IDP im Sommersemester 2013

QualityControlTool

Sarah Barton und Sonja Vogl
Technische Universität München

05. Juli 2013

Zusammenfassung

Diese Ausarbeitung gibt dem Benutzer einen Einblick in die Bedienbarkeit des QualtiyControlTools. Ausgehend von der ersten Version der beiden Entwickler Martin Attenkofer und Johannes Eder werden die Erweiterungen des QualityControlTools vorgestellt. Zudem geht diese Arbeit auf die Funktionen ein, die das QualityControlTool zur Analyse der Biosensoren GeneActive, Somnowatch, Shimmer und GT3X+ bereitstellt.

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit gibt dem Benutzer des QualityControlTools einen Überblick über die Möglichkeiten, die das Programm zur Analyse der Biosensoren GeneActive, Somnowatch, Shimmer und GT3X+ bietet. Zudem wird dem Benutzer nahegebracht, wie das QualityControlTool zu bedienen ist. Hierzu wird zunächst die Benutzeroberfläche (siehe Kapitel 4), wie auch die Funktionsweise der einzelnen Buttons dargestellt. Zusätzlich wird grob auf die Implementierung eingegangen. Um den Einstieg für neue Nutzer des Tools möglichst einfach zu gestalten wird in Kapitel 6 beschrieben wie eine Auswertung einer Datei ablaufen kann. Außerdem werden in Kapitel 7 einige nützliche Hinweise zur richtigen Benutzung des Tools gegeben.

2 Motivation

Das Institut für Medizinische Statistik und Epidemiologie (IMSE) der TU München beschäftigt sich mit der Analyse der Daten der KORA Studien. KORA steht für die *kooperative Gesundheitsforschung in der Region Augsburg.*
Das Ziel der *Feasibilty*-Studie der Nationalen Kohorten war das Testen der Realisierbarkeit mono- und multifunktioneller Apparaturen, die Körperbewegungen messen. So sollten Körperleistung (PA) und der Energieverbrauch analysiert werden. Im sepziellen wurden vier verschiedene Biosensoren betrachtet, die *raw acceleration data* für mindestens 7 Tage lieferten.
Das QualityControlTool soll das Auswerten der csv-Dateien von den Biosensoren erleichtern. Dafür existieren zahlreiche Algorithmen, welche in MATLAB implementiert sind. Allerdings besitzen viele Anwender nur ein eingeschränktes Informatikwissen. Für diese ist eine Auswertung mit einem hohen zeitlichen und hohem Arbeitsaufwand, sich in Matlab einzuarbeiten, verbunden.
Mit dem QualityControlTool ist eine einfache Bedienbarkeit ohne informatische bzw. mathematische Kenntnisse gegeben. Sobald man mit der Benutzeroberfläche vertraut ist, übernimmt der PC das Auswerten der Daten. Zusätzlich werden die ausgewerteten Daten so gespeichert, dass sie schnell auffindbar sind und jederzeit erneut abgerufen werden können.

3 Ausgangslage - die erste Version des Tools

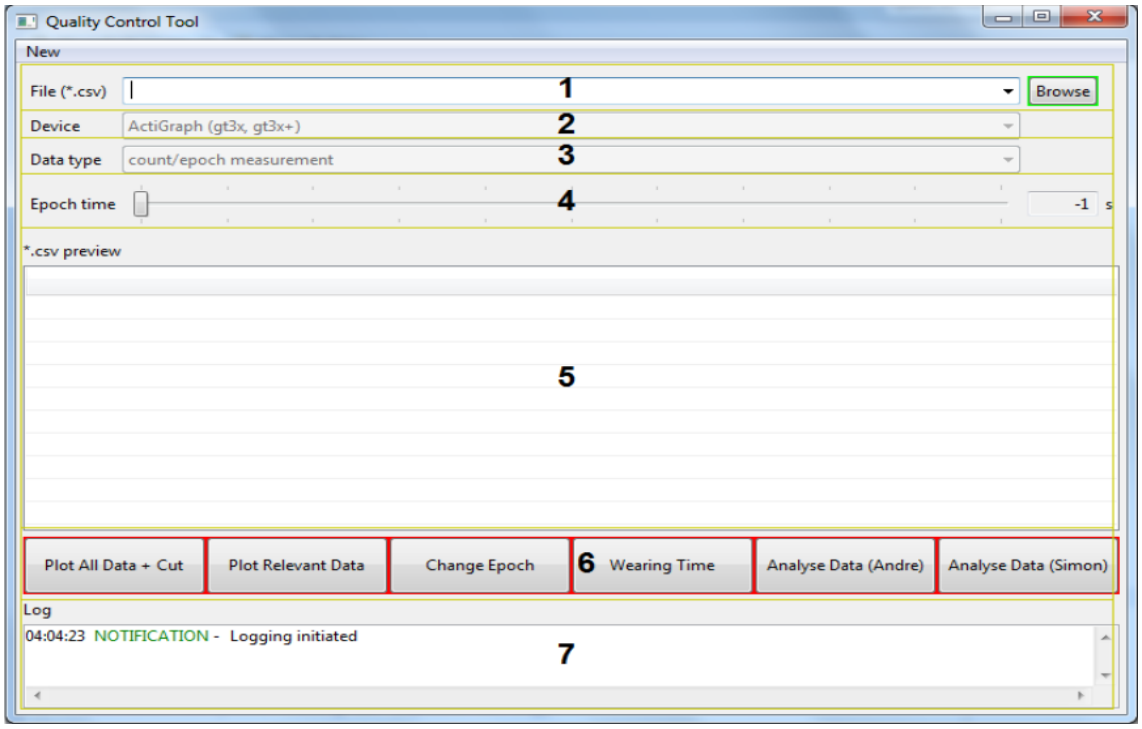


Abbildung 1: GUI des QualityControlTools 1. Version

Die erste Version des QualityControlTools war bislang nur auf den Biosensor ActiGraph GT3X ausgerichtet. Für diesen Sensor ließen sich csv-Dateien des Datentyps *count data* auswerten. Hierzu konnte der Benutzer einen Pfad mit dem entsprechenden Dateinamen übergeben. Beim Betätigen des BROWSE-Buttons (siehe Abb. 1: 1) wurden die Daten der gewählten Datei geladen. Diese wurden dann als TIMESERIES-Format als MATLAB-Datei unter dem Namen *Kora1.mat* gespeichert.

Für den Sensortyp (*Device*) und den Datentyp (*Data type*) war bereits die Vorrichtung für eine Auswahlmöglichkeit des Benutzers vorhanden(siehe Abb. 1: 2 und 3). Diese blieb aber in der ersten Version dem Benutzer gesperrt. Das heißt die Anzeige war automatisch auf *Actigraph(gt3x,gt3x+)* sowie auf *count/epoch measurement* gesetzt und konnte nicht verändert werden.
Zudem ermöglichte es das Tool durch das Betätigen des *Plot All Data + Cut*-Button, aus den eingelesenen Daten eine Grafik zu erstellen. Der, aus diesem Plot, resultierende Graph wurde unter dem Namen *KoraFirstPlot.png* im Dateisystem gespeichert.

In einem nächsten Schritt konnten die gegebenen Daten zugeschnitten werden. Der Benutzer trug hierzu einen Starttag und die Anzahl der Folgetage in ein vorgegebenes Feld ein. Alle Daten, die sich nicht in diesem Zeitraum befanden, wurden weggeschnitten. Die resultierende TIMESERIES überschrieben *Kora1.mat*. Aus den entstandenen Daten wurde wiederum ein Graph erstellt. Dieser wurde unter dem Namen *KoraSecondPlot.png* im Dateisystem abgespeichert.

Als ein so genannter Zwischenschritt war das Verändern der Epochenlänge zu sehen. Dafür musste zunächst die gewünschte Epochenlänge, als ein Vielfaches der bisherigen, eingestellt werden. (siehe Abb. 1: 4) Mit Hilfe von Matrizenberechnungen wurden die Werte eines Intervalls aufsummiert. Das Resultat überschreibt *Kora1.mat*.

Die Daten der csv-Datei ließen sich sowohl mit Hilfe von dem Algorithmus von Andre wie auch nach dem Algorithmus von Simon analysieren. Bei dem erstgenannten wurden für jeden Tag der Eingabedaten eine bestimmte Anzahl an Werte berechnet, welche in einem Vektor gespeichert wurden. Der Vektor ließ sich anschließend in der Datei *finaldata.csv* auffinden.

Auch ließ sich die Tragezeit (siehe Abb. 1: 6) eines Sensors mit Hilfe des Algorithmus von Hecht al 98 berechnen. Resultierend geht daraus ein mit 0 oder 1 befüllten Vektor hervor. Dabei steht 0 für nichtgetragen und 1 für getragen. Aufzufinden war der Vektor sowohl als MATLAB-Datei *wearingTime.mat* und als csv-Datei *wearingTime.csv* vorzufinden. Die csv-Datei bestand aus einer Zeitspalte sowie einer Spalte mit den zugehörigen, errechneten Werten.
Abschließend ermöglichte das Tool eine Auswertung der Daten nach Simon(siehe Abb. 1). Es wurden pro Tag, als auch für den gesamten Datensatz eine bestimmte Anzahl an Daten bestimmt. Die Ergebnisse waren in den Dateien *sum_table_[Klassenname]* und *sum_filt_[Klassenname]* abgelegt. Um Problemen in Java vorzubeugen, wurde am Ende der Workspace geleert.
Für weiteres Interesse verweisen wir Sie auf den Praktikumsbericht *Quality Control Tool for Different Accelerometer Data* von Johannes Eder und Martin Attenkofer.

4 Das erweiterte Tool

Bei der Erweiterung des gerade vorgestellten QualityControlTools kam es, neben den Änderungen in der Funktionalität, auch zu einigen Änderungen in der Oberfläche. Diese werden im Folgenden genauer erläutert. Im Anschluss daran folgt eine detaillierte Erklärung der dahinterstehenden Funktionalitäten und Implementierungen.

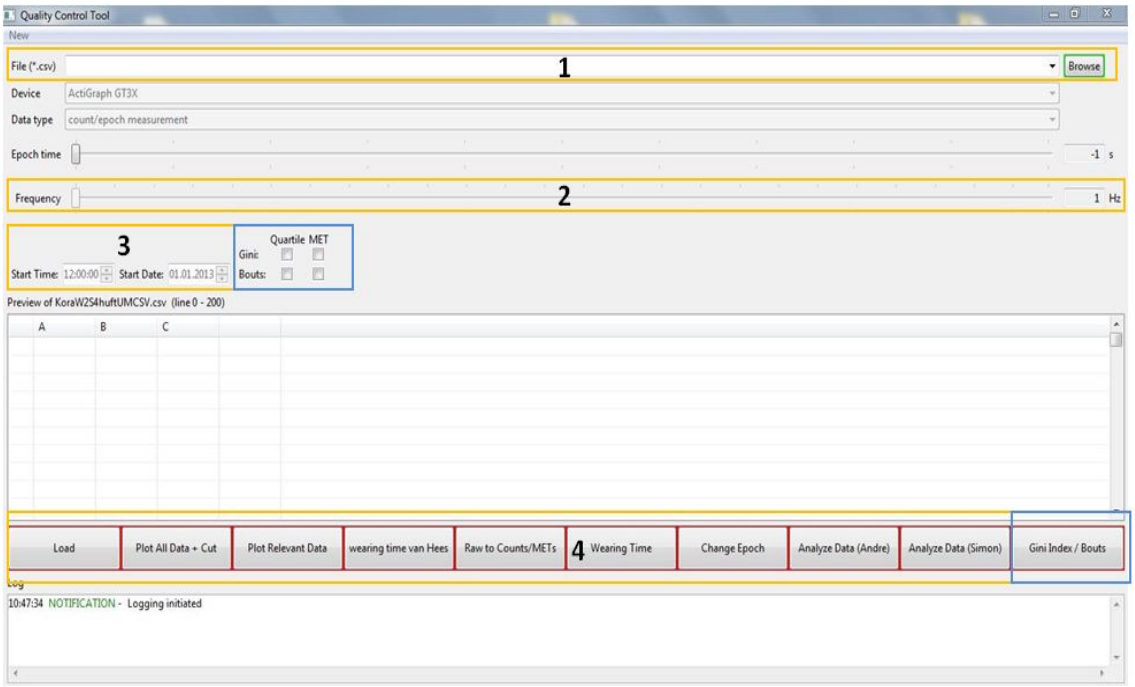


Abbildung 2: GUI des QualityControlTools 2. Version

In Abbildung 2 sind alle Änderungen zur vorherigen Version gekennzeichnet. Dabei wird in dieser Arbeit auf die blau Markierten Felder nicht eingegangen, da diese Gegenstand einer anderen, parallel laufenden, Arbeit sind.

In Feld 1 der Abbildung 2 hat sich für den Betrachter nichts geändert. Allerdings verbirgt sich hinter dem *Browse*-Button eine andere Funktionalität. Statt die Datei zu laden wie in der ersten Version des Tools, wird hier nur noch die Datei ausgewählt und der entsprechende Dateipfad in die Java-Umgebung geladen. Das eigentliche Laden in die Matlab-Umgebung findet erst mit drücken des *Load*-Button (siehe Abb. 2: 4) statt. Da die wesentlichen Änderungen des Tools darin liegen, dass mehrere Sensorarten und auch Rohdaten ausgewertet werden können musste die Funktionalität des *Load*-Buttons ebenfalls erweitert werden. Die genaue Dokumentation dafür ist in Kapitel 4.2 beschrieben.

Wie gerade erwähnt kann das erweiterte Tool auch Rohdaten auswerten. Diese haben keine Epochenlänge in Minuten oder Sekunden wie *count/epoch measurement*, sondern werden mit einer bestimmten Frequenz aufgenommen. Da nicht in allen Rohdaten diese Frequenz angegeben ist, muss der User diese wissen und per Hand eingeben. Dies ist vor Allem bei Shimmer und Somnowatch Daten nötig. Bei ActiGraph GT3X+ und GeneActive Daten wird die Frequenz automatisch ausgelesen (vgl. Kapitel 4.2). Jedoch sollte der User hier immer die Frequenz überprüfen. Damit der User also zugriff auf die Frequenz hat, diese einstellen und überprüfen kann ist die Leise Frequency (siehe Abb. 2: 2) hinzugekommen.

In der ersten Version des Tools mussten die auszuwertenden Daten immer um 12:00 Uhr beginnen. Da die jetzigen Daten auch zu anderen Zeitpunkten anfangen können wurde das Tool um die Felder *StartTime* und *StartDate* erweitert (siehe Abb. 2: 3). Diese muss der User für Somnowatch und Shimmer Daten per Hand eingeben, da in den entsprechenden Dateien weder Informationen über Anfangszeit noch über Anfangsdatum gespeichert sind. Für Daten der anderen Sensoren werden diese Informationen automatisch eingelesen (vgl. Kapitel 4.2).

Im Bereich der Buttons (siehe Abb. 2: 4) sind ebenfalls einige Neuerungen gemacht worden. Zum einen wurde der oben erwähnte *Load*-Button neu erstellt. Zum Anderen wurde für die Auswertung von Rohdaten der Button *wearing Time van Hees* hinzugefügt (vgl. Kapitel 4.5). Um es dem User zu ermöglichen Rohdaten in Epochdaten umzuwandeln wurde zudem der Button *Raw to Counts/METs* neu angelegt (vgl. Kapitel 4.6). Im den nachstehenden Kapiteln wird nun auf die Implementierung der einzelnen Buttons näher eingegangen.

4.1 Der Browse-Button

4.1.1 Aktion des Benutzers

Um mit dem QualityControlTool arbeiten zu können, muss zunächst der *Browse*-Button betätigt werden. Im Zuge dessen muss die zu analysierende csv-Datei aus dem entsprechenden Ordner auf dem Computer des Users ausgewählt werden.

4.1.2 Aktionen des QualityControlTool

Ist die auszuwertende Datei gewählt, stellt das QualityControlTool vorläufige Berechnungen an. Es wird aus der csv-Datei der Name des jeweiligen Sensors ausgelesen und in die GUI des Tools eingetragen. Anhand der Datei wird ebenfalls die Art der Daten eingetragen. Handelt es sich um Rohdaten, erscheint auf der Oberfläche im Feld *Data type raw data measurement*, andernfalls *count/ epoch measurement*. Zusätzlich wird die Frequenz, sofern es sich um Rohdaten handelt, der Sensordaten bestimmt und ebenfalls dem Benutzer mittels der Fre-

quenzleiste dargelegt. In Abbildung 3 sind diese geänderten Einstellungen in der Oberfläche anhand einer Beispieldatei dargestellt.

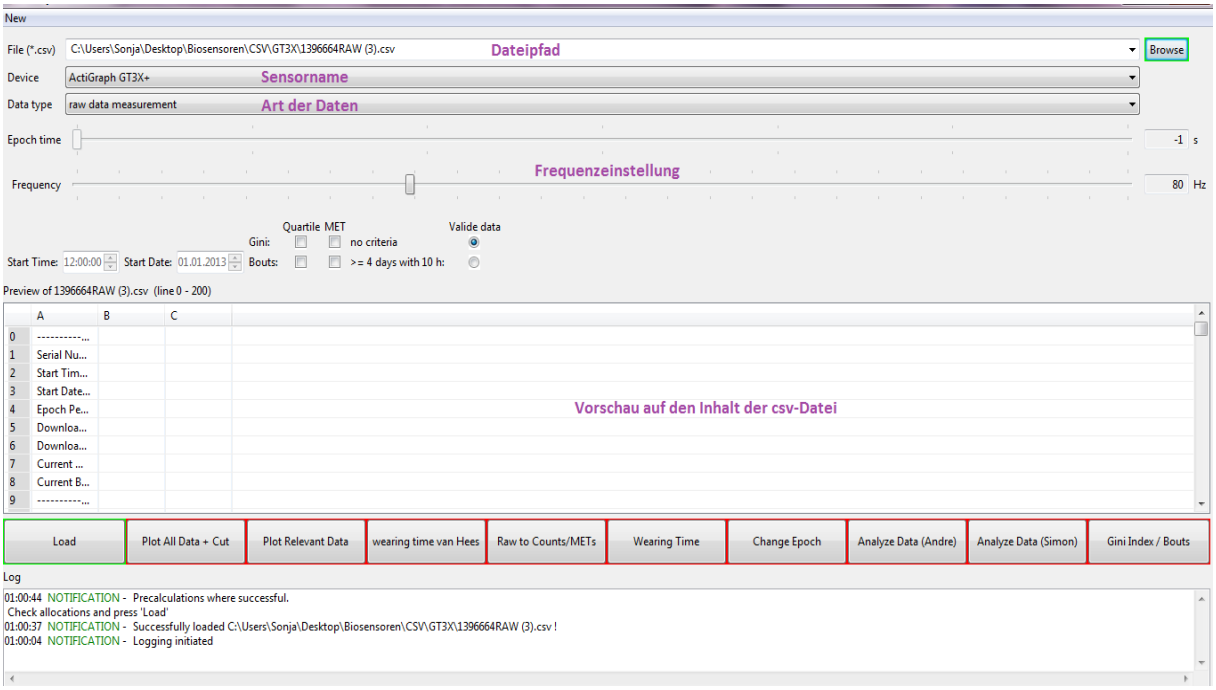


Abbildung 3: GUI nach Betätigen des *Browse*-Buttons

4.1.3 Implementierung

Im JAVA-Code werden in der Klasse *KoraSteps* die Voreinstellungen nach Aktivierung des *Browse*-Button bestimmt. Dies findet in der Methode *public static boolean precalculation(String currentPath, String currentFile, String currentDevice)* statt. Hier wird auf die entsprechenden MATLAB-Funktionen zugegriffen. Den MATLAB-Funktionen wird dabei der Dateipfad und der Dateiname der zuvor ausgewählten csv-Datei übergeben.Es wird die Funktion $FindSensorName(1, file\_path)$ aufgerufen. Entsprechend des Funktionsnamens liest diese aus der Kopfzeile der Datei den Namen des behandelten Sensors. Die Variable für den Namen des Senors wird schließlich in JAVA auf den ermittelten Namen gesetzt und die GUI dementsprechend eingestellt. Misslingt die Suche, behandelt JAVA dies mit einem Auffangen der geworfenen Exception. Zusätzlich wird der User mit der Mitteilung "Unknown Device Type" benachrichtigt, was den Benutzer darauf hinweist den Sensortyp per Hand richtig einzustellen.
Die Frequenz des Sensors wird mit Hilfe der MATLAB-Funktion $FindSensorFrequency(1, file\_path)$ bestimmt. Auch dieses Ergebnis ak-

tualisiert in JAVA die aktuelle Variable für die Frequenz und ist für den Benutzer auf der GUI ersichtlich gemacht. Ein möglicherweise eintretender Fehler wird wiederum mit einer *Exceptionbehandlung* abgefangen und dem User mit der Nachricht "Unknown Frequency!" kenntlich gemacht. Auch hier sollte der Benutzer diese Einstellung per Hand vornehmen.
Erfolgreiches Abschließen aller Vorberechnungen wird mit der Meldung "Precalculations where successful. Check allocations and press "Load"" bestätigt.
Außerdem erscheint eine Vorschau der csv-Datei auf dem Display der Benutzeroberfläche. Dafür ist, wie auch schon in der ersten Version des Tools, die Funktion *private void setupCsvTable(CsvFileLoader csvFile)* in der JAVA-Klasse *MainControlsBackend* zuständig.
Zum Schluss wird der *Load*-Button zur Nutzung freigeschaltet. Dies wird dem Nutzer durch eine grüne Umrandung des Buttons signalisiert.

4.2 Der Load-Button

Der *Load*-Button wird dem Benutzer nur nach dem Ausführen des *Browse*-Vorgangs freigeschaltet.

4.2.1 Möglichkeiten für den Benutzer

Die Trennung des *Browse*-Buttons und des *Load*-Buttons erlaubt dem User vom Programm selbstsändig vorgenommene Einstellungen zu überprüfen und gegebenenfalls zu verifizieren. So lassen sich nach den Voreinstellungen sowohl der Sensorname, als auch die Frequenz, die Datenart und Startzeit und -datum einstellen.

4.2.2 Implementierung

Die endgültigen Einstellungen werden nun erstmals in die MATLAB-Umgebung geladen. In der JAVA-Klasse *KoraSteps* wird die Methode *KoraStep1* ausgeführt. Abhängig von dem jeweiligen Sensor und dem *measurement* werden von hier aus die entsprechenden MATLAB-Funktionen aufgerufen. Für das *count/epoch measurement* existiert in diesem Tool nur eine Implementation für den Sensor ActiGraph GT3X. Für die Sensoren Shimmer, GeneActive und ActiGraph GT3X+ gibt es hingegen lediglich eine Implementierung für *raw data measurement*. Im Falle von falschen Usereinstellungen im Bezug auf die zugrundeliegende Datenart wird dies als Fehler behandelt. Dabei erscheint die Meldung "No code implemented for [sensorname] count/epoch measurement!" beziehungsweise "No code implemented for ActiGraph GT3X raw data measurement".

Für jeden dieser Fälle existiert eine eigene MATLAB-Funktion. Diese weichen in Sensor- und Datentypspezifischen Details voneinander ab. Grund dafür ist der Unterschied im Aufbau der csv-Dateien. So besitzen beispielsweise die GeneActive Dateien einen Dateikopf mit den jeweiligen Informationen

wie Messfrequenz oder Startzeit, wohingegen Somnowatch Dateien direkt mit den gemessenen Beschleunigungsdaten starten. Auch die Anzahl der Spalten weichen voneinander ab.

In Matlab wird in der Funktion $FileLoader\_[sensorname]Kora$ bzw. für $count/epochmeasurement$ $FileLoaderKora\_new$ die Messdaten aus der csv-Datei gefiltert und zeitlich angepasst. Das Ergebnis wird unter einer Datei mit dem Namen $[Filename]\_RAW$ bzw. $[Filename]\_MAT$ gespeichert und ist so für den User jederzeit wieder aufrufbar.

Ist das Laden abgeschlossen, wird sowohl der *PlotAllData + Cut*-Button als auch der *RawToCounts/METs*-Button grün umrandet und somit zur Benutzung freigegeben.

4.3 PlotAllData + Cut -Button

Die geladenen Daten lassen sich in einer Grafik darstellen. In der Grafik sind für Rohdaten drei verschiedene Graphen zu sehen.(siehe Abb. 4) Die dazugehörige Legende beschreibt, welche Bewegungsrichtung der jeweilige Graph darstellt. Zusätzlich ist die Grafik mit dem Sensorname, dem Datentyp, der Startzeit und der Frequenz beschriftet.
An der Darstellung des Plots für Counts hat sich nichts wesentliches im Vergleich zur ersten Version des Tools verändert.

In dem Fenster des *Plots* kann der Benutzer nun Einstellungen bezüglich Startzeit und Dauer der relevanten Daten treffen.

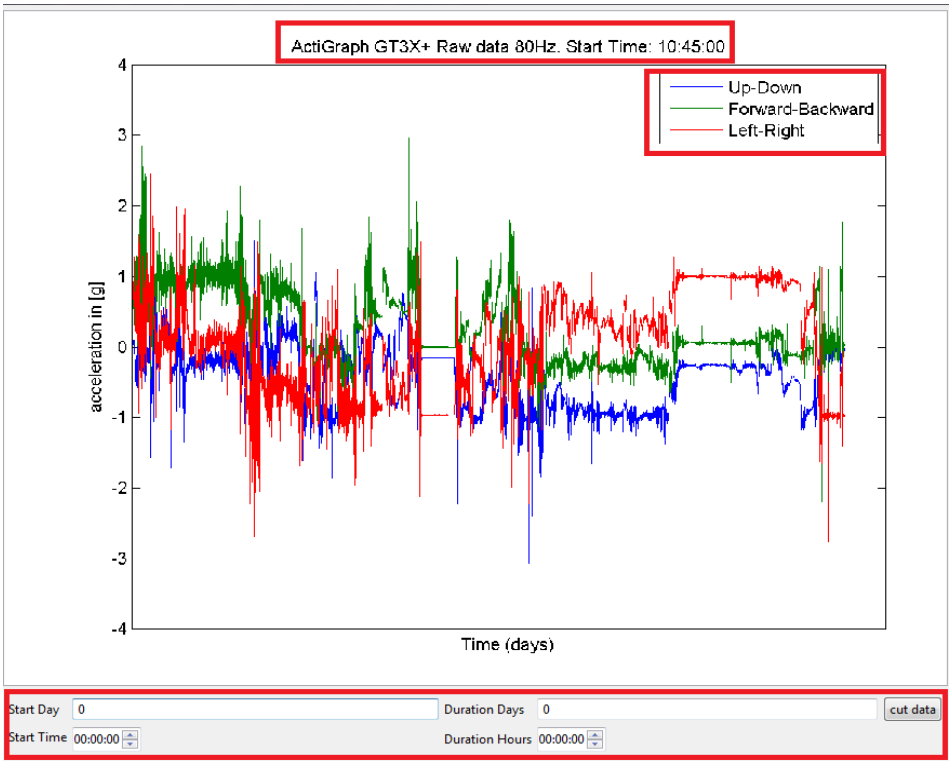


Abbildung 4: Beispiel: *Plot* einer GT3X+ Datei

4.3.1 Implementierung

Auch bei dem *Plotten* wird zunächst von der *KoraSteps.class* auf MATLAB zugegriffen. Von *step2_plotFile* wird je nach *measurement* $cutFirst1\_Raw$ bzw. $cutFirst1\_new$ aufgerufen.

Im Fall von *count/epoch measurement* berechnet die Funktion lediglich ein Schaubild zur einfacheren Visualisierung der Daten, wie schon in der ersten Version des Tools. Die einzigen Änderungen zur vorherigen Version sind eine ausführlichere Titulierung des erstellten Diagramms und eine Anpassung der Beschriftung auf der x-Achse. Auf dieser sind sowohl bei *count/epoch measurement* als auch bei *raw data measurement* die aufgezeichnete Zeit dargestellt. Dabei wird ein Tick immer genau um 12:00 Uhr eines Tages eingezeichnet. Da die Daten im Unterschied zur ersten Version aber auch zu anderen Zeiten starten können musste hier eine zusätzliche Berechnung eingefügt werden.

Bei *raw data measurement* zeigt das Diagramm statt nur einer Achse alle drei gemessenen Achsen. Diese werden, neben allen anderen Berechnungen, ebenfalls in der Methode $cutFirst1\_Raw$, abhängig von dem jeweiligen Sensortyp,

beschriftet.

4.3.2 Cut Data

Wie oben erwähnt gibt es bei den auszuwertenden Daten verschiedene Startzeiten. Damit der User möglichst flexibel ist, was das abschneiden der irrelevanten Daten angeht wurden die zwei Felder *Start Time* und *Duration Hours* hinzugefügt. Neben der schon vorhandenen Funktion im nächsten Schritt nur bestimmte Tage zu plotten, können jetzt auch bestimmte Zeiten ausgewählt werden. So ist es beispielsweise möglich nur die Daten eines Tages von 6:00 Uhr bis 17:00 Uhr zu betrachten.

Ändert der Benutzer nichts an den Einstellungen der Felder *Start Time* und *Duration Hours* so werden die Daten jeweils vom gewünschten Tag ab 00:00 Uhr bis zum letzten, gewünschten Tag um 00:00 Uhr dargestellt. Der letzte Tag berechnet sich dabei aus dem Eintrag im Feld *Start Days* plus dem Eintrag im Feld *Duration Days*. Überschreitet dieser Wert den vom Sensor gemessenen Zeitraum werden alle Daten ab dem Startdatum dargestellt.

Falls das Feld *Duration Days* ebenfalls nicht geändert wird und somit der Eintrag auf 0 bleibt bedeutet dies, dass die auszuwertenden Daten eine Länge von 0 Tagen besitzen. Das führt in weiteren Berechnungen zu Fehlern, da kein Datensatz mehr zur Verfügung steht. Deshalb sollte der User immer darauf achten, beim beschneiden der Daten eine sinnvolle und gültige Zeitangabe zu machen.

4.4 PlotRelevantData-Button

Der *PlotRelevantData-Button* wird dem Benutzer erst nach dem Ausführen des *Plots* aller Daten und dem Schneidevorgang zur Verfügung gestellt. Als Ausgabe erscheint die Grafik der zugeschnittenen, relevanten Daten. Das Fenster entspricht dem im Kapitel 4.3 vorgestellten. Allerdings lässt sich diese Graphik hier nicht weiter schneiden. Demnach lässt dieses Fenster dem Benutzer keine weiteren Angaben zur Dauer bzw. Startzeit zu.

4.4.1 Implementierung

TODO: Umschreiben; Genauer beschreiben!!!!! Wird der Button von dem Benutzer betätigt, wird im JAVA-Prgramm die Methode *KoraStep3* der Klasse *KoraSteps* aufgerufen. Von hier aus wird die MATLAB-Funktion $step3\_cutFile$. Für den Datentyp *raw measurement* wird $cutFirstPlus2\_Raw$ benutzt. Die in der Datei *[sensorname]_RAW.mat* gespeicherten Daten aus dem vorangegangenen Schritt werden *geplottet*. Bei *count/epoch measurement* greift $step3\_cutFile$ auf $cutFirstPlus2\_new$ zu. Für diesen Datentyp werden ebenfalls die zuvor geschnittenen Daten anhand der Datei [*sensorname*].*mat* gezeichnet. Der dabei enstehende *Plot* wird wiederum in [*sensorname*]$\_SecondPlot.png$ beziehungsweise in [*sensorname*]$\_SecondPlot\_RAW.png$ gespeichert. Somit lässt sich die

Grafik von dem User ohne erneute Einstellungen jederzeit aufrufen.

4.5 Wearing time van Hees-Button

Da die Berechnung der Tragezeit nach Hecht nur für Counts möglich ist wurde zusätzlich der Button *Wearing time van Hees* angelegt. Dieser stellt dem Benutzer eine Möglichkeit zur Berechnung der Tragezeiten bei *raw data measurement* zur Verfügung. Hierbei wird der Algorithmus von van Hees (TODO: Referenz zu van Hees Doc einfügen!!!) verwendet. Der Button wird freigeschaltet sobald der Schritt *PlotAllData + Cut* erfolgreich beendet wurde.

4.5.1 Implementierung

Sobald der User den Button gedrückt hat wird über den JAVA-Code die Methode *public static boolean KoraStep5VanHees(String device, String file, int frequency)* in der Klasse *KoraSteps* aufgerufen. Über diese wird dann wiederum die MATLAB-Funktion *wearingTimevH_Raw(sensor, file, hz)* geöffnet und berechnet. Zur Berechnung werden die Daten aus der Datei $[Filename]\_RAW$ verwendet. Das heißt, wurden die irrelevanten Daten bereits mit der Funktion *PlotRelevantData* abgeschnitten, fehlen diese auch in der *wearingTime*-Berechnung. Wurde dieser Schritt (vgl. Kapitel 4.4) bisher noch nicht durchgeführt wird die Tragezeit für alle eingelesenen Daten ausgewertet. Dabei betrachtet der Algorithmus die eingehenden Daten immer in Intervallen von 15 Minuten. Aus diesem Grund müssen die auszuwertenden Daten auch eine Länge von mindestens 15 Minuten aufweisen, andernfalls wir der Fehler "Error while performing van Hees Wearing Time calculation! Data to small!" ausgegeben.

Das Ergebnis wird auf zwei verschiedene Arten abgespeichert: Es wird eine *.mat*-Datei erstellt. In dieser befindet sich eine Matrix B die in den ersten drei Spalten die gemessenen Beschleunigungsdaten des Sensors enthält und in der vierten Spalte jeweils die Information, ob der Sensor laut van Hees getragen wurde oder nicht. Wurde der Sensor getragen ist der Eintrag gleich 1, wurde er nicht getragen ist der Eintrag gleich 0.

Außerdem wird eine csv-Datei abgespeichert. Diese enthält die genaue Zeitangabe jedes berechneten Intervalls und die dazugehörige Trage-Information aus der vierten Spalte der Matrix B. Zu finden sind diese Dateien jeweils unter dem Namen $[Filename]\_wearingTimevHees$ mit der entsprechenden Endung für den Datentyp.

4.6 Raw to Counts/METs-Button

Sobald eine Datei mit dem *Load*-Button (vgl. Kapitel 4.2) in die MATLAB-Umgebung geladen wurde und es sich bei den Daten um *raw data measurement*

handelt besteht die Möglichkeit diese in *count/epoch measurement* umzuwandeln. Diese Funktionalität stellt der Button *Raw to Counts/METs* für den Benutzer zur Verfügung.

4.6.1 Aktionen des QualityControlTool

Der Button wird aktiv, das heißt seine Umrandung wird grün, sobald der *Load*-Vorgang erfolgreich abgeschlossen ist und solange im Feld *Data type* der Typ *raw data measurement* ausgewählt ist. Nachdem die Berechnungen durch diesen Button erfolgreich abgeschlossen wurden wird das Feld *Data type* automatisch auf *count/epoch measurement* gesetzt und alle damit verbundenen Aktionen werden freigegeben. Damit ist der *Raw to Counts/METs*-Button nicht mehr aktiv, also für den User nicht mehr anwählbar.

4.6.2 Implementierung

Drückt der Benutzer den *Raw to Counts/METs*-Button wird in der JAVA-Klasse *KoraSteps* die Methode *public static boolean RawDataToEpoch(String currentPath, String currentFile, String currentDevice, int currentFrequency, String startDate, String startTime)* aufgerufen. Diese öffnet sogleich die MATLAB-Funktion $calculateCounts1(path, file, hz, sensor, d, t)$. Da für die Berechnung der Counts die Daten nochmals direkt aus der csv-Datei gelesen werden wird hier, wie auch bei dem *Load*-Button (Kapitel 4.2), eine Fallunterscheidung zwischen allen Sensortypen gemacht.

In jedem dieser Fälle werden zuerst die Daten aus der angegebenen Datei gelesen und in eine Matrix gespeichert. Dies geschieht jeweils mit einem Sensorspezifischen Funktionsaufruf. Mit dieser Matrix werden in einem zweiten Schritt jeweils die Counts/METs berechnet. Diese Berechnung ist für die Sensoren GeneActive, Shimmer und Somnowatch gleich und wird in der MATLAB-Funktion $rawDataToEpoch3(A, hz, DateTimeNum, file)$ durchgeführt. Für den Sensor ActiGraphGT3X+ muss die Berechnung angepasst werden. Deshalb wird in diesem Fall nur die Funktion $rawDataToEpoch2(path, file, hz)$ aufgerufen. Da diese Funktion also nur für den Sensor ActiGraphGT3X+ verwendet werden kann werden in ihr auch gleich die Daten aus der CSV-Datei eingelesen. Das heißt für diesen Sensor gibt es keinen extra Funktionsaufruf zum Einlesen der Daten.

Bei jedem Sensor werden die Rohdaten automatisch in Counts mit einer Epochenlänge von 60 Sekunden umgewandelt. Die Ergebnisse werden dann auf zwei verschiedene Arten abgespeichert: Als erstes werden sie als timeserie unter dem Namen $[Filename].mat$ gespeichert. Mit dieser Datei ist es dann möglich alle weiteren Berechnungen durchzuführen, die auch schon bei der ersten Version des Tools für *count/epoch measurement* implementiert waren.

Außerdem werden die Counts und METs in einer CSV-Datei unter dem Namen $[Filename]\_Count\_MET.csv$ gespeichert. Diese Datei enthält dann zuerst die genaue Zeitangabe der Daten. Dahinter stehen jeweils, durch Kommata getrennt, die zugehörigen Counts und dann METs.

Damit nach der erfolgreichen Umwandlung die Oberfläche auch das Richtige anzeigt wird im JAVA-Code als letzter Schritt in dieser Berechnung die Epochen-Skala aktualisiert und auf 60 Sekunden gesetzt.

4.7 Analyse nach Andre

4.7.1 Beschreibung

Bei Betätigung des *Analyse (Andre)* -Button werden die Daten nach dem Algorithmus von Andre ausgewertet. Pro Tag der Eingabedaten werden eine bestimmte Anzahl an Werten berechnet. Diese Werte werden in *finaldata.csv* gespeichert. (siehe Abb.5)

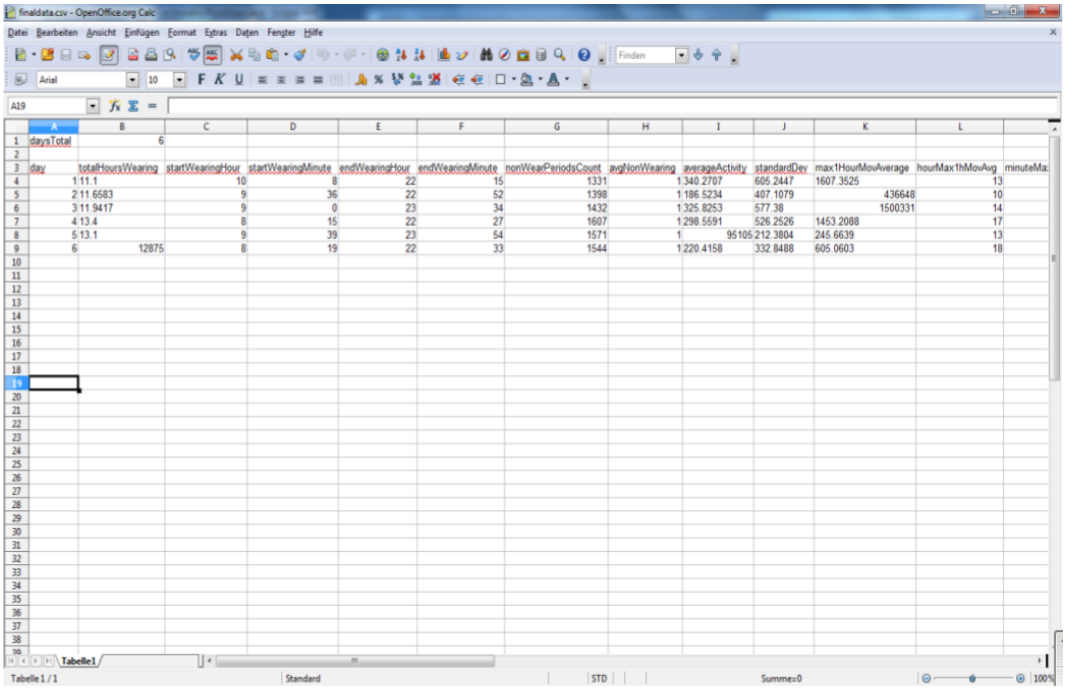


Abbildung 5: csv-Datei des Actigraph (count measurement) nach Analyse von Andre

4.7.2 Implementierung

Zunächst wird in der JAVA-Klasse *KoraSteps* die Methode *KoraStep4* aufgerufen. Diese greift auf die MATLAB-Funktion *step4_analyse* zu, welche auf die Funktion *koradata_new* weiterleitet. Dort findet die eigentliche Berechnung statt.

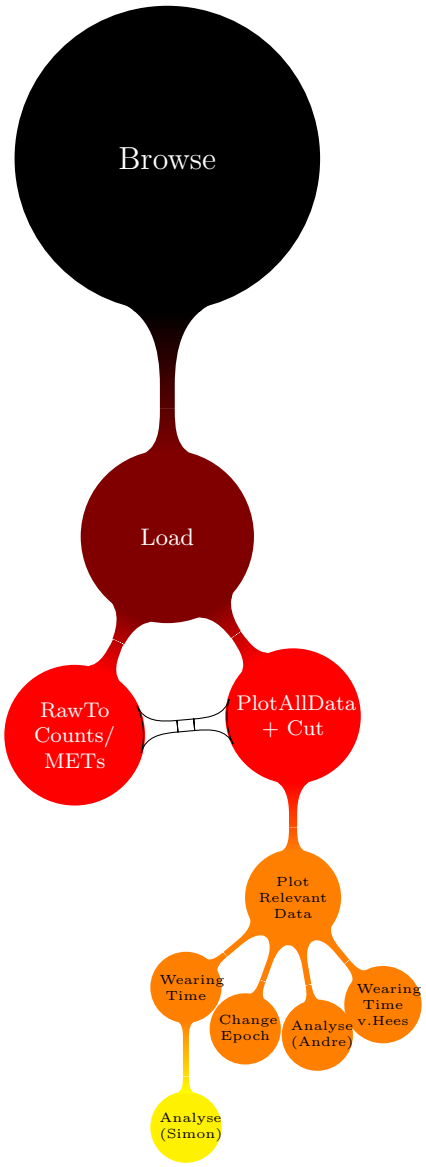
Dabei wurde der Algorithmus so angepasst, dass die Berechnungen nur noch für einen Patienten durchgeführt werden. Zudem sind die Daten für jede Epochenlänge analysierbar. Ein valider Tag beinhaltet mindestens 10 Stunden.

In der Funktion *finaldata2csv* werden die berechneten Daten in der Datei *finaldata.csv* abgespeichert.

4.8 Analyse nach Simon

Bei der Analyse nach Simon hat sich zur ersten Version des Tools nichts verändert. Die Analyse ist weiterhin nur für *count/epoch measurement* für den Sensor Actigraph GT3x ausführbar. Für weitere Informationen zum Aufbau der Implementierung verweisen wir Sie an dieser Stelle auf den Praktkumsbericht "QualityControlTool for Different Accelerometer Data" von Martin Attenkofer und Johannes Eder.

5 Abhängigkeiten im QualityControlTool



6 Beispielablauf einer Analyse von Shimmer Daten

Im Folgenden werden die Funktionen des erweiterten Tools anhand der Auswertung einer Datei veranschaulicht. Dazu dient eine Datei des Sensors Shimmer. Es

wird nicht mehr auf die genaue Implementation eingegangen, sondern lediglich alles beschrieben, was der User für die Benutzung des Tools benötigt.

6.1 Starten des Programms

Der Aufruf des Programms funktioniert ganz einfach über einen Doppelklick auf die Ausführbare .jar-Datei. Sollte das nicht funktionieren, kann das Programm über einen Doppelklick auf die Batch-Datei ebenfalls geöffnet werden.

Damit das Tool auf dem Rechner funktioniert müssen dort einige Vorinstallationen gemacht werden. Neben der Software die bereits für die erste Version des Tools benötigt wird (TODO: Verweis einfügen zu Dokumentation von Jungs), wird jetzt auch die MATLAB **Signal Processing Toolbox** benutzt.

6.2 Laden einer Shimmer Datei

Zuerst wird über den *Browse*-Button (siehe Kapitel 4.1) auf dem Computer eine auszuwertende Datei ausgewählt. In diesem Fall handelt es sich um eine Datei mit Rohdaten von dem Sensortyp Shimmer.

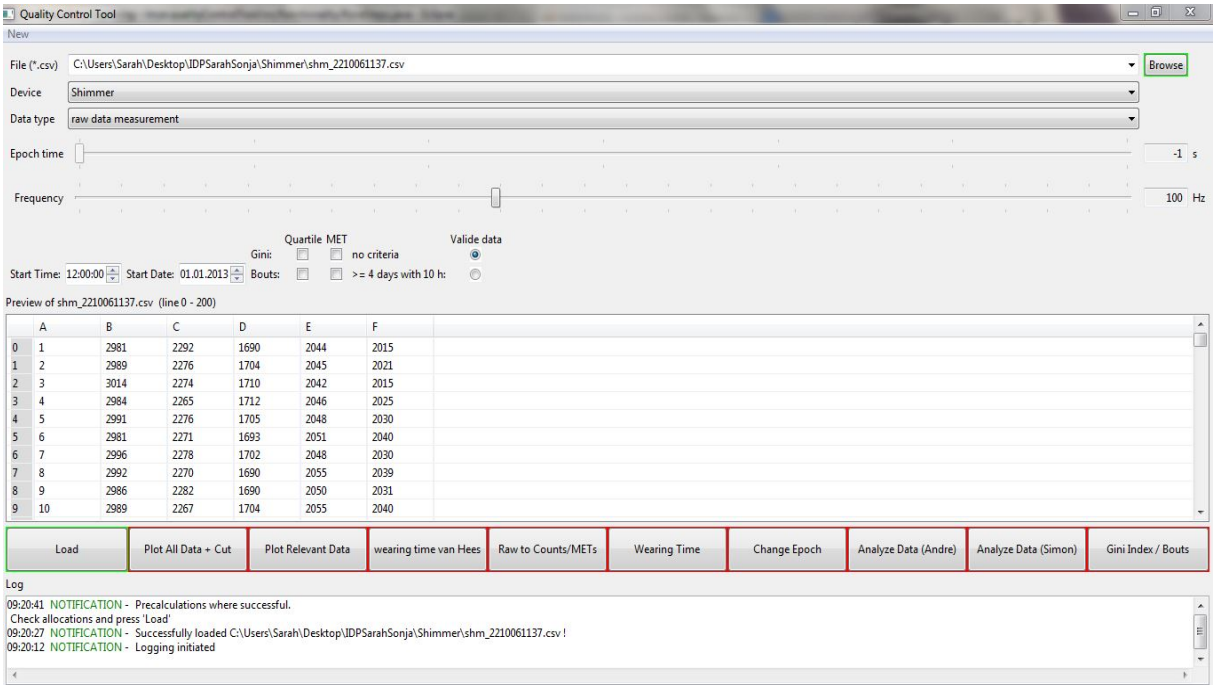


Abbildung 6: Nach auswählen der Datei mittels *Browse*-Button

Diese Informationen werden direkt aus der Datei ausgelesen und erscheinen nach den Vorberechnungen auf der Oberfläche in den jeweiligen Feldern (siehe

Abbildung 6). Somit muss der Nutzer keine Informationen über die Datei haben. Allerdings sollte trotzdem immer überprüft werden ob die Einstellungen wirklich stimmen, da es ansonsten zu Problemen oder falschen Berechnungen führen kann.

Ebenfalls wird die Frequenz automatisch gesetzt. Da diese jedoch nicht in den Daten des Sensortyps Shimmer zu finden sind wurde hier ein Defaultwert von 100 Herz angegeben. Diesen Wert gilt es für den Benutzer unbedingt zu überprüfen, da er auch abweichen kann.

Eine weitere Besonderheit die bei den beiden Sensortypen Shimmer und Somnowatch zusätzliche zu beachten sind, sind die Felder *Start Date* und *Start Time*. Auch diese beiden Informationen sind in den Daten der genannten Sensoren nicht enthalten. Deshalb müssen diese vor drücken des *Load*-Buttons vom Benutzer unbedingt angepasst werden. Geschieht dies nicht, so wie in diesem Beispielablauf, wird jeweils der voreingestellte Defaultwert verwendet.

Hat der Benutzer alle Einstellungen überprüft und gegebenenfalls geändert kann nun der grün umrandete *Load*-Button betätigt werden (siehe Kapitel 4.2). Wurde dieser gedrückt kann es vor Allem bei größeren Daten, abhängig von der jeweiligen Rechnerleistung, zu einer Rechenzeit von mehreren Minuten kommen.

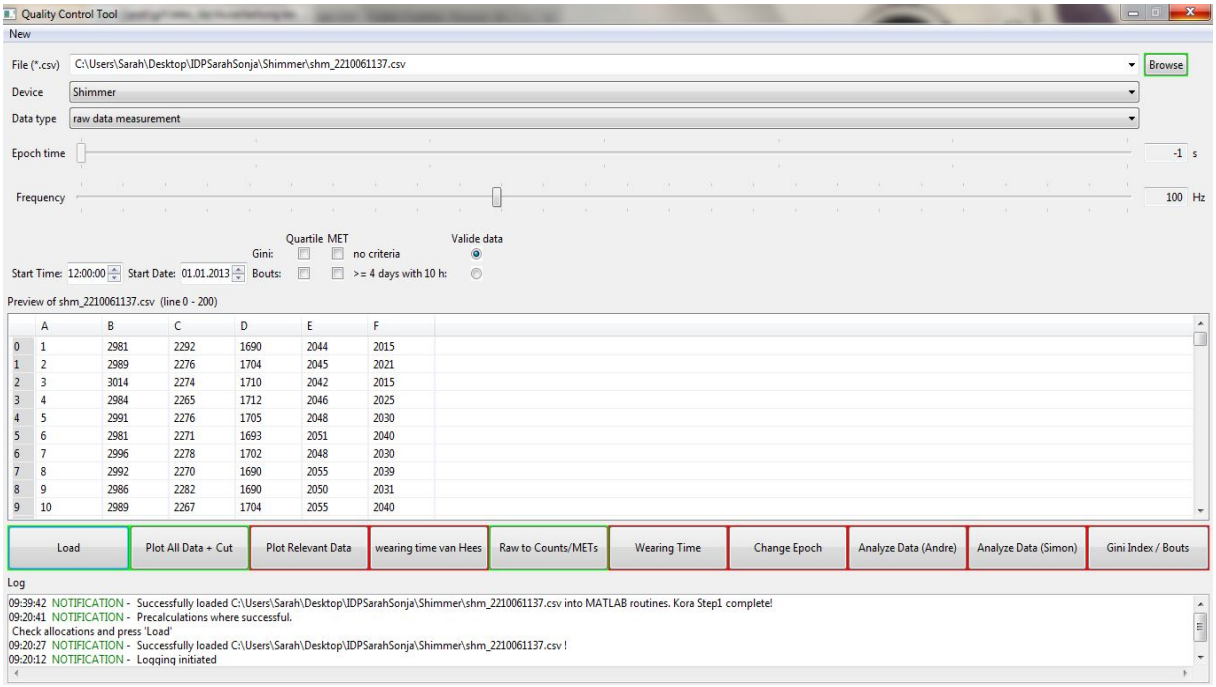


Abbildung 7: Nach laden der Datei mittels *Load*-Button

6.3 Umwandeln von Rohdaten in Counts

In diesem Schritt des Beispielablauf werden nun die Counts aus den Rohdaten berechnet. Natürlich besteht die Möglichkeit zuerst die Rohdaten zu Plotten (PlotAllData + Cut-Button) und zu beschneiden. Allerdings ist es für große Daten sinnvoll zuerst die Counts zu berechnen. Mit diesen dauert die Berechnung des ersten Plots nicht so lange und die Daten können auf die selbe Art beschnitten werden. In den folgenden Abschnitten wird beschrieben wie der Benutzer danach über den *Data type* zurück zu *raw data measurement* wechseln kann um nur noch den Plot für die ausgewählten Rohdaten durchzuführen.

Um die Rohdaten in Counts umzuwandeln muss der User lediglich den, nach dem Laden grün umrandeten, Button *Raw to Counts/METs* (siehe Kapitel 4.6) drücken. Nach der Berechnung wird in der Oberfläche die Auswahl des *Data type* automatisch auf *count/epoch measurement* gesetzt. Nun ist es möglich zwischen beiden Einstellungen hin- und herzuspringen um Berechnungen durchzuführen die es eventuell nur für einen der Datentypen gibt.

Ebenfalls wird nun die Leiste für die Einstellung der Epoche freigegeben und auf 60 Sekunden gesetzt. Gleichzeitig kann der Benutzer nun keine Änderungen mehr in der Frequenz-Leiste mehr machen (vgl. Abb. 8).

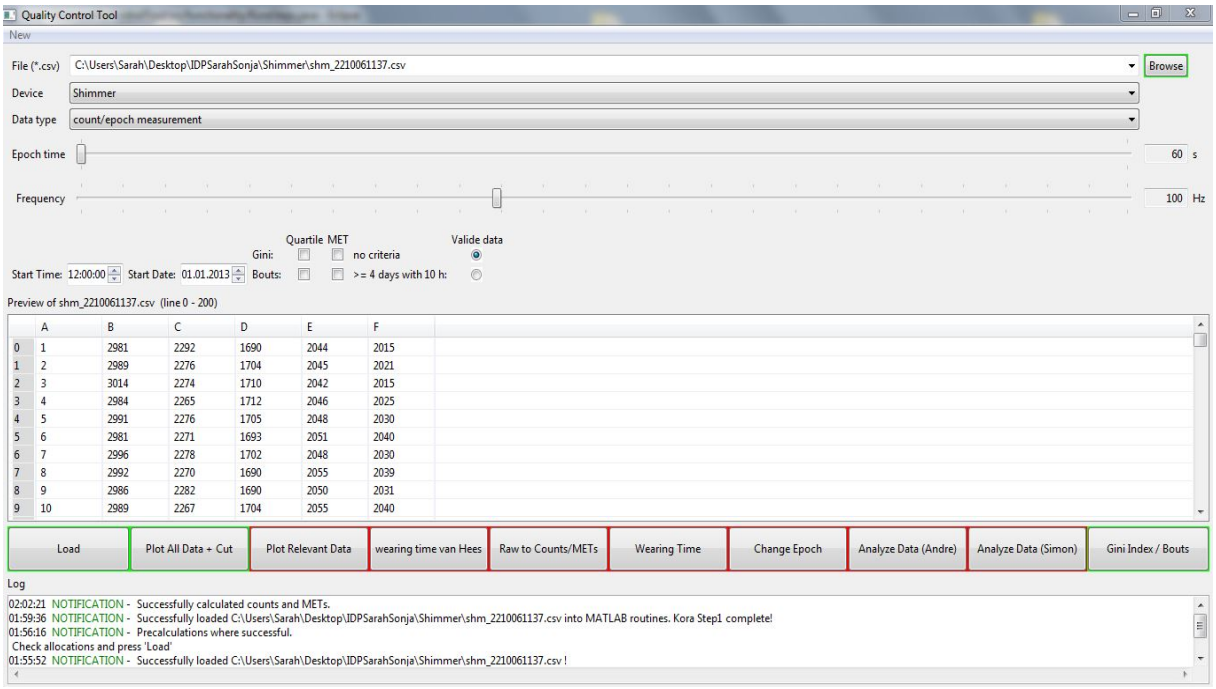


Abbildung 8: Nach berechnen der Counts/METs mittels *Raw to Counts/METs*-Button

6.4 Plotten aller Daten

Um nun die Daten auf einen wesentlichen Teil zu konzentrieren und unwichtige abzuschneiden werden in diesem Schritt erst alle Daten in einer Grafik dargestellt. Anhand dieser Grafik kann der Benutzer dann ein Zeitintervall angeben in dem die wichtigen Daten liegen. Ausschließlich die Daten in diesem Zeitintervall werden dann für weitere Berechnungen benutzt.

Zur Erstellung der Grafik für den ersten Plot wird der *PlotAllData + Cut*-Button gedrückt (siehe Kapitel 4.3). Für die Beispieldaten sieht dieser wie folgt aus:

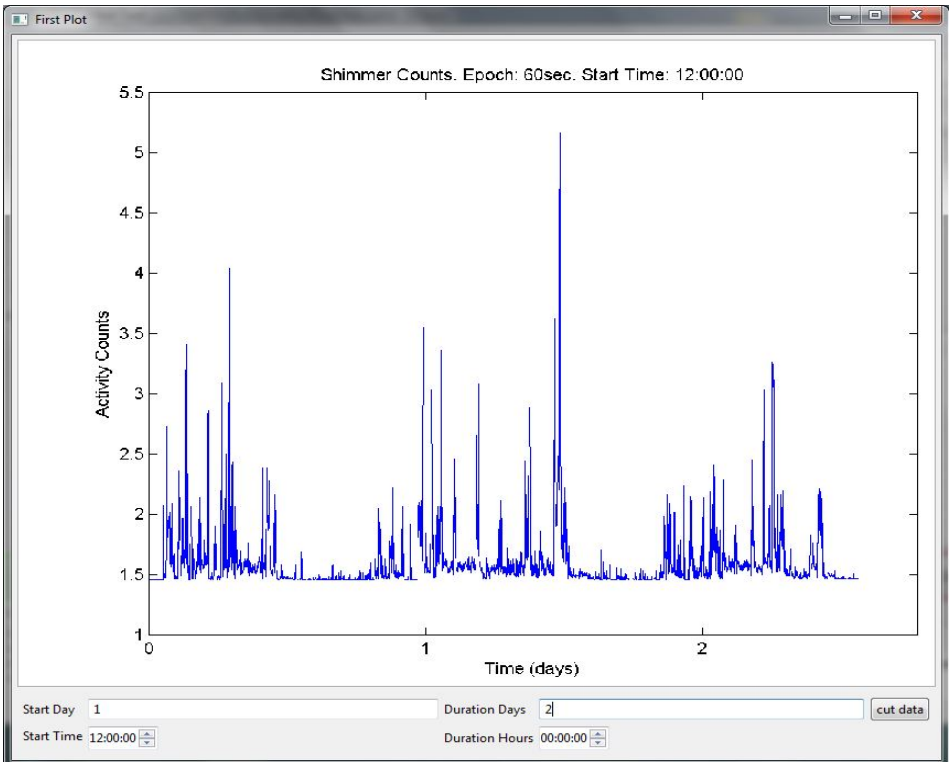


Abbildung 9: Erster Plot

Im unteren Bereich wird nun das Zeitintervall für die zukünftig verwendeten Daten angegeben. In diesem Beispiel wird der Bereich zwischen dem ersten und dem dritten Tag genutzt. Dies ergibt sich aus der *1* im Feld *Start Day* und der *2* im Feld *Duration Days*. Da die Daten in diesem Fall schon vor Ende des dritten Tages aufhören, werden in den folgenden Schritten automatisch alle Daten vom ersten Tag an bis zum Ende verwendet.

Wird vom User nun der Button *PlotRelevantData* angewendet werden nur die gerade gewählten Daten in der Grafik dargestellt (siehe Abb. 10). Ab diesem Schritt werden auch für die Analyse und die Berechnung der Tragezeit nur noch die gewählten Daten verwendet.

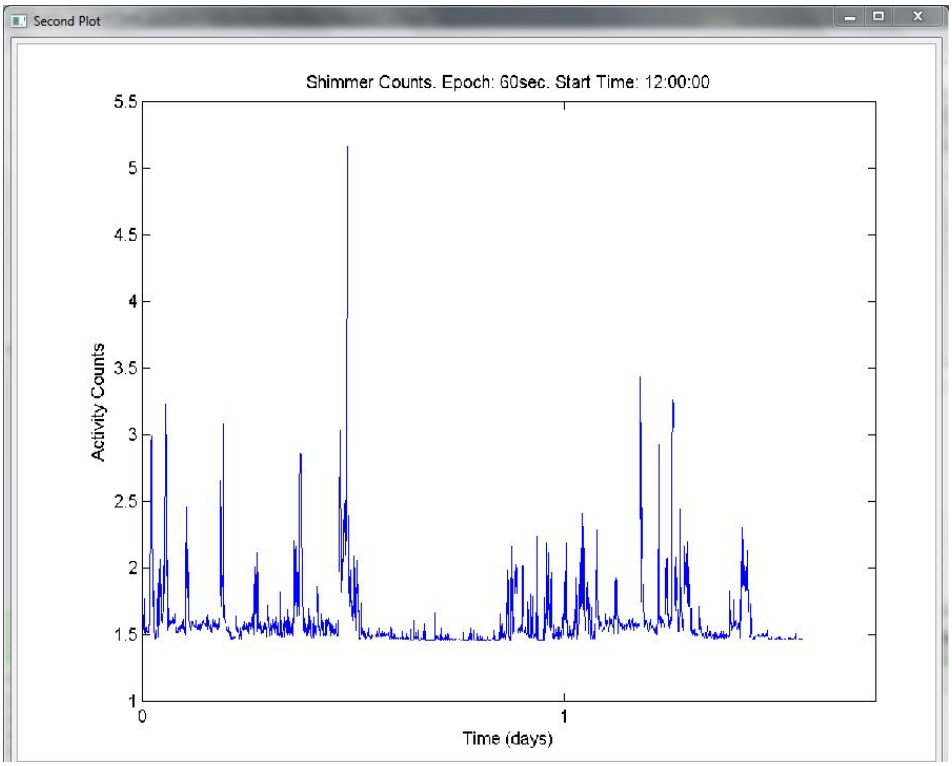


Abbildung 10: Zweiter Plot

6.5 Plotten der relevanten Rohdaten

Damit die im vorherigen Schritt ausgewählten Daten für die Rohdaten genauso beschnitten werden muss der Button *PlotRelevantData* gedrückt werden nachdem in dem Feld *Data type* der Typ *raw data measurement* vom Nutzer eingestellt wurde (vgl. Abb. 11).

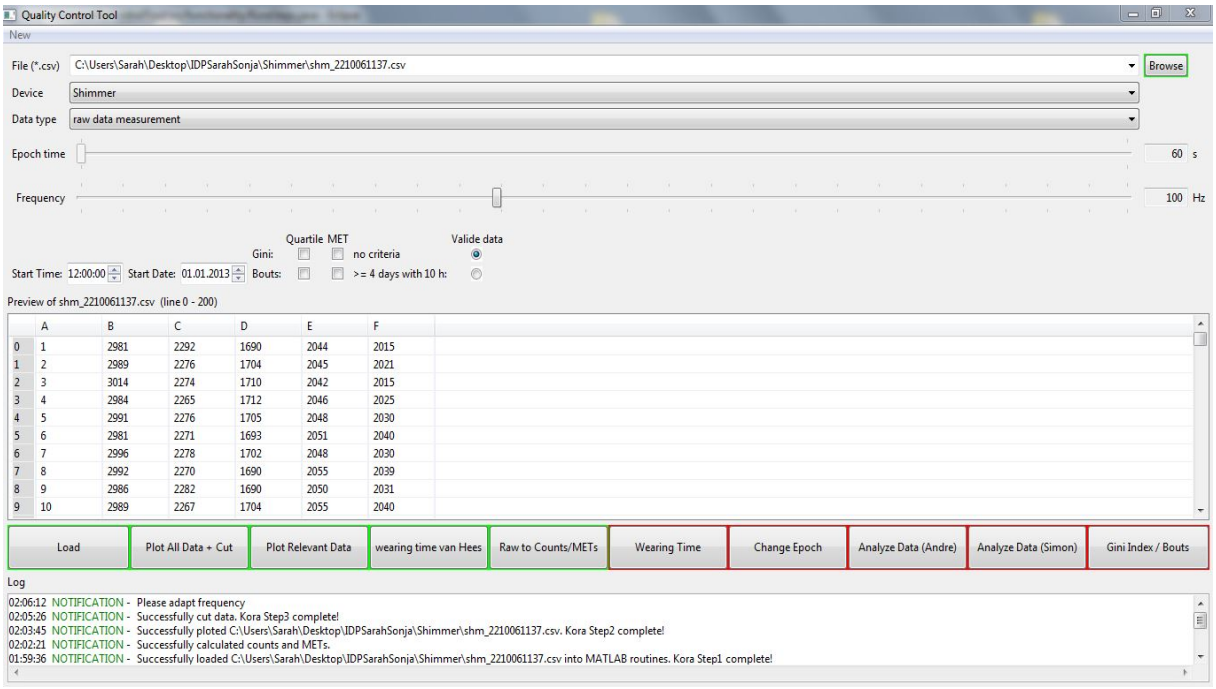


Abbildung 11: Ansicht der Benutzeroberfläche vor Plotten der relevanten Rohdaten

Da jetzt ein Teil der Daten wegfällt funktioniert dieser Schritt schneller als der erste Plot mit Rohdaten, der wie in Kapitel 6.4 beschrieben, umgangen wurde. Dennoch kann es auch hier zu Rechenzeiten von mehreren Minuten kommen. Sind alle Berechnungen abgeschlossen erscheint die Grafik für den zweiten Plot mit Rohdaten:

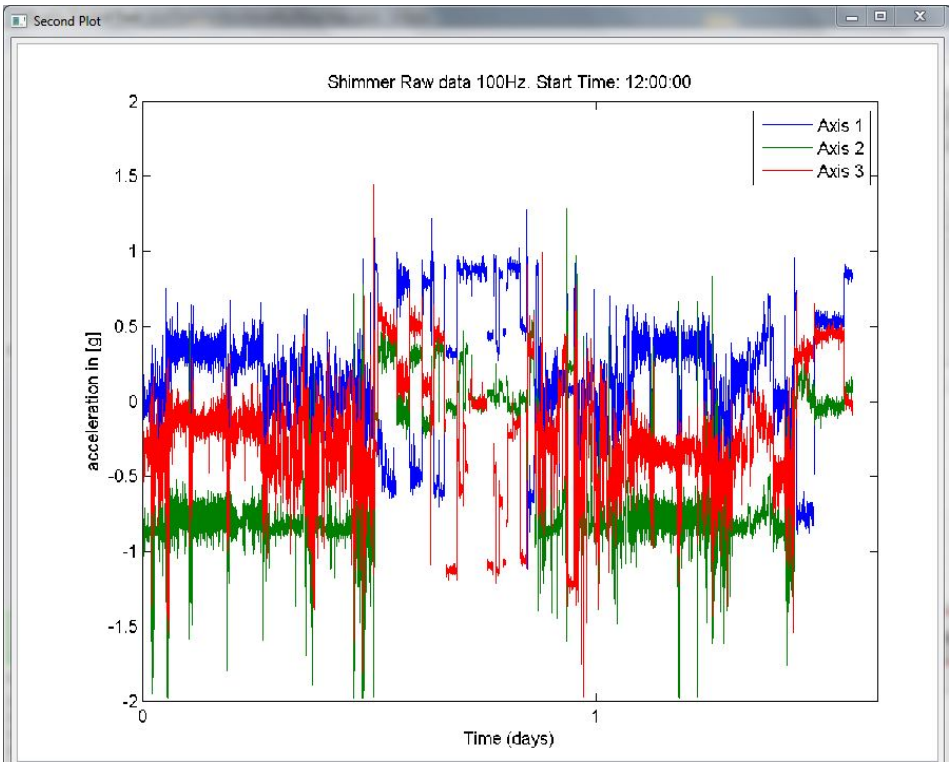


Abbildung 12: Zweiter Plot mit Rohdaten

Nun werden für folgende Anwendungen auch bei Rohdaten nur noch die gewählten, nicht mehr alle, Daten verwendet.

6.6 Berechnung der Tragezeit nach van Hees

TODO: Beschreibung

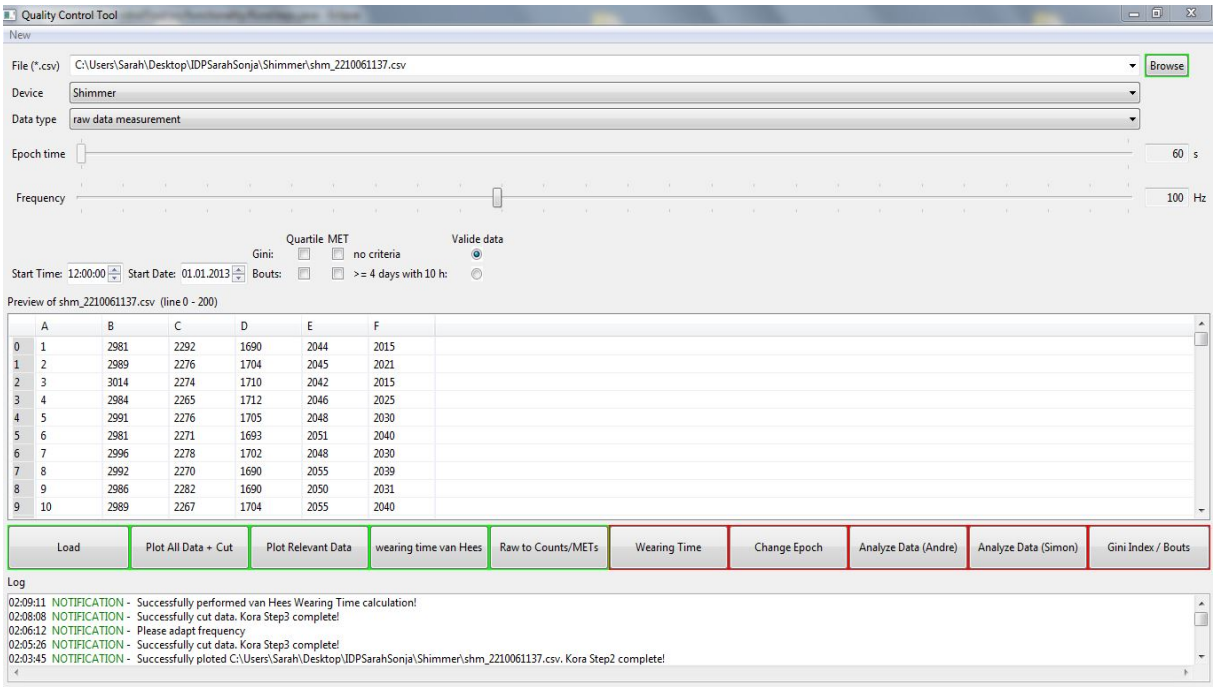


Abbildung 13: Nach der Berechnung der Tragezeit nach van Hees

6.7 Epochenänderung

An dem Schritt zur Änderung der Epoche wurde in der Erweiterung keine Änderung vorgenommen. Mit Hilfe des Schiebereglers (siehe Abb. 14) kann eine beliebige Epoche eingestellt werden.

Abbildung 14: Schieberegler zur Epocheneinstellung

Das Intervall zur Anpassung beginnt bei der von dem Sensor festgelegten initialen Epochenlänge (beispielsweise 2 Sekunden). Das Maximum liegt bei 60 Sekunden. Die Wahl der neuen Länge muss ein vielfaches der initialen Epochenlänge betragen.

6.8 Analyse nach Andre

Die Daten aus der Analyse nach Andre werden in der csv-Datei *finaldata.csv* gespeichert. Wie auch in der vorherigen Version kann es bei dem Öffnen der Datei in *LibreOffice* bzw. *OpenOffice* zu Fehlern bei der Fließkommadarstellung kommen. So wird eine Fließkommazahl teilweise als ganze Zahl dargestellt. Für

weitere Erklärungen verweisen wir Sie auf den Praktikumsbericht der 1. Version.

Bei der Auswertung werden allein Tage mit einer Tragedauer des Sensors von mindestens 10 Stunden berücksichtigt.

6.9 Analyse nach Simon

Die Analyse nach Simon lässt sich erst nach der Berechnung der Tragezeit durchführen. Somit muss noch ehe der Button für die Analyse betätigt werden kann, der *WearingTime*-Button gedrückt worden sein. Die Zeiten sind in einer csv-Datei hinterlegt und können bei der Analyse ausgelesen werden.

Da bei der Erweiterung dieser Schritt nicht bearbeitet wurde, lassen sich nur die *count/epoch measurement* Daten des Sensors Actigraph auswerten.

7 Hinweise zur Benutzung des Tools

Das QualityControlTool läuft auf jedem Computer. Allerdings müssen zunächst die Umgebungen **Java Runtime Environment**[1] , **Matlab Compiler Runtime**[2] und die MATLAB **Signal Processing Toolboxe**[3] installiert werden.

Allerdings gibt es bei der Programmausführung je Computer Unterschiede in der Berechnungsgeschwindigkeit. Dies liegt an der Größe des Arbeitsspeichers eines Computers und an der Anzahl der vorhandenen Kernels. Entsprechend der Kernelanzahl werden die Berechnungen in den MATLAB -Funktionen auf die Kernels verteilt. Je mehr Kernels existieren und belastet werden, desto schneller ist auch das Programm. Dennoch können Berechnungen – vor allem von großen Datenmengen – viel Zeit in Anspruch nehmen. Aus diesem Grund empfehlen wir während der Ausführung des Tools anderweitige gleichzeitige Benutzung des Rechners zu vermeiden bzw. auf das Nötigste zu beschränken. Dazu zählt beispielsweise auch das Ausschalten des Internets. Dadurch kann der Rechner für das Programm vollkommen ausgelastet werden und gewinnt an Geschwindigkeit.

Zudem gibt es für jeden Sensor Besonderheiten, die beachtet werden müssen. Bei den beiden Sensoren Shimmer und Somnowatch muss die Frequenz sowohl zu Beginn eingestellt werden als auch nach dem Wechsel zu den Rohdaten.

/TODO:
- Installationen die vor der Benutzung gemacht werden müssen

[1] mögliche Downloadseite: *http://www.oracle.com/technetwork/java/javase/downloads/index.html*
[2] mögliche Downloadseite: *http://www.mathworks.de/products/compiler/*
[3] mögliche Downloadseite: *http://www.mathworks.de/products/signal/*

- Hinweis: Keine anderen Dinge währenddessen auf dem Rechner machen (ggf. Internet ausschalten), v.a. bei großen Daten
- Besonderheiten für Shimmer, Somnowatch (z.B. Frequenz muss immer wieder neu eingestellt werden wenn zu Rohdaten zurückgewechselt wird!!!), GT3X+, GeneActiv auflisten
Kommentar Sonja: Du mir ist, bei dem letzten Punkt leider nichts weiteres eingefallen. Sobald mir etwas einfällt, werde ich das hinzufügen.

8 mögliche Verbesserungen und Erweiterungen

9 Fazit

Das QualityControlTool erleichtert den Umgang mit den MATLAB -Funktionen. So benötigt der User keine Kenntnisse über das Programmieren mit MATLAB und kann sich allein auf die Auswertungen konzentrieren.

Um dies zu realisieren, wurden zunächst die MATLAB-Funktionen angepasst. Viele Funktionen wurden für alle Sensoren zusammengefasst. Die bestehenden aus der 1.Version wurden an das *raw data measurement* sowie an die verschiedenen Sensoren angepasst bzw. erweitert.(vgl. Kapitel 4) Dabei wurde die grundlegene Hierarchie des Vorgängerversion beibehalten und lediglich um Zusatzschritte erweitert. Mit Hilfe des MATLAB eigenen Deploytools werden diese Funktionen als JAVA-Klassen verfügbar gemacht.

Anschließend wurden in JAVA die entsprechendenden Erweiterungen implementiert, in denen die jeweiliegen MATLAB-Funktionen aufgerufen werden. Dazu wurde die Oberfläche entsprechend angepasst. Durch das Verbinden der Buttons in der Benutzeroberfläche mit den Funktionalitäten, entstand ein einfach benutzbares Programm. Die lässt sich an dem in dieser Ausarbeitung dargestellten Beispielablauf (siehe Kapitel 6).

Es ist ein Programm entstanden, das das Auswerten auf einer großen Bandbreite ermöglicht. Dazu benötigt der Benutzer keinerlei Hintergrundinformationen im Bezug auf die Programmierung. Allerdings benötigt es sehr viel Geduld bei der Analyse. Besonders bei großen Dateien bedarf es für die einzelenen Schritte mehr oder weniger viel Zeit.