ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZET
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

SZAKDOLGOZAT

OE-KVK
2022.

Reichardt Szabolcs
T008377/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Műszertechnikai és Automatizálási Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Reichardt Szabolcs
Szakdolgozat száma: SZD220301091824O2
Törzskönyvi száma: T008377/FI12904/K **Neptun kódja:** [redacted]

Szak: villamosmérnöki
Specializáció: Műszer-automatika specializáció

A dolgozat címe: Androidos szoftver tervezése turbinás spirométerhez
A dolgozat címe angolul: Android software design for turbine spirometer
A feladat részletezése: Készítsen egy turbinás spirométer, illetve végezze el a hozzávaló androidos szoftver tervezését, gyártását. A turbinás spirométer alap spirometriai méréseket végez, melyeket real time lehet ábrázolni a szoftverben. A mérőfej, illetve a telefon bluetooth kapcsolaton keresztül kommunikálnak egymással. A méréshez szükséges a páciens neme, életkora, testsúlya, magassága, ebből számolja ki a szoftver az ideális görbét.

Intézményi konzulens neve: Sándor Tamás

A kiadott téma elévülési határideje: 2025. március 1.
Beadási határidő: 2022. 05. 15.

A szakdolgozat: Nem titkos.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Műszertechnikai és Kommunikációs Rendszerek Intézet

Kiadva: Budapest, 2022. 03. 21.

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
ELEKTRONIKAI ÉS KOMMUNIKÁCIÓS RENDSZEREK INTÉZETI
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI TANSZÉK

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2022.05.12.

.....
hallgató aláírása

Tartalomjegyzék

1. Bevezetés

A feladat célja: Készítsen egy turbinás spirométert, illetve végezze el a hozzávaló androidos szoftver tervezését, gyártását. A turbinás spirométer alap spirometriai méréseket végez, melyeket real-time lehet ábrázolni a szoftverben. A mérőfej, illetve a telefon Bluetooth kapcsolaton keresztül kommunikálnak egymással. A méréshez szükséges a páciens neme, életkora, testsúlya, magassága, ebből számolja ki a szoftver az ideális görbét.

2019 év végén a kínai Wuhan városában egy új betegség, a SARS-CoV-2 által okozott Covid-19 jelent meg. A fertőzés gyors terjedése azt eredményezte, hogy az Egészségügyi Világszervezet (WHO) globális világjárvánnyá nyilvánította a Covid-19-et.

A betegség többnyire felső légúti megbetegedéseket okoz, tünetei között szerepel az influenzaszerű megbetegedés (magas láz, köhögés, torokfájás, nátha), amely a magas kockázatú egyéneknél súlyosabbá is válhat. A kórházba került betegek egy részénél oxigént kell adni, ezzel is segítve a légzésüket. A betegség legyőzése után még fennállhatnak különböző tünetek, melyek monitorozására - amennyiben légzési nehézség áll fenn - kiváló eszköz a spirométer. Szakdolgozatomban próbáltam megvalósítani egy olyan, akár otthon is használható orvostechnikai eszközt, amellyel a páciens a saját otthonából is nyomon tudja követni a saját tüdőkapacitásának változását.

A spirométert gyártóknál is egyik fő szempont a hordozható kivitel, és az ergonomikus kialakítás. Ebből fakadóan döntöttem amellett, hogy egy Androidos programot hozok létre, mely Bluetooth kapcsolaton keresztül tud kommunikálni a mérőfejjel. A mért adatok segítséget nyújtanak az orvosoknak a diagnózis kialakításában. Szakdolgozatom készítése közben igyekeztem elsajátítani a Kotlin programozási nyelvet.

2. Elméleti háttér

A sejtek fennmaradásához szükség van arra, hogy oxigént juttassunk el hozzájuk és széndioxidot szállítsunk el tőlük. Ezt a keringési és a légzési rendszer végzi el. A két rendszer a tüdő alveolusaiban kapcsolódik egymáshoz. A tüdő az ún. külső légzést végzi, a keringési rendszer pedig az ún. belső légzést. A légzés vizsgáló készülékek a külső légzés paramétereit vizsgálják: áramlási sebességet, ki vagy belégzett térfogatot, légzésmechanikai jellemzőket és a ki vagy belégzett levegő összetételét. [1]

A spirometria egy olyan légzésfunkciós vizsgálat, amellyel vizsgálhatjuk a tüdő térfogatát, a légzés minőségét, valamint a gázcsere és a keringés állapotát. A légzésfunkciós vizsgálat (spirometria) segítségével a tüdő működéséről kaphatunk átfogó képet. A vizsgálatot általában akkor írja elő az orvos, amikor a tünetek asztmára, krónikus obstruktív tüdőbetegségre (COPD), esetleg tüdőhegesedésre (fibrózis) utalnak, de elvégzésére légúti betegségek állapotának követésénél is szükség van. A spirometria rendkívül fontos az általános légúti egészség szűrővizsgálataként, ugyanúgy, ahogy a vérnyomás is fontos információkat szolgáltat az általános szív- és érrendszeri egészségről. [2]

2.1 Légzőrendszer anatómiája

A tüdő egy olyan páros szerv, melynek fő funkciója a gázcsere. A mellüregben a szív, a nagyerek és az oesophagus által szabadon hagyott teret tölti ki. A tüdőt a tüdőkapuk kivételével a mellhártya két lemeze borítja. A két mellhártyalemez a tüdőkapunál hajlik át egymásba, ahol a hörgők, az erek és idegek lépnek be a tüdőbe. A mellhártyalemezek között virtuális rés van, melyet vékony folyadékfilm tölt ki.

2.1.1 Tüdő vérellátása

A tüdő vérellátása két rendszeren keresztül történik. A jobb kamrából vénás vért hozó arteria pulmonalis elágazódva követi a bronchusokat és a bronchiolus terminalis szintjén mintegy 0,1-0,15 mm átmérőjű arteriolákká válnak. További elágazódásukkal végül is az alveolusfalakat körülhálózó, azzal igen nagy területen érintkező capillaris hálózatot alkotnak, amely a gázcserét szolgálja. [3]

2.2 A légzés élettana

A légzés vitális élettani funkció, az élet fogalmától elválaszthatatlan, megszűnése a klinikai halált jelenti. Elsődleges feladata az O_2 és CO_2 kicserélődésének folytonos fenntartása a tüdőkben, az alveolo-capillaris membránon keresztül. A külső gázcsere passzív folyamat, nagysága az alveolaris légtér és a tüdőkbe visszatérő kevert vénás vér gáztenzióinak eltérésétől függ. A megfelelő nyomásgrádienseket a ventilációs és perfúziós pumpák összehangolt működése tartja fenn. A ventilációs pumpa a tüdők légterének friss levegővel való átszellőztetésével tartja fenn alveolaris szinten a magas O_2 és az alacsony CO_2 nyomásokat. Ezzel egyidőben a jobb szívkamra alacsony O_2 és magas CO_2 parciális nyomású kevert vénás vért pumpál át a pulmonalis capillarisokon. [3]

2.3 Légzésfunkciós vizsgálatok

A légzőrendszer mechanikai teljesítménye és a működésében bekövetkezett zavarok légzésfunkciós vizsgálatokkal jól mérhetőek. Ezek a vizsgálatok napjainkban már nem nélkülözhetőek a légutak és a tüdő betegségeinek diagnosztikájában, kórlefolyásuk nyomonkövetésében, és a kezelés hatékonyságának, a funkcionális állapot kedvező, vagy kedvezőtlen változásának megállapításában.

A legegyszerűbb készülékekkel a betegek saját maguk is megfigyelhetik, nyomon követhetik pl. a légúti kaliberük változását, és az adatok feljegyzésével nemcsak az orvosnak szolgáltatnak hasznos információkat, adatokat, hanem saját maguk is a kezelést a mérési eredmény szerint, azaz objektív paraméterekkel jellemzett állapotukhoz igazíthatják, természetesen csak bizonyos határokon belül.

Napjainkban a légzésfunkciós vizsgálatoknak a légúti- és tüdőbetegségek diagnosztikájában legalább akkora jelentősége van, mint az EKG-nak a kardiológiában. Gyakorlati szempontból a tüdőfunkció zavarai az alábbi csoportokba sorolható:

- ventiláció (külvilág és a tüdők légtere közötti folytonos gázáramlás) zavara,
- disztibúció zavara (ventiláció és perfúzió illeszkedésének zavara),
- perfúzió (tüdőcirkuláció) zavara,
- diffúzió zavara.

A ventilációs vizsgálatára szolgáló mutatókat spirometriás és légzésmechanikai, valamit statikus és dinamikus paraméterekre szokás felosztani. A légzésmechanikai paraméterek a

nyomás, a nyomás-áramlás és a térfogat-nyomás viszonyokat mutatják. A dinamikus paraméterekkel szemben a statikus paraméterek nem függenek az időtényezőtől.

2.3.1 Dinamikus térfogatok

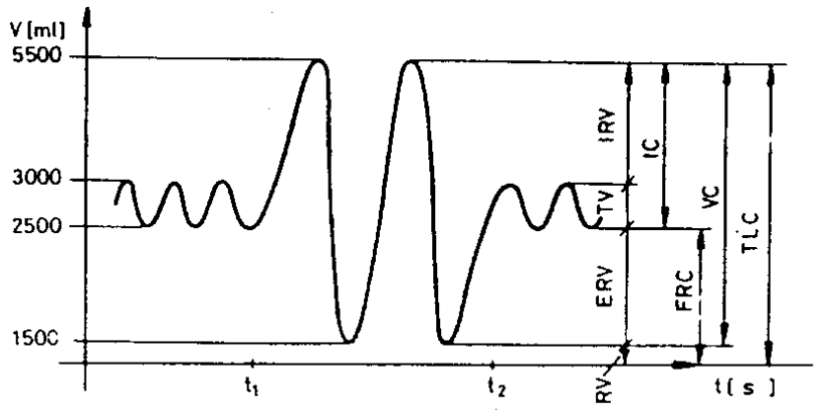
Az időtényezőtől függő térfogatok közül az erőltetett kilégzési vitálkapacitás a maximális belégzés, azaz a totálkapacitás szintjétől indított erőltetett kilégzéssel kifújható levegőmennyiség (FVC).

A teljes belégzés után indított erőltetett kilégzési manőver első másodpercére eső kilégzési térfogatot erőltetett kilégzési másodperctérfogatnak (FEV_1) nevezzük.

A beteg által a lehető legerőteljesebben (forszírozottan) végzett kilégzés során a levegő áramlási sebességét méri a csúcsáramlás (PEF). [3]

2.4 Áramlási sebesség és ki-belégzett térfogat mérése

A tüdő térfogatának változását a légzés során az 1. ábra mutatja. Ezen látható, hogy néhány nyugalmi légzést követően két erőltetett be-, illetve kilégzés történt, majd utána ismételten két nyugalmi légzés következett. Az erőltetett belégzésnél a maximális értékig lélegzett be, míg a kilégzésnél erőltetve kipréselte a levegőt a tüdejéből a páciens.



1. ÁBRA: A TÜDŐ TÉRFOGATÁNAK VÁLTOZÁSA A LÉGZÉS SORÁN

Mérni az áramlási sebességet szokás, ebből számítjuk ki a be-, illetve kilégzett levegő térfogatát integrálással. A levegő áramlási sebességét több módszerrel lehet mérni, ezek közül a nyomáskülönbség mérés, az ultrahang, illetve a turbinás spirométer a legelterjedtebb napjainkban.

Az áramlási sebesség integrálásával előállítható a ki/belégzett térfogat időfüggvénye. Figyelembe kell azonban venni, hogy a belégzett és a kilégzett levegő paraméterei eltérőek. A belégzett levegőre jellemző: 20 ... 25 °C hőmérséklet, 40 ... 70 %-os páratartalom és az atmoszférikusnál kisebb nyomás, míg a kilégzett levegő közel 37 °C hőmérsékletű, 100 %-os páratartalmú és az atmoszférikusnál nagyobb nyomású. Az átszámítást BTPS (body temperature pressure saturated) korrekciónak hívják. [1]

3. Spirométerek fajtái

A spirométereknek több fajtáját különböztetjük meg, ezek közül a három leggyakoribb:

- turbinás spirométer,
- nyomáskülönbség elvén működő spirométer,
- illetve az ultrahangos spirométer.

3.1 Turbinás spirométer

A turbinás spirométer esetében egyszer, vagy többször használatos turbina segítségével mérik a paramétereket. A páciens egy eldobható csutorán keresztül lélegez ki-be a készülékbe. A légzés hatására a csőben lévő turbinalapát forog, ezt a forgást mérik, ebből számolják ki a megfelelő pulmonológiai értékeket. Előnyük, hogy kis méretűek, így könnyen kézben tarthatóak, a turbina cseréje után nem kell újra kalibrálni a készüléket. Hátránya a pontossága, hiszen a turbinalapát áramlás nélkül is képes forogni, ezzel mérési hibát okozva. Turbinás spirométer esetében piacvezető az olasz MIR cég. [4]

2. ÁBRA: MIR TURBINÁS SPIROMÉTER

3.2 Nyomáskülönbség elvén működő spirométer

Egy készülék, amely egy Pitot-csőből és egy gyűrű alakú csőből áll, statikus nyomásnyílásokkal kombinálva. A két port közötti nyomáskülönbség a sebességfej. A nyomáskülönbség-jeladót a két port közötti nyomáskülönbség mérésére használják. A

sebességnek ez a jelzése a cső keresztmetszeti területével együtt jelzi az áramlási sebességet. A Pitot-csöves áramlásmérők akár folyadékokat, akár gázokat képesek mérni. Előnyük, hogy mozgó alkatrészektől mentes, érzéketlen a párára és nedvességre, az áramlási cső cseréje után nem kell újrakalibrálni a készüléket. Magyarországon a Piston Kft. által gyártott PinkFlow: PPF-18 készülék használja ezt az alapelvet. [5]

3. ÁBRA: PINKFLOW SPIROMÉTER

3.3 Ultrahangos spirométer

Ultrahangos spirométer esetében az áramlási csőben ultrahangos szenzorok helyezkednek el egymással szemben (adó, illetve vevő). Az ultrahangos impulzusok átmeneti időbeli különbségei alapján lehet számolni a megfelelő paramétereket, amennyiben légáramlás van. Előnyük: kis méret, nagyon pontos mérést lehet elvégezni velük, könnyen kezelhető, nincs mozgó alkatrész. Hátrányuk a tisztitásban jelentkezik, hisz bár ajánlott megfelelő szűrő használata, viszont a gyártó által meghatározott időközönként fertőtleníteni kell a készüléket. A svájci érdekeltségű NDD 1996 óta fejleszt ultrahangos spirométereket. [6]

4. ÁBRA: NDD SPIRORÉTER

4. Specifikáció

4.1 Hardver specifikáció

Turbinás spirométer létrehozása, mely LEDek, illetve fototranzisztorok segítségével méri a levegő áramlási sebességét, ezt egy ESP32 mikrokontroller dolgozza fel, és Bluetooth kapcsolaton keresztül továbbítja az androidos telefonon futó programba. Az áramlási sebesség integrálásával kiszámítható az átáramlott levegő térfogata.

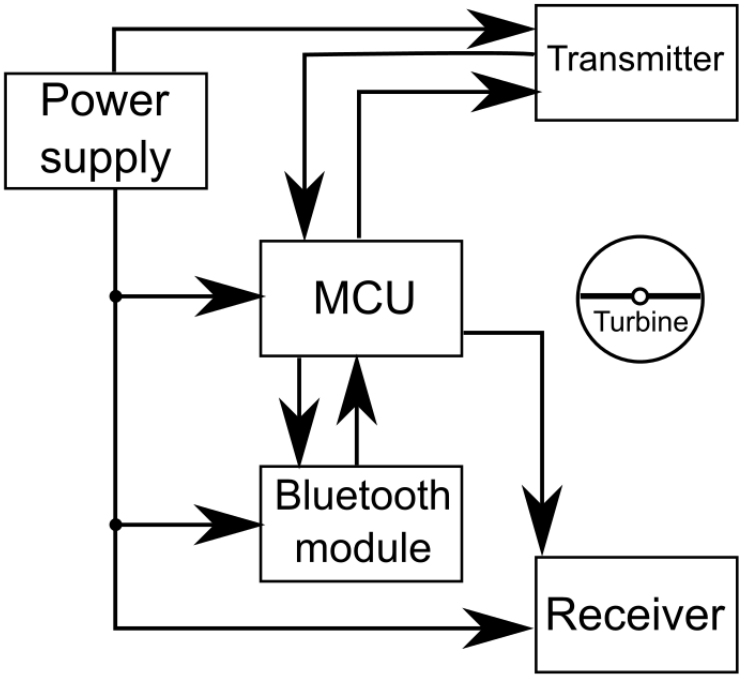
4.2 Szoftver specifikáció

Androidos szoftver, mely Bluetooth kapcsolaton keresztül tudja kezelni a mikrokontroller által küldött adatokat. A szoftver megírásához Kotlin programozó nyelvet használtam. A program adatbázissal rendelkezik, melybe a páciens adatait lehet menteni (név, születési dátum, etnikai csoport, biológiai nem, azonosítószám), illetve a méréshez szükséges adatokat lehet felvinni (testtömeg, magasság, dohányzás fajtája, évei). A szoftver valós időben jelzi ki az alap spirometriai tényezőket (FVC, FEV1), melyek később akár bővíthetőek lehetnek további paraméterekkel is. A készülék csak kilégzési paramétereket mér.

5. Logikai rendszerterv

5.1 Hardver rendszerterv

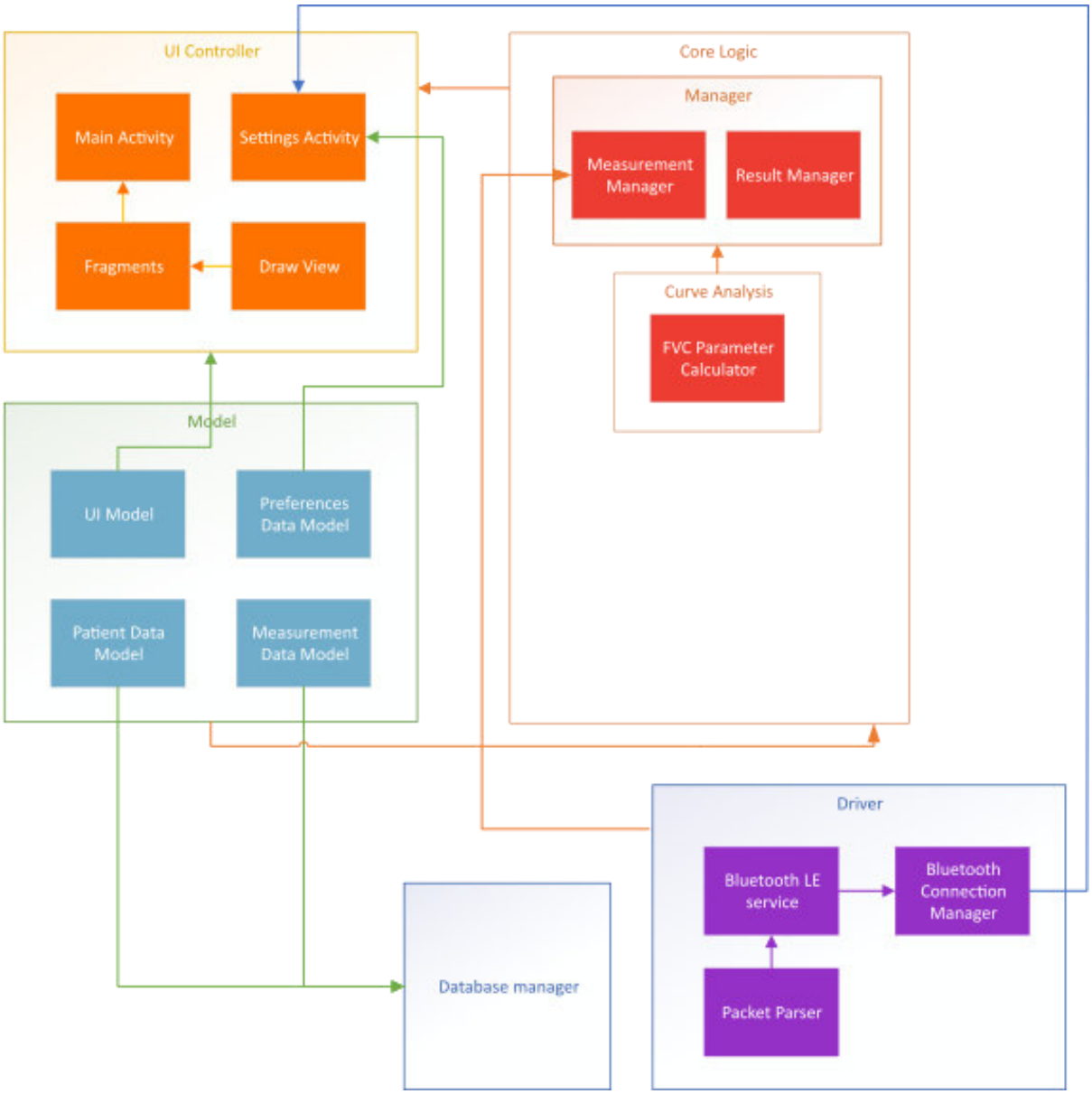
A készülék hardverének a logikai rendszerterve a 5. ábrán látható.



5. ÁBRA: HARDVER LOGIKAI RENDSZERTERVE

A készülék egy ESP32 mikrokontrollert használ, beépített Bluetooth-szal. A turbina forgásának jelét két adó (Transmitter), illetve két vevő (Receiver) szolgáltatja. Adóként két LED, míg vevőként két fototranzisztor működik.

5.2 Szoftver rendszerterv



6. ÁBRA: SZOFTVER LOGIKAI RENDSZERTERVE

6 Spirométer megvalósítása

Az első lépések a turbinás szenzor alapú spirométer fejlesztésében, magának a szenzornak a kiválasztása, tesztelése (mérésekkel), modell keresése és végül az eredmények kiértékelése: mennyire is felel meg az elvárásaiknak.

Az általam választott szenzor online könnyen elérhető, piaci ára alig 2-3 dollár, ami nagy tétel esetében tovább csökken.

A turbinás szenzorok nem újkeletűek. Az egyik legnagyobb spirometriával foglalkozó cég – MIR – spirométerei ezt a technológiákat használják, viszont igen költségesek, és viszonylag korlátozott képességekkel rendelkeznek, nem feltétlenül állják meg a helyüket a pediátria területén. Bizonyos helyeken megjelennek az ultrahangos technológiával működő eszközök. Ezek sokkal pontosabbak, mint turbinás társaik, azonban a mérési technológia sokkal drágább, az eszköz kevésbé robosztus. A szenzorok cseréje sokkal nehezebb, mint kicserélni egy turbinát. Egy másik fontos tényező, hogy nem szükséges az eszköz kalibrációja, ami megint csak sok drága időt spórolhat meg mind az orvosnak, mint a gyártóknak.

A fejlesztés során használt turbina áttetsző műanyagból készül. Ezt az áttetszőséget használjuk ki a turbina lapátsebességének mérésére (*7. ábra*).

7. ÁBRA: TURBINA

6.1 Turbina tesztelés

A turbina teszteléséhez fototranzisztort, illetve infra LED-et alkalmazunk. Az első mérési elrendezést a turbinából származtatható jelek felderítésére használtam.

A fényforrást, illetve a detektort úgy helyeztem el, hogy a turbina közepében lévő lapát a fény útját megszakítsa. Ezen megszakítások megfelelő detektálásával következtethetünk a fordulatszámra, amit a továbbiakban áramlásméréssé szeretnék konvertálni.

6.2 Első mérési elrendezés

Az első mérések során, arról meg tudtam győződni, hogy a szenzorok működnek és az elrendezés jó. A mérést oszcilloszkóp segítségével végeztem. A jeladók táplálása után a turbinát mozgásba lendítve, sikerült 1 volt amplitúdóval rendelkező változó jelet kimérni. Ez a jelalak szolgáltatja a tervezés további lépéseit. A nehézséget az adja, hogy a jelalak szélessége váltakozó, hiszen minél magasabb frekvencián pörög a turbina, a forma szűkebb lesz (növekszik a frekvenciája), annál többször történik változás, azaz, hol látja a vevő az adót hol nem. Ennek a jelnek a mérése több nehézséget is felvet, hiszen ezt a jelet a későbbiekben linearizálni kell, hogy mérhető, használható értéket adjon.

Továbbiakban említésre méltó, hogy a turbinában lévő lapátnak tehetetlensége van. Ez azt jelenti, hogy befújás után, nem áll meg, hanem a lendülete viszi, pörgeti tovább. Erre a problémára a későbbiekben szoftveres megoldás javasolt, ami az itt keletkező hamis adatok kiszűrésére képes.

Fontos kiemelni, hogy a mérés során csak egy szenzorpárt (egyetlen adó-vevő párt) használtam. Ezzel a megoldással is mérhető tartományban lévő jelet kaptam, de a turbinának csak a forgását tudtam érzékelni. Ahhoz, hogy a rendszer képes legyen megállapítani az áramlás irányát, szükségem volt még egy szenzor párra, mely könnyedén elhelyezhető az eszközön. A forgásirányt a két szenzor által mért jelek egymáshoz viszonyított eltéréséből tudom számolni. A magas precizitás elérése érdekében figyelembe kell venni újból a szenzor tehetetlenségét.

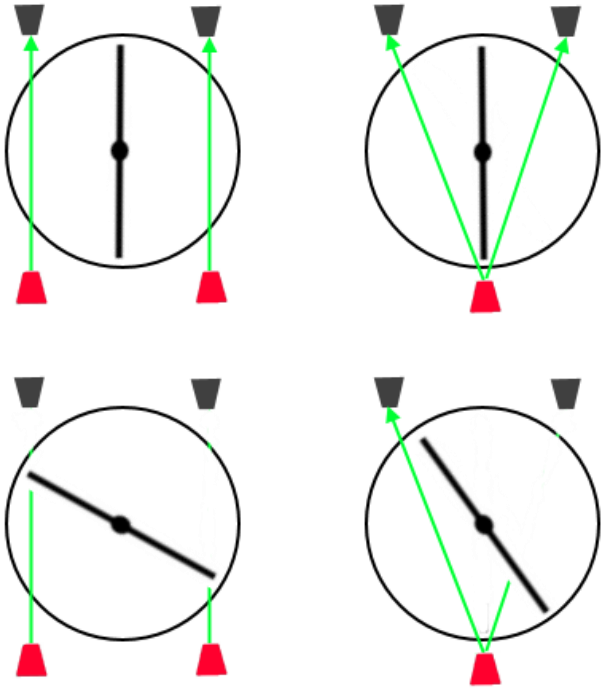
6.3 Második mérési elrendezés

A szenzor működési elvének bizonyítása után következett egy, a turbinához jobban illeszkedő, és a végleges megoldáshoz jobban hasonlító szenzor megalkotása. A cél egy olyan 2 adó-vevő párból álló szenzor kialakítása volt, ami körülöleli a turbinát. Ezzel az elrendezéssel további méréseket végeztem, melyek most az irány meghatározására irányultak.

A kapott eredményekből megállapítható, hogy az adó és vevő párok „fény egyenesei” által bezárt szögtől függő paraméter az egymáshoz viszonyított késés. Ezt az időbeli különbséget szeretném felhasználni irány detektálásra. Nagyobb frekvenciákon ez az idő könnyedén összemehet, így a következő kialakításoknál fontos ennek megnövelése a bezárt szög megfelelő növelésével.

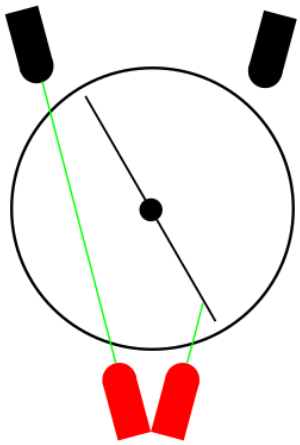
6.4 Harmadik mérési elrendezés

A szenzor továbbfejlesztési lehetőségei közt felmerült, egy újabb kialakítás. Ennek oka egy megfelelőbb kialakítás a szenzorok elhelyezkedését tekintve (*8. ábra*).



8. ÁBRA: SZENZOR KIALAKÍTÁS

A *8. ábrán* jól kivehető, hogy ha a bezárt szög közelít a 0°-hoz akkor a lapát közel egyszerre (bal oldali ábra) szakítja meg a fénysugarakat, míg a második ábrán ez jóval előbb következik be. Ezek alapján alakítottam ki a végső megoldást (*9. ábra*).



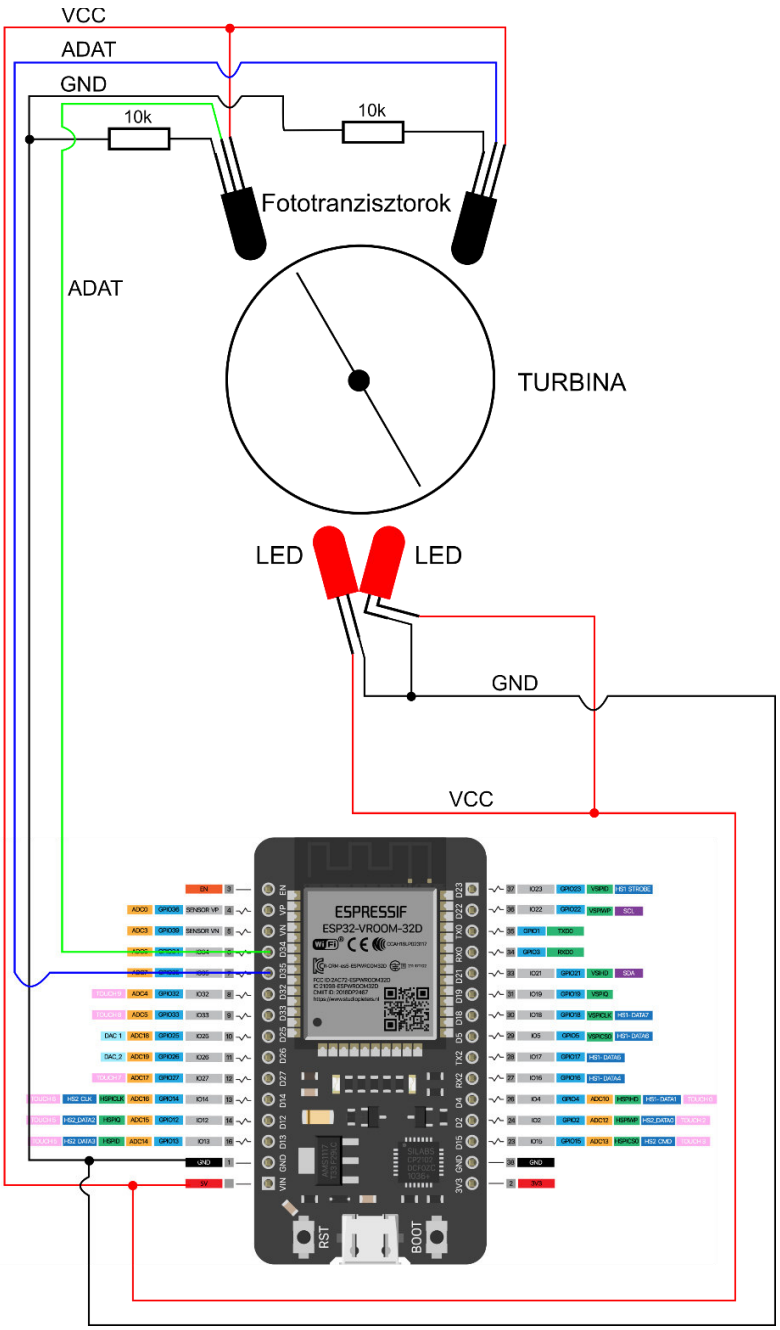
9. ÁBRA: VÉGLEGES SZENZORKIALAKÍTÁS

6.5 Fizikai rendszerterv

A spirométer fizikai rendszertervét a 10. ábra szemlélteti.

Felhasznált alkatrészek:

- NodeMCU ESP32S dual
- TSUS5400 (CQY99) infra LED (5mm, 950nm)
- BPV11F fototranzisztor (5mm, 950nm)



10. ÁBRA A SPIROMÉTER FIZIKAI RENDSZERTERVE

7. Spirometriai szoftver megvalósítása

A szoftver fő célja, hogy kapcsolatot hozzon létre a turbinás spirométer készülékkel, és lehetővé tegye a felhasználó számára az FVC mérések elvégzését. A kiszámított légzésdiagnosztikai paraméterek megjelennek a felhasználó számára.

A szoftvert Android Studio fejlesztői környezetben Kotlin nyelven írtam. A Kotlin egy 2011-ben megjelent modern programozási nyelv, amely a Java alternatívája, amellyel zökkenőmentesen együtt tud létezni. A szakirodalomban számos bizonyíték található arra, hogy a Kotlin egyre nagyobb teret nyer az Android szoftverfejlesztők körében.

Az egyik fő tervezési irányelv, amely a Kotlin nyelv fejlesztéséhez vezetett, a null értékek jobb kezelése. A Java nyelvben a null értékek kezelése speciális ellenőrzések használata nélkül NullPointerExceptionokhoz (NPE) vezethet. A szakirodalomban több tanulmány is beszámol a NullPointerExceptions kiemelkedő szerepéről az Android-alkalmazások összeomlásának okai között.

Az olvashatóság és a tömörség a Kotlin nyelv kulcsfontosságú jellemzői, különösen a számos attribútummal rendelkező objektumok és osztályok deklarációját illetően. A Kotlin Android-fejlesztésre történő egyre növekvő elfogadásának fényében ez az empirikus értékelés olyan gyakorlati bizonyítékokkal kíván szolgálni, amelyek segíthetnek a Java-ról a Kotlinra való átállásban. Különösen a karbantarthatóságra, a tömörségre és néhány gyakori buktató elkerülésére gyakorolt lehetséges hatásokra összpontosítottunk. [7]

A megvalósítás első lépéseként át kellett gondolni, hogy milyen struktúrában szeretném megvalósítani a szoftvert. Ennek következtében több felhasználói felületet használtam, gombokkal, szövegbox-okkal, illetve legördülő menüvel. Első körben igyekeztem Activity osztályt használni. Az Activity egyetlen, célzott dolog, amelyet a felhasználó elvégezhet. Szinte minden Activity interakcióba lép a felhasználóval, ezért az Activity osztály gondoskodik arról, hogy létrehozzon egy ablakot, amelyben a setContentView(View) segítségével elhelyezheti a felhasználói felületét.

7.1 Gombok

Ezen a felületen keresztül könnyen lehetett adatokat beírni, illetve nyomógombokat elhelyezni, azokkal különböző interakciókat elvégezni. Ahhoz, hogy két Activity class között „mozoghassak”, a következő kódrészletet kellett alkalmaznom:

```
val button3: Button = findViewById(R.id.button3)
    button3.setOnClickListener {
        val intent = Intent(this, NewDataQuick::class.java)
        startActivity(intent)
    }
```

7.2 Activity-ből Fragment-be átlépés

Amikor a PatientPage-ből próbáltam átlépni gomb hatására a ListFragment-re, akkor a program összeomlott. Ezért más kódot kellett használnom ezekhez a műveletekhez:

```
val buttonOpen : Button = findViewById(R.id.button2)
    buttonOpen.setOnClickListener {
        val myFragment = ListFragment()
        val fragment : Fragment? =
        supportFragmentManager.findFragmentByTag(ListFragment::
class.java.simpleName)
        if (fragment !is ListFragment){
                supportFragmentManager.beginTransaction()
                .add(R.id.LinearFragment_Container, myFragment,
ListFragment::class.java.simpleName)     //1. kép
                .commit()
        }
        buttonOpen.visibility = View.GONE
    }
```

7.3 Időzítő használata

Ezzel az adott lapról sikerül átlépni a NewDataQuick lapra. Szerettem volna megvalósítani egy Splash képernyőt is, azaz egy adott ideig megjelenik az ablak, majd átvált a következőre. Ekkor a következő kódrészletet kell alkalmazni.

```
Handler().postDelayed({
        val intent = Intent(this@SplashScreen, PatientPage::class.java)
        startActivity(intent)
        finish()
    }, 5000)
```

Jelen esetben 5 másodpercig jelenik meg a Splash képernyő, majd továbblép a PatientPage oldalra.

7.4 Adatbázis létrehozása

Az első komolyabb munka az adatbázishoz, illetve a Bluetooth kapcsolathoz köthető. Az adatbázisnál Room Adatbázist használtam. Ekkor külön csomagokat készítettem a főmappán belül. Első körben elkészítettem a data Package-et. Ebbe 4 osztály (User, UserDatabase, UserRepository, UserViewModel), és egy interfész (UserDao) került kidolgozásra.

A User class-ba felvittem az összes adatot, amelyet szeretnék az adott Felhasználóhoz elmenteni:

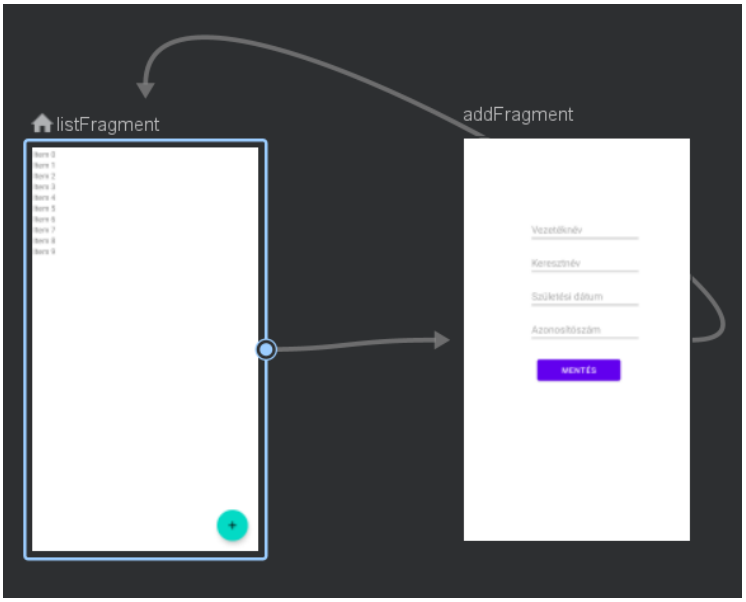
- ID,
- Vezetéknév,
- Keresztnév,
- Születési dátum,
- Azonosítószám.

Az ID automatikusan növekedő szám, minden felvitt adat kap egy sorban következő számot 1-től indulva.

A UserDao kezeli az adatokat, ezen keresztül az alkalmazás le tudja kérdezni, frissíteni, illetve törölni tudja a felhasználó által beírt adatokat. Én a frissítést nem használtam, mivel, amennyiben a megadott adatok nem megfelelőek, egyszerűen törölni lehet, és újra generálni azokat.

A UserDatabase dolga az adatbázis tárolása. Ez nem sima osztály, a UserDatabase-nek absztrakt osztálynak (abstract class) kell lennie.

Miután megtörtént az adatbázishoz az adatokkal kapcsolatos műveletek implementálása, elkészítettem 3 Fragment-et (AddFragment, ListFragment, ListAdapter). Első lépésként létre kellett hozni egy Navigációt a Resource Manager fül alatt található Navigation menüpont alatt (11. ábra).



11. ÁBRA: NAVIGÁCIÓS ABLAK

Itt két Fragment-et kötöttem össze, a ListFragment-et, illetve az AddFragmentet. A ListFragment-ben olvasható az adatbázis, míg az AddFragment lapon lehet az adatokat felvinni, és elmenteni:

```
private fun insertDatatoDatabase() {
    val firstName = addfirstName_et.text.toString()
    val lastName = addlastName_et.text.toString()
    val persID = addpersID_et.text
    val date = adddate_et.text
```

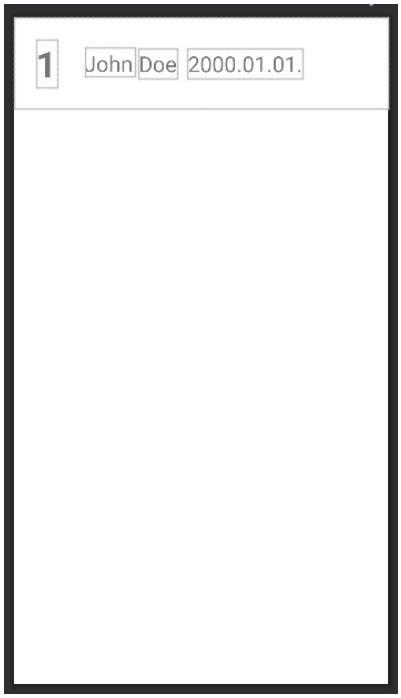
Amennyiben sikeres az adatmentés, a szoftver kiírja, hogy ’Sikeresen hozzáadva!’

```
mUserViewModel.addUser(user)
```

Toast.makeText(requireContext(), "Sikeresen hozzáadva!", Toast.LENGTH_LONG).show()

A ListFragment oldal jobb oldalán elhelyeztem egy Floating Action Button-t, mellyel meg lehet nyitni az AddFragmentet, és fel lehet vinni új felhasználót. Ez egy kör alakú gomb, amely az alkalmazás felhasználói felületének elsődleges műveletét indítja el.

Az adatbázis kinézetéhez létre kellett hozni egy xml kiterjesztést, nálam a custom_row néven szerepel. Itt meg kellett adni, hogy a ListFragment oldalon mit látszódjon az adott felhasználói adatokból. Nálam a program által legenerált ID, a felhasználó teljes neve és születési dátuma szerepel.



12. ÁBRA: CUSTOM_ROW

Itt is ConstraintLayout-ot alkalmaztam. A ConstraintLayout egy ViewGroup, amely lehetővé teszi a widgetek rugalmas elhelyezését és méretezését. Ezen belül meg lehet adni az egyes elemeknek az adatait:

```
<TextView
    android:id="@+id/lastName_txt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

android:layout_marginStart="6dp"

android:text="Doe"

android:textSize="24sp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintStart_toEndOf="@+id/firstName_txt"

app:layout_constraintTop_toTopOf="parent" />

Az adott TextView-nak van egy azonosítója (android:id), meg lehet adni az elrendezés szélességét, magasságát, hol kezdődjön(android:layout_marginStart), mit tartalmazzon (android:text), mekkora legyen a betűméret (android:textSize), vagy akár a másik szöveghez való viszonyát (app:layout_constraintStart_toEndOf), és még sok mást is.

Az adatbázis törléséhez is szükséges létrehozni egy xml kiterjesztést, azaz a delete_menu.xml-t.

<item android:id="@+id/menu_delete"

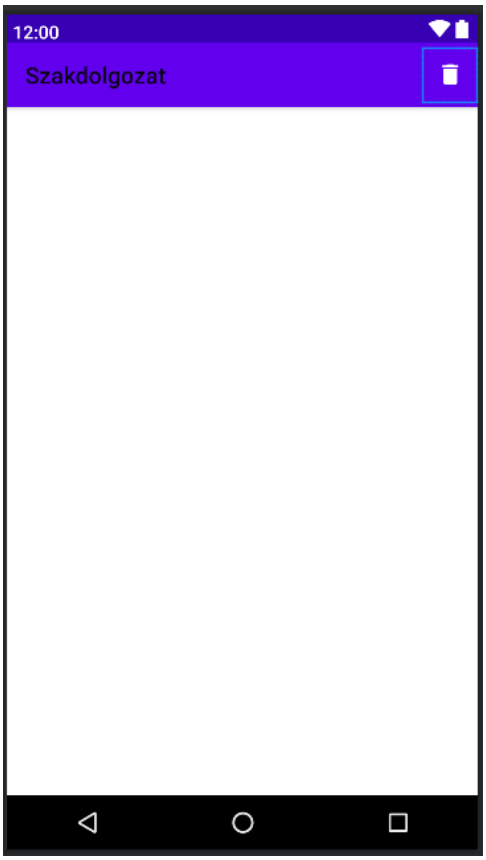
android:title="Delete"

android:icon="@drawable/ic_baseline_delete_24"

android:iconTint="@android:color/white"

app:showAsAction="ifRoom"/>

Itt határoztam meg a törlésnek az ikonját (ic_baseline_delete_24), mely a sokak által használt kuka ikon.



13. ÁBRA: TÖRLÉS IKON

Új ikont a ’drawable’ mappán belül lehet létrehozni. Jobb egér gomb → Új → Drawable Resource File. Egy vector típusú ikont hoztam létre.

7.5 Legördülő menü

A ’values’ mappán belül lévő string.xml fájlban meghatározhatjuk az adott string azonosítóját, és a hozzá tartozó adatokat, esetünkben:

```
<string-array name="etGroup">
    <item> </item>
    <item>Kaukázusi</item>
    <item>Közel-keleti</item>
    <item>Kínai</item>
    <item>Japán</item>
```

```
    <item>Polinéziai</item>
    <item>Észak-indiai</item>
    <item>Dél-indiai</item>
    <item>Pakisztáni</item>
    <item>Afrikai eredetű</item>
    <item>Afroamerikai</item>
    <item>Mexikói-amerikai</item>
  </string-array>
```

Legördülő menü esetén az adott lap xml kiterjesztésében Spinner-t használtam.

```
<Spinner
    android:id="@+id/spinner4"
    android:layout_width="100dp"
    android:layout_height="40dp"
    android:layout_marginTop="400dp"
    app:layout_constraintEnd_toEndOf="@+id/editTextDate2"
    app:layout_constraintTop_toTopOf="parent" />
```

Az adott lap ’.kt’ kiterjesztésében meg kell határozni, hogy melyik azonosítójú string-et használja a program:

```
val spinner2: Spinner = findViewById(R.id.spinner4)
    ArrayAdapter.createFromResource(
        this,
        R.array.etGroup,
        android.R.layout.simple_spinner_item
    ).also { adapter ->
```

```
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_ite
m)

            spinner2.adapter = adapter

        }
```

7.6 Bluetooth kapcsolat

A Bluetooth kapcsolathoz két Class-t hoztam létre, a ControlActivity-t és a SelectDeviceActivity-t. Az AndroidManifest.xml fájlba be kell illeszteni a Bluetooth protokollokat:

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

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

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

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

            <uses-permission   android:name="android.permission.BLUETOOTH_CONNECT"
/>
            <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

A SelectDeviceActivity class-ben 4 változót használtam:

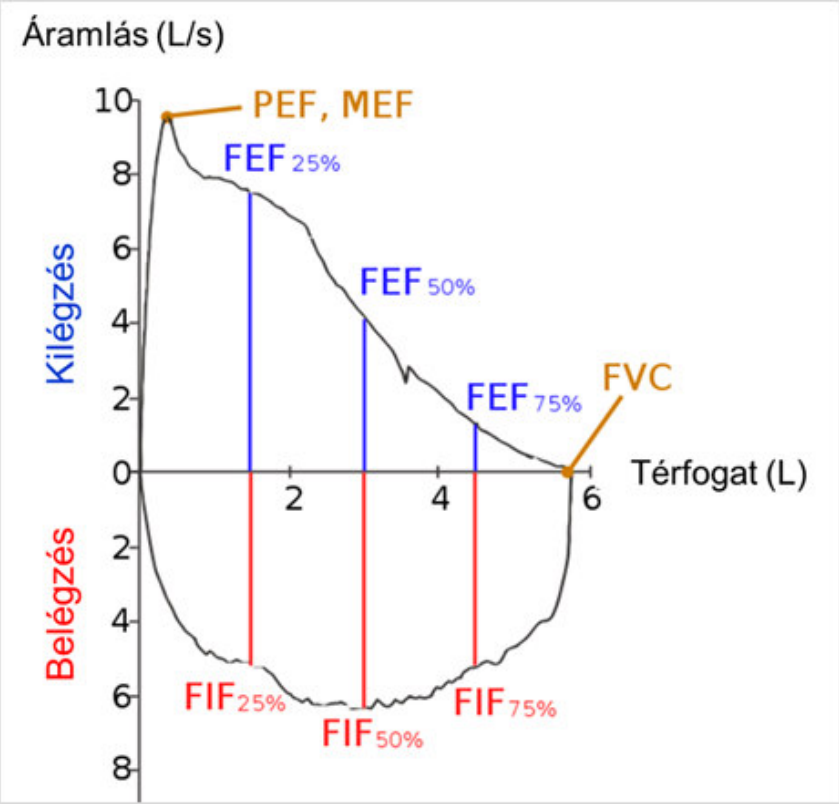
- m_bluetoothAdapter
- m_pairedDevices
- REQUEST_ENABLE_BLUETOOTH
- EXTRA_ADDRESS

Az EXTRA_ADRESS egy ’companion object’-ben adtam meg. A ’companion object’ olyan változók vagy függvények definiálására szolgálnak, amelyek fogalmilag egy típushoz kapcsolódnak, de nem kötődnek egy adott objektumhoz. A társobjektumok hasonlóak a Java statikus kulcsszavának használatához a változók és metódusok esetében.

A SelectDeviceActivity-ben az elérhető Bluetooth eszközök jelennek meg, a Frissítés gombra kattintva. Ezek után lehet kiválasztani a megfelelő eszközt, amihez csatlakozni szeretnénk.

7.7 Mérési eredmények

A szoftver 3 spirometriai értéket mér, FVC, FEV1, és PEF. A hardware meghatározza a sebességet, melynek az integráltja a térfogat, ebből számolható ki az FVC értéke. A FEV1 a kilégzés első másodpercének térfogatadata, míg a PEF az áramlás csúcsértéke. A készülék csak kilégzést mér, a belégzési adatokkal nem foglalkozik. A belégzés azért fontos, mivel a lapát mozgása megfordul, így van egy pillanat, amikor a sebessége nulla, ekkor vált át kilégzésből belégzésbe. A pontosabb méréshez, illetve a többi adathoz szükséges a predikciós algoritmus használata.



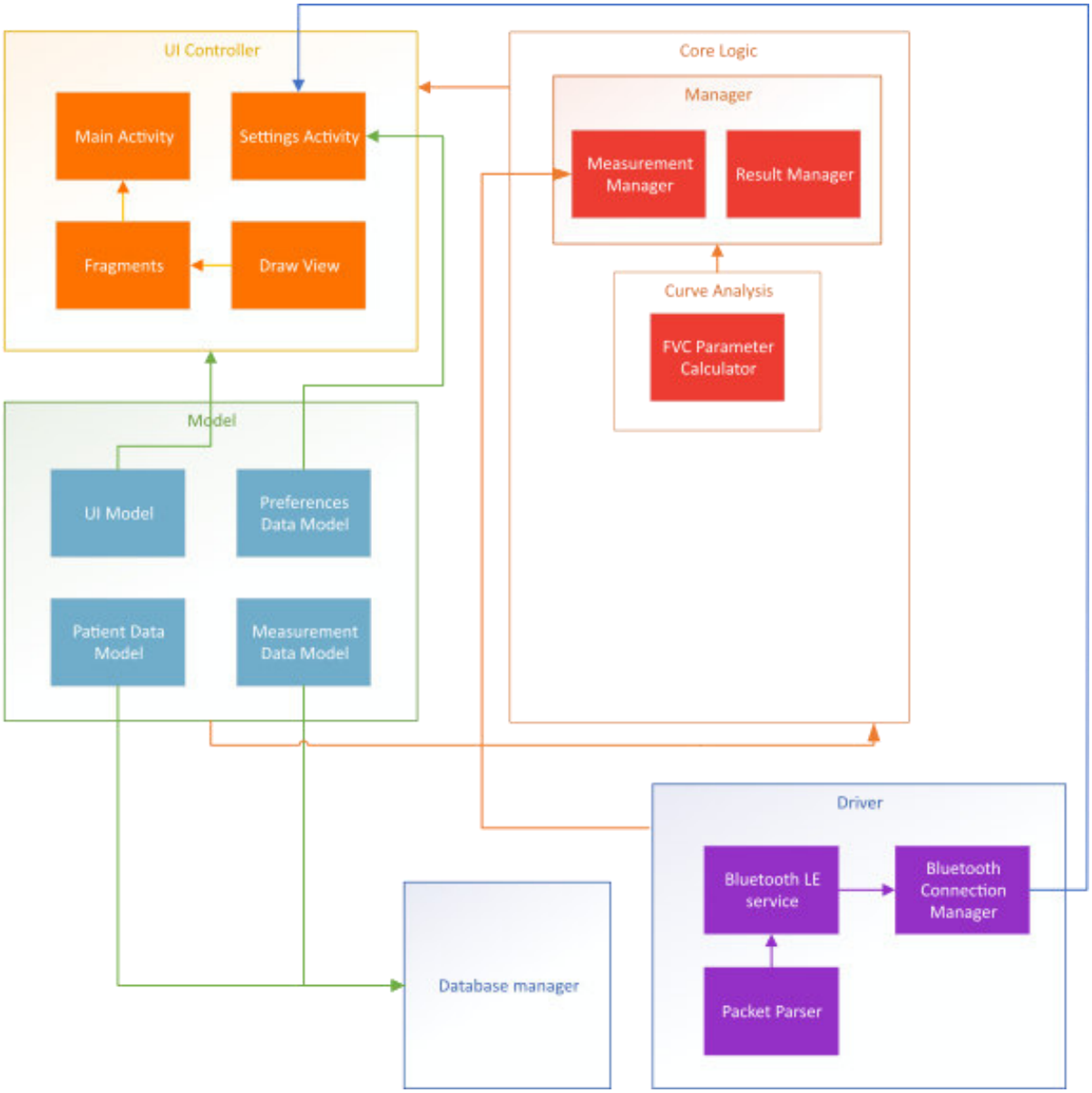
14. ÁBRA: FVC GÖRBE

8. Szoftver felépítése

8.1 Fogalommeghatározások, rövidítések

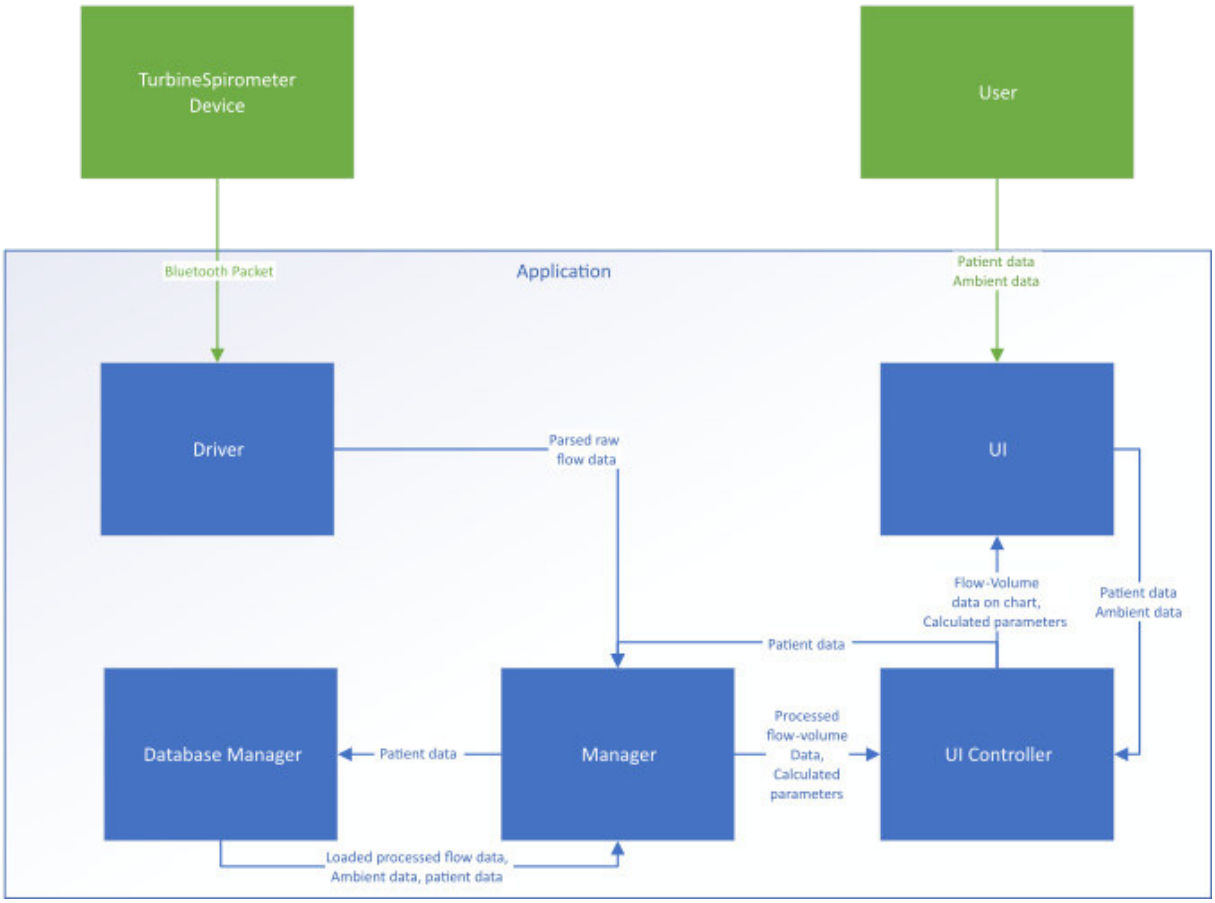
Fogalom	Meghatározás
API	Alkalmazási program interfész. Az API egy olyan módszer, amellyel egy alkalmazási program (egy teljes program, amely közvetlenül a felhasználó számára hajt végre egy adott funkciót) hozzáférhet a számítógép operációs rendszeréhez.
GUI	Grafikus felhasználói felület. A GUI grafikus felületet biztosít a felhasználók számára a rendszerrel való interakcióhoz.
Link	A link a képernyők közötti kapcsolat eszköze.
SDK	Szoftverfejlesztési készlet. Olyan programkészlet, amely lehetővé teszi a szoftverfejlesztők számára, hogy egy adott platformon futtatható termékeket hozzanak létre, vagy egy API-val dolgozzanak.

8.2 A szoftver blokkdiagramja



15. ÁBRA: SZOFTVER BLOKKDIAGRAMMJA

8.3 A szoftver adatáramlási diagramja



16. ÁBRA: SZOFTVER ADATÁRAMLÁSI DIAGRAMMJA

8.4 A funkciók és összetevők részletes leírása

A felhasználói felület (UI) elemeit a "material design standardok" szerint terveztem. A képernyőket és elemeket úgy terveztem, hogy a mobileszköz érintőképernyőjének segítségével gesztusokkal lehessen őket vezérelni.

A kezelőszervek közötti elrendezéseket (layouts) és helyőrzőket (placeholders) a használhatóság és a jó ergonómia jegyében terveztem. A szabványok betartása biztosítja a felhasználók számára a könnyű kezelhetőséget, és minimalizálja a lehetséges felhasználói beviteli hibákat.

8.4.1 Splash képernyő

Azonosítás	SplashScreen
Típus	Class/Activity
Cél	A kezdőképernyő bevezető információt nyújt a felhasználónak. Megjeleníti az alapvető adatokat, hogy megértse, milyen szoftver indult el.
Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • Főmenü képernyő / beteglista képernyő
Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Főmenü képernyő / beteglista képernyő (PatientPage)
Interfészek	A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Általános könyvtári hozzáférési követelmények.
Feldolgozás	Automatikus továbblépés a következő képernyőre a belső időzítő alapján.
Adat	Csak belső adatok. A kezdőképernyőn nincs szükség felhasználói adatokra.

8.4.2 Beteglista képernyő

Azonosítás	PatientPage
Típus	Class/Activity
Cél	A beteglista képernyő megnyit egy felületet egy meglévő beteg kiválasztásához, új beteg létrehozásához vagy egy gyors mérés elvégzéséhez szükséges adatok megadásához, illetve a Bluetooth kapcsolódáshoz.

Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • Új beteg képernyő (AddFragment) • Beteg képernyő (ListFragment) • Gyors méréshez adatok (NewDataQuick) • Bluetooth kapcsolódás (SelectDeviceActivity)
Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Splash képernyő
Interfészek	A linkek/elemek/felületi kezelőszervek a képernyő középső részén helyezkednek el. A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Általános és adatbázis könyvtári hozzáférési követelmények.
Feldolgozás	Manuális továbblépés a következő képernyőre gombnyomásra.
Adat	Belső adatok és felhasználói adatok a mögöttes adatbázisból.

8.4.3 Gyors méréshez adatok képernyő

Azonosítás	NewDataQuick
Típus	Class/Activity
Cél	A ’gyors méréshez adatok’ képernyő megnyit egy felületet a gyors méréshez szükséges adatok megadásához
Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • Gyors mérés (QuickMeasurement)
Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Beteglista képernyő (PatientPage)

Interfészek	A linkek/elemek/felületi kezelőszervek a képernyő középső részén helyezkednek el. A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Általános könyvtári hozzáférési követelmények.
Feldolgozás	Manuális továbblépés a következő képernyőre gombnyomásra.
Adat	Csak belső adatok. A képernyőn nincs szükség felhasználói adatokra.

A felhasználó a következő adatokat adhatja meg:

- Vezetéknév
- Keresztnév
- Születési dátum
- Azonosító szám

A Tovább gombra kattintva a rendszer továbblép a QuickMeasurement képernyőre, ahol további paramétereket lehet beállítani.

8.4.4 Gyors mérés képernyő

Azonosítás	QuickMeasurement
Típus	Class/Activity
Cél	A gyors mérés képernyő megnyit egy felületet a gyors méréshez szükséges adatok megadásához
Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • Mérés (MeasurementScreen)
Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Gyors méréshez adatok képernyő (NewDataQuick)

Interfészek	A linkek/elemek/felületi kezelőszervek a képernyő középső részén helyezkednek el. A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Általános könyvtári hozzáférési követelmények.
Feldolgozás	Manuális továbblépés a következő képernyőre gombnyomásra.
Adat	Csak belső adatok. A képernyőn nincs szükség felhasználói adatokra.

A felhasználó a következő adatokat adhatja meg:

- Magasság (cm)
- Tömeg (kg)
- Dohányzás (stringből választás: Igen-Nem)
- Dohányzás fajtája (stringből választás: Cigaretta, Szivar, Pipa, Elektromos cigaretta)
- Dohányzás éve

A Tovább gombra kattintva a rendszer továbblép a Measurement képernyőre, ahol a mérési eredmények jelennek meg.

8.4.5 Mérés képernyő

Azonosítás	MeasurementScreen
Típus	Class/Activity
Cél	A gyors mérés képernyő megnyit egy felületet a gyors méréshez szükséges adatok megadásához
Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • Beteglista képernyő (PatientPage)

Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Gyors mérés képernyő (QuickMeasurement)
Interfészek	A linkek/elemek/felületi kezelőszervek a képernyő középső részén helyezkednek el. A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Általános könyvtári hozzáférési követelmények.
Feldolgozás	Manuális továbblépés a következő képernyőre gombnyomásra.
Adat	Csak belső adatok. A képernyőn nincs szükség felhasználói adatokra.

8.4.6 ListFragment képernyő

Azonosítás	ListFragment
Típus	Class/Fragment
Cél	A ListFragment képernyő megnyit egy felületet, ahol a páciensadatbázis található.
Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • AddFragment képernyő (AddFragment)
Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Beteglista képernyő (PatientPage)
Interfészek	A linkek/elemek/felületi kezelőszervek a képernyő középső részén helyezkednek el. A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Adatbázis hozzáférési követelmények.

Feldolgozás	Manuális továbblépés a következő képernyőre gombnyomásra.
Adat	Belső adatok és felhasználói adatok a mögöttes adatbázisból.

8.4.7 AddFragment képernyő

Azonosítás	AddFragment
Típus	Class/Fragment
Cél	Az AddFragment képernyő megnyit egy felületet, ahol a páciensadatokat lehet feltölteni.
Alárendeltek	Ez a képernyő a következő képernyőkre mutató linkeket tartalmazza: • ListFragment képernyő (AddFragment)
Függő viszonyok	A következő képernyő erre a képernyőre mutat: • Beteglista képernyő (PatientPage)
Interfészek	A linkek/elemek/felületi kezelőszervek a képernyő középső részén helyezkednek el. A képernyőt úgy terveztem, hogy a mobiltelefonon szabványos felbontással könnyen megtekinthető legyen.
Források	Adatbázis hozzáférési követelmények.
Feldolgozás	Manuális továbblépés a következő képernyőre gombnyomásra.
Adat	Belső adatok és felhasználói adatok a mögöttes adatbázisból.

8.5 UI vezérlő

8.5.1 Main Activity

A Main Activity az alkalmazás "fő tevékenysége", közvetlenül a Splash Screen eltűnése után hívódik meg. Felelős a szükséges engedélyek ellenőrzéséért, valamint a Bluetooth adapter állapotának ellenőrzéséért indításkor, felhasználói műveletet kér, ha bármilyen engedélyre szükség van. Ha a Bluetooth engedélyezve van, megköti és belép a Bluetooth szolgáltatásba. A MainActivity kezeli a Fragment-eket, megvalósítja az összes Fragment-tel kapcsolatos hallgatót, és kezeli az állapot- vagy adatfrissítéseket. Ez az aktivitás figyeli a Bluetooth szolgáltatásból érkező állapotfrissítéseket, és azokat a felhasználónak üzeneteken keresztül továbbítja. Az eszköztár akcióit is itt kezeli. A Bluetooth-szal kapcsolatos ControlActiviy példányt itt hozza létre és kezeli.

8.5.2 Splash Activity

A SplashActivity osztály az alkalmazás belépési pontja, 5 másodpercig jelenik meg, és a következő információkat tartalmazza:

- Turbinás Spirométer

- Szakdolgozat

8.5.3 Fragment

Az összes Fragment az alkalmazás egy képernyőjét képviseli, a Fragmentek kezelése a Main Activity-ből történik a FragmentManager segítségével. Minden Fragment saját interakcióhallgatót tartalmaz, hogy új adatokat szolgáltasson a felhasználónak.

8.5.3.1 AddFragment

Az AddFragment funkciója, hogy a felhasználó számára lehetővé tegye a betegadatok bevitelét a szoftverbe.

A következő beviteli adatokat tartalmazza:

- Vezetéknév
- Keresztnév
- Születési dátum

- Azonosítószám

8.5.3.2 ListFragment

Itt jelenik meg a betegek listája az adatbázisból. Itt lehet új beteget felvenni.

A következő vezérlőelemet tartalmazza:

- Beteglista nézet

Ez egy egyéni listanézet, amely tartalmazza a páciens vezetéknevét, keresztnevét és születési dátumát. Továbbá a páciensre kattintva megnyílik a páciens adatlapja, melyet törölhetünk. Amennyiben a teljes listát szeretnénk törölni, a jobb felső sarokban lévő törlés ikonra kell kattintani.

8.6 Driver

8.6.1 A készülék csatlakoztatásának folyamatábrája

8.6.2 Bluetooth szolgáltatás

Kapcsolódó projektobjektumok:

- ControlActivity class
- SelectDeviceActivity class

A feladat során használt Android API objektumok:

- BluetoothAdapter: a rendszer Bluetooth-adapterének tárolására szolgál
- BluetoothDevice: kapcsolatot hozhat létre az adott eszközzel, vagy lekérdezhet róla információkat, például a nevet, a címet, az osztályt

8.6.2.1 BluetoothAdapter leírás

A rendszer Bluetooth-adapterét jelképezi. A BluetoothAdapter lehetővé teszi az alapvető Bluetooth-feladatok elvégzését, például az eszközkeresés kezdeményezését, a csatlakoztatott (párosított) eszközök listájának lekérdezését, egy BluetoothDevice példányosítását egy ismert MAC-cím használatával, valamint egy BluetoothServerSocket létrehozását más eszközök csatlakozási kéréseinek meghallgatására és a Bluetooth LE-eszközök keresésének elindítására.

8.6.2.2 BluetoothDevice leírás

Egy távoli Bluetooth-eszközt képvisel. A BluetoothDevice segítségével kapcsolatot hozhat létre az adott eszközzel, vagy lekérdezhet róla információkat, például a nevet, a címet, az osztályt és a csatlakozási állapotot.

Ez az osztály valójában csak egy vékony burkolat egy Bluetooth hardvercímhez. Ennek az osztálynak az objektumai megváltoztathatatlanok. Az ezen az osztályon végzett műveletek a távoli Bluetooth hardvercímen történnek, a BluetoothAdapter segítségével, amelyet a BluetoothDevice létrehozásához használtak.

8.7 Adatbáziskezelő

A mögöttes rendszer egy Room adatbázis. A Room persistence könyvtár egy absztrakciós réteget biztosít az SQLite felett, amely lehetővé teszi a gördülékeny adatbázis-hozzáférést, miközben kihasználja az SQLite teljes erejét. A Room különösen a következő előnyöket biztosítja:

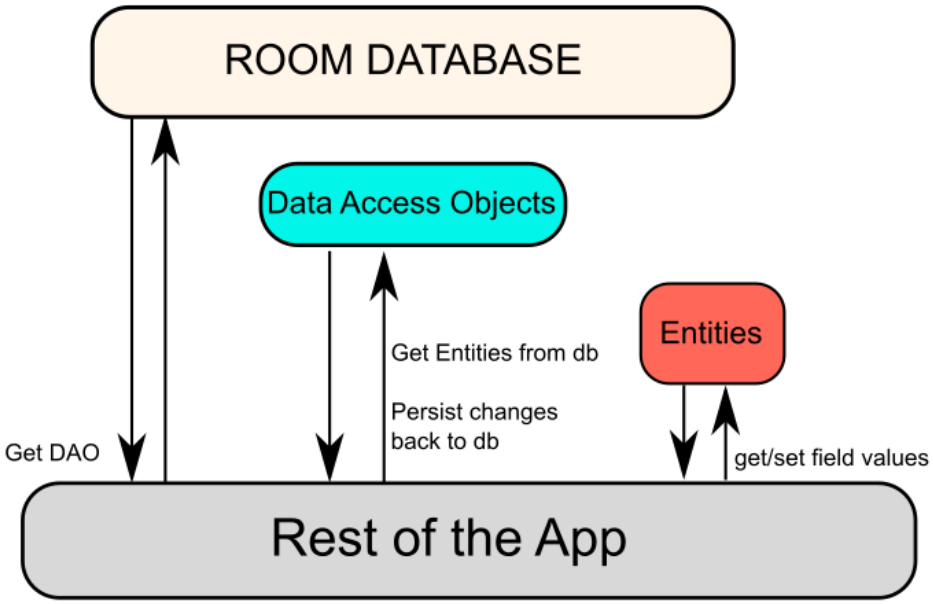
- Az SQL-lekérdezések fordítási idejű ellenőrzése.
- Kényelmi megjegyzések, amelyek minimalizálják az ismétlődő és hibás forráskódot.
- Áramvonalas adatbázis-migrációs útvonalak.

Az olyan alkalmazások, amelyek nem triviális mennyiségű strukturált adatot kezelnek, nagy hasznát vehetik az adatok helyi tárolásának. A leggyakoribb felhasználási eset a releváns adatok gyorsítótárba helyezése, hogy amikor a készülék nem tud hozzáférni a hálózathoz, a felhasználó offline állapotban is böngészhesse a tartalmat.

Három fő komponense van:

- Entity: olyan adategységek, amelyek az alkalmazás adatbázisának tábláit képviselik,
- Dao (Data Access Object): olyan módszereket biztosítanak, amelyekkel az alkalmazás lekérdezheti, frissítheti, beszúrhatja és törölheti az adatokat az adatbázisban.
- Database: az adatbázist tárolja, és az alkalmazás tartósan fennálló adataihoz való kapcsolat fő hozzáférési pontjaként szolgál.

Az adatbázis-osztály biztosítja az alkalmazás számára az adott adatbázishoz kapcsolódó DAO-k példányait. Az alkalmazás viszont a DAO-kat használhatja arra, hogy adatokat kérjen le az adatbázisból a kapcsolódó adatobjektumok példányaiként. Az alkalmazás a definiált adatentitásokat arra is használhatja, hogy sorokat frissítsen a megfelelő táblákból, vagy új sorokat hozzon létre beszúrásra. A *17. ábra* szemlélteti a Room különböző összetevői közötti kapcsolatot. [8]



17. ÁBRA: ROOMDATA

9. Észrevételek, fejlesztési lehetőségek

Szakdolgozatomban egy alap spirometriai mérést valósítottam meg, mely rengeteg fejlesztési tevékenységet tud még maga után vonni. Amennyiben pontosabb adatokat szeretnénk, érdemes ultrahangos, vagy nyomáskülönbség elvén alapuló spirométert használni. A lamella tehetetlensége miatt a mérés pontossága nem megfelelő, ezt korrigálni lehet szoftveresen, de jelentős mennyiségű fejlesztést és tesztelést igényel.

Amennyiben egy piacra szánt orvostechnikai eszközt szeretnénk létrehozni, azt úgy kell megvalósítanunk, hogy figyelembe vesszük a különböző szabványokat. Ezek a szabványok leírják a megfelelő paramétereket, amelyeknek meg kell felelniük az eszközöknek. Ilyen például az ellenállás értéke, hiszen a páciensnek erőltetett kilégzést kell végeznie, így ügyelni kell arra, hogy a lehető legkevesebb legyen a cső ellenállása.

A szoftvert figyelembe véve, érdemes használhatósági vizsgálatot végezni a különböző korosztályoknál, hiszen az új generáció már jól tudja kezelni az okostelefonokat, viszont a páciensek között számos idősebb ember is van, így számukra is elérhetővé, és használhatóvá kell tenni a szoftvert.

Különböző minőségügyi protokollok vannak, így a szoftver képes vizsgálni az értékeket, és a páciens által megadott személyes adatok alapján egy diagnosztikát felállítani, amelyet az orvos felülírhat.

Amennyiben több mérést végzett a beteg, akkor a szoftver képes lehet a legjobbat kiválasztani, és pdf kiterjesztésben elmenteni, majd nyomtatóra csatlakozva, kinyomtathatja azt, vagy email-ben el tudja küldeni az orvosnak, aki azt elemezve megkezdheti a gyógykezelést.

A Covid-19 miatt a távoli diagnosztika, illetve gyógyászat kiemelt jelentőségű lehet, minden technológia adott ehhez, a fejlesztéseknek csak az emberi elme szabhat határt.

10.Összefoglaló

Szakdolgozatom témájának kiválasztásakor igyekeztem figyelembe venni korunk egyik legsúlyosabb egészségügyi problémáját. Munkám kezdetén kutatómunkát végeztem a Covid-19 okozta megbetegedésekről, majd az ezt követő post-Covid tünetekről. A különböző megbetegedéseket a legkönnyebben úgy lehet legyőzni, ha időben felismerjük őket. Ez igaz a tüdővel kapcsolatos megbetegedésekre is. A levegő minősége a városokban egyre rosszabb, így egyre több a légzéssel kapcsolatos megbetegedés. Az ezzel kapcsolatos megbetegedések kiszűrésében segítségünkre lehetnek a különböző spirométerek. Ezeket úgy kell megtervezni, hogy otthoni használatra is alkalmasak legyenek, akárcsak egy vérnyomásmérő.

Ennek megvalósítása során egy turbinás spirométert terveztem, amelynél a lamella mozgását figyeltem LED-ek és fototranzisztorok segítségével. Az ebből a mozgásból számított spirometriai adatokat (FVC, FEV1, PEF) Bluetooth kapcsolaton keresztül továbbítottam egy androidos szoftvernek. A fejlesztéshez a MIR által gyártott turbinát használtam. A tranzisztorok által mért jelek analóg jelek.

A program Androidos környezetben futtatható, mely Kotlin nyelven íródott. A szoftvert úgy alakítottam ki, hogy könnyen kezelhető és áttekinthető legyen. Bluetooth-on keresztül kapcsolódik a készülékhez, és azon keresztül adatokat fogad. A program képes adatbázist létrehozni, kezelni, melyben a páciensadatok jelennek meg. A szoftvert számítógépen teszteltem többször, majd később már mobiltelefonon is.

A munka befejeztével fejlesztési lehetőségeket vázoltam fel, annak érdekében, hogy minél pontosabb, jobb minőségű eszköz készüljön el. Ekkor érdemes, sőt szükséges folyamat a szabványok figyelembevétele és alkalmazása is.

11. Summary

When choosing the topic of my thesis, I tried to take into consideration one of the most serious health problems of our time. At the beginning of my work, I did research on Covid-19-induced diseases and the subsequent post-Covid symptoms. The best way to combat the various diseases is to detect them in time. This is also true for lung-related diseases. Air quality in cities is getting worse, so respiratory diseases are on the increase. Various spirometers can help us detect these diseases. They should be designed for home use, just like a blood pressure monitor.

To achieve this, I designed a turbine spirometer, in which I monitored the movement of the lamellae using LEDs and phototransistors. I transmitted the spirometric data (FVC, FEV1, PEF) calculated from this movement to an android software via Bluetooth connection. For the development I used a turbine manufactured by MIR. The signals measured by the transistors are analogue signals.

The program runs in an Android environment, written in Kotlin. The software has been designed to be easy to use and easy to understand. It connects to and receives data via Bluetooth. The program can create and manage a database in which patient data is displayed. I tested the software several times on my computer and later on my mobile phone.

At the end of the work, I have outlined options for improvement in order to produce a more accurate, higher quality tool. It is then a worthwhile, even necessary, process to consider and apply standards.

12.Felhasznált irodalom

[1] Dr. Jobbágy Ákos: Orvosbiológiai méréstechnika, Budapesti Műszaki és Gazdaságtudományi Egyetem Méréstechnikai és Információs Rendszerek Tanszék, 2003.

[2] M R Miller 1, J Hankinson, V Brusasco, F Burgos, R Casaburi, A Coates, R Crapo, P Enright, C P M van der Grinten, P Gustafsson, R Jensen, D C Johnson, N MacIntyre, R McKay, D Navajas, O F Pedersen, R Pellegrino, G Viegi, J Wanger: Standardisation of spirometry. In: The European respiratory journal 2005 Aug; 26:319-338.

[3] Magyar Pál, Losonczy György: A Pulmonológia kézikönyve, Medicina Könyvkiadó Zrt., Budapest, 2012, 21. oldal, 147. oldal

[4] MIR honlapja: https://spirometry.com/, Letöltés dátuma: 2022.04.29.

[5] Piston Kft. honlapja: http://www.piston.hu/, Letöltés dátuma: 2022.04.29.

[6] NDD honlapja: http://nddmed.com/, Letöltés dátuma: 2022.04.29.

[7] Luca Ardito, Riccardo Coppola, Giovanni Malnati, Marco Torchiano: Effectiveness of Kotlin vs. Java in android app development tasks, in Information and Software Technology, Volume 127, November 2020.

[8] Android Studio honlapja: https://developer.android.com/, Letöltés dátuma: 2022.04.29.

13.Ábrajegyzék