T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL FETAL KALP HIZI MONİTÖRİZASYON SİSTEMİ (FKHMS) GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Hakan KANMAZ

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

TEMMUZ 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL FETAL KALP HIZI MONİTÖRİZASYON SİSTEMİ (FKHMS) GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Hakan KANMAZ
Y1513.010014

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Metin ZONTUL

TEMMUZ 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1513.010014** numaralı öğrencisi **Hakan KANMAZ**' ın **"MOBİL FETAL KALP HIZI MONİTORİZASYON SİSTEMİ (FKHMS)"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 04.07.2018 tarih ve 2018/12 sayılı kararıyla oluşturulan jüri tarafından oybirliği ile Tezli Yüksek Lisans tezi olarak ..kabul..edilmiştir.

Öğretim Üyesi Adı Soyadı	İmzası
Tez Savunma Tarihi : 20/07/2018	
1)Tez Danışmanı: Doç. Dr. Metin ZONTUL	.....
2) Jüri Üyesi : Prof. Dr. Ali GÜNEŞ	.....
3) Jüri Üyesi : Doç. Dr. Duygu Çelik ERTUĞRUL	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum **"Mobil Fetal Kalp Hızı Monitörizasyon Sistemi (FKHMS) Geliştirilmesi"** adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya 'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (16/05/2018)

Hakan KANMAZ

Aileme ve Hocama…

ÖNSÖZ

Lisans ve Yüksek lisans eğitimimi almamda büyük katkı sağlayan ve beni teşvik eden, sabrı, bilgi birikimi ve tecrübesiyle çalışmam süresince benden desteğini esirgemeyen, hocalarım sayın Doç.Dr. Metin ZONTUL'a ve sayın Doç. Dr. Duygu ÇELİK ERTUĞRUL'a saygı ve şükranlarımı sunarım. Eğitim hayatım boyunca ve sonrasında hem okul arkadaşım hem meslektaşım olan ve beraber çalışmaktan mutluluk duyduğum dostum Gökhan TAYMAZ'a saygı ve sevgilerimi sunarım.

Temmuz 2018 Hakan KANMAZ

Bilgisayar Mühendisi

İÇİNDEKİLER

KISALTMALAR

FKH : Fetal Kalp Hızı

FKHMS : Fetal Kalp Hızı Monitörizasyon Sistemi

UK : Uterus Kasılması

NST : Non-Stress Test

FHR : Fetal Heart Rate

UC : Uterus Contraction

UUID : Universally Unique Identifier

ŞEKİL LİSTESİ

TABLO LİSTESİ

MOBİL FETAL KALP HIZI MONİTÖRİZASYON SİSTEMİ (FKHMS) GELİŞTİRİLMESİ

ÖZET

Tez, bir mobile entegre Doppler (mDoppler) cihazı aracılığı ile Fetal Kalp Hızı (FKH)'nın uzaktan izlenmesi (örn. ev ortamı)'ni ele almaktadır. Geliştirilen sistemin amacı, özellikle yüksek riskli gebeliklerde fetüsteki risk durumunun, mobil entegre Doppler cihazı ile izlenmesini ve FKH'nın hesaplanmasını sağlamaktır. FKHMS, evdeki gözlem periyodu sırasında mDoppler cihazını anne adayının karnına bağlayarak ve FKH değerlerini dikkate alarak mevcut fetüs koşullarını analiz etmeyi sağlayan kendi çıkarım mekanizmasına sahiptir. Gözlem periyodu sırasında mDoppler'in çıkış sinyali olarak anında toplanan FKH değerleri, anne adayının mobil cihazında görüntülenir ve ayrıca FKHMS'ne veri girişi yapılır. Mevcut fetüs koşulları, sırasıyla fetüsün Normal, Atipik / Uyarı ve Anormal / Alarm koşullarını gösteren Yeşil, Sarı ve Kırmızı olarak kodlanır. Bu nedenle, FKHMS, alarm durumu haline sorumlu ebeveynleri, hekimi ve ambulans servisini bilgilendirir. Bu araştırmada, FKHMS, Dawes / Redman algoritması kullanılarak değerleri takip etmek amacıyla bir vakanın son 10 gebelik haftası için (30 ila 40. haftalar) FKH örnek değerlerini ele almaktadır. FKHMS, literatürde (Elektronik Fetal Monitorizasyon) EFM görevinin görsel analiz algoritmalarından geliştirilmiştir.

Anahtar Kelimeler— Akıllı Mobil Sistemler; Fetal Kalp Hızı İzleme Sistemi; Mobil Entegre Doppler Cihazları; E-Sağlık Sistemleri; Çıkarım

DEVELOPMENT OF MOBILE FETAL HEART RATE MONITORIZATION SYSTEM (FKHMS)

ABSTRACT

This thesis discusses a new approach to Fetal Heart Rate Monitoring System (FHRMS) via a mobile integrated Doppler device (mDoppler) for monitoring Fetal Heart Rate (FHR) remotely (i.e. home). The aim of the developed system is to provide ease of FHR monitoring and computing current fetus risk conditions especially for high-risk pregnancy cases via a mobile integrated Doppler device. FHRMS has its own inferring mechanism that provides to analyze current fetus conditions by considering FHR values through connecting a hand held mDoppler device to labour's abdomen during observing period at home. The instantly-gathered FHR values as output signal of the mDoppler during observation period are displayed on the labour's mobile device monitor and also input data set to the FHRMS. The current fetus conditions are coded as Green, Yellow, and Red status by the FHRMS signifying the Normal, Atypical/Warning, and Abnormal/Alarm conditions of the fetus respectively. Thus FHRMS informs the responsible parents, physician and the ambulance service if alarm status is observed. In this research, FHRMS considers 10 sets of FHR instance values for the last 10 gestation weeks (started at 30th up to 40th weeks) from a subjects labour's that are traced as a case study to track the Dawes / Redman algorithm of the FHRMS. The FHRMS tracing mechanism is developed from visual analysis algorithms of (Electro Fetal Monitoring) EFM task in literature.

Keywords— Smart Mobile Systems; Fetal Heart Rate Monitoring System; Mobile Integrated Doppler Devices; E-health Systems; Inferencing.

1. GİRİŞ

1.1. Çalışma Konusu

Fetal Kalp Hızı Monitorizasyonu (FKHM), fetüsün anne karnındaki sağlık durumunu kontrol etmenin bir yoludur. Ve FKHM'nin önemi giderek daha çok artmaktadır. FKHM doğum ağrıları varken ve doğum sırasında kullanılır. Bazen bu takip gebelikte daha erken dönemlerde yapılabilir. Bu takibin yapılması bir problemin ortaya çıkmasını engellemez. Fakat doktoru fetüsün sağlık durumu hakkında uyarabilir. Özellikle tehlikeli gebeliklerde FKHM daha fazla önem arz etmektedir. Türkiye İstatistik Kurumu verilerine göre bebek ölüm sayısı, revize edilen 2015 yılı verisine göre 13 bin 654 iken 2016 yılında 13 bin 36 oldu. Bin canlı doğum başına düşen bebek ölüm sayısını ifade eden bebek ölüm hızı, 2015 yılında binde 10,2 iken 2016 yılında binde 10 oldu. Diğer bir ifade ile 2016 yılında bin canlı doğum başına 10 bebek ölümü düştü. [1]

Mobil çözümler içinde bulunduğumuz teknoloji çağının en gerekli kavramlarından biridir. Teknolojinin gelişmesiyle beraber gerek kurumlar gerekse de bireyler gündelik hayatlarını kolaylaştıran, vakitten tasarruf sağlayan, eskiye nazaran daha ucuz ve kaliteli araçlar kullanmaya başlamışlardır. Mobil teknolojiler bu çözümlerin en popüler olanıdır. Hepimiz gündelik yaşantımızda mobil teknolojileri kullanır hale geldik. Bilişim sektörü başta olmak üzere, eğitim, enerji, finans, gıda, inşaat, ulaştırma, lojistik, haberleşme ve sağlık sektörü gibi dünyanın önde gelen sektörlerinin vazgeçilmezi olmaya başlamıştır.

Sağlık sektöründe de son yıllarda hatırı sayılır bir şekilde mobil teknolojiler sıkça kullanmaya başlanmıştır. Mobil teknolojilerin kullanılmasıyla "Mobil Sağlık" (m-Sağlık) diyeceğimiz bir kavram da ortaya çıkmıştır. Dünya genelinde m-Sağlık kavramı hızla yayılmaktadır. Ülkemizde de m-Sağlık alanında birçok umutlandırıcı ve sevindirici hadiseler yaşanmaktadır. Bunlardan bazıları:

- **Uzaktan teşhis:**

Yaşlılar, kronik hastalar, engelli bireyler için oldukça önemli olan uzaktan teşhis sağlık alanında yaşanan teknolojik gelişmelerin en önemlilerinden. 4G mobil internet

devrimi ile beraber veri iletiminin de hızlanmasıyla özellikle acil durumların tespit edilmesi, ekip yönlendirme hizmetleri, belirli değerlerin merkezi sisteme aktarılması gibi konuları içeren uzaktan teşhis en önemli gelişmelerden biri olarak öne çıkıyor.

- **Mobil Hastane Çözümleri:**

Ülkemizde de sağlık bakanlığı ve bir GSM operatörünün iş birliği ile hayata geçirilen model temelde yine verilerin hızlı bir biçimde merkezi sağlık sistemine gönderilebiliyor. Özellikle acil vakalarda hasta henüz yoldayken verilerin iletilmesi büyük önem taşıyor ve zaman kaybını en aza indiriyor.

- **Mobil Tahlil:**

Mobil Tahlil ile kan analizi gibi tahliller ile sıtma, AIDS ve tüberküloz gibi hastalıklara cep telefonu kullanılarak uzaktan tanı ve teşhis konulmasını mümkün oluyor. Yine ülkemizde de hayata geçirilen ve bir GSM operatörünün Sağlık Bakanlığı ile yaptığı iş birliği çerçevesinde sağlık ocağı, klinikler gibi küçük birimlerin gelişmiş sağlık merkezleriyle veri iş birliği yapmasını sağlayan model oldukça verimli olmuştur.

- **Mobil Kan Tahlili:**

Küçük bir kan tahlili cihazı aracılığıyla, rahatsız hisseden bir kişinin sağlık merkezlerine gidip kan tahlili yaptırmasına gerek kalmadan, analizler evde veya ambulansta hızlı ve kolay bir biçimde yapılabiliyor ve mobil ortamda çok hızlı bir şekilde hastaneye iletilebiliyor. Doktorların kullanımında olan sistem doktorlara ambulansta, evde herhangi bir yerde profesyonel laboratuvar kalitesinde sonuçlar alma imkânı veriyor.

- **Mobil kayıt:**

Doktorlara özellikle günlük hasta turları arasında mobil cihazı kullanarak hasta kaydı, takibi, güncelleme yapma imkânı veren yazılımlar ve mobil uygulamalar sağlık alanında teknolojinin öne çıktığı alanlardan bir diğer olarak öne çıkıyor. Hasta ziyaretleri ya da herhangi bir mobilite gerektiren durumda toplanan bilgiler tek tek yazmak yerine sadece telefona girilebilir. Böylece uygulama zaman tasarrufu ve verimlilik sağlar. [2]

m-Sağlık uygulamaları, sağlık sektörünün her branşında sıklıkla karşımıza çıkmaktadır. Özellikle uzaktan tanı-teşhis ve takip sistemleri, yatan hasta takip sistemleri, ayaktan hasta takip sistemleri gibi çok popüler bir kullanım alanına sahiptir. Örneğin bir diyabet hastası artık günlük açlık ve tokluk şeker ölçümlerini mobil uygulaması sayesinde doktoru ile paylaşıp kendi takibini yapabilmektedir. Yine mobil uygulamalarda kullanılan algoritmalar neticesinde kanındaki glukoz miktarına göre hasta bireye ikaz verip çok aksi bir durum varsa doktoruna bu durumu otomatik olarak iletebilmektedir. Aynı şekilde gebelikte sıklıkla karşılaşılan gestasyonel diyabeti olan anne adayları için glukoz ölçümlerinin anlık olarak hastane sistemi ile paylaşılması çok önemli bir husustur. Bu senaryoda yine mobil sağlık uygulamalarının önemi anlaşılmaktadır.

Daha çok hareket kabiliyeti kısıtlı olan kişilerin sağlığı, yaşlı sağlığı ve çocuk sağlığı ile ilgili olan uygulamalar ön plana çıkmaktadır. Bu sebeple özel sağlık kuruluşları olmak üzere bazı sağlık kuruluşları bu hasta profiline uyan hastalar için bazı mobil uygulamalar kullanmaktadır. Tansiyon takibi, şeker takibi, çocuklar için ateş takibi, deri alerjisine bağlı hastalıkların takibi gibi birçok takip sistemi kullanılmaktadır.

1.2. Tezin Amacı

Değişik monitorizasyon yöntemleri vardır. Her yöntemin kendi avantajları veya dezavantajları vardır. Bu takibin yapılması bebeğinizin sağlıklı doğmasına yardım etmektedir. Bu fetüs sağlığı için yapılması gereken rutin bir işlemdir. FKHM fetüsün sağlık durumu anlamanın en kolay yoludur. [3]

FKHM, elektronik sinyalleri elde eden, işleyen ve gösteren bir yöntemdir. Bu sonuçların önemini anlamak için, monitörün neyi hesaplayıp neyi hesaplayamayacağını bilmek önemlidir. Günümüzde eksternal monitorizasyon en sık kullanılan FKHM yöntemidir. [4]

Hastaneye yatırılan tüm hastalara kısa bir süre için monitorizasyon yapılmaktadır. Anne adaylarının büyük bir kısmına gebelik süresince bu takip yapılmaktadır. Aşağıdaki durumlarda doğum eyleminin tamamı boyunca monitorizasyon yapılır:

- Doğum eylemini indüklemek için suni sancı veriliyorsa,
- Diyabetes mellitus (şeker hastalığı), Gestasyonel Diyabet (Gebelik Şekeri), Hipertansiyon, Kalp Hastalığı varsa,
- Epidural anestezi ile doğum yapılacaksa,
- Varolan gebelikte herhangi bir sorun varsa veya önceki doğumlarda sorun yaşandıysa

Teknolojinin, elektronik sistemlerin (özellikle aktif ve pasif devre elemanlarının boyutlarının küçülmesi ve tümleşik devre üretimlerinin artması sonucu) ve yazılım sistemlerinin gelişmesi ile birlikte gelecekte hastaların sağlık kuruluşlarına gelmeden evde kullanabilecekleri mobil sistemlerle tanı ve teşhisin yapılabilmesinin mümkün olacağı öngörülmekte ve bu yönde çalışmalar artarak devam etmektedir. Bu teknolojiler m-Sağlık adı altında birleşerek bireylerdeki rahatsızlıkların teşhis, tanı ve tedavi süreçlerinde yardımcı olmasını sağlamakta ve hastalara sunulan hizmetin verimliliğini arttırmaktadır. Özellikle, planlı takip süreçlerinde, m-Sağlık sistemleri kullanımıyla birlikte, bireylerin her türlü sağlık parametrelerinin takibi, mevcut hastalıklarının seyri hakkında bilgi veren ölçüm sonuçlarını içeren sayısal bilgiler dijital ortama aktarılmış olmaktadır. Mobile entegre FKHM ile birlikte, anne ve bebeğin doğum süreci içerisinde, gebeliğin seyrinde gelişebilecek risklerin önceden belirlenmesi, tanımlanması ve müdahalesi sağlanarak anne ve bebek ölümleri önlenebilecektir. [5]

Tezimizin amacı tam bu noktada Doppler aleti aracılığı ile dönüştürülmiş elektronik sinyallerin mobile entegre FKHMS ile alınması, işlenmesi ve yorumlanmasıdır. Hastane ortamında gerçekleştirilen bu ölçümlerin, tehlikeli gebeliklerde ev ortamında da elde edilebilmesi amaçlanmaktadır. Tezin amacına uygun olarak ev ortamında tehlikeli gebeliğe sahip anne adayı kendi kendine ölçümlerini yapabilecektir ve doktoru ile anlık olarak elde ettiği verileri paylaşabilecektir.

1.3. Literatür Çalışması

m-Sağlık sektörünün son yıllarda kazanmış olduğu ivme ile beraber FKHMS alanında bir çok çalışmayı da beraberinde getirmiştir. Özellikle mobil teknolojilerin sağlık sektörü ile harmanlanmasıyla beraber mobil platformlardaki sağlık uygulamalarının gelişimi olumlu yönde etkilenmiştir. Hastanelerde kullanılmakta olan biyomedikal

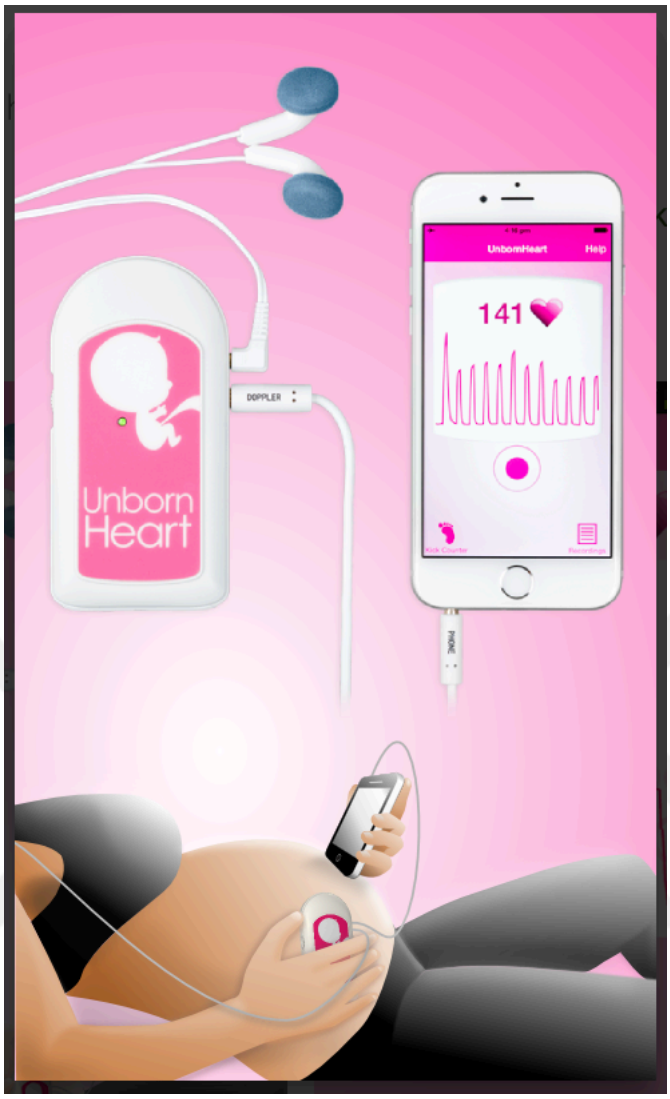
cihazlar artık ev ortamına uygun hale getirilmektedir. Biyomedikal cihazların boyutlarının küçülmesi ve ev ortamında kullanılabilme olasılığının oluşmasıyla birlikte mobil çözümlere de ihtiyaç duyulmaya başlanmıştır.

2013 yılında Çekya Teknik Üniversitesi Elektrik Elektronik Fakültesi öğrencileri ve akademisyenleri olan Luka´s Zach, Vaclav Chudacek, Jakub Kuzılek, Jırı Spilka, Michal Huptych, Miroslav Bursa ve Lenka Lhotska, "Mobile CTG – Fetal Heart Rate Assessment Using Android Platform" isimli çalışmasında fetüsün kalp atım hızını doppler cihazından android platformuna taşımışlardır ve Şekil 1.1de gösterilen anlamlı bir traseyi oluşturabilmişlerdir. [6]

Şekil 1.1 : Fetal Heart Rate Assessment Using Android Platform

2011 yılında Chalmers Teknoloji Üniversitesi öğrencisi Susanne Andersson'un "Acceleration and Deceleration Detection and Baseline Estimation" isimli yüksek lisans tezinde herhangi bir Doppler aletinden alınan elektronik sinyallerin Dawes-Redman algoritması kullanılarak yorumlanmasını anlatmıştır. Dawes-Redman algoritmasına bölüm 3.2'de geniş olarak yer verilmiştir. EK 1'de Dawes-Redman algoritmasına ait diyagrama yer verilmiştir. [7]

Yine 2013 yılında Finlandiya merkezli Odosoft yazılım firması tarafından "UnbornHeart" isimli mobil uygulamasını geliştirmiş olup anne adayına fetüsün kalp atış değerini göstermekle birlikte fetüsün kalp atış sesini dinletme özelliğine sahiptir. Şekil 1.2'deki görselde gösterilmiştir. [8]



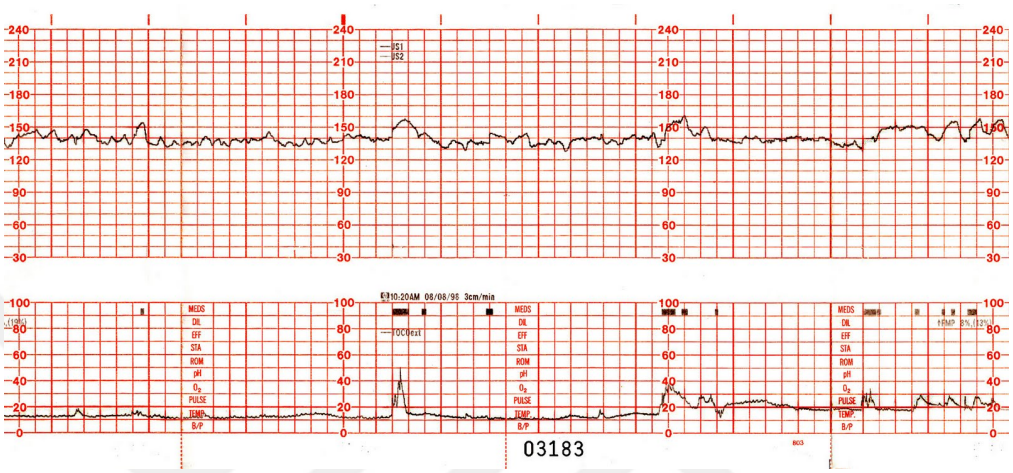
Şekil 1.2 : UnbornHeart Fetal Doppler

Görüldüğü üzere dünya genelinde doğum öncesi dönem ile alakalı birçok mobil çözüm üretilmeye başlanmıştır ve üretilmektedir. Bu bağlamda tezimizde yapılan çalışmalara ek olarak verilerin alınması, işlenmesi, yorumlanıp sağlık kuruluşu ile anlık paylaşılması anlatılacaktır.

1.4. Temel Ölçütler

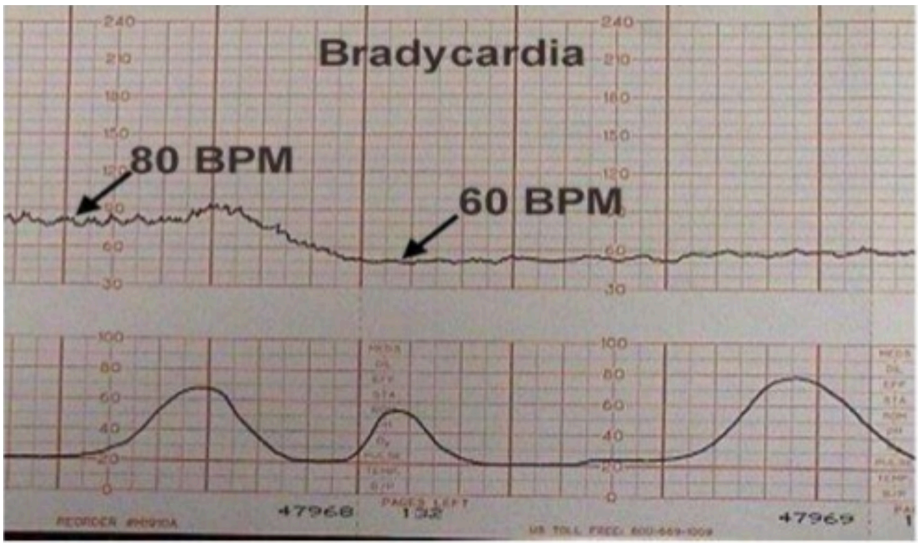
1.4.1. Bazal Fetal Kalp Hızı/Bazal Fetal Düzey

Kontraksiyonlar dışında ölçülen ve en az 10 dakikalık süre içinde belirlenen fetusun ortalama kalp atım hızıdır (Şekil 1.3). FKH'nın referans değeri; 110-160/dk'dır. [9]

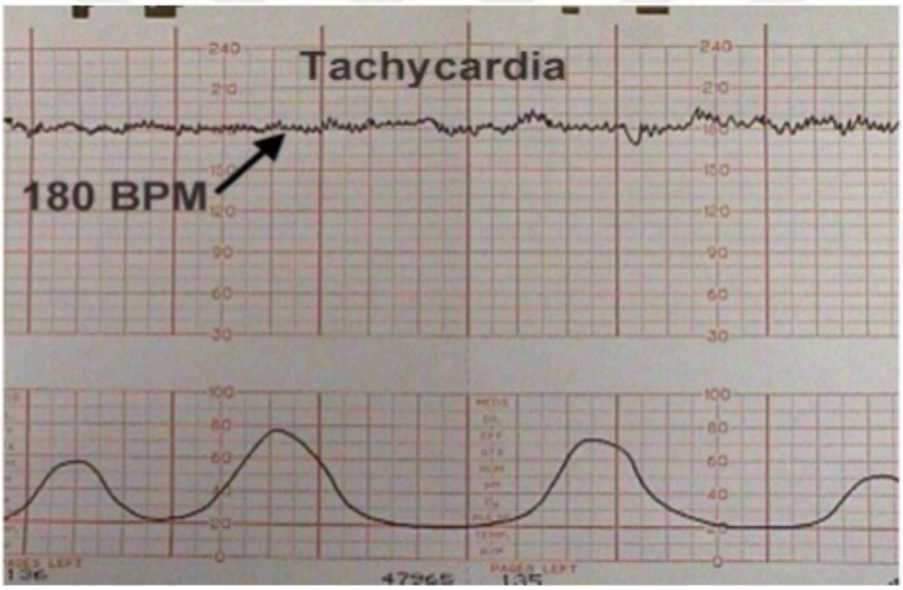


Şekil 1.3 : Bazal fetal kalp hızı ve deselerasyon

FKH'nın 110'un altı "bradikardi", 160'ın üstü "taşikardi" olarak tanımlanır (Şekil 1.4 ve Şekil 1.5). Bradikardinin ve taşikardinin hem anneye hem fetusa ait nedenleri bulunmaktadır. [10] Bradikardinin anneye ait başlıca nedenleri: Hipertonik kontraksiyonlar, plasenta previa, plasenta dekolman, preeklampsi, annenin ilaç kullanımı ve annenin pozisyonudur (özellikle sırt üstü pozisyon) Fetusa ait bradikardinin ana nedenleri ise: Asfiksi, umblikal kord basısı, fetal hipoksi ve fetal başın vagal stimülasyonudur. [11]



Şekil 1.4 : Fetal bradikardi



Şekil 1.5 : Fetal taşikardi

1.4.2. Akselerasyon (Hızlanma)

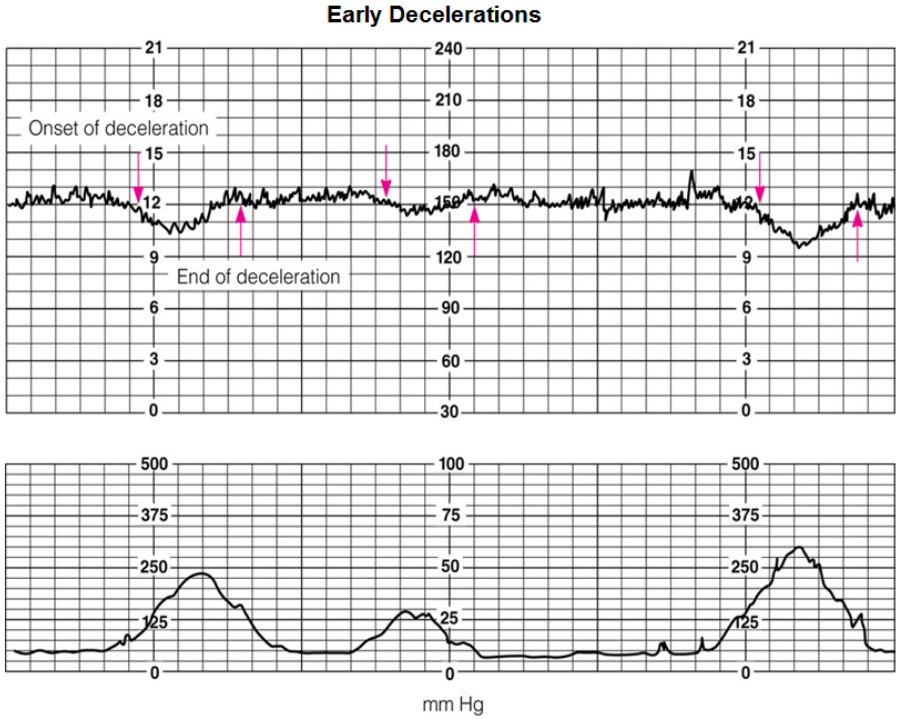
FKH'daki artmayı tanımlar. FKH'nın; bazal fetal düzeyden dakikada en az 15 atım fazla olması ve bu artışın en az 15 saniye sürmesidir (Şekil 1.4). Akselerasyonun olmaması; hipoksik asidoz ve fetüs anomalisini düşündürtmektedir. Otuzikinci haftadan küçük gebeliklerde akselerasyon; FKH'nın bazal fetal düzeyden dakikada en az 10 atımdan fazla olması ve yine en az 10 saniye sürmesi olarak değerlendirilmektedir. [12]

1.4.3. Deselerasyon (Yavaşlama)

Kontraksiyon sırasında, fetal kalp hızında görülen yavaşlamadır. Fetal bazal düzeye göre, FKH'da 15-20 atımlık düşüşlerdir. Erken, geç, değişken ve uzamış olmak üzere dört tipi vardır. [13]

1.4.3.1. Erken Deselerasyon

FKH'da yavaşlama uterus kontraksiyonları ile eş zamanlı ortaya çıkar. Yavaşlamanın en derin noktası, kontraksiyonun zirve noktasıdır (Şekil 1.6). Erken deselerasyon, uterus kontraksiyonunun başlangıç, zirve ve bitiş şeklinin ayna görüntüsüdür. Doğumun ilerleyen evrelerinde fetal baş basısını gösterir. [14]

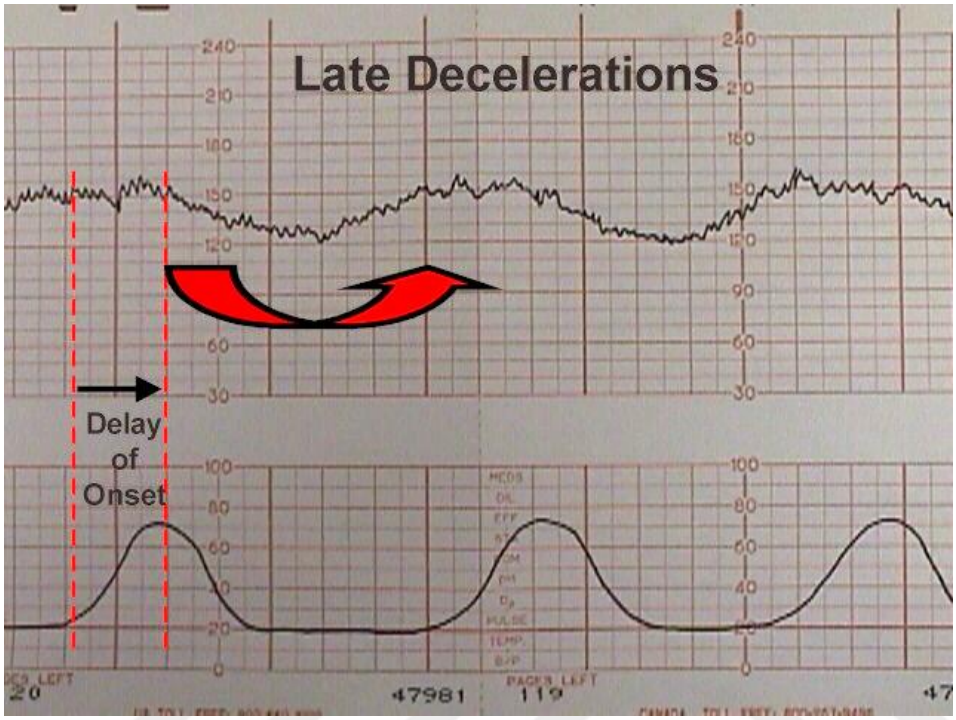


Şekil 1.6 : Erken Deselerasyon

1.4.3.2. Geç Deselerasyon

Anneden plasentaya giden kan akımı, tehlikeye girdiğinde FKH yavaşlar. Bu yavaşlama kontraksiyonun başlaması ile hemen ortaya çıkmaz. FKH'da yavaşlama, uterus kontraksiyonlarının tepe noktasından birkaç saniye sonra başlar, 20-90 saniye sonra en alt noktasına düşer. Kontraksiyonlar bitiminden bir süre sonra, uterusa olan kan akımının artması ile FKH normale döner (Şekil 1.7). Geç deselerasyon, plesental

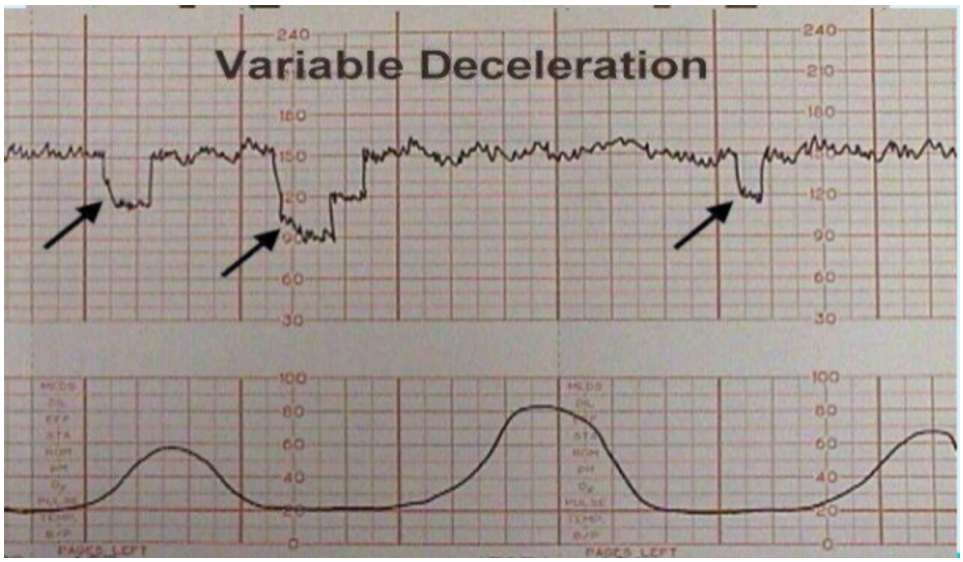
duvardaki sıkışmaya bağlı gelişen "hipoksinin" bir göstergesidir. Başlıca nedenleri: Maternal hipotansiyon (gebenin sırt üstü pozisyonda yatmasına bağlı), uterin hiperaktivite, anestezi, utero- plesental yetmezliktir. [15]



Şekil 1.7 : Geç Deselerasyon

1.4.3.3. Değişken Deselerasyon

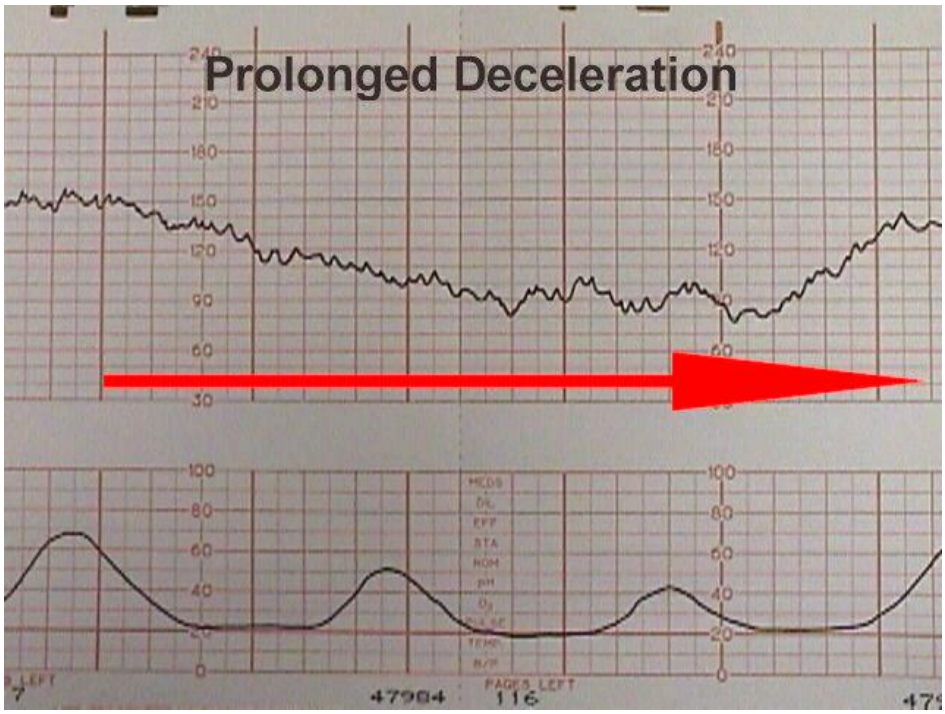
Bu deselerasyonda, FKH'daki düşmeler ile uterin kontraksiyonları arasında ilişki yoktur (Şekil 1.8). Değişken deselerasyonun temel nedeni, umblikal kord sıkışmasıdır. Kord sıkışması; kord prolapsusuna, kord dolanmasına, amniyon sıvısının azlığına ve fetal asidoza bağlı gelişebilir. Deselerasyondaki değişkenlik, kordun sıkışma derecesinin her kontraksiyonda farklı olmasındandır. [16]



Şekil 1.8 : Değişken Deselerasyon

1.4.3.4. Uzamış Deselerasyon

FKH'da bazal düzeyden 15 atımlık/dk düşüşün olduğu, bu düşüşün iki dakikadan daha uzun sürdüğü ve FKH'nın 10 dakika içinde bazal seviyeye döndüğü deselerasyon tipidir (Şekil 1.9). Uzamış deselerasyon, hipoksinin önemli bir işareti olup, utero-plesental yetmezliklerde sık görülür. [17]



Şekil 1.9 : Uzamış Deselerasyon

1.4.4. Variabilite (FKH Değişkenliği)

FKH'nın bazal düzeye göre, bir dakikalik iniş ve çıkışlarıdır (Örn: 120'den 130'a çıkması, 140'dan 128'e inmesi gibi). Sağlıklı bir sinir sisteminin yansımasıdır ve fetal sağlığın önemli bir göstergesidir. Variabilite; orta düzeyde, azalmış, çok düşük/yok ve artmış olarak dört kategoride değerlendirilir. (Şekil 1.10). [18]

1.4.4.1. Orta Düzeyde Variabilite

FKH'da, dakikada 6-25 atımlık/dk değişkenliklerini gösterir. Orta düzeyde variabilite "normal" kabul edilip, fetusun beyin sapının iyi oksijenlendiğini ve santral sinir sisteminin olgunlaştığını gösterir. [19]

1.4.4.2. Azalmış Variabilite

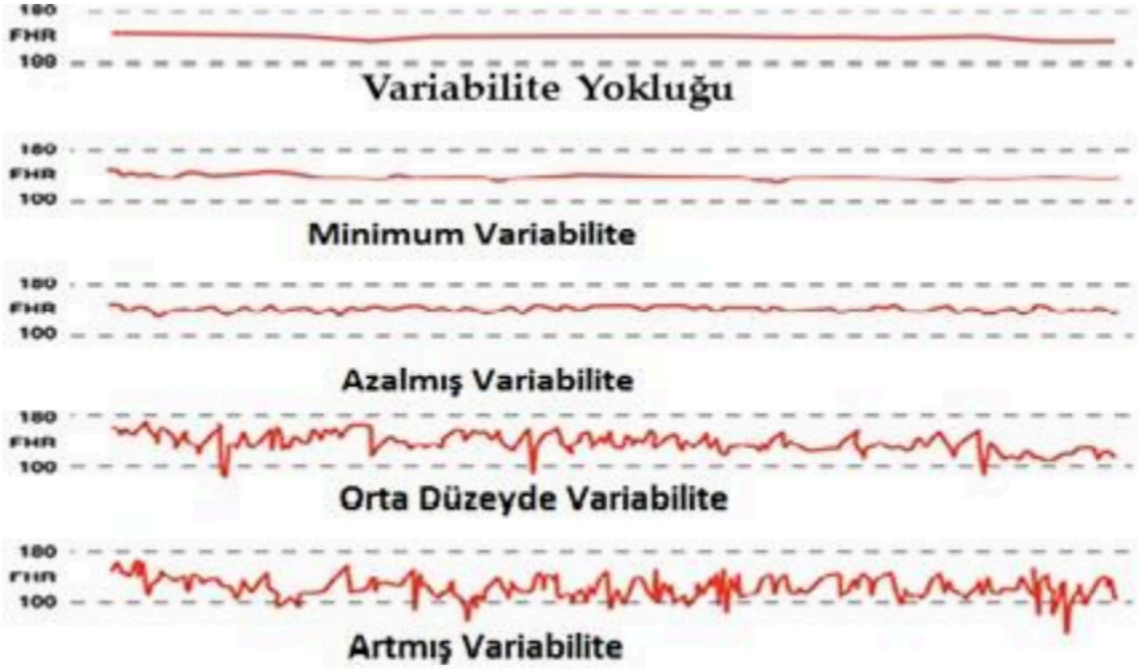
FKH'da, dakikada 3-5 atımlık değişkenliklerini gösterir. Azalmış variabilite, sempatik ve parasempatik sistemin denge içinde çalışmadığını gösterir. Narkotik, trankilizan, magnezyum sülfat gibi ilaç kullanımının yarattığı asidoz durumu ve konjenital kardiyak anomaliler bu variabiliteye neden olur. Fetusun uykuda olması, aneztesi, gebelik haftasının 32 haftadan az olması gibi durumlar azalmış variabilite için "fizyolojik" kabul edilir. [20]

1.4.4.3. Artmış Variabilite

FKH'da dakikada 25 atımdandan fazla değişikliklerin görülmesidir. Genellikle akut hipoksi ve umblikal kord sıkışmasında ortaya çıkmaktadır. [21]

1.4.4.4. Variabilitenin Azlığı/Yokluğu

FKH'da dakikada 0-2 atımlık değişikliklerdir. Bir başka anlatımla, FKH'nın düz bir şekilde devam etmesidir. Variabilitenin azlığı/yokluğu, önemli bir fetal tehlike işaretidir. Plasental kan akımı bozukluğunu gösterir. Fetal uyku, prematürite, ilaçlar (sedatif ve anestezi) ve hipoksik asidoz variabilite azlığının/yokluğunun temel nedenleridir. [22]

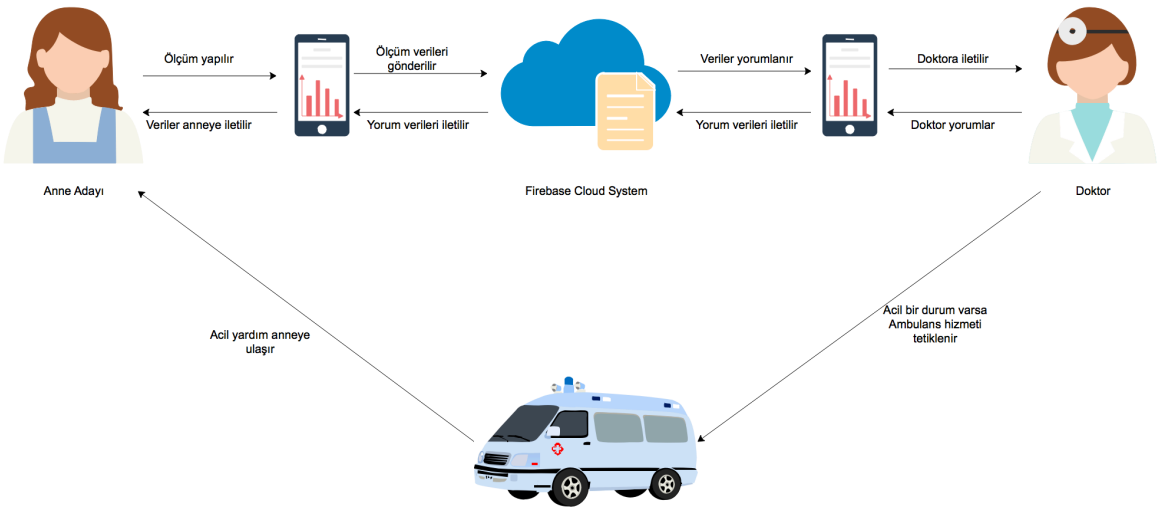


Şekil 1.10 : Variabilite çeşitleri

2. MOBİL UYGULAMANIN GELİŞTİRİLMESİ

2.1. Sistem Mimarisi

FKHMS'i anne adayının evde kendi kendine NST ölçümünü yapıp doktoru ile sonuçlarını anlık olarak paylaşmasına olanak sağlamaktadır. Şekil 2.1'de geliştirilen sistem mimarisi anlatılmaktadır.

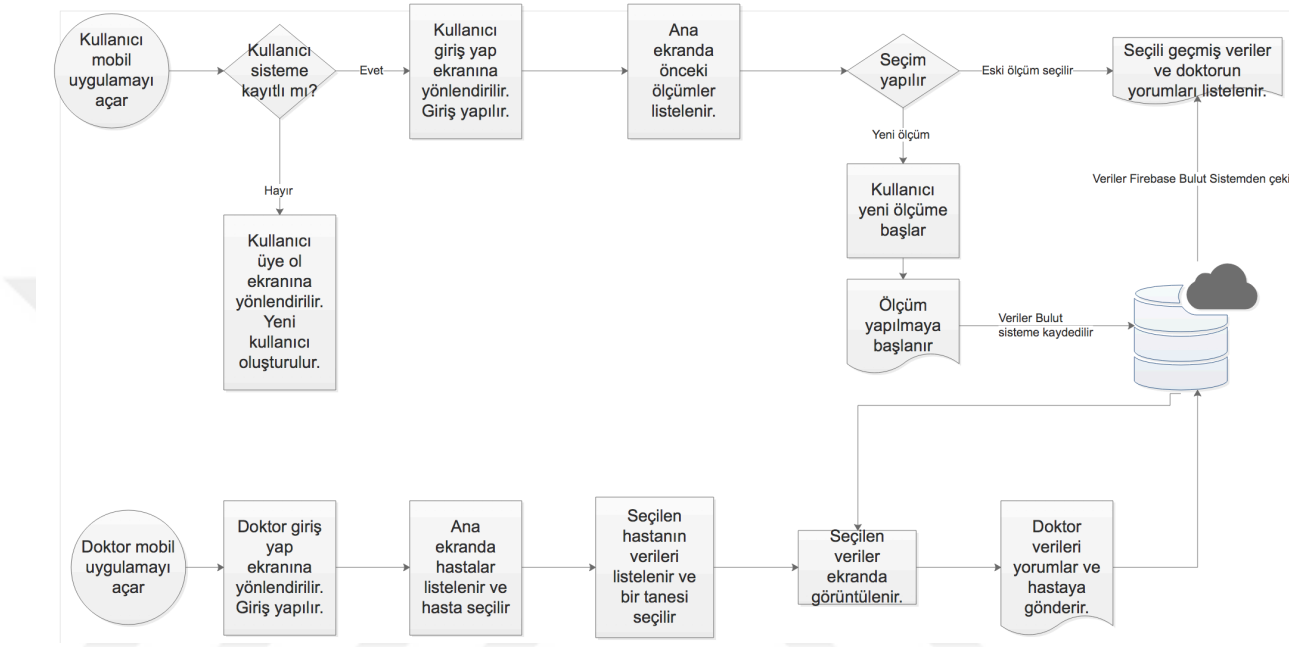


Şekil 2.1 : Sistem Mimarisi

Süreç, anne adayının FKHMS mobil uygulaması üzerinden üyelik oluşturması ile başlar. Daha sonra bluetooth aracılığı ile mobil doppler cihazına bağlanarak ölçüm aşaması ile devam eder. Cihazdan anlık olarak gelen değerleri ekranda görüp Uterus Kasılması değerlerinin doğruluğu için gereken kalibrasyon ayarını yaptıktan sonra kayıt işlemine geçer. Kayıt işlemini bitirdikten sonra yorumlanan değerler sisteme otomatik kaydedilir. Anne adayı yaptığı kayıtı doktoru ile paylaşmak isterse 'Gönder' butonuna basıp doktoru ile kayıt bilgilerini paylaşır.

Anne adayı tarafından doktora gönderilmiş olan bilgiler Firebase Bulut Sistemi'nde saklanır ve doktorun kullandığı mobil uygulama ya da hastane sistemine gönderilir. Doktor anne adayının göndermiş olduğu verileri grafiksel olarak ekranında görür. Hastanede trase üzerinde yorumlanan veriler sistem üzerinden de aynı şekilde yorumlanır ve sonucu anne adayına gönderilir. Eğer acil olabilecek bir durum söz

konusu ise ambulans hizmeti devreye sokularak anne adayının bulunduğu konuma ambulans gönderilir. Eğer herhangi bir kötü durum söz konusu değilse normal olan yorumlama anne adayına sistem üzerinden gönderilir. Yine aynı şekilde doktor tarafından yorumlanan veriler Firebase Bulut Sistemi'nde saklanır. Şekil 12'de sisteme ait akış diyagramı gösterilmektedir. [23]



Şekil 2.2 : Akış Diyagramı

2.2. Yapılan Uygulamanın Örnek Vaka Üzerinde Çalışması

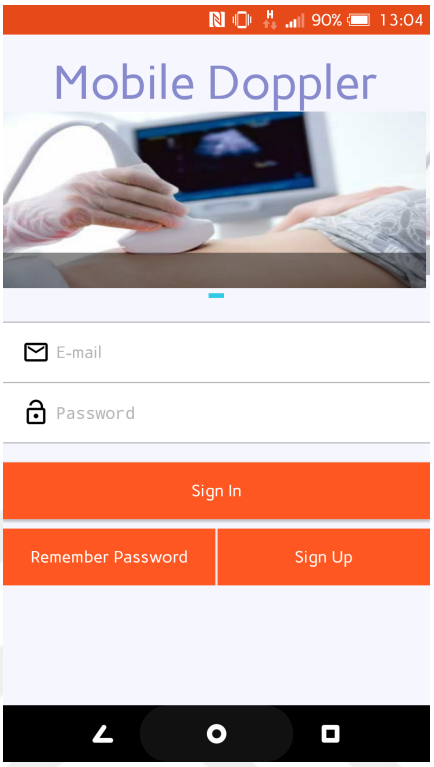
2.2.1. Giriş Ekranı

Anne adayı ve doktor aynı mobil uygulamayı kullanacaktır. Ancak bu iki kullanıcı tipi "user_type" parametresi ile birbirinden ayrılacağı için kullanıcı tipine göre anne adayı ve doktora gerekli olan yerlerde ayrı ekranlar gösterilecektir. Gerekmediği sürece aynı akışa sahip işlemlerde aynı ekranı kullanacaklardır.

Uygulama ilk açıldığında kullanıcının karşısına bir splash ekranı gelecektir. Bu ekranda arka plan işlemleri gerçekleştirilecek ve sonrasında ana ekranla karşılaşılacaktır.

Giriş ekranında kullanıcı giriş yapabilir, yeni kullanıcı oluşturmak için “Yeni kullanıcı oluştur” ekranına geçebilecektir.

Şekil 2.3 : Splash Ekranı



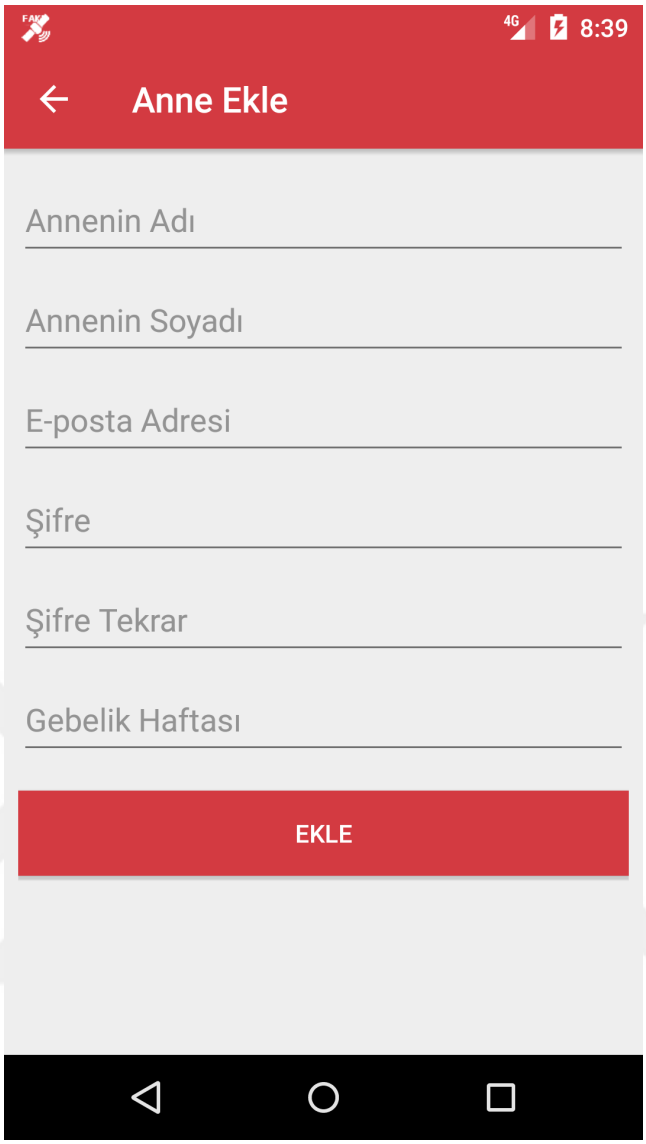
Şekil 2.4 : Giriş Ekranı

2.2.2. Yeni Kullanıcı Oluşturma (Anne Adayı)

Yeni kullanıcı ekranında anne adayından adı ve soyadı, e-posta adresi, şifre bilgilerini ve kayıt esnasındaki gebelik haftasını girmesi istenir. Bu bilgiler alındıktan sonra “Ekle” butonuna basılıp işleme devam edilir. Böylelikle sisteme yeni kullanıcı tanımlanmış olur.

Sistemin hastane sistemleri ile entegre olmasının ardından bu ekrana zorunlu olacak birçok bilgi eklenecektir.

Bu ekranda sadece anne kaydı yapılabilecektir. Doktor sistemde zaten kayıtlı olacağından doktorun kendisi sisteme kendini ekleyemeyecektir.

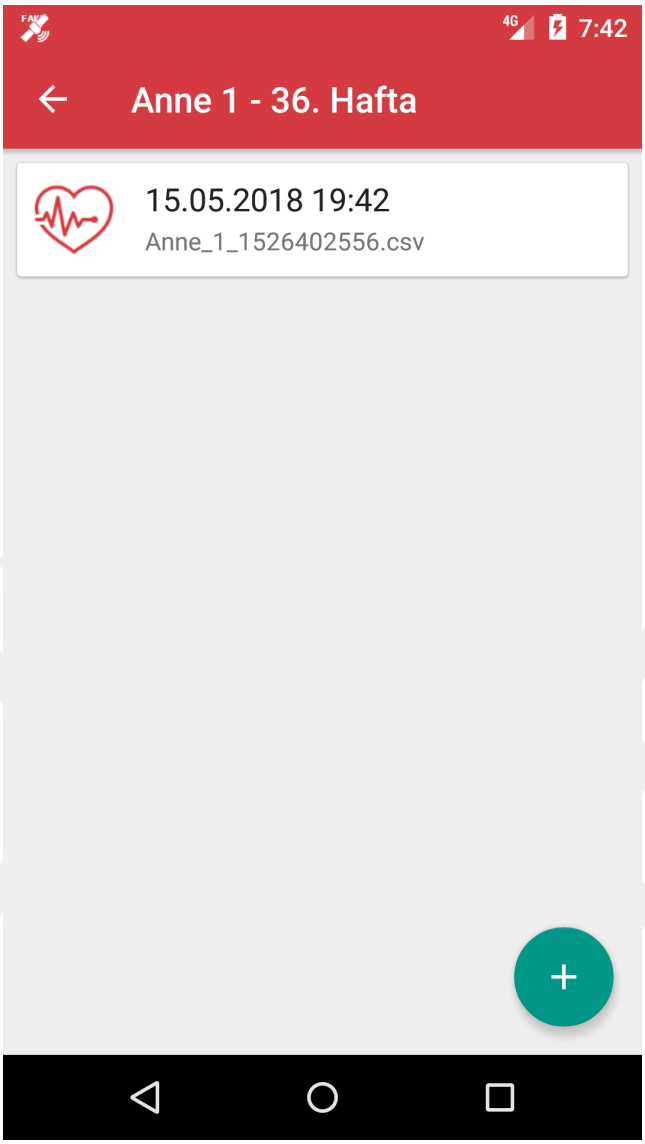


Şekil 2.5 : Anne Ekleme Ekranı

2.2.3. Var Olan Kayıtların Listelenmesi (Anne Adayı)

Anne adayı sisteme kayıt olduktan sonra karşılaşacağı ilk ekrandır. Anne adayı doğal olarak ilk bu ekrana geldiğinde herhangi bir veri ile karşılaşmayacaktır. Ancak daha önce herhangi bir kayıt işlemi gerçekleştirdiyse bu ekranda önceki kayıtlarını liste halinde görebilecek ve istediği kayıt içerisine gidip işlemine devam edebilecektir.

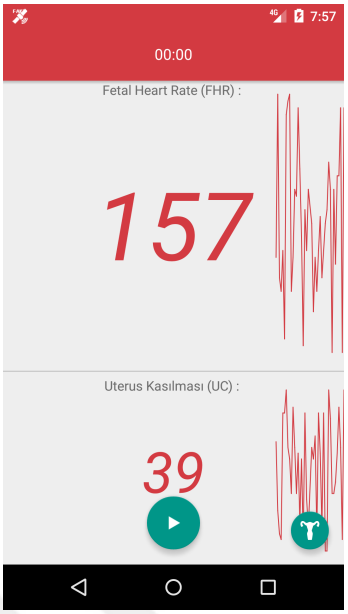
Var olan kayıtlar listelenirken kayıt tarihi, saati ve dosya uuid'si ile listelenecektir. İlerleyen zamanlarda bu ekrana filtre de eklenecektir. Filtre eklenmesi ile beraber anne adayı tarihe göre listeleme yapabilecektir.



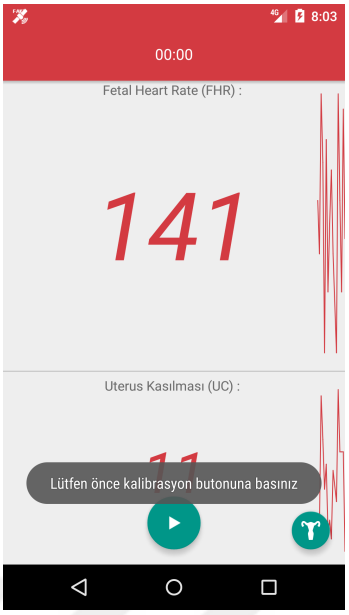
Şekil 2.6 : Var Olan Kayıtların Listelenmesi (Anne Adayı)

2.2.4. Yeni Kayıt Oluşturma (Anne Adayı)

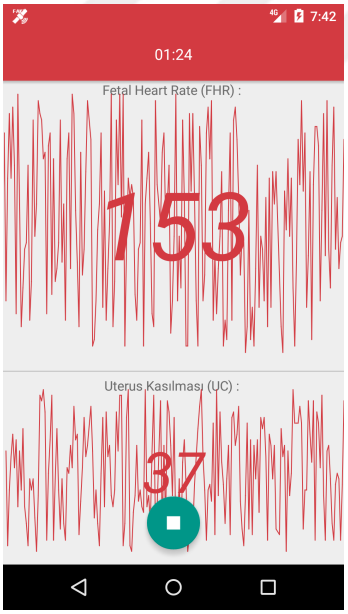
Anne adayı tüm geçmiş kayıtlarını gördüğü ekranda iken sağ alttaki "+" butonuna basarak yeni kayıt oluşturma ekranına geçiş yapacaktır.



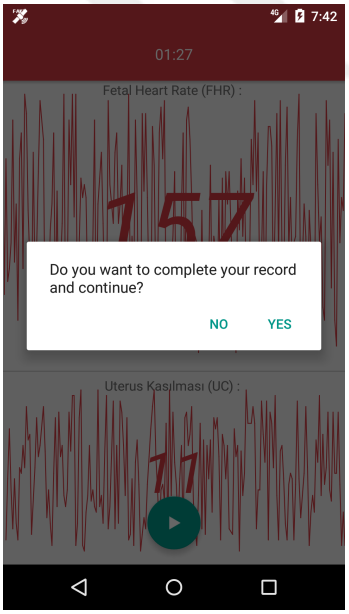
Şekil 2.7 : Yeni Kayıt Oluşturma (Anne Adayı)



Şekil 2.8 : Yeni Kayıt Oluşturma (Anne Adayı)



Şekil 2.9 : Yeni Kayıt Oluşturma (Anne Adayı)



Şekil 2.10 : Yeni Kayıt Oluşturma (Anne Adayı)

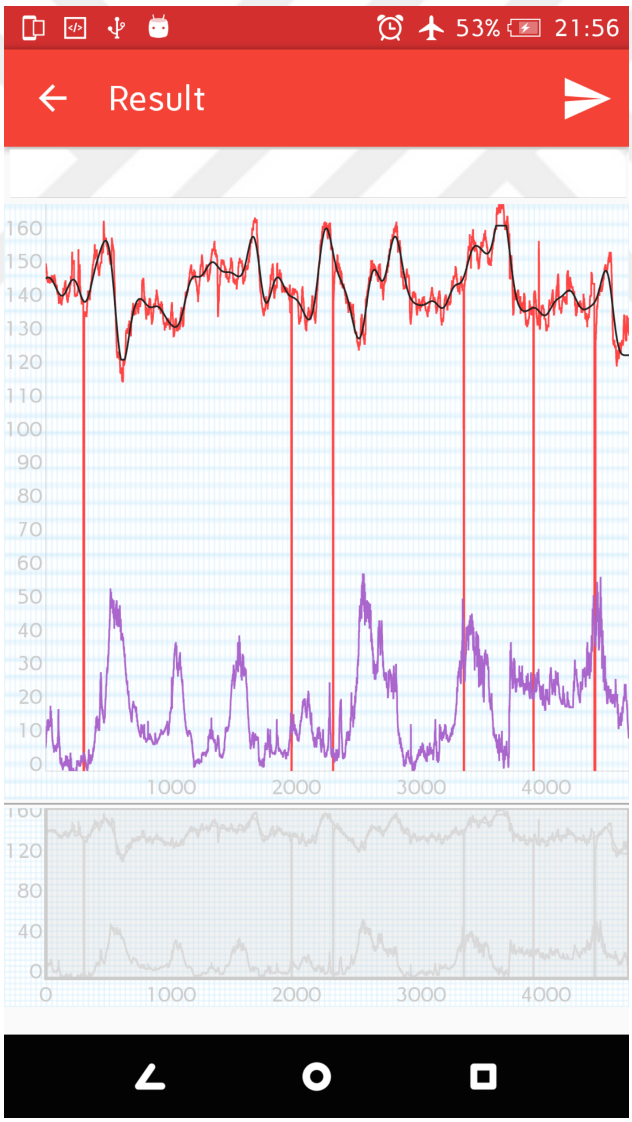
Bu ekranda anne adayı kayıt yapmadan da fetüse ait değerleri ekranda real time olarak görebilecektir. Kayıt aşamasına geçmeden önce uterus kasılması kalibrasyon ayarı için

sağ alttaki “Uterus” butonuna basarak kalibrasyon ayarını yapması gerekecektir. Daha sonrasında orta alttaki “Kayıt” butonuna basarak kayıt işlemini başlatabilecektir.

Kayıt işlemi esnasında yukarıda süreyi görebilecek ve istediği zaman alt ortadaki “Kayıt Durdur” butonuna basarak kayıt işlemini sonlandırabilecektir. Kayıt işlemini sonlandırdıktan sonra verilerinin işlenmiş halini göreceği önizleme ekranına yönlendirilecektir.

2.2.5. Önizleme Ekranı (Anne Adayı)

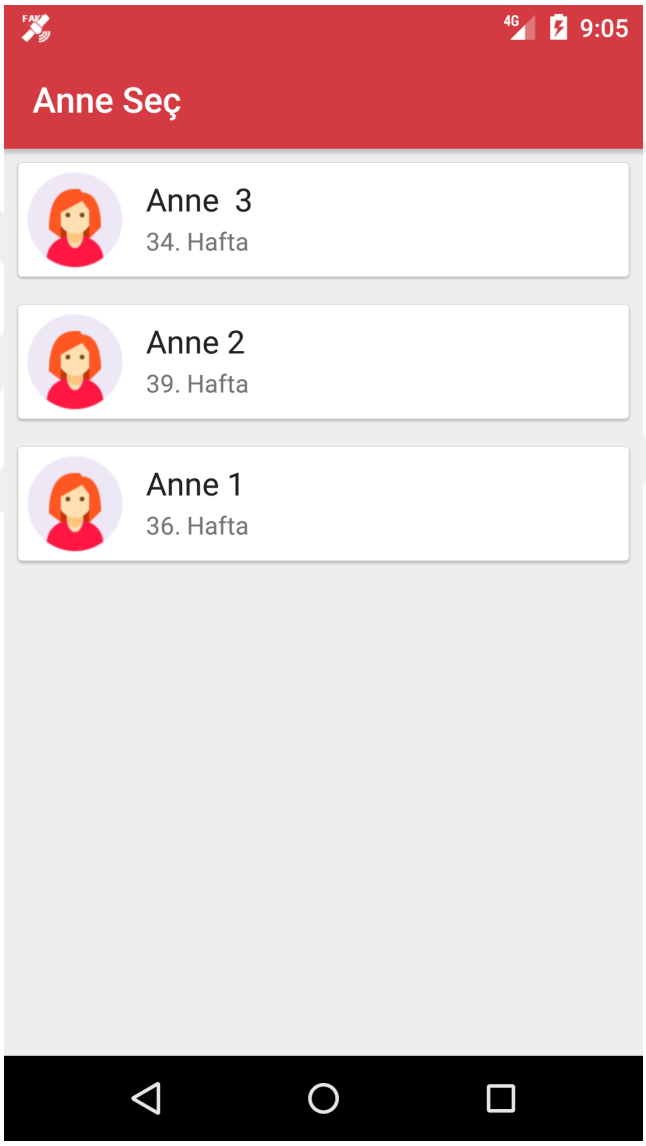
Anne adayı kayıt işlemini tamamladıktan sonra bu ekranda fetüse ait bilgilerin trasesini görecektir. Grafiksel olarak verilerin işlenmiş hali bu ekranda verilecektir. Anne adayı sağ üstteki “Gönder” butonuna basarak verilerini doktoruna gönderecektir.



Şekil 2.11 : Önizleme Ekranı

2.2.6. Hasta Listesi (Doktor)

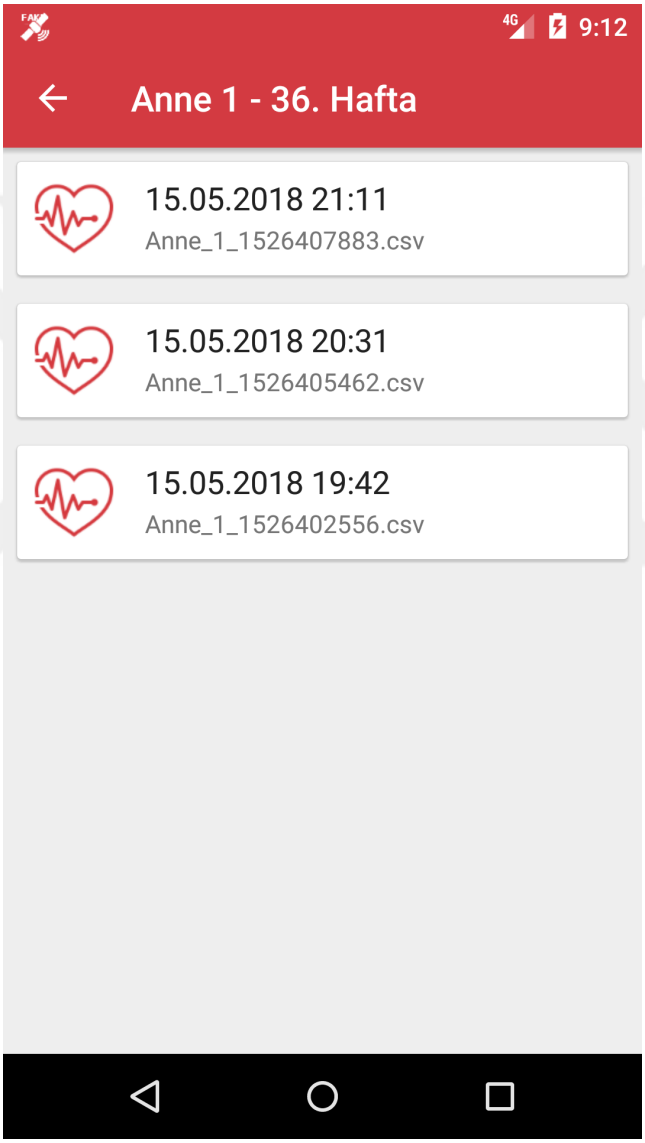
Hasta listesi ekranı doktorun anne adaylarının listesini görebildiği ekrandır. Bu ekranda doktorun takibini sürdürdüğü anne adaylarının bilgileri listelenecektir. Doktor istediği anne adayını seçip işlemine devam edebilecektir. Bu ekrana ilerleyen safhalarda filtreleme özelliği eklenecektir. Filtre özelliği ile anne adayının diğer bilgilerine göre sıralama yapılacaktır.



Şekil 2.12 : Hasta Listesi

2.2.7. Hasta Bilgileri (Doktor)

Hasta bilgileri ekranı, doktorun seçmiş olduğu anne adayına ait kayıtların listelendiği ekrandır. Doktor bu ekranı kullanarak seçmiş olduğu anne adayına ait herhangi bir kayıdın detay bilgilerine gidebilecektir. Bu ekrana ilerleyen safhalarda filtreleme özelliği eklenecektir. Filtre özelliği ile anne adayının kayıtları tarihe ve diğer bilgilerine göre sıralama yapılacaktır.

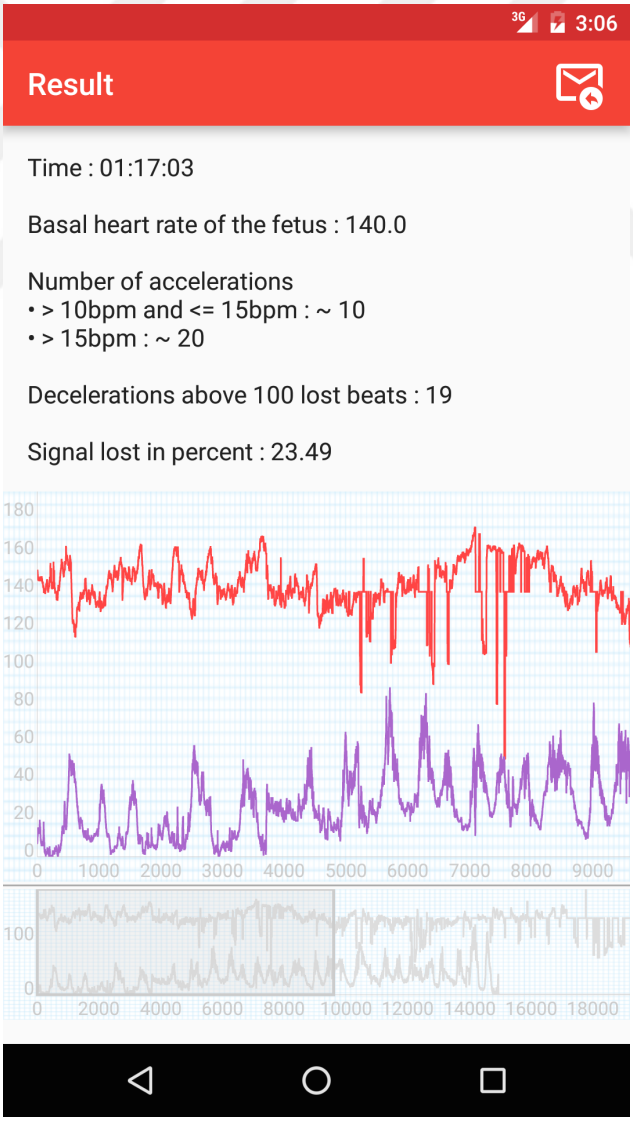


Şekil 2.13 : Hasta Bilgileri

2.2.8. Kayıt Detayları (Doktor)

Kayıt detay ekranı doktorun kayıt listesinden seçtiği kayıta ait detay bilgilerin verildiği ekrandır. Doktor bu ekranda hastanın verilerinin trasesini ve grafiksel gösterimi görmekle beraber Dawes-Redman algoritması ile yorumlanan değerleri de kendisine yardımcı olması amacıyla görecektir. Doktora gösterilen bilgiler kesinlikle anne adayına gösterilmeyecektir.

Doktor kendi yorumuna ve ekranda gösterilen yorumlara göre kararını verip ekranın sağ üstündeki "Mesaj Gönder" butonuna basıp anne adayına mesaj gönderme popup'ına yönlendirilecektir. Mesaj gönderme popup'ına gerekli olan bilgileri anne adayına yazıp gönderebilecektir.



Şekil 2.14 : Kayıt Detay

3. MEDİKAL VERİLERİN TOPLANMASI

Bu bölümde FKHMS mobil uygulaması sayesinde anne adayının evde kendi medikal verilerinin nasıl toplandığı anlatılacaktır.

Anne adayı mobil uygulamayı açıp yeni kayıt oluşturmaya başladığı andan itibaren her saniyede 4 olmak üzere ortalama 10 dakikalık kayıt sırasında 2400 kalp atımı, 2400 de uterus kasılması verisi alınacaktır. Bu alınan ortalama 4800 veri her bir hasta kaydı için bir adet .csv dosyasında saklanacaktır. Verilerin saklandığı .csv dosyaları Firebase Storage'de depolanacaktır. [24]

Medikal veriler daha sonrasında Firebase Storage'den export edilerek herhangi bir platformda da çalıştırılabilmektedir. Toplanan verilerin doğruluğunun anlaşılması için CTGViewerLite programı kullanılarak karşılaştırması yapılmış ve verilerin doğru işlenerek traseye aktarıldığı kanıtlanmıştır. Bazı değerlerin alınamadığı anlık durumlarda oluşan kayıp veriler (missing values) ise Weka programı kullanılarak EMImputaion (Expectation Maximization) ve DBScan algoritmaları ile optimize edilerek bu algoritmalar mobil uygulamaya entegre edilmiştir. Entegrasyon sonrasında tekrar CTGViewerLite programı ile karşılaştırılması yapılmıştır. [25, 26]

3.1. Verilerin Analizi

Bu bölümde verilerin analiz edilmesini anlatan akış diyagramlarına yer verilecektir. Anne adayının kaydı sonrasında .csv dosyasında tutulan verilerden yola çıkarak analizi yapılmaktadır. Bu veriler doktorun traseyi yorumlamasına yardımcı olmaktadır.

3.2. Dawes / Redman Kriterleri

1977 yılında Profesör Dawes ile Profesör Redman fetal kalp hızı trasesi ve sonuçları arasındaki bağlantıyı araştırmaya başladılar. Bu araştırmalar 26. ve 42. Gebelik haftası arasındaki doğum öncesi verileri analiz eden bir bilgisayarlı sistemin gelişmesine yol açtı. Geliştirilen bu sistem bir kaydı değerlendirmek için Dawes / Redman kriterleri olarak adlandırılan bir dizi kriteri kullanmakla beraber 73 500'den fazla trase ile oluşturulan bir veri tabanına dayanmaktadır. [27]

Dawes / Redman kriterleri, fetal kalp hızının otomatik analizinin temelidir. Analizler 10 dakikalık ölçüm sonrasında yapılır ve tüm kriterler yerine getirilirse, sistem fetüsün sağlıklı olduğunu gösterir. Ancak, tüm kriterler yerine getirilmediyse, kayıt devam eder ve tüm kriterler karşılanana veya kayıt 60 dakika boyunca (hangisi ile önce karşılaşılırsa) devam edene kadar her 2 dakikada bir değerlendirilir. Sistem akselerasyonları, deselerasyonları, bazal kalp hızını, variabiliteleri hesaplar ve her yeni hesaplamada baseline yeniden hesaplanır.

Dawes / Redman kriterleri sonucu tahmin etmede yardımcı olacaktır. Ancak, kriterler yerine getirilse bile, fetüsün sağlıklı doğup doğmayacağını garanti etmez. Bununla birlikte, Dawes / Redman analizi ile doğum öncesi riskler azaltılabilir ve başarılı bir doğum olasılığını artırabilir.

Ek 1'de Dawes / Redman kriterlerinin diyagramı verilmiştir.

Dawes / Redman kriterleri birçok medikal cihazda kullanılmaktadır. Bunlardan bir tanesi ve Avrupa ülkelerinde sıklıkla kullanılan Sonicaid Fetalcare yazılımıdır. Bu yazılım Huntleigh Healthcare tarafından Dawes / Redman kriterleri kullanılarak geliştirilmiş ve hastanelerde NST ölçümlerine yardımcı olan bir yazılımdır. [28]

EK 2'de ve Ek 3'te Dawes / Redman kriterleri baz alınarak yazılmış C kodlarına yer verilmiştir.

3.3. Saklanan Örnek Hasta Verisi

Anne adayının mobil uygulamayı kullanarak elde ettiği veriler timestamp, fhr (fetal heart rate) ve uc (uterus contraction) kolonlarında saklanmaktadır.

7146	129.75	12.0
7147	129.25	12.0
7148	129.25	12.0
7149	126.0	12.0
7150	126.0	12.0
7151	127.0	12.0
7152	127.0	12.0
7153	126.5	12.0
7154	126.5	11.0
7155	125.5	11.0
7156	125.5	11.0
7157	125.5	11.0
7158	125.5	11.0
7159	123.0	11.0
7160	123.0	12.0
7161	124.0	12.0
7162	124.0	12.0

Şekil 3.1 : Örnek Hasta Verisi

3.4. Doktor Tarafından Yorumlanması

ACOG 2009 yılında, 116 FKH trasesi için "Üç Aşamalı (Kategorili) Sınıflama Sistemi" ile fetusun durumunun standardize bir protokolle izlenmesini amaçlamıştır. Bu protokolde; Kategori 1, Kategori 2 ve Kategori 3 olmak üzere üç sınıflama türü mevcuttur. Bu kategorik sınıflamanın özellikleri aşağıda Tablo 1'de gösterilmiştir. [29]

Tablo 1.1 : FKH traselerinin üç kategorili sınıflama sistemi

Kategori 1 (Normal): Bu kategoride FKH traseleri normal kabul edilir ve özellikli bir önlem gerektirmez. Fetusun asit-baz dengesinin normal olduğunu gösterir. **Özellikler:** • **Bazal FKH: 110-160/dk** • **Variabilite (değişkenlik):Orta düzeyde (6-25 atım/dk)** • **Akselerasyon ve erken deselerasyon olabilir / olmayabilir** • **Geç ve değişken deselerasyon yok**
Kategori 2 (Arada/Şüpheli): FKH traseleri arada/şüpheli kabul edilir. Bu kategori hızlı bir değerlendirme, yakın izlem ve fetal iyilik halini gösteren diğer testlere ihtiyaç gerektirmektedir. **Özellikler:** • **Bradikardi (Variabilite yokluğunun eşlik etmediği)** • **Variabilite değiÇikliği o Bazal variabilite yokluğu (tekrarlayan deselerasyonların eşlik etmediği) o Minimum bazal variabilite o Artmış bazal variabilite** • **Fetusa vibroakustik uyarı verilmesi, fetusun başından kan örneklemi alınması ve fetal baş uyarısı gibi uyaranlardan sonra akselerasyonların olmaması** • **Deselerasyonlar** **o Sürekli veya belli aralıklarla deselerasyon** **o Minimal veya orta düzeyde bazal variabilitenin eşlik ettiği tekrarlayan değişken deselerasyonlar** **o Uzamış deselerasyonlar (≥2 dakika ila <10 dakika arası** **o Orta düzeyde bazal variabilitenin eşlik ettiği, geç deselerasyonlar** **o Overshoot (sadece değişken deselerasyon sonrası görülen FKH'daki artış), shoulders (deselerasyon öncesi ve sonrası omuz çıkıntısı gibi FKH'da**

artış) gibi durumlarda bazal seviyeye, yavaş dönen değişken deselerasyonlar

Kategori 3 (Anormal): FKH traseleri anormal kabul edilir, fetusun asit-baz durumunun anormal olduğunu gösterir. Hemen değerlendirme ve fetal resüstasyon gerektirir.

Özellikler:

- **Variabilite yokluğu**
- **Tekrarlayan geç deselerasyon**
- **Tekrarlayan değişken deselerasyon**
- **Bradikardi**
- **Sinüzoidal düğüm (FKH'da dakikada 5-15 atımlık yükselmelerin ve trasede sürekli düzgün dalgaların olması)**

3.5. Kullanılan Araçlar

3.5.1. Android Programlama Dili ve Android Studio

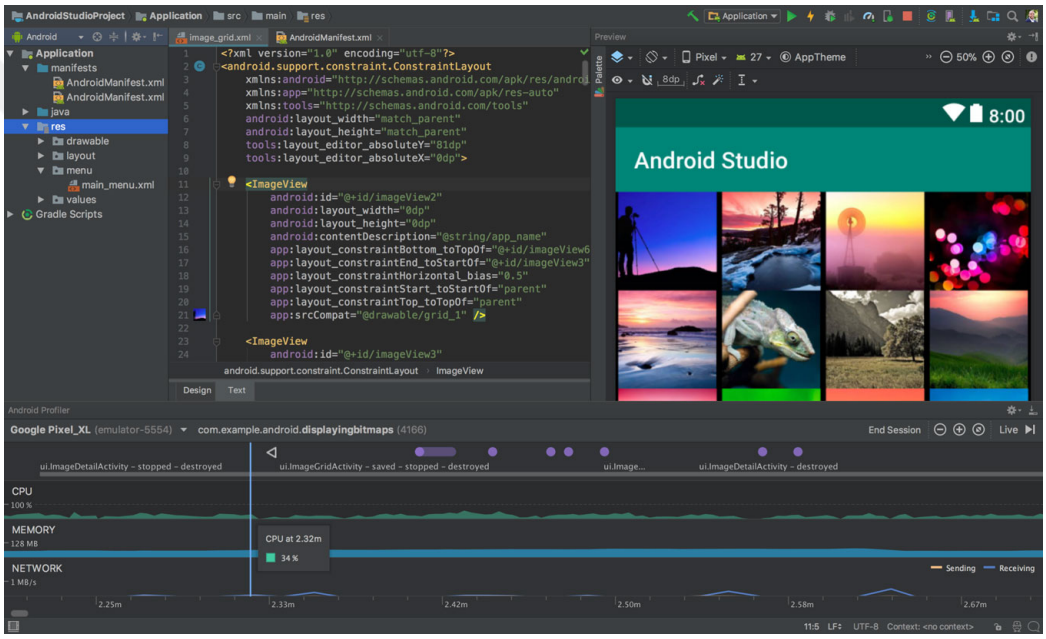
Android; Google ve Open Handset Alliance tarafından, mobil cihazlar için geliştirilmekte olan, Linux tabanlı özgür ve ücretsiz bir işletim sistemidir. Sistem açık kaynak kodlu olsa da, kodlarının ufak ama çok önemli bir kısmı Google tarafından kapalı tutulmaktadır. Google tarafından ücretsiz olmasının sebebi, sistemin daha hızlı ve çabuk gelişmesi, birçok popüler marka tarafından kullanılması ve bu sayede reklamlarını daha fazla kişiye ulaşmasını sağlamaktır. Google, Android sistemi üzerinde çalışan Google Play marketteki oyun ve uygulamalar üzerinde aldığı reklamları yayınlayarak para kazanmaktadır. Android'in desteklenen uygulama uzantısı ".apk"dır.

Android, aygıtların fonksiyonelliğini genişleten uygulamalar yazan geniş bir geliştirici grubuna sahiptir. Android için halihazırda 1 milyondan fazla uygulama bulunmaktadır. Google Play Store ise, Android işletim sistemi uygulamalarının çeşitli

sitelerden indirilebilmesinin yanı sıra, Google tarafından işletilen kurumsal uygulama mağazasıdır. Geliştiriciler, ilk olarak aygıtı, Google'ın Java kütüphanesi aracılığıyla kontrol ederek Java dilinde yazmışlardır.

Android Studio, Android için resmi tümleşik geliştirme ortamıdır. 16 Mayıs 2013 tarihinde Google I/O etkinliğinde tanıtılmıştır. Android Studio, IntelliJ IDEA'ya dayalı olup Android geliştirme için özel olarak tasarlanmıştır.

Android Studio, Apache lisansı ile lisanslanmıştır ve ücretsiz olarak edinilebilmektedir. [30]



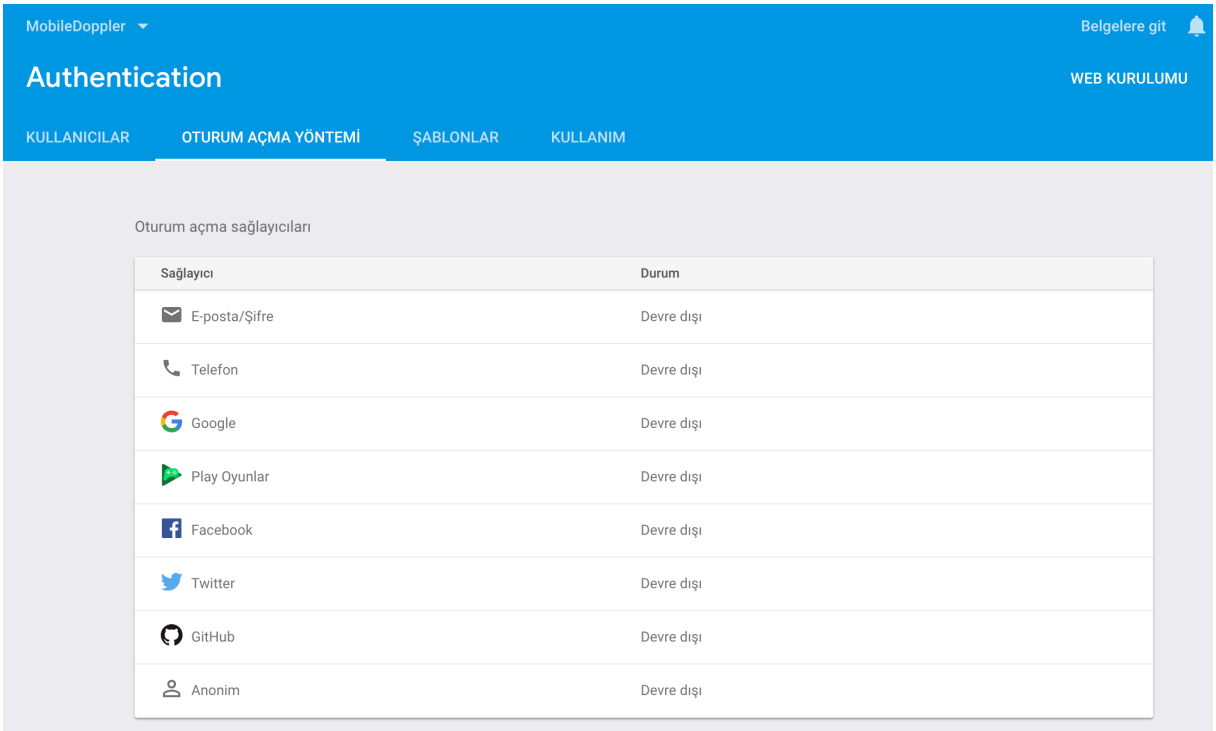
Şekil 3.2 : Android Studio

3.5.2. Firebase Bulut Sistemi

3.5.2.1. Firebase Authentication

Kullanıcılar, Firebase Authentication sayesinde zaten kullandıkları ve güvendikleri bir sistemi kullanarak uygulamanızda oturum açarlar. Uygulamanız daha sonra kullanıcının verilerini güvenli bir şekilde buluta kaydeder ve tüm cihazlarında aynı kişiselleştirilmiş deneyimi sunar.

Firebase Authentication, uygulamanızın kullanıcılarının kimliğini doğrulamak için arka uç hizmetleri, kullanımı kolay SDK'lar ve hazır kullanıcı arayüzü kitaplıkları sağlayarak farklı platformlarda ve uygulamalar arasında daha ilgi çekici bir deneyim sunar. Şifre kullanarak ve Google, Facebook, Twitter gibi popüler birleşmiş kimlik sağlayıcıları kullanarak kimlik doğrulamayı da destekler. Böylece, kullanıcılar içeriğinize kolayca erişebilir ve uygulamalarınıza hızlı ve güvenli bir şekilde girebilirler. [31]



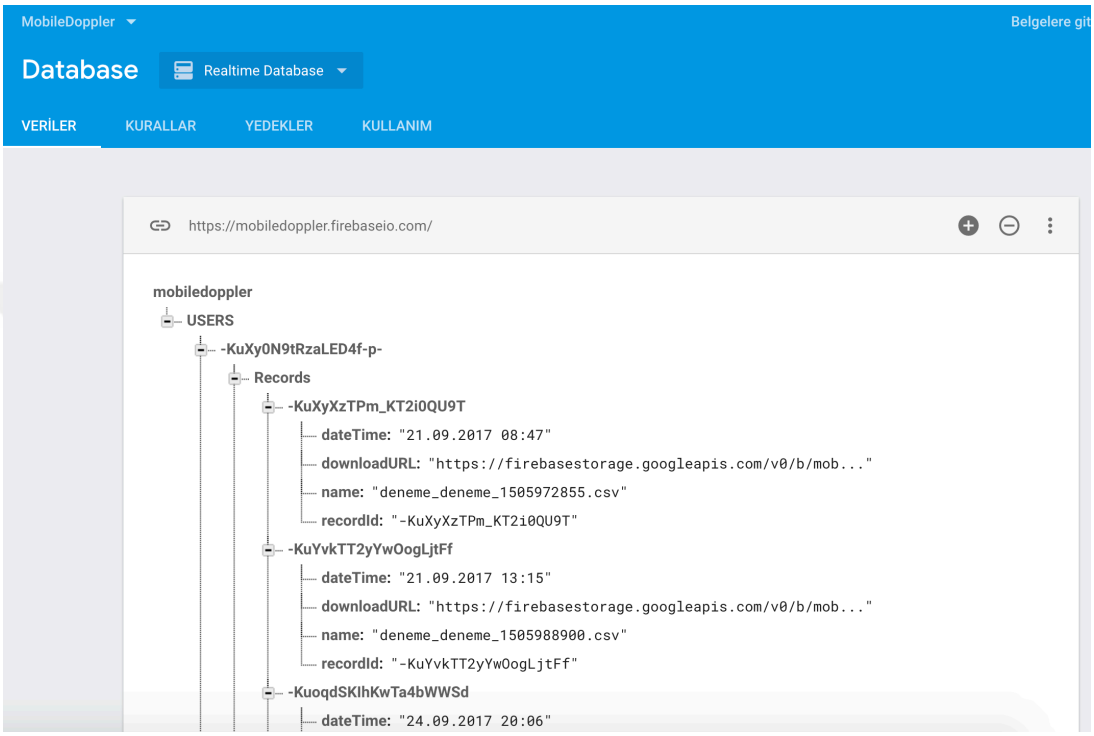
Şekil 3.3 : Firebase Authentication

3.5.2.2. Firebase RealTime Database

Firebase RealTime Database, Google firmasının gerçek zamanlı, NoSQL veritabanını kullanarak kullanıcı ve mobil uygulamarı daha eş zamanlı hale getiren bir hizmetidir.

Veriler tüm istemcilerde gerçek zamanlı olarak senkronize edilir ve uygulamanız çevrimdışı olduğunda bile kullanılabilir durumda kalır.

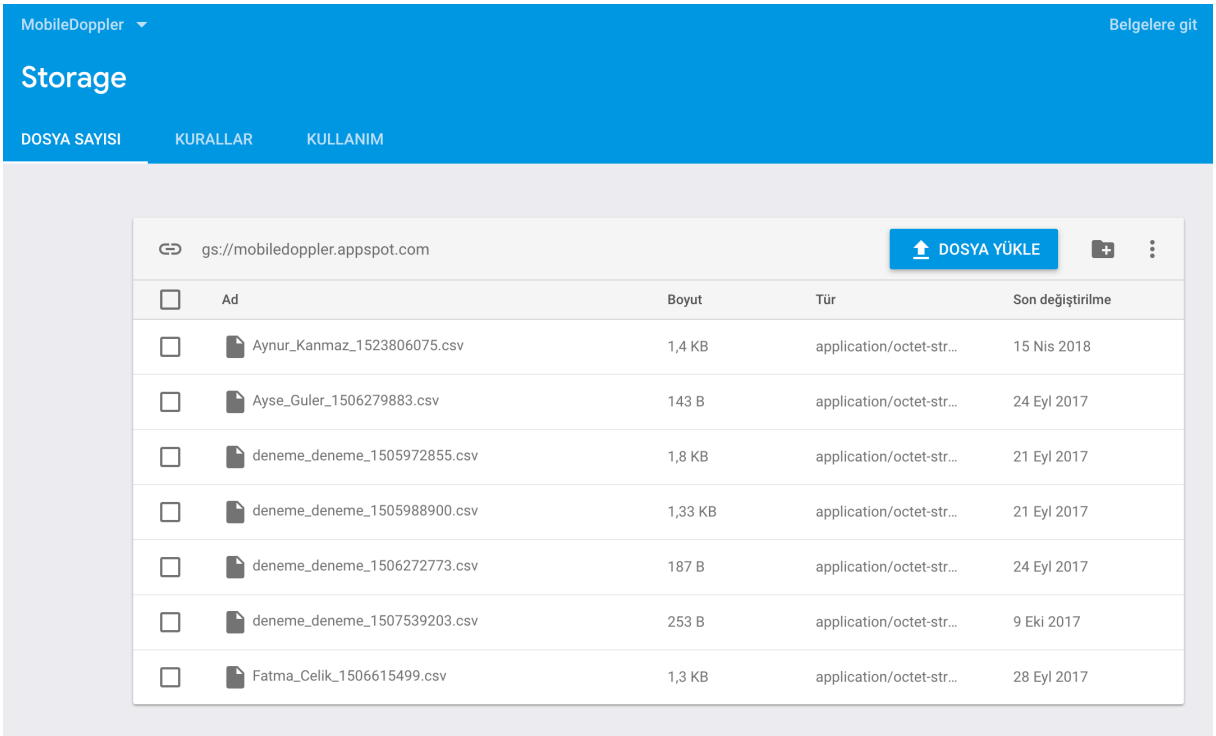
Firebase Gerçek Zamanlı Veritabanı, bulut tarafından barındırılan bir veritabanıdır. Veriler, JSON olarak depolanır ve her bağlı kullanıcıya gerçek zamanlı olarak senkronize edilir. IOS, Android ve JavaScript SDK'larla çapraz platform uygulamaları oluşturduğunuzda, tüm kullanıcılar bir Realtime Veritabanı örneğini paylaşır ve en yeni verilerle güncellemeleri otomatik olarak alır. [32]



Şekil 3.4 : Firebase RealTime Database

3.5.2.3. Firebase Storage

Firebase Cloud Storage, uygulamalarınıza zengin medya içeriği oluşturmanıza olanak tanıyan, resim ve video gibi kullanıcı tarafından oluşturulan içeriği yükleyip paylaşmanıza olanak tanır. Verileriniz, yüksek kullanılabilirlik ve global fazlalık ile bir exabyte ölçekli nesne depolama çözümü olan bir Google Cloud Storage grubunda depolanır. Cloud Storage, bu dosyaları doğrudan mobil cihazlardan ve web tarayıcılarından güvenli bir şekilde yüklemenizi ve kolaylıkla yönetmenizi sağlar. [33]

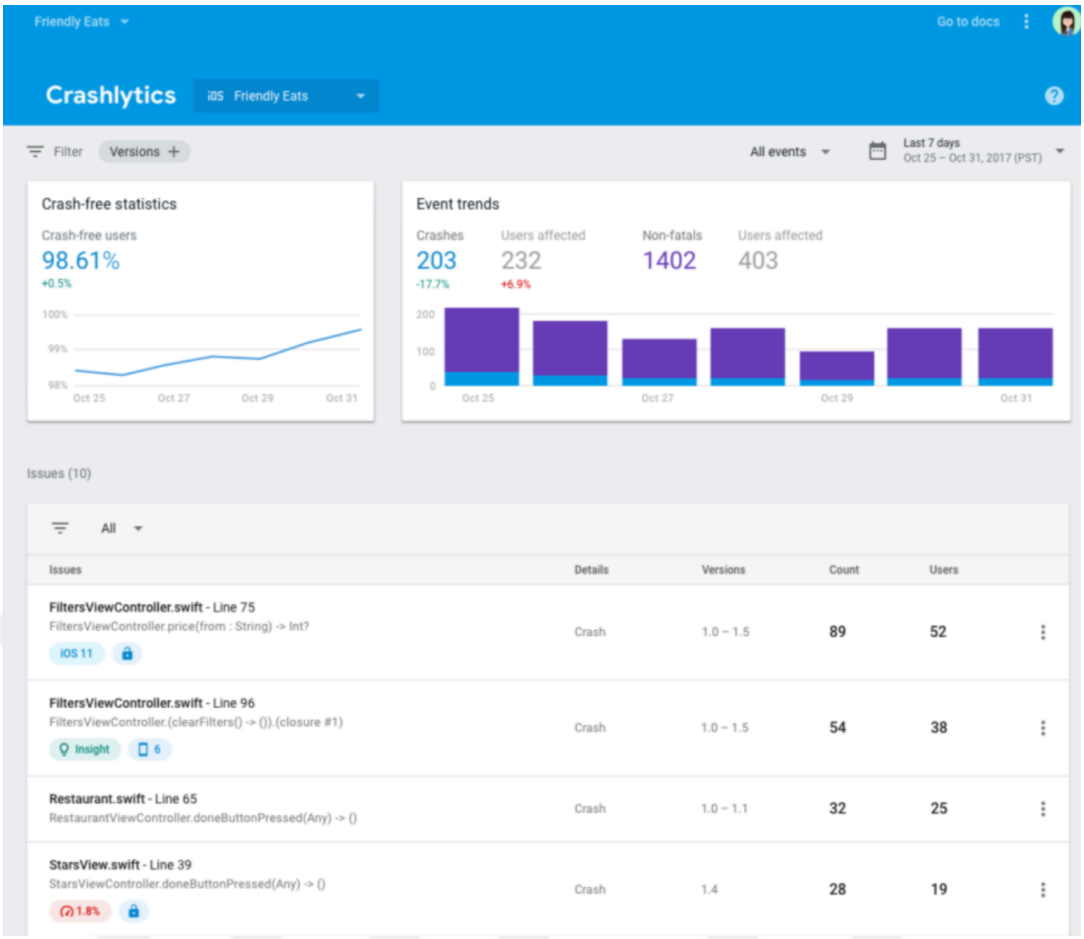


Şekil 3.5 : Firebase Storage

3.5.2.4. Firebase Crashlytics

Firebase Crashlytics, uygulamanızın kalitesini bozan stabilite sorunlarını izlemenize, öncelik vermenize ve düzeltmenize yardımcı olan hafif, gerçek zamanlı bir hata izleme önleme platformudur. Crashlytics, çökmeleri akıllıca gruplayarak ve onlara yol açan durumları vurgulayarak size sorun giderme zamanından tasarruf etmenizi sağlar.

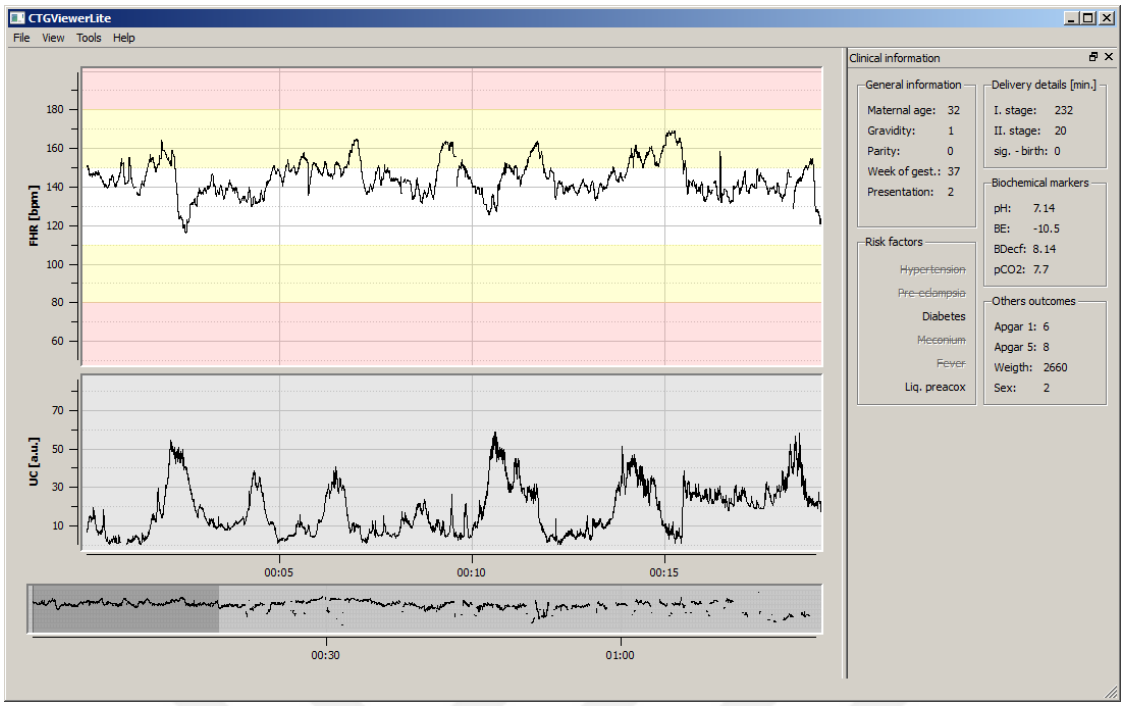
Belirli bir kilitlenmenin çok sayıda kullanıcıyı etkileyip etkilemediğini öğrenebilirsiniz. Bir sorun aniden önem derecesinde arttığında uyarı alıp hangi kod satırlarının kilitlenmesine yol açtığını belirleyebilirsiniz. [34]



Şekil 3.6 : Firebase Crashlytics

3.5.3. CTGViewerLite

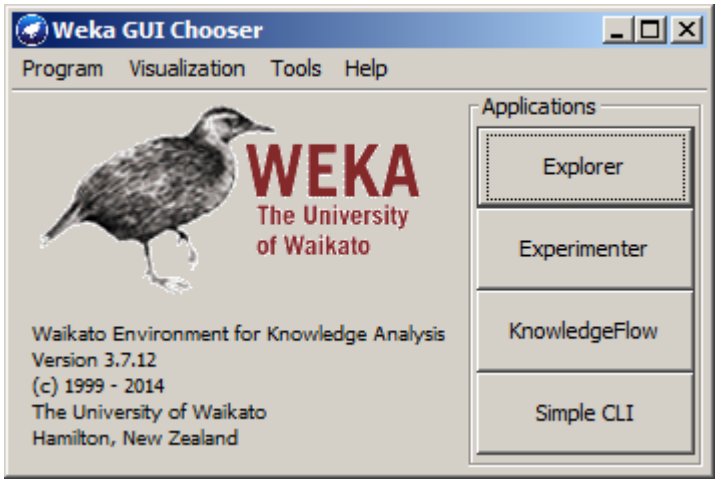
CTGViewerLite programı Çekya Teknik Üniversitesi tarafından geliştirilen bir yazılımdır. Bu yazılım sayesinde hastalara ait elimizde varolan .dat uzantılı datasetleri açıp görüntülememizi sağlamaktadır. Ayrıca bu program aracılığıyla elimizdeki datasetleri danışman doktorumuza yorumlatabiliyoruz. [35]



Şekil 3.7 : CTGViewerLite

3.5.4. Weka

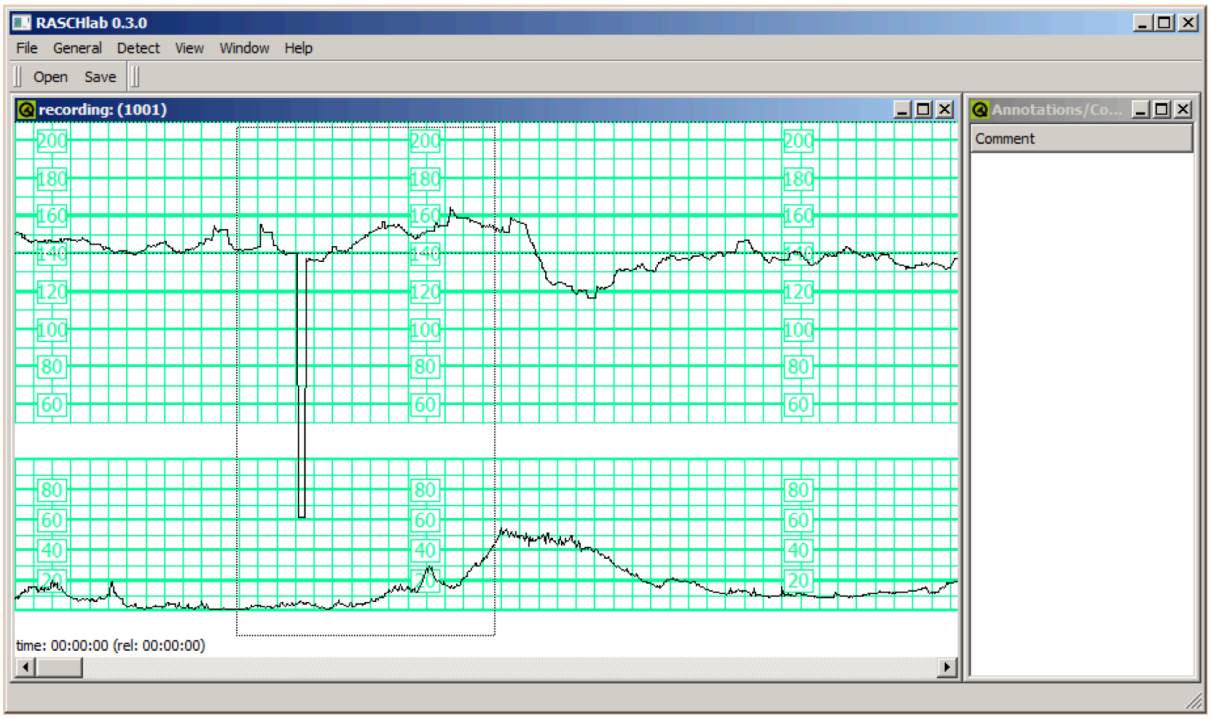
Weka programı big data ve data mining ile uğraşanlar için vazgeçilmez yazılımlardan bir tanesidir. Weka ile elimizdeki datasetlerde bulunan kayıp verileri (missing value) kayıp olmaktan çıkartıp EMImputaion (Expectation Maximization) ve DBScan algoritmalarını kullanarak bulabiliyoruz. [36]



Şekil 3.8 : Weka

3.5.5. libRASCH

libRASCH programı Münih Teknik Üniversitesi tarafından geliştirilen bir yazılımdır. Bu program sayesinde elimizde varolan datasetleri projemizde kullanacağımız Dawes-Redman algoritmasını kullanarak hesaplatabiliyoruz. Bu program aynı zamanda Weka'yı kullanmaz isek kendi içerisinde MissingValueReplace Algoritmasını kullanarak Dawes-Redman algoritmasını çalıştırabiliyor. Ancak bu algoritma EMImputation ve DBScan kadar etkili sonuçlar vermemektedir. [37]



Şekil 3.9 : libRASCH

4. SONUÇLAR

Bu tez çalışması kapsamında mobil Fetal Kalp Hızı Monitorizasyon sistemi geliştirilmiştir. Böylece anne adayı tehlikeli gebeliği sürecinde sağlık kurumuna gitmeden kendi ölçümünü evde yapabilecek ve verileri doktoru ile anlık olarak paylaşabilecektir. Tehlikeli gebelik sonucunda oluşan ölü doğumların bir nebze de olsa önüne geçilebilecek ve daha stabil veri analizleri ile anne adayının güvenli bir gebelik süreci geçirmesi sağlanacaktır.

İlk defa böyle bir sistem mobile indirgenmiştir. Muadil sistemlerin tamamı hastane ortamındaki medikal cihazlarda var olup mobile entegre bir FKHMS henüz kullanılmamıştır.

Bu tez çalışması, Acıbadem Sağlık Grubu ve İstanbul Aydın Üniversitesine ait ortak bir SAN-TEZ projesi olup, 0763.STZ.2014 numarası ile, Bilim, Sanayi ve Teknoloji Bakanlığı tarafından, 2014 yılında destek almış ve 2016 yılı sonlarında başarıyla tamamlanmıştır.

Aynı zamanda tez konusunun makalesi Atlanta'da yapılan Compsac 2016 konferansında yayınlanmış olup sunumu yapılmıştır. Ek 4'te tez çalışmasının makalesine yer verilmiştir.

5. KAYNAKÇA

[1] TEKİNGÜNDÜZ, S., KURTULDU, A., & Türkkan, I. Ş. I. K. (2017). Sağlık Hizmetlerinde Eşitsizlik ve Etik. Aksaray Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 8(4), 32-43.

[2] Sağlıkta Devrim Yaratan Teknolojiler, https://medium.com/@HipoDoc, Alındığı Tarih: Nisan 2018

[3] Groberman, D., Slonim, T., Hayun, S., Rotem, J., & Nenner, M. (2016). U.S. Patent Application No. 14/917,968.

[4] AKTAŞ, S., & Osmanağaoğlu, M. A. (2017). İntrapartum Elektronık Fetal Monıtorızasyon Uygulaması Ve Sağlık Profesyonellerının Sorumlulukları. Life Sciences, 12(1), 14-29.

[5] Istepanian, R., Laxminarayan, S., & Pattichis, C. S. (2006). M-health. New York, NY: Springer Science+ Business Media, Incorporated.

[6] Zach, L., Chudáček, V., Kužílek, J., Spilka, J., Huptych, M., Burša, M., & Lhotská, L. (2011, September). Mobile CTG—Fetal heart rate assessment using Android platform. In Computing in Cardiology, 2011 (pp. 249-252). IEEE.

[7] Andersson, S. U. S. A. N. N. E. (2011). Acceleration and deceleration detection and baseline estimation. Göteborg: Chalmers University of Technology.

[8] UnbornHeart, http://www.unbornheart.com/ , Alındığı Tarih: Mayıs 2018

[9] AKTAŞ, S., & Osmanağaoğlu, M. A. (2017). İNTRAPARTUM ELEKTRONIK FETAL MONITORIZASYON UYGULAMASI VE SAĞLIK PROFESYONELLERININ SORUMLULUKLARI. Life Sciences, 12(1), 14-29.

[10] YAĞMUR, H., & YÜKSEL, A. (2008). Kordosentez. Turkiye Klinikleri Journal of Gynecology Obstetrics-Special Topics, 1(1), 82-87.

[11] Yılmaz, M., İsaoğlu, Ü., & Kadanalı, S. (2009). Investigation of the caesarean section casea in our clinic between 2002 and 2007.

[12] Lee, C. Y., Di, P. L., & O'lane, J. M. (1975). A study of fetal heart rate acceleration patterns. Obstetrics and Gynecology, 45(2), 142-146.

[13] Hon, E. H., & Quilligan, E. J. (1996). The classification of fetal heart rate. Childbirth: The medicalization of obstetrics, 2, 339.

[14] Haverkamp, A. D., Thompson, H. E., McFee, J. G., & Cetrulo, C. (1976). The evaluation of continuous fetal heart rate monitoring in high-risk pregnancy. American Journal of Obstetrics and Gynecology, 125(3), 310-320.

[15] James, L. S., Morishima, H. O., Daniel, S. S., Bowe, E. T., Cohen, H., & Niemann, W. H. (1972). Mechanism of late deceleration of the fetal heart rate. American Journal of Obstetrics & Gynecology, 113(5), 578-582.

[16] O'Leary, J. A., Andrinopoulos, G. C., & Giordano, P. C. (1980). Variable decelerations and the nonstress test: Anindication of cord compromise. American Journal of Obstetrics & Gynecology, 137(6), 704-706.

[17] Low, J. A., Pickersgill, H., Killen, H., & Derrick, E. J. (2001). The prediction and prevention of intrapartum fetal asphyxia in term pregnancies. American Journal of Obstetrics & Gynecology, 184(4), 724-730.

[18] Montan, S., Arulkumaran, S., Nyman, M., & Ratnam, S. S. (1992). Vibro-acoustic stimulation does not alter the duration of high and low fetal heart rate variability episodes. Journal of Perinatal Medicine-Official Journal of the WAPM, 20(5), 331-336.

[19] Hershenson, M., Munsinger, H., & Kessen, W. (1965). Preference for shapes of intermediate variability in the newborn human. Science, 147(3658), 630-631.

[20] Paul, R. H., Suidan, A. K., Yeh, S. Y., Schifrin, B. S., & Hon, E. H. (1975). Clinical fetal monitoring: VII. The evaluation and significance of intrapartum baseline FHR variability. American Journal of Obstetrics & Gynecology, 123(2), 206-210.

[21] Chandraharan, E., & Arulkumaran, S. (2007). Prevention of birth asphyxia: responding appropriately to cardiotocograph (CTG) traces. Best Practice & Research Clinical Obstetrics & Gynaecology, 21(4), 609-624.

[22] AKTAŞ, S., & Osmanağaoğlu, M. A. (2017). İNTRAPARTUM ELEKTRONIK FETAL MONITORIZASYON UYGULAMASI VE SAĞLIK PROFESYONELLERININ SORUMLULUKLARI. Life Sciences, 12(1), 14-29.

[23] Singh, N. (2016). Study of Google Firebase API for Android. International Journal of Innovative Research in Computer and Communication Engineering, 4(9), 16738-16743.

[24] Stonehem, B. (2016). Google Android Firebase: Learning the Basics. First Rank.

[25] Moon, T. K. (1996). The expectation-maximization algorithm. IEEE Signal processing magazine, 13(6), 47-60.

[26] Erman, J., Arlitt, M., & Mahanti, A. (2006, September). Traffic classification using clustering algorithms. In Proceedings of the 2006 SIGCOMM workshop on Mining network data (pp. 281-286). ACM.

[27] Dawes, G. S., Moulden, M., & Redman, C. W. G. (1990). Criteria for the design of fetal heart rate analysis systems. International journal of bio-medical computing, 25(4), 287-294.

[28] Serra, V., Moulden, M., Bellver, J., & Redman, C. W. G. (2008). The value of the short-term fetal heart rate variation for timing the delivery of growth-retarded fetuses. BJOG: An International Journal of Obstetrics & Gynaecology, 115(9), 1101-1107.

[29] American College of Obstetricians and Gynecologists. (2009). ACOG Practice Bulletin No. 106: Intrapartum fetal heart rate monitoring: nomenclature, interpretation, and general management principles. Obstetrics and gynecology, 114(1), 192.

[30] Android İşletim Sistemi, https://tr.wikipedia.org/wiki/Android_(i%C5%9Fletim_sistemi), Alındığı Tarih: Mayıs 2018

[31] Firebase Authentication, https://firebase.google.com/docs/auth/ , Alındığı Tarih: Mayıs 2018

[32] Firebase RealTime Database, https://firebase.google.com/docs/database/ , Alındığı Tarih: Mayıs 2018

[33] Firebase Storage, https://firebase.google.com/docs/storage/ , Alındığı Tarih: Mayıs 2018

[34] Firebase Crashlytics, https://firebase.google.com/docs/crashlytics/, Alındığı Tarih: Mayıs 2018

[35] CTGViewerLite, http://ctg.ciirc.cvut.cz/software/CTGViewerLite/index.html, Alındığı Tarih: Mayıs 2018

[36] Weka, https://tr.wikipedia.org/wiki/Weka , Alındığı Tarih: Mayıs 2018

[37] Schneider, R., Bauer, A., Barthel, P., & Schmidt, G. (2004, September). libRASCH: a programming framework for signal handling. In Computers in Cardiology, 2004 (pp. 53-56). IEEE.

6. ÖZGEÇMİŞ

HAKAN KANMAZ

Bilgisayar Mühendisi

- Telefon : +90 553 466 90 05
- E-Posta : info.hakankanmaz@gmail.com
- Adres : Maslak / Istanbul / Türkiye
- Website : www.hakankanmaz.com

KİŞİSEL BİLGİLER

- Doğum Tarihi : 11/03/1990
- Medeni Durum: Bekar
- Uyruk : T.C.

EĞİTİM

- Yüksek Lisans : İstanbul Aydın Üniversitesi – Bilgisayar Mühendisliği (2015-2018)
- Lisans : İstanbul Aydın Üniversitesi – Bilgisayar Mühendisliği (İngilizce) (2012-2015)
- Lisans : İstanbul Üniversitesi – Fizik (2010 - 2012)
- Ön Lisans : Sakarya Üniversitesi – Bilgisayar Programcılığı (2008-2010)
- Lise : Sarıyer Vehbi Koç Lisesi (2004 - 2008)

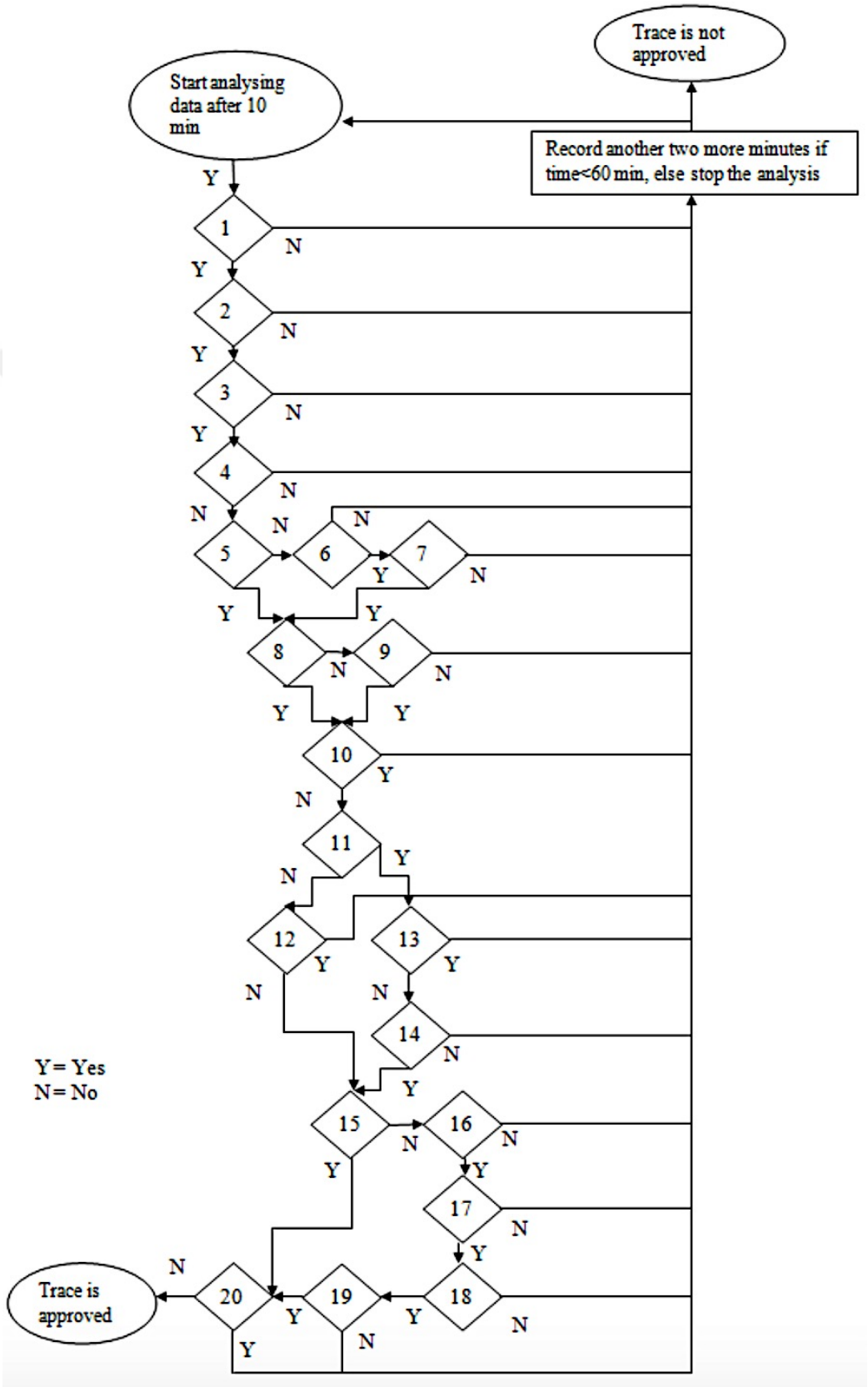
İŞ TECRÜBESİ

- BetBull – Mobil Yazılım Uzmanı (2017- Devam)
- Koç Sistem – Mobil Yazılım Uzmanı (2017)
- Turkcell – Mobil Yazılım Uzmanı (2016 - 2017)
- Acibadem Healthcare Group – Mobil Yazılım Uzmanı (2015- 2016)

- Semantica Software – Mobil Yazılım Uzmanı (2013-2015)

7. EKLER

EK 1 – Dawes / Redman Kriter Diyagramı



1. Does the recording contain one period of high variation?
2. Is STV greater than 3.0 ms?
3. If STV is lower than 4.5 ms, is then the LTV averaged across all episodes of high variations greater than the third percentile for gestational age?
4. Is there any evidence of a high-frequency sinusoidal rhythm?
5. Is there at least one acceleration?
6. Is the fetal movement rate greater than 20/hour?
7. Is the LTV averaged across all episodes of high variation greater than the tenth percentile for gestational age?
8. Is there at least one fetal movement?
9. Does the recording contain at least three accelerations?
10. Are there any decelerations exceeding 100 lost beats?
11. Is the recording less than 30 minutes?
12. Is there more than one deceleration between 21-100 lost beats?
13. Are there any decelerations greater than 20 lost beats?
14. Is the basal heart rate within the interval 116-160 beats/minute?
15. Is the LTV within 3 SDs of its estimated value?
16. Is the STV greater than 5 milliseconds?
17. Is there an episode of high variation with more than 0,5 fetal movements per minute?
18. Is the basal heart rate equal or greater than 120 beats/min?
19. Is the signal loss less than 30%?
20. Are there any suspected artifacts at the end of the recording?

EK 2 – dawes_redman.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <assert.h>

#define _LIBRASCH_BUILD
#include <ra.h>
#include <ra_priv.h>
#include <pl_general.h>
#include <ra_eval.h>
#include <ra_ecg.h>

#include "dawes_redman.h"


LIBRAAPI int
load_ra_plugin(struct plugin_struct *ps)
{
        strcpy(ps->info.name, PLUGIN_NAME);
        strcpy(ps->info.desc, PLUGIN_DESC);
        sprintf(ps->info.build_ts, "%s  %s", __DATE__, __TIME__);
        ps->info.type = PLUGIN_TYPE;
        ps->info.license = PLUGIN_LICENSE;
        sprintf(ps->info.version, PLUGIN_VERSION);

        /* set result infos */
        ps->info.num_results = num_results;
        ps->info.res = ra_alloc_mem(sizeof(results));
        memcpy(ps->info.res, results, sizeof(results));
```

```
	/* set option infos */
	ps->info.num_options = num_options;
	ps->info.opt = ra_alloc_mem(sizeof(options));
	memcpy(ps->info.opt, options, sizeof(options));

	ps->call_gui = NULL;		/* will be set in init_ra_plugin(), eventually */
	ps->proc.get_proc_handle = pl_get_proc_handle;
	ps->proc.free_proc_handle = pl_free_proc_handle;
	ps->proc.do_processing = pl_do_processing;

	return 0;
} /* init_plugin() */


LIBRAAPI int
init_ra_plugin(struct librasch *ra, struct plugin_struct *ps)
{	/* check if gui is available */
	if (ra_plugin_get_by_name(ra, "gui-dawes-redman", 1))
		ps->call_gui = pl_call_gui;

	return 0;
} /* init_ra_plugin() */


LIBRAAPI int
fini_ra_plugin(void)
{
	return 0;
} /* fini_plugin() */


int
```

```
pl_call_gui(proc_handle proc)
{
        if (proc)
                ;

        return 0;
} /* pl_call_gui() */


proc_handle
pl_get_proc_handle(plugin_handle pl)
{
        struct plugin_struct *ps = (struct plugin_struct *)pl;
        struct proc_info *pi = (struct proc_info *)ra_alloc_mem(sizeof(struct proc_info));
        if (!pi)
                return NULL;

        memset(pi, 0, sizeof(struct proc_info));
        pi->handle_id = RA_HANDLE_PROC;
        pi->plugin = pl;

        /* init options */
        pi->options = calloc(1, sizeof(struct ra_dawes_redman));
        set_option_offsets(ps->info.opt, pi->options);
        set_default_options(pi->options);

        return pi;
} /* pl_get_proc_handle() */


void
set_default_options(struct ra_dawes_redman *opt)
```

```
{
        opt->eh = NULL;

        opt->use_start_end_pos = 0;
        opt->start_pos = -1;
        opt->end_pos = -1;

        opt->rh = NULL;
        opt->ch_num = 0;

        opt->use_ignore_value = 1;
        opt->ignore_value = 0.0;

        opt->filter_maternal_pulse = 0;

        opt->ignore_marked_regions = 1;
        opt->ignore_noise_regions = 1;
} /* set_default_options() */


void
set_option_offsets(struct ra_option_infos *opt_inf, struct ra_dawes_redman *opt)
{
        unsigned char *p = (unsigned char *)opt;
        int idx = 0;

        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->eh) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->use_start_end_pos) -
p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->start_pos) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->end_pos) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->rh) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->ch_num) - p);
```

```
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->use_ignore_value) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->ignore_value) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->filter_maternal_pulse)
- p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt-
>ignore_marked_regions) - p);
        opt_inf[idx++].offset = (long)((unsigned char *)&(opt->ignore_noise_regions)
- p);

        /* consistency check (useful when changing option structure) */
        if (idx != num_options)
        {
                fprintf(stderr, "critical error in dawes_redman.c"
                        " - set_option_offsets():\n number of options(%ld) !=
idx(%d)\n",
                        num_options, idx);
                exit(-1);
        }
} /* set_option_offsets() */


void
pl_free_proc_handle(proc_handle proc)
{
        struct proc_info *pi = proc;

        if (pi->options)
        {
                struct ra_dawes_redman *opt;
                opt = (struct ra_dawes_redman *)pi->options;

                free(opt);
        }
```

```
	if (pi->results)
	{
		int i;
		value_handle *t;

		t = pi->results;
		for (i = 0; i < num_results; i++)
			ra_value_free(t[i]);
		free(t);
	}

	ra_free_mem(proc);
} /* pl_free_proc_handle() */


int
pl_do_processing(proc_handle proc)
{
	int ret = 1;
	struct proc_info *pi = proc;
	struct epoch * epochs = NULL;
	double *buf = NULL;
	long l, n_runs, num, num_valid_epoch;
	value_handle *res;
	struct ra_dawes_redman *opt;

	if (pi == NULL)
		return -1;

	opt = (struct ra_dawes_redman *)pi->options;

	/* if rec-handle was not set in options, use root recording */
```

```
        if (!((struct ra_dawes_redman *)pi->options)->rh)
                ((struct ra_dawes_redman *)pi->options)->rh = ra_rec_get_first(pi->mh, 0);

        /* if no eval-handle was set, use default evaluation */
        if (!opt->eh)
                opt->eh = ra_eval_get_default(pi->mh);

        if ((ret = get_data(pi->options, &epochs, &num)) != 0)
                goto clean_up;
        if (num <= 0)
                goto clean_up;

        n_runs = 1;
        pi->num_result_sets = n_runs;
        pi->num_results = num_results;

        /* init results */
        /* first check if there are already some results are stored (maybe from a
           run before with the same proc-handle) */
        if (pi->results)
        {
                int i;
                value_handle *t;

                t = pi->results;
                for (i = 0; i < num_results; i++)
                        ra_value_free(t[i]);
                free(t);
                pi->results = NULL;
        }
        res = malloc(sizeof(value_handle *) * (pi->num_results * pi->num_result_sets));
```

```
        for (l = 0; l < (pi->num_results * pi->num_result_sets); l++)
        {
                int curr_idx = l % pi->num_results;

                value_handle vh = ra_value_malloc();
                set_meta_info(vh, results[curr_idx].name, results[curr_idx].desc,
RA_INFO_NONE);
                res[l] = vh;
        }
        pi->results = res;

        buf = malloc(sizeof(double) * num);
        for (l = 0; l < num; l++)
                buf[l] = epochs[l].epoch_bpm;
        ra_value_set_double_array(res[FHR_EPOCHS], buf, num);

        if ((ret = filter_data(epochs, num)) != 0)
                goto clean_up;

        if ((ret = estimate_baseline(epochs, num)) != 0)
                goto clean_up;

        num_valid_epoch = 0;
        for (l = 0; l < num; l++)
        {
                buf[l] = epochs[l].baseline_bpm;
                if ((epochs[l].percent_noise < 100) && !epochs[l].interpolated)
                        num_valid_epoch++;
        }
        ra_value_set_double_array(res[FHR_BASELINE], buf, num);
        ra_value_set_long(res[NUM_VALID_EPOCHS], num_valid_epoch);

        if ((ret = get_fhr_variations(epochs, num, res)) != 0)
```

```
                goto clean_up;

/*      if (pi->save_in_eval) */
/*              save(opt->eh, res); */

        ret = 0;

 clean_up:
        if (epochs)
                free(epochs);
        if (buf)
                free(buf);

        return ret;
} /* pl_do_processing() */


int
get_data(struct ra_dawes_redman *opt, struct epoch **epochs, long *num)
{
        int ret;
        long s_use, e_use;
        long n, l, start;
        double *raw = NULL;
        value_handle vh;
        double samplerate;

        ret = 0;
        *num = 0;

        if (opt->rh == NULL)
                return -1;
```

```
	vh = ra_value_malloc();
	if (opt->use_start_end_pos)
	{
		s_use = opt->start_pos;
		e_use = opt->end_pos;
	}
	else
	{
		s_use = 0;
		e_use = 0;
		ra_value_set_number(vh, opt->ch_num);
		if (ra_info_get(opt->rh, RA_INFO_REC_CH_NUM_SAMPLES_L, vh) == 0)
			e_use = ra_value_get_long(vh);
	}
	ra_value_free(vh);

	n = e_use - s_use;
	if (n <= 0)
		return 0;

	raw = malloc(sizeof(double) * n);
	n = (long)ra_raw_get_unit(opt->rh, opt->ch_num, s_use, n, raw);

	identify_invalid_values(opt, raw, n);

	if (opt->eh)
	{
		double scale = get_samplerate(opt->rh, opt->ch_num) / get_samplerate(opt->eh, -1);
		handle_ignore_regions(opt, raw, n, scale);
	}
```

```
/* remove not valid values at the beginning (if there are some) */
start = 0;
for (l = start; l < n; l++)
{
        if (raw[l] != HUGE_VAL)
                break;
}
if (l != start)
{
        n -= l;
        memmove(raw, raw+l, sizeof(double) * n);
}

/* remove not valid values at the end (if there are some) */
for (l = (n-1); l >= 0; l--)
{
        if (raw[l] != HUGE_VAL)
                break;
}
n = l+1;

samplerate = get_samplerate(opt->rh, opt->ch_num);
ret = calc_epochs(raw, n, epochs, num, samplerate);
if (*num <= 0)
{
        if (raw)
                free(raw);
        return 0;
}

/* Sometimes it happens that at the beginning that artifacts are recorded.
   Therefore check in the first 5 minutes if the fhr is < 95 bpm or > 180 bpm.
   Go out of the check if 5 minutes are over or when the first minute is ok. */
```

```
	start = 0;
	for (l = 0; l < *num; l += EPOCHS_MIN)
	{
		long m;
		int ok;

		if ((l >= *num) || (l >= (CHECK_BEGIN_MINUTES * EPOCHS_MIN)))
			break;

		ok = 1;
		for (m = l; m < (l + EPOCHS_MIN); m++)
		{
			if (((*epochs)[m].epoch_raw_bpm < CHECK_BEGIN_MIN_FHR)
			|| ((*epochs)[m].epoch_raw_bpm > CHECK_BEGIN_MAX_FHR))
			{
				ok = 0;
				break;
			}
		}
		if (ok)
			break;
	}
	if (l != start)
	{
		*num -= l;
		memmove(*epochs, (*epochs)+l, sizeof(struct epoch) * (*num));
	}
	if (*num <= 0)
	{
		if (raw)
```

```
                free(raw);
        return 0;
    }

    ret = interpolate_epochs(*epochs, *num);

    if (raw)
        free(raw);

    return ret;
} /* get_data() */


int
identify_invalid_values(struct ra_dawes_redman *opt, double *data, long n)
{
    long l;

    if (opt->use_ignore_value)
    {
        for (l = 0; l < n; l++)
        {
            if (data[l] == opt->ignore_value)
                data[l] = HUGE_VAL;
        }
    }

    for (l = 0; l < n; l++)
    {
        if ((data[l] < MIN_FHR) || (data[l] > MAX_FHR))
            data[l] = HUGE_VAL;
    }
```

```
        for (l = 2; l < n; l++)
        {
                double s, perc;

                if ((data[l] == HUGE_VAL) || (data[l-1] == HUGE_VAL) || (data[l-2]
== HUGE_VAL))
                        continue;

                s = (data[l-2] + data[l-1]) / 2;
                perc = data[l] / s;
                if ((perc < 0.65) || (perc > 1.75))
                {
                        long m;

                        for (m = l-2; m <= l; m++)
                                data[m] = HUGE_VAL;
                }
        }

        if (opt->filter_maternal_pulse)
        {
                double last;
                long start = 0;
                last = data[0];
                for (l = 0; l < n; l++)
                {
                        if (data[l] != HUGE_VAL)
                        {
                                last = data[l];
                                break;
                        }
                }
                start = l + 1;
```

```
                for (l = start; l < n; l++)
                {
                        if (data[l] == HUGE_VAL)
                                continue;

                        if (((last - data[l]) >= MIN_DIFF_MATERNAL_PULSE) &&
data[l] <= 100)
                        {
                                data[l] = HUGE_VAL;
                                continue;
                        }

                        last = data[l];
                }
        }

        return 0;
} /* identify_invalid_values() */


double
get_samplerate(any_handle h, long ch_num)
{
        value_handle vh;
        double samplerate = 0.0;
        meas_handle mh;

        mh = ra_meas_handle_from_any_handle(h);

        vh = ra_value_malloc();

        if (ch_num == -1)
```

```
        {
                ra_info_get(mh, RA_INFO_MAX_SAMPLERATE_D, vh);
                samplerate = ra_value_get_double(vh);
        }
        else
        {
                ra_value_set_number(vh, ch_num);
                ra_info_get(h, RA_INFO_REC_CH_SAMPLERATE_D, vh);
                samplerate = ra_value_get_double(vh);
        }

        ra_value_free(vh);

        return samplerate;
} /* get_samplerate() */


void
handle_ignore_regions(struct ra_dawes_redman *opt, double *data, long n, double
scale)
{
        prop_handle annot_prop;
        class_handle annot_clh;
        long l, num_annot;
        value_handle vh, vh_ev_ids;
        const void **clhs;
        const long *ev_ids;
        const char **annots;

        if (!opt->eh)
                return;

        vh = ra_value_malloc();
```

```
	ra_class_get(opt->eh, "annotation", vh);
	clhs = ra_value_get_voidp_array(vh);
	if (clhs == NULL)
	{
		ra_value_free(vh);
		return;
	}
	annot_clh = (class_handle)(clhs[0]);

	if ((annot_prop = ra_prop_get(annot_clh, "annotation")) == NULL)
	{
		ra_value_free(vh);
		return;
	}

	vh_ev_ids = ra_value_malloc();
	if (ra_prop_get_value_all(annot_prop, vh_ev_ids, NULL, vh) != 0)
	{
		ra_value_free(vh);
		ra_value_free(vh_ev_ids);
		return;
	}
	ev_ids = ra_value_get_long_array(vh_ev_ids);
	annots = ra_value_get_string_array(vh);

	num_annot = ra_value_get_num_elem(vh_ev_ids);
	for (l = 0; l < num_annot; l++)
	{
		long m, s, e;
		int ignore = 0;
		int noise = 0;

		if (ra_class_get_event_pos(annot_clh, ev_ids[l], &s, &e) != 0)
```

```
                continue;

            if (strstr(annots[l], "#IGNORE#") != NULL)
                ignore = 1;
            if (strstr(annots[l], "#NOISE#") != NULL)
                noise = 1;

            /* decide if region should be ignored */
            if ((opt->ignore_marked_regions && ignore) || (opt->ignore_noise_regions && noise))
            {
                s = (long)((double)s * scale);
                e = (long)((double)e * scale);

                if (e >= num_annot)
                    e = num_annot-1;
                /* yes, set ignored values to HUGE_VAL */
                for (m = s; m <= e; m++)
                {
                    if (m >= n)
                        break;
                    data[m] = HUGE_VAL;
                }
            }
        }
        ra_value_free(vh);
        ra_value_free(vh_ev_ids);
} /* handle_ignore_regions() */


int
calc_epochs(double *raw, long n_raw, struct epoch **epochs, long *n_epochs, double samplerate)
```

```
{
        long l, curr;
        double epoch_su;

        *n_epochs = 0;
        *epochs = malloc(sizeof(struct epoch) * n_raw);
        memset(*epochs, 0, sizeof(struct epoch) * n_raw);

        epoch_su = EPOCH_LEN * samplerate;

        curr = 0;
        while (curr < n_raw)
        {
                double sum = 0.0;
                long n_sum = 0;

                for (l = curr; l < (curr + epoch_su); l++)
                {
                        if (l >= n_raw)
                                break;

                        if (raw[l] != HUGE_VAL)
                        {
                                sum += raw[l];
                                n_sum++;
                        }
                }
                if (n_sum <= 0)
                {
                        (*epochs)[*n_epochs].epoch_raw_bpm = 0;
                        (*epochs)[*n_epochs].percent_noise = 100;
                }
                else
```

```
                {
                        (*epochs)[*n_epochs].epoch_raw_bpm = sum / n_sum;
                        (*epochs)[*n_epochs].percent_noise = ((epoch_su -
(double)n_sum) / epoch_su) * 100;
                }
                (*n_epochs)++;

                curr += (long)epoch_su;
        }

        *epochs = realloc(*epochs, sizeof(struct epoch) * (*n_epochs));

        for (l = 0; l < (*n_epochs); l++)
        {
                if ((*epochs)[l].epoch_raw_bpm == 0)
                        (*epochs)[l].epoch_raw_ms = 0.0;
                else
                        (*epochs)[l].epoch_raw_ms = 60000.0 /
(*epochs)[l].epoch_raw_bpm;
        }

        return 0;
} /* calc_epochs() */


int
interpolate_epochs(struct epoch *epochs, long num)
{
        long mode, mode_cnt;
        long l, m;
        double last_ok;
        long last_ok_idx;
        int in_75_jump = 0;
```

```
	/* first get mode of epochs */
	mode = get_mode(epochs, num, &mode_cnt);

	/* copy raw epoch values to values which will be used */
	for (l = 0; l < num; l++)
	{
		epochs[l].epoch_bpm = epochs[l].epoch_raw_bpm;
		epochs[l].epoch_ms = epochs[l].epoch_raw_ms;
	}

	last_ok = (double)mode;
	last_ok_idx = 0;
	for (l = 1; l < num; l++)
	{
		double add, diff;
		long cnt;

		if (epochs[l].epoch_bpm == 0)
			continue;

		diff = fabs(epochs[l].epoch_ms - epochs[l-1].epoch_ms);
		if ((epochs[l-1].epoch_bpm != 0) && (diff >= 75))
			in_75_jump = 1;

		if (in_75_jump == 1)
		{
			in_75_jump = 0;
			continue;
		}

		if ((l - last_ok_idx) > 1)
		{
```

```
                        add = (epochs[l].epoch_bpm - last_ok) / (l - last_ok_idx);
                        cnt = 1;
                        for (m = (last_ok_idx+1); m < l; m++)
                        {
                                epochs[m].epoch_bpm = last_ok + (double)cnt * add;
                                epochs[m].percent_noise = 100;
                                epochs[m].interpolated = 1;
                                cnt++;
                        }
                }

                last_ok = epochs[l].epoch_bpm;
                last_ok_idx = l;
                in_75_jump = 0;
        }
        for (m = last_ok_idx; m < num; m++)
        {
                epochs[m].epoch_bpm = last_ok;
                epochs[m].percent_noise = 100;
                epochs[m].interpolated = 1;
        }

        for (l = 0; l < num; l++)
        {
                if (epochs[l].epoch_bpm == 0)
                        epochs[l].epoch_ms = 0;
                else
                        epochs[l].epoch_ms = 60000.0 / epochs[l].epoch_bpm;
        }

        return 0;
} /* interpolate_epochs() */
```

```
long
get_mode(struct epoch *epochs, long n, long *mode_cnt)
{
        long l;
        long mode;
        long *hist = NULL;
        long n_hist, hist_min, hist_max;

        get_histogram(epochs, n, &hist, &n_hist, &hist_min, &hist_max);

        *mode_cnt = 0;
        mode = 0;
        for (l = 0; l < n_hist; l++)
        {
                if ((l + hist_min) == 0)
                        continue; /* skip zero */

                if (hist[l] > (*mode_cnt))
                {
                        *mode_cnt = hist[l];
                        mode = l + hist_min;
                }
        }

        if (hist)
                free(hist);

        return mode;
} /* get_mode() */


int
```

```
get_histogram(struct epoch *epochs, long n, long **hist, long *n_hist, long *min, long
*max)
{
	double mi, ma;
	long l;

	mi = ma = epochs[0].epoch_bpm;
	for (l = 1; l < n; l++)
	{
		if (epochs[l].interpolated)
			continue;

		if (epochs[l].epoch_bpm < mi)
			mi = epochs[l].epoch_bpm;
		if (epochs[l].epoch_bpm > ma)
			ma = epochs[l].epoch_bpm;
	}
	*min = (long)(mi + 0.5);
	*max = (long)(ma + 0.5);
	*n_hist = (*max) - (*min) + 1;
	*hist = malloc(sizeof(long) * (*n_hist));
	memset(*hist, 0, sizeof(long) * (*n_hist));

	for (l = 0; l < n; l++)
	{
		long e;

		if (epochs[l].interpolated)
			continue;

		e = (long)(epochs[l].epoch_bpm + 0.5);
		(*hist)[e - (*min)]++;
	}
```

```
        return 0;
} /* get_histogram() */


/* Digital filter designed by mkfilter/mkshape/gencode   A.J. Fisher
   Command line: /www/usr/fisher/helpers/mkfilter -Bu -Lp -o 4 -a 6.3742032246e-03
0.0000000000e+00 -l */
int
filter_data(struct epoch *epochs, long n)
{
        long l;
        double *buf = NULL;
        double xv[FILT_NZEROS+1], yv[FILT_NPOLES+1];
        long mode, mode_cnt;
        double start;

        /* get start value */
        mode = get_mode(epochs, n, &mode_cnt);
        start = epochs[0].epoch_bpm;
        for (l = 1; l < 64; l++)
        {
                if (l >= n)
                        break;

                if (fabs(epochs[l].epoch_bpm - (double)mode) < fabs(start -
(double)mode))
                        start = epochs[l].epoch_bpm;
        }

        /* first filter pass; store filterd data in buf */
        buf = malloc(sizeof(double) * n);
        for (l = 0; l < (FILT_NZEROS+1); l++)
```

```
            xv[l] = start / FILT_GAIN;
        for (l = 0; l < (FILT_NPOLES+1); l++)
            yv[l] = start;
        for (l = 0; l < n; l++)
        {
            xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4];
            xv[4] = epochs[l].epoch_bpm / FILT_GAIN;
            yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4];
            yv[4] =   (xv[0] + xv[4]) + 4 * (xv[1] + xv[3]) + 6 * xv[2]
                    + ( -0.9006264758 * yv[0]) + (  3.6967592426 * yv[1])
                    + ( -5.6914821245 * yv[2]) + (  3.8953469146 * yv[3]);
            buf[l] = yv[4];
        }

        /* second filter pass in opposite direction to remove phase shift; get data from
buf */
        for (l = 0; l < (FILT_NZEROS+1); l++)
            xv[l] = buf[n-1] / FILT_GAIN;
        for (l = 0; l < (FILT_NPOLES+1); l++)
            yv[l] = buf[n-1];
        for (l = (n-1); l >= 0; l--)
        {
            xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4];
            xv[4] = buf[l] / FILT_GAIN;
            yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4];
            yv[4] =   (xv[0] + xv[4]) + 4 * (xv[1] + xv[3]) + 6 * xv[2]
                    + ( -0.9006264758 * yv[0]) + (  3.6967592426 * yv[1])
                    + ( -5.6914821245 * yv[2]) + (  3.8953469146 * yv[3]);
            epochs[l].filt_bpm = yv[4];
        }

        if (buf)
            free(buf);
```

```
	for (l = 0; l < n; l++)
		epochs[l].filt_ms = 60000.0 / epochs[l].filt_bpm;

	return 0;
} /* filter_data() */


int
estimate_baseline(struct epoch *epochs, long n)
{
	long l;
	long mod_mode;
	double mod_mode_ms;
	double bound_add = 60;

	mod_mode = get_mod_mode(epochs, n);
	mod_mode_ms = 60000 / (double)mod_mode;

	for (;;)
	{
		double lower_bound, upper_bound;
		long last_epoch;

		lower_bound = mod_mode_ms - bound_add;
		upper_bound = mod_mode_ms + bound_add;
		last_epoch = 0;
		for (l = 0; l < n; l++)
		{
			if (epochs[l].filt_ms < lower_bound)
				epochs[l].baseline_ms = lower_bound;
			else if (epochs[l].filt_ms > upper_bound)
				epochs[l].baseline_ms = upper_bound;
```

```
                else
                {
                        epochs[l].baseline_ms = epochs[l].filt_ms;
                        last_epoch = l;
                }

                if ((l-last_epoch) > (EPOCHS_MIN * 10))
                        break;

                epochs[l].baseline_bpm = 60000 / epochs[l].baseline_ms;
        }
        if (l >= n)
                break;

        bound_add += 10;
    }

    return 0;
} /* estimate_baseline() */


long
get_mod_mode(struct epoch *epochs, long n)
{
    long l, cnt, cnt_12_5;
    long *hist = NULL;
    long n_hist, hist_min, hist_max;
    long mode, mode_cnt;
    long mod_mode;

    /* get "modified" mode */
    /* first get histogram */
    mode = get_mode(epochs, n, &mode_cnt);
```

```
get_histogram(epochs, n, &hist, &n_hist, &hist_min, &hist_max);
/* and than find peak which fulfill certain criteria  */
cnt = 0;
cnt_12_5 = (long)((double)n * 0.125) + 1;
mod_mode = mode;
for (l = 0; l < (n_hist-5); l++)
{
        long m;
        int cont = 0;

        cnt += hist[l];
        if (cnt < cnt_12_5)
                continue;

        for (m = l; m < (l + 5); m++)
        {
                if (hist[l] <= hist[m])
                {
                        cont = 1;
                        break;
                }
        }
        if (cont)
                continue;

        if (((double)hist[l] / (double)n) >= 0.5)
        {
                mod_mode = hist_min + hist[l];
                break;
        }
        if (((60000/(double)hist[l]) - (60000/(double)mode_cnt)) <= 30)
        {
                mod_mode = hist_min + hist[l];
```

```
                        break;
                }
        }

        if (hist)
                free(hist);

        return mod_mode;
} /* get_mod_mode() */


int
get_fhr_variations(struct epoch *epochs, long n, value_handle *res)
{
        double sig_lost = 0;
        double bhr = 0.0;
        long n_accel_10 = 0;
        long *pos_accel_10 = NULL;
        long n_accel_15 = 0;
        long *pos_accel_15 = NULL;
        long n_lost_beats_20 = 0;
        long n_lost_beats_21_100 = 0;
        long n_lost_beats_101 = 0;
        double *minute_range = NULL;
        double *minute_range_ms = NULL;
        long n_min = 0;
        long n_high_variations = 0;
        long n_low_variations = 0;
        double ltv_bpm = 0.0;
        double ltv_ms = 0.0;
        double ltv_high_bpm = 0.0;
        double ltv_high_ms = 0.0;
        double stv = 0.0;
```

```
    calc_signal_lost(epochs, n, &sig_lost);
    ra_value_set_double(res[SIGNAL_LOST], sig_lost);

    calc_basal_heart_rate(epochs, n, &bhr);
    ra_value_set_double(res[BASAL_FHR], bhr);

    find_accelerations(epochs, n, &n_accel_10, &pos_accel_10, &n_accel_15,
&pos_accel_15);
    ra_value_set_long(res[ACCEL_10], n_accel_10);
    ra_value_set_long_array(res[POS_ACCEL_10], pos_accel_10, n_accel_10);
    ra_value_set_long(res[ACCEL_15], n_accel_15);
    ra_value_set_long_array(res[POS_ACCEL_15], pos_accel_15, n_accel_15);
    if (pos_accel_10)
        free(pos_accel_10);
    if (pos_accel_15)
        free(pos_accel_15);

    find_decelerations(epochs, n, &n_lost_beats_20, &n_lost_beats_21_100,
&n_lost_beats_101);
    ra_value_set_long(res[LOST_BEATS_20], n_lost_beats_20);
    ra_value_set_long(res[LOST_BEATS_21_100], n_lost_beats_21_100);
    ra_value_set_long(res[LOST_BEATS_101], n_lost_beats_101);

    find_variations(epochs, n, &ltv_bpm, &ltv_ms, &ltv_high_bpm,
&ltv_high_ms, &n_high_variations, &n_low_variations,
                    &minute_range, &minute_range_ms, &n_min);
    ra_value_set_double_array(res[MINUTE_RANGE], minute_range, n_min);
    ra_value_set_double_array(res[MINUTE_RANGE_MS], minute_range_ms,
n_min);
    ra_value_set_long(res[HIGH_VARIATIONS], n_high_variations);
    ra_value_set_long(res[LOW_VARIATIONS], n_low_variations);
    ra_value_set_double(res[LONG_TERM_VARIATIONS_BPM], ltv_bpm);
```

```
        ra_value_set_double(res[LONG_TERM_VARIATIONS_MS], ltv_ms);
        ra_value_set_double(res[LONG_TERM_VARIATIONS_HIGH_BPM],
ltv_high_bpm);
        ra_value_set_double(res[LONG_TERM_VARIATIONS_HIGH_MS],
ltv_high_ms);

        calc_short_term_variations(epochs, n, &stv);
        ra_value_set_double(res[SHORT_TERM_VARIATIONS], stv);

        if (minute_range)
                free(minute_range);
        if (minute_range_ms)
                free(minute_range_ms);

        return 0;
} /* get_fhr_variations() */


int
calc_signal_lost(struct epoch *epochs, long n, double *sig_lost)
{
        long l, cnt;

        *sig_lost = 0;

        cnt = 0;
        for (l = 0; l < n; l++)
                *sig_lost += epochs[l].percent_noise;
        *sig_lost /= (double)n;

        return 0;
} /* calc_signal_lost() */
```

```
int
find_accelerations(struct epoch *epochs, long n, long *n_accel_10, long **pos_accel_10,
                   long *n_accel_15, long **pos_accel_15)
{
        long l;
        long cnt_above, above_start;
        double max_excursion;

        *n_accel_10 = 0;
        *pos_accel_10 = NULL;
        *n_accel_15 = 0;
        *pos_accel_15 = NULL;

        cnt_above = 0;
        max_excursion = 0;
        above_start = -1;
        for (l = 0; l < n; l++)
        {
                double diff = epochs[l].epoch_bpm - epochs[l].baseline_bpm;
                if (diff > 0)
                {
                        if (above_start == -1)
                                above_start = l;

                        cnt_above++;
                        if (diff > max_excursion)
                                max_excursion = diff;
                        continue;
                }

                if (cnt_above > 4)
```

```
        {
                long m, max_pos;
                double max;

                max = epochs[above_start].epoch_bpm;
                max_pos = above_start;

                for (m = above_start; m < l; m++)
                {
                        epochs[m].in_accel = 1;
                        if (epochs[m].epoch_bpm > max)
                        {
                                max = epochs[m].epoch_bpm;
                                max_pos = m;
                        }
                }

                if ((max_excursion > 10) && (max_excursion <= 15))
                {
                        (*n_accel_10)++;
                        (*pos_accel_10) = (long *)realloc((*pos_accel_10),
sizeof(long) * (*n_accel_10));
                        (*pos_accel_10)[(*n_accel_10) - 1] = max_pos;
                }
                if (max_excursion > 15)
                {
                        (*n_accel_15)++;
                        (*pos_accel_15) = (long *)realloc((*pos_accel_15),
sizeof(long) * (*n_accel_15));
                        (*pos_accel_15)[(*n_accel_15) - 1] = max_pos;
                }
        }
```

```
                cnt_above = 0;
                max_excursion = 0;
                above_start = -1;
        }

        return 0;
} /* find_accelerations() */


int
find_decelerations(struct epoch *epochs, long n, long *n_lost_beats_20, long *n_lost_beats_21_100, long *n_lost_beats_101)
{
        long l;
        long cnt_below, below_start;
        double max_excursion;

        *n_lost_beats_20 = 0;
        *n_lost_beats_21_100 = 0;
        *n_lost_beats_101 = 0;

        cnt_below = 0;
        max_excursion = 0;
        below_start = -1;
        for (l = 0; l < n; l++)
        {
                double diff = epochs[l].epoch_bpm - epochs[l].baseline_bpm;
                if (diff < 0)
                {
                        if (below_start == -1)
                                below_start = l;

                        cnt_below++;
```

```
            if (diff < max_excursion)
                max_excursion = diff;
            continue;
        }

        if (((cnt_below > (EPOCHS_MIN/2)) && (max_excursion < -20)) ||
           ((cnt_below > EPOCHS_MIN) && (max_excursion < -10)))
        {
            long m;
            double num_beats_baseline = 0;
            double num_beats_epoch = 0;
            double div = 1.0 / (double)EPOCHS_MIN;

            for (m = below_start; m < l; m++)
            {
                epochs[m].in_decel = 1;

                num_beats_baseline += (epochs[m].baseline_bpm / div);
                num_beats_epoch += (epochs[m].epoch_bpm / div);
            }

            diff = num_beats_baseline - num_beats_epoch;
            if (diff <= 20)
                (*n_lost_beats_20)++;
            if ((diff >= 21) && (diff <= 100))
                (*n_lost_beats_21_100)++;
            if (diff > 100)
                (*n_lost_beats_101)++;
        }

        cnt_below = 0;
        max_excursion = 0;
        below_start = -1;
```

```
        }

        return 0;
} /* find_decelerations() */


int
find_variations(struct epoch *epochs, long n, double *ltv_bpm, double *ltv_ms,
double *ltv_high_bpm, double *ltv_high_ms,
                long *n_high_variations, long *n_low_variations, double
**minute_range, double **minute_range_ms, long *n_min)
{
        long l, n_use;
        int *ignore_minute = NULL;
        long n_minutes_ok = 0;
        int *in_high_variation = NULL;
        int *in_low_variation = NULL;

        *ltv_bpm = 0.0;
        *ltv_ms = 0.0;
        *ltv_high_bpm = 0.0;
        *ltv_high_ms = 0.0;
        *n_high_variations = 0;
        *n_low_variations = 0;

        *n_min = (long)((double)n / EPOCHS_MIN);
        ignore_minute = calloc(*n_min, sizeof(int));
        *minute_range = calloc(*n_min, sizeof(double));
        *minute_range_ms = calloc(*n_min, sizeof(double));
        in_high_variation = calloc(*n_min, sizeof(int));
        in_low_variation = calloc(*n_min, sizeof(int));

        n_use = n - (n % EPOCHS_MIN); /* use only complete minutes */
```

```
    for (l = 0; l < n_use; l += EPOCHS_MIN)
    {
        long m;
        double sig_lost = 0.0;

        for (m = l; m < (l + EPOCHS_MIN); m++)
        {
            if (epochs[m].in_decel)
            {
                ignore_minute[l/EPOCHS_MIN] = 1;
                break;
            }

            sig_lost += epochs[l].percent_noise;
        }
        sig_lost /= (double)EPOCHS_MIN;
        if (sig_lost > 50.0)
            ignore_minute[l/EPOCHS_MIN] = 1;
    }

    for (l = 0; l < n_use; l += EPOCHS_MIN)
    {
        long m;
        double max, min, max_ms, min_ms;

        if (ignore_minute[l/EPOCHS_MIN])
            continue;

        max = min = 0;
        max_ms = min_ms = 0.0;
        for (m = l; m < (l + EPOCHS_MIN); m++)
        {
            double diff = epochs[m].epoch_bpm - epochs[m].baseline_bpm;
```

```
                if (diff > max)
                {
                        max = diff;
                        max_ms = 60000.0/epochs[m].epoch_bpm -
60000.0/epochs[m].baseline_bpm;
                }
                if (diff < min)
                {
                        min = diff;
                        min_ms = 60000.0/epochs[m].epoch_bpm -
60000.0/epochs[m].baseline_bpm;
                }
        }
        (*minute_range)[l/EPOCHS_MIN]= max + fabs(min);
        (*minute_range_ms)[l/EPOCHS_MIN] = fabs(max_ms) +
fabs(min_ms);
        (*ltv_bpm) += (*minute_range)[l/EPOCHS_MIN];
        (*ltv_ms) += (*minute_range_ms)[l/EPOCHS_MIN];
        n_minutes_ok++;
    }
    (*ltv_bpm) /= (double)n_minutes_ok;
    (*ltv_ms) /= (double)n_minutes_ok;

    for (l = 5; l < *n_min; l++)
    {
        long m;
        int below_cnt = 0;
        int above_cnt = 0;
        long s, e;

        for (m = (l - 5); m <= l; m++)
        {
            if (ignore_minute[m])
```

```
                continue;

            if ((*minute_range_ms)[m] <= 30)
                below_cnt++;
            if ((*minute_range_ms)[m] >= 32)
                above_cnt++;
        }

        s = (l - 5) * EPOCHS_MIN;
        e = (l + 1) * EPOCHS_MIN;

        if (below_cnt >= 5)
        {
            for (m = (l - 5); m <= l; m++)
                in_low_variation[m] = 1;
            for (m = s; m < e; m++)
                epochs[m].in_low_variation = 1;
        }
        if (above_cnt >= 5)
        {
            for (m = (l - 5); m <= l; m++)
                in_high_variation[m] = 1;
            for (m = s; m < e; m++)
                epochs[m].in_high_variation = 1;
        }
    }

    for (l = 0; l < *n_min; l++)
    {
        if (in_low_variation[l])
            (*n_low_variations)++;
        if (in_high_variation[l])
        {
```

```
                        (*n_high_variations)++;
                        (*ltv_high_bpm) += (*minute_range)[l];
                        (*ltv_high_ms) += (*minute_range_ms)[l];
                }
        }
        if ((*n_high_variations) > 0)
        {
                (*ltv_high_bpm) /= (double)(*n_high_variations);
                (*ltv_high_ms) /= (double)(*n_high_variations);
        }

        if (ignore_minute)
                free(ignore_minute);
        if (in_low_variation)
                free(in_low_variation);
        if (in_high_variation)
                free(in_high_variation);

        return 0;
} /* find_variations() */


int
calc_short_term_variations(struct epoch *epochs, long n, double *svt)
{
        long n_min, l, n_use;
        int *ignore_minute = NULL;
        long n_minutes_ok = 0;

        *svt = 0.0;

        n_min = (long)((double)n / EPOCHS_MIN);
        ignore_minute = calloc(n_min, sizeof(int));
```

```
    n_use = n - (n % EPOCHS_MIN); /* use only complete minutes */
    for (l = 0; l < n_use; l += EPOCHS_MIN)
    {
        long m;
        double sig_lost = 0.0;

        for (m = l; m < (l + EPOCHS_MIN); m++)
        {
            if (epochs[m].in_decel)
            {
                ignore_minute[l/EPOCHS_MIN] = 1;
                break;
            }

            sig_lost += epochs[m].percent_noise;
        }
        sig_lost /= (double)EPOCHS_MIN;
        if (sig_lost > 50.0)
            ignore_minute[l/EPOCHS_MIN] = 1;
    }

    for (l = 0; l < n_use; l += EPOCHS_MIN)
    {
        long m, cnt;
        double s = 0.0;

        if (ignore_minute[l/EPOCHS_MIN])
            continue;

        cnt = 0;
        for (m = l+1; m < (l + EPOCHS_MIN); m++)
        {
```

```
                        if (epochs[m].interpolated || epochs[m-1].interpolated)
                                continue;
                        if ((epochs[m].epoch_ms == 0) || (epochs[m-1].epoch_ms == 0))
                                continue;

                        s += fabs((epochs[m].epoch_ms - epochs[m-1].epoch_ms));
                        cnt++;
                }
                (*svt) += (s / (double)cnt);
                n_minutes_ok++;
        }

        if (n_minutes_ok > 0)
                (*svt) /= (double)n_minutes_ok;

        if (ignore_minute)
                free(ignore_minute);

        return 0;
} /* calc_short_term_variations() */


int
calc_basal_heart_rate(struct epoch *epochs, long n, double *bhr)
{
        long l, cnt;

        cnt = 0;
        for (l = 0; l < n; l++)
        {
                if (!epochs[l].in_low_variation)
                        continue;
                (*bhr) += epochs[l].epoch_bpm;
```

```
                cnt++;
        }

        if (cnt > 0)
                (*bhr) /= (double)cnt;
        else
                (*bhr) = (double)get_mod_mode(epochs, n);

        return 0;
} /* calc_basal_heart_rate() */


/* TODO: extend eval-structure to handle calculation-results */
int
save(proc_handle proc, eval_handle eh)
{
        if (proc || eh)
                ;

        return 0;
} /* save() */
```

EK 3 – dawes_redman.h

```
#ifndef DAWES_REDMAN_H
#define DAWES_REDMAN_H

#include <ra_defines.h>
#include <ra_dawes_redman.h>


/* ------------------- defines ------------------- */
#define PLUGIN_VERSION   "0.2.0"
```

```
#define PLUGIN_NAME      "dawes-redman"
#define PLUGIN_DESC      gettext_noop("calculate FHR variations using the Dawes/Redman criteria")
#define PLUGIN_TYPE      PLUGIN_PROCESS
#define PLUGIN_LICENSE   LICENSE_LGPL

#define EPOCH_LEN  3.75  /* in seconds */
#define EPOCHS_MIN   16  /* number of epochs in one minute */

#define MIN_DIFF_MATERNAL_PULSE  20 /* minimum difference (in bpm) between succeeding values to identify maternal pulse */

#define MIN_FHR   50
#define MAX_FHR  200


/* defines for artifact-check at the beginning */
#define CHECK_BEGIN_MINUTES    5
#define CHECK_BEGIN_MIN_FHR   95
#define CHECK_BEGIN_MAX_FHR  180

/* Defines for digital filter designed by mkfilter/mkshape/gencode   A.J. Fisher
   Command line: /www/usr/fisher/helpers/mkfilter -Bu -Lp -o 4 -a 6.3742032246e-03 0.0000000000e+00 -l */
#define FILT_NZEROS 4
#define FILT_NPOLES 4
#define FILT_GAIN   6.549261075e+06


enum result_idx
{
        FHR_EPOCHS,                /* epochs of the fetal heart rate (averages over 3.75 seconds) */
```

```
        FHR_BASELINE,           /* baseline of the fetal heart rate */
        SIGNAL_LOST,
        NUM_VALID_EPOCHS,
        BASAL_FHR,
        ACCEL_10,
        POS_ACCEL_10,
        ACCEL_15,
        POS_ACCEL_15,
        LOST_BEATS_20,
        POS_LOST_BEATS_20,
        LOST_BEATS_21_100,
        POS_LOST_BEATS_21_100,
        LOST_BEATS_101,
        POS_LOST_BEATS_101,
        MINUTE_RANGE,
        MINUTE_RANGE_MS,
        HIGH_VARIATIONS,
        MIN_HIGH_VARIATIONS,
        LOW_VARIATIONS,
        MIN_LOW_VARIATIONS,
        LONG_TERM_VARIATIONS_MS,
        LONG_TERM_VARIATIONS_BPM,
        LONG_TERM_VARIATIONS_HIGH_MS,
        LONG_TERM_VARIATIONS_HIGH_BPM,
        SHORT_TERM_VARIATIONS
};


/* -------------------- structures -------------------- */
struct epoch
{
        double percent_noise;

```

```
        /* epoch values coming from the original data */
        double epoch_raw_ms;
        double epoch_raw_bpm;

        /* epoch values after some checking/interpolation */
        double epoch_ms;
        double epoch_bpm;
        int interpolated;       /* flag if epoch value was interpolated */

        /* epoch values after low-pass filter */
        double filt_ms;
        double filt_bpm;

        /* baseline */
        double baseline_ms;
        double baseline_bpm;

        /* fhr variations */
        int in_accel;
        int in_decel;
        int in_low_variation;
        int in_high_variation;
}; /* struct epoch */


struct event_infos
{
        long index;
        double pos;
        double value_1;
        double value_2;

        int dont_use_event;
```

```
}; /* struct event_infos */


/* -------------------- globals -------------------- */
static struct ra_result_infos results[] = {
        {"FHR_EPOCHS", gettext_noop("epochs of the fetal heart rate (averages over 3.75 seconds)"), RA_VALUE_TYPE_DOUBLE_ARRAY, 1},
        {"FHR_BASELINE", gettext_noop("baseline of the fetal heart rate"), RA_VALUE_TYPE_DOUBLE_ARRAY, 1},
        {"SIGNAL_LOST", "signal lost in percent", RA_VALUE_TYPE_DOUBLE, 1},
        {"NUM_VALID_EPOCHS", "number of valid epochs", RA_VALUE_TYPE_LONG, 1},
        {"BASAL_FHR", "basal heart rate of the fetus", RA_VALUE_TYPE_DOUBLE, 1},
        {"ACCEL_10", "number of accelerations > 10bpm and <= 15bpm", RA_VALUE_TYPE_LONG, 1},
        {"POS_ACCEL_10", "positions of the accelerations between 10 and 15bpm", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"ACCEL_15", "number of accelerations > 15bpm", RA_VALUE_TYPE_LONG, 1},
        {"POS_ACCEL_15", "positions of the accelerations > 15bpm", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"LOST_BEATS_20", "decelerations < 20 lost beats", RA_VALUE_TYPE_LONG, 1},
        {"POS_LOST_BEATS_20", "positions of the decelerations < 20", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"LOST_BEATS_21_100", "decelerations between 20 and 100 lost beats", RA_VALUE_TYPE_LONG, 1},
        {"POS_LOST_BEATS_21_100", "postions of the decelerations between 20 and 100", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"LOST_BEATS_101", "decelerations above 100 lost beats", RA_VALUE_TYPE_LONG, 1},
```

```
        {"POS_LOST_BEATS_101", "postions of the decelerations above 100", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"MINUTE_RANGE", "minute range for each minute in bpm", RA_VALUE_TYPE_DOUBLE_ARRAY, 1},
        {"MINUTE_RANGE_MS", "miniute range for each minute in msec", RA_VALUE_TYPE_DOUBLE_ARRAY, 1},
        {"HIGH_VARIATIONS", "number of minutes with high variations", RA_VALUE_TYPE_LONG, 1},
        {"MIN_HIGH_VARIATIONS", "the minutes with the high variations", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"LOW_VARIATIONS", "number of minutes with low variations", RA_VALUE_TYPE_LONG, 1},
        {"MIN_LOW_VARIATIONS", "the minutes with low variations", RA_VALUE_TYPE_LONG_ARRAY, 1},
        {"LONG_TERM_VARIATIONS_MS", "long term variations for all minutes in msec", RA_VALUE_TYPE_DOUBLE, 1},
        {"LONG_TERM_VARIATIONS_BPM", "long term variations for all minutes in bpm", RA_VALUE_TYPE_DOUBLE, 1},
        {"LONG_TERM_VARIATIONS_HIGH_MS", "long term variations for minutes with high variations in msec", RA_VALUE_TYPE_DOUBLE, 1},
        {"LONG_TERM_VARIATIONS_HIGH_BPM", "long term variations for minutes with high variations in bpm", RA_VALUE_TYPE_DOUBLE, 1},
        {"SHORT_TERM_VARIATIONS", "short term variations", RA_VALUE_TYPE_DOUBLE, 1}
};
long num_results = sizeof(results) / sizeof(results[0]);

static struct ra_option_infos options[] = {
        {"eh", gettext_noop("evaluation used; if '= NULL' use default evaluation"), RA_VALUE_TYPE_VOIDP, -1, 0},

        {"use_start_end_pos", gettext_noop("use values between start_pos and end_pos"), RA_VALUE_TYPE_SHORT, -1, 0},
```

```
        {"start_pos", gettext_noop("start-pos in sample-units"),
RA_VALUE_TYPE_LONG, -1, 0},
        {"end_pos", gettext_noop("end-pos in sample-units"),
RA_VALUE_TYPE_LONG, -1, 0},

        {"rh", gettext_noop("recording handle"), RA_VALUE_TYPE_LONG,
-1, 0},
        {"ch_num", gettext_noop("channel with the FHR"),
RA_VALUE_TYPE_LONG, -1, 0},
        {"use_ignore_value", gettext_noop("flag if there are values which are
not valid"), RA_VALUE_TYPE_SHORT, -1, 0},
        {"ignore_value", gettext_noop("value which indicate not valid values"),
RA_VALUE_TYPE_DOUBLE, -1, 0},
        {"filter_maternal_pulse", gettext_noop("run filter to find maternal
pulse and remove these values"),
         RA_VALUE_TYPE_SHORT, -1, 0},
        {"ignore_marked_regions", gettext_noop("ignore regions which are
marked in the annotations with the IGNORE flag"),
         RA_VALUE_TYPE_SHORT, -1, 0},
        {"ignore_noise_regions", gettext_noop("ignore regions which are
marked in the annotations with the NOISE flag"),
         RA_VALUE_TYPE_SHORT, -1, 0},
};
long num_options = sizeof(options) / sizeof(options[0]);


/* ---------- prototypes ---------- */
int pl_call_gui(proc_handle proc);
proc_handle pl_get_proc_handle(plugin_handle pl);
void pl_free_proc_handle(proc_handle proc);
int pl_do_processing(proc_handle proc);

void set_default_options(struct ra_dawes_redman *opt);
```

void set_option_offsets(struct ra_option_infos *opt_inf, struct ra_dawes_redman *p);
int get_data(struct ra_dawes_redman *opt, struct epoch **epochs, long *num);
int identify_invalid_values(struct ra_dawes_redman *opt, double *data, long n);
double get_samplerate(any_handle h, long ch_num);
void handle_ignore_regions(struct ra_dawes_redman *opt, double *data, long n, double scale);
int calc_epochs(double *raw, long n_raw, struct epoch **epochs, long *n_epochs, double samplerate);
int interpolate_epochs(struct epoch *epochs, long num);
long get_mode(struct epoch *epochs, long n, long *mode_cnt);
int get_histogram(struct epoch *epochs, long n, long **hist, long *n_hist, long *min, long *max);
int filter_data(struct epoch *epochs, long n);
int estimate_baseline(struct epoch *epochs, long n);
long get_mod_mode(struct epoch *epochs, long n);
int get_fhr_variations(struct epoch *epochs, long n, value_handle *vh);
int calc_signal_lost(struct epoch *epochs, long n, double *sig_lost);
int find_accelerations(struct epoch *epochs, long n, long *n_accel_10, long **pos_accel_10, long *n_accel_15, long **pos_accel_15);
int find_decelerations(struct epoch *epochs, long n, long *n_lost_beats_20, long *n_lost_beats_21_100, long *n_lost_beats_101);
int find_variations(struct epoch *epochs, long n, double *ltv_bpm, double *ltv_ms, double *ltv_high_bpm, double *ltv_high_ms,
long *n_high_variations, long *n_low_variations, double **minute_range, double **minute_range_ms, long *n_min);
int calc_short_term_variations(struct epoch *epochs, long n, double *svt);

```
int calc_basal_heart_rate(struct epoch *epochs, long n, double *bhr);

/* int save(proc_handle proc, eval_handle eh); */

#endif /* DAWES_REDMAN_H */
```