T.C.

Fırat Üniversitesi

Yazılım Mühendisliği Bölümü

|  |
| --- |
| Kalıcı Bitkisel Durumdaki (PVS) Hastaların Ev Bakımında Hayati Bulgularının Takibi Ve Hasta Bakıcılarına Uyarı Sistemi |
| Görkem GÖK |
| HAZİRAN - 2021 |

T.C.

Fırat Üniversitesi

Yazılım Mühendisliği Bölümü

|  |  |
| --- | --- |
| Başlığı: | Kalıcı Bitkisel Durumdaki (PVS) Hastaların Ev Bakımında Hayati Bulgularının Takibi Ve Hasta Bakıcılarına Uyarı Sistemi |
| Yazarı: | Görkem GÖK |
| Proje Danışmanı: | Özgür KARADUMAN |
| Teslim Tarihi: | 21.06.2021 |

Beyan

Fırat Üniversitesi Yazılım Mühendisliği Bölümü bitirme projesi yazım kurallarına uygun olarak hazırladığım “Kalıcı Bitkisel Durumdaki (PVS) Hastaların Ev Bakımında Hayati Bulgularının Takibi Ve Hasta Bakıcılarına Uyarı Sistemi” Başlıklı proje dokümanımın içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

21.06.2021

Görkem GÖK

**Önsöz**

Gerçekleştirilen çalışmanın genel amacı, bazı sebepler nedeniyle insanların kalıcı bitkisel duruma geçmesi ve bu durumda bilişsel fonksiyonlarını yerine getirememesinden doğmuştur. Bu tarz hastaların kalan ömürleri çok uzun olmamakla birlikte, hastalıkla karşılaştıktan sonra hayatını yatağa bağımlı geçirmeye devam ederler.

Hastaların bakımını üstlenen hasta yakınlarının, hayatlarına devam ederken aynı zamanda uzaktan hastanın durumunu izleyebilmesi hedeflenmektedir. Hastanın vital (hayati) bulgularının takibi çok önemlidir. Bu bulgular hastanın yaşam belirtileridir ve yaşam standardını etkileyecek durumdadır.

Günümüz hayatında gelişen teknolojiler ve algoritmalar sayesinde en ucuz donanımlar ile dış dünyadaki olayları denetleyebilmekteyiz. Bu tez çalışmasında uygun donanımlar ve yazılımların bir arada kullanılmasıyla hastanın vital bulgularının bulunması ve takip edilmesi gerçekleştirilmiştir.

Çalışmalarımda Kalıcı Bitkisel Durumdaki hastalar hakkında bilgi veren Fırat Üniversitesi Yazılım Mühendisliği öğrencisi Elif Bahar ÖZDOĞRU’ya sonsuz teşekkürlerimi iletiyorum. Ayrıca projeyi geliştirmemde yardımcı olan, her adımda fikirlerini esirgemeyen Sayın Dr. Öğr. Üyesi Özgür KARADUMAN hocamıza teşekkürlerimi iletiyorum.

Görkem GÖK

Elazığ, 2021

İçindekiler

Sayfa

[ÖNSÖZ iv](#_Toc75542562)

[İÇİNDEKİLER v](#_Toc75542563)

[ÖZET vi](#_Toc75542564)

[ŞEKİLLER LİSTESİ vii](#_Toc75542565)

[TABLOLAR LİSTESİ viii](#_Toc75542566)

[KISALTMALAR LİSTESİ ix](#_Toc75542567)

[1. GİRİŞ 1](#_Toc75542568)

[2. PVS HASTALARI 2](#_Toc75542569)

[3. VİTAL BULGULAR VE VİTAL BULGULARIN BULUNMASI 2](#_Toc75542570)

[4. HASTA DURUMUNUN TAKİP EDİLMESİ 4](#_Toc75542571)

[5. ARDUİNO EKOSİSTEMİ 4](#_Toc75542572)

[6. RASPBERRY Pİ 4 8](#_Toc75542573)

[7. YAPAY SİNİR AĞLARI 9](#_Toc75542574)

[8. ARDUİNO İLE ANDROİD HABERLEŞMESİ 13](#_Toc75542575)

[9. MATERYAL VE METOT 14](#_Toc75542576)

[10. ÇALIŞAN SİSTEMİN TEST EDİLMESİ 33](#_Toc75542577)

[11. BULGULAR 37](#_Toc75542578)

[12. SONUÇLAR 39](#_Toc75542579)

[KAYNAKLAR 39](#_Toc75542580)

Özet

Dummy

Kalıcı Bitkisel Durumdaki (PVS) Hastaların Ev Bakımında Hayati Bulgularının Takibi ve Hasta Bakıcılarına Uyarı Sistemi

Görkem GÖK

Fırat Üniversitesi

Yazılım Mühendisliği Bölümü

Kendi ihtiyaçlarını karşılayamayan, içinde bulunduğu durumun farkında olmayan ve bakımına ihtiyaç duyulan insanların hayati bulguları, pahalı ve büyük cihazlarla yapılmaktadır. Vital (hayati) bulgular bir insanın yaşam belirtileridir. Bunlar satürasyon (kandaki oksijen oranı), nabız, vücut sıcaklığı, tansiyon ve solunum sayısıdır. Hasta monitörü vb. cihazlar ile vital bulgu takibi yapılmaktadır. Bu cihazların kullanımı esnasında olası önemli durumları öğrenebilmek için oda veya ev içerisinde bulunmak gerekmektedir.

PVS hastaları yani Kalıcı Bitkisel Durumdaki hastalar komadan farklı olarak uyanıktır fakat bilişsel fonksiyonlarını kaybetmişlerdir. Bilişsel durumu yerinde olmayan hastalar sağlık durumlarına tepki verememektedir. Nefessiz kalma, nabzındaki ani değişimler ve vücut sıcaklığının değişiminin farkında değillerdir. Hasta kişi, bu tarz durumları da yakınlarına iletebilecek durumda değildir. Bu hastaların verilerinin sürekli izlenmesi ve gerekli durumlarda müdahale edilmesi gerekmektedir.

PVS Hastalarının bakımında özel bakıcı tutmak hem maliyet hem de görevlerinin yerine getirip getirememe gibi durumlarda hasta yakınlarında endişe oluşturmaktadır. PVS hastalarına bakma durumunda olan insanların sosyopsikolojik durumlarını bir nebze rahatlatmak, evde işte veya farklı bir yerde hastasının durumu hakkında bilgilendirmek bu tez çalışmasının amaçları arasındadır. Günümüzde hayatında akıllı cihazlar ile birlikte uzaktan iletişim kolaylaşmakta ve yaygınlaşmaktadır. Bu cihazları birbirleri arasında haberleştirerek verileri gönderip okuyabilmekteyiz.

Çalışmanın temel amacı PVS hastalarının evden uzak ortamlarda takibi ve gerekli zamanlarda müdahale edilmesidir. Bir mikrodenetleyici olarak hayatımızda yer alan Arduino kullanarak oluşturulan sensörlerden okunan değerleri, mini bilgisayar olarak nitelendirebileceğimiz Raspberry Pi ’ye göndererek bir yapay sinir ağı modelinde kontrolü sayesinde hasta yakınlarına bluetooth bağlantı üzerinden Android uygulama ile bilgi verilmesi hedeflenmiştir.

**Anahtar Kelimeler:** PVS hastaları, Yapay Sinir Ağı, Raspberry Pi, Bluetooth, Android, Sensör.

Şekiller Listesi

Sayfa

[Şekil 1 Sistem Blok Diyagramı 4](#_Toc75542586)

[Şekil 2 Arduino Uno’nun görüntüsü 5](#_Toc75542587)

[Şekil 3 Arduino IDE görüntüsü 6](#_Toc75542588)

[Şekil 4 Fototransistör türleri 7](#_Toc75542589)

[Şekil 5 MCP9808 Yüksek Hassasiyetli Sıcaklık Sensörü 7](#_Toc75542590)

[Şekil 6 Raspberry Pi 4 8](#_Toc75542591)

[Şekil 7 Yapay Sinir Ağı Katman Görüntüsü 10](#_Toc75542592)

[Şekil 8 Gerçekleştirilecek Sistemin Blok Şeması 14](#_Toc75542593)

[Şekil 9 Sensörün Arduino’ya Bağlantı Şeması 15](#_Toc75542594)

[Şekil 10 Sıcaklık Sensörünün Arduino Bağlantısı 16](#_Toc75542595)

[Şekil 11 HC-06 Bluetooth Modülünün Arduino Bağlantısı 16](#_Toc75542596)

[Şekil 12 Pulse Oksimetre Görüntüsü 17](#_Toc75542597)

[Şekil 13 Oluşturulan Modelin Özeti 24](#_Toc75542598)

[Şekil 14 Android Uygulama Ana Ekran Tasarımı 28](#_Toc75542599)

[Şekil 15 Android Uygulama Veri Ekranı Tasarımı 29](#_Toc75542600)

[Şekil 16 Hazırlanan Nabız ve Satürasyon Sensörü 34](#_Toc75542601)

[Şekil 17 Sistemin Tamamlanmış Hali 34](#_Toc75542602)

[Şekil 18 Bluetooth Açma İsteği 35](#_Toc75542603)

[Şekil 19 Eşlenmiş Cihaz Listesi 35](#_Toc75542604)

[Şekil 20 Veri Bekleniyor Ekranı 36](#_Toc75542605)

[Şekil 21 Başarı Oranı Yüksek Verilerde Panel Görüntüsü 36](#_Toc75542606)

[Şekil 22 Kontrol Gerekli Olan Hasta Veri Paneli 37](#_Toc75542607)

[Şekil 23 Seri Çizicide Nabız Atımları 38](#_Toc75542608)

[Şekil 24 YSA Başarı Oranının Görüntülenmesi 38](#_Toc75542609)

Tablolar Listesi

Sayfa

[Tablo 1 Veri Setinin Görüntüsü 23](#_Toc75542610)

Kısaltmalar Listesi

**Kısaltmalar**

YSA : Yapay Sinir Ağı

API : Uygulama Programlama Arayüzü

IDE : Tümleşik geliştirme ortamı

BLE : Düşük Enerji Bluetooth

SpO2 : Satürasyon

Bpm : Dakikadaki nabız sayısı

PVS : Persistent vegetative state (Kalıcı bitkisel durum)

1. Giriş

Bir hastalık türü olan PVS (Kalıcı Bitkisel Durum), komadan farklı olarak hasta bireyin uyanık olduğu durum olarak nitelendirilebilir. Hasta gün içerisinde uyku halinde veya uyanık olabilir. Dış dünya ile iletişimi yoktur. Titreme gibi durumlar gözlenebilir. Bu tip hastalar hayati bulgularını dış dünyaya iletemezler ve takip edilmesi gerekir. Hayati bulgular olarak hastada önemli olan veriler satürasyon, nabız ve vücut sıcaklığıdır. Bunların takip edilmesi sensörler vasıtası ile gerçekleştirilebilir.

Sensörler dış dünyayı gözetlememizi sağlayan araçlardır. Dış dünyadan algıladığı verileri işleyerek bize bilgi olarak verir. Örnek verecek olursak bir kızılötesi termometre, insanların veya nesnelerin yaydığı kızılötesi dalgaları algılayarak bize o alan hakkında sıcaklık değerini verir. Bu çalışmada ise piyasada sıklıkla bulunan Arduino Parmak Nabız Ölçer sensörünün modifiye edilmiş haliyle nabız ve satürasyon verileri hastadan alınmaktadır. Aynı zamanda vücut sıcaklığı için MCP9808 sıcaklık sensöründen yüksek çözünürlükte ve hassasiyette sıcaklık verisi alınmaktadır.

Bu sensörlerin algoritması ve verilerin hesaplanması Arduino Mega üzerinden yapılmıştır. Arduino Mega bir mikrodenetleyici olup, sensörlerden alınan veriler ile işlem yapabilme yeteneğine sahiptir.

İşlenmiş veriler daha sonradan USB haberleşme ile Raspberry Pi ‘ye gönderilir. Keras, Tensorflow, Pandas ve Scikit-Learn kütüphaneleri kullanarak önceden eğitilmiş model üzerinde başarısı test edilmektedir. Bu başarı oranı ile hastanın sağlıklı verilere sahip olup olmadığına karar verilmektedir.

Sonuç Raspberry Pi üzerinden tekrar Arduino ’ya gönderilerek burada HC-06 Bluetooth Modülü Android cihazda gerçekleştirmiş olduğum programa aktarılmaktadır. Bu program Android’in bluetooth izinlerini kullanarak seri haberleşme gerçekleştirerek çalışmaktadır. Bu kısımda hasta yakınına bütün bilgiler verilmektedir ve aynı zamanda olası durumlarda ekranda uyarı oluşturmaktadır.

1. PVS Hastaları

PVS yani Türkçe karşılığı Kalıcı Bitkisel Durum olan bu hastalık türü genelde bir travma, kaza, beynin metabolik bozukluğu vb. problemlerin sonucunda meydana gelmektedir. Projede hedeflediğimiz bu hastalık türünde hasta, bilişsel fonksiyonlarından yoksundur. Bu hastalar yemek-içme, ihtiyaçlarını giderme veya vücudunda gerçekleşen olayların farkında değillerdir.

* 1. Genel Tanım

Bitkisel durum, kronik veya uzun vadeli bir durumdur. Koma ile aynı gibi gözükse de birbirlerinden ayrı incelenmektedir. Koma halinde olan bir insan hem bilişsel fonksiyonlardan hem de uyanıklıktan yoksun bir durumdadır. Bitkisel durumdaki hastalar koma durumundan çıkmış olabilir, ancak yine de bilişsel fonksiyonlarını geri kazanamamışlardır. Bitkisel durumda hastalar ara sıra göz kapaklarını açabilir ve uyku-uyanma döngüleri sergileyebilir, ancak bilişsel işlevden tamamen yoksundur. Bitkisel durumda geçirilen zaman arttıkça, farkındalığın yeniden kazanma şansı önemli ölçüde azalır.

Kalıcı bitkisel durum, çok sayıda nörolojik ve diğer testlerden sonra yapılan, kapsamlı ve geri döndürülemez beyin hasarı nedeniyle bir hastanın vejetatif(bitkisel) bir durumdan daha yüksek işlevler elde etme olasılığının çok düşük olduğu tıbbi bir teşhis için standart kullanımdır. Bitkisel durumdaki hastanın kalıcı bitkisel durumda kabul edilmesi birkaç ayı bulmaktadır. Amerika için üç ay olan bu süreç Birleşik Krallıkta ise altı ay olarak kabul edilmektedir.[1]

* 1. Çalışmanın Amacı

Gerçekleştirilen çalışmada hedef kitle olarak PVS hastaları düşünülmüştür. Bu hastaların bilişsel fonksiyonlarını olmaması, bakımının dışarıdan bir hemşire veya hasta bakıcı tarafından yapılması gerekmektedir.

1. Vital Bulgular Ve Vital Bulguların Bulunması

Vücut sıcaklığı, nabız, bilinç, tansiyon ve solunum olarak düşünülebilir. Canlıların hayatlarının devam edip etmediğini algılamamıza yarar.

* 1. Genel Tanım

Vital bulgular yani hayati bulgular insanlarda ve canlılarda yaşam durum ve kalitesini belirleyen en temel işlevlerin ölçümleridir. Tıp dünyasında dört ana yaşam belirtisi vardır;

1. Nabız
2. Satürasyon
3. Vücut Sıcaklığı
4. Kan Basıncı (Tansiyon)

Bu belirtileri izlemek belirli cihazlar ile sağlanabilmektedir. Hasta başı monitörü veya satürasyonun ve nabzın bulunması için oksimetre, vücut sıcaklığı için termometre, kan basıncı için tansiyon aleti kullanılabilir. Bu bitirme tezinde hedef olarak belirlediğimiz hasta türünde üç ana yaşam belirtisini kontrol edeceğiz.

* 1. Nabız

Akciğerlerde temizlenip kalbin sol karıncığına dolan kanın, vücuda dağılırken atardamarlara uyguladığı basınca nabız denir. Canlılığın devamı için oksijene ihtiyaç vardır. Oksijenli kanın vücuttaki organlara, dokulara ve hücrelere yeterli miktarda ve zamanında ulaşabilmesi için kan dolaşımının gerçekleşmesi ve kanın devir daim olması gerekmektedir. Nabız sayısı kritik değerin altına düşerse hipoksi, felç gibi durumlar ortaya çıkabilir. Kritik değerin üstünde beyin kanaması, nefes darlığı gibi sonuçlar doğurabilir.

* 1. Satürasyon

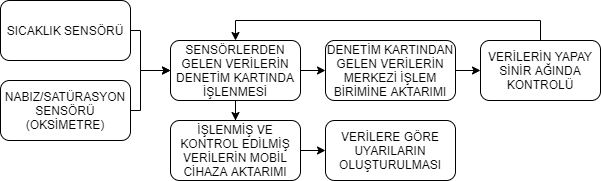
Kandaki oksijen doygunluğuna ulaşan hemoglobini ifade eder. Kanda aynı anda hem oksijenli hemoglobin (oksihemoglobin) hem de oksijensiz hemoglobin (deoksihemoglobin) bulunmaktadır. Oksijenli hemoglobinin tüm hemoglobine oranı satürasyonu verir. Oksijenli solunum yapan tüm canlılar için hayati önem taşımaktadır. İnsanın oksijensiz kalması boğulma, akciğerlerde işlev bozukluğu gibi pek çok durumla ilişkilendirilebilir. Oksijensiz kalma durumu hipoksi olarak adlandırılır ve yüksek ihtimalle bir veya daha fazla organ veya dokunun hasar almasıyla sonuçlanır. Oksijensiz kalma durumunda satürasyonu düşen kan canlılığın devam etmesi gereken hücreleri oksijenle besleyemez. Satürasyonun düşmesi sinir hücreleri gibi rejenerasyonu ve onarımı mümkün olmayan kısımlarda gerçekleşmesi kişilerde kalıcı hasarlara yol açabilir.

* 1. Vücut Sıcaklığı

Her canlı organizmanın yaşamsal faaliyetlerini optimum düzeyde devam ettirebilmesi için belirli sıcaklık aralığı vardır. Metabolizmanın devamı için elzem olan enzimler yalnızca belirli sıcaklıklarda doğru çalışabilirler. İnsan organizması için bu optimum sıcaklık 36.5°C ile 37°C arasındadır. Yüksek ateş proteinlerde bozulmaya, çok düşük vücut sıcaklığı ise hipotermiye sebep olur.

1. Hasta Durumunun Takip Edilmesi

PVS Hastalarının takibi yapılırken sensörlerden, denetleme kartından ve merkezi işlem biriminden yararlanılacaktır. Ayrıca hasta yakının takibi için bir uygulama geliştirilecek ve bu sayede çok kısa süre içerisinde hastasının durumunu gözetleyebilecektir.



Şekil Sistem Blok Diyagramı

Sistem blok diyagramı incelendiğinde projenin taslağına göre kullanılacak iki sensör ile hastaya bağlı olarak çalışılacaktır. Hastanın vücudundaki değişimleri algılamamızı sağlayan bu sensörler ile vital bulgularının takibini yapacağız.

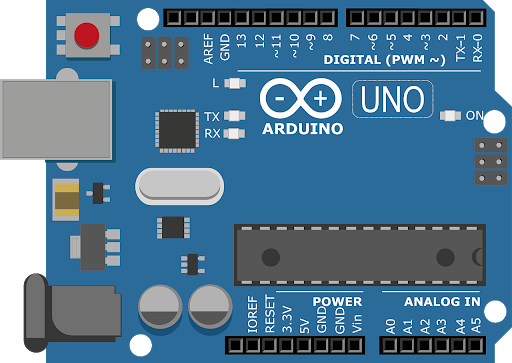
Sensörlerden alınan veriler daha sonra bir mikrodenetleyici üzerinde işlenecektir. Ham verilerden bilgi üretilip bu bilgileri bir mikroişlemciye göndererek daha önceden oluşturulmuş yapay sinir ağında kontrolü sağlanacaktır.

Yapay sinir ağında kontrolü sağlanan verilerin bir başarı oranı oluşmaktadır. Bu başarı oranı ile hastanın durumunu inceleyeceğiz. Başarı oranı elde edildikten sonra mikrodenetleyiciye gönderilmektedir. Mikrodenetleyicide hâlihazırda olan sensörlerden alınarak işlenmiş veriler ile başarı oranı mobil cihaza aktarılır. Bu aktarım kablosuz iletişim teknolojilerinden olan bluetooth ile gerçekleştirilmektedir.

Bluetooth ile alınan veriler bir mobil cihaza aktarılır. Aktarılan verilerin görüntülenmesi ve görüntülenirken oluşacak uyarıların oluşturulması burada gerçekleşecektir. Başarı oranına göre uyarılar oluşturulup hasta yakını bilgilendirilmiş olunur.

1. Arduino Ekosistemi

En basit tabiri ile donanımları kontrol edebilen cihazlardır. Üzerlerinde bulunan giriş çıkış pinleri sayesinde, doğru bağlanmış sensörler, modüller vb. cihazları kontrol etmeye, dış dünyayı gözetlemeye yarayan girdi ve çıktı üretebilen kartlardır.



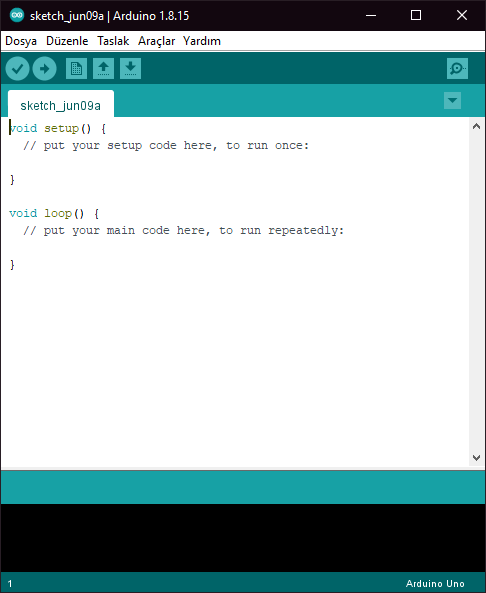
Şekil Arduino Uno’nun görüntüsü

* 1. Genel Tanım

Arduino, dünya çapında ismi çok sık duyulmaya başlanan, daha ilkokuldan çocuklara eğitimi verilen, ülkemizde de TÜBİTAK tarafından projeleri desteklenen, elektronik cihazları yönetmeye ve çözümlemeye yarayan açık kaynaklı bir sistemdir. Üzerinde bulunan Atmel marka mikrodenetleyicisi sayesinde sensörler DC motorlar, solenoidler, step motorlar, LCD ekranlar ve LED’leri kontrol etmek için kullanılabilmektedir. Kartta bulunan USB Seri dönüştürücüsü sayesinde üzerinde yer alan ve beyni sayabileceğimiz Atmega işlemciyi programlamak mümkündür. Bunu kolaylaştırmak adına Arduino bizlere bir IDE sunmaktadır.

* 1. Arduino IDE

Bir mikrodenetleyici içeren Arduino kartını programlamak için kullanılan masaüstü uygulamadır. Uygulama kendi içerisinde sunduğu kütüphane havuzu ile yüzlerce sensörün kütüphanesine erişim hizmeti vermektedir. Bunun yanı sıra Seri Port Ekranı ile veri izleme veya haberleşme, Seri çizici ile seri porttan gelen verileri grafiklendirebilmekteyiz.



Şekil Arduino IDE görüntüsü

* 1. Sensörler

En genel tanımıyla dış dünyadaki olayları insanların veya bilgisayarların anlayabileceği biçime çeviren cihazlardır. Tek başlarına işlevsiz olmakla birlikte, bir mikrodenetleyici veya mikroişlemci ile kullanılarak karar algoritmaları ve mantıksal sistemler oluşturulabilmektedir.

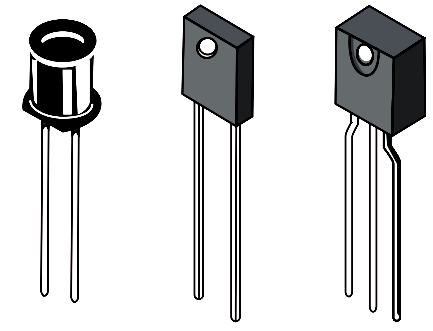
Sensörler giriş birimlerine göre 6’ya ayrılmışlardır;

* Mekanik sensörler
* Termal sensörler
* Elektriksel sensörler
* Manyetik sensörler
* Işıma sensörleri
* Kimyasal sensörler

Gerçekleştirdiğimiz projede de 2 adet sensör bize yardım etmektedir. Değişen ışık miktarını ölçebilmek için bir fototransistör ve vücut sıcaklığını ölçebilmek için MCP9808 Hassas Sıcaklık sensörü.

* + 1. Fototransistör

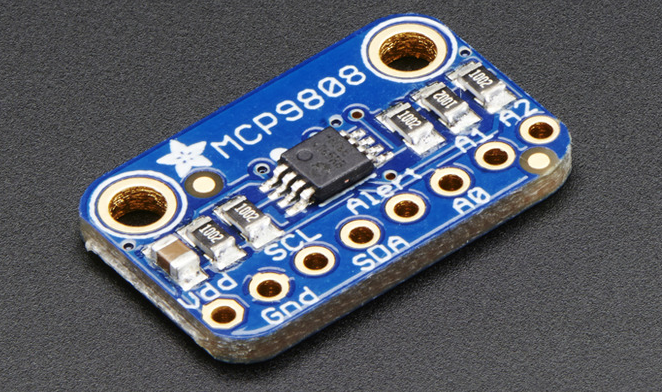
Fototransistör, üzerine düşen ışığı elektrik akımına çeviren bir sensördür. Işığa duyarlı bir taban bölgesine sahip yarı iletken cihazdır. Taban ışığı algılar ve iki bağlantı bacağı arasında akan akıma dönüştürür. Sistemde kullanılacak olan bu sensör kırmızı ve kızılötesi ledin parmak üzerinden geçen ışık miktarını hesaplamak için kullanılacaktır.



Şekil Fototransistör türleri

* + 1. MCP9808 Hassas Sıcaklık Sensörü

MCP9808 Sensörü Microchip firmasının geliştirmiş olduğu bir sıcaklık sensörüdür. I2C protokolünü kullanarak mikroişlemci veya mikrodenetleyici ile iletişime geçmektedir. Tipik olarak -40°C ile 125°C arasında ölçüm yapmaktadır. Kullanıcı seçenekli 4 farklı çözünürlük seçeneği bulunmaktadır. Hata payı tipik olarak ± 0.25°C’dir. 2.7V-5.5V arası voltaj geriliminde çalışmaktadır.



Şekil MCP9808 Yüksek Hassasiyetli Sıcaklık Sensörü

1. Raspberry Pi 4

En basit tanımıyla kart boyutunda bir bilgisayardır. İçerisinde bulunan ARM tabanlı işlemcisi, GPIO pinleri, USB giriş-çıkışları, Wifi, Bluetooth gibi dizüstü bilgisayarlarda sahip olduğumuz her şeye sahiptir. Film izleme, programlama, internette gezinme gibi birçok işlemi gerçekleştirebileceğimiz bu sistem harici bir monitör veya sanal server üzerinden ekran aktarma ile yönetilebilmektedir. Tüm bunlara güç vermesi için de harici olarak küçük bir adaptörü satılmaktadır.



Şekil Raspberry Pi 4

* 1. Genel Tanım

Raspberry Pi 4 Model B olarak karşımıza çıkan bu mini bilgisayarlar 2, 4 ve 8 GB RAM’e sahip modelleri ile piyasaya sürülmüştür. İçerisinde 4 çekirdekli Cortex-A72 işlemcisi ARM v8 mimarisine sahiptir. 64 Bit olarak üretilen bu işlemcinin tüm çekirdekleri 1.5 Ghz hızında çalışmaktadır.

Kendilerinin geliştirmiş olduğu Debian tabanlı Raspberry Pi OS ile çalışan bu sisteme harici olarak Linux ve Ubuntu tabanlı sistemler kurulabilmektedir. Sistem üzerine kurulabilecek Visual Studio, PyCharm gibi editörler ile programlama yapılabilmektedir. Bunun haricinde oluşturulmuş kullanıcı arayüzü ile işlemler kolaylaştırılmış, her yaştan insanın kullanabileceği bir sisteme çevrilmiştir.

* 1. Arduino ile Haberleşme

Raspberry Pi’nin Arduino ile haberleşmesi bu proje için USB üzerinden gerçekleştirilecektir. Arduino ve Raspberry Pi’nin sahip olduğu USB seri haberleşme yöntemleri kullanılarak iki cihaz arasında üç saniyede bir iletişime geçilecektir ve veri alışverişi sağlanacaktır.

Raspberry Pi tarafında projenin programlanması Python üzerinden yapılacaktır. Debian işletim sisteminin sahip olduğu USB seri portları kullanabilmeyi Python frameworklerinden olan PySerial ile oluşturulacaktır. Arduino standart bir bilgisayara bağlanmış gibi tüm verilerini gönderebilecek, Raspberry Pi üzerinden programlama yapılabilecektir.

* + 1. PySerial Kullanımı

PySerial framework’ünün kullanılabilmesi için öncelikle sistem genelinde kurulması gerekmektedir. Bunun için ilk önce sisteme yüklenmelidir;

pip install pyserial

Kodunu kullanarak terminal üzerinden PySerial kurulumunu gerçekleştiriyoruz. Sistem genelinde yükleme yapıldıktan sonra kod bloğumuzun içerisinde kütüphane tanımlama yerinde PySerial’ı eklememiz gerekmektedir.

import serial

Kütüphaneyi ekledikten sonra Seri Haberleşmeyi başlatmak için kullanılabilecek kod bloğu aşağıdaki gibidir;

PythonSerial = serial.Serial('COM Portu',”BAUD-Oranı”)

COM Portu bizim Arduino’muzun bilgisayarımıza bağlı olduğu USB portunu ifade etmektedir. Baud oranı ise seri haberleşme hızımızdır. Bu işlemleri yaptıktan sonra tanımlanmış olan Serial değişkeni üzerinden seri haberleşmede okuma ve yazma yapılabilmektedir.

arduinoData = PythonSerial..readline()

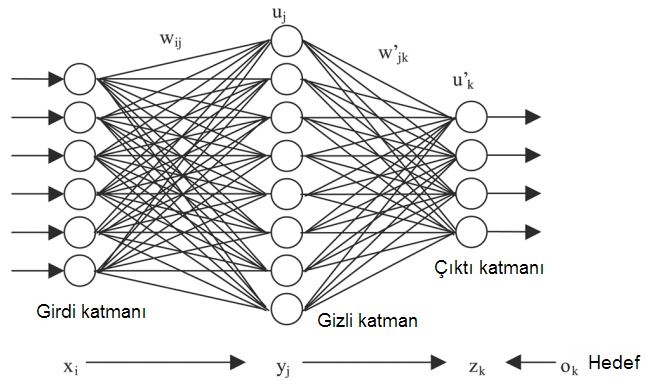
Yukarıdaki kod ise Arduino üzerinden gelen seri haberleşme verilerinin satır satır okunması için yazılmıştır.

1. Yapay Sinir Ağları

İnsan vücudundaki sinir ağlarından ilham alınmış hesaplamalı bir öğrenme sistemidir. Büyük verileri anlamak için kullanılabilmektedir. Belirli bir dosyadan aldığı verileri öğrenerek, başka veriler üzerinde karar algoritması oluşturmak için kullanılabilmektedir. Finansal işlemlerde, ticarette ve tıp alanında olmak üzere birçok yerde karşımıza çıkmaktadır.

* 1. Genel Tanım

Yapay sinir ağları, makine öğrenimi algoritmalarından ve araçlarından sadece birisidir. Bir yapay sinir ağının öğrenmesi için karışık algoritmalara gerek yoktur. Etiketleri ile verilen veri setindeki ilişkileri anlamaya çalışır. Verilen veriler yani örnekler düzenlenerek öğretilecek veriler ve etiketleri ayrılır. İlişkiler düzenlenir ve gerekli kod bloğu hazırlanır. Örnekler ile yapay sinir ağları eğitilerek ağırlıklar hesaplanır ve yapay sinir ağımızın sonuçlardaki hata oranı kabul edilebilir ise bu ağırlıklar kabul edilir, hata oranı kabul edilemeyecek düzeyde ise yapay sinir ağımız tekrar tekrar eğitilerek hata oranı düşürülmeye çalışılır. Tüm bu işlemler sonucu yapay sinir ağından bir model oluşturulur. Bu model dışa aktarılabilir veya kod bloğu içerisinde sürekli olarak kullanılabilir.



Şekil Yapay Sinir Ağı Katman Görüntüsü

* + 1. Yapay Sinir Ağlarının Özellikleri

Yapay sinir ağlarının kompleks problemlerde çözüm yöntemi olarak kullanılabilmesinin başlıca sebepleri vardır. Doğrusal olmayan hücre yapısı, yapay sinir ağının da doğrusal olmamasını sağlamakla beraber kompleks problemlerin çözümünde kolaylıklar sağlamaktadır.[2]

* Doğrusal Olmama
* Paralel Çalışma
* Öğrenme
* Genelleme
* Hata Toleransı ve Esneklik
* Eksik Verilerle Çalışma
* Çok Sayıda Değişken ve Parametre Kullanma
* Uyarlanabilirlik
  1. Tensorflow

Tensorflow açık kaynaklı bir makine öğrenimi platformudur. Google tarafından geliştirilen açık kaynaklı bir kütüphane olup veri akış grafikleri ile sayısal hesaplama kullanarak makine öğrenimi ve derin sinir ağları araştırmalarına katkıda bulunur. Google Brain ekibi tarafından 2015 yılından bu yana genel kullanım için açılmıştır. Makine öğrenimi ve derin sinir ağlarını araştırmak için geliştirilmiştir. Kendine has kütüphanesi ile Linux, Mac ve Windows gibi platformlara destek vermektedir. Birçok yazılım dilini desteklemektedir. Derin öğrenme için büyük avantaj sağlamaktadır. CPU ve GPU gibi hesaplama aygıtlarına destek vererek uzun süren derin öğrenme süreçlerinde karmaşayı ortadan kaldırarak daha kısa sürede sonuçlanmasını sağlamaktadır.

Bu tez çalışmasında yapay sinir ağı modellemesini TensorFlow kütüphanesini Python 3.9 üzerinde kullanarak gerçekleştireceğiz. Yapılacak bu çalışmada TensorFlow’un karmaşık kod yapısını basitleştirmek için işlemleri TensorFlow’u backend olarak kullanan Keras kütüphanesi üzerinden gerçekleştireceğiz.

* + 1. Tensorflow Kurulumu

Projede kullanacağımız cihaza uygun kütüphaneleri indirmek gerekmektedir. Yapay sinir ağı oluşturacağımız cihaz Raspberry Pi 4 ve Debian tabanlı Raspberry Pi İşletim Sistemi olduğundan dolayı buna uygun kütüphaneleri indirerek yüklememiz gerekmektedir.

sudo apt install libatlas-base-dev

Bu kod ile Debian üzerinden tüm lineer cebir çekirdeklerini indiriyoruz. Tensorflow’un sistemde çalışması için gerekli kütüphanedir.

pip3 install tensorflow

Lineer cebir çekirdeklerini yükledikten sonra Tensorflow ‘u sistemimize yüklüyoruz. Yükleme yaklaşık 10-15 dakika sürmektedir.

import tensorflow as \*

“\*” ile tanımlanan yer kullanıcıya bağlıdır. Kod bloğu içerisinde Tensorflow’u nasıl çağırmak istiyorsa o şekilde bir değişken ismi belirlemelidir. İşlemler tamamlandıktan sonra kod bloğumuz içerisinde kullanıma hazır hale geliyor. Kütüphaneyi ekleyerek programlama için kullanabiliriz.

* 1. Keras

Keras, bir Python kütüphanesi olmakla birlikte derin öğrenme modelleri oluşturmak ve eğitmek için kullanılabilecek API’dir. Modelleri kolay ve hızlı şekilde oluşturmak için kullanılabilmektedir. CPU ve GPU hesaplama aygıtlarını kullanma imkânı sağlayarak zamandan tasarruf sağlamaktadır.

* + 1. Keras Kurulumu

Keras kütüphanesini yükleyebilmek için sistemde Tensorflow’un yüklü olması gerekmektedir. Tensorflow’un yüklü olduğundan emin olduktan sonra;

pip install keras

Komutu ile sisteme yükleme işlemini gerçekleştiriyoruz. Ardından geliştirmek istediğimiz projeye;

import tensorflow as tf

from tensorflow import keras

Kodlarını kullanarak projeye dahil ediyoruz.

* 1. Pandas

Veri bilimi, veri analizi ve makine öğrenimi görevleri için yaygın şekilde kullanılan açık kaynaklı Python kütüphanesidir. İstatiksel analiz, veri görüntüleme, veriyi normalize etme, veri ekleme ve veri silme gibi birçok işlemin yapılabileceği basit bir yapısı vardır.

* + 1. Pandas ile Yapılabilecekler
* Veri temizleme
* Veri doldurma
* Verileri normalize etme
* Birleştirir ve ekleme
* Veri görüntüleme
* İstatistiksel analiz
* Veri denetimi
* Veri yükleme ve kaydetme
  + 1. Pandas Kurulumu

Raspberry Pi OS üzerinde terminal açılarak aşağıdaki kod blokları yazılır. Bu işlemler sayesinde sistem genelinde Pandas kurulumu tamamlanır.

pip install pandas

veya

pip install python3-pandas

Yukarıdaki kod blokları Debian üzerinde hata verebilmektedir. Bunun için PyCharm üzerinden Python derleyici ayarlarından proje içerisine Pandas indirilerek işlemler tamamlanabilir.

import pandas as \*

Kod içerisine eklemek için yukarıda belirtilen komut yazılabilir. “\*” ile belirtilen yere kullanıcı kod bloğu içerisinde Pandas’ı nasıl çağırmak istiyorsa o şekilde tanımlama yapabilir.

* 1. Sci-kit Learn

Python üzerinde denetimli ve denetimsiz birçok makine öğrenimi algoritmalarının gerçekleştirilmesini sağlayan kütüphanedir. Numpy, Pandas gibi kütüphanelerin üzerine inşa edilmiştir.

* + 1. StandardScaler

StandardScaler, ortalamayı çıkararak ve ardından birim varyansa ölçeklendirerek bir özelliği standart hale getirir. Birim varyans, tüm değerlerin standart sapmaya bölünmesi anlamına gelir.[3] Bu bitirme tezinin projesinde veri setinin düzenlenmesi için kullanılacaktır.

* + 1. Sci-kit Learn Kurulumu

Sci-kit Learn diğer kütüphaneler gibi sistem genelinde kurulabilmektedir. En güncel sürüm Python 3 üzerinde çalışmaktadır. Kurulum adımları;

pip3 install scikit-learn

Komutu kullanılarak sistem genelinde kurulumu gerçekleştirilebilir.

from sklearn.preprocessing import StandardScaler

Yukarıdaki kod ise kod bloğu içerisine StandardScaler ölçeklendiriciyi eklememizi sağlar.

1. Arduino İle Android Haberleşmesi
   1. Android Studio

Android uygulamaları geliştirmek için içerisinde emülatör barındıran bir IDE’dir. Temel olarak İntelliJ İdea’ya dayanan bu sistem üzerinde Android’in farklı sürümlerine hitap edebilen uygulama ve oyun geliştirmeleri sağlanabilmektedir. Yazılım dilleri olarak Java ve Kotlin kullanmaktadır. Geliştirmeleri desteklemek için Gradle tabanlı oluşturma sistemi, şablonlar ve Github entegrasyonunu desteklemektedir.

* 1. Bluetooth API Paketi

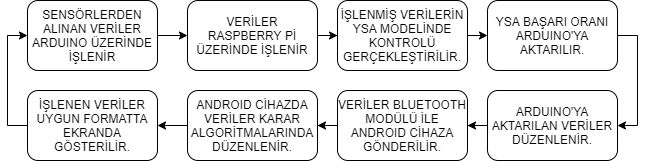
Cihazları tarama, cihazlara bağlanma ve cihazlar arasında veri aktarımını yönetme gibi bluetooth işlevlerini yöneten sınıflar sağlar. Hem düşük güç tüketim blueooth servislerini hem de klasik bluetooth servislerini desteklemektedir. [4]

Bluetooth API aşağıdaki işlemlere izin vermektedir:

* Diğer Bluetooth cihazlarını (BLE cihazları dahil) tarama,
* Eşleştirilmiş Bluetooth cihazları için yerel Bluetooth adaptörünü sorgulama,
* RFCOMM kanalları/soketlerini oluşturma,
* Diğer cihazlarda belirtilen soketlere bağlama,
* Diğer cihazlara ve cihazlardan veri aktarımı,
* Yakınlık sensörleri, kalp ritim monitörleri, fitness cihazları vb. gibi BLE cihazlarıyla iletişim kurmayı desteklemektedir.

1. Materyal ve Metot

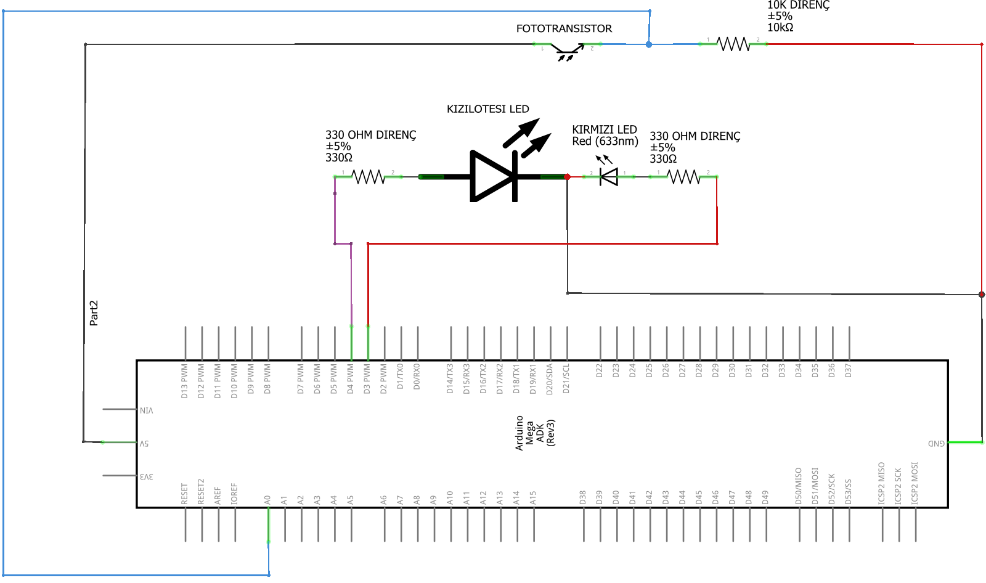
Bu proje çalışmasında deneylerin gerçekleştirilebilmesi için bazı cihazlara ihtiyaç vardır. Bu cihazların en başında hastadan veri okuyacağımız Arduino Mega ve sensörleri, YSA üzerinde alınan verileri kontrol etmek için kullanabileceğimiz Raspberry Pi 4, verileri uygulamadan okuyabileceğimiz bir Android cihaza ihtiyaç vardır. Oluşturulacak sistemin çalışma şeması aşağıdaki gibidir:



Şekil Gerçekleştirilecek Sistemin Blok Şeması

* 1. Nabız ve Satürasyon Sensörünün Oluşturulması

Nabız ve satürasyon verileri tek sensörü üzerinden ölçülecektir. Bu sensörün tasarımında bir adet IR LED, bir adet kırmızı LED, bir adet fototransistör, iki adet 330 Ohm direnç ve bir adet 10KOhm direnç kullanılacaktır.

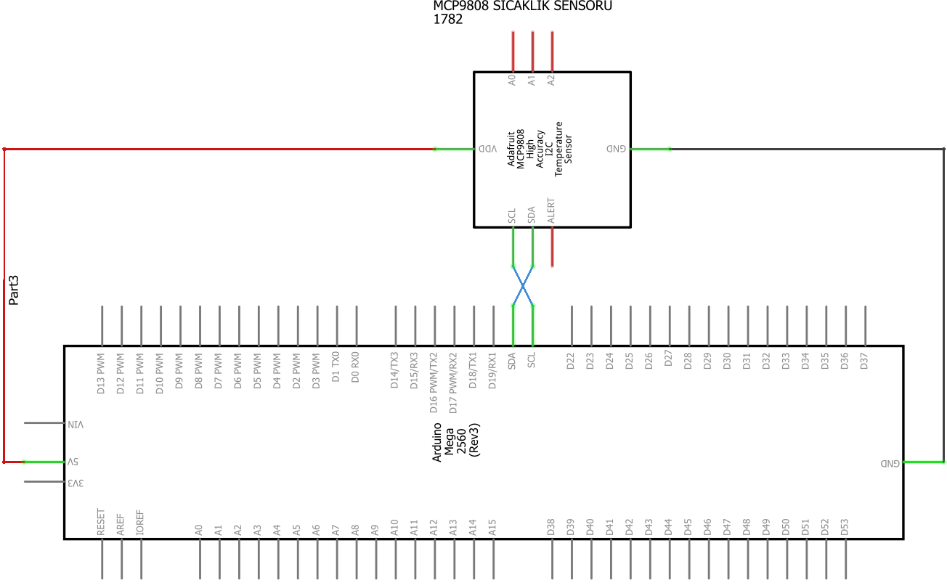


Şekil Sensörün Arduino’ya Bağlantı Şeması

Kırmızı ve kızılötesi ledlerin 5 Volt gerilime dayanamamasından dolayı önlerine 330 Ohm‘luk dirençler bağlanmıştır. Bunlar koruma amaçlı olup bir işlevleri yoktur. Fototransistör ise parmak üzerinden geçen ışık miktarına göre değişken direnç değerleri oluşturmaktadır ve diğer bacağının bağlı olduğu analog pin üzerinden okuma yapılacaktır.

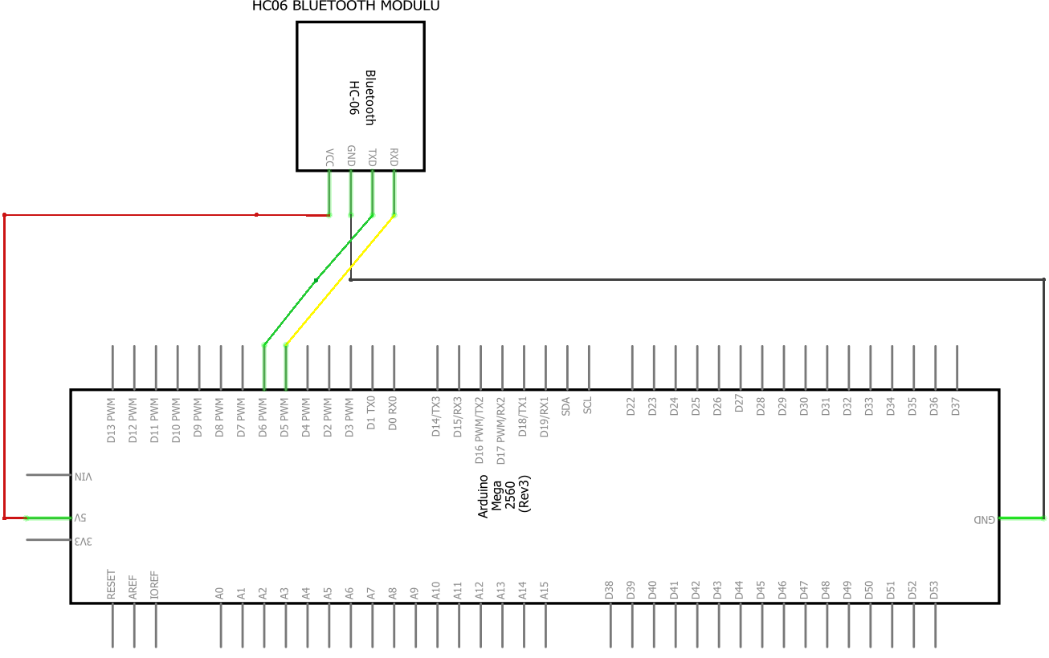
* 1. Diğer Modüllerin Sisteme Bağlanması

Nabız ve satürasyon verilerinin oluşturulduğu şema tamamlandıktan sonra bağlantısı yapılması gereken iki cihaz kalmaktadır. Bunlardan ilki MCP9808 Sıcaklık Sensörüdür. Bu sensör I2C haberleşme birimini kullanmaktadır. Arduino Mega üzerinde 20. ve 21. pinlere bağlanması gerekmektedir. Bağlantı şeması aşağıdaki gibidir;



Şekil Sıcaklık Sensörünün Arduino Bağlantısı

Sıcaklık sensörü haricinde Arduino üzerinden kullanacağımız bir diğer modül ise HC-06 Blueotooth Haberleşme modülüdür. Bu modül iOS, Android vb. birçok bluetooth iletişimi destekleyen cihazlar ile birlikte çalışabilmektedir. İki taraflı iletişimi destekler yani hem veri gönderebilir hem veri alabilir. Bağlantı şeması aşağıdaki gibidir;



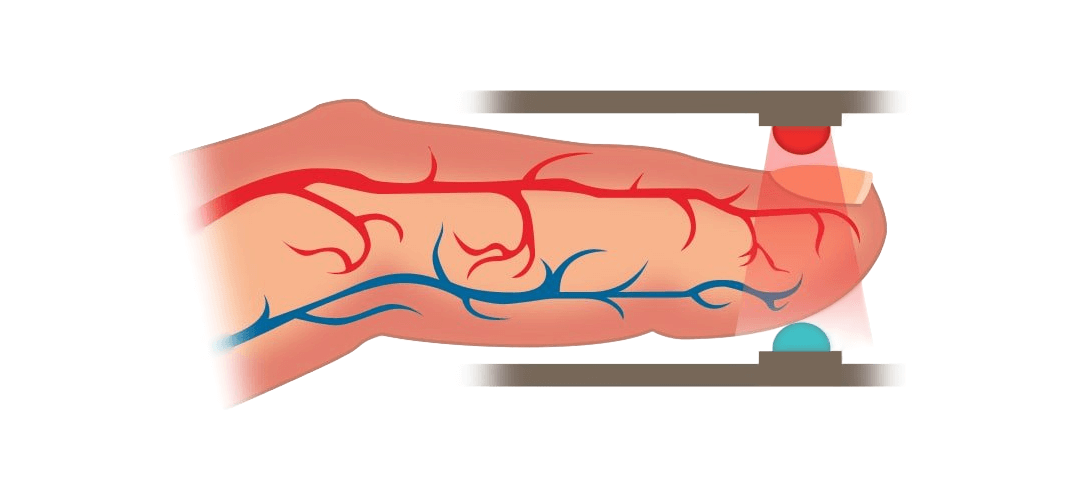
Şekil HC-06 Bluetooth Modülünün Arduino Bağlantısı

* 1. Sensörlerin Çalıştırılması ve Verilerin Okunması

Bağlantılar tamamlandıktan sonra Arduino IDE üzerinde Arduino programlanmaya başlanabilir.

Programlama bölümünde ilk olarak nabız ve satürasyon cihazı yani oksimetrenin oluşturulmasıyla başlanmaktadır. Standart bir oksimetre incelendiği zaman içerisinde bir fototransistör, bir kırmızı led ve bir kızılötesi led içermektedir. Bu cihazlarda ölçüm yapılırken dokularda oluşan farklılıklardan yararlanılmaktadır.

Kızılötesi led’in kullanılması nabız ölçümü için gereklidir. Kalp her attığında damarlarda oluşan genişleme ışık geçirgenliğini azaltmaktadır. Nabız her attığında değişen bu geçirgenlik miktarının hesaplanması sayesinde nabız sayısı bulunabilmektedir.



Şekil Pulse Oksimetre Görüntüsü

Kırmızı LED ise kandaki oksijen oranını bulmak için kullanılmaktadır. Satürasyonun hesaplanması kandaki hemoglobin oranı kullanılarak gerçekleştirilir. Oksijene doymuş hemoglobinin toplam hemoglobine oranı bize satürasyonu (SpO2) vermektedir. Oksijene doymuş hemoglobin, oksijensiz hemoglobine göre daha kırmızıdır. Kırmızı ışığı soğurduğundan dolayı parmağın kırmızı ışık geçirgenliği azalır ve buradan satürasyon hesabı yapılabilir.

Sistemde 20 milisaniyelik okumalar ile birlikte toplamda 40 milisaniye çalışma süresi vardır. Bir döngü 40 milisaniyede tamamlanır ve nabız satürasyon ölçümleri yapılmış olur.

// kızılotesi led çalıştırılır.

//kırmızı led kapatılır.

digitalWrite(REDLed,LOW);

digitalWrite(IRLed,HIGH);

n = 0;

start = millis();

reader = 0.;

do{

reader += analogRead (sensorPin);

n++;

}

while(millis() < start + T); //T değişkeni 20’dir ve 20 milisaniye boyunca kızılötesi ledden okuma yapıldı

reader /= n; //okuma sayısına göre ortalama veri çıkartılmaktadır.

IR LED için yapılan çalışmada analog pinden alınan değerler reader değişkenine atılır ve ölçüm sayısına bölünerek(n) ortalaması alınır.

Ptr olarak belirtilen değer sistemin kaç kere çalıştığıdır. “orn\_siz” olarak belirtilen değer örnek sayısıdır. Burada yapılan işlem önceki ölçülen değerin çıkartılarak son ölçülen değerin ortalamaya eklenip örnek boyutuna bölünmesidir. Bu şekilde ölçülen son kızılötesi değeri bulunur. Tüm bu işlemler aynı şekilde kırmızı led için de yapılıp son kırmızı led değeri elde edilmektedir.

if(ornekSayaci>=ornekler){

ornekSayaci =0;

IRmax = 0; IRmin=1023; REDmax = 0; REDmin=1023;

for(int i=0;i<max\_period;i++) {

if( okuIRMM[i]> IRmax) IRmax = okuIRMM[i];

if( okuIRMM[i]>0 && okuIRMM[i]< IRmin ) IRmin = okuIRMM[i];

okuIRMM[i] =0;

if( okuREDMM[i]> REDmax) REDmax = okuREDMM[i];

if( okuREDMM[i]>0 && okuREDMM[i]< REDmin ) REDmin = okuREDMM[i];

okuREDMM[i] =0;

}

R = ( (REDmax-REDmin) / REDmin) / ( (IRmax-IRmin) / IRmin ) ;

}

ortIR -= okuIR[ptr];

ortIR += reader;

okuIR[ptr] = reader;

sonIR = ortIR / orn\_siz;

Burada yapılan işlemler sonucu kırmızı ve kızılötesi ledlerin maksimum ve minimum değerleri bulunup değişkenlere kaydedilmektedir. Alınan maksimum ve minimum değerler sonucu kaydedilen dizinin tüm elemanları sıfırlanıp sonraki ölçümde hata oluşmaması için sıfırlanmaktadır.

R olarak belirtilen değer ise referans değeridir. Satürasyon için bir fonksiyon olarak da tanımlanmaktadır. Bu referans değerinin hesaplanması ile satürasyon doygunluğu hesaplanabilmektedir.[5]

if (sonIR < 50 ) {

….

….}

Yukarıdaki bölümde yapılan işlem tamamen parmak kontrol amaçlıdır. Fototransistör üzerinde bir parmak olmadığı zaman kızılötesi ledin verdiği değer 50’den küçük olmaktadır. Bu şekilde parmağa bağlı olmayan cihazın durumunu hasta yakınına iletebilmekteyiz.

Burada gerçekleşen işlemler ise satürasyon ve nabızın hesaplanması ile ilgilidir. Son kızılötesi değer önceki kızılötesi değerden büyük olduğu sürece bu algoritmaya girilecektir. Bunun anlamı ise bir nabız atımının bulunmuş olması ve nabızın hesaplanabilir olmasından gelmektedir. Toplamda üçten fazla (yani ger\_olcum değerinden yüksek) olan durumlarda ise sistem ortalamalar üzerinden işlem yapmaya başlar. İki ölçüm arasındaki periyot farkı %10’dan küçükse yani arada fazla bir zaman farkı yok ise hesaplanan referans değeri ortalama referans değerine eklenir.

if (sonIR > oncekiIR){

yuks\_say++;

if (!yukselen && yuks\_say > ger\_olcum){

// Yükselen bir eğri saptadık, bu bir nabiz atışına işaret eder.

// Son atıştan sonraki zaman kaydedilir ve son 10 atım ortalama değer yakalamak için takip edilir

// Yükselişi takip etmek aynı yükselimleri birden fazla kez ölçmemizi engeller

yukselen = true;

olcumR[m] = R;

olcumPeriod[m] = millis() - son\_atim;

son\_atim = millis();

int period = 0;

for(int i =0; i<olcumler; i++){

period += olcumPeriod[i];

}

// max ve min değerleri bulmak için ortalama periyot ve örneklerin sayısı hesaplanır

period = period / olcumler;

ornekler = period / (2\*T);

int ortPeriod = 0;

int c = 0;

// c son 10 atımdaki iyi olcumlerin adetini tutar

for(int i =1; i<olcumler; i++) {

if ( (olcumPeriod[i] < olcumPeriod[i-1] \* 1.1) &&

(olcumPeriod[i] > olcumPeriod[i-1] / 1.1) ) {

c++;

ortPeriod += olcumPeriod[i];

ortR += olcumR[i];

}}

m++;

m %= olcumler;

// Gösterilen bpm ve R değerleri son 5 iyi atımın ortalaması alınarak hesaplanır.

ortBPM = 60000 / ( ortPeriod / c) ;

ortR = ortR / c ;

if(c > 4) {

SpO2 = -23 \* R + 103;

…….

Ortalama nabız hesaplanması ise KY-039 Nabız sensörünün kodlarından alınmıştır.[6] Ortalama referans değeri hesaplanması toplam doğru ölçümlere bölünerek tekrar bir ortalama alınması ile oluşturulmuştur. Doğru olan değerlerin de ortalaması alınarak toplamda minimum dört ölçümün ortalaması alınarak zamanla düzgün ölçüm yapan(lineer) bir sistem gerçekleştirimi esas alınmıştır.

“SpO2 = -23 \* R + 103;” hesaplanması ise referans alınan bir oksimetrenin değerlerine en yakın ulaşılacak şekilde ayarlanmıştır. Bir direncin farklı olmasında bile değişen gömülü sistem uygulamalarında referansa göre düzenlenmiş bir yazılım ile birbirine yakın sonuçlar alınabilmektedir.

Artık nabız ve satürasyon verilerini hastadan alabildiğimize göre sıcaklık sensörünü sisteme bağlamak kalıyor. Şekil 9.2.1’e göre MCP9808 bağlandıktan sonra kod bloklarının eklenmesi kalıyor.

#include "mcp9808.h"

#include <Wire.h>

MCP9808 sicaklikSensor(24);

void setup(){

sicaklikSensor.setResolution(3);

}

void loop(){

int vSicaklik = sicaklikSensor.getTemperature();

}

Oluşturulan kod bloğundan sonra vücut sıcaklığı değerimizi vSicaklik adında oluşturduğumuz değişkende tutabilmekteyiz. “#include "mcp9808.h" #include <Wire.h>” komutları kullanılarak gerekli kütüphanelerin eklemesini yapmaktayız. “Wire.h” kütüphanesi I2C haberleşmeyi kullanabilmemiz için gerekli olan kütüphanedir. "mcp9808.h" ise sensörden alınan değerlerin matematiksel olarak hesaplanıp bize santigrat derece veya fahrenhayt olarak sıcaklık verisini vermektedir. “setResolution(3)” ile sıcaklık sensörümüzün çözünürlüğünü belirliyoruz. Bu çözünürlük virgülden sonra gelen değerler ve doğruluğu etkilemektedir. Hastada kritik olarak düşünebileceğimiz sıcaklık vital bulgusunun doğru ve keskin sonuçlar vermesi önemlidir.

“getTemperature();” ile sensörden alınacak değer her loop döngüsünde çalışacak ve vSicaklik değerine yazdırılacaktır.

Raspberry Pi ile Arduino seri haberleşmesi oluşturulurken USB üzerinden haberleşmenin gerçekleştirilmesi hedeflenmiştir. Bir porta bağlı olan Arduino, verileri seri monitöre yazdırırken bağlı olduğu cihaza göndermektedir. Bu verileri Raspberry Pi’den alıp değişkenlerde tutmak temel hedeftir.

* 1. Yapay Sinir Ağı Modelinin Oluşturulması

Yapay sinir ağlarında modelin oluşturulması için ilk önce bir veri setimizin bulunması gerekmektedir. Bu veri seti, oluşturulan bir Python kodu ile rastgele değerler üretilerek oluşturulacaktır. Ardından işlemlerin ve eğitim sürecinin hızlanması için Google Colab üzerinde CPU temelli olarak model eğitilip dışarı aktarılacaktır.

Dışarı aktarılan model Raspberry Pi’ye aktarılacak ve burada Arduino ve sensörlerden gelecek verilerin kontrolü gerçekleştirilecektir.

* + 1. Veri Setinin Oluşturulması

Veri seti için normal bir insanın vital bulgu değer aralıkları ile bunun dışında oluşturulacak problemli veriler ile gerçekleştirilecektir.

Satürasyon için: İnsan kanındaki oksijen seviyesinin yüzde 95-100 arasında olması normal kabul edilir.[7]

Vücut sıcaklığı için: Koltuk altından ölçülen vücut sıcaklığının 36 dereceden 37.2 dereceye kadar olan değerler normal olarak kabul edilmektedir.[8]

Nabız için: Normal nabız değerleri kişiden kişiye farklılık gösterir ancak belirli bir aralık içinde olması gerekir. Yetişkin ve sağlıklı insanlarda kalp atış hızı dakikada 60-100 aralığında olması normaldir.[9]

Bitirme tezi çalışmasında gerçekleştirilen değerler tamamen test ve bilgi amaçlı olup kesinlik belirtmemektedir. PVS hastalarında kullanılması gereken değerler hastanın yaşı, sağlık durumu vb. durumlara göre değişiklik gösterebilmektedir.

Bu verilerden yola çıkarak oluşturulan veri seti ise şu şekildedir;

import random

import pandas as pd

spO2 = [ random.randint(65,88)if(i<4000) else random.randint(89,98) for i in range(20000)]

sicaklik = [ round(random.uniform(35.8,37.5),2) if( i<16000) else round(random.uniform(37.5,40),2) if(i<18000)else round(random.uniform(32,35.8),2) for i in range(20000)]

nabiz = [ random.randint(60,94) if(i<16000) else random.randint(50,59) if(random.randint(0,1)) else random.randint(95,130) for i in range(20000)]

random.shuffle(spO2)

random.shuffle(temperature)

random.shuffle(pulse)

problem = [ 1 if( i<=88 or j>37.5 or j<35.8 or k<=60 or k>=100 ) else 0 for i,j,k in zip(spO2,temperature,pulse)]

df = pd.DataFrame({"spO2":spO2,"temperature":temperature,"pulse":pulse,"problem":problem})

df.to\_csv('ornek\_'+str(problem.count(1))+'.csv',index=False)

Toplamda 20000 verilik bir veri seti oluşturulmuştur. Bu veri setinde;

Satürasyon için: 65-88 arasında 4000’den az veri, geri kalan veriler ise 89-98 arasında oluşturulmuştur.

Sıcaklık için: 35.8 ile 37.5 derece arasında 16000’den az veri, 37.5 ile 40 derece arasında 2000’den az veri ve 32.5 ile 35.8 derece arasında ise geri kalan veriler oluşturulmuştur.

Nabız için: 60-94 bpm arasında 16000’den az veri, geriye kalan veriler ise 50 ile 59 bpm arasında ve 95 ile 130 bpm arasında oluşturulmuştur.

Tüm değerler daha sonra kendi aralarında karıştırılmıştır. Vital bulgular için geçerli olan değerler yani normal kabul edilen değerler “0“ ile etiketlenmiştir. Geri kalanlar ise problemli olarak belirlenmiş ve “1” olarak etiketlenmiştir.

Daha sonra Pandas kütüphanesi kullanılarak tüm bu veriler bir csv dosyasına kaydedilerek yapay sinir ağında kullanılabilir bir veri seti elde edilmiştir.

* + 1. Google Colaboratory

Google Colab olarak da geçen bu sistem tarayıcı üzerinde hiçbir yapılandırmaya gerek kalmadan python kodlarını yazabilmemize olanak sağlamaktadır. Yapay sinir ağlarının daha hızlı ve modüler oluşturulması için kullanılacak bu sistemde öncelikle veri setimizi yüklememiz gerekmektedir.

from google.colab import files

uploaded = files.upload()

Komutlarını kullanarak yükleme ekranını çalıştırırız. Seçtiğimiz veri setini yükleyerek “uploaded” adında değişkende tutarız.

import pandas as pd # data processing

import matplotlib.pyplot as plt

import io

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import tensorflow as tf

from tensorflow import keras

from keras import Sequential

from keras.layers import Dense

Ardından gerekli olan kütüphaneleri ekleriz. Bu kütüphaneler yapay sinir ağı modelini oluşturmamızda gerekli olacak verilerdir.

data = pd.read\_csv(io.BytesIO(uploaded['ornek\_9783.csv']))

print(data)

Eklediğimiz veri setini baytlara ayırarak data değişkeninde tutarız. Daha sonra bize verileri göstermesini isteriz.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | spo2 | sicaklik | nabiz | problem |
| 0 | 94 | 36.03 | 76 | 0 |
| 1 | 69 | 39.55 | 69 | 1 |
| 2 | 96 | 33.72 | 61 | 1 |
| 3 | 93 | 37.12 | 82 | 0 |
| 4 | 92 | 38.28 | 88 | 1 |
| 5 | 90 | 36.08 | 72 | 0 |
| … | … | … | … | … |

Tablo Veri Setinin Görüntüsü

Bu işlemler neticesinde veri setimizin düzenli görüntülendiğini gördükten sonra etiketi ayırmak kalıyor.

X = data.drop('problem', axis=1)

y = data['problem']

Bu işlemler ile “X” değişkeninde problem harici olan “spo2, nabiz, sicaklik” verileri tutulurken “y” değişkeninde ise problem satırları tutulmaktadır.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=1)

Burada X ve y değişkenlerinden test ve eğitim için verileri ayırmaktayız. %10’luk kısım test için ayrılırken geri kalan bölüm ise eğitim için ayrılmaktadır.

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

Bu bölümde ise StandardScaler kullanarak X\_train ve X\_test verilerimizi normalize etmekteyiz. Oluşturulacak olan modelin verileri anlaması için gerekli olan bölümdür.

* 1. Model Oluşturulması

Eğitim ve test için veriler ayarlandıktan sonra model oluşturulmaya geçilebilir.

model = Sequential()

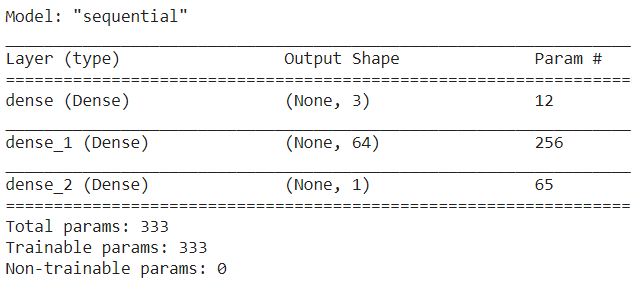
model.add(Dense(3, activation='relu', input\_dim=3))

model.add(Dense(64, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.summary()

Modelimiz Sequential olarak belirlenmiştir. Doğrusal katmanlara sahiptir. Toplamda üç adet katmanı vardır. Summary ile gerçekleştirilen modelin özeti aşağıdaki gibidir.



Şekil Oluşturulan Modelin Özeti

Tüm işlemlerin ardından modeli derlemek ve eğitmek kalıyor. Bunun için aşağıdaki python kodlarını yazarak tamamlıyoruz.

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

history=model.fit(X\_train, y\_train, batch\_size=5, validation\_data=(X\_test, y\_test), epochs=10, verbose=1)

Burada model fit edilirken beş örnekte bir çalıştırıyoruz. Toplamda 10 kere çalıştırılarak modeli oluşturduk. Şimdi de modeli Raspberry Pi üzerinde direkt olarak kullanmak için dışa aktaracağız. Burada yine Keras kütüphanesinden yardım alarak işlemlere devam ediyoruz.

!mkdir -p saved\_model

model.save('saved\_model/my\_model')

“saved\_model” adında bir klasör oluşturup bunun içerisindeki “my\_model” klasörüne modelimizi ve model ağırlıklarımızı kaydediyoruz.

* 1. Raspberry Pi ile Seri Haberleşmenin Oluşturulması

Raspberry Pi ile haberleşirken Arduino bölümünde standart seri monitör kullanımında olduğu gibi seri haberleşmeden veriler gönderilecektir. Bu veriler ise nabız, satürasyon ve vücut sıcaklığı olacaktır.

Serial.begin(9600);

void loop(){

Serial.println("" + String(ortBPM) + "," + String(SpO2) + "," + String(vSicaklik));

}

Sensörlerden okunan veriler virgüllerle ayrılarak aktarılmaktadır. Sıcaklık değerleri noktalı olarak gönderilmektedir. Sırasına göre gönderilen veriler sırasına göre alınmalıdır.

Burada Arduino verileri gönderirken Raspberry Pi tarafından Python üzerinden verilerin alınması gerekmektedir. Bunun için PySerial kütüphanesinden yardım alarak iletişime geçilecektir.

* + 1. PySerial

Gelen veriler ilk başta bir Windows işletim sistemine sahip dizüstü bilgisayar üzerinden alınarak kullanılacaktır. Yazılacak Python kodu içerisine PySerial kütüphanesi dahil edilerek işleme başlanır.

Arduino’nun bağlı olduğu COM portu seçilerek Arduino’da belirlenmiş olan haberleşme hızı birbirine denk olarak iletişim başlatılır. “while 1:” olarak tanımlanan bölüm kod çalıştığı sürece veri almamızı ve göndermemizi sağlayacak bölümdür.

import serial

arduinoCom= serial.Serial("COM5", baudrate=9600)

while 1:

arduinoVeri = arduinoCom.readline()

temp\_dizi = str(arduinoData).split(',')

temp\_dizi[0], temp\_dizi[1], temp\_dizi[2] = temp\_dizi[0][2:], temp\_dizi[1], temp\_dizi[2][:4]

print('SpO2:', temp\_dizi[1], "ortBPM:", temp\_dizi[0], "Sicaklik:", temp\_dizi[2])

nabiz = temp\_dizi[0]

spo2 = temp\_dizi[1]

sicaklik = temp\_dizi[2]

Arduino üzerinden satır satır gönderilen bilgiler Python tarafında da satır satır alınmaktadır. Alınan verilerden virgüller çıkartılıp sayısal değerler dizinin elemanlarına aktarılır. Aktarılan elemanlar “nabiz, spo2, sicaklik” değişkenlerinde tutularak yapay sinir ağına gönderilir.

* 1. Verilerin Yapay Sinir Ağı Üzerinde Test Edilmesi

Arduino’dan alınan sensör verileri Raspberry Pi cihaza aktarıldıktan sonra yapay sinir ağında kontrol için hazırlanmalıdır. Bunun için gerekli 7. bölümde kütüphaneler yüklü olmalıdır.

import pandas as pd # data processing

import serial

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

ysa\_model = tf.keras.models.load\_model("saved\_model/my\_model")

while 1:

veri\_test = pd.DataFrame({"spo2": [spo2], "sicaklik": [sicaklik], "nabiz": [nabiz]})

veri\_test = scaler.transform(veri\_test)

test\_basari = ysa\_model.predict(veri\_test)

print("{:.3f}".format(float(test\_basari)))

print(int(test\_basari \* 100))

Kütüphaneler eklendikten sonra oluşturulmuş ve dışarı aktarılmış modeli içe aktarırız. Bu model “ysa\_model” nesnesi olarak tutulur.

Ardından Pandas kütüphanesinin DataFrame fonksiyonu kullanılarak Arduino’dan gelen veriler tablo haline dönüştürülür. Tablo halindeki veriler StandardScaler kullanılarak modelin anlayabileceği şekilde normalize edilir. “ysa\_model.predict” fonksiyonu kullanılarak verilerin model üzerindeki başarısı alınır. Bu başarı daha sonradan yüzdelik dilimde yazdırılır.

* 1. Başarı Oranının Arduino ’ya Aktarılması

Raspberry Pi üzerinde verilere göre model başarısı alındıktan sonra Arduino’ya gönderilmektedir. HC-06 modülü kullandığımız Arduino’ya gönderilen model başarısı, Android haberleşmesinde kullanıcıya bilgi amaçlı kullanılabilecektir.

arduinoCom.write(str(int(test\_basari \* 100)).encode())

Komutu kullanılarak Arduino tarafına gönderimi sağlanmaktadır. Arduino tarafında ise veriler gönderildikten sonra yapay sinir ağında test süresi olarak 50 milisaniyelik bir bekleme yapılır. Ardından bir değişkende başarı oranı tutulur.

Serial.println("" + String(ortBPM) + "," + String(SpO2) + "," + String(vSicaklik));

delay(50);

basariOrani = Serial.parseInt();

Tüm bu işlemler tamamlandıktan sonra artık Android tarafında geliştirmenin gerçekleştirilmesi ve verilerin kullanıcıya gönderilmesi gerekmektedir.

* 1. Android Uygulamanın Geliştirilmesi
     1. Bluetooth İzinlerinin Oluşturulması

AndroidManifest adında ve xml uzantılı dosya android projelerin hepsinde hâlihazırda gelmektedir. Projenin temel kaynak setidir. Genellikle uygulama işlevleri izinleri ve gereksinimlerini burada ifade ederiz. Bizim için önemli olan kısım, HC-06 Bluetooth Modül ile iletişime geçebilmek ve gerekli olan bluetooth donanım eşleştirmesi için burada belirli uygulama izinlerini tanımlayarak sistemi hazırlamaktır.

Uygulama direk olarak çalıştığında doğru şekilde çalışması için kullanıcı tarafından verilmesi gereken izinleri <uses-permission> arasında yazmak gerekmektedir.

<uses-permission android:name="android.permission.BLUETOOTH" />

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />

İlk yazdığımız xml kodu ile uygulamaya bluetooth donanımına erişim izni tanır. İkinci yazdığımız xml kodu ise uygulamamıza, cihazın çevresindeki bluetooth aygıtlarını bulma ve eşleştirme iznini tanır.

* + 1. Arayüz Tasarımı

Arayüz planlamasında iki adet olarak ayarlanması uygun görülmüştür. Birisi uygulamanın açılış ekranı, diğeri ise hasta verilerinin görüntülendiği bölüm.

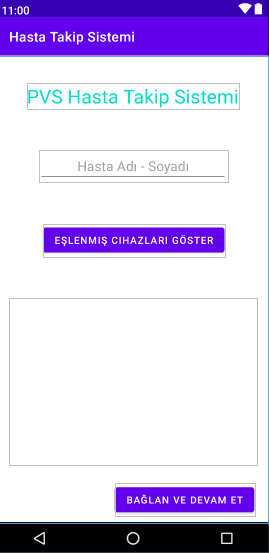
Ana ekranda olması beklenenler;

* Bağlanılacak cihaz seçimi
* Hasta Adı-Soyadı Girişi

Takip ekranında olması gerekenler;

* Nabız verisi
* Satürasyon verisi
* Sıcaklık verisi
* Hastanın durum bilgisi
* Verinin alındığı saat bilgisi
* Cihazın hastaya bağlı olup olmadığı
* Hasta Adı – Soyadı
* Yapay Sinir Ağı Başarı Oranı

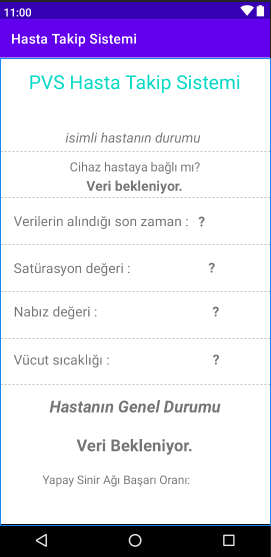
Bu verilere uygun olarak XML tabanlı olarak Android Studio üzerinden tasarlanmış arayüzler aşağıdaki gibidir.



Şekil Android Uygulama Ana Ekran Tasarımı

Yukarıdaki ekran görüntüsünde uygulamanın ana ekranı görülmektedir. Bu ana ekran üzerinde ilk olarak uygulamanın ismi geçmektedir. Ardından kullanıcı girişli olarak hasta adı ve soyadı alınarak veri ekranında gözükecek hasta bilgileri hazırlanacaktır.

Eşleştirilmiş cihazları göster butonu ile kullanılan akıllı telefonun daha önceden bağlanmış cihazları görüntüleme ve bağlanma izni verilmektedir. Alt tarafında yer alan tabloda ise daha önce bağlanılmış cihazlar görüntülecek, seçim yapıldıktan sonra bağlan ve devam et butonuyla veri ekranına geçilebilecektir.



Şekil Android Uygulama Veri Ekranı Tasarımı

Şekil 9.9.2’de yer alan ekran görüntüsünde uygulamanın veri ekran bölümü gözükmektedir. Bu ekranda veriler gösterilecek ve bu verilere bağlı olarak en altta hastanın durumu bildirilecektir.

* + 1. Bluetooth Bağlantısının Oluşturulması

Android üzerinde projeye dahil ettiğimiz BluetoothAdapter ile bluetooth bağlantıları oluşturabilmekteyiz. Bağlantı, bağlantıyı kesme gibi işlemleri yaptırabilmekteyiz. Ana ekranda ilk öncelikle listeleme bölümümüze önceden eşleştirilmiş cihazları yansıtıyoruz.

Bu işlemlerin ardından “Bağlan ve Devam Et” butonu ile seçilen cihaza ilk bağlantı işlemleri başlatılıyor. Bağlanılan cihazın bilgileri “veriEkrani” sınıfımıza aktarılarak bluetooth bağlantısı için hazırlık yapılıyor.

BluetoothDevice device = ((MyAdapter) (listView.getAdapter())).getSelectedItem();

Intent intent = new Intent(anaProgram.this, veriEkrani.class);

intent.putExtra(DEVICE\_EXTRA, device);

intent.putExtra(DEVICE\_UUID, mDeviceUUID.toString());

intent.putExtra(BUFFER\_SIZE, mBufferSize);

startActivity(intent);

protected List<BluetoothDevice> doInBackground(Void... params) {

Set<BluetoothDevice> pairedDevices = mBTAdapter.getBondedDevices();

List<BluetoothDevice> listDevices = new ArrayList<BluetoothDevice>();

for (BluetoothDevice device : pairedDevices) {

listDevices.add(device);

}

return listDevices;

}

Ana ekrandan veri ekranına geçildiğinde artık bluetooth haberleşmesi ile kablosuz haberleşmeyi başlatmak kalıyor. BluetoothAdapter’in fonksiyonlarından “connect()” metodu kullanılarak bağlantı başlatılıyor. Bağlantı başarısız olur ise kullanıcıya uyarı veriliyor.

protected Void doInBackground(Void... devices) {

try {

if (mBTSocket == null || !mIsBluetoothConnected) {

mBTSocket = mDevice.createInsecureRfcommSocketToServiceRecord(mDeviceUUID);

BluetoothAdapter.getDefaultAdapter().cancelDiscovery();

mBTSocket.connect();

}

}

catch (IOException e) {

e.printStackTrace();

mConnectSuccessful = false;

}

return null;

}

* + 1. Verilerin Arduino’dan Alınması

Arduino ile Android arasındaki bluetooth haberleşmesi tamamlandıktan sonra yine Raspberry Pi’ye gönderilen veriler gibi virgüllerle ayrılarak gönderilir. Toplamda 5 veri gönderilmektedir.

try {

inputStream = mBTSocket.getInputStream();

while (!bStop) {

byte[] buffer = new byte[256];

if (inputStream.available() > 0) {

inputStream.read(buffer);

int i = 0;

for (i = 0; i < buffer.length && buffer[i] != 0; i++) {

}

final String strInput = new String(buffer, 0, i);

mTxtReceive.post(new Runnable() {

@Override

public void run() {

ArrayList<String> veriler = new ArrayList<>(Arrays.asList(strInput.split(",")));

}

Kodu ile Arduino üzerinden gelen verileri bir dizide kaydediyoruz. Daha sonra sırası ile gönderilen verileri değişkenlere atayıp durum bildirimlerini ayarlayacağız.

* + 1. Zamanın Hesaplanması

Arduino’dan en son alınan verilerin zamanının bulunması önemlidir. Çünkü hastayla ilgilenen birey hastasından en son ne zaman veri aldığını görmek ister. Bunun için Android bölümünde “Date()” fonksiyonunu kullanarak “SimpleDateFormat” türünde işlemleri gerçekleştireceğiz.

if (inputStream.available() > 0) {

Date date = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");

String zaman = formatter.format(date);

…

}

Burada yapılan işlem Bluetooth üzerinden herhangi bir veri geldiği zaman ve “inputStream” kullanıldığı durumda “zaman” değişkenimize o anki saati, dakikayı ve saniyeyi aktarmaktır.

* + 1. Durum Bildirimlerinin Ayarlanması

Gelen verilere göre hastanın sağlık durumu hakkında bilgilendirmek gerekiyor. Bunu, gelen verilerin doğru aralıkta olup olmadığını kontrol ederek gerçekleştiriyoruz.

if(veriler.size() == 5){

char controlDurum = veriler.get(3).charAt(0);

String temp = controlDurum =='1' ? "Acil kontrol gerekli !": controlDurum == '2' ? "Kontrol gerekli !" : controlDurum == '3' ? "İyi" : controlDurum == '4' ? "Çok iyi !" : "Veri bekleniyor.";

int colorDurum = controlDurum =='1' ? Color.RED: controlDurum == '2' ? Color.rgb(252, 128, 0) : controlDurum == '3' ? Color.rgb(0, 120, 0) : controlDurum == '4' ? Color.BLUE : Color.BLACK;

durum.setTextColor(colorDurum);

int controlSat = Integer.parseInt(veriler.get(0));

controlSat = controlSat >= 90 ? Color.rgb(0, 120, 0): controlSat >=80 ? Color.rgb(252, 128, 0) : controlSat < 80 ? Color.RED : Color.BLACK;

satDeger.setTextColor(controlSat);

int controlNab = Integer.parseInt(veriler.get(1));

controlNab = controlNab < 60 ? Color.RED : controlNab >100 ? Color.RED : Color.rgb(0, 120, 0);

nabDeger.setTextColor(controlNab);

float controlSicak = Float.parseFloat(veriler.get(2));

controlSicak = controlSicak<35.5 ? Color.RED : controlSicak > 38 ? Color.RED : Color.rgb(0, 120, 0);

sicakDeger.setTextColor((int) controlSicak);

Burada gelen her veri için bir renklendirme yapılıyor. Sırasıyla;

1. “veriler.size() == 5” olarak kontrol edilen bölümde toplamda beş elemandan oluşan bir diziye sahip olup olmadığımızı görüntülüyoruz. Bu bizi olası hatalı gelecek verilerden ve bilgilerden korumak için gerçekleştirilmektedir.
2. “controlDurum” bizim hastanın durumuyla ilgili veriyi ekranda gösterdiğimiz bölüm. Arduino üzerinde ayarlanan yapay sinir ağı başarısına göre 1-2-3 ve 4 değerlerini almaktadır. Bunların haricinde “0” gönderiliyorsa hastadan veri alınamıyor anlamına gelmektedir. Aynı zamanda “colorDurum” bölümünde bu duruma uygun renklendirmeler yapılarak göze hitap etmesi amaçlanmıştır.
3. Satürasyon değeri “controlSat” bölümünde kontrol edilmektedir. 90’dan büyük değerlerde yeşil olacak yazı rengi, satürasyon düştükçe sarı ve kırmızı olacaktır.
4. “controlNab” bizim nabız değerlerini kontrol ettiğimiz bölümdür. 100’ün üzerinde ve 60’ın altındaki değerlerde kırmızı olan yazı rengi, 60-100 arasında ise yeşil olarak ayarlanmıştır.
5. “controlSicak” vücut sıcaklığının kontrol edildiği bölümdür. 35.5 santigrat dereceden düşük ve 38 dereceden yüksek vücut sıcaklığında kırmızı olarak yazılacak, 35.5-38 derece arasındaki sıcaklıklarda ise yeşil olarak belirtilecektir.
6. Tüm bunların haricinde hiçbir veri gelmiyorsa ekranda “Veri bekleniyor” olarak uyarı vermektedir.
7. “controlDurum” değişkeninde cihazın hastaya bağlı olup olmadığı da kontrol edilmektedir. Arduino üzerinden gelen “0” değeri cihazın Arduinoya bağlı olmadığını göstermektedir. Bu durum ekranda “Cihaz bağlantısı kesildi.” olarak gözükmektedir.
   * 1. Verilerin Ekranda Gösterilmesi

Alınan veriler kontrol edilip planlanan renk şemasında hazırlandıktan sonra ekranda yazdırmaya hazırdır. Bu veriler bilgi ekranında belirlediğimiz “textView” değişkenlerini “setText” fonksiyonunu kullanarak yazdırıyoruz.

durum.setText(temp);

satDeger.setText("%" + veriler.get(0));

nabDeger.setText(veriler.get(1) + " bpm");

sicakDeger.setText(veriler.get(2) + " °C");

veriZaman.setText(zaman);

basariOran.setText("% " + veriler.get(4));

if(Integer.parseInt(String.valueOf(controlDurum)) == 0){

baglilik.setText("Cihaz bağlantısı kesildi. Son veriler:");

baglilik.setTextColor(Color.RED);

}

else{

baglilik.setText("Cihaz bağlandı.");

baglilik.setTextColor(Color.rgb(0, 120, 0));

}

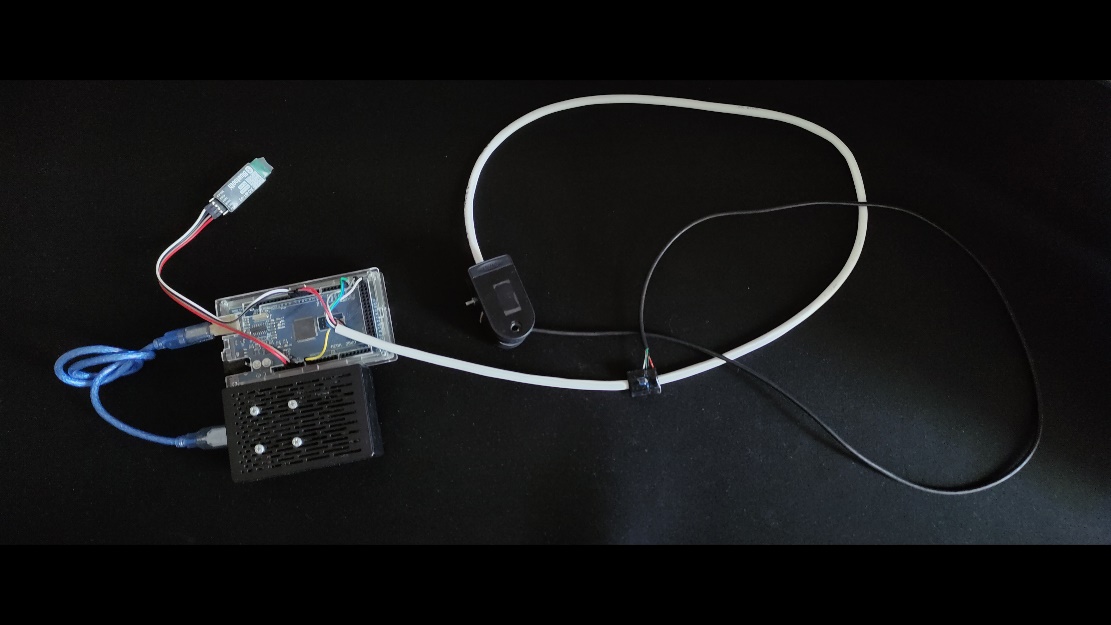
1. Çalışan Sistemin Test Edilmesi

Yapılan bağlantılar ve hazırlanan kodlar sonucunda sistem oluşturulmuştur. Hazırlanan ve bağlantıları tamamlanmış sensör yapısı aşağıdaki gibidir.



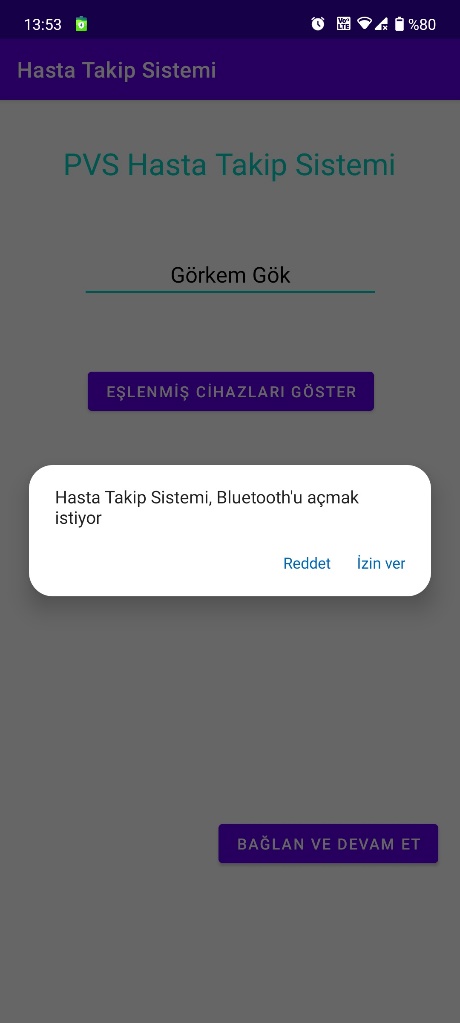
Şekil Hazırlanan Nabız ve Satürasyon Sensörü

Arduino ile Raspberry Pi’nin birbirlerine bağlanmış ve sensör yapısının Arduino’ya bağlanmış hali Şekil 17’deki gibidir.



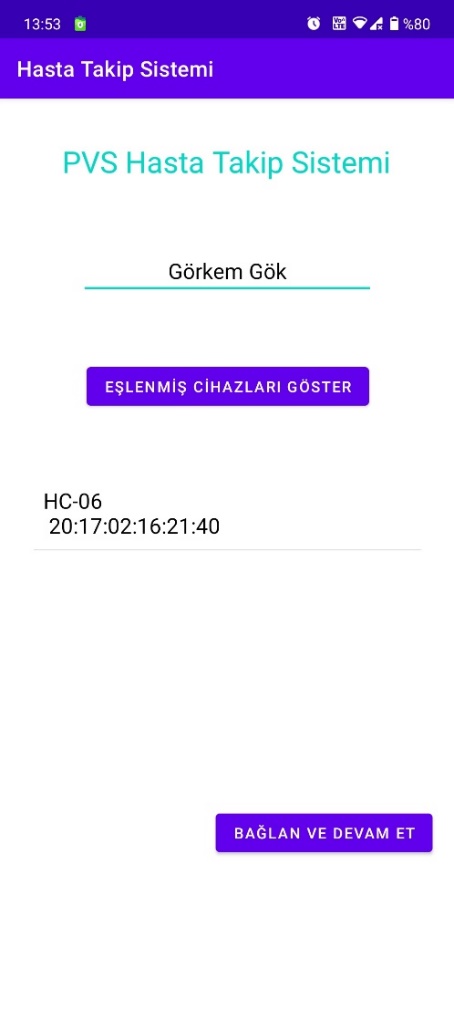
Şekil Sistemin Tamamlanmış Hali

Tüm bunlardan sonra Android ile iletişim kalmaktadır. Android sisteminde ilk kontrol bluetooth aktif/pasif durumudur.” Eşlenmiş cihazlar” butonuna basıldığında daha önceden mobil cihaz ile eşleşmiş cihazlar gösterilmektedir. Eğer bluetooth açık değilse ilk onun uyarısı oluşturulur. Bluetooth açık ve herhangi bir daha önceden bağlanmış cihaz yok ise bunun da uyarısı oluşturulur.



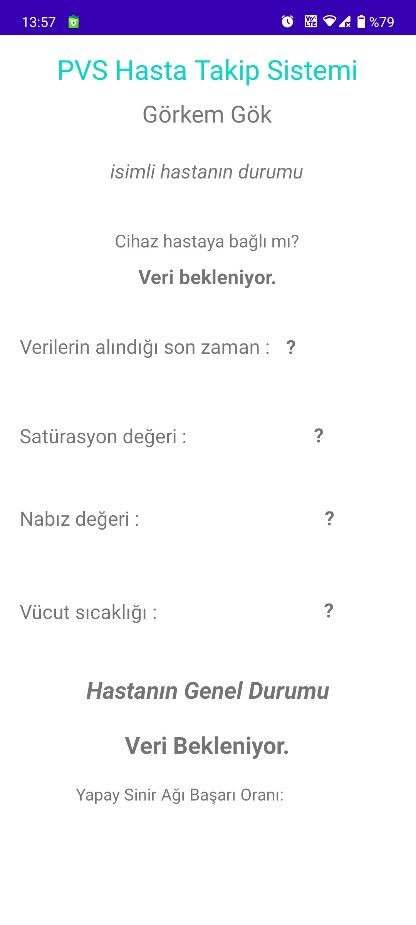
Şekil Bluetooth Açma İsteği

Bluetooth açıldıktan sonra karşımıza eşlenmiş cihazlar çıkmaktadır. Ayrıca üst tarafta yer alan bölüme zorunlu olmak koşuluyla hasta adı-soyadı girişi bulunmaktadır. Buradan kendi sistemimize ait bluetooth eşleştirilerek veri alışverişi başlatılabilir.



Şekil Eşlenmiş Cihaz Listesi

Artık bir bağlantı oluşturuldu ve verilerin Android cihaza gönderilmesi ekrana gelmesi beklenmektedir.



Şekil Veri Bekleniyor Ekranı

Cihaz herhangi bir bireye takıldığı zaman belirli bir süre okuma yapmaktadır. Bunun sebebi hastanın hayati bulgularının belirli bir sayıda alınıp ortalamasına göre işlem yapılmasıdır. Her başarılı okumanın ardından üç saniyede bir olmak üzere veriler gönderilir.



Şekil Başarı Oranı Yüksek Verilerde Panel Görüntüsü

Panelde verilerin alındığı zaman, satürasyon, nabız ve vücut sıcaklıkları gözlenmektedir. Bunun haricinde panelin alt bölümünde YSA başarı oranına göre hastanın genel durumu gösterilmektedir. Bunun haricinde kötü bir durum ile karşı karşıya kalan hastanının durumu ise hasta yakınına Şekil 22’deki gibi bildirilir.

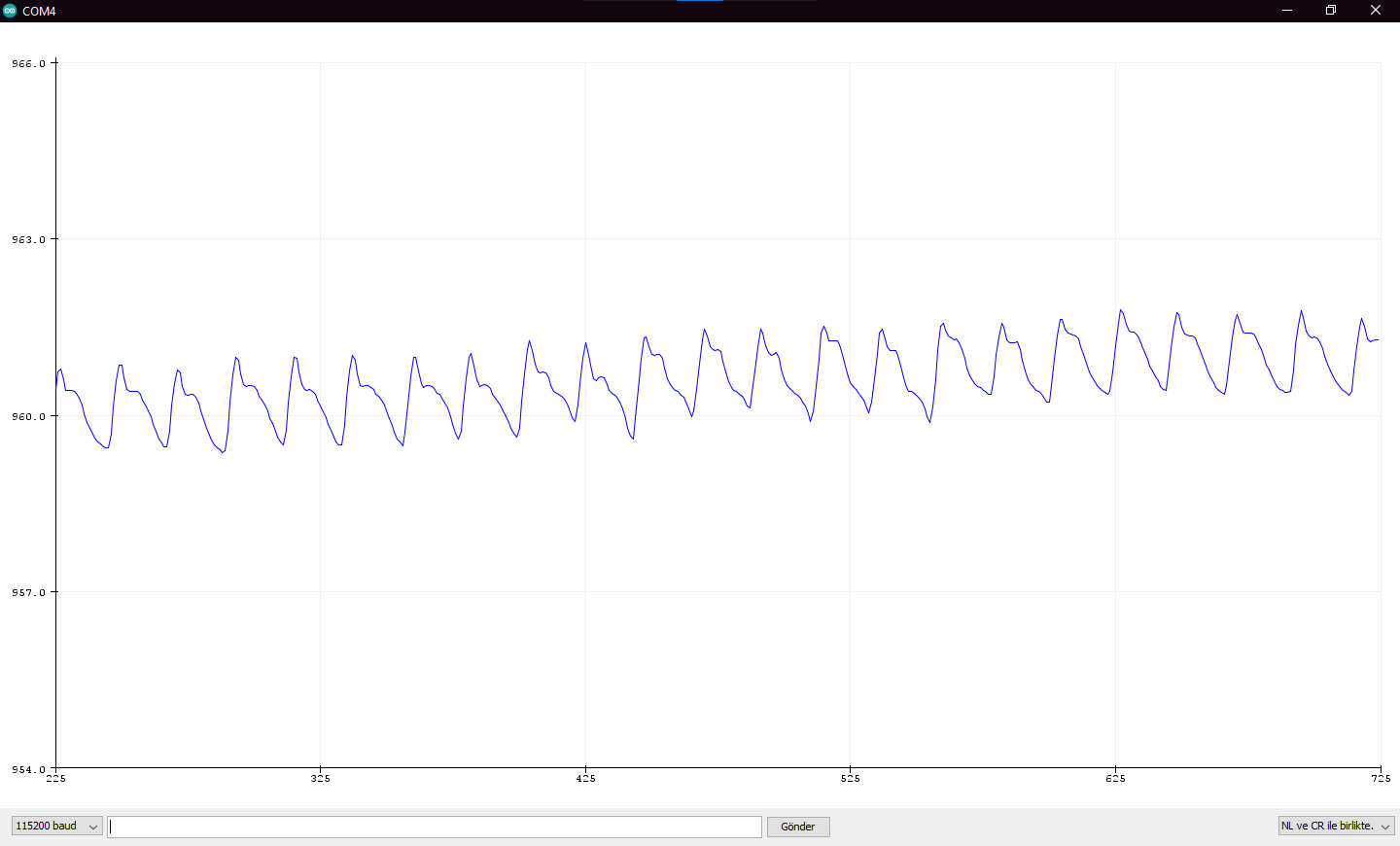


Şekil Kontrol Gerekli Olan Hasta Veri Paneli

Şekil 22’de verileri problemli olan bir birey görülmektedir. Bu bireyin nabız ve vücut sıcaklığı olması gereken aralığın dışındadır. YSA başarı oranı ise %0 olduğundan dolayı bireyin acil kontrole ihtiyacı var olarak gözükmektedir.

1. Bulgular
   1. Seri Çizide Nabız İzleme

Arduino IDE üzerinde bulunan Seri Çizici kullanılarak nabız atımlarını görmek mümkündür.

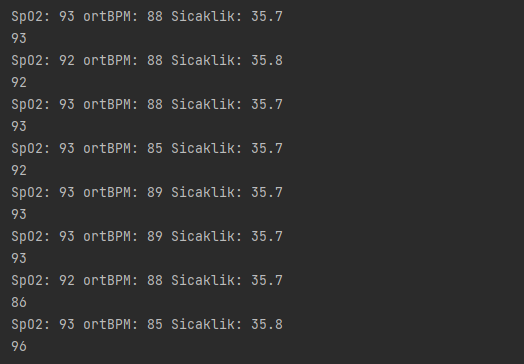


Şekil Seri Çizicide Nabız Atımları

Şekilde mavi çizgiler nabız atımlarını belirtir. Kalp her attığında damarlarda genişleme olmaktadır. Parmakta da olan bu durumu incelersek seri çizicideki artış ve azalış parmak üzerindeki damarların genişleyip daralmasından kaynaklanmaktadır. Burada izlemekte olduğumuz durum kızılötesi ledin fototransistör tarafından alınmasıdır. Nabız verisine çevirmek için kod üzerindeki gibi hesaplanması gerekmektedir.

* 1. Raspberry Pi YSA Oranını İzleme

Raspberry Pi üzerinde PyCharm uygulamasında, hazırlanmış olan kod çalıştırıldıktan sonra çıktı olarak üretilen değerler aşağıdaki gibidir;



Şekil YSA Başarı Oranının Görüntülenmesi

Değişken değerlere göre önceden eğitilmiş modelin de verdiği başarı oranı değişmektedir.

1. Sonuçlar

Tüm bu çalışmalarda Arduino’ya bağlı sıcaklık sensörü ve oksimetre sensörü ile hastadan vital bulgular alınabilmiştir. Alınan veriler düzenlenerek ve gerekli fonksiyonlar ile hesaplanarak Raspberry Pi üzerinde çalışan bir YSA modelinde başarısı test edilmiştir. Bu başarı oranı bize hastanın sağlıklı olup olmadığını göstermek için kullanılmış, uyarı durumlarının sadece tek bir kaynak üzerinden hesaplanmasında kolaylık sağlamıştır. Bu YSA başarı oranı ile geliştirilmiş olan Android uygulamada hastanın verileri ve sağlık durumu hakkında bilgilendirme ekranları hazırlanmış, çok kısa gecikmede çalışan hasta takip sistemi oluşturulmuştur.

Bitirme tezinde gerçekleştirilen bütün adımlar PVS hastaları veya bu tarz bir bakıma ihtiyaç duyan hastalar tarafından kullanılabilir. Düşük bütçeyle oluşturulabilecek sensörler ile vücutta oluşan değişimler kaydedilebilir, saklanabilir ve grafiğe dökülebilir. İnsan vücut sıcaklığını ölçmek için tasarlanmış veya kızılötesi sensörler ile daha kesin vücut sıcaklığı saptanabilir. Nabız ve satürasyon verilerinin alındığı oksimetre sensöründe referans değerinin düzenlenmesi şartıyla daha kaliteli komponentler kullanarak daha kesin sonuçlara ulaşması mümkündür. Oluşturulan yapay sinir ağı modeli eğitilirken hastanın kendisinden alınacak veya olması beklenen aralıktaki veriler ile yapılabilecek çalışmada bireysel olarak kullanım imkânı sunulabilecektir. Veri setinde yer alan veri sayısı 20.000’den daha yüksek sayıya çıkartılarak farklı verilerde daha kesin sonuçlara gidebilmek mümkündür. Android uygulama geliştirilirken göze hitap etmesi ve basit olması amaçlanmıştır. Bunun yanısıra grafik veya listeler kullanarak hastanın belirli bir zaman dilimindeki hayati bulgularını takip etme işlemi gerçekleştirilebilir.

Kaynaklar

[1] “Persistent vegetative state”,Wikipedia.org (Ocak 2021), Web. 11 Haziran 2021.

[2] “Yapay Sinir Ağları (YSA) Nedir?”, kod5.org (2016), Web. 11 Haziran 2021.

[3] “Scale, Standardize, or Normalize with Scikit-Learn”, towardsdatascience.com (2019), Web. 13 Haziran 2021

[4] “Developers/android.bluetooth”, android.com (2021), Web. 13 Haziran 2021

[5] Andrea De Pascalis (2015). *Sviluppo e validazione di un pulsossimetro a riflettanza indossabile al dito*, POLITECNICO DI MILANO, İtalya. 13 Haziran 2021

[6] “From KY-039 To Heart Rate”, arduino.cc (2018), Web. 17 Haziran 2021

[7] “Oksijen doygunluğu (tıp)”, wikipedia.org (2021), Web. 17 Haziran 2021.

[8] “Yüksek Ateş”, acibadem.com.tr (2020), Web. 17 Haziran 2021.

[9] “Nabız nedir? İdeal nabız kaç olmalı?”, Prof. Dr. Müslüm Şahin, medicalpark.com.tr (2020), Web. 17 Haziran 2021.