

TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



MOBİL CİHAZLARDAKİ SENSÖR VERİLERİNİN
TÜMÜNÜ TOPLAMAK VE BİR FRAMEWORK
GELİŞTİRİLMESİ

18011706 – Burak BOZ
18011704 – Bahattin KOÇ

BİLGİSAYAR PROJESİ

Danışman
Doç. Dr. Ferkan YILMAZ

Mayıs, 2021

TEŞEKKÜR

We would like to thank our advisor, Associate Professor Ferkan YILMAZ, who shared his valuable information with us throughout the semester in the realization of this project, took time to listen to our questions with interest, and did his best to be useful to us.

Again, we would like to express our endless thanks to our valuable teachers, Lecturer Ahmet ELBİR and Research Assistant Sercan AYGÜN, who assumed the coordinatorship during the development process of this project, who constantly helped us and informed us, for all the information they gave us.

We would like to thank our other university professors for everything they have given us in our university life and all kinds of contributions and support provided by Yıldız Technical University.

Burak BOZ
Bahattin KOÇ

İÇİNDEKİLER

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
ÖZET	vii
ABSTRACT	ix
1 Giriş	1
2 Ön İnceleme	2
3 Fizibilite	3
3.1 Teknik Fizibilite	3
3.1.1 Yazılımsal Fizibilite	3
3.1.2 Donanımsal Fizibilite	4
3.2 Ekonomik Fizibilite	4
3.3 İşgücü ve Zaman Planlaması	5
4 Sistem Analizi	6
4.1 Senaryo 1	6
4.2 Senaryo 2	8
4.3 Senaryo 3	9
4.4 Senaryo 4	9
4.5 Senaryo 5	10
4.6 Senaryo 6	12
4.7 Senaryo 7	13
4.8 Senaryo 8	15
4.9 Senaryo 9	16
4.10 Senaryo 10	17
4.11 Senaryo 11	19
5 Sistem Tasarımı	21
5.1 Yazılım Tasarımı	21

5.1.1	Android	21
5.1.2	Sunucu	23
5.1.3	Matlab ve Python'da kullanılacak framework	24
5.2	Veritabanı Tasarımı	24
5.3	Girdi-Çıktı Tasarımı	25
6	Uygulama	27
6.1	Android	27
6.2	Sunucu	30
6.3	Veritabanı	31
6.4	Matlab	32
6.5	Python	33
7	Deneyisel Sonuçlar	36
7.1	Suncunun Kapalı Olma Durumu	36
7.1.1	Android	36
7.1.2	Sensörler Ekranı	37
7.1.3	Programın Arkaplane Alınması	38
7.2	Sunucu	38
7.3	Framework	39
7.3.1	MATLAB	39
7.3.2	Python	40
8	Performans Analizi	41
8.1	Android	41
8.2	Sunucu	41
8.3	Framework	42
8.4	MATLAB ve Python	42
9	Sonuç	43
	Referanslar	46
	Özgeçmiş	47

KISALTMA LİSTESİ

TL	Türk Lirası
API	Application Programming Interface
MB	Megabyte
GB	Gigabyte
RAM	Random Access Memory
GHZ	Gigahertz

ŞEKİL LİSTESİ

Şekil 3.1	İş planlaması	5
Şekil 3.2	Gantt diyagramı	5
Şekil 5.1	Ana Ekran Sınıf Diyagramı	21
Şekil 5.2	Sensör Ekranı Sınıf Diyagramı	22
Şekil 5.3	Sunucu Sınıf Diyagramı	23
Şekil 5.4	Matlab ve Python'da kullanılacak frameworkün Sınıf Diyagramı .	24
Şekil 5.5	ER Diyagramı	24
Şekil 5.6	Kullanıcı Arayüzü	25
Şekil 6.1	Karşılama ekranı	27
Şekil 6.2	Kullanıcı arayüzü 2	27
Şekil 6.3	Verilerin iletme geçmesi	28
Şekil 6.4	Cihazdan toplanan sensör verilerinin json hali	29
Şekil 6.5	Verilerin Boyutları	29
Şekil 6.6	Sıkıştırılan Veri	30
Şekil 6.7	Sunucuda Gelen Veriler	30
Şekil 6.8	Veritabanı Tabloları	31
Şekil 6.9	Header Tablosu	31
Şekil 6.10	Sensor values Tablosu	31
Şekil 6.11	Matlabdan Framework'e Ulaşılması	32
Şekil 6.12	Dataların Alınması	32
Şekil 6.13	Işık Sensörlerine Ait Tüm Veriler	32
Şekil 6.14	Framework'ün Ortam Değişkeni Olarak Eklenmesi	33
Şekil 6.15	Python'dan Framework'e Ulaşılması	33
Şekil 6.16	Dataların Alınması	34
Şekil 6.17	Yerçekimi Sensörlerine Ait Tüm Veriler	35
Şekil 7.1	Karşılama Ekranında Sunucuya Ulaşılama Durumu	36
Şekil 7.2	Sensör Ekranında Sunucuya Ulaşılama Durumu	37
Şekil 7.3	/sensors İsteğine Uyumsuz Veri Gelmesi Durumu	38
Şekil 7.4	/verigetir İsteğine Uyumsuz Veri Gelmesi Durumu	39
Şekil 7.5	getDataModel() Fonksiyonunun MATLAB'da Sınanması	39
Şekil 7.6	getDataModel() Fonksiyonunun Python'da Sınanması	40

MOBİL CİHAZLARDAKİ SENSÖR VERİLERİNİN TÜMÜNÜ TOPLAMAK VE BİR FRAMEWORK GELİŞTİRİLMESİ

Burak BOZ
Bahattin KOÇ

Bilgisayar Mühendisliği Bölümü
Bilgisayar Projesi

Danışman: Doç. Dr. Ferkan YILMAZ

Günlük hayatımızda kullandığımız birçok cihaz sensörler sayesinde akıllı hale gelmiş, hayatımıza girmiş ve hayatımızı kolaylaştırmaya devam etmektedir. Bu projede, günümüzde hemen herkeste bulunan mobil cihazlardaki sensör verilerinin toplanmasını, kaydedilmesini ve işlenebilir duruma getirilmesini amaçlamıştır. Android işletim sisteminde tanımlı olan 13 adet sensör bulunmaktadır. Bu sensörlerin bazılarının donanımları tüm cihazlarda bulunmamaktadır. Bazı sensör verileri ise bir donanımdan gelen verilerin farklı şekillerde işlenmesi ile türetilmektedir. Örneğin jiroskop sensörü ivme ölçmektedir. Jiroskop sensöründen gelen veri işlenerek yerçekimi sensörü verisi oluşturulmaktadır. Projemizde düzenlenen bu veriler, saniyelik periyod ile toplanmıştır. Toplanan veriler kullanıcının belirlediği periyodlar ile sıkıştırılarak sunucuya gönderilmiştir. Bu veriler sunucuda önce çözümlenmiş ardından veritabanına uygun şekilde kayıt edilmiştir. Kaydedilen verilere özellikle sayısal veri işlemede çok önemli bir programlama dili olan MATLAB ve yapay zeka sistemlerinde sıkça kullanılan pythondan da ulaşabilmek için JAVA ile bir framework geliştirilmiştir. Bu sayede kaydedilen verilere farklı platformlardan da ulaşılabilirlik kazandırılmıştır.

Sensörler analog dünyanın dijitalleştirilmesindeki en önemli paydaşlardan biridir. Doğada var olan birçok nitelik ve nicelik sensörler sayesinde dijitalleşmiş ve hayatımıza farklı icatların ve buluşların girmesine katkıda bulunmuştur. Bu durum

bize sensör verilerinin toplanmasının ne derece önemli olduğunu göstermektedir. Bu projede toplanan sensör verileri android işletim sistemine sahip cihazları kapsamaktadır. Genel amaç ise bu sensör verilerine sunucu sayesinde uzaktan ulaşabilmek ve farklı platformlardan işleyebilmektir. Projede ileride bir çok cihazdan sensör verisi toplanabileceği hesaba katılmıştır ve bu sebeple internet bant genişliğini yormamak adına verileri iletilirken sıkıştırılarak iletilmesi sağlanmıştır. Veri sıkıştırırken huffman algoritması temelli çalışan bir kütüphaneden faydalanılmıştır. İletilecek veriye göre yaklaşık yüzde 50'ye yakın kazanç sağlanabilmektedir.

Anahtar Kelimeler: Android sensör verileri, Restful API, Huffman, Veri sıkıştırma, JAVA, MATLAB, PYTHON, MYSQL

ABSTRACT

COLLECT TO SENSOR DATAS ON THE MOBILE PHONES AND DEVELOP A FRAMEWORK

Burak BOZ
Bahattin KOÇ

Department of Computer Engineering
Computer Project

Advisor: Assoc. Prof. Dr. Ferkan YILMAZ

Many devices we use in our daily life have become smart with sensors and these sensors have entered our lives and continue to make our lives easier. In this project, it was aimed to collect, record and process sensor data in mobile devices, which are found in almost everyone today.

There are 13 sensors defined in the Android operating system. Some of these sensors are not equipped on all devices. Some sensor data is derived by processing data from a hardware in different ways. For example, the gyroscope sensor measures acceleration. Gravity sensor data is created by processing the data from the gyroscope sensor.

In this project, these organized data were collected in seconds. The collected data was compressed and sent to the server at the periods specified by the user. These data were first analyzed on the server and then recorded in the database accordingly. A framework has been developed with JAVA to access the recorded data from MATLAB, which is a very important programming language especially in numerical data processing, and python, which is frequently used in artificial intelligence systems. In this way, the recorded data can be accessed from different platforms.

Sensors are one of the most important stakeholders in the digitization of the analog world. Many qualities and quantities existing in nature have been digitalized thanks to sensors and have contributed to the introduction of different inventions and inventions into our lives. This shows us how important it is to collect sensor data. The sensor data

collected in this project includes devices with android operating system. The general purpose is to access these sensor data remotely through the server and to process them from different platforms. In the project, it has been taken into account that sensor data can be collected from many devices in the future, and therefore, in order not to tire the internet bandwidth, the data is compressed while transmitting. While compressing the data, a library based on the huffman algorithm was used. According to the data to be transmitted, a gain of approximately 50 percent can be achieved.

Keywords: Android sensor values, Restful API, Huffman algorithm, Data compressing, JAVA, MATLAB, PYTHON, MYSQL

1

Giriş

Doğada varolan çok sayıdaki nitelik ve nicelik, sensörler sayesinde dijitalleştirilerek, şaşırtıcı teknolojik keşifler ve buluşların yapılmasında önemli rol oynamıştır. Bu durum sensörlerden verilerin toplanmasını önemli hale getirmektedir. Hareket, çevre, lokasyon ve insanın duyu özelliklerinden görme, duyma, koku alma, tat alma ve deri ile hissetme gibi nitelikleri teknolojik olarak hayata geçiren sensörlerin kullanım alanları her geçen gün yaygınlaşmaktadır. Mobil cihazlar üzerinde ortalama 10 sensör vardır[1]. Bu sensörlerin tümünün verilerinin toplanması için JAVA dilinde bir framework geliştirmek bu projedeki esas amaçtır. Android mobil cihazlardan toplanan bu sensör verileri, bir sunucuya gönderilecek ve sunucuda bu veriler saklanacaktır.

Gerek duyulması halinde bu verilere geliştirilen framework sayesinde python ve özellikle sayısal veri işlemede önemli bir rolü olan MATLAB aracılığıyla da ulaşılabilecektir. Mobil cihazlardan toplanan bu veriler sunucuya gönderilebilecektir. Bu durumda iletilen verinin boyutu önem kazanmaktadır. Bu sebeple gönderilecek olan veri sıkıştırılarak iletilecektir. İletilen veri içerisinde; hangi kullanıcıdan geldiği, verinin hangi saat diliminde alındığı ve android cihazda mevcut bulunan sensörlerden okunan değerler olacaktır. Bu projede nesne yönelimli programlama metodu kullanılacak ve hakim dil olarak JAVA kullanılacaktır. İnceleyecek kişilerin temel olarak Android işletim sistemine hakim olmaları ve orta düzeyde JAVA bilmeleri, yapılacak işlemlerin anlaşılmasını kolaylaştıracaktır.

2 Ön İnceleme

Android cihazlardaki sensör verileri ile ilgili yapılan çalışmalar incelendiğinde, sıklıkla sadece gerekli olan sensörün verileri ile ilgili işlem yapıldığı saptanmıştır. İncelenen projelerde genellikle sensör bilgileri sadece cihazda kullanıcıya gösterilmiş ya da cihazın veritabanına kaydedilmiştir. Bu tarz uygulamalarda farkedilen en büyük eksik, sensör verilerinin lokal ortamda kalıyor olmasıdır.

Lokal ortamda kalan bilgilere ulaşabilmek için cihaz ile fiziki olarak aynı ortamda bulunmak gerekmektedir. Bu durum, sensör verileriyle yapılabilecek uygulamaların potansiyelinin açığa çıkmasını engellemektedir. Örneğin araç ile yolculuk yapan aşırı süratli bir insanın hızı sensörler yardımıyla ölçülebilir. Fakat bu bilginin sadece cihazda kalması, faydasız bir işlem olacaktır. Eğer bu veriye kurulacak bir sunucu yardımıyla erişebilirsek süratli yolculuğa engel olabilecek ve oluşabilecek kazaların önüne geçebileceğiz.

Restful API'lar ile veri iletimi yapan uygulamalar incelendiğinde ise genellikle gönderilen ve alınan verilerin üzerinde işlem yapılmadan yalın haliyle gönderildiği gözlemlenmiştir. Yine az önceki süratli yolculuk yapan insan üzerinden inceleme yapacak olursak, birçok mobil cihazdan bu verilerin gönderilmesi gerektiği sonucuna ulaşabiliriz. Eğer tüm veriler olduğu gibi yalın halde gönderilirse sunucular ve trafik bandı üzerinde, aslında önüne geçebileceğimiz, bir yük ile karşı karşıya kalmış olacağız. Google veri sıkıştırmasının önemini youtube kanalındaki video ile açıkça ifade etmektedir[2]. Bu bilgi bizi verilerin sıkıştırılarak gönderilmesi gerektiği sonucuna götürmektedir. Eğer veriler sıkıştırılarak gönderilirse yüzde 50 ye yakın tasarruf etmiş olabileceğiz. Bu sayede belki de internet altyapısı yatırımı gerektirmeden mevcut altyapıdan daha iyi faydalanabileceğiz.

İncelemelerimiz sonucunda ulaştığımız sonuç; projemizin mobil cihazlardaki sensör verilerinin tümünün toplanması, verilerin sıkıştırılarak sunuculara iletilmesi ve gerektiği durumda farklı platformlardan da ulaşılabilmesi ve bu işlemlerin en optimum yollar ile sağlanması olmuştur.

3.1 Teknik Fizibilite

Bu bölümde donanımsal ve yazılımsal kaynakların hangi amaçla seçildikleri hakkında açıklamalar yapılacaktır.

3.1.1 Yazılımsal Fizibilite

- **Frontend İşletim Sistemi** Bilindiği üzere mobil cihazlar yılların geçmesi ve yeni işletim sistemi versiyonlarının yayınlanması ile eskimekte ve belli bir süre sonra destek alamamaktadır. Android 8.0'ın seçilme amacı şu an piyasada kullanılabilir durumda olan en eski cihaza dahi destek sunulabilmesidir. Minimum sistem gereksinimi 21 Ağustos 2017'de çıkan Android 8.0 olarak belirlenmiştir.
- **Frontend Yazılım Geliştirme Dili** Projemizdeki gerekli isterler android işletim sistemi üzerinde çalışabilen ve sensör verilerini bize ulaştırabilecek olmasıdır. Bu isterleri ücretsiz karşılayabilecek iki adet dil mevcuttur. Bunlardan biri kotlin, diğeri ise JAVA'dır.

Projenin geliştirilmesinde hem kaynaklardan faydalanabilme hem de piyasadaki hakim konumu sebebiyle JAVA seçilmiştir. JAVA'nın kütüphane ağının geniş olması ve sunulan Google destekli düzenli dökümantasyonu sebebiyle[3] bu projede kullanılan frontend dili olarak seçilmiştir. Uygulama yine ücretsiz olan ve Google kaynaklı Android Studio ile geliştirilmiştir.

- **Backend İşletim Sistemi** Günümüzün en güncel sistemi ve ulaşılabilir işletim sistemi olan Windows 10 bu projede işletim sistemi olarak seçilmiştir. Sunucu sistemi JAVA 8 ve MYSQL desteği sağlayan Xampp versiyon 3.3.0 ile çalışmaktadır.
- **Backend Yazılım Geliştirme Dili** JAVA, sunmuş olduğu önemli backend frameworkleri ile yazılımcının işini epey kolaylaştırmaktadır. Mobil cihazdan

gelen istekler JAVA'nın sunmuş olduđu Gzip kütüphanesi[4] ile sıkıştırılarak gönderilmektedir. Bu sıkıştırılmış veriyi backendde kolayca işleyebilmek için burada da JAVA tercih edilmiştir. Uygulamayı geliştirmek için Eclipse ide'si kullanılmıştır. Sunucuyu ayağa kaldırabilmek için Eclipse idesine Tomcat versiyon 9.0 kurulmuştur.

3.1.2 Donanımsal Fizibilite

- **Mobil Cihaz İçin Minimum Gereksinimler**

1. Android 8.0 işletim sistemi
2. 100 MB boş hafıza
3. 2 GB RAM
4. İnternete erişim

- **Sunucu İçin Minimum Gereksinimler**

1. 64 bit 2 GHZ ve 2 çekirdekli işlemci
2. 1 GB Hafıza
3. 8 GB RAM
4. İnternete erişim

3.2 Ekonomik Fizibilite

Bu projenin tamamlanması için tanınan süre 15 haftadır. Projede 2 adet yazılımcı görev alacaktır. Yapılan planlamaya göre bu yazılımcılar toplam 95 iş günü çalışacaklardır. Başlangıç seviyesi olarak çalışan bir yazılımcının saatlik ücreti 25 TL ve günlük çalışma süresi 8 saatten 200 TL dir. Toplam maliyet 95 gün için 19.000 TL olacaktır.

Test için kullanılacak 2 adet Android işletim sistemli telefonun birim fiyatı 2.000 TL dir. 2 Adet telefon için 4.000 TL maliyet hesaplanmıştır.

Projede kullanılacak olan sunucu test aşamasında lokal olarak kullanılacak fakat sistem entegrasyonlarının tamamlanmasıyla birlikte bir sunucu kiralanacaktır. Bu kiralanacak dedicated sunucunun bir aylık maliyeti ise 50 dolardır. 8 TL sabit kur ile hesaplandığında maliyeti 400 TL dir.

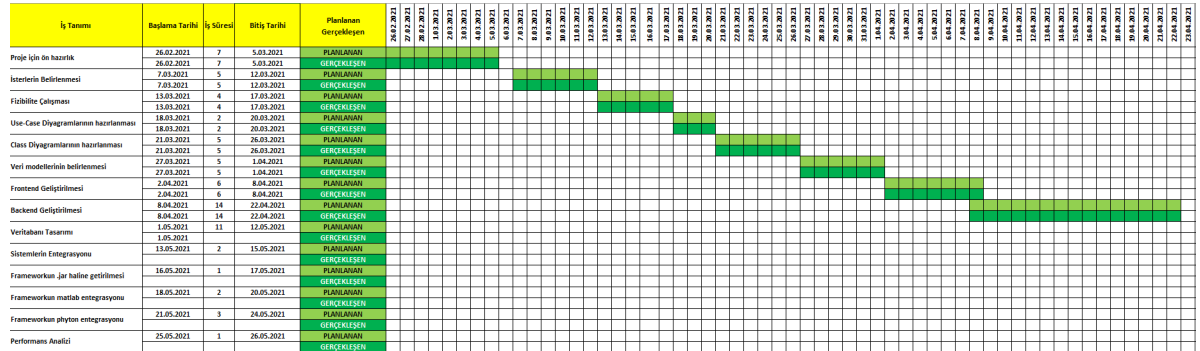
Tüm bu tutarlar hesaplandığında total maliyet 23.400 TL olacaktır.

3.3 İşgücü ve Zaman Planlaması

Aşağıda verilen iş planlaması diyagramı çalışanların hangi gün hangi görevden sorumlu olduklarını göstermektedir. Gantt diyagramı ise planlanan iş ve gerçekleşen iş arasındaki bağlantıyı göstermektedir.

İş ID	İş Tanımı	Başlangıç Tarihi	Bitiş Tarihi	Kişi	İlerleme
1	Proje için ön hazırlık	26.02.2021	5.03.2021	Burak	100%
2	Proje için ön hazırlık	26.02.2021	5.03.2021	Bahattin	100%
3	İsterlerin Belirlenmesi	7.03.2021	12.03.2021	Burak	100%
4	İsterlerin Belirlenmesi	7.03.2021	12.03.2021	Bahattin	100%
5	Fizibilite Çalışması	13.03.2021	17.03.2021	Bahattin	100%
6	Use-Case Diyagramlarının hazırlanması	18.03.2021	20.03.2021	Bahattin	100%
7	Class Diyagramlarının hazırlanması	21.03.2021	26.03.2021	Burak	100%
8	Veri modellerinin belirlenmesi	27.03.2021	1.04.2021	Burak	100%
9	Frontend Geliştirilmesi	2.04.2021	8.04.2021	Burak	100%
10	Backend Geliştirilmesi	8.04.2021	22.04.2021	Burak	100%
11	Backend Geliştirilmesi	8.04.2021	22.04.2021	Bahattin	100%
12	Veritabanı Tasarımı	1.05.2021	12.05.2021	Burak	50%
13	Sistemlerin Entegrasyonu	13.05.2021	15.05.2021	Burak	
14	Frameworkun .jar haline getirilmesi	16.05.2021	17.05.2021	Burak	
15	Frameworkun matlab entegrasyonu	18.05.2021	20.05.2021	Bahattin	
16	Frameworkun phyton entegrasyonu	21.05.2021	24.05.2021	Bahattin	
17	Performans Analizi	25.05.2021	26.05.2021	Burak	
18	Performans Analizi	25.05.2021	26.05.2021	Bahattin	

Şekil 3.1 İş planlaması



Şekil 3.2 Gantt diyagramı

Bu analiz nesne yönelimli olacaktır ve use-case senaryoları içermektedir.

4.1 Senaryo 1

Adı: Olumlu Senaryo

Bağlantılı Diyagramlar: Sensör listesi, kullanıcı bilgisi

Ön Koşullar: Kullanıcının telefonunda minimum android 8.0 işletim sistemi kurulu olmalıdır. Kullanıcının cihazında uygulama kurulu olmalıdır ve cihazda kurulu olan uygulama internete erişebilmelidir. Sunucu aktif olmalıdır.

Aktörler: Kullanıcı, mobil cihaz, sunucu

Senaryo:

Kullanıcı:

- 1 - Kullanıcı uygulamayı çalıştırır.
- 2 - Kullanıcı karşılama ekranını görür.
- 3 - Kullanıcı "Uygulamaya Giriş" butonuna basar.
- 12 - Kullanıcı verilerini paylaşmak istediği sensörleri seçer.
- 13 - Kullanıcı verilerin ne sıklıkla iletileceğini spinner'dan seçer.
- 14 - Kullanıcı "Veri İletimini Başlat" butonuna tıklar.
- 23 - Kullanıcı "İletimi durdur" butonuna basarak veri akışını kesebilir ya da seçtiği sensörleri switchler yardımıyla kaldırabilir. Tamamını kaldırırsa iletim durur.

Mobil Cihaz:

- 4 - Mobil cihaz, sunucuya kontrol isteğini gönderir.
- 7 - Mobil cihaz sunucudan olumlu dönüşü alır.
- 8 - Sensör bilgilerinin görüntülendiği ekran açılır.
- 9 - Cihazın id si alınır.
- 10 - Cihazda mevcut olan sensörler belirlenir.
- 11 - Cihazda bulunan sensörler listeye yanlarında switch olacak şekilde listelenir.
- 12 - Cihazda bulunmayan sensörler listeye eklenirken, switch yerine sensörün bulunmadığını anlatan bir ikon ile birlikte listeye eklenir.
- 15 - Kullanıcının seçmiş olduğu sensör verileri her saniye alınır.
- 15 - Alınan verilere hangi saatte alındığı bilgisi eklenir
- 16 - Kullanıcının seçmiş olduğu frekans ile veriler sıkıştırılır.
- 17 - Veriler sıkıştırılmış halde sunucuya iletilir.
- 22 - Kullanıcı veri iletimini durdurana kadar 15. adıma dönülür.

Sunucu:

- 5 - Sunucu kontrol isteğini alır.
- 6 - Sunucu 200 cevabı döndürür.
- 18 - Sunucu sıkıştırılmış veriyi alır.
- 19 - Sunucu sıkıştırılmış veriyi çözümler.
- 20 - Sunucu çözülen veriyi veritabanına kaydeder.
- 21 - Sunucu mobil cihaza 200 dönüşü yapar.

Alternatif Yollar:

- 7 a - Sunucu aktif değilse ya da olumlu cevap döndüremeyecek durumdaysa kullanıcıya hata gösterilir ve sensör ekranına geçilmez.

13 a - Varsayılan olarak 15 saniyede bir veri iletimi seçilidir.

15 a - Toplanan veri sayısı 7201'i geçerse (Her saniyede bir alınan veri için 60 saniye x 60 x 2 = 2 saati temsil eder.) ilk veri silinir ve listenin sonuna yeni veri eklenir. Bu sayede mobil cihaz belleğinin taşma durumu ortadan kalkar.

22 a - Sunucudan olumsuzlu cevap dönmezse 30 saniyede bir tekrar denenir. Bu arada mobil cihaz verileri toplamaya devam eder. 4. denemede de başarısız olunursa iletim durdurulur ve kullanıcıya bildirilir.

4.2 Senaryo 2

Adı: Sunucu inaktif

Bağlantılı diyagramlar: Olumlu senaryo

Ön koşullar: Kullanıcının telefonunda minimum Android 8.0 işletim sistemi kurulu olmalıdır. Kullanıcının cihazında uygulama kurulu olmalıdır ve cihazda kurulu olan uygulama internete erişebilmelidir. Sunucu inaktif olmalıdır.

Aktörler: Kullanıcı, mobil cihaz, sunucu

Senaryo:

Kullanıcı :

- 1 - Kullanıcı uygulamayı çalıştırır.
- 2 - Kullanıcı karşılama ekranını görür.
- 3 - Kullanıcı "Uygulamaya Giriş" butonuna basar.
- 9 - Kullanıcı uygulamayı kapatır.

Mobil Cihaz:

- 4 - Mobil cihaz sunucuya kontrol isteği gönderir.
- 7 - İstek 10 saniye sonra zaman aşımına uğrar.
- 8 - Kullanıcıya sunucuya bağlanılamadığı bilgisi hata kodu ile birlikte gösterilir.

Sunucu:

5 - Sunucudan cevap dönmez.

4.3 Senaryo 3

Adı: Sensör listesi

Bağlantılı diyagramlar: Olumlu senaryo

Aktörler: Kullanıcı, mobil cihaz

Ön koşullar: Kullanıcı karşılama ekranında "Uygulamayı Başlat" butonuna basmıştır. Sunucudan olumlu dönüş gelmiştir ve sensör ekranına geçilmiştir.

Senaryo:

Mobil Cihaz:

- 1 - Sistemde tanımlanmış olan tüm sensörler belirlenir.
- 2 - Belirlenen sensörlerin cihazda mevcut olup olmadığı kontrol edilir.
- 3 - Mevcut olan sensörler listeye switch ile birlikte eklenir.
- 4 - Mevcut olmayan sensörlere listede switch yerine sensörün mevcut olmadığını anlatan bir ikon eklenir.
- 5 - Mevcut olan sensörlerin değerleri 1 saniyede bir alınır.
- 6 - Alınan bu veriler ekranda mevcut sensörün altında gösterilir.
- 7 - Kullanıcı verilerini paylaşmak istediği sensörün switchini true durumuna getirir.

4.4 Senaryo 4

Adı: MATLAB Kullanılarak Cihaz IDsi ve Tarih Aralığı ile Değer Okuma

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda MATLAB kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, MATLAB, Sunucu

Senaryo:

Kullanıcı:

- 1 - Kullanıcı MATLAB'i çalıştırır
- 2 - Kullanıcı bir değişkene framework atamasını yapar
- 4 - Kullanıcı gerekli olan fonksiyonu çağırır
- 5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer
- 13 - Kullanıcı veriyi görür
- 14 - Kullanıcı programı kapatır

MATLAB:

- 3 - MATLAB frameworkü tanır
- 6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 7 - Fonksiyonu çalıştırır
- 12 - Sunucu üzerinden gelen veri değişkene atanır

Sunucu:

- 8 - MATLAB ile Java üzerinden gelen isteği alır
- 9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir
- 10 - Arama sonucuna ulaşır
- 11 - Arama sonucunu MATLAB'e gönderir

Alternatif Yollar:

- 10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

4.5 Senaryo 5

Adı: MATLAB Kullanılarak Sensör ve Tarih ile Spesifik Değer Arama

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda MATLAB kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, MATLAB, Sunucu

Senaryo:**Kullanıcı:**

- 1 - Kullanıcı MATLAB'i çalıştırır
- 2 - Kullanıcı bir değişkene framework atamasını yapar
- 4 - Kullanıcı gerekli olan fonksiyonu çağırır
- 5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer
- 13 - Kullanıcı sensör adı ve tarih bilgisini girer
- 21 - Kullanıcı sonucu görür
- 22 - Kullanıcı programı kapatır

MATLAB:

- 3 - MATLAB frameworkü tanır
- 6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 7 - Fonksiyonu çalıştırır
- 12 - Sunucu üzerinden gelen veri değişkene atanır
- 14 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 15 - Fonksiyonu çalıştırır
- 20 - Sunucu üzerinden gelen sonuç değişkene atanır

Sunucu:

- 8 - MATLAB ile Java üzerinden gelen isteği alır
- 9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir
- 10 - Arama sonucuna ulaşır
- 11 - Arama sonucunu MATLAB'e gönderir
- 16 - MATLAB ile Java üzerinden gelen isteği alır
- 17 - Önceden çekilmiş olan veri tabanından alınan parametrelere göre bir arama

gerçekleştirir

18 - Arama sonucuna ulaşır

19 - Arama sonucunu MATLAB'e gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

18 a - Kullanıcının aradığı parametrelere uygun sonuç bulunamaz ve kullanıcı bilgilendirilir

4.6 Senaryo 6

Adı: MATLAB Kullanılarak Sensör Adı Filtreli Değer Arama

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda MATLAB kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, MATLAB, Sunucu

Senaryo:

Kullanıcı:

1 - Kullanıcı MATLAB'i çalıştırır

2 - Kullanıcı bir değişkene framework atamasını yapar

4 - Kullanıcı gerekli olan fonksiyonu çağırır

5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer

13 - Kullanıcı sensör adı bilgisini girer

21 - Kullanıcı sonucu görür

22 - Kullanıcı programı kapatır

MATLAB:

3 - MATLAB frameworkü tanır

6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer

7 - Fonksiyonu çalıştırır

12 - Sunucu üzerinden gelen veri değişkene atanır

14 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer

15 - Fonksiyonu çalıştırır

20 - Sunucu üzerinden gelen sonuç değişkene atanır

Sunucu:

8 - MATLAB ile Java üzerinden gelen isteği alır

9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir

10 - Arama sonucuna ulaşır

11 - Arama sonucunu MATLAB'e gönderir

16 - MATLAB ile Java üzerinden gelen isteği alır

17 - Önceden çekilmiş olan veri tabanından alınan parametrelere göre bir arama gerçekleştirir

18 - Arama sonucuna ulaşır

19 - Arama sonucunu MATLAB'e gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

18 a - Kullanıcının aradığı parametrelere uygun sonuç bulunamaz ve kullanıcı bilgilendirilir

4.7 Senaryo 7

Adı: MATLAB Kullanılarak Tarih Filtreli Değer Arama

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda MATLAB kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, MATLAB, Sunucu

Senaryo:**Kullanıcı:**

- 1 - Kullanıcı MATLAB'i çalıştırır
- 2 - Kullanıcı bir değişkene framework atamasını yapar
- 4 - Kullanıcı gerekli olan fonksiyonu çağırır
- 5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer
- 13 - Kullanıcı tarih bilgisini girer
- 21 - Kullanıcı sonucu görür
- 22 - Kullanıcı programı kapatır

MATLAB:

- 3 - MATLAB frameworkü tanır
- 6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 7 - Fonksiyonu çalıştırır
- 12 - Sunucu üzerinden gelen veri değişkene atanır
- 14 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 15 - Fonksiyonu çalıştırır
- 20 - Sunucu üzerinden gelen sonuç değişkene atanır

Sunucu:

- 8 - MATLAB ile Java üzerinden gelen isteği alır
- 9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir
- 10 - Arama sonucuna ulaşır
- 11 - Arama sonucunu MATLAB'e gönderir
- 16 - MATLAB ile Java üzerinden gelen isteği alır
- 17 - Önceden çekilmiş olan veri tabanından alınan parametrelere göre bir arama

gerçekleştirir

18 - Arama sonucuna ulaşır

19 - Arama sonucunu MATLAB'e gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

18 a - Kullanıcının aradığı parametrelere uygun sonuç bulunamaz ve kullanıcı bilgilendirilir

4.8 Senaryo 8

Adı: Python Kullanılarak Cihaz IDsi ve Tarih Aralığı ile Değer Okuma

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda Python3.8 kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, Python, Sunucu

Senaryo:

Kullanıcı:

- 1 - Kullanıcı Python IDE'sini çalıştırır
- 2 - Kullanıcı bir değişkene framework atamasını yapar
- 4 - Kullanıcı gerekli olan fonksiyonu çağırır
- 5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer
- 13 - Kullanıcı veriyi görür
- 14 - Kullanıcı programı kapatır

Python:

- 3 - Python frameworkü tanır
- 6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 7 - Fonksiyonu çalıştırır

12 - Sunucu üzerinden gelen veri değişkene atanır

Sunucu:

8 - Python ile Java üzerinden gelen isteği alır

9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir

10 - Arama sonucuna ulaşır

11 - Arama sonucunu Python'a gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

4.9 Senaryo 9

Adı: Python Kullanılarak Sensör ve Tarih ile Spesifik Değer Arama

Ön Koşullar: Kullanıcının geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda Python 3.8 kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, Python, Sunucu

Senaryo:

Kullanıcı:

1 - Kullanıcı Python'ı çalıştırır

2 - Kullanıcı bir değişkene framework atamasını yapar

4 - Kullanıcı gerekli olan fonksiyonu çağırır

5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer

13 - Kullanıcı sensör adı ve tarih bilgisini girer

21 - Kullanıcı sonucu görür

22 - Kullanıcı programı kapatır

Python:

3 - Python frameworkü tanır

6 - Fonksiyona paremetre olarak kullanıcıdan aldığı verileri girer

7 - Fonksiyonu çalıştırır

12 - Sunucu üzerinden gelen veri değişkene atanır

14 - Fonksiyona paremetre olarak kullanıcıdan aldığı verileri girer

15 - Fonksiyonu çalıştırır

20 - Sunucu üzerinden gelen sonuç değişkene atanır

Sunucu:

8 - Python ile Java üzerinden gelen isteği alır

9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir

10 - Arama sonucuna ulaşır

11 - Arama sonucunu Python'a gönderir

16 - Python ile Java üzerinden gelen isteği alır

17 - Önceden çekilmiş olan veri tabanından alınan parametrelere göre bir arama gerçekleştirir

18 - Arama sonucuna ulaşır

19 - Arama sonucunu Python'a gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

18 a - Kullanıcının aradığı parametrelere uygun sonuç bulunamaz ve kullanıcı bilgilendirilir

4.10 Senaryo 10

Adı: Python Kullanılarak Sensör Adı Filtreli Değer Arama

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir.

Bilgisayarda Python 3.8 kurulu olmalıdır. Bilgisayar internete bağı olmalıdır.

Aktörler: Kullanıcı, Python, Sunucu

Senaryo:

Kullanıcı:

- 1 - Kullanıcı Python'ı çalıştırır
- 2 - Kullanıcı bir değişkene framework atamasını yapar
- 4 - Kullanıcı gerekli olan fonksiyonu çağırır
- 5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer
- 13 - Kullanıcı sensör adı bilgisini girer
- 21 - Kullanıcı sonucu görür
- 22 - Kullanıcı programı kapatır

Python:

- 3 - Python frameworkü tanır
- 6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 7 - Fonksiyonu çalıştırır
- 12 - Sunucu üzerinden gelen veri değişkene atanır
- 14 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer
- 15 - Fonksiyonu çalıştırır
- 20 - Sunucu üzerinden gelen sonuç değişkene atanır

Sunucu:

- 8 - Python ile Java üzerinden gelen isteği alır
- 9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir
- 10 - Arama sonucuna ulaşır
- 11 - Arama sonucunu Python'a gönderir

16 - Python ile Java üzerinden gelen isteği alır

17 - Önceden çekilmiş olan veri tabanından alınan parametrelere göre bir arama gerçekleştirir

18 - Arama sonucuna ulaşır

19 - Arama sonucunu Python'a gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

18 a - Kullanıcının aradığı parametrelere uygun sonuç bulunamaz ve kullanıcı bilgilendirilir

4.11 Senaryo 11

Adı: Python Kullanılarak Tarih Filtreli Değer Arama

Ön Koşullar: Kullanıcın geçerli bir cihaz idsi ile bir tarih aralığı bilgisi girmesi gerekir. Bilgisayarda Python 3.8 kurulu olmalıdır. Bilgisayar internete bağlı olmalıdır.

Aktörler: Kullanıcı, Python, Sunucu

Senaryo:

Kullanıcı:

1 - Kullanıcı Python'ı çalıştırır

2 - Kullanıcı bir değişkene framework atamasını yapar

4 - Kullanıcı gerekli olan fonksiyonu çağırır

5 - Kullanıcı cihaz id ve tarih aralığı bilgisini girer

13 - Kullanıcı tarih bilgisini girer

21 - Kullanıcı sonucu görür

22 - Kullanıcı programı kapatır

Python:

3 - Python frameworkü tanır

6 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer

7 - Fonksiyonu çalıştırır

12 - Sunucu üzerinden gelen veri değişkene atanır

14 - Fonksiyona parametre olarak kullanıcıdan aldığı verileri girer

15 - Fonksiyonu çalıştırır

20 - Sunucu üzerinden gelen sonuç değişkene atanır

Sunucu:

8 - Python ile Java üzerinden gelen isteği alır

9 - Veri tabanında alınan parametrelere göre bir arama gerçekleştirir

10 - Arama sonucuna ulaşır

11 - Arama sonucunu Python'a gönderir

16 - Python ile Java üzerinden gelen isteği alır

17 - Önceden çekilmiş olan veri tabanından alınan parametrelere göre bir arama gerçekleştirir

18 - Arama sonucuna ulaşır

19 - Arama sonucunu Python'a gönderir

Alternatif Yollar:

10 a - Bir sorun çıkar ve aranan sonuç bulunamaz ise kullanıcı bilgilendirilir

18 a - Kullanıcının aradığı parametrelere uygun sonuç bulunamaz ve kullanıcı bilgilendirilir

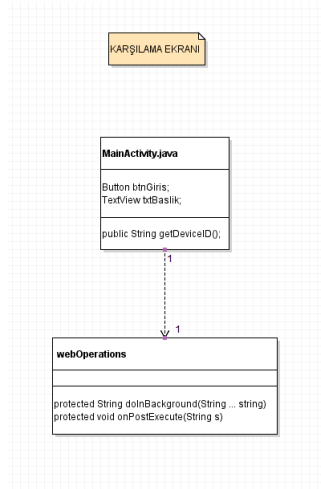
5

Sistem Tasarımı

Bu bölüm tasarlanan sistemi nesne yönelimli programlama metoduyla tanıtmaktadır. Yazılım tasarımı, veritabanı tasarımı ve girdi-çıkı tasarımı bölümleri altında detaylı olarak incelenecektir.

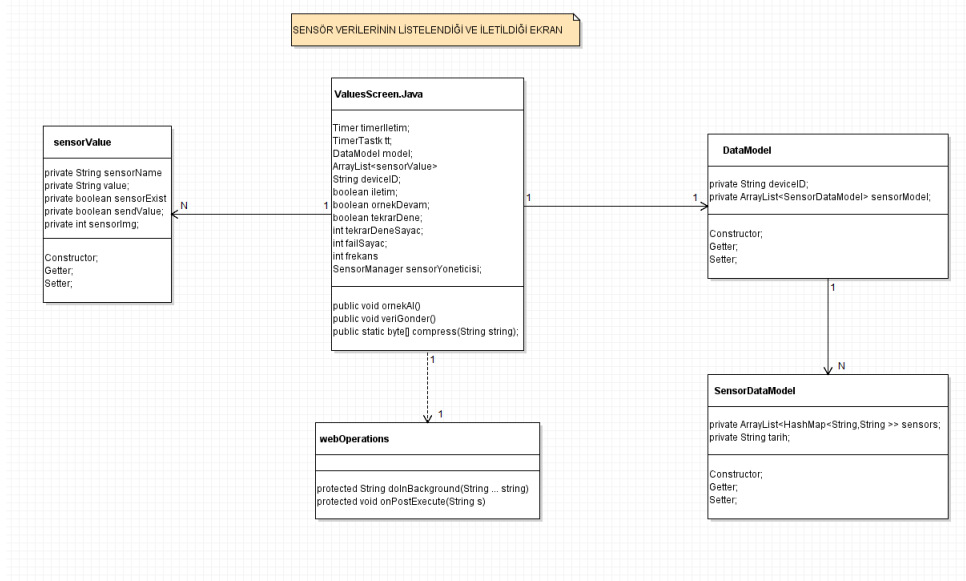
5.1 Yazılım Tasarımı

5.1.1 Android



Şekil 5.1 Ana Ekran Sınıf Diyagramı

Uygulamamız açıldığı anda kullanıcıya hoşgeldin ekranı gösterilecektir. Bu hoşgeldin ekranında kullanıcı "Uygulamaya başla" butonuna basacak ve sensör verilerinin listelendiği ekrana geçecektir. Bu sebeple ana ekran için `MainActivity` classı ve bu classın içerisinde inner class olarak bulunan ve arka planda çalışan `webOperations` classı da eklenmiştir.



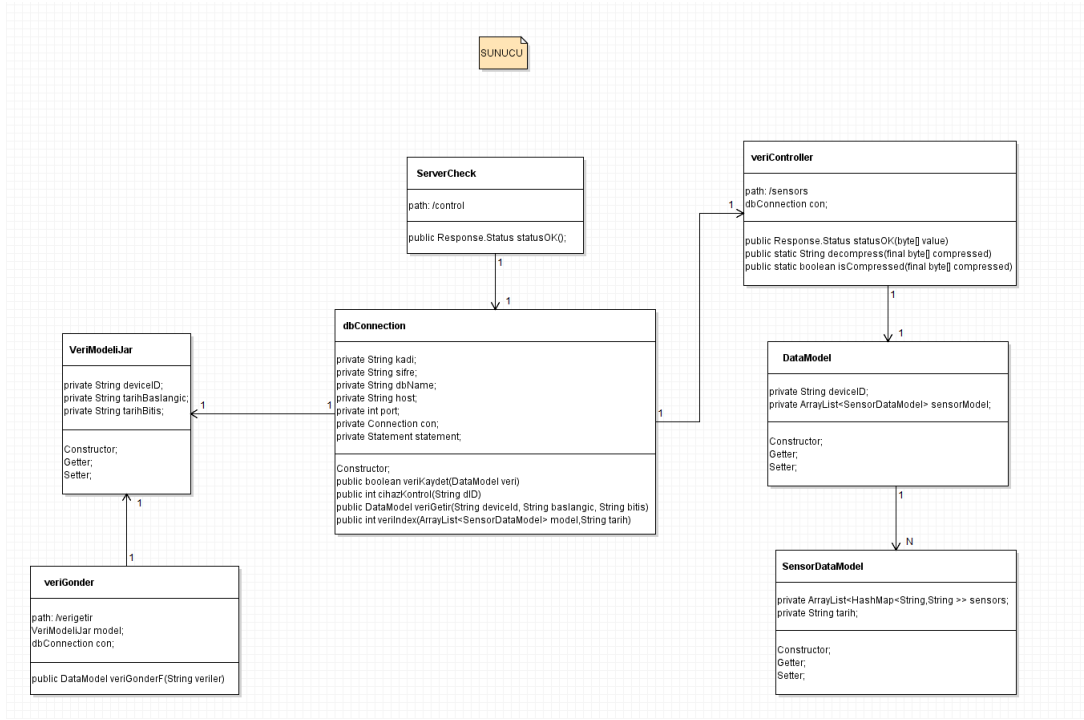
Şekil 5.2 Sensör Ekranı Sınıf Diyagramı

Uygulamamızın temel amacı sensör verileri üzerinde çalışmaktır. Bu sebeple sensör isimli sınıf oluşturulmuştur. Bu sınıf içerisinde sensörün adı, resmi, değeri, cihazda mevcut olup olmadığı ve verisinin sunucuya gönderilip gönderilmeyeceği bilgileri tutulacaktır.

Bu verilere ek olarak sisteme hangi cihazdan bağlanıldığı bilgisi tutulacaktır. Sensörlerden okunan verinin hangi zaman noktasında okunduğu da tutulacak bilgilerden bir diğeridir. Bu bilgileri sunucuya göndermek için bir istemci oluşturulmalıdır. Bu istemci **webOperations** nesnesi ile oluşturulmaktadır.

Verileri sıkıştırmak için Gzip kütüphanesi kullanılacaktır. Bu kütüphane dosya ya da veri sıkıştırmada huffman sıkıştırma metodunu kullanan bir kütüphanedir. Verileri byte diziye çevirir.

5.1.2 Sunucu

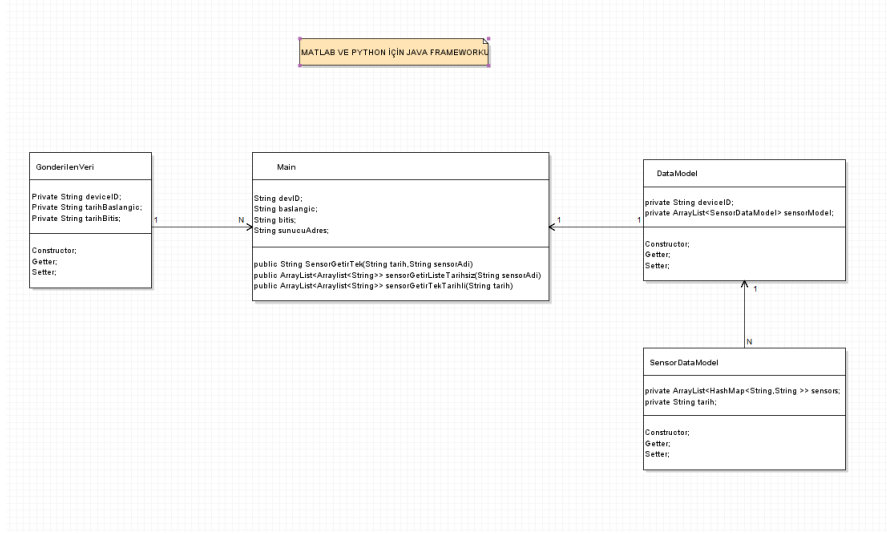


Şekil 5.3 Sunucu Sınıf Diyagramı

Veriler sunucuya gönderilir ve sunucu veriyi decompress fonksiyonu ile çözer. Bu fonksiyon gzip kütüphanesiyle sıkıştırılmış veriyi çözmek için tasarlanmıştır.

Veri içerisindeki deviceId alınır. Sensör bilgileri ise sensör bilgilerini tutan SensorDataModel isimli sınıfta saklanır. Bu veriler DataModel isimli sınıfta birleştirilir ve dbConnection sınıfından türetilen verinin fonksiyonları kullanılarak veriler veritabanına kaydedilir.

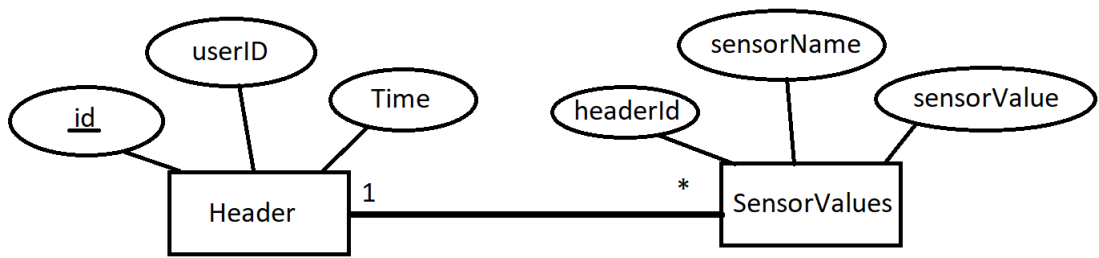
5.1.3 Matlab ve Python'da kullanılacak framework



Şekil 5.4 Matlab ve Python'da kullanılacak frameworkün Sınıf Diyagramı

Sunucuya kaydedilen verileri matlab ve python'da kullanabilecek hale getirmeyi sağlayan bir framework oluşturulmuştur. Main sınıfında bağlanacağımız sunucuya ait bilgi ve sunucudan beklenen veriler için, hangi cihazın hangi tarih aralığındaki verileri, parametreleri alınır. GönderilenVeri sınıfından bir nesne türetilir ve sunucuya istek atılır. Gelen veri parse edilerek DataModel sınıfından türetilen nesneye atılır.

5.2 Veritabanı Tasarımı



Şekil 5.5 ER Diyagramı

İki adet tablo bizim için yeterli olacaktır. Header isimli ilk tablo istemciden gelen verilerdeki userID ve zaman noktasını tutacaktır. Bu bilgilere ulaşabilmek için benzersiz ve otomatik artan id değeri kullanılacaktır.

SensorValues isimli ikinci tabloda ise gelen her sensör verisi ile birlikte kaydedilecektir. Bu verilerin hangi kişiden ve hangi zaman diliminden geldiğini anlayabilmemiz için headerId alanı bizim için yeterli olacaktır.

Bu sayede istediğimiz kişinin istediğimiz zaman dilimine ait sensör bilgilerine rahatlıkla ulaşabileceğiz.

5.3 Girdi-Çıktı Tasarımı



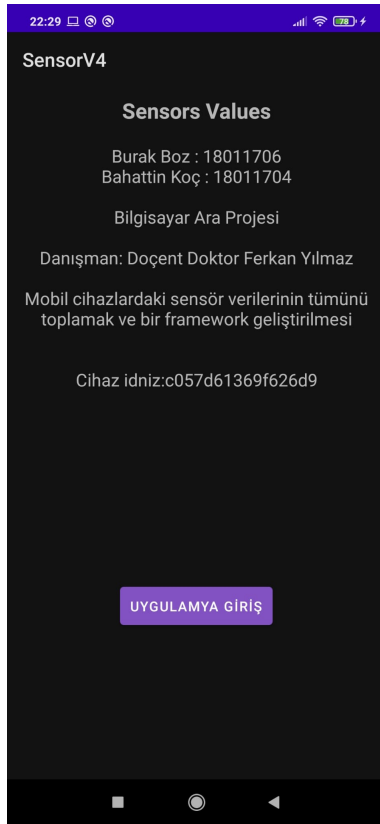
Şekil 5.6 Kullanıcı Arayüzü

Program başlatıldığında zaman kullanıcıya, öncelikle bir karşılama ekranı gösterilecektir. Bu karşılama ekranında bazı bilgiler verilecek ve bir buton yardımı ile şekilde gösterilen ekrana geçiş yapabilecektir. Ekranın sağ üst kısmında hangi sıklıkla veri iletimi yapılacağı seçilmektedir. Varsayılan olarak saniyede bir kere veri alınmaktadır. Kullanıcı veri iletimi için 15 saniyede bir ile başlayan ve 1 saatte bir kere kadar uzanan aralıkta seçim yapabilmektedir. Kullanıcı isterse bu değeri değiştirebilir.

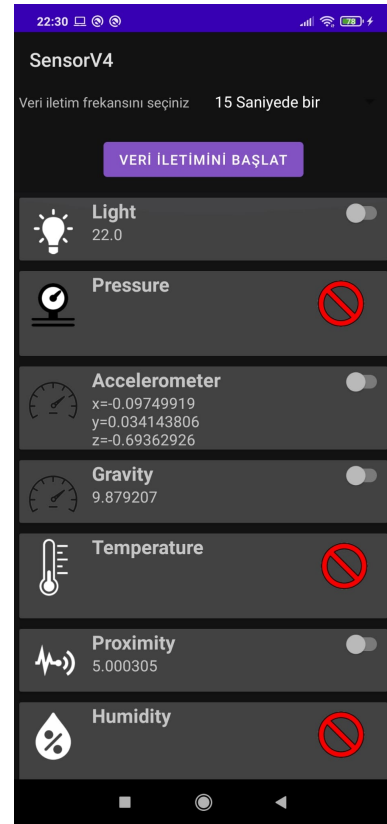
Ekranın listenin her bir kısmı bir sensör için ayrılmıştır. Mevcut olmayan sensörlerin yanında blok ikonu gösterilmektedir. Mevcut olan sensörlerin yanında ise switch nesnesi bulunmaktadır. Kullanıcı bu ekranda sensörlerden alınan değerleri görebilir. Eğer kullanıcı switchleri aktif hale getirirse ve butona basarsa bu sensörlerin verisi sunucuya iletmeye başlayacaktır.

Problem olmadığı sürece uyarı verilmez fakat veri iletimi kesilirse kullanıcıya uyarı verilecektir.

6.1 Android



Şekil 6.1 Karşılama ekranı

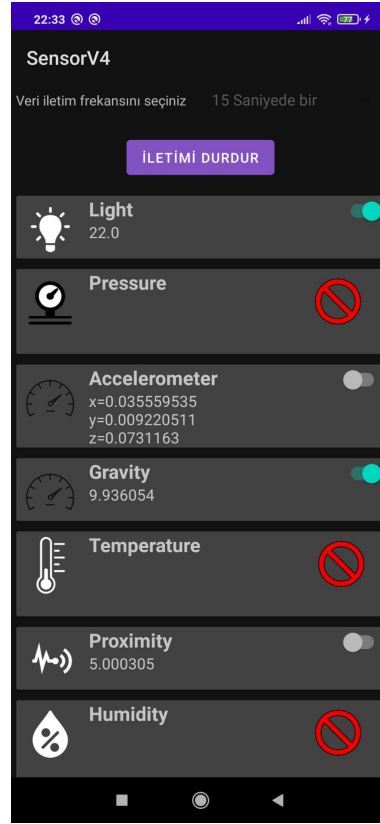


Şekil 6.2 Kullanıcı arayüzü 2

Kullanıcı program açılınca önce şekil 6.1'deki karşılama ekranı ile karşılaşılır ve uygulamaya giriş butonuna tıklayınca sensör verilerinin listelendiği ekran ile karşılaşılır. Listelemek için recyclerview[5] kullanılmıştır. Burada listelenen sensör verilerine ulaşabilmek için androidin sunmuş olduğu Sensor[1] kütüphanesinden faydalanılmıştır.

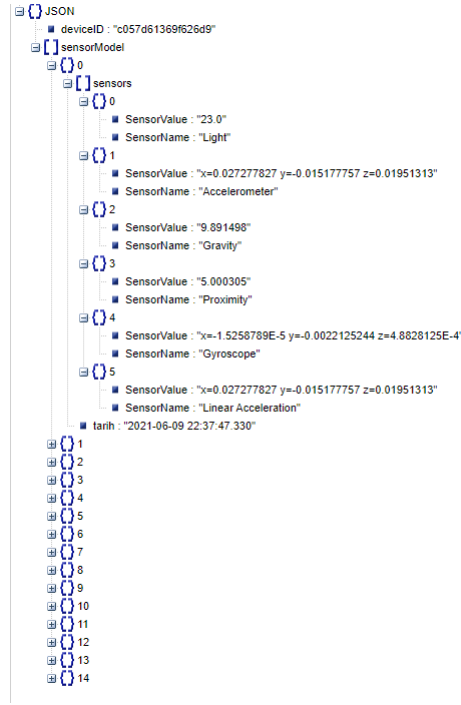
Butona basıldığı anda seçili sensörler için her saniyede bir kere sensör değeri okunmaktadır. Sağ üst kısımda bulunan combobox yardımı ile ne sıklıkla veri

iletileceđi seřimini yapılır. Bu deđer varsayılan olarak 15 saniyede bir adet ile
ilklendirilmiřtir.



řekil 6.3 Verilerin ilettime geřmesi

Kullanıcı iletmek istediđi verilerin switchlerini řekil 6.3'daki gibi aktif hale getirir ve
Veri iletimini bařlat butonuna tıklar.



Şekil 6.4 Cihazdan toplanan sensör verilerinin json hali

Gönderilecek veriler işaretlendikten sonra mobil cihazda oluşan verilerin json hali şekil 6.4'deki gibidir. Bu veriler sunucuya ziplenerek gönderilecektir.

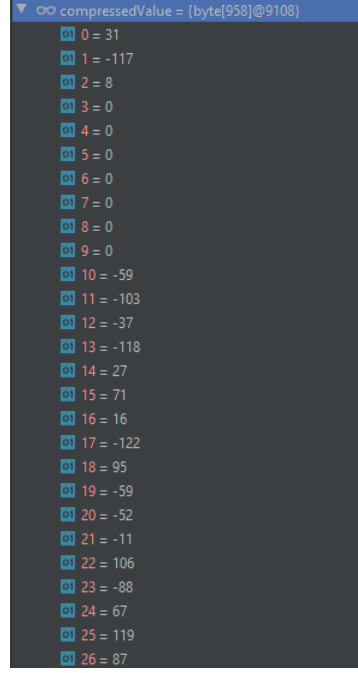
```

var veri = '{"deviceID":"c057d61369f626d9","sensorModel":[{"sensors":[{"SensorValue":"23.0","SensorName":"Light"}, {"SensorValue":"x=0.027277827 y=-0.015177757 z=0.01951313","SensorName":"Accelerometer"}, {"SensorValue":"9.891498","SensorName":"Gravity"}, {"SensorValue":"5.000305","SensorName":"Proximity"}, {"SensorValue":"x=-1.5258789E-5 y=-0.0022125244 z=4.8828125E-4","SensorName":"Gyroscope"}, {"SensorValue":"x=0.027277827 y=-0.015177757 z=0.01951313","SensorName":"Linear Acceleration"}]}]}'
var veri.length() = 7850
var compressedValue = [byte[958]]@9108
var compressedValue.length = 958

```

Şekil 6.5 Verilerin Boyutları

Gzip kütüphanesi ile sıkıştırılan veri compress fonksiyonundan geçirildikten sonra byte diziye dönüştürülmüştür. Bu veri compressedValue isimli değişkende görünmektedir. Veri isimli değişkenin uzunluğu 7850 karakter iken sıkıştırılmış hali 958 karakterden oluşmaktadır.



Şekil 6.6 Sıkıştırılan Veri

Sıkıştırılan verinin içeriği şekil 6.6 deki gibidir. Bu sıkıştırılan veriye gerekli headerlar eklendikten sonra sunucuya gönderilir.

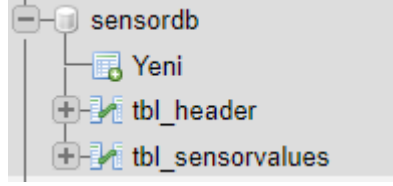
6.2 Sunucu

Name	Value
▼ X? "value"	(id=124)
> [0...99]	
> [100...199]	
> [200...299]	
> [300...399]	
> [400...499]	
> [500...599]	
> [600...699]	
> [700...799]	
> [800...899]	
> [900...974]	
X? "value.length"	975
> X? "cozulenVeri"	{"deviceId":"c057d61369f626d9","sensorModel":{"sensors":[{"SensorValue":"22.0","SensorName":"Light"}, {"SensorValue..."
X? "cozulenVeri.length()"	7861
✚ Add new expression	

Şekil 6.7 Sunucuda Gelen Veriler

Sunucuya gelen veri şekil 6.7 deki gibi byte array olarak gelir. Bu array decompress fonksiyonundan geçirilir ve json string haline döner. Ardından DataModel isimli sınıfa parse edilir ve veritabanına kaydedilir.

6.3 Veritabanı



Şekil 6.8 Veritabanı Tabloları

Veriler veritabanında 2 tablo halinde tutulmaktadır.

id	deviceId	dateTime
74	c057d61369f626d9	2021-06-08 00:23:08
75	c057d61369f626d9	2021-06-08 00:23:09
76	c057d61369f626d9	2021-06-08 00:23:10
77	c057d61369f626d9	2021-06-08 00:23:11
78	c057d61369f626d9	2021-06-08 00:23:57
79	c057d61369f626d9	2021-06-08 00:23:58
80	c057d61369f626d9	2021-06-08 00:23:59
81	c057d61369f626d9	2021-06-08 00:24:00
82	c057d61369f626d9	2021-06-08 00:24:01

Şekil 6.9 Header Tablosu

Header tablosu içerisinde gelen her bir tarih noktası için yeni bir veri oluşturulur. Bu veri device id si ile birlikte kaydedilir ve bir anahtar ile belirlenir.

headerID	sensorName	sensorValue
74	Işık Sensörü	22.0
74	Yerçekimi Sensörü	9.900194
75	Işık Sensörü	22.0
75	Yerçekimi Sensörü	9.893025
76	Işık Sensörü	22.0
76	Yerçekimi Sensörü	9.90144
77	Işık Sensörü	18.0
77	Yerçekimi Sensörü	9.904132
78	Işık Sensörü	23.0

Şekil 6.10 Sensor values Tablosu

Belirlenen id ile tarih noktasındaki sensör verileri şekil 6.10'deki gibi tbl-sensorvalues tablosuna kaydedilir. Bu sayede belirlenen cihazın istenilen tarih aralığındaki verilerine ulaşılabilir.

6.4 Matlab

```
>> nesne = com.sensors.jar.Main
Kullanım Klavuzu
Verilere ulaşabilmek için öncelikle getDataModel() fonksiyonunu çalıştırınız.
Kullanılabilecek fonksiyonlar:
getDataModel('Cihaz id (String)', 'başlangıç tarihi (String YYYY-AA-GG SS:DD:ss)', 'bitiş tarihi (String YYYY-AA-GG SS:DD:ss)', 'sunucu adresi')
Bu fonksiyon belli bir cihaz için belirtilen tarih aralığındaki tüm sensör verilerini getirir

getOneSensor((String tarih = YYYY-AA-GG SS:DD:ss), (String sensorAdi))
Belirtilen tarihteki belirtilen sensörün değerini döndürür. String döndürür.

getSensorsWithoutDate(String sensorAdi)
Belirtilen isimdeki sensörün tüm tarihlerdeki değerlerini getirir. ArrayList<ArrayList<String>> döndürür.

getSensorsWithSpecificDate(String tarih = YYYY-AA-GG SS:DD:ss)
Bir tarih noktasındaki tüm sensörleri ve değerlerini döndürür. ArrayList<ArrayList<String>> döndürür.

nesne =

com.sensors.jar.Main@5b619d14
```

Şekil 6.11 Matlabdan Frameworke Ulaşılması

Framework'ü jar dosyası haline getirdikten sonra MATLAB'ın lokal dosyalarında bulunan classpath.txt dosyasına jar dosyamızın yolu eklenmelidir. Bu işlemi yapıp MATLAB'ı yeniden başlattıktan sonra şekil 6.11'deki komutu girerek MATLAB'da yeni bir java nesnesi oluşturulmuştur. Bu nesnenin yapıcı metodunda fonksiyonların kullanım klavuzları tanımlanmıştır.

```
>> dataModel = nesne.getDataModel("c057d61369f626d5", "2021-06-08 00:24:20", "2021-06-08 20:32:59", "http://192.168.1.6:8080/sensorv4/webapi/")

dataModel =
com.sensors.jar.DataModel@3016fd5e
```

Şekil 6.12 Dataların Alınması

Oluşturulan nesne içerisindeki `getDataModel()` fonksiyonu ile veriler şekil 6.12'deki gibi sunucudan alınır.

3 farklı şekilde verilere ulaşılabilmektedir. Spesifik bir sensörün spesifik bir tarih noktasındaki verilerine `getOneSensor()` fonksiyonu ile ulaşılmaktadır.

Veri grubundaki bir sensöre ait tüm verileri almak için `getSensorsWithoutDate()` fonksiyonu kullanılabilir.

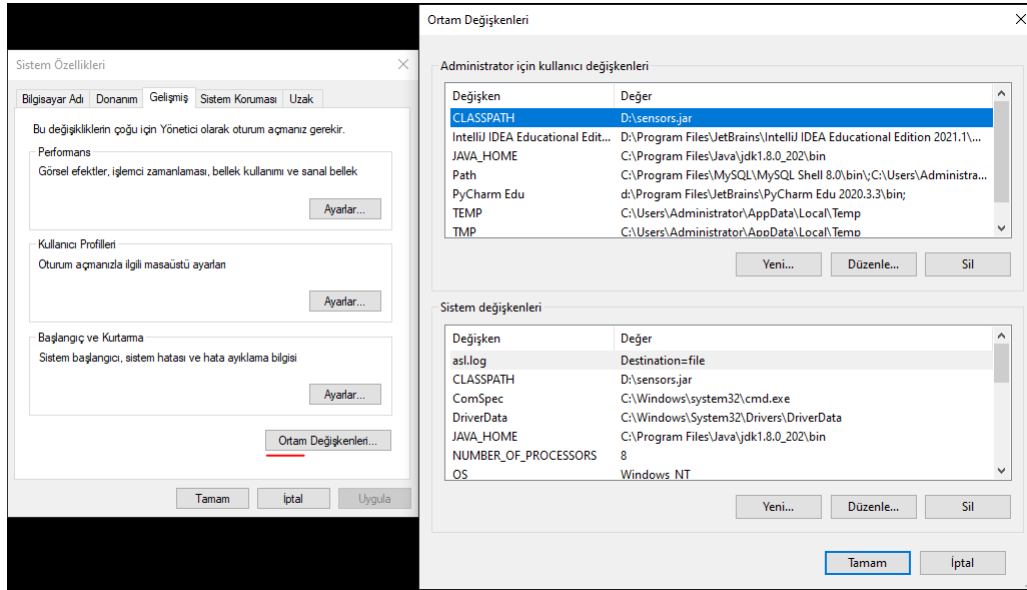
Bir diğer yöntem ise belli bir tarih noktasındaki tüm sensör verilerine ulaşmaktır. Bunun için `getSensorsWithSpecificDate()` fonksiyonu kullanılabilir.

Örneğin ışık sensörüne ait tüm verileri görmek için şekil 6.13'deki komut işlenebilir.

```
>> isikSensorleri = nesne.getSensorsWithoutDate(dataModel,"Işık Sensörü")  
  
isikSensorleri =  
  
[[[2021-06-08 19:49:14.0, 204.0], [2021-06-08 19:49:14.0, 204.0], [2021-06-08 19:49:14.0, 204.0], [2021-06-08 19:49:14.0, 204.0], [2021-06-08 19:49:14.0, 204.0]]]
```

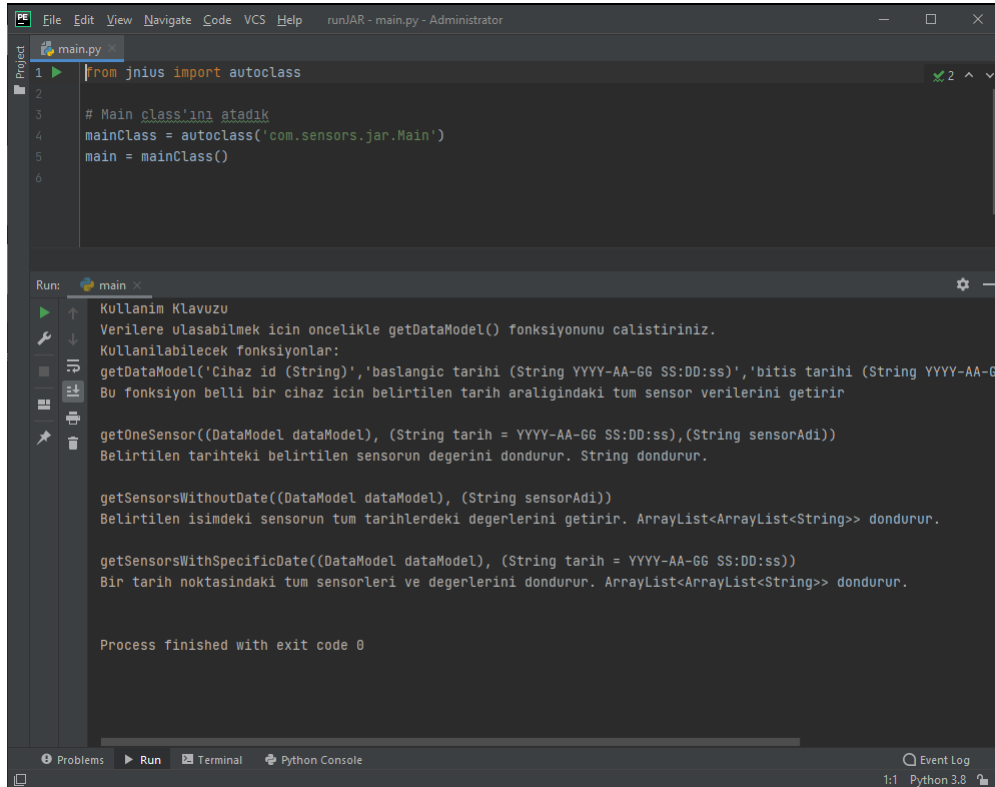
Şekil 6.13 Işık Sensörlerine Ait Tüm Veriler

6.5 Python



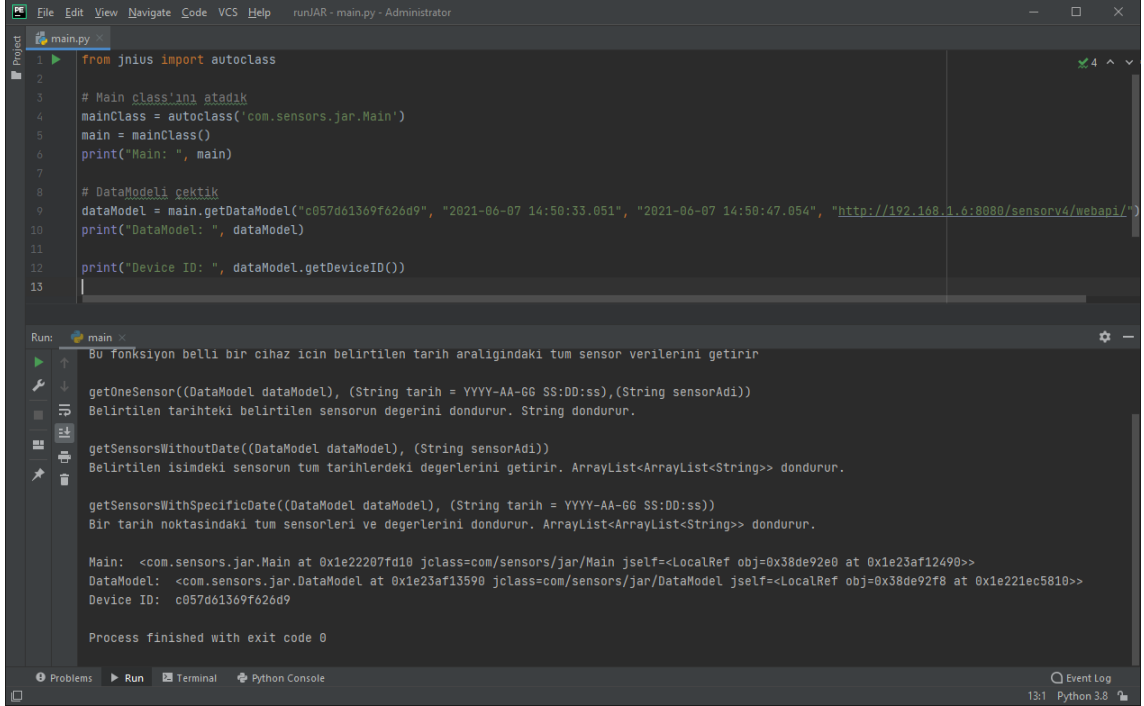
Şekil 6.14 Framework'ün Ortam Değişkeni Olarak Eklenmesi

Oluşturmuş olduğumuz frameworkün Python tarafından tanınabilmesi için dosyamızın bilgisayarımızda classpath'e eklenmesi gerekmektedir. Bunun için şekil 6.14'teki eklemeleri yapmamız gerekmektedir. Bu işlem yapıldıktan sonra kullanılan IDE yeniden başlatılmalıdır.



Şekil 6.15 Python'dan Framework'e Ulaşılması

Python'da Java dosyalarının çalışabilmesi için PYJNIUS [6] kütüphanesi kullanılmıştır. PYJNIUS sayesinde tıpkı MATLAB'te olduğu gibi frameworkümüzü bir değişkene atayıp Java fonksiyonlarımızı çalıştırabilmekteyiz. Bunun için ister IDE aracılığı ile istersek manuel olarak PYJNIUS kütüphanesini projemize import etmemiz gerekmektedir.



```
1 from jnius import autoclass
2
3 # Main class'ını atadık
4 mainClass = autoclass('com.sensors.jar.Main')
5 main = mainClass()
6 print("Main: ", main)
7
8 # DataModel'i gettik
9 dataModel = main.getDataModel("c057d61369f626d9", "2021-06-07 14:50:33.051", "2021-06-07 14:50:47.054", "http://192.168.1.6:8080/sensorv4/webapi/")
10 print("DataModel: ", dataModel)
11
12 print("Device ID: ", dataModel.getDeviceID())
13
```

```
Run: main
Bu fonksiyon belli bir cihaz için belirtilen tarih aralığındaki tüm sensor verilerini getirir
getOneSensor((DataModel dataModel), (String tarih = YYYY-AA-GG SS:DD:ss),(String sensorAdi))
Belirtilen tarihteki belirtilen sensorun degerini dondurur. String dondurur.
getSensorsWithoutDate((DataModel dataModel), (String sensorAdi))
Belirtilen isimdeki sensorun tüm tarihlerdeki degerlerini getirir. ArrayList<ArrayList<String>> dondurur.
getSensorsWithSpecificDate((DataModel dataModel), (String tarih = YYYY-AA-GG SS:DD:ss))
Bir tarih noktasındaki tüm sensorleri ve degerlerini dondurur. ArrayList<ArrayList<String>> dondurur.
Main: <com.sensors.jar.Main at 0x1e22207fd10 jclass=com/sensors/jar/Main jself=<LocalRef obj=0x38de92e0 at 0x1e23af12490>>
DataModel: <com.sensors.jar.DataModel at 0x1e23af13590 jclass=com/sensors/jar/DataModel jself=<LocalRef obj=0x38de92f8 at 0x1e221ec5810>>
Device ID: c057d61369f626d9
Process finished with exit code 0
```

Şekil 6.16 Dataların Alınması

Oluşturulan nesne içerisindeki getDataModel() fonksiyonu ile veriler şekil 6.16'deki gibi sunucudan alınır.

3 farklı şekilde verilere ulaşılabilmektedir. Spesifik bir sensörün spesifik bir tarih noktasındaki verilerine getOneSensor() fonksiyonu ile ulaşılmaktadır.

Veri grubundaki bir sensöre ait tüm verileri almak için getSensorsWithoutDate() fonksiyonu kullanılabilir.

Bir diğer yöntem ise belli bir tarih noktasındaki tüm sensör verilerine ulaşmaktır. Bunun için getSensorsWithSpecificDate() fonksiyonu kullanılabilir.

Örneğin yerçekimi sensörüne ait tüm verileri görmek için şekil 6.17'deki komut işlenebilir.

7

Deneyisel Sonuçlar

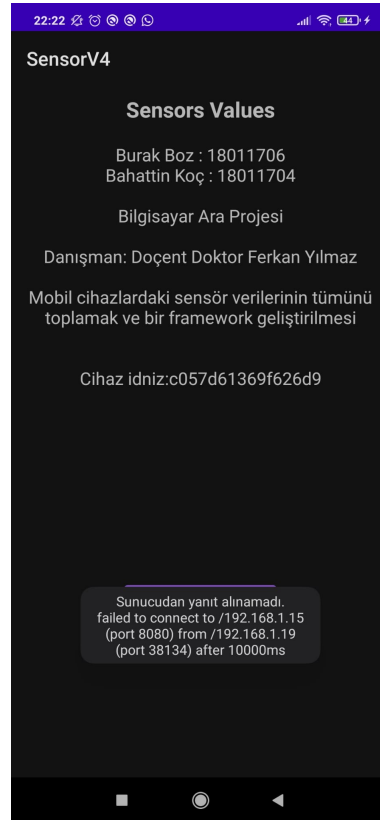
Bu bölümde uç durum testleri yapılacaktır.

7.1 Suncunun Kapalı Olma Durumu

Sununun bir şekilde kapanması ve ulaşılamaması durumunda farklı platformlarda oluşan sonuçlar:

7.1.1 Android

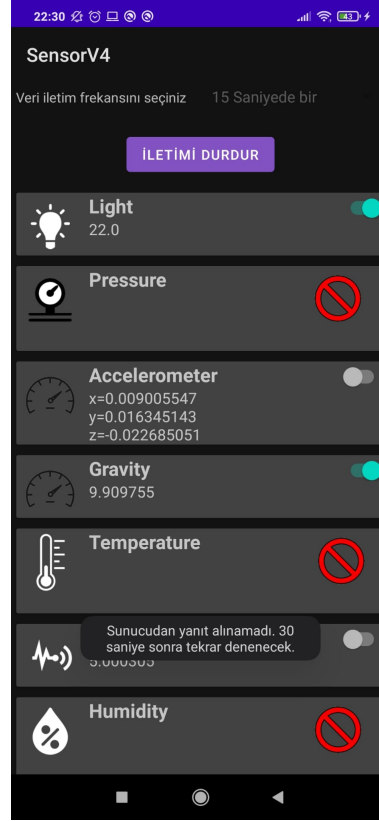
7.1.1.1 Karşılama Ekranı



Şekil 7.1 Karşılama Ekranında Sunucuya Ulaşılamama Durumu

Kullanıcı karşılama ekranındayken "Uygulamaya Giriş" butonuna basar ve sunucuya kontrol isteği gönderilir. Eğer 10 saniye içinde sunucudan olumlu cevap dönmezse sensör verilerinin bulunduğu ekrana geçilmez ve hata kodu kullanıcıya bildirilir.

7.1.2 Sensörler Ekranı



Şekil 7.2 Sensör Ekranında Sunucuya Ulaşılamama Durumu

Veril iletimine başlandıktan sonra kullanıcının belirlediği frekansta sunucuya veri iletilir. Eğer bu isteklere 10 saniye içerisinde olumlu dönüş olmazsa kullanıcıya 30 saniye sonra tekrar deneneceği bilgisi verilir. Bu süre zarfında veri toplama işlemi devam eder. Süre tamamlandığında tekrar istek gönderilir. Olumlu cevap gelirse bellekteki veriler sunucuya iletilir ve bellekten temizlenir. Veri iletim işlemi devam eder. Sunucudan olumsuz cevap gelme durumu üst üste dört kez tekrarlanırsa veri iletimi durdurulur ve kullanıcıya bildirilir. Veriler bellekten temizlenmez. Kullanıcı iletimi tekrar başlatığında yeni veriler eski verilerin üzerine eklenmeye devam eder.

Belleği doldurmamak adına 7200 veri toplanırsa (7200 veri iki saatlik veriyi temsil etmektedir. Saniyede bir veri alınır. 60x60x2) veriler kuyruk metoduyla ilk sıradan silinmeye başlanır. Bu sayede maksimum veri sayısı korunur.

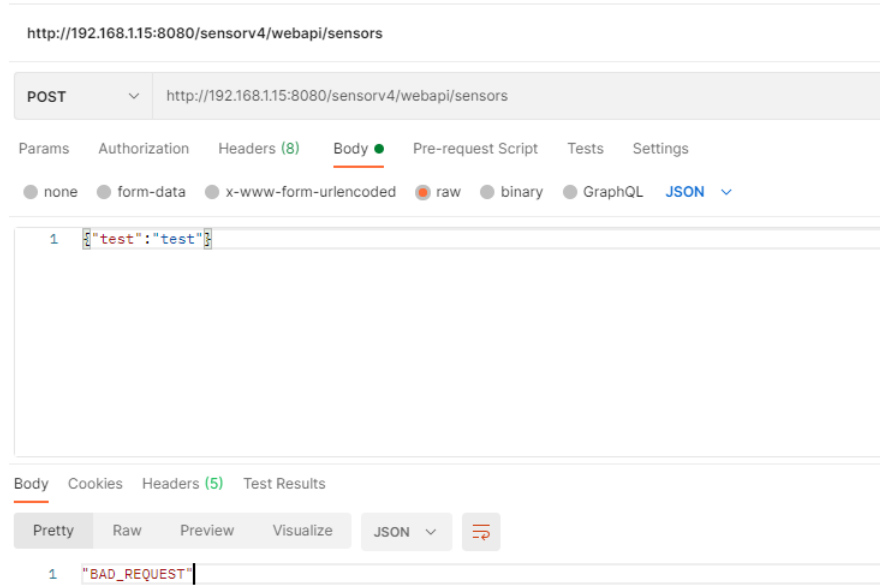
7.1.3 Programın Arkaplane Alınması

Android cihazlarda home tuşuna basarak program arkaplane alınmaktadır. Sensör verilerinin toplanması ve sunucuya iletilmesi timer ile yapıldığı için program arkaplane çalışmaya devam eder. Timerlar [7] android cihazlarda arkaplane threadinde çalışmaktadır.

7.2 Sunucu

Sunucuda 3 adet istek karşılayan fonksiyon bulunmaktadır. Bu fonksiyonlardan ilki sunucunun aktif olup olmadığını kontrol eden "/control" isimli fonksiyondur. Bu adrese get isteği atılır ve sunucu 200 döndürür.

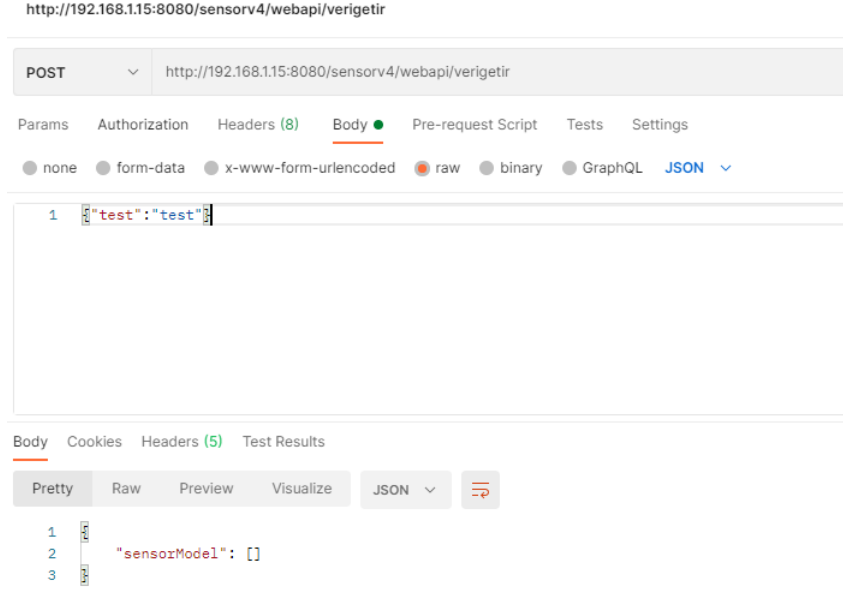
Bir diğer istek verilerin kaydedilmesini sağlayan "/sensors" adresinin tetiklenmesiyle çalışan fonksiyondur. Eğer veriler beklenen şekilde ziplenmiş ve byte array olarak gönderildiyse fonksiyon verileri çözümler ve sunucuya kaydeder.



Şekil 7.3 /sensors İsteğine Uyumsuz Veri Gelmesi Durumu

Sunucuya ulaşan veri beklenen şekilde gelmezse status olarak şekil 7.3'deki gibi "Bad Request" döndürülür.

Son istek ise MATLAB ve pythondan verilere ulaşılmasını sağlayan ve "/verigetir" adresinde çalışan fonksiyondur. Bu fonksiyon cihaz idsini, başlangıç tarihini ve bitiş tarihini içeren bir json metni bekler.



Şekil 7.4 /verigetir İsteğine Uyumsuz Veri Gelmesi Durumu

Bu fonksiyona ulaşan veri beklenen şekilde değilse veri modeli şekil 7.4'deki gibi boş döndürülür. Bu sonuç framework içerisinde işlenir ve kullanıcıya bildirilir.

7.3 Framework

7.3.1 MATLAB

```
>> dataModel = nesne.getDataModel("c057d61369f626d9222","2021-06-08 00:23:08","2021-06-08 20:53:49","http://192.168.1.15:8080/sensorv4/webapi/")
dataModel =
com.sensors.jar.DataModel@3b00856b
>> dataModel = nesne.getDataModel("", "2021-06-08 00:23:08","2021-06-08 20:53:49","http://192.168.1.15:8080/sensorv4/webapi/")
Kayıt Bulunamadı!
dataModel =
[]
```

Şekil 7.5 getDataModel() Fonksiyonunun MATLAB'da Sınanması

MATLAB'da nesne türetildikten sonra getDataModel() fonksiyonuna şekil 7.5'deki gibi cihaz idsi girilmemiştir. Kullanıcıya "Kayıt Bulunamadı" ibaresi gösterilmiştir.

7.3.2 Python

```
1 import jnius_config
2 from jnius import autoclass, JavaException, PythonJavaClass, JavaClass, MetaJavaClass, java_method
3 import os
4
5 # os.environ['CLASSPATH'] = "D://sensors.jar"
6 # jnius_config.set_classpath('D://sensors.jar')
7
8 # Main class'ını atadık
9 mainClass = autoclass('com.sensors.jar.Main')
10 main = mainClass()
11
12 # DataModel'i çektik
13 dataModel = main.getDataModel("c057d01369f626d92", "2021-06-08 00:23:08", "2021-06-08 20:53:49", "http://192.168.1.15:8880/sensorv4/webapi/")
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

main

Kullanım Klavuzu

Verilere ulaşabilmek için öncelikle getDataModel() fonksiyonunu çalıştırınız.

Kullanılabilir fonksiyonlar:

getDataModel('Cihaz id (String)', 'başlangıç tarihi (String YYYY-AA-GG SS:DD.0)', 'bitis tarihi (String YYYY-AA-GG SS:DD.0)', 'sunucu adresi')

Bu fonksiyon belli bir cihaz için belirtilen tarih aralığındaki tüm sensor verilerini getirir

getOneSensor((DataModel dataModel), (String tarih = YYYY-AA-GG SS:DD.0), (String sensorAdi))

Belirtilen tarihteki belirtilen sensorun değerini döndürür. String döndürür.

getSensorsWithoutDate((DataModel dataModel), (String sensorAdi))

Belirtilen isindeki sensorun tüm tarihlerdeki değerlerini getirir. ArrayList<ArrayList<String>> döndürür.

getSensorsWithSpecificDate((DataModel dataModel), (String tarih = YYYY-AA-GG SS:DD.0))

Bir tarih noktasındaki tüm sensorleri ve değerlerini döndürür. ArrayList<ArrayList<String>> döndürür.

Kayıt Bulunamadı!

Şekil 7.6 getDataModel() Fonksiyonunun Python'da Sınanması

Python'da nesne türetildikten sonra getDataModel() fonksiyonuna şekil 7.6'deki gibi cihaz idsi yanlış girilmiştir. Kullanıcıya "Kayıt Bulunamadı" ibaresi gösterilmiştir.

8

Performans Analizi

Performans analizleri platformlara göre ayrı ayrı değerlendirilecektir.

8.1 Android

- Sistemin düzenli çalışabilmesi için cihazın düzenli bir internet bağlantısına sahip olması gerekmektedir. Atılan isteklerin timeout durumuna düşme süreleri 10 saniye olarak belirlenmiştir. Eğer cihaz uygun bir internet bağlantısına sahip değilse ve 10 saniye içerisinde olumlu dönüş alamazsa sensör verileri sürekli toplanacak fakat 4 başarısız iletimden sonra veri iletimi durdurulacaktır.
- Sensör verileri ondalık sayı şeklinde gözükmemektedir fakat androidin sunmuş olduğu veriler metinsel ifadelerdir. Verileri işlemek isteyen kişiler bu verileri uygun formata çevirerek işlemelidirler.
- Tüm sensör verileri aynı kalıpta gelmemektedir. Örneğin ışık sensörü tek bir veri döndürürken jiroskop sensörü üç eksen için de ayrı ayrı veri döndürmektedir. Bu veriler tek bir metinsel değişken içerisinde yer almaktadır. Veriler "x=0.00/ny=0.00/nz=0.00" şeklinde tutulmaktadır. Bu veriyi işlemek isteyen kişi veriyi kendisine göre parse etmelidir.
- Cihazda üretilen device idsi, eğer cihaz fabrika ayarlarına döndürülürse değişmektedir. Bunun takibi yapılmamaktadır.

8.2 Sunucu

- Bir çok cihazdan aynı anda veri alınabilmektedir fakat bu sınırın ne kadar büyük olacağı sunucunun kurulduğu makinenin kapasitesi ile doğru orantılıdır.
- Sunucu için kullanılan jersey 2.31 kütüphanesi bazı tomcat versiyonlarında problem çıkarabilmektedir. Bu projede en uyumlu versiyonlar birlikte seçilmiştir.

Tomcat 9.0, Jersey 2.31, Mysql connector 5.1.39 ve javax servlet 2.5 versiyonu uyum içerisinde çalışmaktadır.

8.3 Framework

- Frameworkten bir nesne türetildiği anda kullanıcıyı bilgilendiren kullanım klavuzu ekranda gösterilmektedir.
- Frameworkte verilere ulaşabilmek için üç farklı yöntem bulunmaktadır. Hazırlanan framework MATLAB ve python da çalışmaktadır fakat temel olarak java ile hazırlanmıştır. Kullanıcının bu verilere düzenli şekilde ulaşabilmesi için temel düzeyde java bilgisine sahip olması gerekmektedir.
- Kullanıcı verilere ulaşabilmek için verilerini istediği cihazın device idsi bilgisine sahip olmalıdır.

8.4 MATLAB ve Python

- Üretilen frameworkün MATLAB classpathine elle eklenmesi gerekmektedir. Kullanıcı bu düzenlemeyi yapmazsa frameworke ulaşamaz. Aynı prosedür python için de geçerlidir.
- Çalışma alanına eklenen değerler java class formatındadır. Dolayısıyla bu alanda verilere doğrudan ulaşamaz. Fonksiyonlar kullanılarak verilere ulaşılmalıdır.

9 Sonuç

Analog dünyanın dijitalleştirilerek verilerin işlenmesinde sensörler en önemli paydaşlardan biridir. Sensörler sayesinde bir çok icat ve buluş hayatımıza girmiştir. Bu durum sensör verilerinin toplanmasının ve işlenmeye hazır hale getirilmesinin önemini bize anlatmaktadır. Günümüzde hemen herkeste bulunan akıllı cep telefonlarında birçok sensör bulunmaktadır. Projemiz bu sensör verilerinin toplanması, internet ortamına ulaştırılarak kayıt altına alınması ve geliştirilen bir JAVA frameworkü ile MATLAB ve python dillerinden de bu verilere ulaşılabilmesini amaçlamaktadır.

Projenin ileride bir çok kullanıcı tarafından kullanılabileceği hesaplanmıştır. Sürekli veri iletimi gerçekleşecektir. Bu sebeple internet bant genişliklerini yormamak adına verilerin sıkıştırılarak iletilmesi sağlanmıştır. Kullanıcının iletilmesini istediği veriler önce bir java classında toplanmış, gson kütüphanesi ile json metni haline dönüştürülmüştür. Bu metinsel ifade huffman algoritması temelli çalışan gzip kütüphanesi kullanılarak sıkıştırılmıştır. Yapmış olduğumuz gözlemler sonucu verilerin boyutları üzerinde büyük kazanç sağlanmıştır.

Veriler sunucuya sıkıştırılmış bir byte dizisi şeklinde gelir. Sunucu bu veriyi açar ve uygun java classı formatına çevirir. Ardından mysql veritabanına mysql-connector kullanarak ekler. Veritabanında iki adet tablo mevcuttur. İlk tablo cihaz idsini ve verinin ulaştığı tarihi tutmaktadır ve bu bilgi benzersiz bir id ile etiketlenir. İkinci tabloda sensör verileri yer almaktadır ve bu veriler ilk tablodaki id etiketini kullanmaktadır. Bu sayede tablolarda veri tekrarının bir nebze önüne geçilmiştir.

Toplanan ve kaydedilen bu verilerin MATLAB ve pythonda kullanılabilmelerini sağlayan bir java frameworkü geliştirilmiştir. Bu frameworkte androidde ve sunucuda kullanılan veri yapısı aynı şekilde yer almaktadır. Framework ile sunucudaki verilere ulaşabilmek için bir nesne oluşturulmalıdır. Bu nesnenin yapıcı metodunda frameworkün kullanımına ait bilgiler yer almaktadır. Dolayısıyla nesne üretildiği anda kullanıcı bilgilendirilir. Frameworkün düzenli çalışabilmesi için öncelikle verilerin

sunucudan alınması gerekmektedir. İlgili fonksiyon cihaz idsi, istenen verilerin başlangıç tarihi ve bitiş tarihi ve sunucunun adresi ile birlikte çalıştırılır ve kullanılan programlama diline ait bir değişkende tutulur. Kullanıcıya verilere ulaşabilmek için üç farklı yöntem sunulmuştur. İlk yöntem belirtilen tarihteki belirtilen bir sensörün verisini getirmektedir. İkinci yöntem belirtilen sensöre ait tüm verileri getirmektedir. Son yöntem ise belli bir tarih noktasındaki tüm verileri getirmektedir.

Oluşturulan bu frameworkün MATLAB dilinde kullanılabilmesi için jar dosyası uygulamaya tanıtılmalıdır. Bunun için programın kurulu olduğu dizindeki "MATLAB/[MatlabVersiyon]/toolbox/local/classpath.txt" dosyasına giderek jar dosyasının konumu ismiyle birlikte eklenmelidir. Bu işlem yapılırken MATLAB uygulaması açık ise yeniden başlatılmalıdır. Program açıldıktan sonra herhangi bir değişkene "x = com.sensors.jar.Main" ataması yapılır ve framework kullanıma hazır hale gelir. Frameworkü pythonda kullanabilmek için bir classpath ortam değişkeni oluşturularak jar dosyasının adresi atanmalıdır. Bu panele Bilgisayarım->Özellikler->Gelişmiş sistem ayarları->Ortam değişkenleri adımları takip edilerek ulaşılmaktadır. Pythonda java nesnelerini kullanabilmek için uygulamamıza "pyjnius [6]" kütüphanesi import edilmelidir. Java nesnesi ilk kez tanımlanırken "x = autoclass('com.sensors.jar.Main')" komutu kullanılmalıdır. Nesnemiz tanımlandıktan sonra java dili ile verilere ulaşılabilir.

Projede kullanılan tüm platformlar için harici durum kontrolleri yapılmıştır. Sunucuya ulaşılamaması durumunda cihazlar ve diğer platformlar kullanıcıyı bilgilendirmektedir. Sunucuya gönderilecek olan veriler beklenen şekilde gelmezse oluşacak harici durumların kontrolleri yapılmıştır. Mobil cihazda toplanan verilerin belleği doldurmaması için limit tanımlanmıştır.

Proje planlaması içerisinde bulunan tüm durumlar tamamlanmıştır.

Çalışmamız sonucunda benzer bir proje geliştirmek ya da çalışmamızı daha ileriye taşımak isteyenlere katkıda bulunabilmek için sunacağımız önerilerimiz şunlardır:

- Android işletim sistemi sensör verilerinin istenilen frekansta doğrudan alınmasına müsaade etmemektedir. Bu durum timer kullanımını zorunlu kılmaktadır.
- Projemizde sensörleri listelemek için recyclerview kullanılmıştır. Recyclerview yapısı çok fazla veri içeren listeler için sistemi yormamak adına tasarlanmıştır. Sensör sayısı kısıtlı olduğu için farklı listeleme viewları kullanılabilir. Tasarım aşamasını kolaylaştıracaktır.

- Projemiz yalnızca android cihazlar için yapılmıştır. IOS cihazlar için de bir uygulama geliştirilebilir. Sistemde bir sunucu bulunması, internete bağlı her cihaza destek sağlamaktadır. Farklı platformlar için de uygulama geliştirilebilir.
- Tüm veriler metinsel ifade olarak saklanmaktadır. Frameworkü kullanacak olan kullanıcıların bu verileri parse etmeleri gerekmektedir. Yazılabilecek fonksiyonlar ile bu sorun ortadan kaldırılabilir.

Referanslar

- [1] Google. (2019). “Developers.android,” [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview (visited on 03/02/2021).
- [2] —, (2009). “Google search central,” [Online]. Available: https://www.youtube.com/watch?v=Mjab_aZsdxw.
- [3] —, (2021). “Google android developers,” [Online]. Available: <https://square.github.io/okhttp/interceptors/> (visited on 09/05/2021).
- [4] —, (2021). “Developers.android,” [Online]. Available: <https://developer.android.com/reference/java/util/zip/GZIPOutputStream> (visited on 08/02/2021).
- [5] —, (2021). “Developers.android,” [Online]. Available: <https://developer.android.com/guide/topics/ui/layout/recyclerview> (visited on 04/02/2021).
- [6] Kivy. (2018). “Kivy, pyjnius,” [Online]. Available: <https://pyjnius.readthedocs.io/en/stable/> (visited on 10/06/2021).
- [7] Google. (2021). “Timer on android,” [Online]. Available: <https://developer.android.com/reference/java/util/Timer>.

BİRİNCİ ÜYE

İsim-Soyisim: Burak BOZ
Doğum Tarihi ve Yeri: 03.10.1995, İstanbul
E-mail: 11118706@std.yildiz.edu.tr
Telefon: 0530 852 87 90
Staj Tecrübeleri: Evrimnet İnternet Hizmetleri
Magis Teknoloji

İKİNCİ ÜYE

İsim-Soyisim: Bahattin KOÇ
Doğum Tarihi ve Yeri: 15.07.1997, İstanbul
E-mail: 11118704@std.yildiz.edu.tr
Telefon: 0536 745 34 96
Staj Tecrübeleri:

Proje Sistem Bilgileri

Sistem ve Yazılım: Windows 10 İşletim Sistemi, Android 8.0, JAVA 8, Android Studio, Eclipse, Tomcat 9.0, Xampp
Gerekli RAM: Mobil cihaz için: 2GB, Sunucu için: 8GB
Gerekli Disk: 500MB