T.C.

BEYKENT ÜNİVERSİTESİ

MÜHENDİSLİK-MİMARLIK FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

**MADEN GÜVENLİK SİSTEMİ**

**Yazılım Mühendisliği Bitirme Çalışması**

Bitirme Çalışmasını Hazırlayanlar

17030130040 MELEK CEVİZ

17030130001 İLAYDA İŞLEK

17030130042 ŞULE KARAKURT

DANIŞMAN : DR. ZEYNEP ALTAN

İstanbul -2021

**İÇİNDEKİLER**

1- Şekiller Listesi……………………………………………………………………………....3

2- Kısaltmalar………………………………………………………………………………..…4

3- Özet………………………………………………………………………………………….4

1- Giriş………………………………………………………………………………………

2- Genel Kısımlar……………………………………………………………………………..

3- Kullanılan Araç ve Yöntem………………………………………………………………….

3.-1- Arka Uç Arduino Geliştirme……………………………………………………………

3.2- Arka Uç RPI Geliştirme…………………………………………………………………

3.3- Android Geliştirme……………………………………………………………………

3.3.1- Mobil Uygulama Ekran Çıktıları………………………………………………

4- Sonuç…………………………………………………………………………………….

Kaynaklar………………………………………………………………………………….

**ŞEKİLLER LİSTESİ**

Şekil 1: Arka Uç Arduino Geliştirme: Tüm classları dahil etme……………………………….

Şekil 2: Arka Uç Arduino Geliştirme: Main Objeleri başlatma………………………………

Şekil 3: Arka Uç Arduino Geliştirme: Main setup……………………………………………

Şekil 4: Arka Uç Arduino Geliştirme: Main Timer…………………………………………..

Şekil 5: Arka Uç Arduino Geliştirme: Main runAlgorithm() fonksiyonunu çağırma………..

Şekil 6: Arka Uç Arduino Geliştirme: Main sensör verilerini ayıklama……………………….

Şekil 7: Arka Uç Arduino Geliştirme: Abu.h gerekli headerleri ekleme……………………

Şekil 8: Arka Uç Arduino Geliştirme: abu.h ABU içindeki fonksiyonların titleları……….

Şekil 9: Arka Uç Arduino Geliştirme:ABU.cpp Sensörlerden gelen veri türüne göre işlem yapma…………………………………………………………………………………………..

Şekil 10: Arka Uç Arduino Geliştirme: ABU.cpp anormal verileri işleme…………………..

Şekil 11: Arka Uç Arduino Geliştirme: Abu.cpp kritik verileri işleme………………………

Şekil 12: Arka Uç Arduino Geliştirme: ABU.cpp verileri transfer etmek üzere iletme………

Şekil 13: Arka Uç Arduino Geliştirme: Alarm.cpp Alarmı açma ve kapatma …………………

Şekil 14: Arka Uç Arduino Geliştirme: O2 Sensoru.h dosyası………………………………….

Şekil 15: Arka Uç Arduino Geliştirme: CO2 Sensoru.h dosyası………………………………..

Şekil 16: Arka Uç Arduino Geliştirme: CO Sensoru.h dosyası…………………………………

Şekil 17: Arka Uç Arduino Geliştirme: CH4 Sensoru.h dosyası…………………………

Şekil 18: Arka Uç Arduino Geliştirme: SicaklikSensoru.h dosyası……………………………

Şekil 19: Arka Uç Arduino Geliştirme: NemSensoru.h dosyası………………………………..

Şekil 20: Arka Uç Arduino Geliştirme: BasincSensoru.h dosyası…………………………

Şekil 21: Arka Uç Arduino Geliştirme:Havalandırma.cpp Havalandırmayı başlatma………..

Şekil 22: Arka Uç Arduino Geliştirme: Havalandırma.cpp Havalandırma seviyesini yükseltme………………………………………………………………………………………

Şekil 23: Arka Uç Arduino Geliştirme: Havalandırma.cpp Havalandırmayı kapatma………….

Şekil 24: Arka Uç Arduino Geliştirme:Zamanlama.cpp ……………………………………

Şekil 25: Arka Uç Arduino Geliştirme: Zamanlama.cpp………………………………………..

Şekil 26: Arka Uç RPI Geliştirme: Main.py…………………………………………………

Şekil 27: Arka Uç RPI Geliştirme: Manager.py ……………………………………………..

Şekil 28: Arka Uç RPI Geliştirme: DBYonetim.py…………………………………………..

Şekil 29: Arka Uç RPI Geliştirme: logfile.txt log dosyası örneği……………………………

Şekil 30: Arka Uç RPI Geliştirme: DBYonetim.py Verilerin veritabanına kaydı…………….

Şekil 31: Android Geliştirme: Kullanıcı Girişi(Welcome.js)………………………………….

Şekil 32: Android Geliştirme: Parola Sıfırlama(ForgotPassword.js)…………………………..

Şekil 33: Android Geliştirme: Alarmın Fiziksel Olarak Değiştirilmesi(Home.js)......................

Şekil 34: Android Geliştirme: Havalandırmanın Fiziksel Olarak Değiştirilmesi(Home.js).......

Şekil 35: Android Geliştirme: Havalandırma Şiddetinin Değiştirilmesi(Home.js).....................

Şekil 36: Android Geliştirme: Fiziksel Koşulların Görüntülenmesi(Pyschial.js)………………

Şekil 37: Android Geliştirme: 112 Acil Servis Arama(Home.js).................................................

Şekil 38: Android Geliştirme: Günlük Raporların Görüntülenmesi(reports.js)……………

Şekil 39: Android Geliştirme: Haftalık Raporların Görüntülenmesi(reports.js)………………..

Şekil 40: Android Geliştirme: Aylık Raporların Görüntülenmesi(reports.js)………………

Şekil 41: Android Geliştirme: Yıllık Raporların Görüntülenmesi(reports.js)………………….

Şekil 42: Home.js Bildirimlerin Modal ekranında görüntülenmesi……………………………

Şekil 43: Home.js Bildirimlerin kapatılması……………………………………………………

Şekil 44: Home.js Görüntülenecek bildirimlerin yönetimi……………………………………..

Şekil 45: Mobil Arayüzü: Kullanıcı Giriş Ekranı………………………………………………

Şekil 46: Mobil Arayüzü: Başarısız Kullanıcı Giriş Ekranı ………………………………

Şekil:47: Mobil Arayüzü: Şifre Yenileme Ekranı…………………………………………..

Şekil:48: Mobil Arayüzü: Yeni Şifre İçin Yönlendirme Ekranı………………………………

Şekil:49: Mobil Arayüzü: Yeni Şifre Belirleme Ekranı……………………………………….

Şekil:50: Mobil Arayüzü: Başarılı Şifre Yenileme Ekranı……………………………………..

Şekil 51: Mobil Arayüzü: Kullanıcı Arayüzü…………………………………………………

Şekil 52: Mobil Arayüzü: Havalandırma Kontrol ve Geribildirim Ekranı …………………..

Şekil 53: Mobil Arayüzü: Havalandırma Kontrol ve Geribildirim Ekranı…………………..

Şekil 54: Mobil Arayüzü: Alarm Kontrol ve Geribildirim Ekranı ………………………….

Şekil 55: Mobil Arayüzü: Havalandırma Seviyesi Kontrol Ekranı…………………………..

Şekil 56: Mobil Arayüzü: Havalandırma Seviyesi Kontrol Ekranı……………………………

Şekil 57: Mobil Arayüzü: Ortamın Fiziksel Koşullarını Anlık Olarak Görüntüleme Ekranı…

Şekil 58: Mobil Arayüzü: Periyodik Rapor Görüntüleme Ekranı……………………………..

Şekil 59: Mobil Arayüzü: 112 Acil Servis Kontrol Ekranı …………………………………….

Şekil 60: Mobil Arayüzü: 112 Acil Servis Kontrol Ekranı …………………………………….

Şekil 61: Maden Güvenlik Sistemi Prototip…………………………………………………….

**KISALTMALAR**

ABU: Arduino Base Unit

CH4 : Metan

CO: Karbonmonoksit

CO2: Karbondioksit

O2 : Oksijen

IoT : Internet Of Things

MGS: Maden Güvenlik Sistemi

RPI: Raspberry Pi

**ÖZET**

Maden Güvenlik Sistemi, madenlerde oluşan grizu patlamalarını ve yangın gibi felaketleri önlemek amacıyla geliştirilen bir sistemdir. Madencilik sektörü, iş kazalarının ölümle sonuçlandığı sektörlerin başında geliyor. Biz bu sorunu gördükten sonra bu problemi çözmeye yönelik yazılım ürünü geliştirme kararı aldık.

Akıllı sistemlerin kontrolü ve güvenliğinin sağlanması IoT teknolojisinin kilit konularından biridir. Bu projede de maden işçilerinin güvenliğini sağlamak için IoT teknolojisi kullanılmıştır.

Maden ocaklarındaki patlama ve yangınların engellenmesini hedefleyen bu proje sayesinde ortamın fiziksel koşulları sürekli olarak ölçülüp mobil uygulamaya aktarılacaktır. Fiziksel koşullar havanın nemi, sıcaklığı, basıncı ve ortamda bulunan metan, karbonmonoksit, karbondioksit ve oksijen gazlarıdır. Bu ölçümler ardunioya bağlı sensörler tarafından gerçekleştirilir. Ölçümlerde meydana gelebilecek tehlike durumlarında alarm veya havalandırma otomatik olarak devreye girecektir. 7 farklı sensörden alınan veriler arduino aracılığıyla Raspberry Pi’ ye aktarılır. Bu sayede aldığımız verileri periyodik raporlar halinde yorumlayabiliriz . Bu projeyi seçme amacımız şu anda madencilikte çalışan bir mobil uygulamaya sahip olunmamasıdır.

Projeyi geliştirirken asıl amacımız hali hazırda var olan maden güvenlik uygulamasının daha fazla işlevi yerine getirmesini sağlamak ve kullanıcıya verilen yetkileri arttırmaktır. Bu sayede kullanıcı bütün olası durumlara karşı çeşitli önlem alabilecektir. Projemizin diğerlerinden farkı iki veya üç ölçüm yapması değil, 7 farklı fiziksel koşulu sensörler aracılığıyla entegre bir şekilde çalıştırması ve maliyetinin çok düşük olmasıdır. Böylelikle tüm maden ocaklarında kolay bir şekilde kullanılabilecektir.

**1.GİRİŞ**

Maden Güvenlik Sisteminde, CH4, CO, C02, O2, sıcaklık, nem ve basınç değerlerinin, sensörler tarafından ölçülerek ABU ‘ ya aktarılması ve bu değerlerin entegre olarak karşılaştırılması yapılır. Daha sonra karşılaştırma sonucu çıkan değerlerin, risk durumuna göre sınıflandırılması ile tehlike seviyeleri oluşturulur. Tehlike seviyeleri düşük, normal, yüksek ve anormal olarak sınıflandırılmıştır. RPI ‘ye aktarılan değerler hem veritabanına kaydedilir hemde oluşan tehlike seviyesine göre otomatik alarm ve havalandırma devreye girer. Tehlikeli bir ölçüm yapıldığında devreye giren otomatik havalandırma sistemi, ortamdaki ölçüm seviyesinin durumuna göre 1 den 5 e seviyeli olarak artmaktadır.

Sistemimizin otomatik olması dışında, kullanıcı gerekli gördüğü durumlarda havalandırma ve alarmı devreye koyabilir. Eğer otomatik olarak veya kullanıcı tarafından alarm veya havalandırma devreye koyulmuşsa kullanıcıya bildirim gönderilir.

Güvenlik uzmanı olan kullanıcı arayüz üzerinden ortamın fiziksel koşullarına, günlük, haftalık, aylık ve yıllık olarak ölçülüp kaydedilmiş 7 farklı ölçümün değerlerine erişebilmektedir.

Oluşabilecek kaza durumlarında Acil Servis arayüz üzerinden aranabilecektir. Sistemimize kullanıcı kaydı manuel olarak veritabanına gerçekleştirilmektedir. Bunun yapılmasının sebebi bir güvenlik sistemi olması ve kullanıcının kendinin kayıt olamaması gerekliliğidir.

MGS projesi React-Native teknolojisi kullanılarak mobil olarak geliştirilmiştir. Maden ocaklarındaki patlama ve yangınların engellenmesini hedefleyen bu proje sayesinde ortamın fiziksel koşulları sürekli ve otomatik olarak ölçülüp mobil uygulamaya aktarılacaktır..

Mobil uygulamamızda ortamdan ölçülen değerlerde herhangi bir yanlış değer ölçüldüğünde RPI tarafından kullanıcı uyarılmaktadır. Sistemimiz her aşamasında test edilmiş ve doğru olarak çalışmaktadır.

Madencilikte güvenlik sistemi arayüzünde belirtilen gereksinimleri sağlamakta ve kolay bir kullanıma sahiptir. Kullanıcı sisteme girişi e- posta adresi ve şifre ile sağlamaktadır. Kullanıcı şifre unutması durumunda e-posta üzerinden şifresini güncelleyebilmektedir. Tüm aşamaları tasarımımıza bağlı kalarak geliştirdik. Sistemimizde kullanıcı giriş ekranı, ana sayfa, fiziksel koşullar ve rapor tabloları olmak üzere 4 farklı arayüz bulunmaktadır. Arayüzlerin az kullanılmasının sebebi insan-bilgisayar ilişkisinde kolay kullanımı ve kullanıcının kolay adapte ve kontrolü sağlayabilmesi içindir.

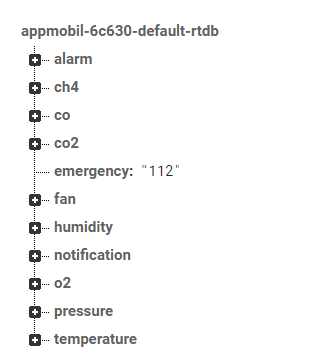
**2.GENEL KISIMLAR**

MGS; Arduino Based Unit(ABU), Raspberry Pi(RPI), Mobil uygulama ve Firebase Realtime Database(DB) üzerinde çalışmaktadır.

ABU, üzerinde bulunan sensörler vasıtasıyla topladığı verileri işleyerek anlık olarak ortamda bulunabilecek tehlike durumuna alarm ve havalandırma açıp kapatarak müdahale eder. Aynı zamanda ölçüm verilerini 1-belirlenen periyodlarda rutin olarak, 2-bunun dışında anormal veya kritik bir ölçümle karşılaştığında anlık olarak 3-ayrıca alarm ve havalandırmanın güncel durumunu RF24 modülü aracılığı ile RPI ile paylaşır. Aynı zamanda RPI’dan gelen iletişimi dinleyerek mobil uygulama kullanıcısının fiziksel sistemlere müdahale etmesine olanak tanır.

nRF24L01 modülü çift yönlü iletişime izin verdiği ve aniden tüm değerlerin hızlıca değiştiği kritik durumların veri hızını karşılayabileceği için prototipleme aşamasında ekonomik nedenlerle tercih edilmiş olsa da yeraltında yetersiz kalacağı sabittir.

RPI, RF24 modülü ile ABU ile, DB üzerinden de mobil uygulama ile iletişim halindedir. ABU’dan gelen verileri mobil uygulamaya, mobil uygulamadan gelen istekleri ABU’ya taşıyan bir köprü görevi görür. Ayrıca ABU’dan gelen verileri organize edip kritik durumlarda kullanıcıya sunulacak bilgilendirmeleri yönetir.



Firebase Realtime Databese, ağaç yapısıyla gruplandırılmış verileri kolayca yönetebilmesi, aynı anda birden fazla erişim ile veri yazma ve okumaya izin vermesi nedeniyle tercih edilmiştir.

Ağaç yapısında her bir dal, bir data grubunu işaret edecek şekilde ayrılmıştır.



Alarm ve fan durumları o anki leveli ve levelin yazıldığı anın timestamp türünden zamanını tutar.

Bu projeyi seçme amacımız şu anda madencilikte çalışan mobil uygulamaya sahip olunmamasıdır. Projeyi geliştirirken asıl amacımız hali hazırda var olan maden güvenlik uygulamasının daha fazla işlevi yerine getirmesini sağlamak ve kullanıcıya verilen yetkileri arttırmaktır. Bu sayede kullanıcı bütün olası durumlara karşı çeşitli önlem alabilecektir. Yapılacak olan tüm bu çalışmaların amacı en başta maden kazalarını minimum seviyeye düşürmek ve insan sağlığını ve yaşamını olası kazalara karşı korumaktır.

**3.KULLANILAN ARAÇ VE YÖNTEM**

Madencilikte güvenlik sistemini yazılımsal tasarıma bağlı kalarak kodladık. Oluşturduğumuz yazılımın kağıt üzerinde planlanan hali ile diyagramlara, kullanım senaryolarına, gereksinimlere, yazılım kalite güvencesi ve testine bağlı kalarak gerçekleştirmeye çalıştık. Bu sistemi geliştirirken part-whole bir yol izledik. Sistemimiz birçok yazılım parçasından oluştuğu için böyle bir yol izledik.

Mobil uygulamamızı geliştirirken React Native teknolojisini kullandık. Uygulamadaki veriler Firebase veritabanında Realtime Firebase üzerinde tutulmaktadır. Böylelikle anlık verileri senkronize etmeyi ve daha sonra günlük, haftalık, aylık ve yıllık olarak çağrılacak olan verileri uygun zaman dilimlerinde depolar. Verileri hiçbir sql sorgusuna gerek kalmadan json parametreleri ile yönetebilir.

Uygulamamızı geliştirirken Lean Metodolojisini kullandık. Lean Metodolojisi, verimlilik temasına odaklanan bir proje yönetimi metodolojisidir. Değeri tanımlayarak başlar ve ardından değer akışını optimize ederek ve israfı ortadan kaldırarak sürekli iyileştirme yoluyla verimliliği maksimuma çıkarır. Lean Metodolojisi, karı en üst düzeye çıkarmakla ilgilidir. Lean yönetimi, müşteri bakış açısını dikkate alır. Lean Metadolojisi sayesinde uygulama yalın bir şekilde geliştirilmiştir. Uygulamamızı parçalara ayrırarak geliştirmemiz sayesinde birçok yazılımcı tarafından kontrol edilebilir bir sistem oluşturulmuş olur. Her parça geliştirilmeye başlanmadan önce planlanmaktadır. Madencilikte güvenlik sistemi Arduino ve Rpi sistem parçaları dışında 4 arayüzden oluşmaktadır. Bunlar ; Kullanıcı girişi arayüzü, Anasayfa arayüzü, Raporlar arayüzü, Fiziksel Koşullar arayüzüdür. Maden Güvenlik Sistemi, sensörlerden aldığı verileri bu 6 bileşene sırasıyla ve uygun zamanda gönderir.

Düşünülmüş olan ve daha sonra kağıt üzerinde tasarlanmış olan mantıksal cümlelere, mantıksal diyagram ve tablolara bağlı kalınarak kodlama aşamasına geçilmiştir. Fonsiyonel olan ve fonksiyonel olmayan gereksinimlere bağlı kalarak use case senaryoları ve squence diyagramları oluşturulmuştur. Düşündüğümüz mantıksal cümlelere bağlı kalarak tüm bu durumları karşılayacak yazılım kodları yazdık.

Kodumuzu yazarken ve daha son aşamada düzenlerken clean code ilkerine de bağlı kalmaya çalıştık. Kodumuzun usability , stability, efficiency, reliability, resilience, adaptability olmasına yönelik bir tasarım ile kodladık.

Kodumuzun clean code ilkelerinden olan Maintainability ‘ i sağlayabilmesi için kodumuzun rigit olmayan yani esnek olan bir yapıda geliştirmeye çalıştık. Böylelikle koddaki değiştirilmesi gereken bir bölüm, başka bir yeri etkilemeyecektir. Yazılım parçalarımız kodlanırken open/closed solid ilkesi benimsenmiştir. Kodumuz değişime kapalı ve genişlemeye açık bir yapıdadır.

Kodumuzu modüler bir yapıda geliştirmeye çalıştık. Modüller arası düşük coupling’ i sağlayarak kodumuzun daha kolay yönetilebilir ve geliştirilebilir olmasını sağladık. Aynı zamanda kodumuzun ; erişilebilirlik, kavramsal bütünlük, birarada çalışabilirlik, müdahale edilebilirlik, yönetilebilirlik, performans, güvenilirlik, tekrar kullanılabilirlik, ölçeklenebililrlik, desteklenebilirlik, test edilebilirlik, güvenlik ve kullanılabilirlik yazılım kalite niteliklerini karşılamasını sağladık.

Projemizde temelde 3 ana bölüm vardır:

İlk adım 7 farklı fiziksel koşulun, sensörler yardımı ile ölçülüp ABU’ya aktarılmasıdır. Burada her fiziksel koşulu için ABU tarafından bir karşılaştırma algoritması çalıştırılır ve belirlenen tehlike durumuna göre otomatik havalandırma veya otomatik alarmın tetiklenmesi sağlanır.

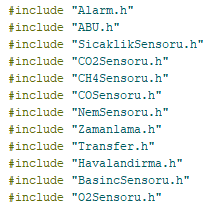
İkinci aşamada ise 7 farklı sensörden alınan veriler ABU aracılığıyla ana makinemiz olan RPI’ ye aktarılır. RPI, topladığı veriler ile veritabanını güncel tutar.

Üçüncü adım ise mobil uygulama aşamasıdır. Kullanıcı, arayüz üzerinden istekte bulunur. Kullanıcının istediği bilgiler veritabanından alınarak mobil uygulama aracılığıyla kullanıcıya gösterilir

**3.1.Arka Uç Arduino Geliştirme ()**

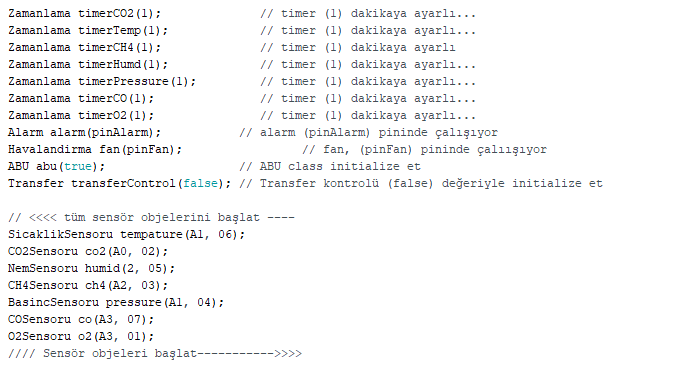
**Projenin Arduino kısmında gerçekleştirilen iş akışı şu şekildedir:**

1. Sensörlerden ortam fiziksel verilerinin toplanması
2. Toplanan verilerin risk seviyelerine göre gruplandırılması
3. Risk grubunda olmayan verilerin belirli zaman aralıklarında aktarımı için zamanlayıcı kurulması
4. Risk grubunda veri yakalandığında ortamda alarm ve havalandırma düzenleyicilerinin fiziksel olarak çalıştırılması
5. Risk grubundaki verilerin Raspberry Pi'a transfer edilmesi
6. Risk grubunda olmayan verilerin belirlenen aralıklarla Raspberry Pi'a transfer edilmesi



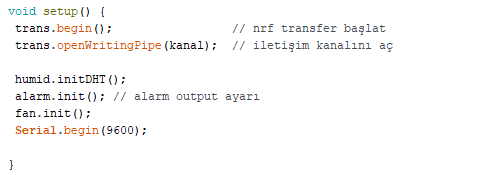
**Şekil 1: Arka Uç Arduino Geliştirme: Tüm classları dahil etme**

Arduino dili, C ve C++ dillerinden derlenmiş, birçok OOP özelliği içinde bulunduran yarı OOP bir dildir. “ino” dosyası main classı gibi çalışır. Öncelikle, tüm classlar header ve implementasyon(cpp) dosyası olarak oluşturulmuş ve header dosyaları, main classa include edilmiştir.



**Şekil 2: Arka Uç Arduino Geliştirme: Main Objeleri başlatma**

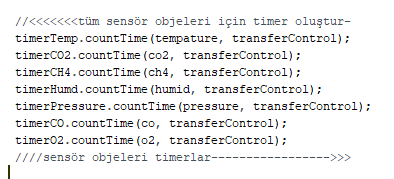
Tanımlanmış sensör classlarından nesne yaratılmış, Zamanlama classı sensör sayısı kadar örneklenmiştir. Ayrıca Alarm, fan ve transfer nesneleri yaratılmış, Manager objesi olarak görev yapacak olan ABU classından da bir obje üretilmiştir.



**Şekil 3: Arka Uç Arduino Geliştirme: Main setup**

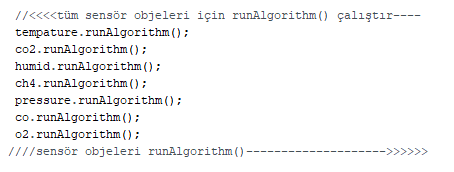
Arduino dilinde setup fonksiyonunun içinde, Arduino çalışmaya başladığında bir kere çalışacak kodlar bulunur. Burada, pinlerin fonksiyonları ayarlanır. init() fonksiyonlarında alarm ve fan pinleri output, DHT pini input olarak belirlenmiştir.

NRF11 transfer kütüphanesinin begin() ve openWritingPipe() fonksiyonları kablosuz iletişimi başlatmış, kullanılmaya hazır hale getirmiştir. setup() fonksiyonunu çalıştırmayı bir kere bitirdikten sonra, cihaz yeniden başlatılana veya program sonlandırılana kadar loop() fonksiyonu bir sonsuz döngü olarak içindeki kodları çalıştırmaya başlar.



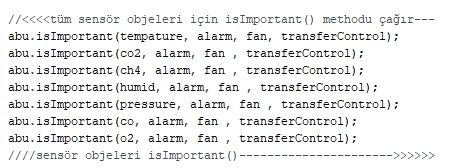
**Şekil 4: Arka Uç Arduino Geliştirme: Main Timer**

Oluşturduğumuz sensör nesneleri ile nesne sayısı kadar oluşturduğumuz timer nesnesini birleştirip, transferleri yönettiğimiz transferControl nesnesini ekleyerek timer classındaki countTime fonksiyonunu çağırdık. Bu fonksiyon, nesne oluşturulurken verilen saniye kadar sürede bir, Başlatma son aldığı değerleri transfer objesine taşır. Böylece, verilerde bir anomali olup olmamasına bakılmaksızın belirli aralıklarla sensörlerden okunan veriler veri tabanına yazılmak üzere RPI’a taşınır.



**Şekil 5: Arka Uç Arduino Geliştirme: Main runAlgorithm() fonksiyonunu çağırma**

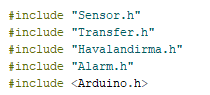
Sensörlerin, okudukları veriyi değerlendirip raw datayı, information haline getirdiği fonksiyondur. Sensörlerden okunan veriler birleştirilir, sensörün bağlamı içinde ilişkili hale getirilir.



**Şekil 6: Arka Uç Arduino Geliştirme: Main sensör verilerini ayıklama**

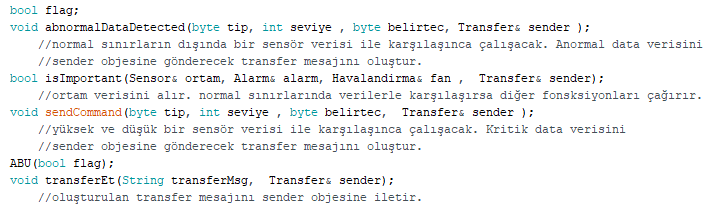
isImportant() fonksiyonu, runAlgorithm() fonksiyonundan aldığı information’ı, anlamlandırarak knowledge haline getirir ve yalnızca önemli knowledge’ı RPI’a iletilmek üzere transferControl objesine taşır.

Burada yalnızca önemli verilerin(normalden düşük ölçüm, normalden yüksek ölçüm, anormal ölçüm) seçilerek iletilmesinin ve diğer verilerin belirli zaman aralıkları ile iletilmesinin sebebi cihazların ve iletim modülünün veri yoğunluğu yaşayarak karşılıklı iletişimde tıkanmasının önüne geçmektir. Yalnızca anlamlı bir wisdom üretmeye yardımcı olacak bilgiler RPI ve ABU modülleri arasında aktarılır.

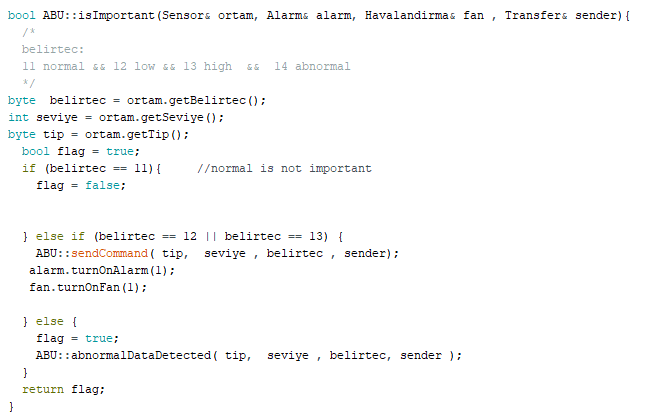


**Şekil 7: Arka Uç Arduino Geliştirme: Abu.h gerekli headerleri ekleme**

ABU classı, Arduino kodlarının çekirdeğini oluşturan, sensörlerden gelen tüm verilerin yönetildiği manager’dır. Bu özelliğinden dolayı, yönetimi altında bulundurduğu tüm classların header dosyalarını, kendi header’ında içermektedir.

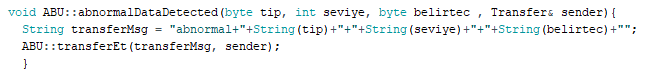


**Şekil 8: Arka Uç Arduino Geliştirme: abu.h ABU içindeki fonksiyonların titleları**



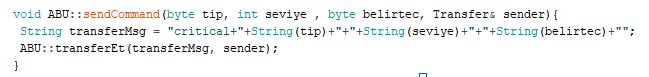
**Şekil 9: Arka Uç Arduino Geliştirme:ABU.cpp Sensörlerden gelen veri türüne göre işlem yapma**

Main penceresinde her sensör objesi için çağırılan isImportant() fonksiyonu, anormal veya kritik seviyelerde bir nesneyle çağrılması halinde abnormalDataDetected() ve send Command() fonskiyonlarını çağırır.



**Şekil 10: Arka Uç Arduino Geliştirme: ABU.cpp anormal verileri işleme**

Anormal bir data tespit edildiğinde çağırılan abnormalDataDetected() fonksiyonu, anormal data ile ilgili bir transfer mesajı oluşturarak doğrudan RPI’a transfer edilmek üzere gönderilmesi için transferEt() fonksiyonunu çağırır.



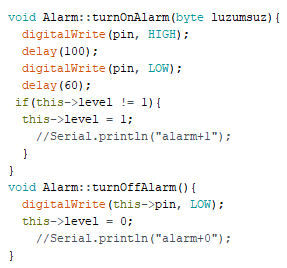
**Şekil 11: Arka Uç Arduino Geliştirme: Abu.cpp kritik verileri işleme**

Kritik sınırları aşan bir data tespit edildiğinde çağırılan abnormalDataDetected() fonksiyonu, anormal data ile ilgili bir transfer mesajı oluşturarak doğrudan RPI’a transfer edilmek üzere gönderilmesi için transferEt() fonksiyonunu çağırır.

.

 **Şekil 12: Arka Uç Arduino Geliştirme: ABU.cpp verileri transfer etmek üzere iletme**

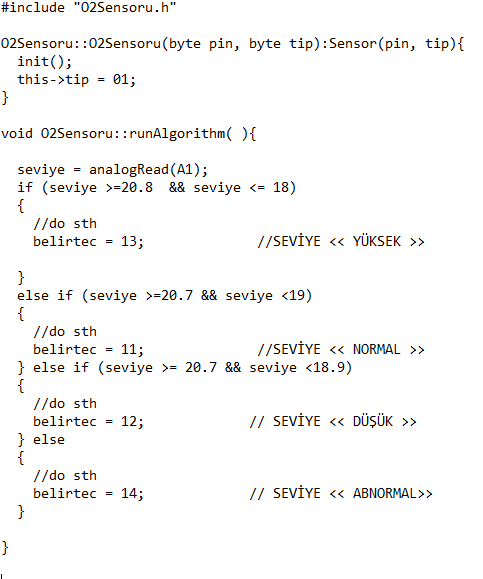
transferEt() fonksiyonu, kendisine iletilen transfer mesajını Transfer classının transferEt() fonksiyonuna iletir.



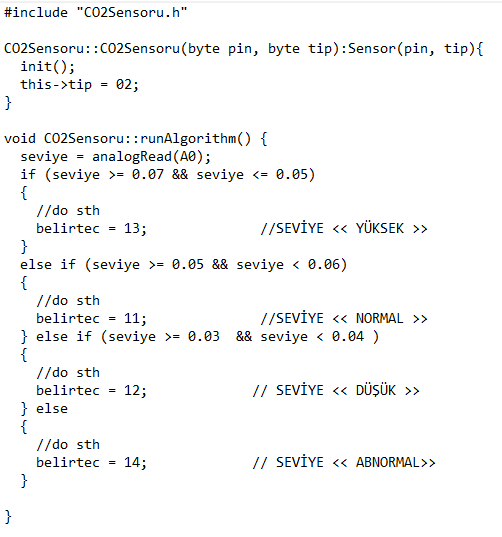
**Şekil 13: Arka Uç Arduino Geliştirme: Alarm.cpp Alarmı açma ve kapatma**

**Alarm.cpp:** Alarmın açılıp kapanması durumlarını ayarlayan fonksiyonlardır. Alarm seviyesi nesnenin attribute’u olan level değişkeninde tutulur. Ayrıca alarmın açık kapalı olması durumlarında, fiziksel olarak alarm pinine gidecek akımı ayarlar.

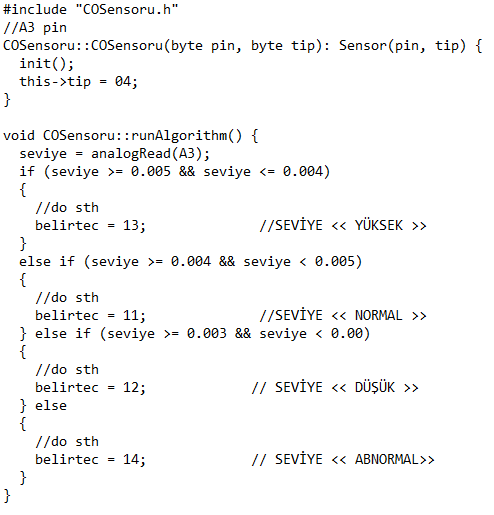
<Tip>Sensoru.cpp : Sensor classları, abstract Sensor.h classından türetilerek üretilmiştir. Her sensör için ayrı class kullanılmış olmasının nedeni, kod okunurluğunu artırmak, seviyelere daha kolay müdahale edebilmek, ileride yapılması planlanan ve yapılabilecek geliştirmelerde sensör bazında eklenen değişikliklerin diğer sensörlerde bir farklılığa yol açmadan doğrudan koda eklenebilmesi ve çalıştırılabilmesidir.



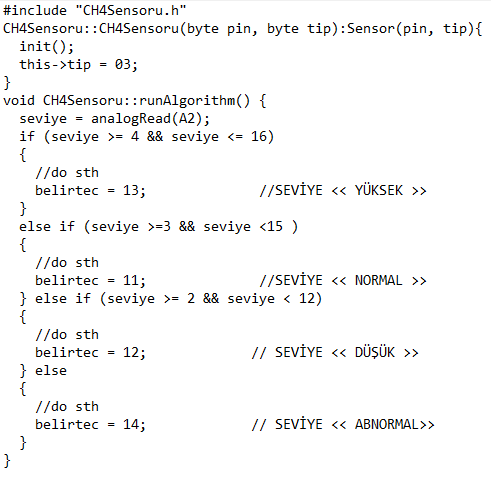
**Şekil 14: Arka Uç Arduino Geliştirme: O2 Sensoru.h dosyası**



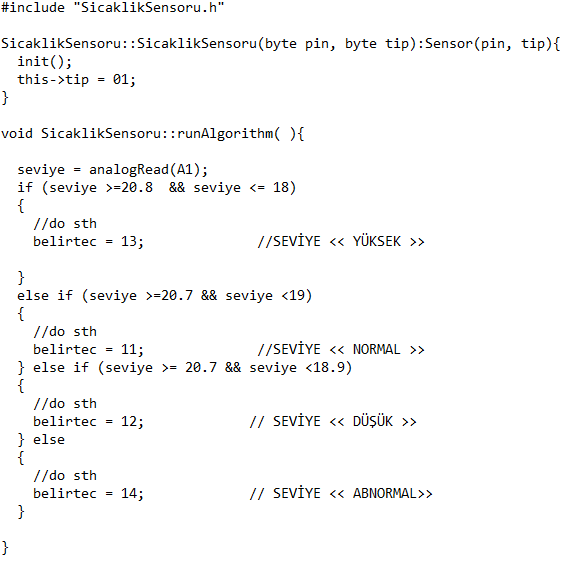
**Şekil 15: Arka Uç Arduino Geliştirme: CO2 Sensoru.h dosyası**



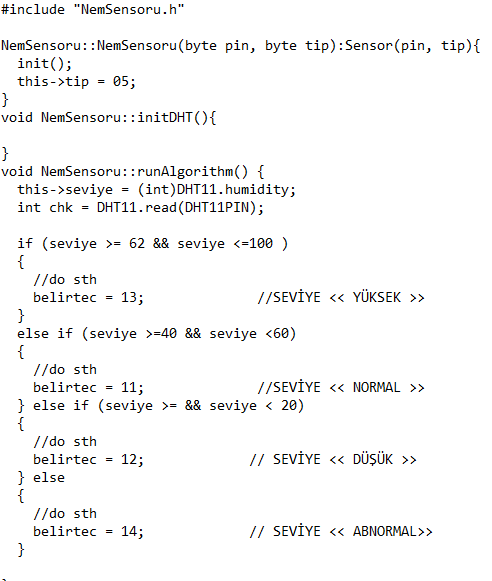
**Şekil 16: Arka Uç Arduino Geliştirme: CO Sensoru.h dosyası**



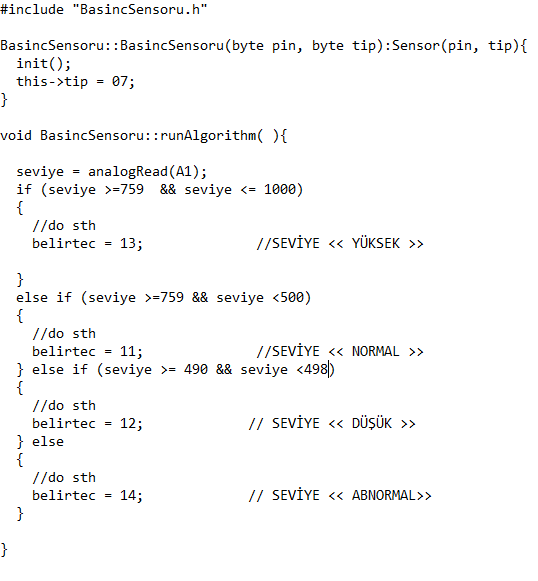
**Şekil 17: Arka Uç Arduino Geliştirme: CH4 Sensoru.h dosyası**



**Şekil 18: Arka Uç Arduino Geliştirme: SicaklikSensoru.h dosyası**



**Şekil 19: Arka Uç Arduino Geliştirme: NemSensoru.h dosyası**



**Şekil 20: Arka Uç Arduino Geliştirme: BasincSensoru.h dosyası**

**pin:** sensörün fiziksel olarak bağlı olduğu Arduino üzerindeki pin'i tutar.

**tip:** sensörün ölçtüğü değeri temsil eden değer(01 -> O2 sensörü, 02 -> CO2 sensörü...)

**seviye:** sensörden gelen ölçüm verisi

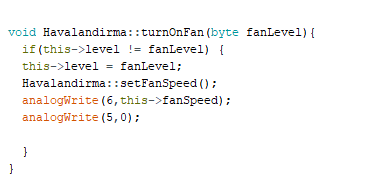
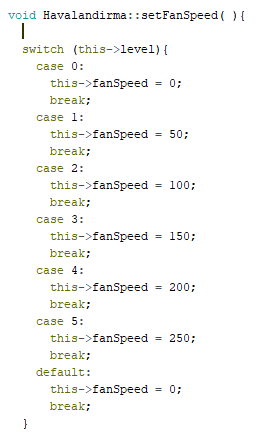
**belirtec:** alınan verinin risk grubunu temsil eden atanan değer. (11 -> normal, 12-> düşük, 13-> yüksek, 14-> anormal)

tip ve belirtec parametreleri, Arduino'nun düşük bellek kapasitesi göz önünde bulundurularak temsili byte değerleri ile işaretlenmiştir.

**init** fonksiyonları, setup bölümünde çalıştırılması gereken pinlerin belirlenmesi gibi kodları içermektedir.

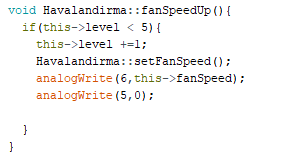
**runAlgorithm** fonksiyonu, verilerin okunması, bu verilerin olması gereken değerlerle karşılaştırmasını ve bu değerleri risk grubuna göre belirtec atanması işlemini gerçekleştirmektedir.

Burada tüm sensörlerin çalışması görülmektedir. Sensörden okunan veriye göre nesnenin belirteç attribute’unu; 11, 12, 13, 14 değerlerinden biri olarak ayarlar. Burada, belirteç değerinin byte olarak belirlenmiş olmasının sebebi, minimum bellek alanı kullanımı sağlamaktır. Arduino üzerinde tüm değişkenler, minimum bellek kullanımı sağlayacak şekilde kodlara indirgenmiştir



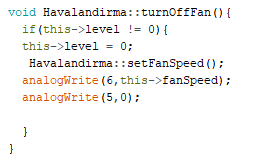
**Şekil 21: Arka Uç Arduino Geliştirme:Havalandırma.cpp Havalandırmayı başlatma**

Havalandırmayı parametre olarak verilen levelle başlatır. 0-5 arasında verilen havalandırma seviyesini fanın bağlı olduğu pine iletilecek akıma göre dönüştürür.



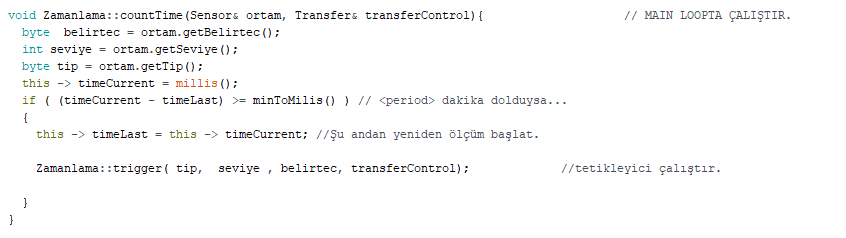
**Şekil 22: Arka Uç Arduino Geliştirme: Havalandırma.cpp Havalandırma seviyesini yükseltme**

Fan leveli 5 değilken, fen levelini bir artırır ve setFanSpeed() fonksiyonunu çağırır.

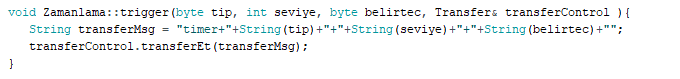


**Şekil 23: Arka Uç Arduino Geliştirme: Havalandırma.cpp Havalandırmayı kapatma**

Fanı kapatır, level ve fanSpeed’i 0 olarak güncelleyerek fanın çalışmasını fiziksel olarak durdurur.



**Şekil 24: Arka Uç Arduino Geliştirme:Zamanlama.cpp**



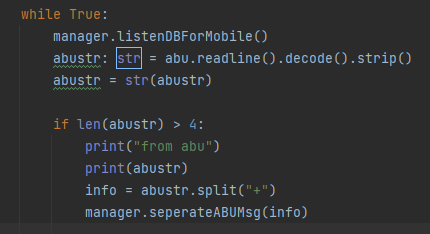
**Şekil 25: Arka Uç Arduino Geliştirme: Zamanlama.cpp**

Belirlenen periyodda zaman sayacıyla tetikleyiciyi çalıştırma ve Zaman.cpp tetiklendiğinde verileri transfer etmek üzere aktarma.

**3.2.Arka Uç Raspberry Pi Geliştirme ()**

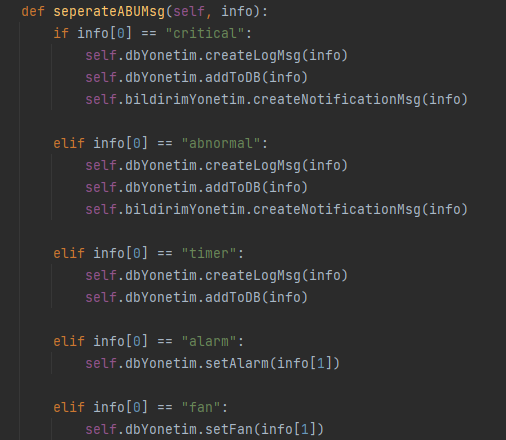
**Projenin Raspberry Pi kısmında gerçekleştirilecek iş akışı şu şekildedir:**

1. Arduino üzerinden gelen verilerin ayıklanması
2. Ayıklanan verilerin log dosyasına aktarılması
3. Ayıklanan verilerin veri tabanına aktarılması
4. Ayıklanan verilerden gerekli hallerde kullanıcıya iletilecek bildirim oluşturulması
5. Veritabanından verilerin okunması
6. Okunan verilerden geçmiş verilerin silinmesi, gruplandırılarak daha az yer kaplayacak şekilde aylık ortalamalarla yazılması
7. Okunan verilerden gerekli olanların Arduino'ya emir şeklinde iletilmesi.



**Şekil 26: Arka Uç RPI Geliştirme: Main.py**

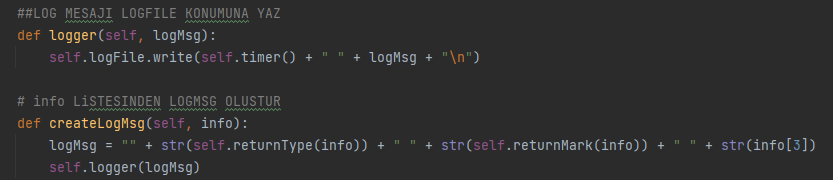
Manager classı’nın listener fonksiyonları main.py dosyasında sonsuz döngüde çalıştırılarak RPI açık olduğu sürece database ve ABU’dan gelen mesajları takip edip ilgili fonksiyonlara yönlendirme.





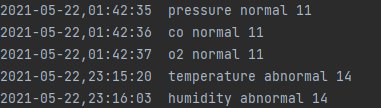
**Şekil 27: Arka Uç RPI Geliştirme: Manager.py**

Database ve ABU’dan gelen mesajların çözümlenerek uygun fonksiyonlara aktarılması.

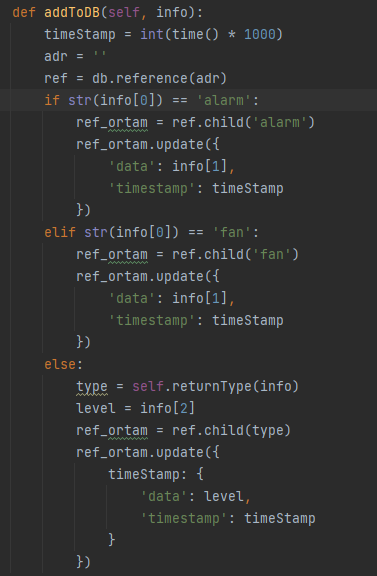


**Şekil 28: Arka Uç RPI Geliştirme: DBYonetim.py**

**Ortam verilerinin log dosyasına yazılacak şekilde düzenlenerek log dosyasına yazılması.**



**Şekil 29: Arka Uç RPI Geliştirme: logfile.txt log dosyası örneği**

****

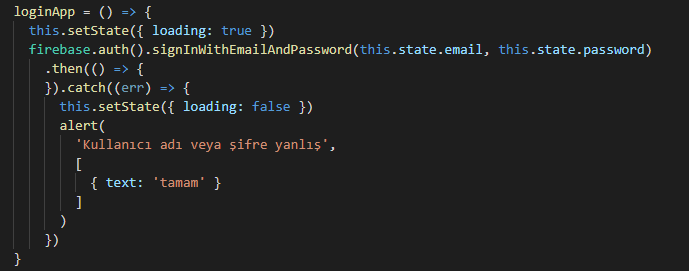
**Şekil 30: Arka Uç RPI Geliştirme: DBYonetim.py Verilerin veritabanına kaydı**

ABU’dan gelen bilgiler doğrultusunda veritabanına aktarılacak veriler alarm, fan ve ortam koşulları olarak gruplanıp veritabanına anlık timestamp verisi ile birlikte yazılır.

**3.3.Android Geliştirme ()**

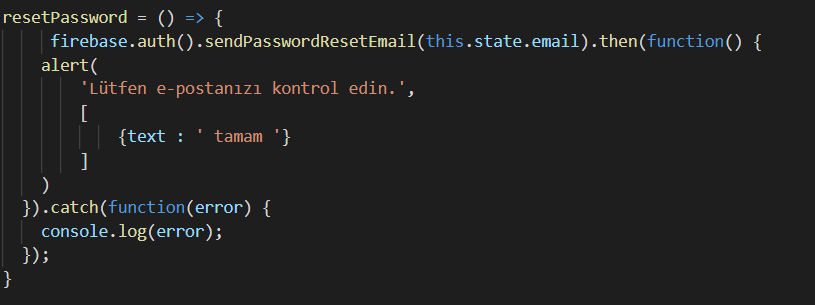
**Projenin Android kısmında gerçekleştirilen iş akışı şu şekildedir:**

1. Kullanıcı girişi yapılması
2. Fiziksel koşulların görüntülenmesi
3. Fiziksel koşulların dönemlik raporlar halinde görüntülenmesi
4. 112 Acil Servis Arama işleminin gerçekleştirilmesi
5. Arduino ucunda alarmın fiziksel olarak açık/kapalı durumunun görüntülenmesi
6. Arduino ucunda havalandırmanın fiziksel olarak açık/kapalı durumunun ve seviyesinin görüntülenmesi
7. Arduino ucunda alarmın fiziksel olarak açık/kapalı durumunun değiştirilmesi
8. Arduino ucunda havalandırmanın fiziksel olarak açık/kapalı durumunun ve seviyesinin değiştirilmesi
9. Gelen bildirimlerin gösterilmesi



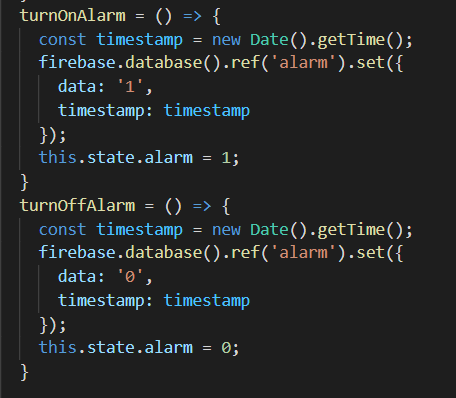
**Şekil 31: Android Geliştirme: Kullanıcı Girişi(Welcome.js)**

Welcome.js üzerinde kullanıcı girişi firebase authentication ile yapılmıştır. Az sayıda profesyonel kullanıcılar tarafından kullanılacak uygulamada her kullanıcı grubunun veritabanını kendi hesaplarıyla ilişkilendirmiş halde kendi ellerinde tutması planlandığı için bir kullanıcı kayıt bölümü bulunmamaktadır. Kullanıcı kayıtları veritabanına doğrudan, manuel olarak eklenecektir..

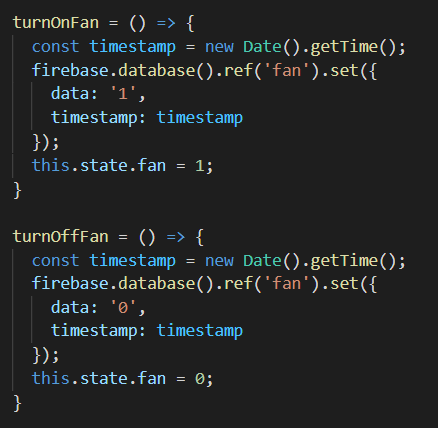


**Şekil 32: Android Geliştirme: Parola Sıfırlama(ForgotPassword.js)**

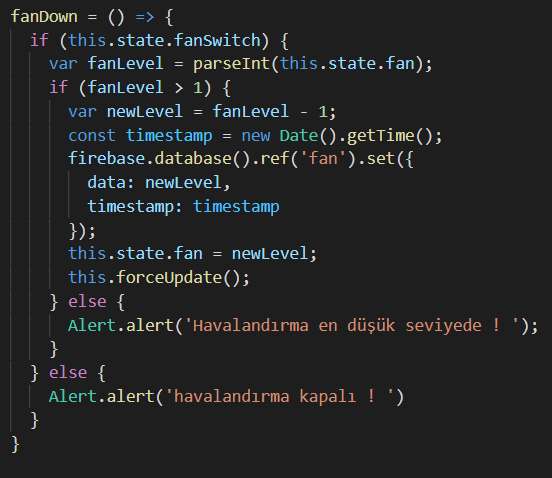
Kullanıcının mail adresini girip parolayı sıfırla butonuna tıklamasıyla resetPassword metodu çalışır. Kullanıcının girdiği mail adresi ile veritabanından alınan mail adresi eşleşiyorsa kullanıcıya yeni şifresi mail olarak gönderilir. Kullanıcının Giriş Yap butonuna tıklamasıyla tekrar giriş ekranına yönlendirilir.

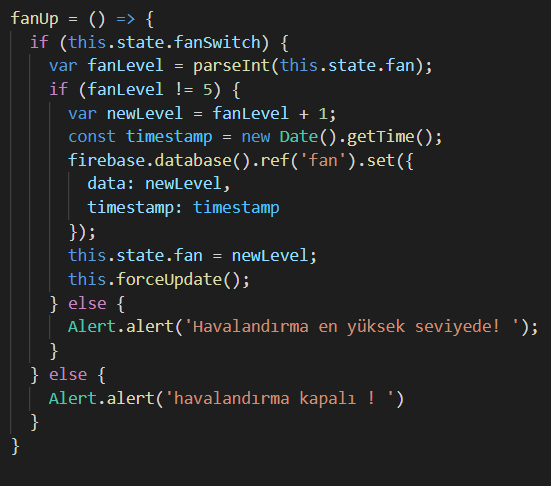


**Şekil 33: Android Geliştirme: Alarmın Fiziksel Olarak Değiştirilmesi(Home.js)**



**Şekil 34: Android Geliştirme: Havalandırmanın Fiziksel Olarak Değiştirilmesi(Home.js)**

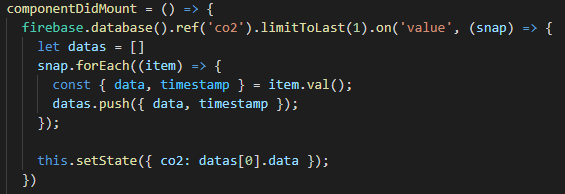




**Şekil 35: Android Geliştirme: Havalandırma Şiddetinin Değiştirilmesi(Home.js)**

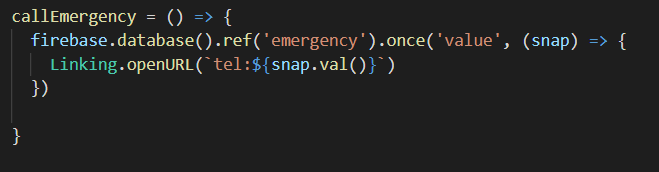
Home.js dosyasındaki bu kod parçası, alarm ve fanın açılması emirlerini veritabanına taşımaktadır.

Raspberry Pi üzerine kurulacak database listener ile bu veriler sürekli olarak izlenecek ve bir değişiklik olması halinde Arduino ucuna iletilecektir. Bu sayede kullanıcı gerekli durumlarda alarm veya havalandırmayı açıp kapatabilecek, havalandırma şiddetini artırıp azaltabilecektir.



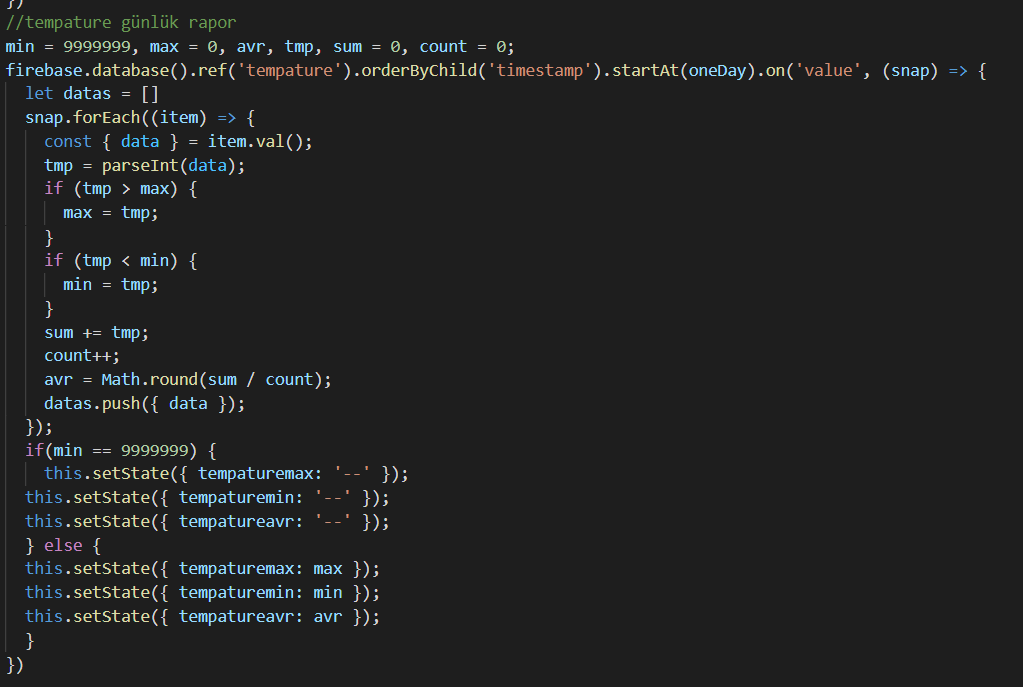
**Şekil 36: Android Geliştirme: Fiziksel Koşulların Görüntülenmesi(Pyschial.js)**

Physical.js dosyasındaki bu kod parçasında, fiziksel koşullardan biri olan CO2 verisi, veritabanından sürekli olarak sorgulanmakta, en son kaydedilen veri ekranda kullanıcıya gösterilecek state üzerine yazılmaktadır. Bu işlem her yedi koşul için gerçekleştirilmektedir.



**Şekil 37: Android Geliştirme: 112 Acil Servis Arama(Home.js)**

Bu kod parçasında 112 Acil Servis Ara butonuna tıklanmasıyla acil servis numarası veritabanından çekilerek mobil uygulama üzerinden arama işlemi gerçekleştirilmektedir.



**Şekil 38: Android Geliştirme: Günlük Raporların Görüntülenmesi(reports.js)**

OneDay timestamp değeri, şu anki zamandan 24 saat öncesinin milisaniye cinsinden değerini

içermektedir.

Bu sorguyla istenen ortam değerinin son 24 saat içerisindeki tüm verileri çekilmekte, içerisinden en yüksek, en düşük ve ortalama değerleri alınmaktadır.Bu eylem, tüm ortam değişkenleri için yapılmıştır.



**Şekil 39: Android Geliştirme: Haftalık Raporların Görüntülenmesi(WeeklyReport.js)**

OneWeek timestamp değeri, şu anki zamandan 1 hafta öncesinin milisaniye cinsinden değerini

içermektedir.

Bu sorguyla istenen ortam değerinin son 1 hafta içerisindeki tüm verileri çekilmekte, içerisinden en yüksek, en düşük ve ortalama değerleri alınmaktadır. Bu eylem, tüm ortam değişkenleri için yapılmıştır.

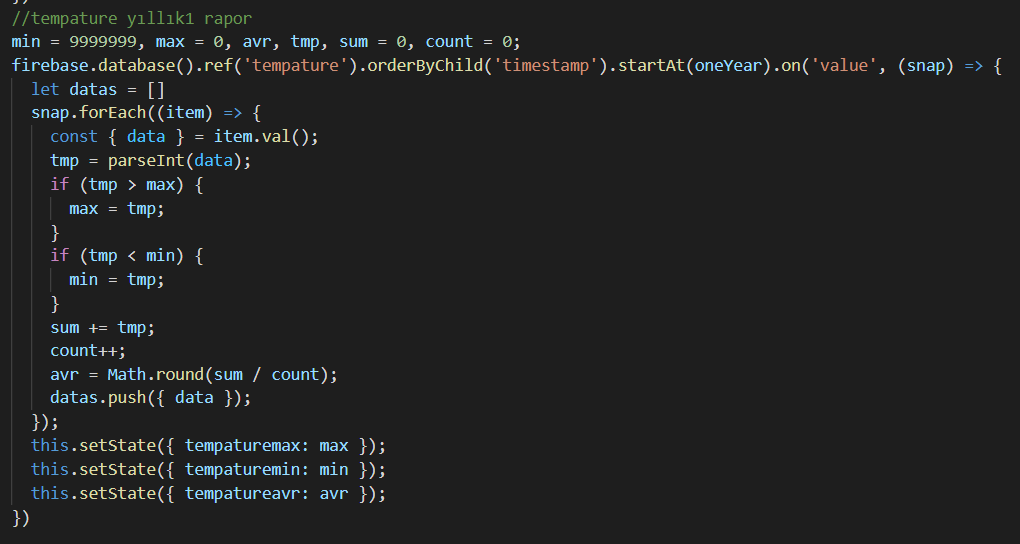


**Şekil 40: Android Geliştirme: Aylık Raporların Görüntülenmesi(MonthlyReport.js)**

OneMonth timestamp değeri, şu anki zamandan 1 ay öncesinin milisaniye cinsinden değerini

içermektedir.

Bu sorguyla istenen ortam değerinin son 1 ay içerisindeki tüm verileri çekilmekte, içerisinden en yüksek, en düşük ve ortalama değerleri alınmaktadır. Bu eylem, tüm ortam değişkenleri için yapılmıştır.

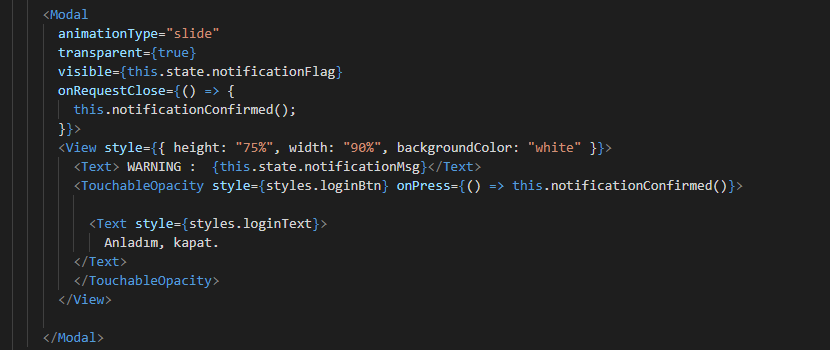


**Şekil 41: Android Geliştirme: Yıllık Raporların Görüntülenmesi(AnnualReport.js)**

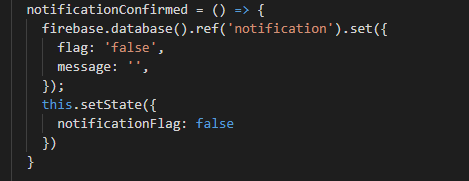
OneYear timestamp değeri, şu anki zamandan 1 yıl öncesinin milisaniye cinsinden değerini

içermektedir.

Bu sorguyla istenen ortam değerinin son 1 yıl içerisindeki tüm verileri çekilmekte, içerisinden en yüksek, en düşük ve ortalama değerleri alınmaktadır. Bu eylem, tüm ortam değişkenleri için yapılmıştır.

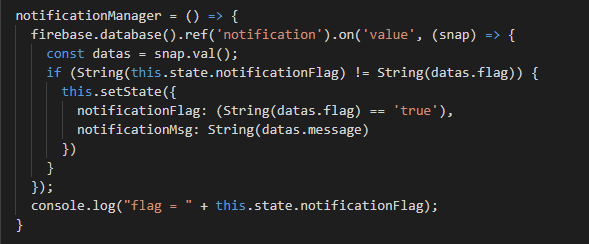


**Şekil 42: Home.js Bildirimlerin Modal ekranında görüntülenmesi**



**Şekil 43: Home.js Bildirimlerin kapatılması**

Kullanıcı bildirimi kapatırken mesaj silinir ve flag fieldi false olarak ayarlanır.



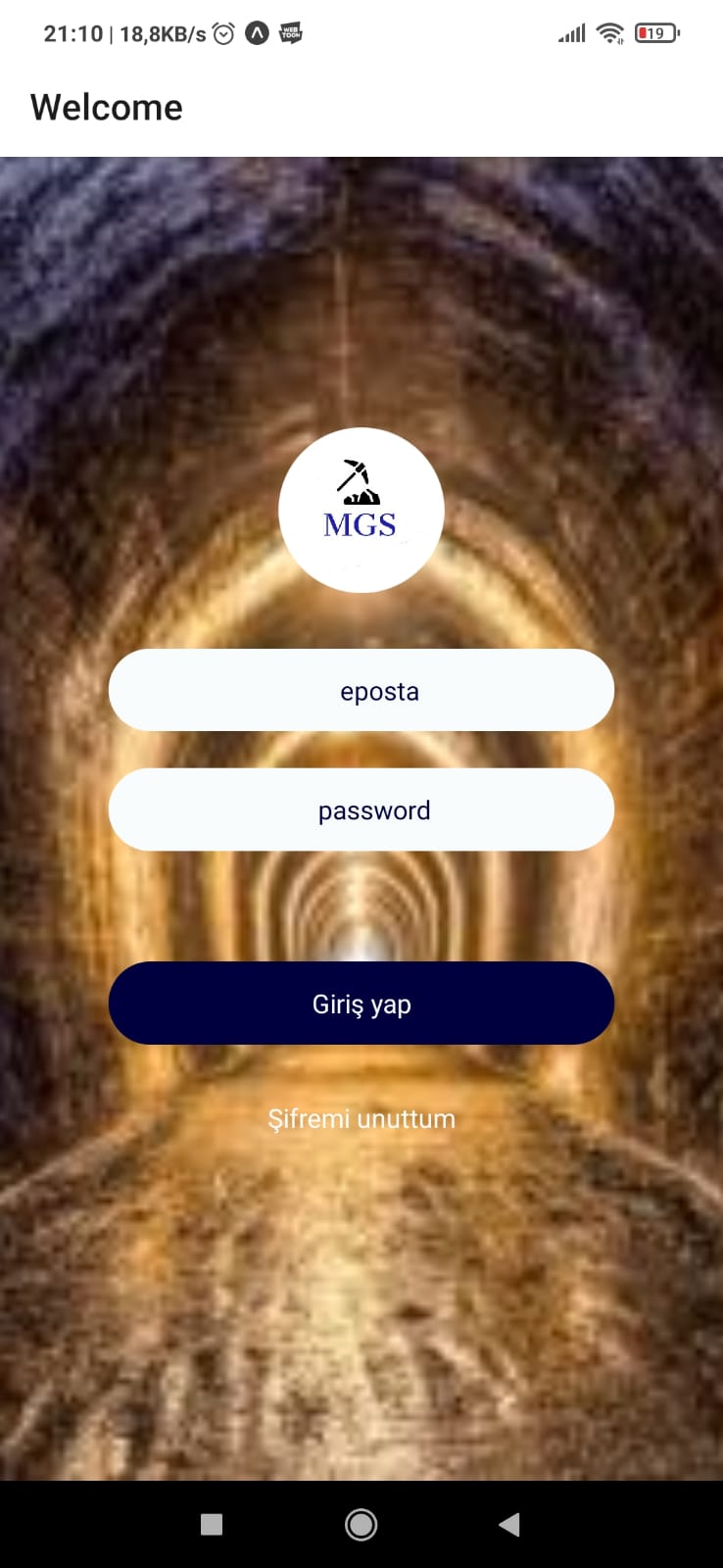
**Şekil 44: Home.js Görüntülenecek bildirimlerin yönetimi**

Açıklama: Veritabanında bildirimler boolean bir flag ve bir mesaj olarak tutulur. Bildirimler, veritabanına kaydedilirken flag true ayarlanır. Flag field’i true olan bildirim henüz kullanıcı tarafından onaylanıp kapatılmamıştır demektir ve bu kapatma gerçekleşene kadar kullanıcıya gösterilir.

Kullanıcı bildirimi kapatırken mesaj silinir ve flag fieldi false olarak ayarlanır.

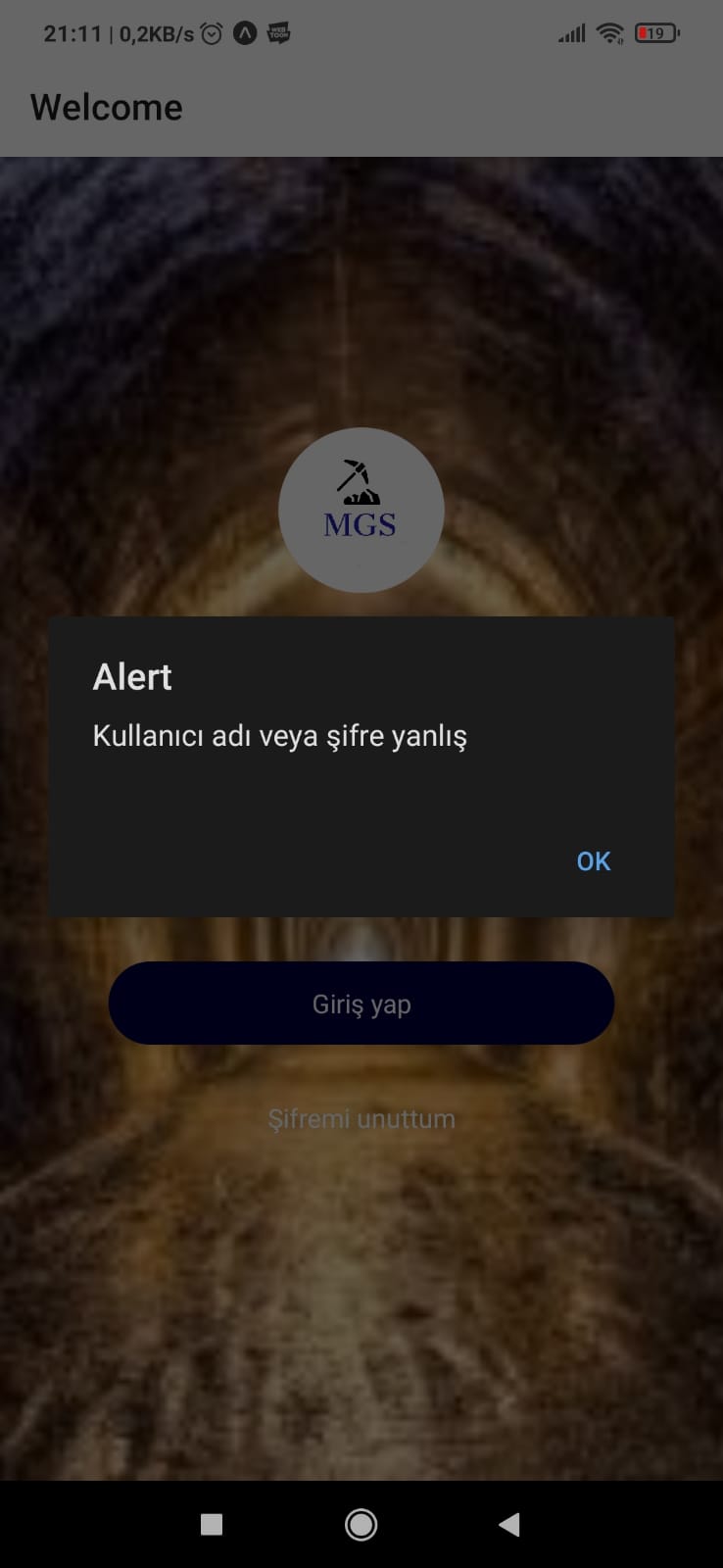
**3.3.1.MOBİL UYGULAMA EKRAN ÇIKTILARI**

Bu bölümde mobil uygulama ekran çıktıları gösterilmektedir.



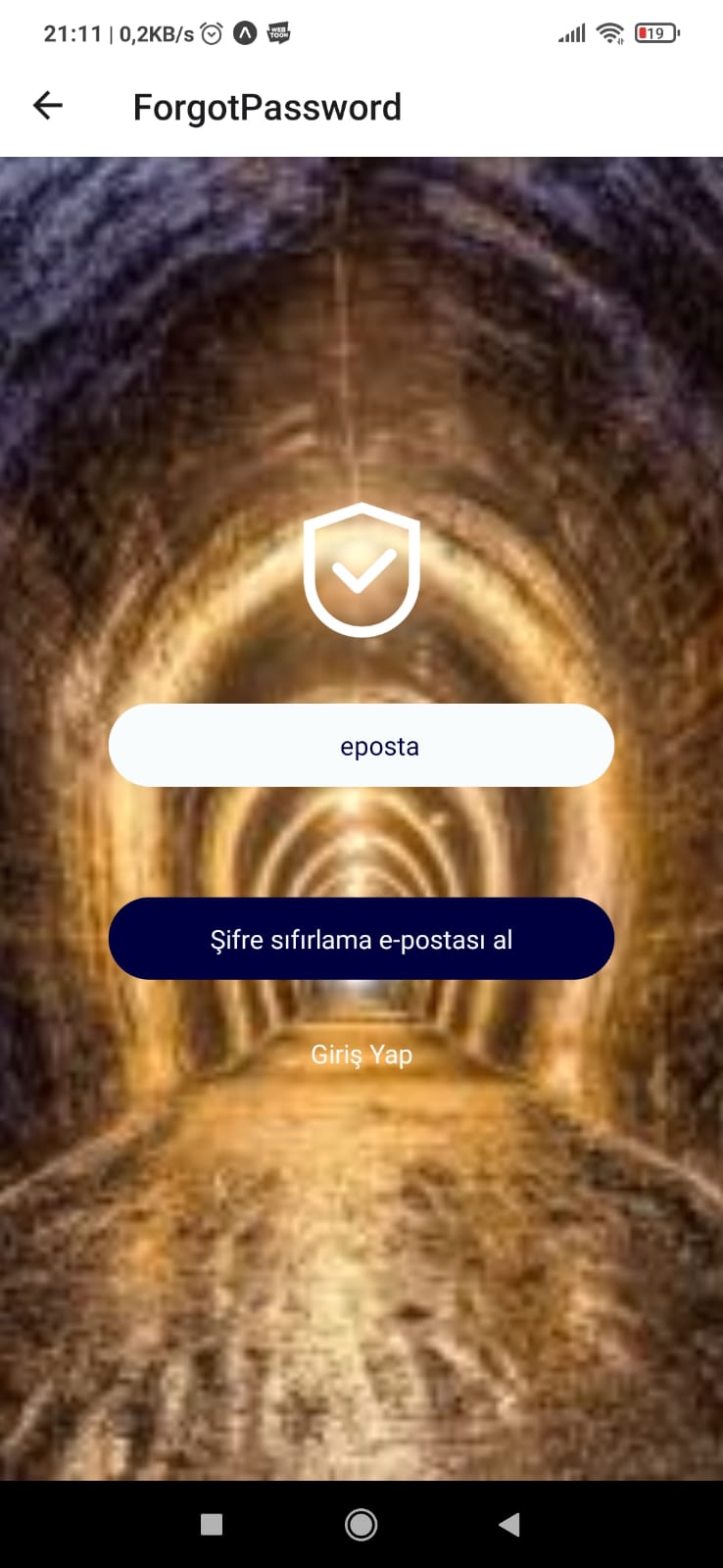
**Şekil:45: Mobil Arayüzü: Kullanıcı Giriş Ekranı**

Kullanıcı Maden Güvenlik Sistemine erişebilmek için uygulamaya kullanıcı adı parola bilgilerini girerek kolaylıkla giriş yapabilir.



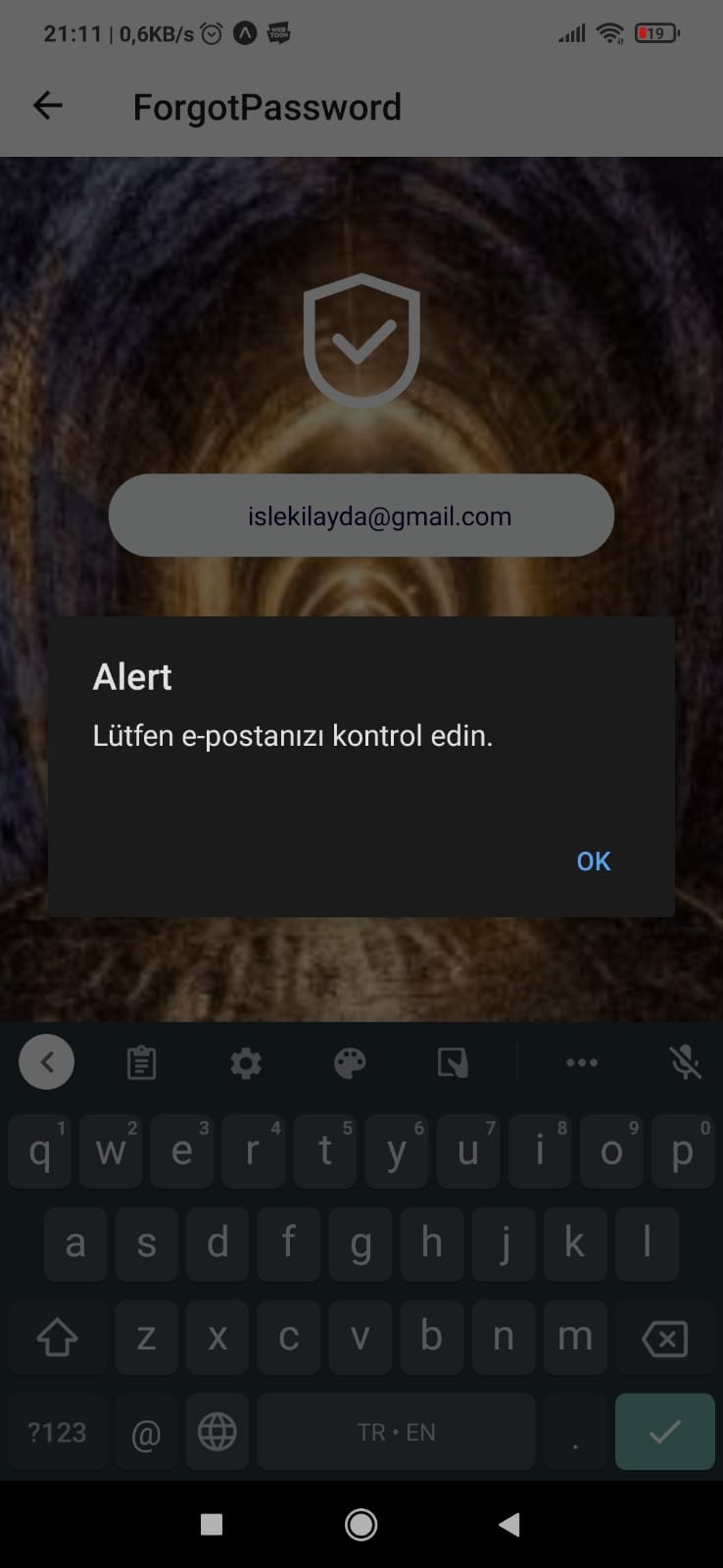
**Şekil:46: Mobil Arayüzü: Başarısız Kullanıcı Giriş Ekranı**

Kullanıcı uygulamaya giriş yaparken kullanıcı adı veya şifre bilgisini eksik veya hatalı olarak girerse kullanıcıya bir hata mesajı gösterilir.



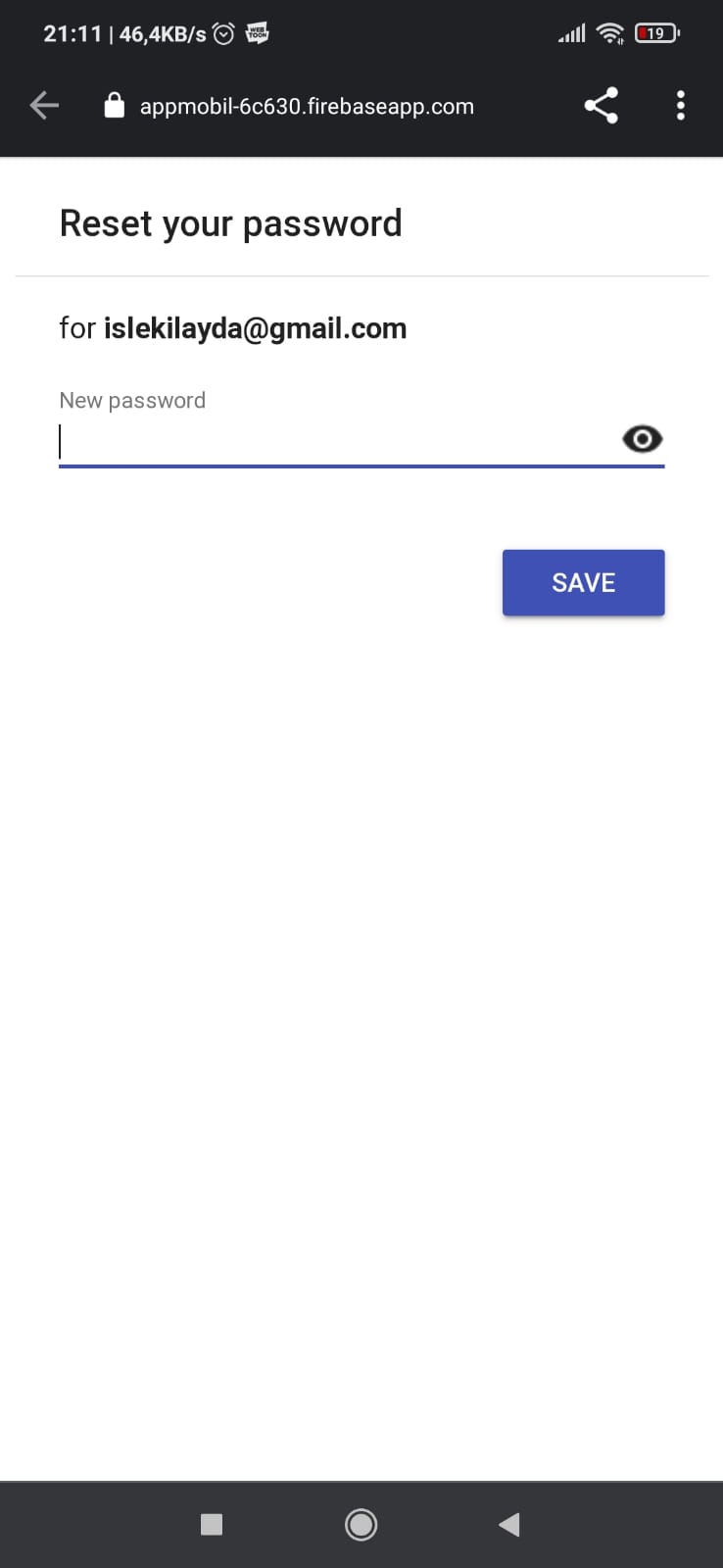
**Şekil:47: Mobil Arayüzü: Şifre Yenileme Ekranı**

Kullanıcının hesap bilgilerini unutmasıyla kullanıcı giriş ekranındaki şifremi unuttum butonuna tıklayarak şifre yenileme ekranına ulaşır. Kullanıcı e-mail adresine gelen şifre bilgisi ile sisteme tekrar giriş sağlar.



**Şekil:48: Mobil Arayüzü: Yeni Şifre İçin Yönlendirme Ekranı**

Kullanıcının şifremi unuttum butonuna tıklanmasıyla kullanıcı e postasına yeni şifre belirlemesi için bir link gönderilir.



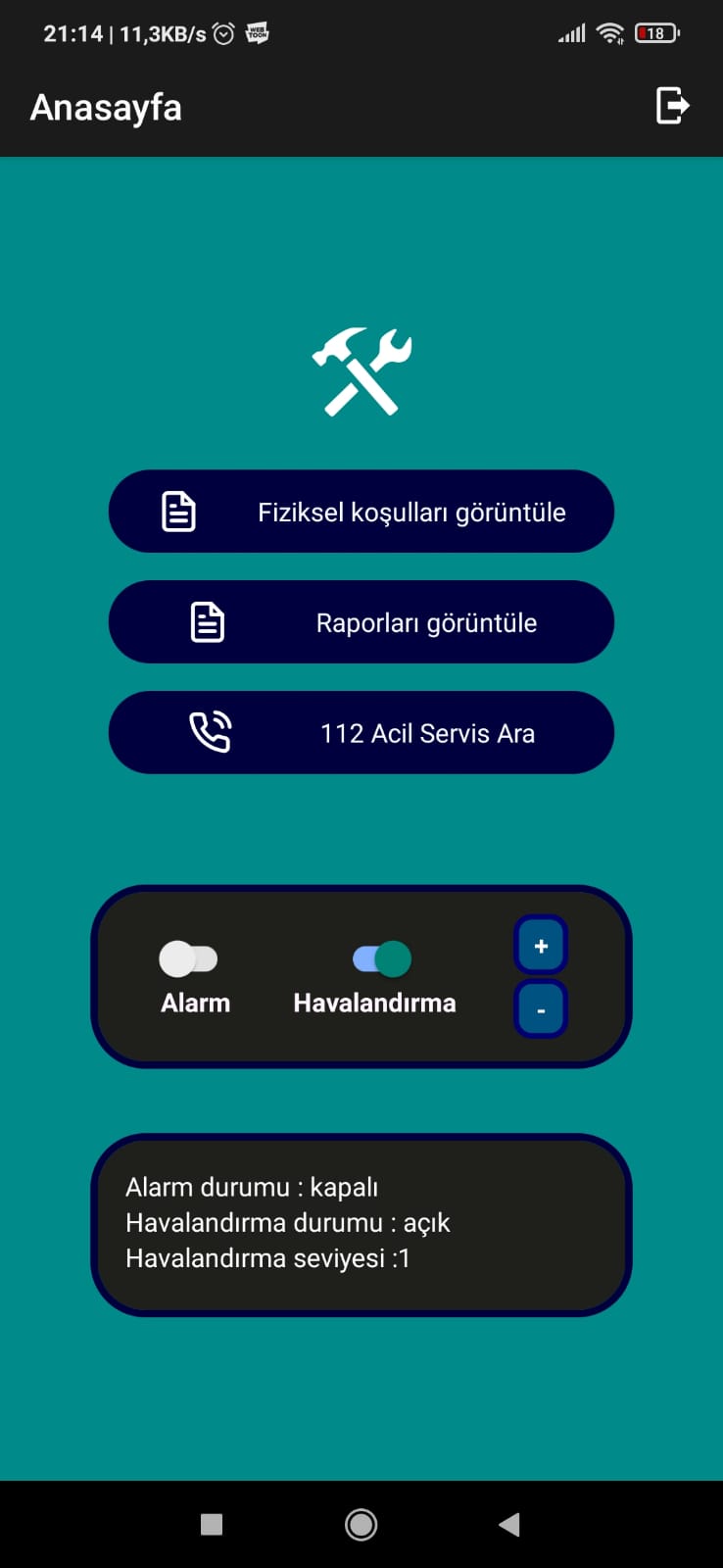
**Şekil:49: Mobil Arayüzü: Yeni Şifre Belirleme Ekranı**

Kullanıcı gönderilen maildeki linke tıklayarak yeni şifresini belirler.



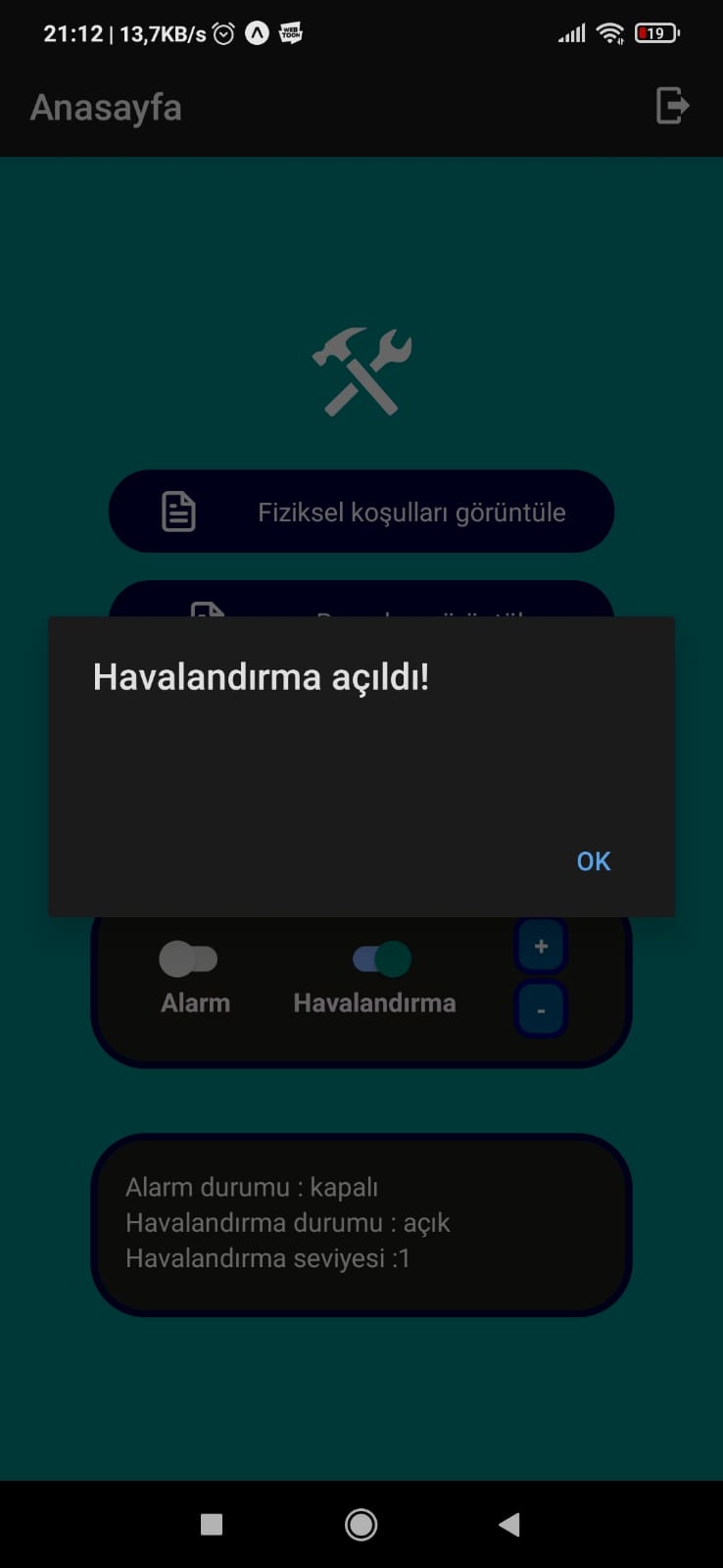
**Şekil:50: Mobil Arayüzü: Başarılı Şifre Yenileme Ekranı**

Kullanıcı e-mail adresine gelen link ile yeni şifresini belirleyerek uygulamaya kendi belirlediği şifre ile tekrar giriş sağlayacaktır.



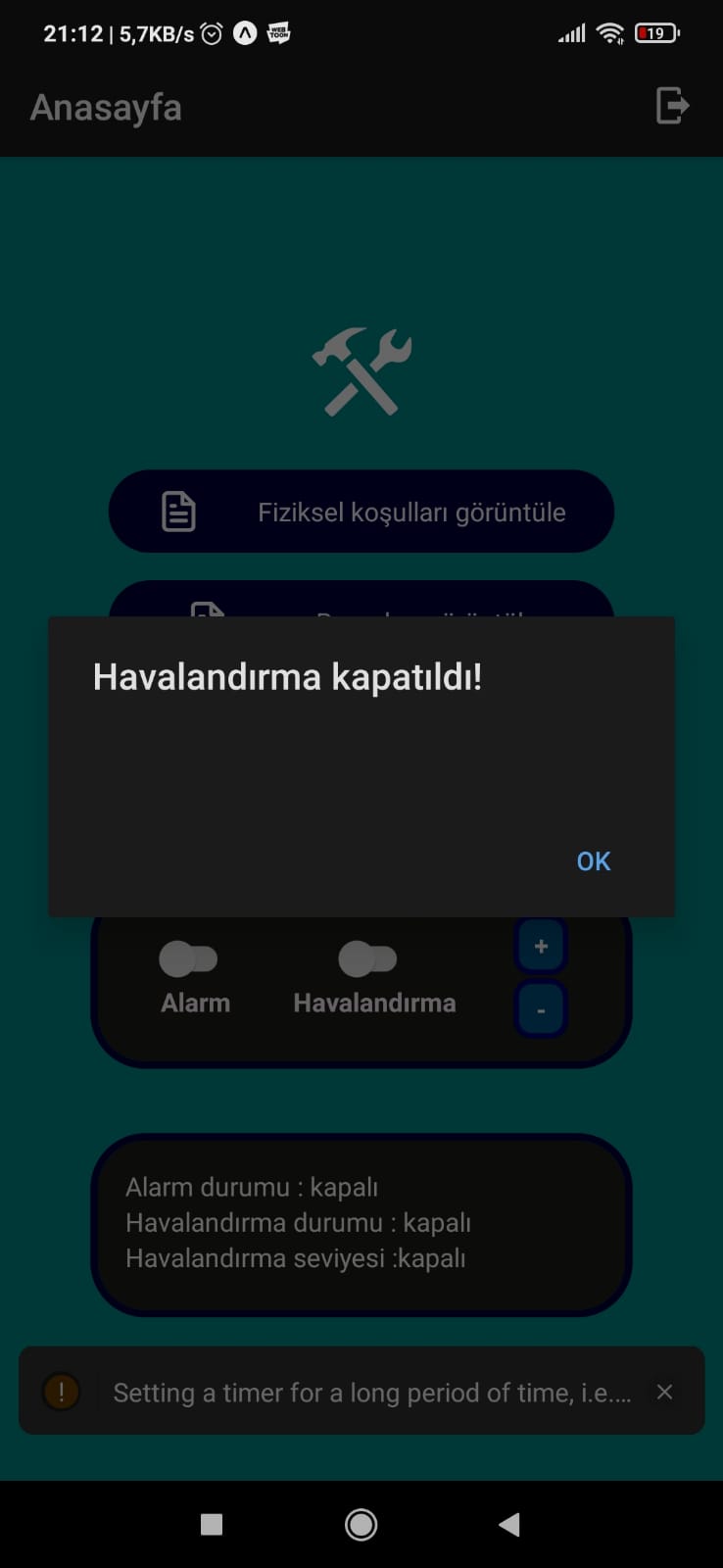
**Şekil 51: Mobil Arayüzü: Kullanıcı Arayüzü**

Kullanıcı uygulamaya giriş yaptıktan sonra arayüze erişir ve arayüz üzerinden fiziksel koşulları görüntüleme, periyodik rapor görüntüleme, alarm ve havalandırmayı açma/kapatma, havalandırma seviyesini artırıp/azaltma ve 112 Acil Servis Arama işlemlerini gerçekleştirir**.**



**Şekil 52: Mobil Arayüzü: Havalandırma Kontrol ve Geribildirim Ekranı**

Kullanıcı düşük seviyeli tehlike yaratabilecek durumları fark etmesiyle geçiş anahtarına tıklayarak havalandırmayı etkinleştirir. Kullanıcının havalandırma sistemini etkinleştirdiğine dair kullanıcıya geribildirim gönderilecektir



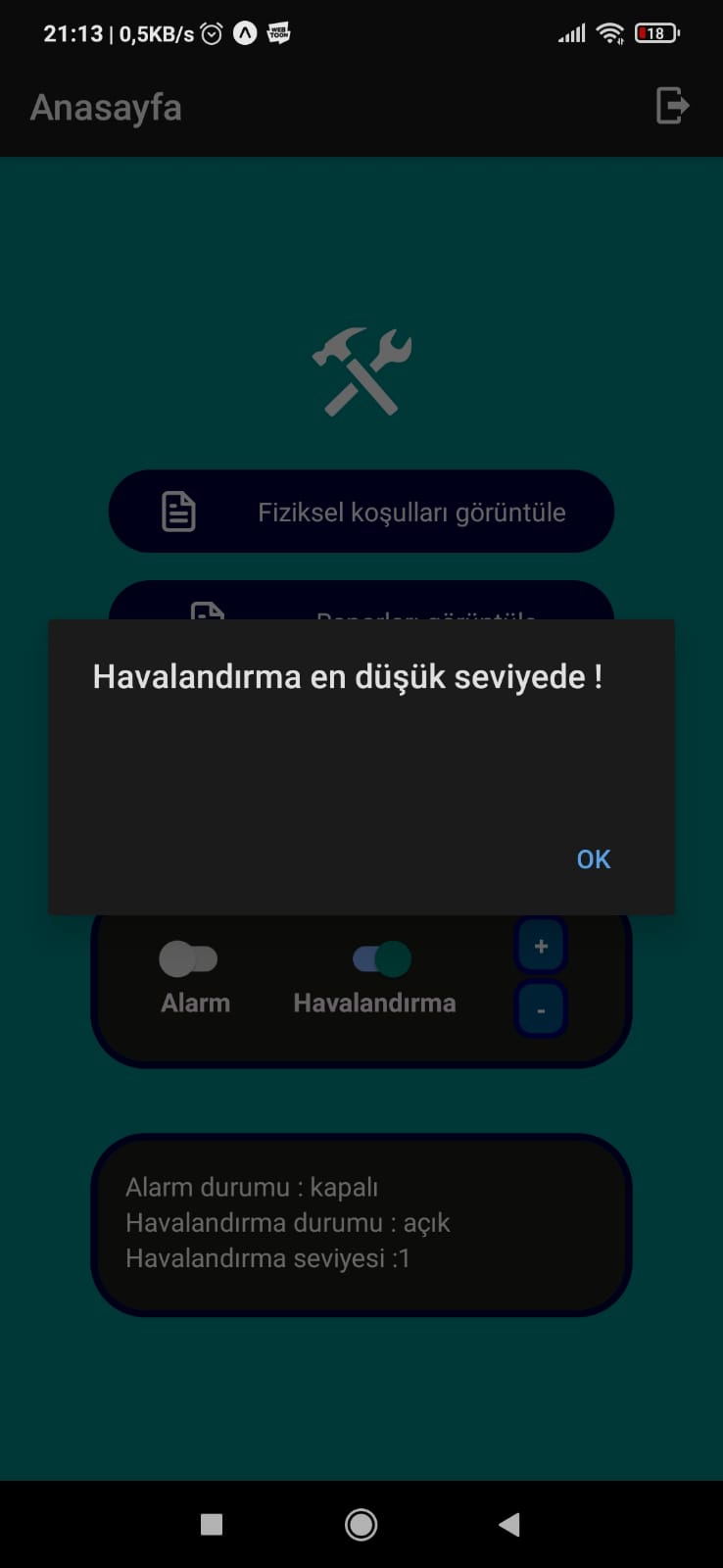
**Şekil 53: Mobil Arayüzü: Havalandırma Kontrol ve Geribildirim Ekranı**

Düşük seviyeli tehlike durumu ortadan kalktığında kullanıcı havalandırmayı tekrar geçiş anahtarına tıklayarak kapatır ve kullanıcıya havalandırmanın kapatıldığına dair geri bildirim gönderilir



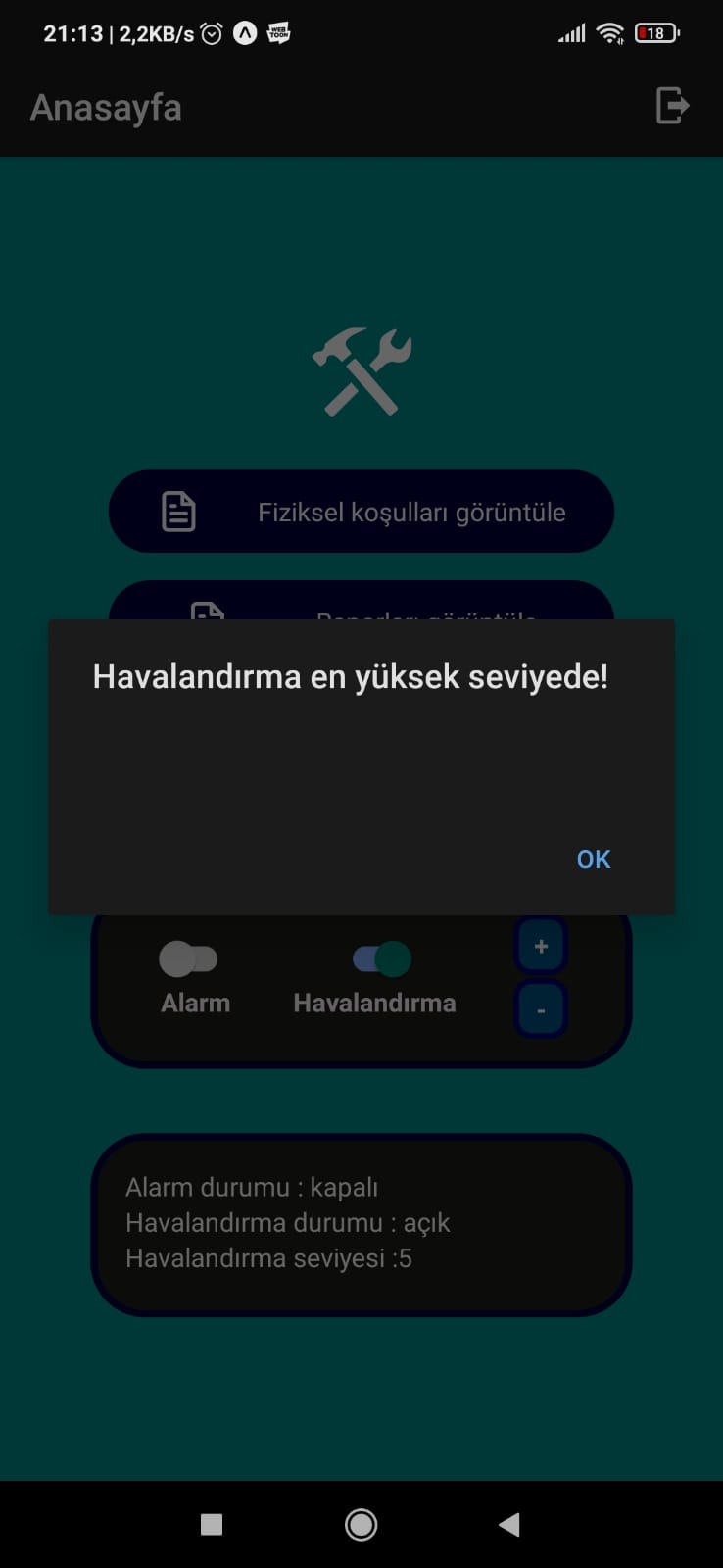
**Şekil 54: Mobil Arayüzü: Alarm Kontrol ve Geribildirim Ekranı**

Kullanıcı ortamda herhangi bir yüksek tehlikenin meydana gelme olasılığına karşı geçiş anahtarına tıklayarak alarmı aktifleştirebilir. Kullanıcıya alarmı etkinleştirdiğine dair geribildirim gönderilir



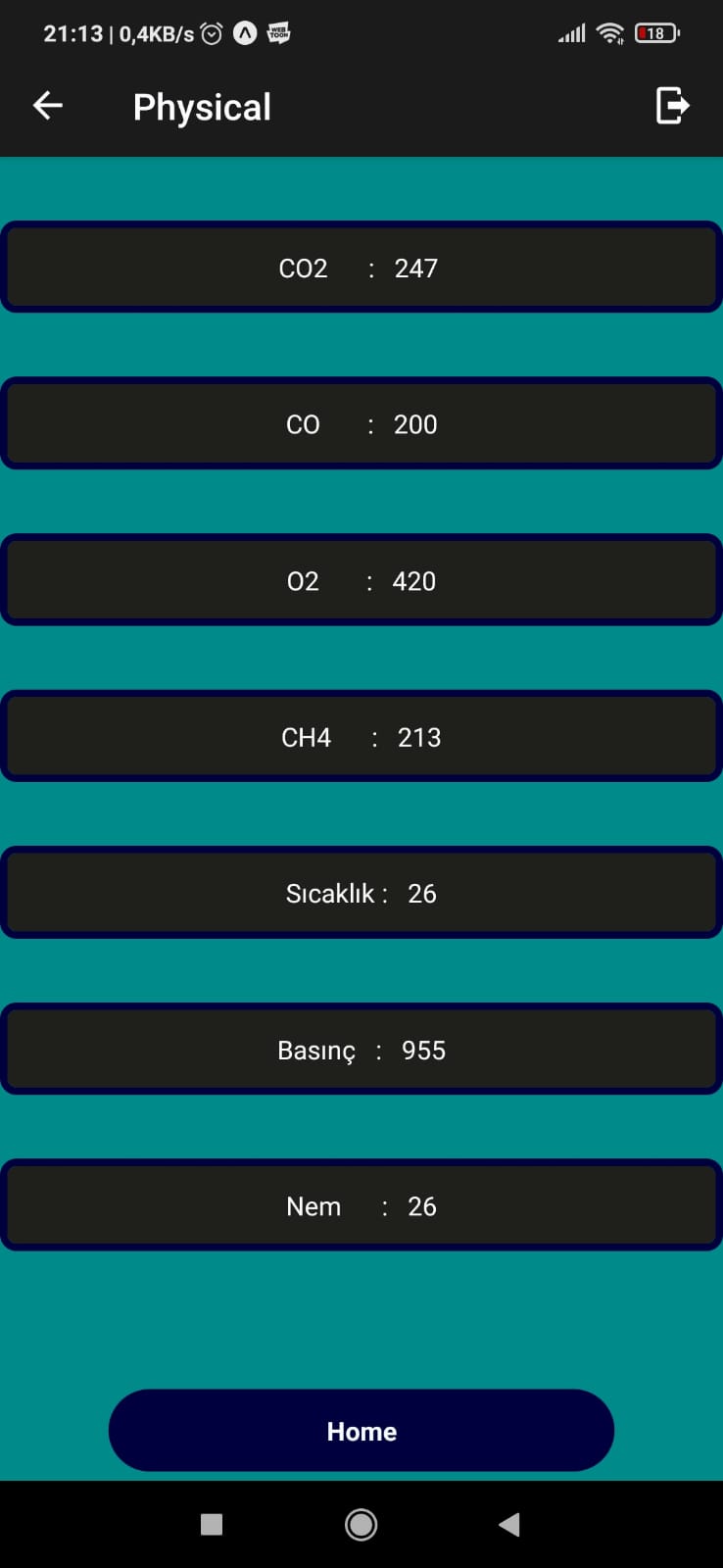
**Şekil 55: Mobil Arayüzü: Havalandırma Seviyesi Kontrol Ekranı**

Kullanıcı, havalandırma açıkken havalandırma seviyesini artırıp azaltabilir. Burada minimum havalandırma seviyesi 1 dir. Havalandırma seviyesi 1 den az olamaz. Havalandırma seviyesi 1 iken havalandırma şiddeti azaltma butonuna tıklandığında kullanıcıya bildirim gösterilir.



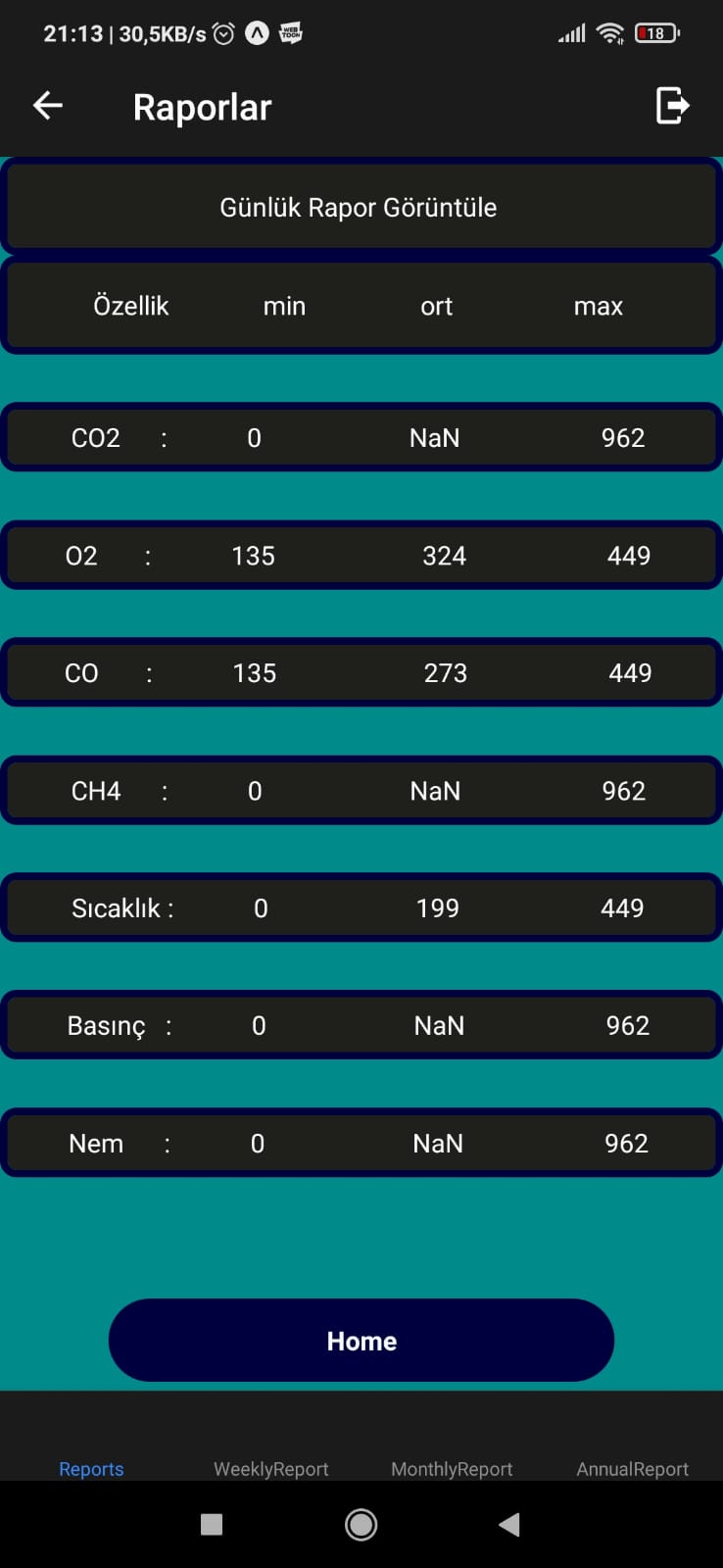
**Şekil 56: Mobil Arayüzü: Havalandırma Seviyesi Kontrol Ekranı**

Kullanıcı, havalandırma açıkken havalandırma seviyesini artırıp azaltabilir. Burada maksimum havalandırma seviyesi 5 tir. Bu maksimum seviyenin aşılması durumunda kullanıcıya bildirim gösterilir.



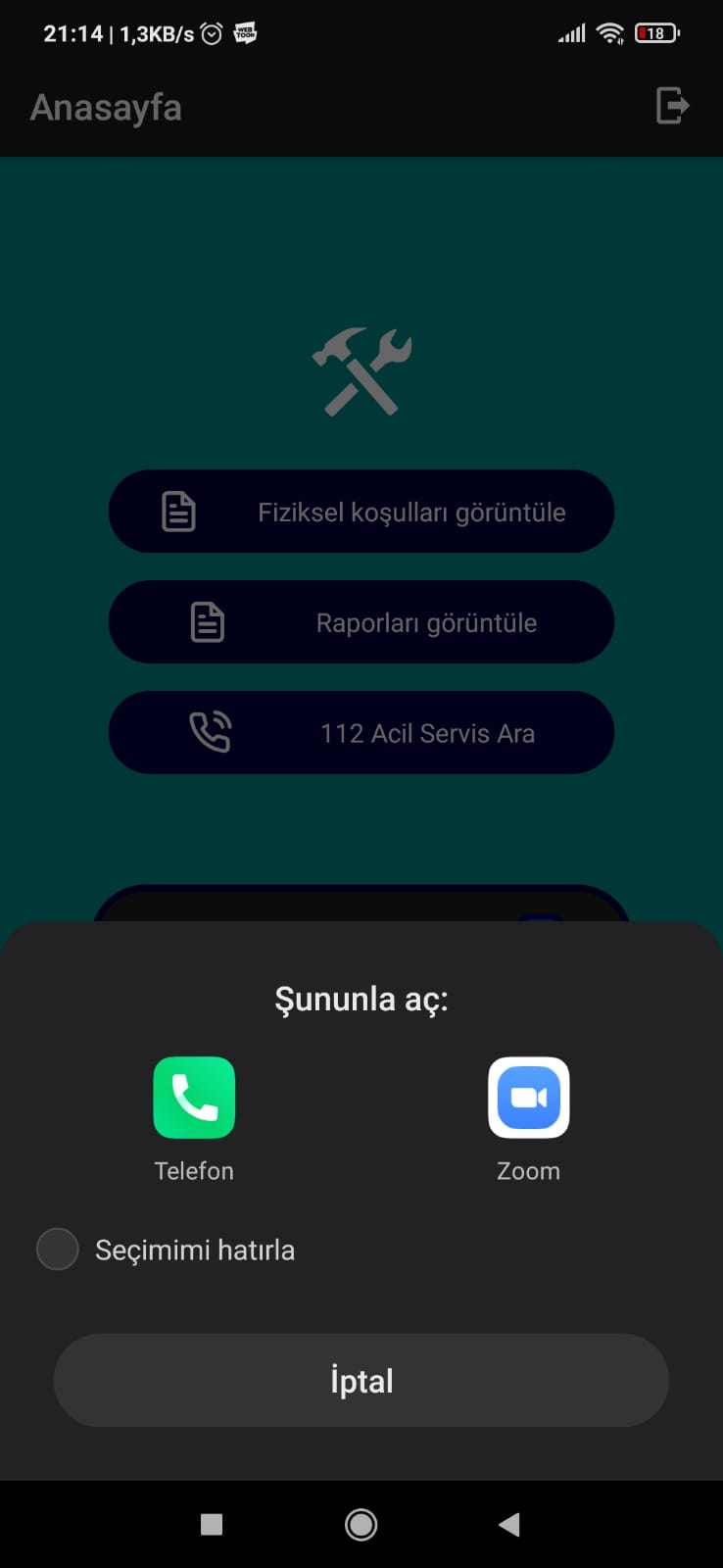
**Şekil 57: Mobil Arayüzü: Ortamın Fiziksel Koşullarını Anlık Olarak Görüntüleme Ekranı**

Kullanıcı arayüzündeki Fiziksel Koşul Görüntüle Butonuna tıklayarak ortamın fiziksel koşulları sorgulayabilir. Kullanıcı ortamın 7 farklı fiziksel özelliğinin ortamda bulunma seviyelerini tek bir ekranda izler.



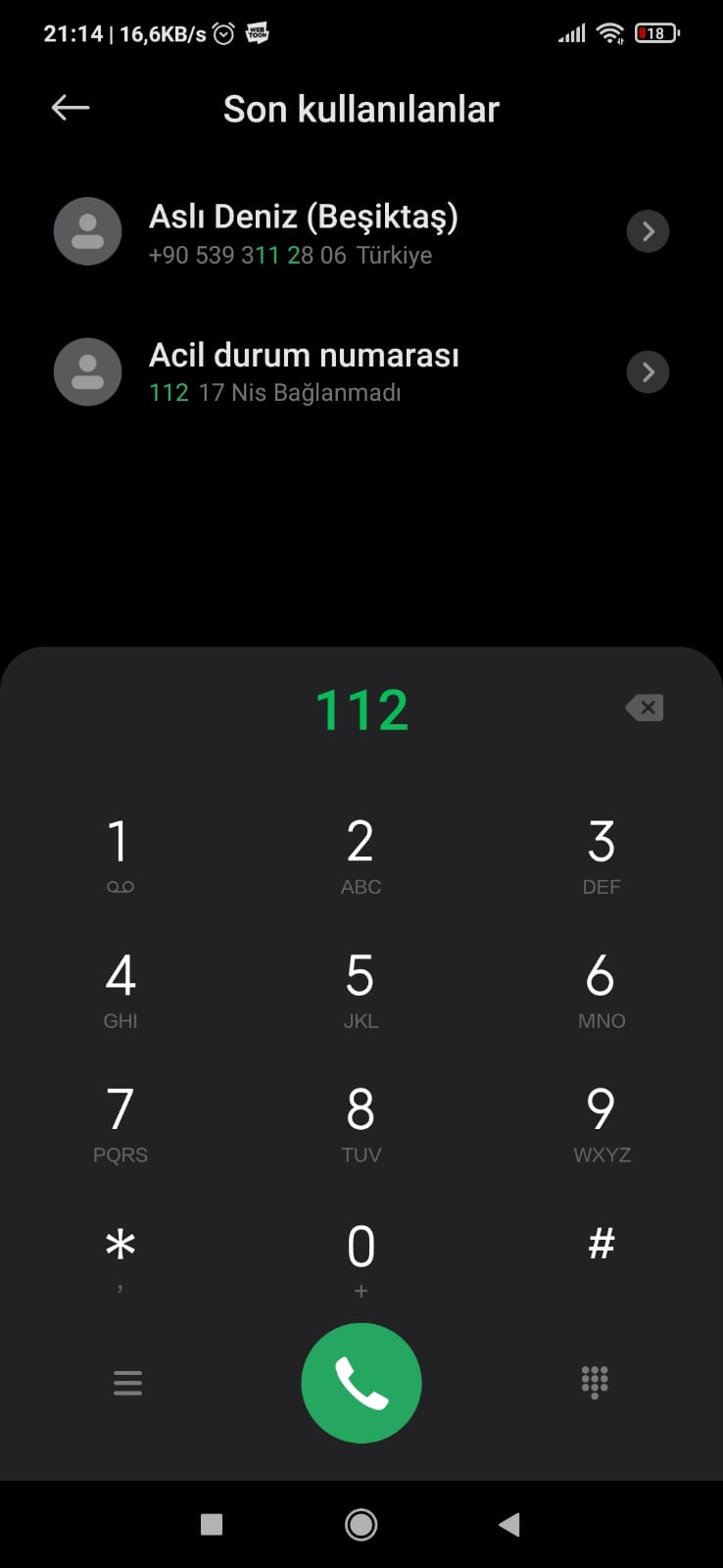
**Şekil 58: Mobil Arayüzü: Periyodik Rapor Görüntüleme Ekranı**

Kullanıcı, periyodik rapor bilgilerini sorguladıktan sonra seçimini yaparak günlük/haftalık/aylık/yıllık periyodik rapor bilgilerine erişebilir. Burada her bir fiziksel koşulun periyodik zaman içerisindeki ortalama , minimum ve maksimum seviyeleri görüntülenir.



**Şekil 59: Mobil Arayüzü: 112 Acil Servis Kontrol Ekranı**

Çalışma ortamında meydana gelen sağlık sorunları ve kazalara karşı kullanıcı 112 Acil Servis butonuna tıklayarak 112’yi arar.



**Şekil 60: Mobil Arayüzü: 112 Acil Servis Kontrol Ekranı**

Kullanıcı, 112 Acil Servis butonuna tıklanmasıyla arama ekranına yönlendirilir.

**SONUÇ**

Uygulamamızı parçadan bütüne giderek geliştirmemiz parçaları kolaylıkla yönetebilmemizi sağlamıştır. Sistemimiz entegre olarak çalıştığı için birçok donanım ve yazılım parçasından oluşmaktadır. Sistemi parçalara ayırmamız oluşabilecek karmaşıklıkların önününe geçmiştir.

Yazılımımız geliştirilmeden önce gereksinimleri karşılayacak şekilde kağıt üzerinde tasarlanmıştır ve bu tasarıma uygun olarak tüm planlanan mantıksal cümlelere , diyagramlara ve kullanım senaryolarına uygun olarak kodlanmıştır.

Sistemimizin en önemli işlevlerinden biri 7 farklı ölçüm değerinin birbiriyle tehlikeli olabilecek seviyelerde karşılaştırılabilecek bir algoritmanın oluşturlması, buna uygun olarak otomatik alarm ve havalandırmanın devreye girmesiydi. Bu yazılım algoritmasını oluştururken hassas veriler üzerinde çalışılmıştır.

Mobil uygulamamızda veri tabanı için Firebase ‘ de kullanılan Realtime Database kullanılmıştır. Bu sayede veriler asenkron olarak alınmaktadır. Kullanıcı arayüzlerini geliştirmek için React açık kaynak kodlu javascript kütüphanesi başarılı bir şekilde kullanılmıştır.

Sistemimiz maliyet ve zamandan tasarruf sağlar.​

Çalışmamızda prototip olarak sensörler, Rpi ve Arduiono microişlemcisi birbiri ile iletişim kurarak ortamdaki fiziksel koşulları ölçüp, karşılaştırdıktan sonra rapor haline getirmektedir.​

Sistemimizdeki yazılım ürünleri ve düzenekler geliştirilerek , daha hassas bir duruma getirilerek , farklı alan ve ortamlarda kullanılabilir.​

Yazılımımızı geliştirirken yazılım yaşam döngüsünden olan planlama, analiz, tasarım, test, gerçekleştirme, test aşamaları başarılı olarak gerçekleştirilmiştir. Bu çalışma sayesinde madencilikte güvenliği sağlayan ilk mobil uygulama geliştirilmiş oldu. Uygulama sonucunda hedeflenen ve planan tüm aşamalar başarılı bir şekilde gerçekleştirilmiştir. Giriş arayüzü, anasayfa arayüzü, fiziksel koşul görüntüleme arayüzü ve raporlar arayüzü başarılı bir şekilde çalışmaktadır. Uygulama arayüzüne getirilen veriler hedeflenen bir şekilde ve doğru olarak back-end den front end tarafında görüntülenmektedir.

Geliştirdiğimiz mobil uygulama insan güvenliğini sağlamaya yönelik olduğu için sensörler tarafından yapılan ölçümlerde hassas veriler üzerinde çalışılmıştır ve bu hassas verilerin doğru seviyeleri konusunda zorlanılmıştır.

Mobil uygulamamız geliştirilmeye açık bir sistemdir. Sistemimiz dinamik bir yapıda oluşturulmuştur. İleride daha büyük sistemler için kullanılabilecektir. Aynı zamanda başka sistemler ile birleştirilerek daha büyük ve karmaşık sistemlerde kullanılabilecektir.

Gelecek hedeflerimiz arasında mobil uygulamamızın bütün madencilik şirketlerinde kullanılabilmesi vardır. Aynı zamanda projemizi daha fazla geliştirerek yeni özellikler eklemeyi düşünüyoruz. Bu özellikler de yine madende çalışan insanların sağlığını ve can güvenliğini korumaya yönelik yazılımsal özellikler olacaktır. Uygulamaya madencilerin konumlarını bulma özelliği getirilebilir. Tabi bu özelliği getirirken yerin çok derinliklerinde çalışan madencileri düşünüldüğünde farklı yöntemler ve yine IoT teknolojisi ile birlikte yapılması gerekmektedir.

**iç mekan, yer içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Şekil 61: Maden Güvenlik Sistemi Prototip**

MGS ,madencilikte oluşabilecek her türlü patlama ve gaz zehirlenmesine  karşı geliştirildi.

Sensörlerin gaz ölçmesine ve ölçtüğü bilgilerin iletilmesine yönelik olduğu için IoT teknolojisi başarı ile kullanıldı.

Çalışmalarımızda madencilikte oluşan kazalardaki patlama ve zehirlenmeye yol açan gerçek gaz seviyeleri kullanıldı.

Mobil uygulama ile taşınma ve kullanım kolaylığı sağlandı.

Sistemimiz ortamdaki tehlikeli gaz seviyelerini ölçüp karşılaştırdıktan sonra rapor haline getirir. Raporlama özelliği sayesinde olası tehlikeli durumlar kolaylıkla tespit edilir.

Projemiz insan hayatını korumaya yönelik olduğu için kullandığımız yazılımın devamlı güncel kalabilecek şekilde geliştirilmeye özen gösterildi.

Sistemimizin, tasarım aşamasında planlanan tüm  aşamaları gerçekleştirildi ve planlanan seviyede hedefe ulaşıldı.

**KAYNAKLAR**

* React Native Dökümanı: <https://tr.reactjs.org/docs/getting-started.html>

• React Native Navigation Dökümanı : <https://reactnavigation.org/docs/getting-started>

* React Native Stack Navigation Dökümanı : <https://reactnavigation.org/docs/stack-navigator/>
* React Native Top Tab Navigation Dökümanı: <https://reactnavigation.org/docs/material-top-tab-navigator/>
* React Native Bottom Navigation Dökümanı : <https://reactnavigation.org/docs/bottom-tab-navigator/>
* React Native Vector Icon : <https://oblador.github.io/react-native-vector-icons/>
* Mobx Dökümanı : <https://mobx.js.org/README.html>
* React Native Elements Dökümanı : <https://react-native-elements.github.io/react-native-elements/>
* React Native ImagePicker Dökümanı : <https://github.com/react-native-community/react-native-image-picker>
* React Native Modal Dökümanı : <https://reactnative.dev/docs/modal>
* React Native Dökümanı : <https://medium.com/bili%C5%9Fim-hareketi/react-native-mobil-uygulama-geli%C5%9Ftirme->
* React Native Dökümanı : <https://dev.to/washingtonsteven/reacts-new-context-api-and-actions-446o>
* React Native Context Dökümanı : <https://dev.to/emeka/state-management-in-react-native-using-context-14no>
* React Native Tutorials : <https://www.digitalocean.com/community/tutorials/build-mobile-friendly-web-apps-with-react-native-web>
* React Native Tutorials: <https://www.simplilearn.com/react-native-tutorial-article>
* Arduino Dökümanı: <https://www.arduino.cc/en/main/docs>
* RPI Dökümanı: <https://www.raspberrypi.org/documentation/>
* Arduino Tutorial: <https://learn.adafruit.com/lesson-0-getting-started/the-lessons>

<https://www.makerspaces.com/arduino-uno-tutorial-beginners/>

* RPI Tutorial: <https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/introduction.html>