to his players. “In this way, the players get an insight in what they did in the game and where they missed out on”, he said. Moreover, he was asked if he wanted to keep track of these statistics all by himself. His answer came down to the fact that he would prefer someone else to do it, because he wanted to still be able to give good personal instructions to the players as well as keeping a clearer overview of the game. This is really comprehensible, so we decided to integrate that part in the final design. In addition, something that also made a change in our design, was the question to the coach which results he really found useful. His answer was that this is different for every team. Through these insights, we believe better decisions were made. Limiting the options on the Co-Coach lowers the difficulty to operate the device by a whole lot, because less functions need to be remembered.

prototype we showed at the user tes

Insights from the players' perspective

While showing the results to the coach, some of the players of the concerned team gathered around to see what they had done during the game. They experienced that they gave up a lot of things in the second half, which could also be concluded out of the data gathered. They also found the statistics very interesting and started discussing themselves about what went well and what went wrong during that game. The statistics gathered showed clear evidence that they performed less in the second half, which was an aspect one of the players addressed as well. For them, as well as for us, it was really interesting to see that you can get quite a clear image of how a game progressed by means of statistical data.

Beginning of the match HMVV - De Valk

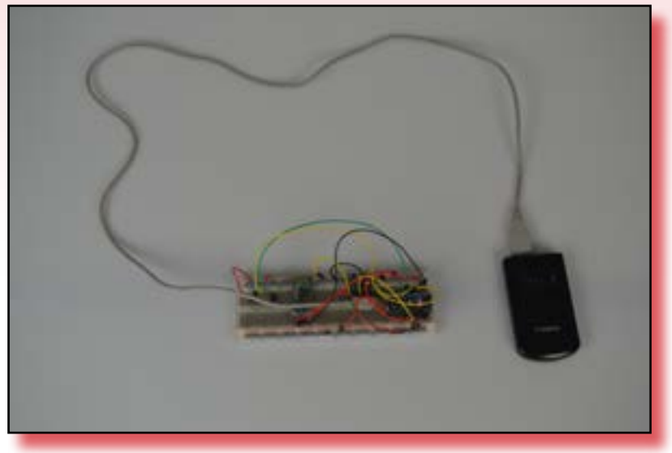
Technology

Arduino

During the design process, we were constantly working on implementing the functions in Arduino while also looking at the hardware we wanted to use. We used the regular Arduino Uno as microcontroller at first, so we could start experimenting with what exactly was needed for our final design. We knew that we had to be able to make gestures in order to register certain statistics. Moreover, we wanted to make the device operational through wireless connection. In a later stage, we chose to implement everything on an Arduino Nano, because we wanted to make our prototype as small as possible.

MPU 6050

That is why we implemented a gyroscope (MPU 6050) in the circuit. We used a library developed by Jeff Rowberg for our arduino code. At first we had some troubles with what values we could actually use out of the serial port. After some searches we managed to get values which were meaningful to us, namely the tilting angle in degrees. We used these values to experiment with how much the Co-Coach should be tilted before registering a certain statistic. We also had some troubles with the rate of data flowing through the serial port. Once we tilted the whole device, the stats were incremented way too much, so we wrote a piece of code that it only counted one time, until it is back in its original position.



This is how the hardware looks like

Bluetooth

For our design to be functional for the coach, it would be really pleasant to have it working on wireless connection. That is why we integrated a Bluetooth module (HC-06) in the circuit. Setting up the Bluetooth connection took some time, but after some time we managed to get it working. We learned that it was not able to upload sketches to the arduino while the transmit and receive pins were connected, so every time we made some changes in the sketch, we had to pull out the pins, which was annoying sometimes but not really stressful in any sense. While using Bluetooth, we had some troubles with the COM-ports every now and then, but in the end it worked as it was supposed to do.

graph shown in proceccing via bluetooth

Buttons

We wanted to make the use of the Co-Coach as easy as possible, that is why we implemented three buttons on the circuit. One button is used to 'notify' the device that a tilt movement is coming. Without this button, no tilting movements are registered. We did this because a coach can accidently tilt his hand, and the meaning is not to register a statistic while it was not the coach's intention to do so.

The other two buttons are used for registering events which are happening frequently. We chose to do this because we do not want the user to move with his hands too much. Moreover, it would make the device more difficult to use when we linked too much actions to a certain registration. These buttons had to be pressed on one time to per increasment. This causes another problem: bouncing. We had to look on the internet how we could debounce these buttons and we wrote a piece of code which made sure that the buttons could only be pressed four times per second.

Group refelction

This semester, we have been designing a product within the domain of sports. We have designed the Co-Coach: a device which helps coaches of soccer teams to track their teams data in order to improve their performance. This will make the team more aware of what they are doing in a game and with the data retrieved, they can improve on more specific aspects, such as passing and duels.

During the early stages of the process, we tried to come up with this concept, but before this came up to us, we have been working on different concepts. Though, we got stuck on these concepts, leading to other ideas which we thought were more feasible and better overall. From this, we learned that a design process does not always go fluently. Moreover, changing your mind is not that bad, if you just keep going if there is a good reason behind it.

Our teamwork made all of these changes possible, because we argued in the right way. We listened to each other and actively participated in these discussions, leading towards the right decisions afterwards. We are reasonable towards each other and tried to think of different solutions whenever this was needed. This made us bring the project to a good end. However, there is one thing we as a group could improve on, namely our meetings. Some of them were quite inefficient in the early stages of the project, making time go to waste. If we had planned more while sticking to the planning, we probably could have done more for a better result.

After the first quartile, we divided the tasks as follows: Tess and Willem S focused on the aesthetics of our design, while Willem G and Ward focused on the technology. In this way, we developed our expertise areas more detailed for the part we were responsible for, while also getting involved in the expertise areas the other group was working on. Every now and then, we discussed with each other what we had done and if we had any additive advice for each other. This sped up our progress noticeably, which felt good for all of us. We had the feeling we made good progress while still being one team. This can be considered as a big learning point for all of us, and it is something we should keep doing in future projects.

Throughout our project, we were documenting a lot. We made a lot of pictures of for example our prototypes. Also, we kept our dummies up to date, resulting in a good overview of what we have done and what still has to be done. Now, we can say that we have a really good documentation of the whole progress we made together.

To conclude, we will not give up as fast as we previously did. Moreover, we will make a concrete planning which will also make our meetings more efficient. We want to use the insights gained from this project in the future as well, so we can work more efficiently in other projects.

grouppicture during the final demo day
photo made by Thijmen van Wees

Conclusion

All in all, we are satisfied with the end result. We achieved our goals for the final demo-day and we have made the quality of what we were presenting as high as possible within the time available. If we had more time, we would have done more user testing with our real working prototype and maybe iterate some of the functions. Also, we would try to make the electronics fit in hand size, because it was still a bit too big in our opinion. Moreover, we would have looked into the value of the function of voice recording and considered to implement it.

<u>Project goal reflection</u>

We can conclude that we have achieved our project goal to the utmost extent. We have all developed some expertise areas, except the Business and Entrepreneurship area, because we wanted to focus more on the concept and technology of our design. We believe that teams can perform better after a longer use of the Co-Coach, therefore enhancing the quality of sports. The Co-Coach can help the coach in communicating towards his team in a more formal way. In this way, players will also get more involved in sports as they use statistics to improve themselves, which was one of the goals we set ourselves for this project.

References

Tinkertoys. (2nd ed.). United States: Ten Speed Press.)

Kompier, M. and Cooper, C. (1999). Preventing Stress, Improving Productivity: European Case Studies in the workplace. Retrieved 28 February, 2016, from https://books.google.nl/books?hl=nl&lr=&id=OFsSz45OxewC&oi=fnd&pg=PR11&dq=preventing+stress&ots=S5aT_FXkW3&sig=UMq6x9qkW6jfjXkQSdJlaunM4vo#v=onepage&q=preventing%20stress&f=false books?hl=nl&lr=&id=OFsSz45OxewC&oi=fnd&pg=PR11&dq=preventing+stress&ots=S5aT_FXkW3&sig=UMq6

Mcgonigal, K. (June 2013). Ted talk: How to make stress your friend?. Retrieved 1 maart, 2016, from https://www.ted.com/talks/kelly_mcgonigal_how_to_make_stress_your_friend?language=nl

Falaire, E., Sagnol, M., Ferrand, C., Maso, F., & Lac, G. (2001, 01 juni). Psychophysiological stress in judo athletes during competitions. Geraadpleegd van http://search.proquest.com/openview/f99d3670162ec1beecb2729d095644cf/1?pq-origsite=gscholar

Richard, E., & Barbara, G. (1991, 12 september). Stress control system and method.
Consulted from

Appendix

Arduino code

```
//This code is used for the Co-Coach, a device developed by Tess Ernest, Ward de Groot, Willem Schaeffers en Willem Gelden from the University of Technology Eindhoven at the faculty of Industrial Design.
//This code code is based on the MPU6050_DPM6 example from the MPU 6050 library from Jeff Rowberg.
//06/09/2016

// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2.0)
// 6/21/2012 by Jeff Rowberg <jeff@rowberg.net>
// Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
//
// Changelog:
//      2013-05-08 - added seamless Fastwire support
//                 - added note about gyro calibration
//      2012-06-21 - added note about Arduino 1.0.1 + Leonardo compatibility error
//      2012-06-20 - improved FIFO overflow handling and simplified read process
//      2012-06-19 - completely rearranged DMP initialization code and simplification
//      2012-06-13 - pull gyro and accel data from FIFO packet instead of reading directly
//      2012-06-09 - fix broken FIFO read sequence and change interrupt detection to RISING
//      2012-06-05 - add gravity-compensated initial reference frame acceleration output
//                 - add 3D math helper file to DMP6 example sketch
//                 - add Euler output and Yaw/Pitch/Roll output formats
//      2012-06-04 - remove accel offset clearing for better results (thanks Sungon Lee)
//      2012-06-01 - fixed gyro sensitivity to be 2000 deg/sec instead of 250
//      2012-05-30 - basic DMP initialization working

/* ============================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

```
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
===============================================
*/

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
//#include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

/* =========================================================================
   NOTE: In addition to connection 3.3v, GND, SDA, and SCL, this sketch
   depends on the MPU-6050's INT pin being connected to the Arduino's
   external interrupt #0 pin. On the Arduino Uno and Mega 2560, this is
   digital I/O pin 2.
 * ========================================================================= */

/* =========================================================================
   NOTE: Arduino v1.0.1 with the Leonardo board generates a compile error
   when using Serial.write(buf, len). The Teapot output uses this method.
   The solution requires a modification to the Arduino USBAPI.h file, which
   is fortunately simple, but annoying. This will be fixed in the next IDE
   release. For more info, see these links:

   http://arduino.cc/forum/index.php/topic,109987.0.html
   http://code.google.com/p/arduino/issues/detail?id=958
 * ========================================================================= */


// uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock (for
// more info, see: http://en.wikipedia.org/wiki/Gimbal_lock)
#define OUTPUT_READABLE_YAWPITCHROLL
```


```
// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
boolean increaseAllowed = true;

//digital pins for the buttons
int button1 = 3;
int button2 = 4;
int button3 = 5;

//state of the buttons
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;

// we need this to prevent complications concerning the buttons
long lastDebounceTime2 = 0;
long lastDebounceTime3 = 0;
long debounceDelay = 250;

//statistics
int stat1 = 0;
int stat2 = 0;
int stat3 = 0;
int stat4 = 0;

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// ================================================================
// ===               INTERRUPT DETECTION ROUTINE                ===
// ================================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}
// ================================================================
// ===                      INITIAL SETUP                       ===
// ================================================================
```

```
void setup() {
  //set buttons as input
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
  pinMode(button3, INPUT);

  // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif
  // initialize serial communication
  // (115200 chosen because it is required for Teapot Demo output, but it's
  // really up to you depending on your project)
  Serial.begin(115200);
  //while (!Serial); // wait for Leonardo enumeration, others continue immediately

  // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3v or Ardunio
  // Pro Mini running at 3.3v, cannot handle this baud rate reliably due to
  // the baud timing being too misaligned with processor ticks. You must use
  // 38400 or slower in these cases, or use some kind of external separate
  // crystal solution for the UART timer.

  // initialize device
  // Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  // verify connection
  // Serial.println(F("Testing device connections..."));
  //Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
  // wait for ready
  //Serial.println(F("\nSend any character to begin DMP programming and demo: "));
  //  while (Serial.available() && Serial.read()); // empty buffer
  //  while (!Serial.available());                 // wait for data
  //  while (Serial.available() && Serial.read()); // empty buffer again
  // load and configure the DMP
  //Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();
  // supply your own gyro offsets here, scaled for min sensitivity
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
    // turn on the DMP, now that it's ready
    //Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);
    // enable Arduino interrupt detection
```


```
    //Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();
    // set our DMP Ready flag so the main loop() function knows it's okay to use it
    //Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;
    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    //Serial.print(F("DMP Initialization failed (code "));
    //Serial.print(devStatus);
    //Serial.println(F(")"));
  }
}
// ================================================================
// ===                    MAIN PROGRAM LOOP                     ===
// ================================================================
void loop() {
  // state of the buttons
  buttonState1 = digitalRead(button1);
  buttonState2 = digitalRead(button2);
  buttonState3 = digitalRead(button3);
  // if programming failed, don't try to do anything
  if (!dmpReady) return;
  // wait for MPU interrupt or extra packet(s) available
  while (!mpuInterrupt && fifoCount < packetSize) {
    // other program behavior stuff here
    // if you are really paranoid you can frequently test in between other
    // stuff to see if mpuInterrupt is true, and if so, "break;" from the
    // while() loop to immediately process the MPU data

  }
  // reset interrupt flag and get INT_STATUS byte
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();
  // get current FIFO count
  fifoCount = mpu.getFIFOCount();
  // check for overflow (this should never happen unless our code is too inefficient)
  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
    // reset so we can continue cleanly
    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));
    // otherwise, check for DMP data ready interrupt (this should happen frequently)
  } else if (mpuIntStatus & 0x02) {
    // wait for correct available data length, should be a VERY short wait
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
    // read a packet from FIFO
```

```
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    // track FIFO count here in case there is > 1 packet available
    // (this lets us immediately read more without waiting for an interrupt)
    fifoCount -= packetSize;
#ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    if ( (millis() - lastDebounceTime2) > debounceDelay)  { //debounce button
      if (buttonState2 == HIGH) {
        stat1 = stat1 + 1; //if the button is pressed, add 1 to statistic 1
        Serial.print(stat1);//print statistics in serial port
        Serial.print(',');
        Serial.print(stat2);
        Serial.print(',');
        Serial.print(stat3);
        Serial.print(',');
        Serial.print(stat4);
        Serial.println(',');
        lastDebounceTime2 = millis();
      }
    }
    if ( (millis() - lastDebounceTime3) > debounceDelay) {
      if (buttonState3 == HIGH) {
        stat2 = stat2 + 1;
        Serial.print(stat1);
        Serial.print(',');
        Serial.print(stat2);
        Serial.print(',');
        Serial.print(stat3);
        Serial.print(',');
        Serial.print(stat4);
        Serial.println(',');
        lastDebounceTime3 = millis();
      }
    }
    if (increaseAllowed == true && buttonState1 == HIGH) {
      if (ypr[2] * 180 / M_PI > 25) {
        stat3 = stat3 + 1; // if button 1 is pressed and the divice is tilted 25 degrees to one side, add 1 to statistic 3
        increaseAllowed = false;
        Serial.print(stat1);
        Serial.print(',');
        Serial.print(stat2);
        Serial.print(',');
        Serial.print(stat3);
        Serial.print(',');
        Serial.print(stat4);
        Serial.println(',');
      }
```


```
      if (ypr[2] * 180 / M_PI < -25) {
        stat4 = stat4 + 1;
        increaseAllowed = false;
        Serial.print(stat1);
        Serial.print(',');
        Serial.print(stat2);
        Serial.print(',');
        Serial.print(stat3);
        Serial.print(',');
        Serial.print(stat4);
        Serial.println(',');
      }
    }

    // the device has to be in its neutral posistion again before statistics 3 and 4 can increase again
    else {
      if ((ypr[2] * 180 / M_PI < 25) && (ypr[2] * 180 / M_PI > -25)) {
        increaseAllowed = true;
      }
    }
#endif
    }
}
```

Processing code

```
//This code is used for the Co-Coach, a device developed by Tess Ernest, Ward de Groot, Willem Schaeffers en Willem
Gelden from the University of Technology Eindhoven at the faculty of Industrial Design.
//06/09/2016

import processing.serial.*;
Serial port;

// array to store statistics
float[]a = {
  0, 0, 0, 0
};

//length of the graphs
float lengteA=0;
float lengteB=0;
float lengteC=0;
float lengteD=0;

//variables
String serial;
int b=30;
int c=b+20;

//images
PImage bg;
PImage pic;


void setup() {
  size(1024, 754); //size of canvas
  bg = loadImage("duel.jpg"); // load background picture
  pic = loadImage("Co-Coach v3.png"); // load logo
  port = new Serial(this, "COM3", 115200); // connect to arduino
  port.bufferUntil('\n');
  port.clear();  // function from serial library that throws out the first reading, in case we started reading in the middle of
a string from Arduino
  serial = null; // initially, the string will be null (empty)
}

void draw() {
  background(bg); // set background
  fill(222, 0, 0, 150); //red frame color
  strokeWeight(8);
  rect(b, 20, (width/2)-60, 500);//red frames
  rect(542, 20, (width/2)-60, 500);
  fill(0, 0, 0, 150);// black frame color
  rect(c, 410, 412, 100);//black frames
  rect(c+(width/2), 410, 412, 100);
```


```
while (port.available () > 0) { //as long as there is data coming from serial port, read it and store it
  serial =(port.readStringUntil ('\n'));
}
if (serial!=null) {
  //store statistics from arduino in array a
  a= float(split(serial, ','));
}

if (a[0]>0||a[1]>0 || a[2]>0 ||a[3]>0) {
  //calculate graph length
  lengteA= (a[0]/(a[0]+a[1]))*350;
  lengteB= (a[1]/(a[0]+a[1]))*350;
  lengteC= (a[2]/(a[2]+a[3]))*350;
  lengteD= (a[3]/(a[2]+a[3]))*350;
}
//draw graphs
rect(c, 400, 100, -lengteA);
rect((width/2)-150, 400, 100, -lengteB);
rect(c+(width/2), 400, 100, -lengteC);
  rect(width-150, 400, 100, -lengteD);

  fill(255); //text color
  textSize(32);// text size
  //text

  text(int(a[0]), 85, 450);
  text(int(a[1]), 397, 450);
  text(int(a[2]), 85+(width/2), 450);
  text(int(a[3]), 397+(width/2), 450);

  text("Passes", c+155, 450);
  text("Duels", c+(width/2)+160, 450);
  textSize(24);
  text("Complete", 85, 490);
  text("Incomplete", 290, 490);
  text("Won", 85+(width/2), 490);
  text("Lost", 370+(width/2), 490);

  //logo
  imageMode(CENTER);
  image(pic, width/2, 640);
}
```