
UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE


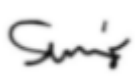
**Ventilator mecanic portabil, controlat
de la distanță, pentru asistarea
pacienților
cu probleme respiratorii**

PROIECT DE DIPLOMĂ

Coordonator științific:
dr. Losonczy Lajos

Absolvent:
Simó Zsuzsa

2021

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA Facultatea de Științe Tehnice și Umaniste din Târgu Mureș Specializarea: Calculatoare		Viza facultății:
LUCRARE DE DIPLOMĂ		
Coordonator științific: ș.l. dr. ing. Losonczy Lajos	Candidat: Simó Zsuzsa Anul absolvirii: 2021	
a) Tema lucrării de licență: VENTILATOR MECANIC, CONTROLAT DE LA DISTANȚĂ, PENTRU ASISTAREA PACIENȚILOR CU PROBLEME RESPIRATORII b) Problemele principale tratate: Studiu bibliografic privind sistemele de ventilație asistate - Realizarea sistemului de control și comandă pentru mecanismul de ventilație mecanică - Elaborarea programelor firmware și software pentru sistem - Elaborarea interfeței cu utilizator c) Desene obligatorii: - Schema bloc a aplicației - Schema electrică a aplicației - Organigramele privind firmware-ul și software-ul realizat. d) Softuri obligatorii: - Aplicația încorporată - Interfața cu utilizatorul e) Bibliografia recomandată: <ul style="list-style-type: none">- Sumanta, B., Sriram, K., India, 2012, Real-Time Breath Rate Monitor based Health Security System using Non-invasive Biosensor- Tobin, M.J, 2006. Principles and Practice of Mechanical Ventilation, Ed. McGraw-Hill, NewYork- Espressif Systems Copyright, 2019. ESP32 Series Datasheet, Espressif Inc., China- RMVS001 Rapidly Manufactured Ventilator System, Medicines & Healthcare Products Regulatory Agency, 2020 London		
f) Termene obligatorii de consultații: săptămânal g) Locul și durata practicii: Universitatea Sapientia, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, Lambda Communications SRL, Târgu Mureș Primit tema la data de: 31.03.2020 Termen de predare: 27.06.2021		
Semnătura Director Departament	Semnătura coordonatorului 	
Semnătura responsabilului programului de studiu	Semnătura candidatului 	

Declarație


Subsemnata/ul Simó Zsuzsa, absolvent(ă) al/a specializării Calculatoare, promoția 2021 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data: 2021.07.05

Absolvent

Semnătura



Ventilator mecanic portabil, controlat de la distanță, pentru asistarea pacienților cu probleme respiratorii

Extras

Ventilația mecanică reprezintă metoda prin care se asistă sau se înlocuiește respirația spontană. Din pricina leziunilor pulmonare cauzate de epidemii, nevoia de ventilatoare mecanice mobile eficiente și care izolează cadrele medicale de pacienți, este în creștere la nivel mondial. Scopul propus reprezintă o nouă abordare în domeniul ventilatoarelor mecanice și constă în dezvoltarea unui sistem complementar pentru monitorizarea și controlarea de la distanță a pacienților cu dificultăți de respirație, care poate funcționa ca un asistent respirator artificial neinvaziv, independent.

Pentru a realiza obiectivul propus, am combinat un nou tip de mecanism pentru ventilația mecanică cu un sistem de control modern, bazat pe un microcontroler de ultimă generație, ESP32 precum și o aplicație instalată pe un dispozitiv mobil cu interfață grafică, care permite acestor dispozitive să fie utilizate și controlate în mod ușor și eficient. De asemenea, echipamentul dispune de un control al respirației într-un ciclu închis, prin două căi, operare directă sau de la distanță.

Controlul și monitorizarea de la distanță bazate pe tehnologia Internet of Things (IoT) permite personalului autorizat conectat la o rețea wireless să monitorizeze parametrii respiratori și să ajusteze setările în timp real, să urmărească mai mulți pacienți în același timp, să intervină operativ sau chiar să ia decizii clinice majore, fără a fi nevoie ca să intre în contact direct cu aparatul sau cu pacientul.

Parametrii controlați și reglementați de aparatul dezvoltat, precum și limitele inferioare respectiv superioare ale parametrilor caracteristici, respectă prevederile RMVS001 ale regulamentul institutului internațional al agenției de reglementare a medicamentelor și a produselor medicale.

**SAPIENTIA ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK**

Távvezérelt, hordozható lélegeztetőgép

DIPLOMADOLGOZAT

**Témavezető:
dr. Losonczi Lajos**

**Végzős hallgató:
Simó Zsuzsa**

2021

Kivonat

Kulcsszavak: lélegeztetőgép, távvezérlés, beágyazott mikrovezérlő, légzésszabályozás, felhasználói interfész

A járványok által okozott tüdőkárosulások miatt megnőtt az igény a nagy hatásfokú, mobilis és a kezelő személyzetet a páciensről elszigetelő lélegeztetőgépek iránt. Feladatunkként ki egy olyan lélegeztető asszisztens kifejlesztését, amely egy új megközelítést jelent a mechanikus ventilátorok területén. Egy új koncepciójú levegőztető mechanizmust kombináltunk egy modern ESP32 mikrovezérlőre alapuló vezérlő rendszerrel és mobilgrafikus felhasználói felületre telepített programmal, amely lehetővé teszi ezen gépek könnyű hordozhatóságát, kezelését és szabályozását, ugyanakkor zárt visszacsatolású légzésszabályozást is biztosít. A tárgyak internete (IoT) technológián alapuló kétirányú, távirányított megfigyelési és vezérlési funkció lehetővé teszi, hogy egy vezeték nélküli hálózathoz csatlakozott felhatalmazott szakorvos figyelemmel kísérje a légzőkészülék paramétereit, és valós időben módosítsa a beállításokat, képes legyen több beteget is egyszerre nyomonkövetni, vagy akár klinikai döntéseket is hozzon, anélkül, hogy közvetlenül a készülékkel vagy a beteggel kellene érintkeznie.

A gépünk által kontrollált és szabályozott paraméterek, valamint ezek minimum és maximum határértékei megfelelnek a Medicines & Healthcare Products Regulatory Agency Nemzetközi intézet RMVS001 előírásainak.

Abstract

Keywords: ventilator, remote controlled, embedded microcontroller, breath control, user interface

Due to the lung damage caused by the epidemics, the need for high-efficiency, mobile ventilators that isolate operating personnel from the patient is growing worldwide. The aim is to develop a complementary system for remote patient monitoring and controlling which can act fully as a noninvasive artificial respiration assistant that represents a new approach in the field of mechanical ventilators. We have combined a completely new concept ventilation mechanism with a modern ESP32 microcontroller-based control system and a program installed on a mobile graphical user interface that allows these machines to be easily carried, operated and controlled, while also providing closed-loop breath control. Two-way, remote monitoring and control based on Internet of Things (IoT) technology allows an authorized physician connected to a wireless network to monitor respiratory parameters and adjust settings in real time, track multiple patients simultaneously, or even make clinical decisions, without having to come into direct contact with the device or the patient.

The parameters controlled and regulated by our machine, as well as their minimum and maximum limits, comply with the RMVS001 regulations of the international institute Medicines & Healthcare Products Regulatory Agency.

Tartalomjegyzék

1. Bevezető	12
2. Elméleti megalapozás és bibliográfiai tanulmány	13
2.1. Szakirodalmi tanulmány	13
2.2. RMVS001	14
2.3. Megvalósított lélegeztetőgépek	15
3. A rendszer specifikációja és architektúrája	15
Követelmény specifikáció	15
Felhasználói követelmények:	16
Rendszer követelmények:	16
Funkcionális követelmények	16
Nem funkcionális követelmények	17
Termékhez kötött követelmény	17
Szervezethez kötött követelmény	18
3.1. Alkatrészek	18
3.1.1 Mikrovezérlő	18
3.1.2. Motorvezérlő	19
3.1.3. Motor	20
3.1.4. OLED	21
3.1.5. Feszültségszabályzó	21
3.1.6. Szenzorok	22
3.1.7. Botkormány	23
3.1.8. A tervezett vezérlő modul elektromos rajza	23
3.2. Vezérlőprogram	26
3.2.1. Fejlesztői környezet	26
3.2.2. Felhasznált könyvtárak	26
3.2.3. Szoftver tesztelése	28
3.2.4. Fizikai architektúra	28
3.3. Alkalmazott mérőműszerek	29
3.3.1. Kronométer	29
3.3.2. Nyomásmérő	29

3.3.3. Áramlásmérő	30
3.3.4. A tervezett mérések architektúrája	30
4. A részletes tervezés	31
4.1. Használati eset	31
4.2. Paraméter definíció	32
4.3. Fejlesztés	34
4.4. Főprogram folyamatábrája	35
4.5. Távvezérelt működés	37
4.5.1. Get	37
4.5.2. Post	38
4.6. Megszakítási rutin	39
4.7. A vezérlő programkódja	40
4.7.1. Első verzió	41
4.7.2. Második verzió	41
4.7.3. Harmadik verzió	42
4.8. Felhasználói platform	52
5. Üzembe helyezés és kísérleti eredmények	57
5.1. Felmerült problémák és megoldásaik	58
5.2. Szoftvertesztek	58
5.3. Manuális tesztek	60
5.4. A megvalósult lélegeztetőgép	61
5.5. Mérési eredmények	62
6. Következtetések	62
6.1. Megvalósítások	62
6.2. Hasonló rendszerekkel való összehasonlítás	63
6.3. Továbbfejlesztési lehetőségek	64
6.3.1. Hardver továbbfejlesztési lehetőségek	64
6.3.2. Szoftver továbbfejlesztési lehetőségek	64
7. Irodalomjegyzék	65
8. Függelékek	67

Ábrák jegyzéke

2.1. ábra - Modern lélegeztetőgép tömbvázlata	14
3.1.2. ábra - ESP32-WROOM-32 lábkiosztása	19
3.1.2. ábra - TB67S109 léptetőmotorvezérlő lábkiosztása	20
3.1.3. ábra - 57HS57-3004A08-D211 léptetőmotor paraméterei és bekötése	21
3.1.4. ábra - SSD1306 OLED lábkiosztása	21
3.1.5. ábra - NCP1117 feszültségszabályzó lábkiosztása	22
3.1.6. ábra - BME280 lábkiosztása	22
3.1.7. ábra - HW-504 lábkiosztása	23
3.1.8. ábra - Vezérlő modul elektromos rajza	24
3.1.9. ábra - A vezérlő botkormányok menüje	25
3.2.4. ábra - A rendszer fizikai architektúrája	29
3.3.1. ábra - DT-1045 kronométer	29
3.3.2. ábra - MTA5 B N08 nyomásmérő	30
3.3.3. ábra - KME 715 R1L áramlásmérő	30
3.3.4. ábra - A tervezett mérések architektúrája	30
4.1. ábra - Használati eset diagram	31
4.4. ábra - A főprogram folyamatábrája	35
4.5.1. ábra - Szekvencia diagram - GET	37
4.5.2. ábra - Szekvencia diagram - POST	38
4.6.1. ábra - Megszakításrutin folyamatábrája	39
4.7.1. ábra - OLED karakter inicializálás	43
4.7.2. ábra - Plateau Pressure GET kérés	45
4.7.3. ábra - Plateau Pressure POST kérés	45
4.7.4. ábra - BreathingServer inicializálása	46
4.7.5. ábra - Új paraméter értéket számoló függvény	46
4.7.6. - 4.7.8. ábrák - Egyszerű get függvények	47
4.7.9. ábra - Tidal Volume Setup számítás	47
4.7.10. ábra - Breaths/min növelés	47
4.7.11. ábra - Nyomógomb irányt ellenőrző függvény	48
4.7.12. ábra - Nyomógomb felengedést ellenőrző függvény	48
4.7.13. ábra - Enum használata az irányok meghatározásáért	48
4.7.14. ábra - A megszakítási rutint engedélyező és inicializáló függvény	49
4.7.15. ábra - Az enum irányoknak megfelelő függvényhívások a főprogramban	51

4.7.16. ábra - ActualReadB igaz értékek a függvényhívások	51
4.7.17. ábra - ActualReadVP igaz értékek a függvényhívások	51
4.7.18. ábra - ActualReadB vagy ActualReadB igaz értékek a függvényhívások	52
4.8.1. ábra - A UI üzemmódváltó V1	53
4.8.2. ábra - A UI nyomógomb része	53
4.8.3. ábra - UI szenzor része	53
4.8.4. ábra - UI aktuális paraméter megjelenítő része	53
4.8.5. ábra - Kezdetleges UI és a fejlesztői környezet	54
4.8.6. ábra - UI Manual üzemmód	55
4.8.7. ábra - UI Távvezérelt üzemmód - Térfogat- és Nyomás Szabályozás	55
4.8.8. ábra - UI Távvezérelt üzemmód - Percenként belégzési idő és Be- illetve kilégzési arány szabályozás	56
4.8.9. ábra - UI Távvezérelt üzemmód - Szenzorok	56
5.1. ábra - OLED indítás	57
5.2. ábra - OLED szenzorok	57
5.3. ábra - OLED paraméterek	57
5.4. ábra - OLED IP cím sűgó	57
5.1.1. ábra - Szenzorok GET tesztjeinek eredménye	58
5.1.2. ábra - TV GET teszt függvénye	59
5.1.3. ábra - TVS GET Teszt eredményei	59
5.1.4. ábra - TVS UP POST teszt eredménye	59
5.1.5. ábra - TVS UP 100 iteráció utáni POST test eredménye	60
5.3.1 ábra - OLED eredmény	60
5.3.2 ábra - UI eredmény	60
5.3.1. ábra - A lélegeztetőgép funkcionális modellje a dugattyú felőli oldalról nézve	62

Táblázatok jegyzéke

3.1.1. táblázat - Mikrovezérlők összehasonlítása	19
4.2. táblázat - A RMVS001 és az általunk választott paraméterek	32
5.3.3. táblázat - Manuális és távvezérelt üzemmódok időbeli összehasonlítása	61

1. Bevezető

A COVID-19 okozta tüdőkárosodások miatt, világszerte megnőtt az igény mesterséges úton történő lélegeztetések iránt. Egy ilyen vírussal begyulladt tüdő számára a legnagyobb problémát az jelenti, hogy a lég hólyagok irányába nem képes a levegő eljutni, hiszen ezek a gyulladás következtében összeesnek. Egy lélegeztetőgép felhasználása újra lehetőséget nyújt abban, hogy olyan részein a tüdőnek, amelyek még nem gyulladtak be oxigén áramolhasson, ezáltal képes legyen a beteg szervezetében a már összeesett lég hólyagokat ismét kinyitni, és fenntartani ezt az állapotot.

Mialatt a számítástechnika fejlődött, újabb, összetettebb lélegeztetési módszerek váltak ismertté, amelyek az előre megadott paraméterekhez mérten akár optimálisan is képesek a lélegeztetést megváltoztatni. Emellett járvány időben, elsődleges követelménynek számít az ápoltak és az őket figyelő ápolók, orvosok között biztosítani egy fertőzésmentes távolságot. annak érdekében, hogy a vírus terjedését korlátozni lehessen. A lélegeztetőgépeknek felügyeletre van szüksége és a már beállított paramétereket értékeinek is időközönként módosításra szorulnak, emiatt a személyzetnek állandóan a beteg közelében kell lennie, ezáltal megnövelve az esélyét a fertőzés terjedésének. Ilyen alkalmakkor fontos lenne a beteg és az őt kezelő személyzet között a kellő távolság fenntartása. Miközben nem szabad megfélekezni azokról a betegekről sem, akik nem juthatnak kórházi kezeléshez, ezért szükséges a hordozhatóságot megteremteni, amely alkalmas arra is, hogy otthon használják, illetve olyan esetekben ahol sürgősségi helyzet alakul ki és a képzett orvos személyes ottléte nem oldható meg.

Így lett a cél megvalósítani egy olyan kézi hordozhatóságú egészségügyi rendszert, amely képes a légzési elégtelenségekkel küszködő betegeket távolról megfigyelni és ellenőrizni, illetve képes úgy funkcionálni, mint egy nem-invazív mesterséges lélegeztető készülék. Ezért vált feladattá egy olyan lélegeztetőgép megvalósítása, amelynek újszerű vezérlő rendszerét lehet akár számítógépekről/mobiltelefonokról is irányítani, emellett a felhasználói felület tegye lehetővé azt, hogy az erre kijelölt orvos nyomon követhesse a gép változásait, és képes legyen valós idejű szabályozásra, vagy épp olyan klinikai döntéseket hajtson végre, mialatt nincs rákényszerítve arra, hogy fizikailag a beteg mellett kelljen tartózkodnia, kiteve magát a fertőzés kockázatának.

2. Elméleti megalapozás és bibliográfiai tanulmány

2.1. Szakirodalmi tanulmány

A klinikákban használatos lélegeztetőgépek nagy része a friss gáz beáramlását a tüdőbe, úgy érik el, hogy a mellkas számára a légkörinél nagyobb nyomást tesznek lehetővé, és az így a fellépő nyomásnak a különbsége miatt a tüdő fele áramoltatja a levegőt. Az ilyen módszerre példa a folyamatos pozitív légúti nyomású lélegeztetés (CPAP), melynek előnye, hogy a kezelt személy ébren lehet, így képes azonnal segítséget hívni, ha kockázatok merülnének fel. Emellett az intubáció¹ során fellépő sérülések és kockázatoktól is védve van. Így a pozitív végkilégzési nyomás² (PEEP) megjelenésével számos előnye származott a lélegeztetéseknek [1].

Egyéb lélegeztetési módszer a BiPAP³, amely a CPAP-hoz hasonló módon pozitív légnyomást biztosít egy orrmaszkon keresztül a beteg számára. Vezérlése szintén egy asztali eszköz felhasználásával történik. A fő különbség hogy a légnyomás áramoltatása a légúthoz vezet. A CPAP folyamatos légnyomást biztosít, tehát kilégzéskor magasabb nyomásra és belégzéskor alacsonyabbra vált, aminek hatására folyamatosan nyitva maradnak a légutak. Ezzel szemben a BiPAP technológiára épülő gépek két nyomást figyelnek, a belégzési és a kilégzési nyomást. A kettő közti különbség minél nagyobb, annál jobban elősegíti mély lélegeztetést. Emellett rendelkezik olyan szenzorokkal, ami elősegíti, hogy abban az esetben is nyomást gyakoroljon, ha a beteg abbahagyta a légzést vagy állapotából adódóan nem képes rá [2].

A mai lélegeztetőgépek tömbvázlata a 2.1. ábrán látható, amely a mérőműszerek alapján számos előírt lélegeztetést tudnak előállítani, a beteg állapotának függvényében.

Az elmúlt évtizedek során a lélegeztetőgépeket egyre inkább mikrokontrollerek felhasználásával vezérlik, amelyek mérőműszerekkel vannak összekötve, így számos lélegeztetési szabályozást lehet a beteg számára biztosítani. Emellett, manapság már lehetőség nyílik arra, hogy szabályozni lehet a légvételek között szabályozni lehessen, a PRVC⁴ vagy az APV⁵ felhasználásával, amelyben lélegeztetéskor a kívánt belégzési térfogatot határozza meg,

¹ intubáció - olyan lélegeztetési módszer, amelyben csövet helyeznek a légutakba, mialatt a páciens nincs magánál

² Pozitív végkilégzési nyomás - olyan nyomás, amelyet a lélegeztetőgép minden egyes légzés végén alkalmaz, annak biztosítására, hogy az alveolusok ne legyenek hajlamosak az összeomlásra. Ez a módszer javítja az oxigénellátást.

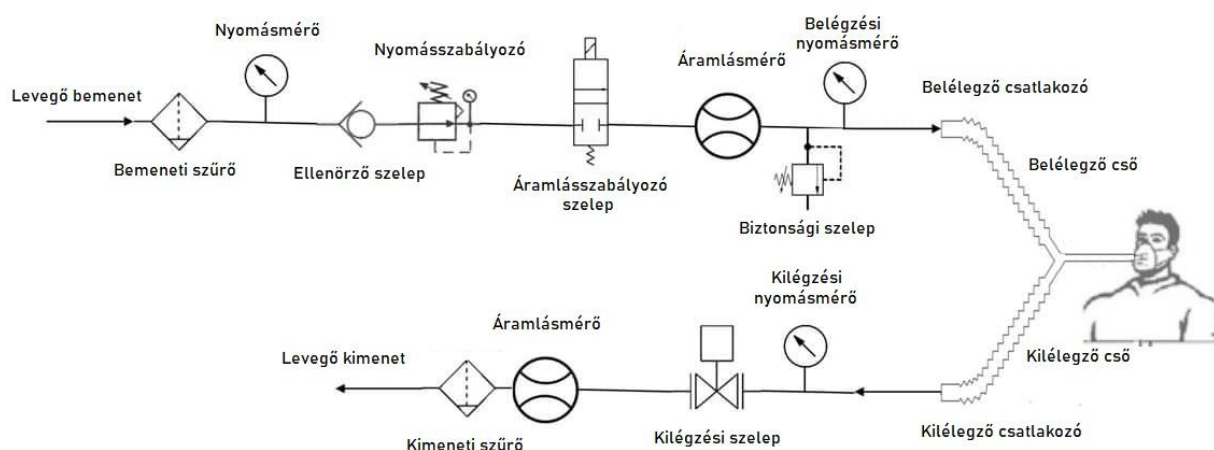
³ BiPAP - <https://www.healthline.com/health/what-is-a-bipap-machine#vs-cpap>

⁴ PRVC - Pressure Regulated Volume Controlled

⁵ APV - Adaptive Pressure Ventilation

majd ezt követően a lélegeztetőgép rövid időn belül olyan nyomást állít be, amellyel el lehet érni a kívánt térfogatot. Fontos megemlíteni követel meg a légzési cikluson belüli szabályozásokat (automatikus áramlás), vagy a PAV⁶, amely nyomástámogatásra képes. Ilyen típusú rendszerekben a meghatározott térfogat elérése miatt egy belégzési ciklus alatt képes áramlás szabályozni [3]. Az ilyen szabályozott légzéstámogatás során fontosságát veszti az intubációt.

a 2.1. ábrán látható tömbvázlatszerűen egy modern lélegeztetőgép is mérőműszereket, szabályzókat és szűrőket is alkalmaznak.



2.1. ábra - Modern lélegeztetőgép tömbvázlata

Az [4] alatt található cikkben a kutatók azt vizsgálták, hogy a CPAP valóban egy megfelelő stratégia, annak érdekében, hogy COVID-19 tüdőgyulladással fertőzött betegeket kezeljenek. A felmérést Angliában végezték, amelyben 33 személyt vizsgáltak meg. A vizsgálat befejezése után azt a következtetést vonták le, hogy a CPAP valóban hatékony kezelési módszernek bizonyul a oxigénhiányos állapot⁷ (hypoxia) javítása érdekében a súlyos légzési elégtelenségekkel küzdő betegek jelentős részénél.

2.2. RMVS001

Az RMVS001⁸ dokumentum tartalmazza a minimálisan klinikailag elfogadható specifikációját az Egyesült Királyság lélegeztetőgépei számára, a SARS-CoV-2 vírus által okozott jelenlegi COVID-19 járvány idején. A minimális klinikai követelményeket megegyezés alapján határozták meg olyan szakemberek, akik számos orvostechnikai eszközök segítségével

⁶ PAV - Proportional Assist Ventillation

⁷ Oxigénhiányos állapot: szervezet kóros állapota, amikor a beteg teste meg van fosztva a megfelelő oxigénellátástól.

⁸ RMVS001-https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/879382/RMVS001_v4.pdf

méréseket végeztek. Ezek az értékek nagy mértékben hozzájárulnak ahhoz, hogy kedvezőbb terápiás előnyben részesítsék a lélegeztetést igénylő beteget. Az ilyen betegek kezdeti gondozásában a SARS-CoV-2 okozta légzési elégtelenség miatt sürgős lélegeztetést igényelnek. A szakemberek úgy határozták meg, hogy az ennél alacsonyabb specifikációjú gépeket nem lehet klinikailag felhasználni. A dokumentum ezek mellett több javaslattal is szolgál, amelyeket felhasználva a lélegeztető gépet elfogadhatóság szerint is tesztelni lehet (például mű tüdő alkalmazása a térfogat és nyomás meghatározásának érdekében), monitorizálásához javaslatokat, a kellő biztonság fenntartásához lépéseket és számos kísérleti eredményeket, amelyekhez hasonlítani lehet a fejlesztésre kívánt lélegeztetőgépet. [5].

2.3. Megvalósított lélegeztetőgépek

Mióta az RMVS001 dokumentum publikussá vált, számos próbálkozás született a kritériumoknak megfelelő lélegeztetőgép iránt. Ilyen például a Q-vent, amely egy többutas vezérlő szeleppel ellátott lélegeztetőgép, amelyhez csatlakoztatva vannak olyan szűrők, amellyel akár fűteni és párasítani is lehet. Céljuk az volt, hogy a baktériumok terjedését megakadályozzák, ezért három anyag ötvözetéből, 3D nyomtatót felhasználva, készítették ezeket a részeket. Ez az automatizált eszköz légzőzsákokkal is felszerelhető, emellett nagyfokú vezérelhetőséget biztosít. A rendszerben köhögés érzékelőt használtak fel, amely figyelmeztet ha a beteg magához tért. Áramszünet esetén akkumulátor tartalékra kapcsol. A kifejlesztett eszközt intubált modellen tesztelték, amelynek eredményeként minden elvárás teljesült. Ezzel elnyerték az RMVS001 validációját [6].

Egy másik tanulmány egy alacsony költségű kézi lélegeztetőgép vezérlési tervét használja fel, amelyben a levegő pumpálás útján lehet a beteg tüdejébe levegőt juttatni. Bemutatja, hogy a felhasznált szenzort egy adaptációs algoritmus dolgozza fel, amely kimenetén javasol egy megfelelő áramlást, a beteg állapotát is figyelembe véve. A tanulmányban emellett kiemeli, hogy milyen szerepe van a visszacsatolásnak a szenzorok mérésében [7].

3. A rendszer specifikációi és architektúrája

Követelmény specifikáció

A tervezés előtt a követelmény specifikáció azért volt szükség, mert ez alapján meg lehetett fogalmazni, hogy a rendszer hogyan működjön, illetve a felhasználói felületen milyen

komponensek szerepeljenek. Ezek a megszorítások két csoportba sorolhatóak felhasználói és rendszer követelmények.

Felhasználói követelmények:

- Egy beteg számára biztosítson levegő ki- és beáramlást
- Legyen olcsó és emiatt könnyen beszerezhető
- Legyen kézben hordozható
- A felhasználói felület egy szakképzett orvos számára is könnyen használható legyen
- Lehessen állítani olyan paramétereket, amelyek a légzési térfogatot, a percenkénti belégzések számát, a plató nyomást illetve a ki- és belégzési arányt változtatva szabályozza a lélegeztetést

Rendszer követelmények:

Funkcionális követelmény:

- Ha probléma merülne fel a távvezérléssel, egy hibaüzenet jelenjen meg, ami egyéb alternatívát javasol a felhasználó számára
- A távvezérlési felületen lehessen a szenzoroknak olyan értéket adni, amely elérésével egy esemény történik: fényjelzés és előugró ablak jelzi a hibaüzenetet
- A távvezérlési felület legyen elérhető böngészőből, ha az eszköz által szolgáltatott hálózathoz csatlakozik a felhasználó
- A MANUAL üzemmódnál, ha a felhasználó elmozdítja az első botkormány jobb irányba a plató nyomás aktuális értéke növekszik, ha a botkormány tolja bal irányba a plató nyomás aktuális értéke csökken.
- REMOTE CONTROLLED üzemmódnál, ha a felhasználó megnyomja a PLATEAU PRESSURE + gombot a plató nyomás aktuális értéke növekszik, ha a PLATEAU PRESSURE - gombot nyomja le a plató nyomás aktuális értéke csökken.
- A MANUAL üzemmódnál, ha a felhasználó elmozdítja az első botkormány felfelé a légzési térfogat aktuális értéke növekszik, ha a botkormány tolja lefelé a légzési térfogat aktuális értéke csökken.
- A REMOTE CONTROLLED üzemmódnál egyaránt, ha a felhasználó megnyomja a TIDAL VOLUME + nyomógombot a légzési térfogat aktuális

értéke növekszik, ha a TIDAL VOLUME - gombot nyomja le a légzési térfogat s aktuális értéke csökken

- A MANUAL üzemmódnál, ha a felhasználó elmozdítja a második botkormányt bal irányba a lélegeztetési arány aktuális értéke növekszik, ha a botkormányt tolja jobbra a légzési arány aktuális értéke csökken.
- A REMOTE CONTROLLED üzemmódnál egyaránt, ha a felhasználó megnyomja a CYCLE RATIO + nyomógombot a légzési arány aktuális értéke növekszik, ha a CYCLE RATIO - gombot nyomja le a légzési arány aktuális értéke csökken
- A MANUAL üzemmódnál, ha a felhasználó elmozdítja a második botkormányt felfele a belégzések számának percenkénti aktuális értéke növekszik, ha a botkormányt tolja lefele a belégzések számának percenkénti aktuális értéke csökken.
- A REMOTE CONTROLLED üzemmódnál egyaránt, ha a felhasználó megnyomja a BREATH/MIN + nyomógombot a légzési arány aktuális értéke növekszik, ha a BREATH/MIN - gombot nyomja le a légzési arány aktuális értéke csökken
- Ha a felhasználó hosszan nyomja a második botkormányt MANUAL üzemmódban, amellyel a belégzések számának percenkénti változását és a lélegeztetési arányt lehet szabályozni, akkor a lélegeztető gép egy úgy nevezett tétlen állapotba kerül, amelyben a motor még nem indul el. Ha még egyszer benyomódik lassan az a gomb a lélegeztető gép elindul, a botkormány újbóli lenyomása esetén a gép visszakerül tétlen állapotba.
- Miután a gép kikerült tétlen állapotból és a motor elindult a gép kijelzőjén jelenjenek meg a következő paraméterek és az aktuális értékeik: PLATEAU PRESSURE, TIDAL VOLUME, CYCLE RATIO és BREATH/MIN. Illetve újbóli megnyomással a következő szenzorok nevei és aktuálisan mért értékei: HUMIDITY, SPO2, PEEP LEVEL, TEMPERATURE. Újbóli megnyomás esetén váltson vissza az előzőleg felsorolt paraméterek megjelenítésére.

Nem funkcionális követelmény:

Termékhez kötött követelmény

- A lélegeztető gép hordozhatósága

- A távvezérlési felületen megtekinthető legyen a távvezérléshez szükséges használati utasítás
- Lehessen nyomon követni a távvezérlési felületen az aktuális paramétereket, attól függetlenül, hogy milyen üzemmódban van a vezérlés (PLATEAU PRESSURE, TIDAL VOLUME, CYCLE RATIO, BREATH/MIN)
- MANUAL üzemmódnál két botkormány van, amelynél az egyik a PLATEAU PRESSURE és TIDAL VOLUME értékeit lehet szabályozni, a másikon a BREATH/MIN és CYCLE RATIO szerepel. Ezeket x és y irányban lehet növelni illetve csökkenteni. REMOTE CONTROLLED üzemmódnál ugyanezek szerepelnek nyomógombként, viszont ezek x és y irányú elmozdulását felváltja a külön növelő és csökkentő nyomógombok. Ezekből így nyolc található.

Szervezethez kötött követelmény

- A gép szabályozott paraméterei és ezeknek minimális és maximális határértékei feleljenek meg a Medicines & Healthcare Products Regulatory Agency nemzetközi intézet RMVS001 előírásainak.

3.1. Alkatrészek

A felhasznált elektronikai eszközöknek meg kell felelniük a megfogalmazott elvárásoknak és olyan tulajdonságokkal rendelkezzenek, amelyek ellátják a fent sorolt követelményeket, így nemcsak most segítik elő a helyes működést, hanem majd a továbbfejlesztés során is alkalmasak legyenek a bővítésre.

3.1.1. Mikrovezérlő

Az alkalmazás legfontosabb alkotóeleme a mikrovezérlő, emiatt első sorban a két legismertebb mikrovezérlő család tulajdonságait kellett összemérni két típus keretein belül: a Raspberry Pi Pico illetve az ESP32-WROOM-32 mikrovezérlőket⁹. A 3.1.1. táblázat összefoglalja az előbb említett mikrovezérlők összehasonlítását. Habár a programozási nyelv és a soros kommunikációs interfészek (SPI, I2C, UART) száma is elég lenne ha a Raspberry Pi Pico mikrovezérlő került volna választásra, mégis kizáró szempont az, hogy szükség volt a távvezérléshez szükséges WLAN¹⁰ kommunikációra, ezért az ESP32-WROOM-32

⁹ Mikrovezérlők összehasonlítása - <https://www.iottrends.tech/blog/raspberry-pi-pico-vs-esp-32/>

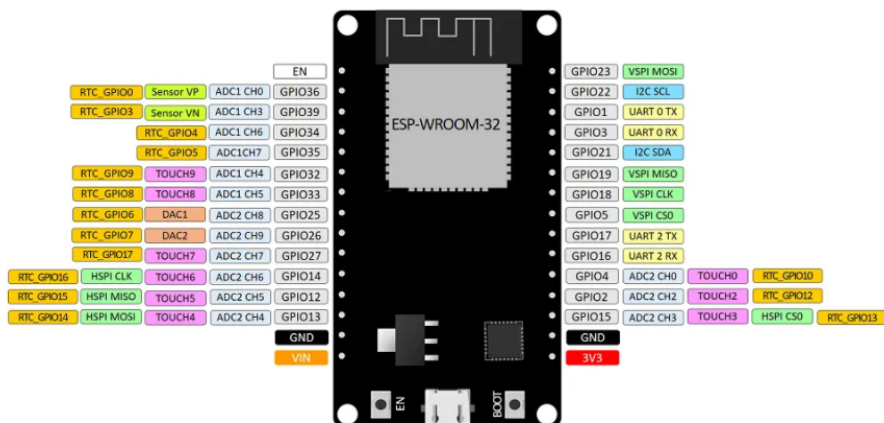
¹⁰ WLAN - Vezeték nélküli mikrohullámú kommunikáció

mikrovezérlőre esett a választás. Ráadásul az analóg digitális konverterek nagy száma, illetve az SRAM- szinte kétszer akkora mérete miatt számos fejlesztési előnyöket szolgál.

	Raspberry Pi Pico	ESP32-WROOM-32
SRAM	264 KB	520 KB
SPI	2	4
I2C	2	2
ADC	3 (12 bit)	18 (12 bit)
UART	2	3
Programozási Nyelv	MicroPython, C, C++	MicroPython, C, C++
WiFi	Nem támogatott	802.11

3.1.1. táblázat- Mikrovezérlők összehasonlítása

Miután mikrovezérlő típusa meghatározódott egyéb verziókat keresve olyan tulajdonságokra volt szükség, amelyek lényeges többlettel szolgálnak a fejlesztés során. Ilyen verziók a ESP32-WROOM-32D, illetve az ESP32-WROOM-32U. Mivel a legnagyobb különbség a mikrochipek mérete, illetve a teljes mikrovezérlő hosszában mutatkozott meg ezért nem volt igény egy alverzió használatára, így végül felhasználva az ESP32-WROOM-32 mikrovezérlő maradt, melynek a lábkiosztása a 3.1.2. ábrán látható. A processzor 240 MHz órajellel működik, 520 Kbyte SRAM memóriával, egy 18 bites ADC-el és két 8 bites DAC-el rendelkezik, illetve 3,3V feszültséggel működik [8].

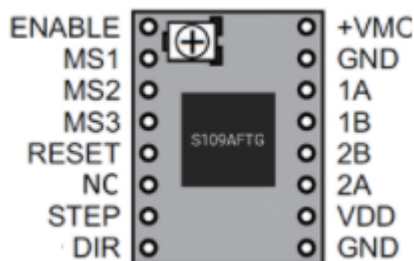


3.1.2. ábra - ESP32-WROOM-32 lábkiosztása [8]

3.1.2. Motorvezérlő

A léptetőmotor meghajtásáért felelős alkatrész. A választott vezérlő neve TB67S109, amely képes teljes, fél, negyed, nyolcad, tizenhatod, illetve harmincketted lépés engedélyezésére.

Maximális bemeneti feszültsége 50V, maximális kimeneti árama jó hőelvezetés esetén 4A is lehet. Lábkiosztása a 3.1.2. ábrán látható [9].



3.1.2. ábra - TB67S109 léptetőmotor vezérlő lábkiosztása

3.1.3. Motor

A légzés támogatáshoz szükséges motor kiválasztásához elsősorban szükség volt egy összehasonlításra a nyílt és zárt hurok között.

A nyílthurkú rendszerek legfőbb előnye, hogy a működésük egyszerű és könnyen beszerezhetőek, mivel nem képesek arra, hogy visszajelzést küldjenek a vezérlő számára, arról, hogy valóban megtörtént az elmozdulás vagy sem. Tehát az ilyen típusú rendszerek csak a bemenet alapján működnek és üzemeltetésük során nem használnak fel olyan kimenetről érkező választ amely segítségével futtatás során javíthatnak állapotukon. [10].

A zárt hurkú vezérlő rendszerekkel ellátott rendszereknél a motorra szerelt szenzor küld információt a vezérlő logikának. A visszacsatolás arra ad lehetőséget, hogy információt lehessen kapni arról, hogy a motor nem a kiadott vezérlés szerint mozog (megszorul) és a vezérlőáram növelésével növelhető a motor nyomatéka, hogy leküzdje a nagyobb terhelés támasztotta akadályt. Így a mozgási hibákat lehet bizonyos határok között kiküszöbölni [10].

Mivel elsősorban a könnyen megfizethetősége való törekvés volt a cél az említett követelmények szerint, ezért a választás a 57HS57-3004A08-D211 léptetőmotorra esett, amelyről több információ a motor paramétereiről (fázis, lépés, szög, nyomaték, súly) és a módszerről, ahogy a motort be kell kötni részletesen látható 3.1.3. ábrán. Ugyancsak itt van pontosítva, hogy melyik kábelhez milyen szín és milyen bemenet csatlakozik [11].

MOTORPARAMÉTEREK

FÁZIS	LÉPÉS SZÖG	ÁRAM/ FÁZIS	ELLENÁLLÁS/ FÁZIS	INDUKCIÓ/ FÁZIS	NYOMATÉK	ROTOR INERCIA	SÚLY
		A	Ω	mH	Nm	gcm ²	Kg
2	1,8°	3	0,9	2,5	1,2	300	0,7

A MOTOR ÉS VEZÉRLŐ BEKÖTÉSE

BIPOLÁRIS

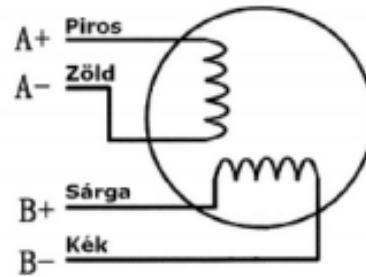
FÁZIS ÁRAM = 3A
NYOMATÉK = 1,2Nm

1. TEKERCS

- PIROS A+
- ZÖLD A-

2. TEKERCS

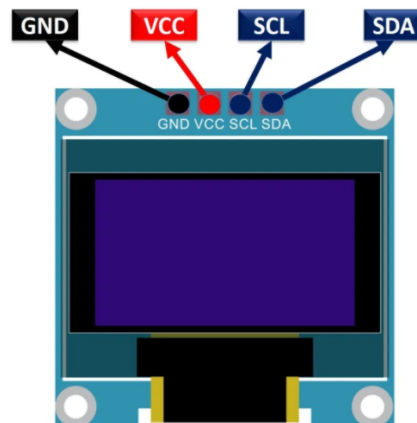
- SÁRGA B+
- KÉK B-



3.1.3. ábra - 57HS57-3004A08-D211 léptetőmotor paraméterei és bekötése

3.1.4. OLED

A manuális üzemmódhoz elengedhetetlen a már említett OLED típusú kijelző. A választott kijelző neve SSD1306 OLED, amely képes 128*64 pixelt megjeleníteni, csakis I2C kommunikációra képes. VCC-vel van ellátva, amelynek köszönhetően 3,3V-tól 5V-ig kell legyen a kapott feszültség a működéséhez. A lábkiosztását a 3.1.4. ábra szemlélteti [12].

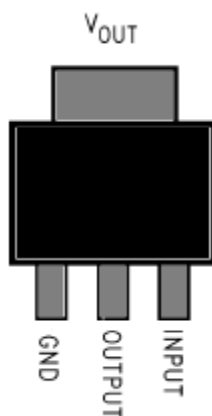


3.1.4. ábra - SSD1306 OLED lábkiosztása

3.1.5. Feszültség szabályzó

A választott feszültség szabályozó neve NCP1117, amely képes arra, hogy a következő fix kimeneti feszültségeket biztosítsa: 1,5 V, 1,8 V, 1,9 V, 2,0 V, 2,5 V, 2,85 V, 3,3 V, 5,0 V és 12 V.

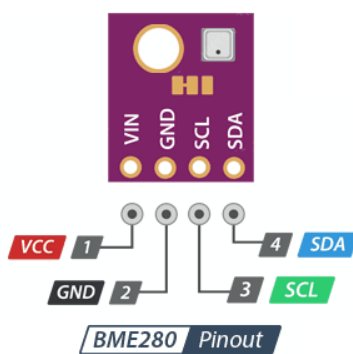
Emellett áramkorlátozással van ellátva, emiatt biztonságos működtetés és túlmelegedés ellen is védelmet ad. Lábkiosztása a 3.1.5. ábrán látható [13].



3.1.5. ábra - NCP1117 feszültszabályzó lábkiosztása

3.1.6. Szenzorok

Az alkalmazásban olyan szenzorokra volt szükség, amelyek segíthetnek további értékek kimutatásában. Első sorban hőmérsékletet, nedvességet és nyomást kellett mérni, például a légúti nedvességtartalom meghatározásához. Mivel az ilyen szenzorok külön-külön sok helyet foglalnak fizikálisan, illetve a mikrovezérlő kimenetei is korlátozottak ezért egy olyan szenzort kellett keresni, amellyel egyszerre lehet több mérést is végezni. Erre megoldás a Bosch által kifejlesztett BME280¹¹ digitális hőmérséklet, nedvesség és nyomás szenzor [14].



3.1.6. ábra - BME280 lábkiosztása

A 3.1.6. ábrán látható a BME280 lábkiosztása. A VIN a modul tápegysége, amely 3.3V-tól 5 V-ig terjed ki. A GND a mikrovezérlő földeléséhez csatlakozik. Az SCL egy soros órajel láb az I2C interfészhez. Az SDA egy soros adatláb az I2C interfészhez [14].

¹¹ BME280 - <https://lastminuteengineers.com/bme280-arduino-tutorial/>

Előnye, hogy a szenzorok aktív működése során kevesebb, mint egy mA-s a fogyasztása, alapjáraton pedig 5 μ A, amely alacsony energiafogyasztásnak számít [14].

3.1.7. Botkormány

A paraméterek manuális beállításához két botkormány kellett. A választott botkormánynak két analóg és egy digitális kimenete van. A két kimenethez potenciométer van rendelve így lehet Ox és Oy irányban is mérni az elmozdulást. A digitális kimenetnél le kell nyomni a gombot. Lábkiosztása a 3.1.7. ábrán látható [15].



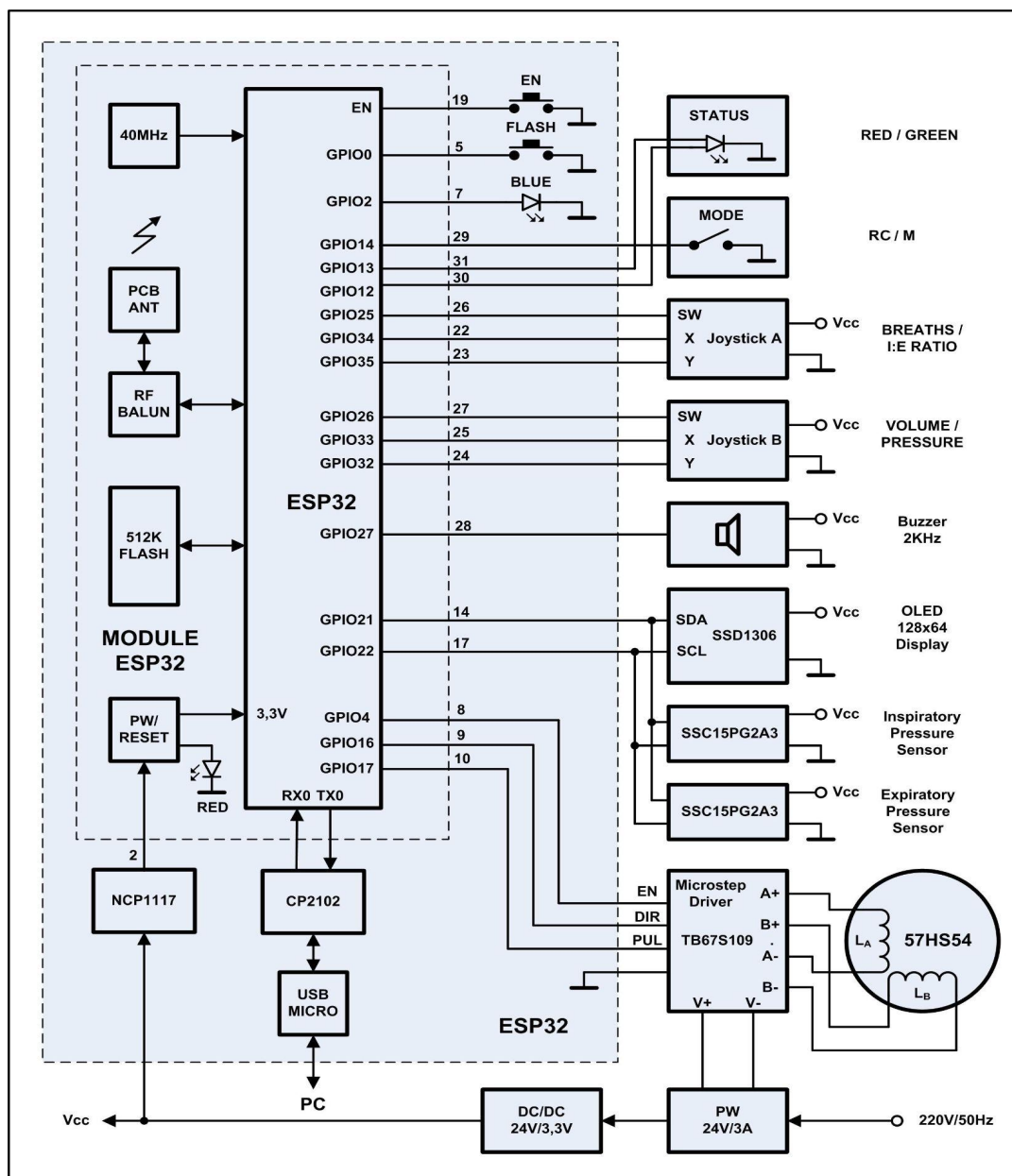
3.1.7. ábra - HW-504 lábkiosztása

3.1.8. A tervezett vezérlő modul elektromos rajza

A tervezett vezérlő modulnak az elektromos rajza, a 3.1.18. ábra tömbvázlatán látható. A ESP32 mikrovezérlő kék háttérrel van megjelenítve. A felsorolt perifériák az alábbi módon kapcsolódnak a mikrovezérlős rendszerhez.

A rendszer alkotóelemei sorrendben a következők:

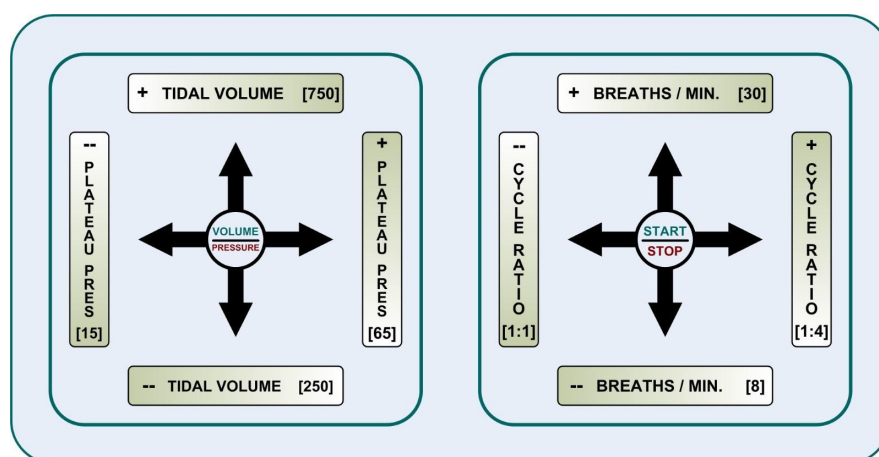
- *Állapotkijelző*: zölden világít a normál működés során és pirosan, ha a paraméterek túllépik a megengedett határokat.
- *Üzem mód kapcsoló*: a Manuális és Távvezérelt üzemmód közötti választásra szolgál.
- *Botkormány A*: függőleges irányban elmozdítva változtatni lehet a légzés ütemét (frekvenciáját) 8 és 30 légzés/perc értékek között, 1 légzés/perc lépésekben. Alapértelmezett (implicit) érték: 20 légzés/perc. Vízszintes irányban elmozdítva változtatni lehet a belégzési és kilégzési idők arányát 1:1 és 1:4 értékek között, 0,2 lépésekben. Alapértelmezett érték: 1:2. A botkormányt benyomva (mikrokapcsoló aktiválása) váltakozva el lehet indítani illetve megállítani a lélegeztetést.
- *Botkormány B*: függőleges irányban elmozdítva változtatni lehet a belégzési térfogat nagyságát 250 és 750 ml értékek között, 50 ml lépésekben. Alapértelmezett érték: 400 ml. Vízszintes irányban elmozdítva változtatni lehet a belégzési csúcsnyomást 15 és 65 mB értékek között, 5 mB lépésekben. Alapértelmezett érték: 20 mB. A botkormányt benyomva (mikrokapcsoló aktiválása) váltakozva át lehet térni a térfogatszabályozásról a nyomásszabályozásra és fordítva.



3.1.8. ábra - Vezérlő modul elektromos rajza

- *Buzzer*: paraméter váltáskor röviden jelez (csipog), mért paraméter határérték túllépéskor riaszt (hosszabban csipog).
- *Grafikus kijelző*: a beállított és a mért paramétereket jeleníti meg. A bal oldalon jelennek meg a két botkormánnyal beállított értékek (frekvencia, időarány, térfogat, nyomás), míg a jobb oldalon a mért paraméterek (pozitív végkilégzési nyomás, belélegzett levegő nedvességtartalma, hőmérséklet, vér-oxigénkoncentráció). A kijelzőn megjelenhetnek az esetleges paraméterhibák vagy üzemzavar állapotok üzenetei.

- *Belégzési nyomásmérő szenzor:* I2C soros porti kommunikációval periódusonként lekérdezett analóg erősítővel, ADC konverterrel és soros kommunikációs interfésszel ellátott okos érzékelő.
- *Kilégzési nyomásmérő szenzor:* I2C soros porti kommunikációval periódusonként lekérdezett analóg erősítővel, ADC konverterrel és soros kommunikációs interfésszel ellátott okos érzékelő.
- *Léptetőmotor meghajtó:* saját mikrovezérlővel ellátott DSP jelfeldolgozás¹², mikro lépés generátor. A bemeneti vezérlőjelek optikailag szigeteltek a nagyfeszültségű, erősáramú jelektől. Túlfeszültség és rövidzárvédelme is van.
- *Léptetőmotor:* az alkalmazott 57HS54-3004A08 típusú bipoláris léptetőmotor paramétereit a 3.6. ábrán látszanak.
- *Feszültségstabilizált tápforrás:* a léptetőmotor kellő nyomatékának elérése érdekében 24V/3A-es tápforrást alkalmaztunk, mely készen kapható az ipari kereskedelemben (laptop töltőhöz hasonló kiszerelésben). A külső tápforrás a lélegeztetőgéphez standard DC-022 hüvelycsatlakozóval kapcsolódik.
- *3,3V/1A DC/DC kommutációs tápforrás (buck konverter)*¹³: az elektronikus alkatrészek táplálását oldja meg. A kis veszteségek elérése miatt volt szükség erre a megoldásra, mivel relatív nagy feszültségről (24V) kell letranszformálni kis feszültségre (3,3V).



3.1.9. ábra - A vezérlő botkormányok menüje

A 3.1.9. ábrán látható a felsorolt követelményeknek megfelelően a vezérlő botkormányok menüje. Ez tartalmazza azt a négy értékpárt amellyel a lélegeztetőgépet adott irányokban lehet

¹² DSP jelfeldolgozású - Digitálisan feldolgozza az analóg jeleket

¹³ Kommutációs tápforrás - A tápforrásban lévő PWM átalakító lehetővé teszi, hogy nagyobb feszültségű bemenethez, alacsony feszültségű kimenet társuljon

szabályozni. A név mellett megtekinthetők azok az alapértelmezett értékek amelyek a rendszer indítása során vannak használatban. Emellett fel van tüntetve, hogy az első botkormány lenyomása esetén lehet váltani a térfogat és nyomás szabályozás között, illetve a második botkormány segítségével lehet elindítani és megállítani az eszközt.

3.2. Vezérlőprogram

3.2.1. Fejlesztői környezet

A vezérlő funkcionális elektronika beágyazott programját (firmware), Arduino integrált fejlesztői környezetben (IDE) került megírásra, figyelembe véve az alkalmazás korábban definiált követelményeit.

Az Arduino IDE nyílt forráskódú, ahol a felhasználónak számos lehetősége van a fejlesztéshez. A környezet a C/C++ nyelveket támogatja amelyhez saját funkciókat és modulokat lehet írni. Az Arduino IDE úgy működik, hogy miután a megírt kód fordítása során nincs hiba, a fordító generál egy .hex kiterjesztésű fájlt, majd ennek a fájlnak a tartalmát elküldi az USB kábelén keresztül a mikrovezérlőnek [16]. Mivel az volt a cél, hogy a felhasználói platform megjelenítését bármilyen böngésző tudja feldolgozni, legyen az mobilon vagy asztali eszközön, emiatt statikus HTML¹⁴ került kiválasztásra. Emellett előnynek számít még, hogy az eszköz számára nem nyújt plusz terhet, mert a weboldal renderelése a kliens oldalon történik. A HTML mellett JS¹⁵ illetve CSS¹⁶ komponenseket is fel lettek használva. Ezek megjelenítése és tervezése egy ingyenes fejlesztői környezetben valósult meg, amely a *w3schools*¹⁷ által létrehozott online HTML szerkesztő.

Viszont figyelni kell arra, hogy a tárhely kapacitása nem engedi meg, hogy komolyabb funkciókat legyenek hozzárendelve, tehát limitált mennyiségű kóddal kell dolgozni.

3.2.2. Felhasznált könyvtárak

A *WiFi.h* könyvtár lehetővé teszi, hogy a mikrovezérlő csatlakozni tudjon más hálózati eszközökhöz, illetve más hálózati eszköz tudjon hozzá visszacsatlakozni (képes legyen Wifit

¹⁴ Hypertext Markup Language (HTML) - Weboldalak tartalmának felépítésére használt jelölő nyelv.

¹⁵ JavaScript (JS) - Weboldal viselkedését leíró nyílt forráskódú szkriptnyelv.

¹⁶ Cascading Style Sheets (CSS) - Weboldalak elemeinek elrendeléséért felelős stílusleíró nyelv.

¹⁷ W3schools - <https://www.w3schools.com>

szórni, WPA2¹⁸ szintű titkosítással). Így képes lesz arra, hogy kiszolgálja a bejövő kapcsolatokat a kientől.

A *Wire.h* könyvtár lehetővé teszi a I2C¹⁹ illetve a TWI²⁰ soros buszokon történő kommunikációt. Hátránynak számít, hogy a *Wire.h* könyvtár egy 32 byte méretű puffert alkalmaz, emiatt minden kimenő adat csak 32 byte-os határ alatt mehet végbe. Ha meghaladja a határt adatvesztés történik [17].

Az *SPI.h* könyvtárat együtt alkalmazva a *Wire.h* könyvtárral lehetőség nyílik az I2C-t illetve az SPI buszt felhasználva, hogy a mikrovezérlő kommunikáljon más eszközökkel.

Az SPI (soros perifériás interfész) szinkron kommunikációs kapcsolatkor mindig van egy master-slave architektúra amelyben a mester eszköz, jelen esetben ez a mikrokontroller (ESP32-WROOM-32D), amely vezérli a szolgákat, vagyis a perifériás eszközöket: motorvezérlőt, szenzorokat, OLED kijelzőt [17].

Egyes modulok megvalósításához, olyan könyvtárakat vettem át, amelyek a github.com oldalán is megtalálhatóak. Ilyen például az *Adafruit_GFX.h*²¹ könyvtár, amely minden organikus kijelző²² (OLED) számára alternatívát nyújtanak egy alap grafikai megjelenítés eléréséhez, illetve egyéb alacsonyabb szintű funkciókat biztosít.

Egyéb nyílt forráskódú könyvtár a *Adafruit_SSD1306.h*²³ könyvtár, amely a fentebb említett I2C illetve SPI buszokat használ a kommunikációra. A felhasználható pinek minimális száma kettő, maximális száma öt lehet.

Ahhoz, hogy a betűméret típusát és méretét is megfelelően lehessen szemléltetni szükséges volt egy olyan könyvtár, amellyel a karakterek milyenségét lehet meghatározni, ez a könyvtár volt a *FreeSerif12pt7b.h*²⁴.

Aszinkron kapcsolat bevezetésére azért volt szükség, mert a kliens-szerver kommunikációhoz egyszerre több kapcsolatot is kellett kezelni. Ehhez az *ESPAsyncWebServer.h*²⁵ könyvtárat kellett használni.

¹⁸WPA2 - A négyirányú kézfogást alkalmaz, hogy a hozzáférési pont és a vezeték nélküli kliens függetlenül képesek arra, hogy bebizonyítsák egymásnak, hogy ismerik a kulcsokat, közzététel nélkül

¹⁹ Inter-Integrated Circuit (I2C) - A kommunikációt két sínen keresztül valósítja meg: egy órajelnek és egy adat számára. Mester és szolga üzemmódban egyaránt használható.

²⁰ Two Wire Interface (TWI) - <https://www.engineersgarage.com/i2cinter-integrated-circuit-twitwo-wire-interface/>

²¹ *Adafruit_GFX.h* - <https://github.com/adafruit/Adafruit-GFX-Library>

²² Organic Light-Emitting Diode (OLED) - Olyan organikus ledeket tartalmaz, amelyek elektromos áram hatására erős fényt bocsát ki magából.

²³ *Adafruit_SSD1306.h* könyvtár forrása - https://github.com/adafruit/Adafruit_SSD1306

²⁴ *FreeSerif12pt7b.h* könyvtár forrása - <https://github.com/adafruit/Adafruit-GFX-Library>

²⁵ *ESPAsyncWebServer.h* könyvtár forrása - <https://github.com/me-no-dev/ESPAsyncWebServer>

Az alkalmazott szenzor számára egyéb könyvtárak telepítésére volt szükség: *Adafruit_Sensor.h*²⁶ és *Adafruit_BME280.h*²⁷, amelyek felhasználásával sokkal egyszerűbben lehet az illető szenzor értékeit rögzíteni és átvinni a mikrovezérlő számára.

3.2.3. Szoftver tesztelése

A Postman²⁸ egy olyan tesztelési keretrendszer, amely felhasználásával lehetőség nyílik REST API-kat tesztelni. A felhasználói felületet egyszerű és számos előre megírt függvényekkel és funkcionalitásokkal segíti a felhasználót.

Konkurenciának számít a Flask²⁹ tesztelési keretrendszer, amelyben például egy GET kéréshez egy teljesen új útvonalra és ahhoz tartozó funkcióra van szükség. Emellett számos kódrészletben is implementálni kell, hogy milyen módon kell a válasznak kinéznie, amit ki kell íratni a konzolra vagy olyan módszer alkalmazni, amivel meg lehet azt jeleníteni.

A Postman számára egy ilyen tesztelés azért egyszerűbb, mert csak meg kell adni paraméterként a címsorba, a bal oldalon található opciók közül ki kell választani a GET kulcsszót és a válasz kinézetének meg lehet adni, hogy JSON formátum. A SEND nyomógomb segítségével el lehet küldeni a kérést, majd a válasz megtekinthető JSON-ként, kiolvasható a válasz HTTP státuszkódja, headere, cookie-jai. Lehetőség van ezekre a kérésekre validációkat helyezni, amiket az adott kérés elküldésekor le fog ellenőrizni az alkalmazás, mely így automatizált tesztelésre is használható.

A Postman képes emellett futtatni POST, PUT, PATCH és DELETE kéréseket is, illetve ingyenesen elérhető a legtöbb operációs rendszeren.

3.2.4. Fizikai architektúra

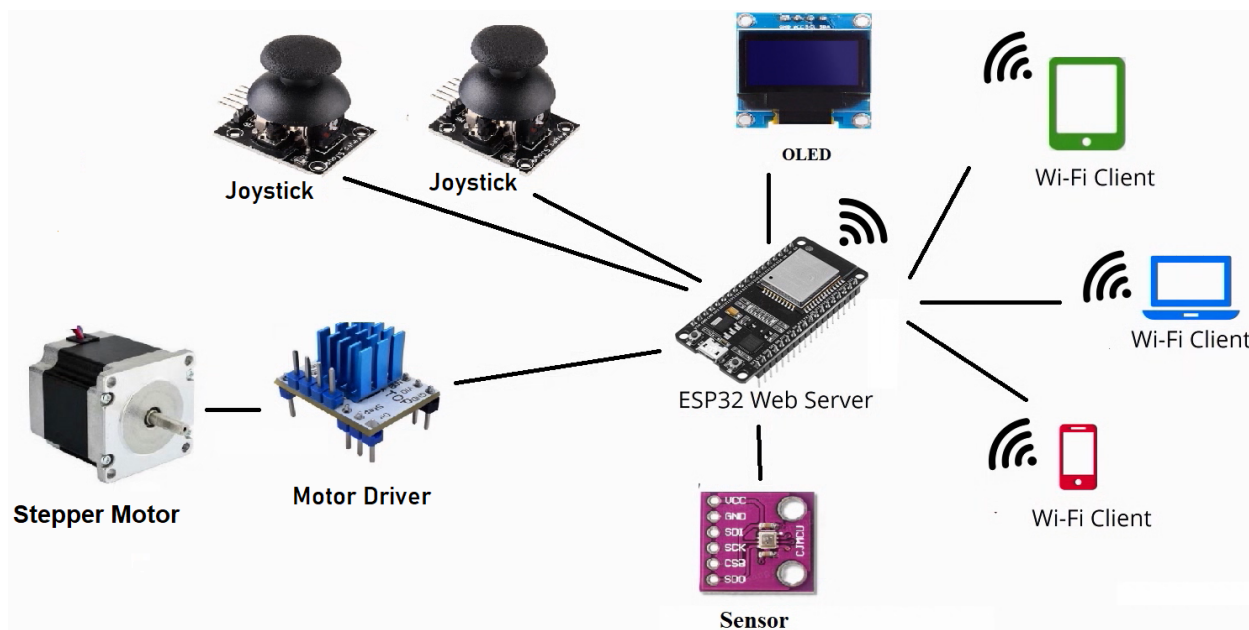
A 3.11. ábrán látható a fizikai architektúra, amely magába foglalja a mikrovezérlő kapcsolatát a routerrel, amelyre más eszközöket (laptop, számítógép, telefon) lehet csatlakoztatni, illetve azt, hogy a mikrovezérlő kommunikál a szenzorral, az OLED kijelzővel, a két botkormánnyal illetve a motorvezérlővel. A motorvezérlő továbbá a léptetőmotorral van kapcsolatban.

²⁶ *Adafruit_Sensor.h* - https://github.com/adafruit/Adafruit_Sensor

²⁷ *Adafruit_BME280.h* - https://github.com/adafruit/Adafruit_BME280_Library

²⁸ Postman - <https://www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it>

²⁹ Flask - <https://flask.palletsprojects.com/en/2.0.x/>



3.2.4. ábra - A rendszer fizikai architektúrája

3.3. Alkalmazott mérőműszerek

A megjelenített értékek validálásához egy mérőstandot kellett állítani és három mérőműszert alkalmazni: kronométert, áramlásmérőt és nyomásmérőt.

3.3.1. Kronométer

Egy szabályozott nagy pontosságú kronométer másodpercmutatóját elindítva lehetőség adódik a pontos idő lemérésére. A felhasznált kronométer a DT-1045, amelyet a 3.3.1. ábrán látható kép szemlélteti. Maximális mérhető időegység: 99 perc 59 másodperc.



3.3.1. ábra - DT-1045 kronométer

3.3.2. Nyomásmérő

A pontosság érdekében ipari nyomásmérőt kellett felhasználni, amely képes alacsony értékek között mérni (0-600 mBar). Az ilyen műszereket orvosi laboratóriumokban és gépfelszereléseknél egyaránt használják. Az EN 837-3³⁰ szerint 1.6-os osztályba sorolható. A mérőműszer a 3.3.2. ábrán látható. Neve MTA5 B N08 [18].

³⁰ EN 837-3 - Névleges méret és pontosság szerinti skála alapján határozott osztályzat a nyomásmérők számára



3.3.2. ábra - MTA5 B N08 nyomásmérő [18]

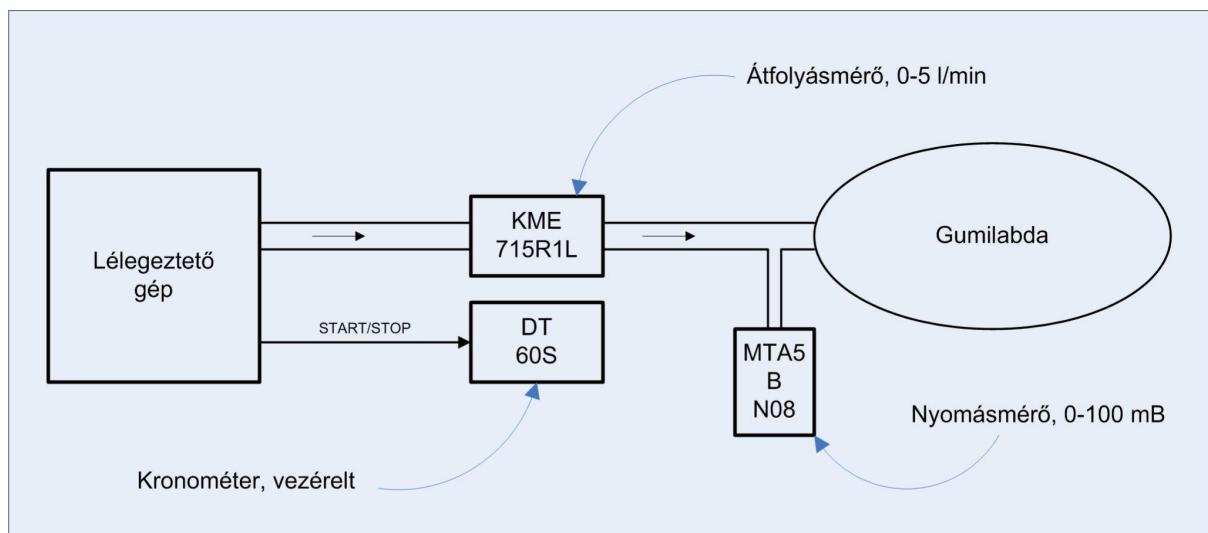
3.3.3. Áramlásmérő

A térfogat méréséhez KME 715 R1L átfolyásmérő volt alkalmazva. Pontossága $\pm 3\%$ és képes 2,2-848,2 Nm³/h térfogatáram mérésére, emellett analód kimenettel és USB csatlakozóval rendelkezik. A mérőműszert a 3.3.3. ábra szemlélteti [19].



3.3.3. ábra - KME 715 R1L áramlásmérő [19]

3.3.4. A tervezett mérések architektúrája



3.3.4. ábra - A tervezett mérések architektúrája

A mérésekhez szükséges eszközöket és a köztük lévő kapcsolatot a 3.3.4. ábra szemlélteti. A lélegeztetéshez szükséges orrmask helyére egy felfújható gumilabdára kell

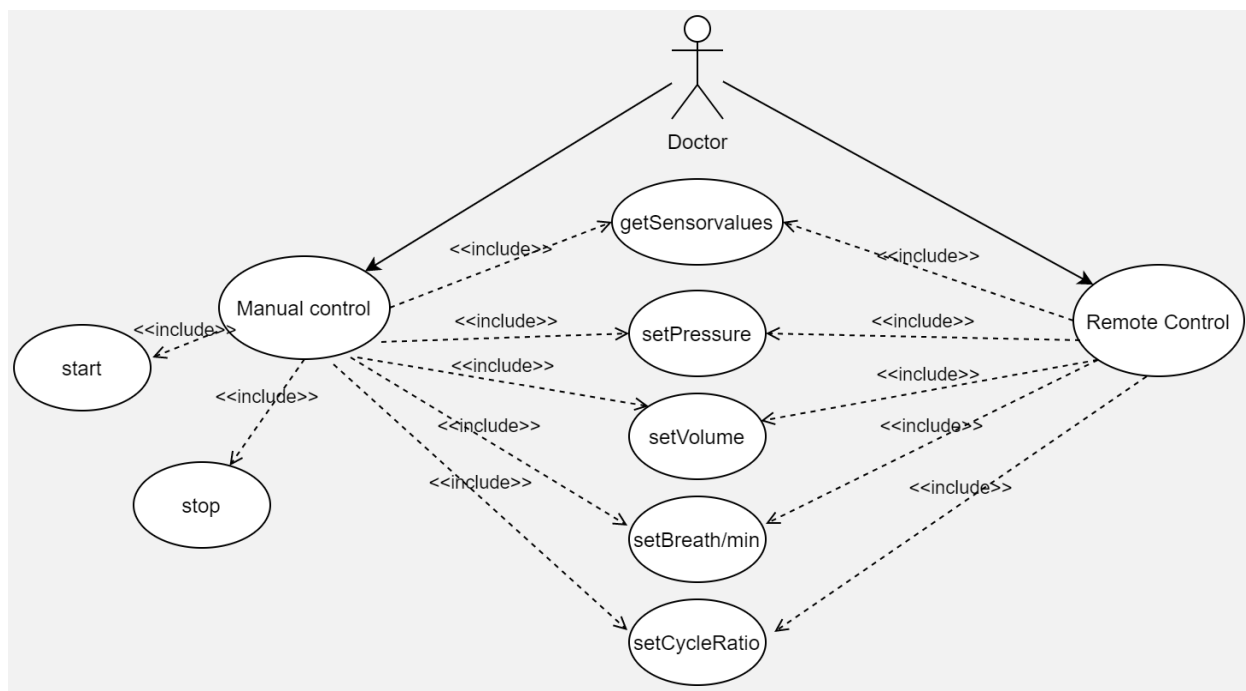
pótolni, a lélegeztető cső kimenetére az áramlásmérő van csatlakoztatva, illetve a labda bemenetére a nyomásmérő van kapcsolva. Az idő mérésére pedig az indítható/állítható kronométer.

4. A részletes tervezés

Ahhoz hogy megfelelően el lehessen indulni az alkalmazás tervezése során, szükség volt olyan diagramok elkészítésére, amelyek segítségével a funkcionális követelményeknek megfelelően egy átfogó kép alakult ki a rendszerről.

4.1. Használati eset

A első felhasznált diagram a használati eset diagram, amely megtekinthető a 4.1. ábrán, amely bemutatja a rendszert felhasználó alanyt (a doktort), akinek a szemszögéből a rendszer bemutatásra kerül. A doktor két állapotba léphet, az egyik a lélegeztető gép manuális irányítása, mialatt a másik a távvezérelt irányítás. A két állapotban az a közös, hogy mindkét részből lehetőség van a szenzor értékek eléréséhez, illetve a nyomás, térfogat, percenkénti belégzési szám és a ki/be légzési arányt értékeit lehet változtatni. A fő különbség pedig az, hogy csak manuális üzemmódban lehet a gépet elindítani és leállítani.



4.1. ábra - Használati eset diagram

4.2. Paraméter Definíció

	Breaths Per Minute	Cycle Division Ratio	Tidal Volume Setup	Plateau Pressure Setup
RMVS001	10 – 30 (in steps of 2)	1:1 – 1:3	250 – 600 (in steps of 50)	15 – 60 (in steps of 5)
Our values	10 – 30 (in steps of 2)	1:1 – 1:4 (in steps of 0.2)	250 – 750 (in steps of 10)	15 – 65 (in steps of 5)
RMVS001 default values	-	1:2 inspiration:expiration	350/400 ml	35 cmH2O
Our default values	20 breaths/minute	1:2 inspiration:expiration	400 ml	20 cmH2O

4.2. táblázat- A RMVS001 és a fejlesztés során választott paraméterek [5]

A 4.2. táblázat látható, hogy ahhoz hogy a gép klinikailag elfogadható legyen és terápiás előnyökkel járjon, az invazív lélegeztetést igénylő beteg számára a RMVS001 dokumentum leírásának kellett eleget tenni. A táblázatban nincs feltüntetve, viszont a paraméterek betartása több esetben megengedett egy ajánlottnál kisebb határértéket is. Viszont az adott értékek kötelezően a minimum értékek kell legyenek. Például a választott belégzési és kilégzési aránynál egy belégzéshez képest a kilégzés négyszer akkora. A RMVS001 által ajánlott értéknél négyszeres helyett háromszoros van. (lásd 4.2 ábra harmadik oszlop). A dokumentációban a belégzési térfogat értéke 250-től 600 ml-ig terjed ki a választott érték a két szám között helyezkedik el (lásd 4.2. táblázat negyedik oszlop). Ott ahol nem volt meghatározva, hogy mennyi legyen az alapértelmezett érték az alsó és felsőhatár átlagát volt felhasználva. Ilyen volt a percenkénti belégzések száma (lásd 4.2. táblázat második oszlop). A plató nyomás könnyebb kezelhetőség érdekében át kellett alakítani mB-ba, amely során 1 cmH2O megközelítőleg megfelel 0.98 mB-nak. A többi érték megfelel a RMVS001 ajánlásainak [5].

Miután ki lettek választva a lélegeztetéshez szükséges paraméterek maximális illetve minimális értékét, meg kellett határozni azokat a képleteket amelyeket később a programkódban lehet felhasználni.

Az első felmerülő probléma, amelyre megoldást kellett keresni az volt, hogy a motor számára nem lehet konkrétan beállítani, hogy hány fordulatot végezzen egy ciklus alatt, ezért a lépések számát kell meghatározni. Ez úgy oldódott meg, hogy számításba kellett venni az egy perc alatt történő belégzés számát illetve a ki és belégzési arányt, majd meghatározni, hogy mennyi lépésnek felel meg, majd ezekre a paraméterekre ki lehetett számolni, hogy mennyi impulzust kell átadni a motor számára.

A képletek így a következőképpen vannak levezetve:

- 1. Meghatározni a paraméterek minimális és maximális értékeit (alapul véve a RMVS001 javaslatait)**

Breaths Per Minute (BPM): 8 – 30 b/m

Cycle Division Ratio (CDR): 1:1 – 1:4

Tidal Volume Setup (TVS): 250 – 750 ml

Plateau Pressure Setup (PPS): 15 – 65 mB

2. Meghatározni milyen faktor értékek között lehessen ezeket az értékeket növelni (alapul véve a RMVS001 javaslatait)

Mivel a botkormány segítségével nem állíthatóak csak egész számú többszörösei ezért faktor értékek bevezetésére volt szükség. Így a kormányok mozgatásával ezek az értékek változnak.

Cycle Division Factor (CDF): 0 – 15

Tidal Volume Factor (TVF): 0 – 50

Plateau Pressure Factor (PPF): 0 – 10

A Breath Per Minute esetén nem volt szükséges faktort bevezetni, viszont ki kellett számolni a ciklusok időfelbontását és a belégzési ciklust:

Cycles Time Resolution (CTR) = 100 μ S

Breath Cycle Time (BCT) = (60 * BPM)/10000

A 10000 osztó megfelel 100 μ S-nak (1 s).

3. Ezután a faktorok alapján kiszámolni, hogy hány lépésnek felel meg egy paraméter csökkentés/növelés, úgy hogy egy paraméter maximális és minimális értékének hányadosát el kellett osztani a hozzá tartozó faktor maximális értékével.

CDR – 0,2 lépés

TVS – 10 lépés

PPS – 5 lépés

4. Meghatározni azokat a képleteket amelyek segítségével ki lehet számolni a paraméterek értékét

Mivel BPM esetében a növekedés egyesével történik nem volt szükség faktor bevezetésére

Cycle Division Ratio (CDR) = 1,0 + 0.2 * CDF

Tidal Volume Setup (TDS) = 250 + 10 * TVF

Plateau Pressure Setup (PPS) = 15 + 5 * PPF

5. Ezt követően meghatározni a belégzési és kilégzési ciklus-időt

Inspiration Cycle Time (ICT) = BCT/CDR

$$\text{Expiration Cycle Time (ECT)} = \text{BCT} - \text{ICT}$$

6. Kiszámolni hogy egy lépés alatt, hány köbcenti levegőt lehet átvinni

$$\text{Calculated Basic Volume (CBV)} =$$

$$= (\text{Vtorus} * \text{Sector}) / \text{Steps} = (288 * \frac{1}{4}) / 200 = 0,48 \text{ cm}^3$$

Ahol a Vtorus az a tóruszgyűrű formájú dugattyúház, a Sector az az arány, hogy egy dugattyú szelet mennyi ideig dolgozik és mennyi ideig pihen egy kör leforgása alatt, a Steps pedig a léptetőmotor adott lépésszáma egy teljes forgás során.

7. Végül meghatározni, hogy a motor hányat kell lépjen belégzés és kilégzés alatt illetve azt, hogy ez mennyi ideig tartson.

$$\text{Steps During Inspiration (SDI)} = \text{TVS} / \text{CBV}$$

$$\text{Inspiration Step Time (IST)} = \text{ICT} / \text{SDI}$$

$$\text{Steps During Expiration (SDE)} = \text{SDI}$$

$$\text{Expiration Step Time (EST)} = \text{ECT} / \text{SDE}$$

A motor helyből indulását véve alapul, előre meg kellett határozni a fordulatszámot. Az alapgondolat az volt, hogy kísérleti úton olyan kezdeti lépés hosszokat kellett keresni, amelyek a léptetőmotor induló gyorsulását segítik elő. Így az első 20 lépés valamivel hosszabb, mint a kiszámolt (nominális) érték, mert a táblázat értékei rendre hozzáadódik az ahhoz tartozó lépések nominális idejéhez. Az első lépéshez 100 µs-ot, a másodikhoz 80 µs-ot, a harmadikhoz 65 µs-ot, majd végül a huszadikhoz 2 µs-ot adunk hozzá, ezt követően a huszonegyedik lépéstől a nominális lépésidő marad és nulla µs-ot adunk hozzá.

4.3. Fejlesztés

Olyan szoftverfejlesztési megközelítést kellett alkalmazni, amely során hatékonyan és gyorsan fel lehessen mérni az esetlegesen megjelenő hibákat és sikerült olyan megoldásokat keresnem, amellyel biztosítani lehet a teljes rendszer elvárt működését. A megközelítés neve, amelynek alapelvét felhasználtam a tesztvezérelt fejlesztés.

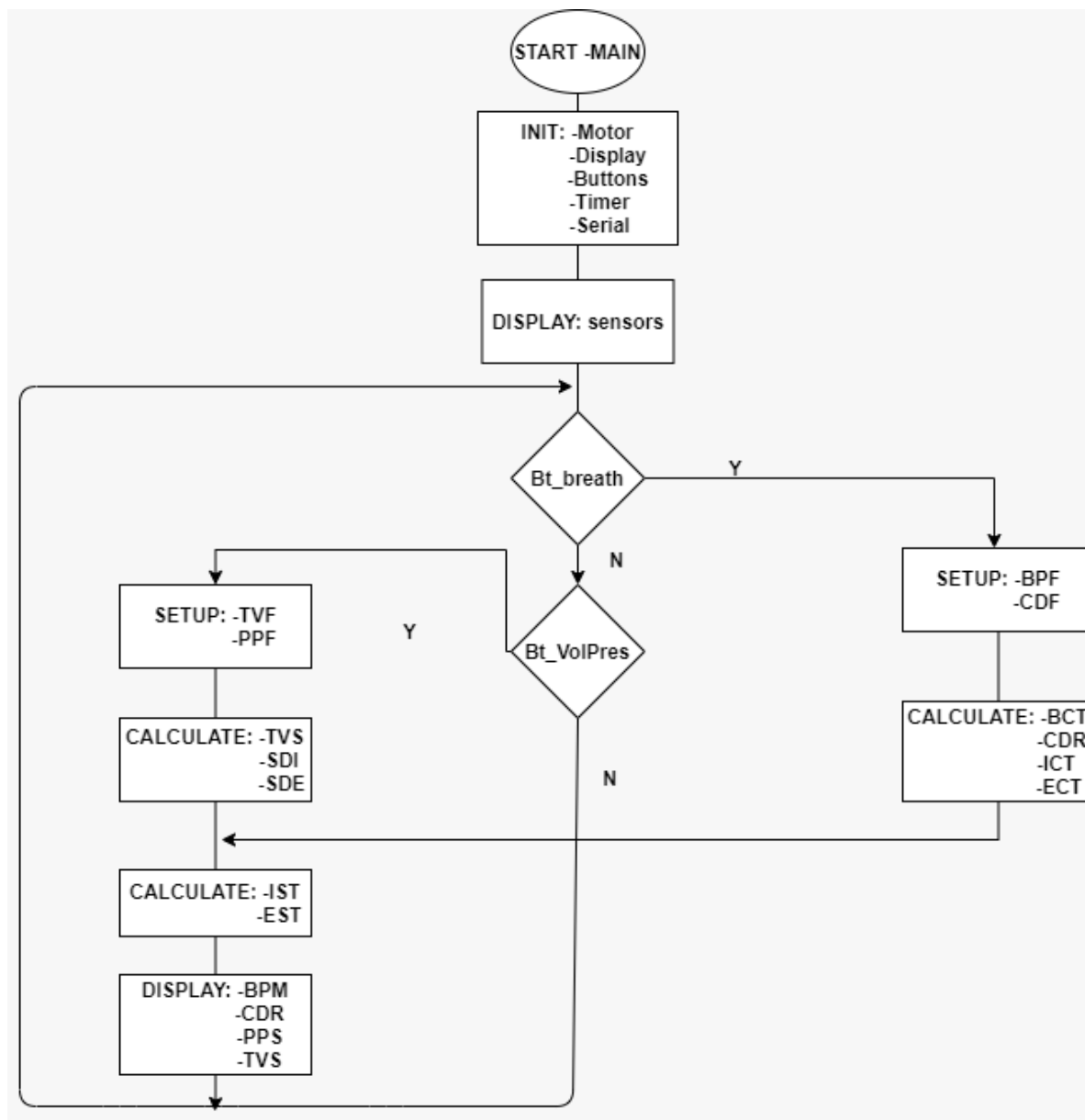
A tesztvezérelt fejlesztés³¹ (TDD) alapelve tehát az, hogy a fejlesztés során apró funkciókra bontáson megy keresztül az alkalmazásunkat és minden ilyen apró funkciót létre kell hozni majd letesztelni. Csak akkor lehet tovább menni, ha az elkészített funkcionalitás az elvárásokhoz mértén működik. Ez a megközelítés nem teszi lehetővé azt, hogy olyan részek is

³¹ Test Driven Development (TDD) - https://en.wikipedia.org/wiki/Test-driven_development

szerepeljenek az alkalmazásban, amelyek nincsenek felhasználva, viszont nagy előnynek számít, hogy a tesztelni kívánt részek aránylag kicsinek számítanak így nagy segítséget nyújt abban, hogy azonnal felfedje az előforduló hibákat. Emellett előnynek tekinthető, hogy a fejlesztőnek magának kell olyan programkódot írni, ami a kívánt tesztek kielégíti. Ugyanakkor a szoftver minőségét és megbízhatóságát növeli.

Az alkalmazás nem tette lehetővé az ilyen teszteknek az automatizálását, így fejlesztés során a hibák kiküszöbölését manuálisan kellett végrehajtani.

4.4. Főprogram folyamatábrája



4.4. ábra - A főprogram folyamatábrája

A 4.3. ábra összefoglalja a főprogram folyamatábráját, amelyből az látszik, hogy első lépésben a program inicializálja a motort, kijelzőt, gombokat, időzítőt és a soros kommunikációs portot. Ezt követően megjeleníti a szenzorok lemért értékeit. Ezután végig azt figyeli, hogy el volt-e mozdítva a valamelyik botkormány, ha az első botkormány volt, akkor a faktorok segítségével kiszámolja a ki-és belégzési időt, ha a második botkormány akkor az azokhoz tartozó faktorokat felhasználva meghatározza a be-és kilégzéshez szükséges lépések számát. Majd az eddigi számításokat felhasználva kiszámítja, hogy a léptetőmotor számára mennyi a ki-és belégzési idő. Végül a kijelzőn megjeleníti a beállított paraméterek értékeit. Majd ezután visszatér újból a gombok figyelemmel tartására.

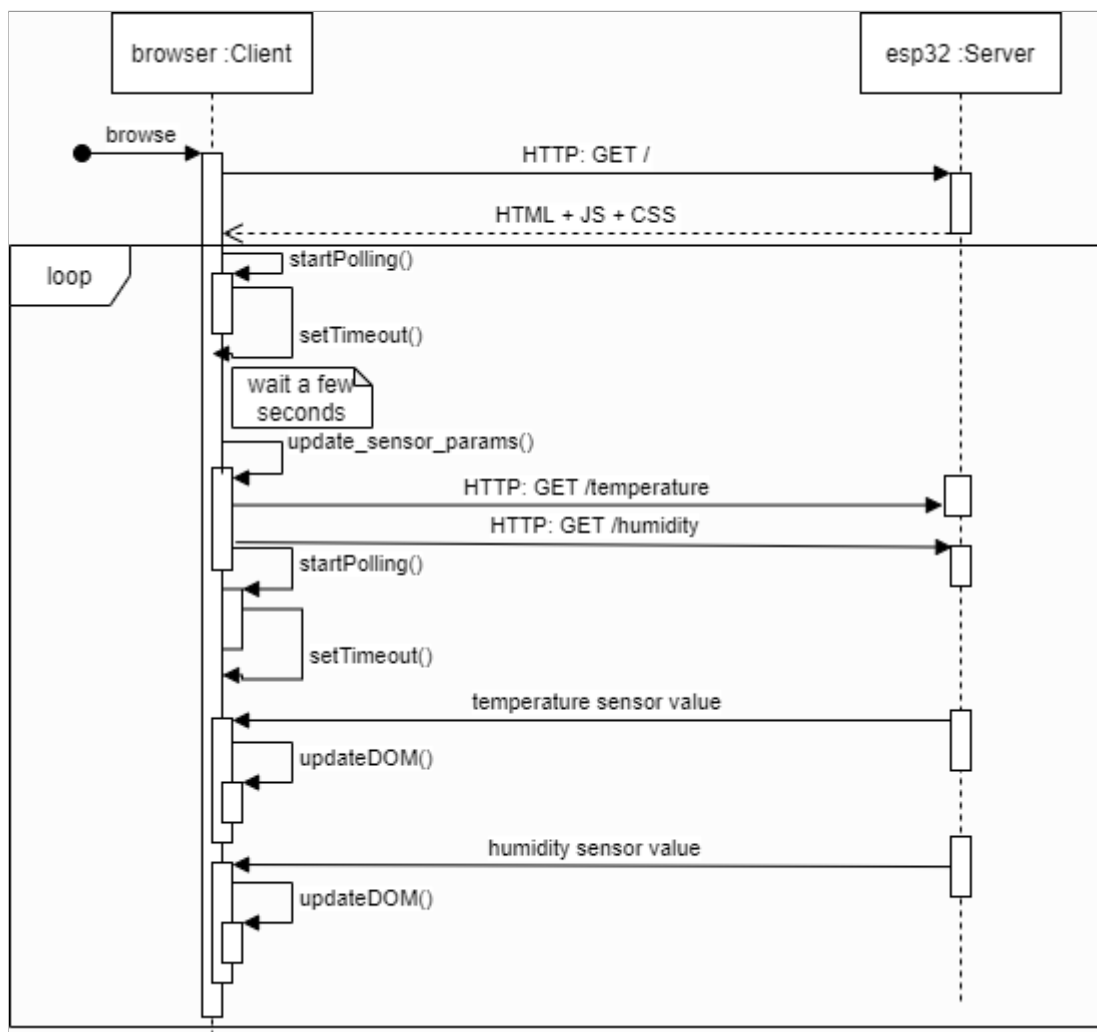
4.5. Távvezérelt működés

A távvezérelt működéshez szükség volt olyan elemek felhasználására, amelyekkel valós időben aszinkron módon nyomon lehetett követni az esetleges beérkező eseményeket (gombnyomás, értékváltozás). Ilyen függvény volt az XMLHttpRequest³² GET kérés, amelyet arra kellett használni, hogy adatot lehessen lekérni a szerverről, illetve az XMLHttpRequest POST kérés azért, hogy frissíteni lehessen egy bizonyos részét a HTML-nek.

4.5.1. GET

A 4.5.1.-es ábra ábrázolja azt a folyamatot, amikor a felhasználó legelső alkalommal jeleníti meg a weboldalt, illetve GET kérések küld annak érdekében, hogy megjelenítse az illető szenzorok aktuális értékeit. Az ábra kezdetén látszik, hogy a kliens rákeres (browse) a weboldalra, azaz HTTP GET kérést küld a gyökér(root) számára, ezáltal a szerver válaszként visszaküldi a weboldal HTML, CSS és JS elemeit, amit meg fog jeleníteni a böngésző. Ezután a kliens elkezd lekérdezni (startPolling), azért hogy a HTML DOM elemeket frissíteni tudja. A setTimeout() függvényre azért van szükség, mert bizonyos időnként előnyös lekérdezni értékeket. A lekérdezések során meghívódik az update_sensors_params() függvény, amely tartalmazza a szenzorok és paraméterek aszinkron GET kéréseit. Az ábrán két szenzor GET függvénye van feltüntetve a következő endpointok számára /temperature és /humidity. Ezt elküldi a szerver számára, aki majd értelmezi és aszinkron módon egy bizonyos idő után választ küldd vissza. Ha megérkezett a válasz a DOM elemek frissítése történik, Mialatt újra a megadott időnként végrehajtódik a lekérdezés.

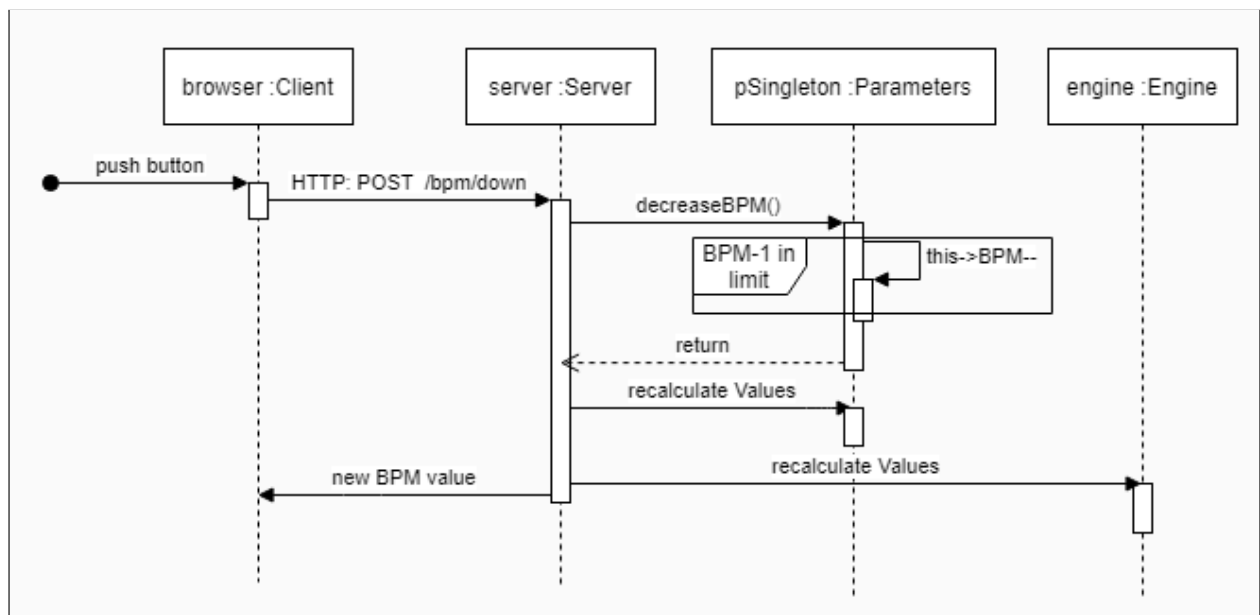
³² XMLHttpRequest - https://www.w3schools.com/xml/xml_http.asp



4.5.1. ábra - Szekvencia diagram - GET

4.5.2. POST

A 4.5.2. ábra összefoglal egy POST kérést, amely során a felhasználó lenyomja a gombot a felületen (jelen esetben a percenkénti lélegeztetés csökkentését), így küld a szerver számára aszinkron módon egy kérést (*/bpm/down* végpontra). Ezt a szerver egy singleton segítségével csökkenti (*decreaseBPM*), majd megnézi, hogy sikeresen a megadott határértékek között maradt-e a szám. Ekkor beállítódik egy flag, ami azt jelzi, újra kell számítani bizonyos értékeket. Az új érték aszinkron módon visszakerül a kliens számára, majd a motor számára is újraszámolást kér.



4.5.2. ábra - Szekvencia diagram - POST

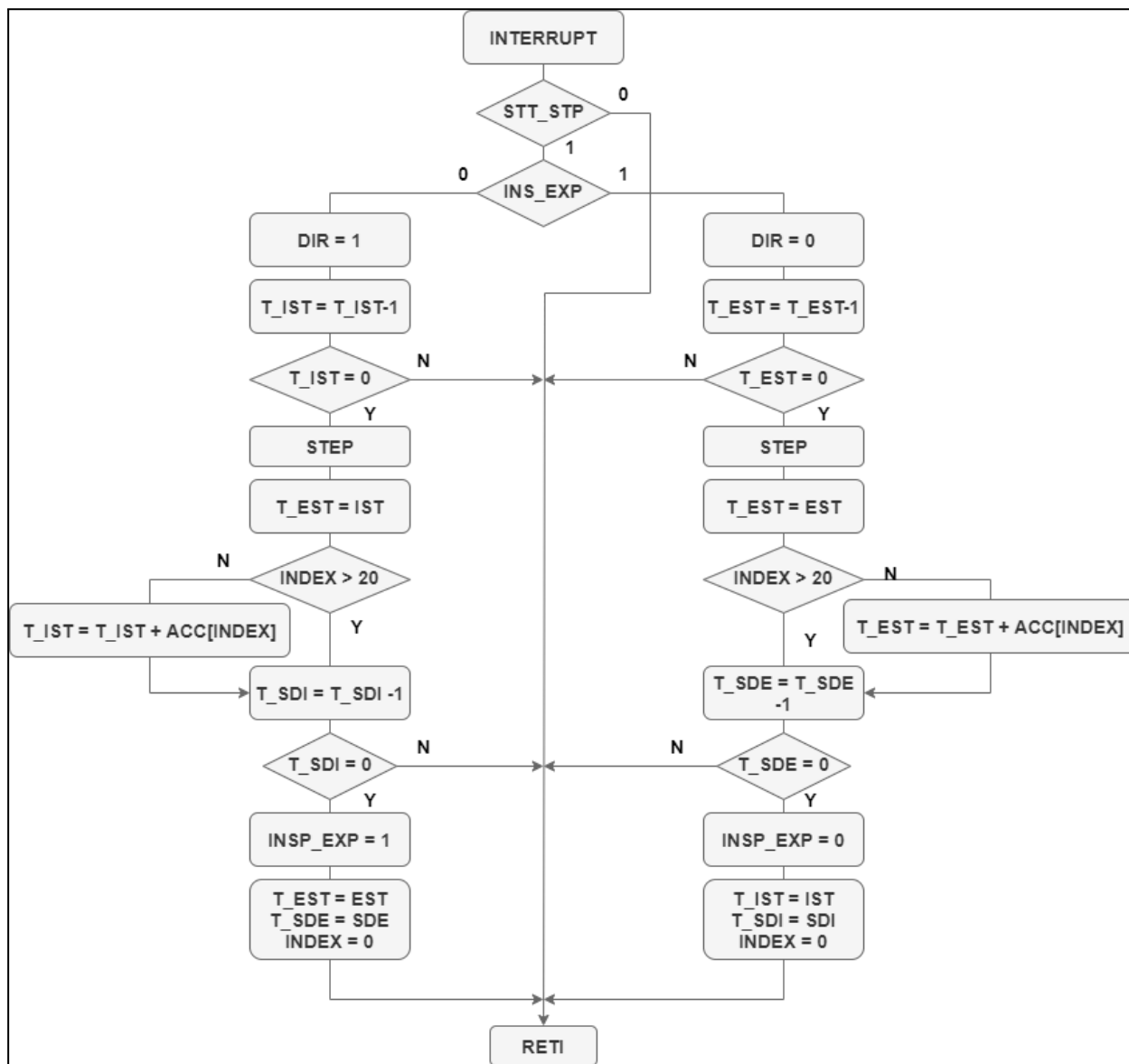
4.6. Megszakítási rutin

Hogy a párhuzamos feladatok szinkronizálását meg lehessen könnyíteni, illetve a valós idejű folyamatokat kezelni lehessen megszakítási eljárásra³³ volt szükség. Az ilyen eljárás lényegében azt jelenti, hogy ha egy megszakításkérés érkezik egy perifériás modultól és az engedélyező bitek értéke egy, akkor az éppen végrehajtásban levő főprogrami utasítás még befejeződik, a következő utasítás címe elmentődik a veremtárba és a program hardveres vezérléssel átugrik a megszakításvektor címére. Tehát ez az a programmemória cím, ahonnan a megszakítás rutin kezdődik. Ugyanakkor, a megszakításvektorra való ugrás pillanatában a GIE engedélyező bit hardveresen törlődik (0-ra áll), azért, hogy egy újabb esetleges megszakításkérés ne legyen kiszolgálva mindaddig, amíg az éppen futó megszakítási rutin be nem fejeződik. Ez az eljárást a léptetőmotor vezérlésénél volt alkalmazva. Az megszakítás jelet az ESP32-nek az idő modulja minden 100 uS-ban kigenerálja. A rutin alatt a mikrovezérlő előre vagy hátra vezérli a léptető motort, a főprogramban kiszámított ütemben és arányban. Ugyancsak itt lett megvalósítva a motor gyorsuló és lassuló vezérlése az irányváltoztatások előtt és után, hogy meg lehessen tartani a maximális nyomatékot minden pillanatban.

A 4.6.1. ábrán látható az alkalmazott megszakítási rutin folyamatábrája, amelyben az STT_STP a léptetőmotor működését figyeli: ha áll akkor visszatér a megszakítási rutinból,

³³ Megszakítási eljárás - <https://docs.microsoft.com/hu-hu/azure/rtos/threadx/chapter3>

különben figyeli, hogy belégzés (0) vagy kilégzés (1) az INS_EXP értéke, ennek megfelelően változtatja a motor irányát előre (DIR = 1) vagy visszafelé (DIR = 0).



4.6.1. ábra - Megszakítási rutin folyamatábrája

Ha belégzés történik, akkor a belélegző tampon értékét csökkenti, ha elérte a nullát akkor visszalép. Ha az nulla lett a leszámolás után akkor visszatér. Ha nem akkor impulzust küld ki aminek hatására a motor elmozdul. Az állóhelyzetből történő elmozdulás segítségére a már említett gyorsulást segítő tömböt használja fel ACC[] néven és azt figyeli, hogy a 20 elemű tömbön végigment-e az index.

Ezután csökkenti a belégzési lépések számát, egészen addig amíg el nem éri a nullás értéket, amikor negálja az INS_EXP értékét, majd ezzel inicializálja a következő megszakítást

ami a kilégzési paraméterek támpontokba történő áthelyezését és az index újból nullára állítása jelenti.

A kilégzés során hasonlóan történnek a folyamatok, annyi eltéréssel, hogy az irány ellentétes, a műveletek a kilégzéshez kötöttek végül ha a kilégzési lépésidő elérte a nullát akkor ismét tagadja az INS_EXP értéket, inicializálja a soron következő belégzéshez az értékeket illetve az indexet nullára állítja, majd visszatér a főprogramba a RETI segítségével.

4.7. A vezérlő programkódja

A felmerült ötletek kivitelezéséhez és a folyamatosan bővülő elvárások kielégítésének érdekében nem lehetett megállni az első működő verziónál, hanem cél volt, hogy további funkcionalitásokkal legyen bővítve az alkalmazás. Így az első működőképes kód nem tartalmazza a távvezérléshez szükséges könyvtárakat és azok alkalmazását, illetve a felhasználói platformért felelős megjelenítéseket, hanem csak a manuális vezérléshez szükséges funkcionalitásokat. A további verziókban már megjelennek ezek, illetve a szenzorok megmért értékei is.

A verziókra általánosan is jellemző, hogy a 3.1.8. ábrának megfelelően voltak az I/O portok kiválasztva az adatok fogadása és kiküldése érdekében.

A manuális vezérléshez említett botkormányok portjainak kiválasztásánál a *btB_VRxBin* nevében megjelenő B betű a *breathing* (lélegeztetés) miatt kapta a nevét, mert ezen a tologombon van a percenként lélegeztetés illetve a ki-és belégzés arány szabályozása. A port nevében lévő x jelenti a vízszintes irány szerinti vezérlést, a *btB_VRyPin* esetében pedig az y jelenti a függőleges irány szerinti vezérlést.

Hasonlóképpen van a *btVP_VRxBin* port, viszont itt a VP megfelelője a *Volume-Pressure*, vagyis itt térfogat és nyomásszabályozás történik. Párja a *btVP_VRyPin*, amely a merőleges irány szerinti vezérlésért felelős kommunikációs port. A *btVP_SelPin* pin felel azért, hogy a másik botkormány térfogat- és nyomásszabályozása között váltakozni lehessen.

A következőkben bemutatásra kerülnek a verziók közötti legfőbb célok és különbségek, illetve a legutolsó verzióban részletesen bemutatásra kerül a már majdnem teljesen modulokra bontott programkód.

4.7.1. Első verzió

Az első verziónak kivitelezése során az volt a cél, hogy a manuális vezérlés helyesen működjön. Első sorban az indítás és megállítás funkcionáljon. Ezt követően az, hogy a gombok megfelelő irányba elmozdítása során helyes műveletet végezzenek, illetve a felhasznált OLED kijelzőre a 4.5 fejezetben található előre meghatározott képletek szerint kerüljenek fel az értékek.

Az első verzióban a számításhoz szükséges értékek és az említett megszakítási rutin változói globálisan voltak meghatározva.

A *setup()* függvényben a botkormányok és a motorvezérléshez szükséges pin-ek voltak meghatározva aszerint, hogy kimeneti (OUTPUT), bemeneti-lenyomásra (INPUT PULLUP) működjének. Ez után következett az OLED kijelző inicializálása. Illetve elindult a TMR0 inicializálása, amely minden 100 µs-ban megszakítást generál.

Ezután következett a *loop()* függvény, amelyben elsőként azt figyelem, hogy lenyomódott-e az indításért felelős tológomb, ha igen akkor inicializálom a megszakítás értékeket és elindítom a motort. Ezt követően folyamatosan figyelem azt, hogy melyik botkormány volt elmozdítva és annak alapján számításokat végzek, amelyeket az OLED számára adok át, hogy megjelenítse. Legvégül azt figyelem ha a motor működésben van akkor a led lámpa fel- és kigyuljon.

4.7.2. Második verzió

A második verzió elsődleges célja az volt, hogy bekerüljenek a weboldalhoz szükséges komponensek, illetve a kliens-szerver kommunikáció létrejöjjön. Emellett függvényekre bontani a lehető legtöbb részt.

A kivitelezés során egy olyan ötlet merült fel, hogy a kijelző inicializálása után pár másodperccel jelenjen meg egy olyan rész, amelyben a mért szenzorok értékei kerülnek ki. Ennek a megvalósítása után, az említett környezetben következett a weboldal tervezés és tesztelés, amelyet, nem előnyös módon a kódban van elhelyezve, mint egy konstans karaktertömb. Ezután azonnal felmerült a továbbfejlesztési lehetőség, hogy a HTML CSS és JS elemek a fő programtól külön szervezve legyenek.

Miután sikeresen megjelenítette a főprogram a weboldalt, megírásra kerültek a GET függvények, amelyeknek köszönhetően a manuális vezérlés során a weboldal alsó részében megjelentek, rövid időeltolással az aktuálisan beállított paraméter értékek. Ezt követően a szenzorok számára is hasonló GET függvények kerültek megírásra.

Ahhoz hogy a távvezérlés valóban sikeres legyen, szükség volt POST függvényekre. Gyakorlatilag a távvezérlés úgy oldódott meg, hogy minden egyes alkalommal, amikor a kliens egy gombot nyomott le, egy POST kérést küldött a szerver számára. Ez a kérés meghívott egy olyan függvényt szerver oldalon, amely a gombnak megfelelő botkormány számára olyan értéket adott, mintha manuálisan lett volna irányítva. Ezt követően kiszámolta az aktuális értéknek megfelelő új értéket és a POST válaszában csatolta ezt a kliens részére, ahol ezt beállította a megfelelő azonosítóhoz tartozó mezőbe.

Végül a főprogramban külön függvénybe kerültek azok a részek, amelyek azért felelnek, hogy megjelenítsék az OLED számára az értékeket, illetve azok a művelet sorok, amelyek azt tesztelik, hogy egy botkormány valóban elmozdult-e, ha igen milyen irányba és a felhasználó elengedte-e a tológombot.

4.7.3. Harmadik verzió

A legújabb verzió célja az volt, hogy az előző verziók hibáiból tanulva, olyan kódot létrehozni, amelyik sokkal átláthatóbb, külön modulokba csoportosítás révén, vagy esetenként osztályokat bevezetni. Emellett a szenzorokat bekötni és mérni az értékeiket, hiszen minden ehhez szükséges megjelenítés és kliens-szerver kommunikáció már el van készítve.

Az első modul az OLED kijelzőre történő műveletekért felel. A *display.h* file tartalmaz öt függvényt, amiből négy a megjelenítésért felel, az első az inicializáláshoz szükséges üdvözlő szöveg (*display_initialize()*), a második amelyik a szenzorok neveit és értékeit tünteti fel (*display_sensorvalues*), a harmadik pedig a beállított paraméterek megjelenítéséért felelős (*display_parameters()*). Végül a *display_ip()*, megjeleníti azt az IP címet, amin keresztül elérhető a weboldal. A törlő függvény meghívásával lehet ezek között váltakozni (*display_clear()*). A header file implementációja megtalálható a *display.cpp* fileban. Itt vannak felhasználva azok a könyvtárak, amelyek az OLED felhasználásához szükségesek. Az Arduino IDE-be beépített *Wire.h* és *SPI.h* könyvtárai illetve, illetve a githubról letöltött *Adafruit_SSD1306.h* és *FreeSerif12pt7b.h* könyvtárak.

Ezek után következett a *display.h*-ban már említett az OLED inicializálásáért felelős függvény amiben legelőször az OLED belső áramköröit kellett elindítani (*display.begin(SSD1306_SWITCHCAPVCC, 0x3C)*), ezután ki kellett választani a felhasznált könyvtár betűtípust, illetve kiválasztani annak a méretét és színét (4.7.1. ábra). A *setCursor()* függvény segítségével lehetett a 128x64 pixeles képernyő bizonyos részeihez navigálni, úgy hogy két paramétert kellett megadni a két pixel közül. A *println()* függvény paramétereként

lehetett megadni, hogy milyen szöveg jelenjen meg a kijelzőn. Majd a teljes szöveg megjelenítéséért a *display()* függvény meghívása kellett.

```
display.setFont();  
display_clear();  
display.setTextSize(1);  
display.setTextColor(WHITE);
```

4.7.1. ábra - OLED karakter inicializálás

A második függvény felelt a szenzorok értékeinek megjelenítéséért, ezeket paraméterekként kapta meg, majd hasonló módon mint az előző függvény előkészített az OLED-et amelynél az illető szenzor neve után hozzáfűzte a paraméterül kapott értékét, majd a szenzorhoz tartozó mértékegységet. Ezután a megjelenítő függvény következett.

A paraméterek megjelenítésére szolgáló függvény hasonlóan a szenzor megjelenítő függvényhez inicializál és jeleníti meg az karaktereket, viszont a szenzorok értékeit egy singleton függvény segítségével éri el a *Parameters.h* osztályból. Erre egy példa: a percenkénti belégzések számának eléréséért a *Parameters::getInstance().getBPM()* függvény kerül meghívásra.

Végül a törlő függvény az OLED tartalmát távolítja el.

A második modul az internetkapcsolat megteremtéséért felelős. A *wifi.h* file tartalmaz egyetlen függvényt, amelynek neve *connect_to_wifi()*. A megvalósítás során egy olyan probléma merült fel, hogy nem túl előnyös egy router jelszavát és SSID-ját a kódba beleégetni, hiszen minden egyes alkalommal, ha a lélegeztető gép más helyszínre kerül újból kellene futtatni rá a már kicserélt paraméterű kódot. Emiatt egy olyan megoldásra volt szükség, amelyben az esp32-öt, mint egy routert használunk fel, ezáltal szolgáltatva számára egy olyan biztonságot, amit az esp32 WPA2 szintű titkosítása nyújthat, hiszen csak azok a személyek tudnak a mikrovezérlőre csatlakozni akik előre ismerik a megadott jelszót. Erre megoldás volt a SoftAp³⁴ felhasználása, amely egy virtuális router és az ehhez tartozó függvény paramétereként meg kellett adni a jelszót és az SSID-t. Ez a függvény a főprogramban lesz meghívva.

A következő modul neve a *server.h* itt találhatóak azok a részek, amelyek a webserver kezeléséhez szükségesek. Első sorban az aszinkron szerver létrehozásához szükséges könyvtár (*ESPAsyncWebServer.h*) van felhasználva, majd a szenzor működéséhez szükséges könyvtárak (*Wire.h*, *Adafruit_Sensor.h*, *Adafruit_BME280.h*). Ezt követően létre van hozva egy osztály *BreathingServer* néven, amelynek privát adatai egy *AsyncWebServer* példány, és egy *Adafruit_BME280* szenzor példány. Emellett még privát öt olyan metódus, amely előkészíti a kliens által hívott végpontokra a választ. Ezekből az első csak a szenzorokért és a következő

³⁴ SoftAp - <https://en.wikipedia.org/wiki/SoftAP>

négy pedig a paraméterekért felel. Az osztály publikus konstruktorában előre meg vannak határozva az aszinkron szerver 80-as számú portja és a szenzor típusa. Emellett található egy bool típusú metódus *setupAndStart()* néven.

A `server.cpp` fájlban található legelől egy `const static char index_html[]` tömb amelyet egy `PROGMEM` szó követ. Ez egy jelzés a fordító számára, hogy a következő információkat a flash memóriába tegye, az alapértelmezett SRAM helyett, így nem terhelve azt. Ebben a tömbben található az összes HTML, CSS és JS utasítás.

A HTML head részében találhatóak a JS utasítások, a body részében a weboldalon megjelenő szöveg részek, elrendezés, illetve minden egyes nyomógombnak külön ID van rendelve, amelyeket lenyomva a JS segítségével számos műveletet lehet elvégezni. A weboldal aljában található táblázatban minden értékhez ugyancsak saját ID van létrehozva, azért, hogy a változó értékeket meg lehessen jeleníteni majd. A body részének legalján van a CSS. Itt olyan hibát kellett megoldani, amely kiküszöböli azt, hogy egy új sort `#` karakterrel lehessen kezdeni, ugyanis a C/C++ ezt úgy kezelte, mintha a direktívája volna, vagyis speciális parancsként. Ez akkor következett be amikor osztályoknak kellett CSS-ben kinézetet szolgáltatni. Ezeket `#` karakterrel kellett megjelölni, ezért elsősorban le kellett cserélni az osztályokat illetve ahol nem lehetett, a *#osztálynév* elé vesszővel elválasztva egyéb nem osztálynév volt használatban.

Az JS függvények közül, melyek nem a HTML webserver kommunikációhoz szükségesek, azért felelnek, hogy az oldalon megjelenítésének illetve eltüntessenek bizonyos részeket.

Az első függvény neve `showOneHideTwo()`, amelynek az a szerepe, hogy a paraméterül megadott elemek közül a képernyő közepére egy időben csak egyet jelenítsen meg. Ilyen elemből három található, majd ezeknek meg kell változtatni CSS parancsként a megjelenítését (például `display = block`, `display = none`).

A `switchMode()` függvény lényege, hogy amikor manuális üzemmódban van a rendszer, ne jelenjenek meg azok a részek, amelyek csak a távvezérelt üzemmódban találhatóak, tehát csak a szenzor értékeket mutassa középén.

Ahhoz, hogy a súgó szöveg látható és elrejtető is legyen, a kérdőjel gombnak egy olyan függvényt kellett hívni, amelyik megjeleníti illetve elrejt a megadott szövegrészt, ennek neve *showHideHelp()*.

A webszerver elsődeleges függvényéhez tartozik a *switch_mode()*, amely a SWITCH gomb lenyomásával hívódik meg. Ebben a függvényben változóba menti a gomb elemet ID szerint és letiltja a kattinthatóságát. Kiküld egy új XMLhttprequestet és egy eseménykezelőt (*onreadystatechange*), amely akkor hívódik meg ha a *readyState* értéke megváltozik, ami a

felhasználói felülethez kapcsolódó szálból hívódik meg. Ha ez sikeresen lefutott (a függvényben levő if feltétel beteljesült), akkor a SWITCH nyomógomb újra kattintható. Végül elküld egy POST kérést a `/switchmode` endpoint számára.

A következő függvényben folyamatosan GET kérések vannak implementálva a szerver számára (`update_sensors_params()`). Itt minden alkalommal azt figyeli, hogy ha a szerveren belül megváltozott az érték akkor frissítse az új érték tartalmára a megfelelő DOM elemet. Itt található a négy szenzor és a négy paraméter kérései. Minden változás után a következő ciklusban ID szerint az `innerHTML` DOMString segítségével a válaszszoveget frissíti.

A gombok lenyomásáért felelős függvények külön-külön szerepelnek a JS részben, tehát ha például a percenkénti belégzések számát szeretnénk csökkenteni, akkor az ahhoz tartozó `bpm_down()` gomb lenyomásával POST kérést küld el a szerver számára.

A `server.cpp` file másik felében a `server.h` file függvényei vannak implementálva. Például a `setupPPSEndpoints()` függvény paraméterében bool típusú cím szerinti érték van átadva. A függvény belsejét a 4.7.2. ábra foglalja össze.

```
server.on("/pps", HTTP_GET, []) (AsyncWebServerRequest *request) {  
    request->send(200, "text/plain", String(Parameters::getInstance().getPPS()));  
};
```

4.7.2. ábra - Plateau Pressure GET kérés

A kódsor azt jelenti, hogy a `/pps` endpointra aszinkron módon egy HTTP GET kérést küld. A 200 jelenti, azt hogy sikeres volt a küldés, a `text/plain` string kifejezi, hogy egy szöveg lesz visszaküldve, illetve a `Parameters::getInstance().getPPS()` segítségével lehet elérni a PPS új értékét. Hasonlóan a többi három paraméterhez implementálva van a szerver oldali válasz.

A `pps_down` HTTP POST kérése a 4.7.3. ábrán látható.

```
server.on("/pps/up", HTTP_POST, [&didChange] (AsyncWebServerRequest *request) {  
    Parameters::getInstance().increasePPF();  
    didChange = true;  
    request->send(200, "text/plain", String(Parameters::getInstance().getPPS()));  
});
```

4.7.3. ábra - Plateau Pressure POST kérés

A lambda kifejezésben megadott referencia (`didChange`), arra szolgál, hogy a kérés hozzá tudjon férni a bool kifejezéshez, aminek igazra állítása jelzi, hogy az egyik nyomógomb a kliens oldalon megváltozott és emiatt új számításokra van szükség. Kiszámítódik az új érték, a küldött változás igaz lesz és válaszként visszaküldi az új kiszámított értéket.

Végül az összes ilyen *Setup* függvény elindítója a `setupAndStart()` függvény, amely a root (gyökér) számára érkezett HTTP_GET kérésben, visszatéríti az `index.html` file tetején definiált

oldalt. Ezt követően meghívja az öt endPoint függvényt. Első indításkor a *changedVP* és a *changedB* értékei hamisak. Egy másik HTTP_POST válasz is itt van implementálva, ez a */switchmode* endpointra történő kérés. Végül itt kell a *server.begin()* függvény segítségével elindítani az aszinkron szervert (4.7.4. ábra).

```
void BreathingServer::setupAndStart(bool& changedVP, bool& changedB){
    server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request){
        request->send(200, "text/html", index_html);
    });
    setupSensorEndpoints();
    setupBPMEndpoints(changedB);
    setupCDREndpoints(changedB);
    setupPPSEndpoints(changedVP);
    setupTVSEndpoints(changedVP);

    server.on("/switchmode", HTTP_POST, [] (AsyncWebServerRequest *request){
        request->send(200, "text/plain","");
    });
    // Start server
    server.begin();
}
```

4.7.4. ábra - BreathingServer inicializálása

Mivel a globális paraméterek kiküszöbölése a jobb optimalizálás érdekében elengedhetetlen ezért a kiválasztott értékek a *parameter.h* modulba kerültek. Itt vannak privát adattagként az alapértelmezett értékek, a faktorok illetve a paraméterek minimum és maximum értékei definiálva. Ezek megfelelő változásáért felel a *changeValue()* függvény, amely leegyszerűsíti az új érték kiszámítását, hiszen ötvözni lehetett a két esetet, úgy hogy a függvény híváskor előre meg kellett adni a minimum, maximum értékeket, cím szerint, hogy melyik paraméter lesz módosítva illetve (*value*), illetve, hogy eggyel történő növelés vagy csökkentés történik (*changeBy*). A függvényen belül kiszámítódik az új érték és ha az benne található a minimum és maximum értékek között, akkor az illető paramétert lecseréli az új értékre (4.7.5. ábra).

```
void changeValue(int maximum, int minimum, int* value, int changeBy) {
    int newValue = *value + changeBy;
    if(newValue <= maximum && newValue > minimum){
        *value = newValue;
    }
}
```

4.7.5. ábra - Új paraméter értéket számoló függvény

Mivel egyes függvényeknél számos paramétert kell átadni (például a szenzorok megjelenítésért felelősnek) ezért az egyke programozása mintát felhasználva egy olyan példányt kell létrehozni,

amelyet minden modulból el lehet érni. Ennek a konstruktorát privátra kellett állítani és egyetlen egy statikus példányt létrehozni.

Mivel a *parameter.h* fájlban a függvények nagy része csak egy visszatérítést ezért nem készült jelenleg *.cpp* az implementáció érdekében. Ilyen függvények a getterek (4.7.6 - 4.7.8. ábrák), illetve azok a függvények, amelyek az új értékek kiszámításáért felelősek. Erre példa a Tidal Volume Setup kiszámítása a faktor segítségével (4.7.9. ábra).

```
//Tidal Volume Setup
int getTVS() const{
    return TVS;
}
```

```
//breath/min
int getBPM() const{
    return BPM;
}
```

```
//Cycle Division Ratio
double getCDR() const{
    return CDR;
}
```

4.7.6. - 4.7.8. ábrák - Egyszerű get függvények

Végül azok a függvények vannak implementálva, amelyeket a már említett *server.h* függvényei minden kliens nyomógomb eseményre meghívnak(4.7.10. ábrák)

```
void calculateTVS() {
    TVS = 250 + 10 * TVF;
}
```

```
void increaseBPM() {
    changeValue(maxBPM, minBPM, &BPM, +1);
}
```

4.7.9. ábra - Tidal Volume Setup számítás

4.7.10. ábra - Breaths/min növelés

A következő modul szerepe az, hogy manuális üzemmódban a botkormányok minden elmozdítására egy függvényt hívjon meg (*button.h*). A két botkorány függvénye, amely az figyel, hogy vízszintes vagy függőleges irányban történt-e elmozdulás nevei *isButtonBUsed()* és *isButtonVPUsed()*. Emellett található egy enum típusú felsorolás *Direction* név alatt, amely a botkormányok öt irányát sorolja fel (UP, DOWN, LEFT, RIGHT, CENTER). Külön szerepel két függvény, amely azt nézi le voltak-e nyomva a botkormányok *isButtonBPushed()* és *isButtonVPPushed()*.

A *button.cpp* implementálja az előbb említett függvényeket. Legfelül vannak azok a konstans értékek, amelyek pinjéhez vannak csatlakoztatva a botkormányokon történő műveletek. Minden egyes ilyen művelet amely vízszintes és függőleges irányban észlel elmozdulást egy int típusú számot térít vissza 0 és 3300 között. Ha felfelé és jobbra történik az elmozdulás akkor biztosan 1650 alatti számot küld vissza, lefelé és balra történt az elmozdulás akkor 1650 és 3300 közötti értéket lehet kiolvasni. Mivel a gyakorlatban nem észszerű, hogy a nagyon kicsi elmozdulást a középponttól is elmozdulásának számítsa, ezért inkább kisebb értéket kell nézni, ezért lett a minimális határérték pozitív irányban 0 és 1400 között, illetve a maximális értékek negatív irányban 2300 és 3300 között. Ezek ellenőrzésére segédfüggvények voltak szükségesek, amelyek

azt nézik, hogy benne található-e abban a határértékben. Erre példa a 4.7.11 ábra, amely függvénye igaz értéket térít vissza, ha a gomb értéke (*buttonValue*) 1400 alatt található.

```
bool isButtonPositive(int buttonValue)
{
    return buttonValue < 1400;
}
```

4.7.11. ábra - Nyomógomb irányt ellenőrző függvény

Emellett azt is nyomon kell követni, hogy ezeket a gombokat mikor engedi el a felhasználó, ezért figyelni kell azt is, hogy ezek a gomb értékek mikor érnek vissza középpállásba több, mint 1400 és kevesebb, mint 2300 közé. Végül a kiválasztott értékek a 4.7.12. ábrán találhatóak (1600-tól 2000-ig számít a botkormány elengedésének).

```
bool isButtonReleased(int buttonValue)
{
    return buttonValue < 1600 || 2000 < buttonValue;
}
```

4.7.12. ábra - Nyomógomb felengedést ellenőrző függvény

A teljes botkormány elmozgatásáért felelős rész pedig az enum típusú *directionButtonB()* és *directionButtonVP()* függvények mutatják be. Ebből a *directionButtonB()* implementációja a 4.7.13 ábrán látható. Itt első lépésben az történik, hogy elmentem egy változóba (*btB_VRx*) azt az értéket amid visszaküld az *analogRead()* függvény.

```
Direction directionButtonB() {
    int btB_VRx = analogRead(btB_VRxPin);
    if (isButtonPositive(btB_VRx)) {
        return RIGHT;
    }
    if (isButtonNegative(btB_VRx)) {
        return LEFT;
    }
    int btB_VRy = analogRead(btB_VRyPin);
    if (isButtonPositive(btB_VRy)) {
        return UP;
    }
    if (isButtonNegative(btB_VRy)) {
        return DOWN;
    }
    return CENTER;
}
```

4.7.13. ábra - Enum használata az irányok meghatározásáért

Utána első esetben azt kell nézni, hogy vízszintes irányban, vagy hogy pozitív értéket térít-e vissza, ha igen akkor tudom, hogy jobbra történt az elmozdulás. Ha negatív irányban akkor balra. Ezt követően ugyanígy kell tenni függőleges irányban is csak akkor a megfelelő értékeknek UP/DOWN érték térül vissza. Ha egyik eset sem teljesült akkora CENTER esetet kell visszatéríteni (4.7.13 ábra).

Az *engine.h* tartalmazza a megszakítási rutinhoz szükséges változók gettereit és settereit, illetve egy *enableEngineRoutine()* függvényt, amely elindítja és megállítja a motorvezérlést.

Az *engine.cpp* file első sorban tartalmazza a motorvezérléshez szükséges paramétereket volatile jelzővel. Erre azért van szükség beágyazott rendszereknél, mert ezek globális változók és jelezni kell a dekódernek, hogy ezek értéke bármikor megváltozhat a megszakításon belül, anélkül, hogy a program bármilyen intézkedést végrehajtania. A file ezen felül tartalmazza a megfelelő I/O portokat az irány és a lépés számára, amelyet a motorvezérlőnek küld ki közvetlen.

A globális változók, amelyeket fel kell használni a megszakítási rutinban első sorban a belégzéshez kötöttek (lépések száma belégzéskor, belégzési lépésidő), másodsorban a kilégzéshez (lépések száma kilégzéskor, kilégzési lépésidő), mind tartozik egy ezzel egyenértékű tampon, amely értéke le lesz számolva a megszakítás rutin során, annak érdekében, hogy ne veszítsük el az eredeti értékeket. Emellett található egy légzéstípust (ki-/belégzés) nyomon követő bool változó, melynek hamis értéke a belégzés és igaz értéke a kilégzés, illetve egy motorműködést ellenőrző bool típusú változó, amelynek hamis értéke ha a motor működésben van, illetve igaz értéke ha a motor leállt. A motor gyorsulást segítő tömb is itt van feltüntetve, aminek az indexeit az *STS* kezdetleges 0 értékének változtatásával lehet elérni.

A motor indításáért és időzítőjének beállításáért az *enableEngineRoutine()* függvény felel (4.7.14. ábra). A függvény első két sora 80 MHz frekvenciával működjön. Illetve referenciaként átadja magát a megszakítási rutint. A következő két sor pedig beállítja, hogy 100 µs alatt megszakítást hívjon meg.

```
void enableEngineRoutine() {  
    hw_timer_t * timer = timerBegin(0, 80, true);  
    timerAttachInterrupt(timer, & onTimer, true);  
    timerAlarmWrite(timer, 100, true);  
    timerAlarmEnable(timer);  
}
```

4.7.14. ábra - A megszakítási rutint engedélyező és inicializáló függvény

A megszakításrutin implementálásban első sorban azt figyeli ha a motor elindult és a légzési flag belégzést mutat. Ha igen akkor a motor iránya jobbra állítódik (*digitalWrite(DirPin, HIGH)*),

csökkenteni kell a tampont értéket és ha ez nulla akkor át lehet menni a kilégzési részhez. Ha nem nulla akkor minden egyes alkalommal amíg nullára nem csökken, a StepPin számára egy HIGH jelet kell adni, pontosabban annyiszor ahány lépés van a belégzés során. Tehát ez alapértelmezetten a gyakorlatban úgy nézne ki le fog futni 2000-szer (belégzési lépés idővel). Mialatt minden alkalommal hozzárendel egy motorgyorsulást segítő változót, ami a belégzési lépés számhoz 20 ciklus idejéig. És miután megállt a ciklus a konstans belégzési lépésidőt rendeli hozzá. Ezt követően a következő ellenkező irányba tartó kilégzés (*digitalWrite(DirPin, LOW)*) és hasonló módon, mint az előzőhöz a kilégzéshez tartozó változók és tampon párjaikat felhasználva lehet számolni. Minden ilyen alkalommal egy HIGH jelet küldve a motorvezérlő számára. Ha leszámolta 2000 lépést a ciklus a légzési flaget vissza kell állítania, hogy a következő lépésben újra belélegzés legyen.

Végül az összes modult vezérlő főprogramban vannak elsősorban a megírt fileok tartalmait meghívva. Egyes globális bool típusú változók, amelyeket még nem sikerült külön modulokban felhasználni itt maradtak. Ezek főként a botkormányokkal kapcsolatosak. Az egyik pár figyelmeztet, hogy a botkormány lenyomásból fel lett-e engedve (*Prel_B, Prel_V*), a következő pár azért felel, hogy jelezze ha valóban egy valós elváltozása történt-e a botkormányoknak (*ActualRead_B, ActualRead_VP*). Illetve egy FirstClear, amely azt vizsgálja majd, hogy az OLED display tartalma el legyen távolítva.

Mivel az I/O portokat csak a setup() függvényben lehet beállítani, emiatt nem sikerült őket átszervezni a *buttons.h* forrásba. A soros kapcsolat létrehozásához az értéket az eddigi 115200-ról le kellett csökkenteni 9600-ra hiszen túlságosan kevés idő maradt arra, hogy a szenzorok értékeit le lehessen olvasni.

A következő részben első indításra az üdvözlő szöveg jelenik meg, ezt követően értékei a szenzorok vagy a paraméterek értékei jelennek meg annak függvényében, hogy a motor el van indítva vagy sem. Végül itt hívódik meg a connect_to_wifi() függvény és a szerver példány számára a setupAndStart() alapértelmezetten két false értékkel, hiszen a gombokkal még nem történt vízszintes vagy függőleges irányú elmozdulás. A már említett *enableEngineRoutine()* függvény inicializálja a léptetőmotort.

Ezután következik a *loop()* függvény amely folyamatosan lefut. Itt elsősorban a motorindítás van nyomon követve, úgy hogy ha a megfelelő botkormány először le volt nyomva akkor első indításra még vár egy ciklust, utána ha megint lenyomás történik (így számít validnak az indítás) akkor a motor megindul (*digitalWrite(EN_Pin, HIGH)*) és megjeleníti kijelzőn a paraméterek értékeit. Illetve beállítja a tampon számára a belégzési lépések számát, illetve a kilégzési lépések

számát. Ugyancsak itt vannak implementálva a már részletesen leírt botkormány elmozdulást figyelő függvények is. A már implementált enum értékekhez itt egy switch tartozik, mindegyik meghív egy ahhoz tartozó műveletet (4.7.15. ábra). Például az UP esetre a *Parameters::getInstance().increaseCDF()* függvényt. Ezt követően az ehhez tartozó változási flaget(*ActualReadB*) igazra állítja majd ez a *loop()* függvényben van jelentősége, hiszen ha egyesre állítódott az értéke akkor újra műveleteket végez el.

```
switch (direction)
{
case UP:
    Parameters::getInstance().increaseCDF();
    break;
case DOWN:
    Parameters::getInstance().decreaseCDF();
    break;
case RIGHT:
    Parameters::getInstance().increaseBPM();
    break;
case LEFT:
    Parameters::getInstance().decreaseBPM();
    break;
}
```

4.7.15. ábra - Az enum irányoknak megfelelő függvényhívások a főprogramban
Ha az *ActualReadB* igaz értéket kap akkor a 4.7.16 ábrán lévő függvények hívódnak meg.

```
Parameters::getInstance().calculateBCT();
Parameters::getInstance().calculateCDR();
Parameters::getInstance().calculateICT();
Parameters::getInstance().calculateECT();
```

4.7.16. ábra - *ActualReadB* igaz értékekor a függvényhívások
Ha az *ActualReadVP* számára lesz igaz érték akkor 4.7.17. ábrán lévő függvényeket hívja meg.

```
Parameters::getInstance().calculateTVS();
Parameters::getInstance().calculatePPS();
setSDI(Parameters::getInstance().getTVS() / CBV);
setSDE(getSDI());
```

4.7.17. ábra - *ActualReadVP* igaz értékekor a függvényhívások
Ha vagy az *ActualReadB* vagy az *ActualReadVP* értéke igaz lesz akkor kiszámítódik az új ki- és belégzési ciklusidő és visszatérnek hamis értékre a figyelő flagek (4.7.18. ábra). Majd minden ilyen sikeres számítást egy csipogó hang követ

```
setIST(Parameters::getInstance().getICT() / getSDI());  
setEST(Parameters::getInstance().getECT() / getSDE());  
  
ActualRead_B = false;  
ActualRead_VP = false;
```

4.7.18. ábra - ActualReadB vagy ActualReadB igaz értékekor a függvényhívások

Az *isButtonVPPushed()*, teljesülésével, megjelenik egy szöveg, ami arra kéri a felhasználót, hogy a böngészőbe írja bele az IP címet, ahhoz hogy a gépet távvezérelten tudja használni.

Az előbb implementált modulokat az I. mellékletben levő komponens diagram mutatja be

4.8. Felhasználói platform

A felhasználói platform során a választott nyelv az eddigi fejlesztéshez mérten az angol maradt. Az volt a cél hogy a legegyszerűbben és legerősebben legyenek megjelenítve a felhasználhatók számára funkcionalitások, mert rájuk lesz bízva más emberek pillanatnyi egészségi állapotuk, így segítve őket azzal, hogy könnyebben el tudják végezni a kijelölt munkájukat.

A tervezés első fázisában a már működő irányításhoz hasonlóan a felhasználói platformra fel kellett kerülnön a négy nyomógomb párt, amelyekkel a paraméterek értékeit lehetett szabályozni. Ezek a felhasználói platformon párban, név szerint a légzési térfogat és légúti plató nyomás, illetve a belégzések száma/perc és a lélegeztetési arány. A platformon emellett fontos volt egy olyan gomb feltüntetése, amely arra szolgál hogy a két üzemmód (távvezérelt és manuális) között biztosítsa a váltást.

A tervezés megvalósításához, a már említett HTML volt felhasználva, amelyben a kinézet CSS-sel volt díszítve, az oldalon történő interakciókat JavaScript felhasználásával valósítottam meg.

A weboldal tervezés első fázisában az volt a cél, hogy a már említett tologombok fel legyenek tüntetve, a szenzorok mért értékei is láthatóak legyenek és legfőképp az egyik nyomógomb azért szolgáljon, hogy lehessen váltani távvezérelt illetve manuális üzemmód között. Az elkészített első verzió, amely összefoglalja az eddigi elvárásokat a 4.8.1 - 4.8.4 ábrákon láthatóak.

Remote Controlled Breathing Machine

Current Operation Mode:

MANUAL

SWITCH

Tidal Volume +

Plateau Pressure -

Plateau Pressure +

Tidal Volume -

Breaths/Min +

Cycle Ratio -

Cycle Ratio +

Breaths/Min -

4.8.1. ábra - A UI üzemmódváltó V1

A 4.8.1. ábrán megtekinthető a UI címe: *Remote Controlled Breathing Machine*, a jelenleg működésben lévő üzemmód és az alatta megjelenő érték, amely jelen esetben MANUAL. A SWITCH nyomógomb pedig arra szolgál, hogy át lehessen váltani REMOTE CONTROLLED üzemmódra egyaránt.

A beállítható paraméterek megnevezése követi a már meglévő gomb-struktúrát: A tologombokat felváltották az vízszintes és függőleges ellentett érték-párok. Ez látható a 4.8.2. ábrán is. A páronkénti elrendezés megegyezik a tologombokéval, viszont a két egymás mellett elhelyezkedő tologomb, a UI felületén most egymás alá került.

4.8.2. ábra - A UI nyomógomb része

Temperature: °C

Humidity: %

Pressure: Bar

SpO2: %

Current Values

Tidal Volume: 400

Plateau Pressure: 20

Breaths/Minute: 20

Cycle Ratio: 1.2

4.8.3. ábra - UI szenzor része

4.8.4. ábra - UI aktuális paraméter megjelenítő része

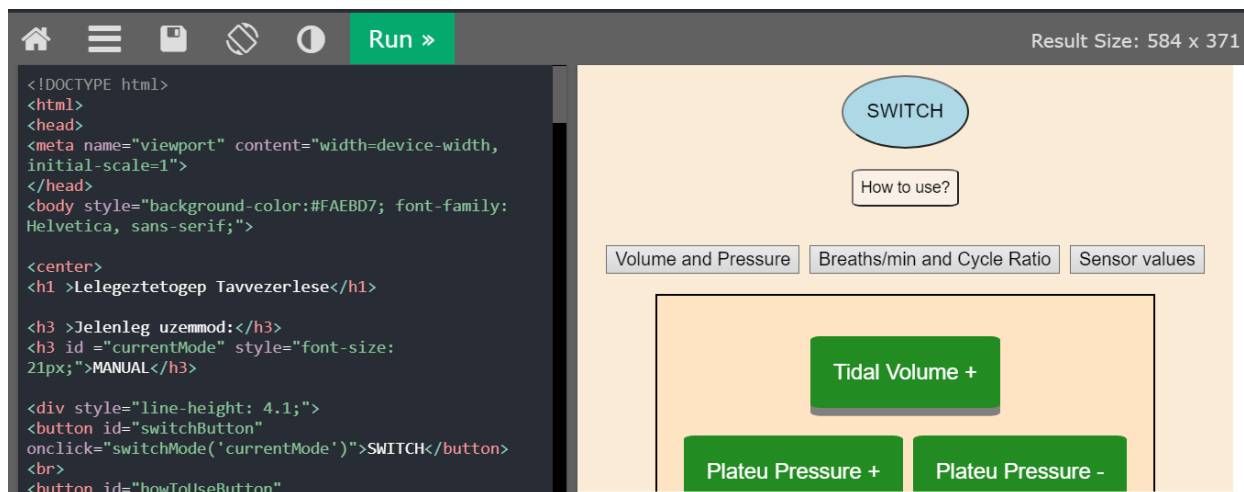
A 4.8.3. ábra szemlélteti a szenzorok kezdetleges megnevezését illetve a mennyiségeket amelyekkel szeretnénk megmérni.

Végül a 4.8.4. ábrán látható a UI legalsó része, amely bemutatja hogy a jelenleg beállított paramétereknek milyen értékek felelnek meg.

Az első tervezés megvalósítása után, olyan ötlet merült fel, hogy ne kelljen a felhasználót arra kényszeríteni, hogy több nyomógomb és paraméter vegye körül, illetve ne kelljen olyan sokat görgetnie az egyoldalas felhasználói felületen, hogy a UI három részre legyen felbontva. Ami közös minden résznél az a már 4.8.3. ábrán, illetve a 4.8.4. ábrán megjelenő részek. Ezek az

oldal legtetején illetve a legalján szerepelnek. Tehát a közös részek közötti három részből az első kettő megtalálható a 4.8.2. ábrán, vagyis a két nyomógomb pár külön-külön illetve a 4.8.3. ábrán látható szenzorok.

A specifikációban említett online felhasználói felületen zajlott a fejlesztés, amelyről egy a az első tervezés utáni változat 4.8.5. ábrán látható. A bal oldalon szerepel a kód, illetve jobb oldalon látható az eredmény a RUN parancs futtatása után.

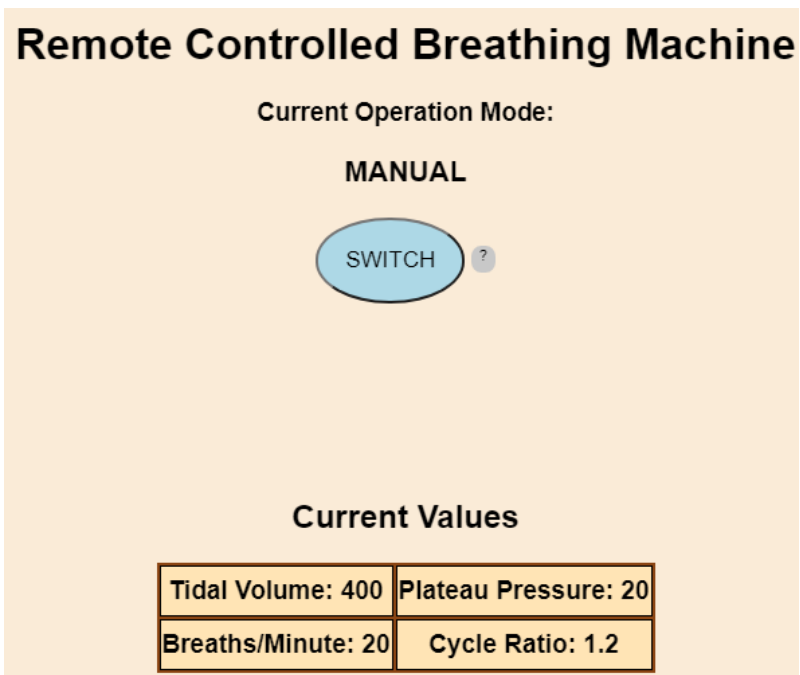


4.8.5. ábra - Kezdetleges UI és a fejlesztői környezet

A UI második tervezésének megvalósításánál már fontossá vált azt is, hogy a kinézet tartson egy olyan szintet, amely megfelel a végfelhasználók számára: Az orvosok és asszisztensek számára legyen színes, egyértelmű (a UI felületén lévő interakciókról magyarázat mellékelve, ha szükséges) és minél inkább emlékeztessen a manuális üzemmód jellegzetességeire (jobb oldalt és felül legyen a paraméterek növeléséért felelős nyomógomb, illetve bal oldalt és alul legyen a paraméterek csökkentéséért felelős nyomógombok, emellett gombnyomás esetén 3D-s lenyomással járó hatás következzen be).

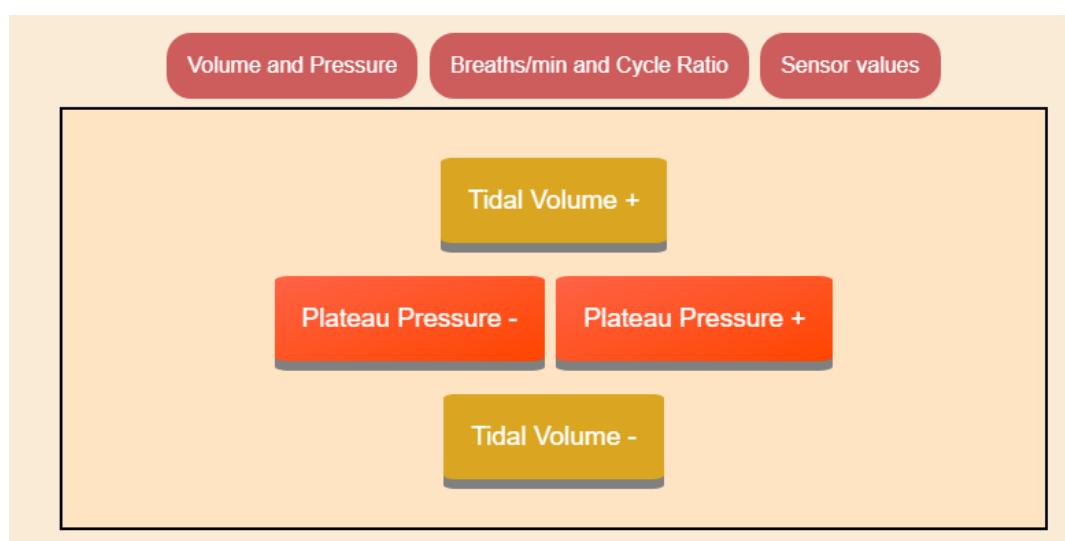
A 4.8.6.-4.8.9. ábrákon keresztül szemléltetve vannak a második tervezés után megvalósított UI.

A 4.8.6. ábrán látszik, hogy mivel nem lehet manuális üzemmódban paramétereket szabályozni, ezért azokat a részeket amelyek ezért felelősek, nem jelennek meg. Viszont a már említett közös rész az érvényes mindkét üzemmódra: a cím, a jelenlegi üzemmód és a SWITCH üzemmód változtató nyomógomb, illetve az oldal alján táblázat formában az aktuális paraméterek értékei.

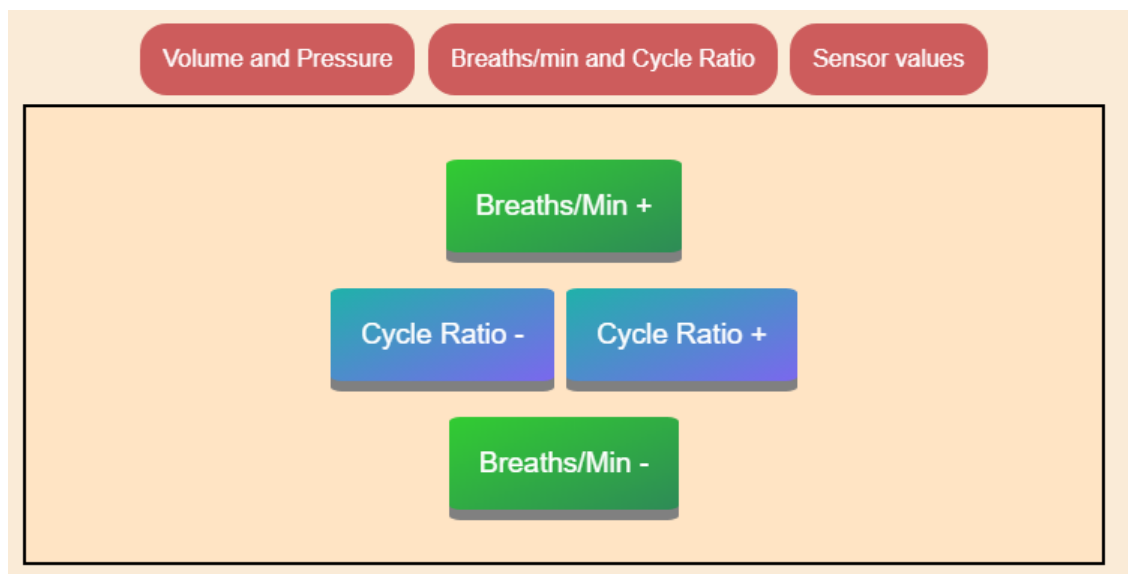


4.8.6. ábra - UI Manual üzemmód

A 4.8.7. ábrán látszik a távvezérelt üzemmódra kapcsolás után megjelenő három nyomógomb, amelyek lenyomása a második tervezésben felmerülő részekért felelős. Tehát ez a három utólag bevitt nyomógomb feladata, hogy a tervben szereplő három részt egymástól függetlenül meg lehessen jeleníteni. Ugyancsak a 4.8.7. ábra alsó részében látható, hogy a fent található Volume and Pressure fül lenyomása után a belégzéshez szükséges térfogatot és nyomást lehet szabályozni.

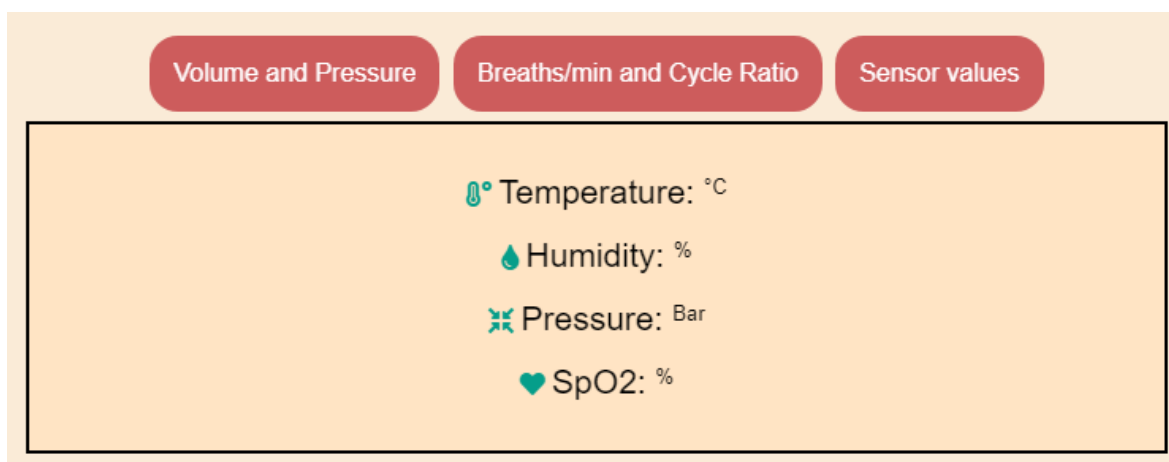


4.8.7. ábra - UI Távvezérelt üzemmód - Térfogat- és nyomásszabályozás



4.8.8. ábra - UI Távvezérelt üzemmód - Percenként belégzési idő és Be- illetve kilégzési arány szabályozás

A 4.8.8. ábra szemlélteti a megtervezett második részt: miután lenyomódott a *Breaths/min and Cycle Ratio* nyomógomb a felhasználónak lehetősége van a be- és kilégzési arány illetve a percenként belélegzett idő szabályozására.



4.8.9. ábra - UI Távvezérelt üzemmód - Szenzorok

Végül a harmadik részben lévő 4.8.8. ábra összefoglalja a szenzorokat és azok megjelenő értékeit. A látványosabb kinézet érdekében ezek külön ikonokkal lettek ellátva.

Annak érdekében, hogy a felhasználót segíteni lehessen egy olyan lehetőséget nyújt a UI, hogy elolvashat egy rövid leírást, amelyben megtudja, hogyan használhatja az oldalt távvezérlésre. A 4.8.8. ábrán található SWITCH nyomógomb mellett látható egy kérdőjel, amelyre kattintva, az utóbbi két elem alatt megjelenik egy doboz amelyben egy kisebb leírást olvas. A doboz eltüntethető a kérdőjel gomb újbóli lenyomása esetén.

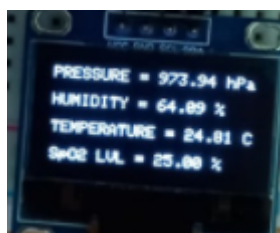
5. Üzembe helyezés és kísérlet eredmények

Egy klinikában felhasznált gép esetében elengedhetetlen olyan tesztek és kísérletek elvégzése, amelyeknek köszönhetően olyan hibákat lehet időben kiküszöbölni, amik a rendszer minőségét és megbízhatóságát növeli. Elsősorban azokat a komponenseket kellett tesztelni, amelyek azért felelnek, hogy egy beteg számára valóban egy olyan légzést lehessen biztosítani, amelyek megfelelnek azoknak a kiszámított paramétereknek, amelyek eddig a dolgozat keretein belül bemutatásra kerültek.

Elsőként megemlíthető az OLED kijelzőn lévő értékek elhelyezése és pontosságára fektetett hangsúly, ugyanis itt sikerült ellenőrizni, hogy a weboldalon található értékek valóban megfelelnek a kijelzőn látottakkal. Manuálisan lehetett tesztelni, hogy ha áram alá kerül a gép akkor elsőként egy üdvözlő szöveg jelenik meg (5.1. ábra), majd a szenzorok értékei (5.2. ábra). Az egyik botkormány kétszeri lenyomásával elindítható a motor és az OLED kijelzőn a paraméterek értékei jelennek meg (5.3. ábra). Újbóli lenyomás esetén a szenzorok értékeire tér vissza.



5.1. ábra - OLED indítás

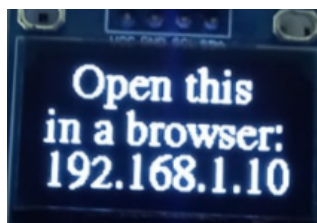


5.2. ábra - OLED szenzorok



5.3. ábra - OLED paraméterek

Bármilyen adat szerepeljen az OLED kijelzőn, ha a felhasználó a második botkormányt tartja lenyomva, pár másodpercig megjelenik az IP cím, amelyet a felhasználónak a böngészőbe kell írnia.



5.4. ábra - OLED IP cím sugó

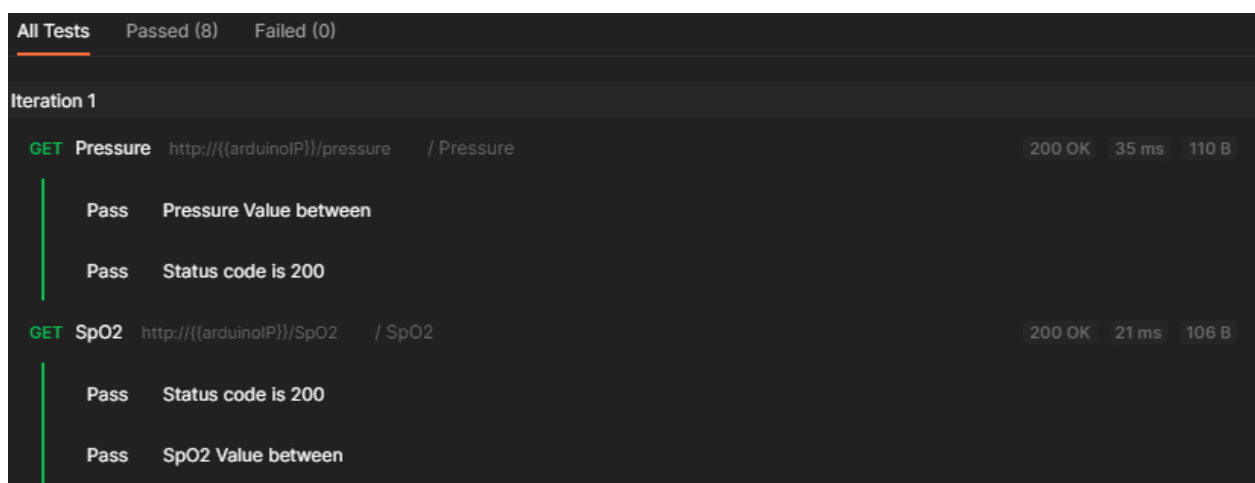
5.1. Felmerült problémák és megoldásaik

Az egyik felmerült probléma abból állt, hogy a motor helyből indulás során megszorult. Ez azért történt, mert pontatlan volt a 3D nyomtatás, s ezért a dugattyú a kelleténél szorosabban forgott. Erre a szoftveres megoldás az volt, hogy a motor indulását ki kellett mindkét irányban segíteni, úgy, hogy szélesebb impulzusokat kellett adni, majd ezt addig csökkenteni amíg nem tudott egy konstans ütemet felvenni. Emellett megoldásként szolgált még nagyobb méretű henger/dugattyú használata.

Egy másik felmerült probléma abból állt, hogy a programkódba bele kellett írni a jelszót és az SSID-t ahhoz, hogy csatlakozni lehessen az internethez. Ennek az a hátránya, hogy minden helyváltoztatáskor újra kellene futtatni az aktuális új adatokkal a kódot, emiatt olyan megoldásra volt szükség, amelyik felhasználja a esp32 azon tulajdonságát, hogy képes hozzáférési pontként funkcionálni. Ehhez a már említett SoftIP bevezetése jelentett segítséget. Az egyik hátránya a módszernek, hogy nem rendelkezik csak WPA2 szintű titkosítással, ami a jelenleg elterjedt WPA3 későbbi változata. Viszont így megoldást nyújtott arra, hogy bárhol legyen a mikrovezérlő rácsatlakozva, a közeléből szabályozni lehet a paramétereket.

5.2. Szoftvertesztek

A weboldallal kapcsolatos teszteket a harmadik fejezetben tárgyalt Postman tesztelési keretrendszer került felhasználása. Itt elsősorban olyan tesztek íródtak, amelyek azt vizsgálták, hogy a végpontok valóban léteznek és a hozzájuk tartozó GET és POST kérésekre elfogadható választ adnak.



5.1.1. ábra - Szenzorok GET tesztjeinek eredménye

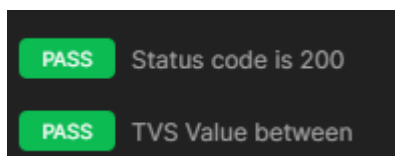
Elsőként a szenzorok végpontjai voltak megvizsgálva illetve, hogy a válaszárték beletartozik-e a megadott intervallumba. Az 5.1.1. ábrán látható a nyomás és az SpO2 végpontjaira futtatott tesztek helyes eredménye illetve, mindkét kapott érték az előre megadott intervallumban található.

Ezt követően a paraméterek GET végpontjai voltak tesztelve, azzal a függvénnyel ami ellenőrzi, hogy a kapott válasz benne van az adott két határérték között (5.1.2. ábra 250 és 750 értékek).

```
pm.test("TVS Value between", function () {  
  pm.expect(parseInt(pm.response.text())).to.above(250);  
  pm.expect(parseInt(pm.response.text())).to.below(750);  
});
```

5.1.2. ábra - TVS GET teszt függvénye

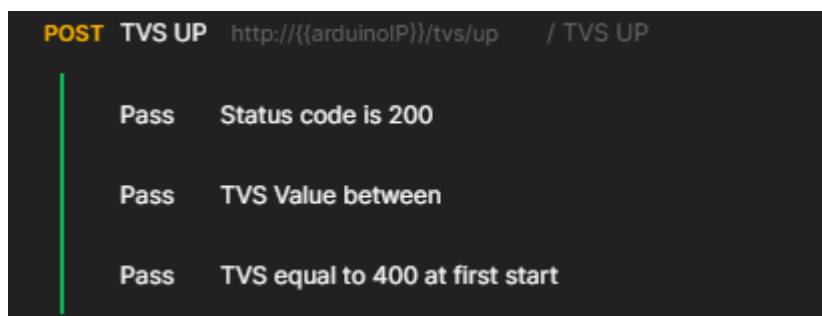
Válaszul a konzol értéke az 5.1.3. ábrán látható. A zöld háttérű PASS szöveg jelenti sikerességét.



The screenshot shows two lines of test results on a dark background. Each line starts with a green button labeled 'PASS'. The first line continues with 'Status code is 200' and the second line with 'TVS Value between'.

5.1.3. ábra - TVS GET Teszt eredményei

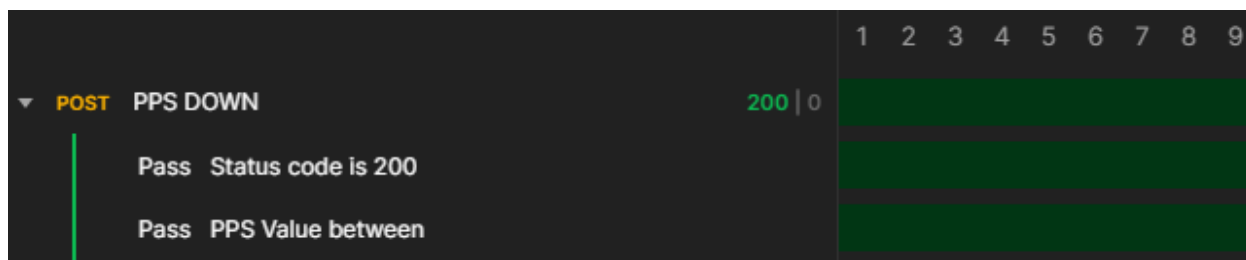
Ezt követően a POST kérések tesztelése következtek, első sorban azt kellett vizsgálni, hogy kezdetlegesen az érték megegyezik-e az alapértelmezett értékkel, utána azt hogy a növelt érték még mindig határon belül található. Erre példa az 5.1.4. ábra.



The screenshot shows the header and results of a POST test. The header is 'POST TVS UP http://{{arduinoIP}}/tvs/up / TVS UP'. Below it, there are three lines of results, each starting with a green vertical bar and the word 'Pass'. The results are: 'Status code is 200', 'TVS Value between', and 'TVS equal to 400 at first start'.

5.1.4. ábra - TVS UP POST teszt eredménye

Végül a szerver leterhelése következett, 100 iteráción keresztül futtatta a két meghatározott tesztet és mindig figyelte, hogy tényleg nem lépett ki a meghatározott értékek közül. Az 5.1.5. ábra jobb oldalán levő számok jelentik az iterációkat, a 200 jelöli, hogy 100*2 tesztet futott le.



5.1.5. ábra - TVS UP 100 iteráció utáni POST test eredménye

5.3. Manuális tesztek

Mialatt a vezérlő funkciói implementálásra kerültek minden új függvény felhasználása esetében le kellett ellenőrizni, hogy a felhasználói felület és az OLED kijelzője is ugyanazokat az értékeket mutatja. Illetve azt is ellenőrizni kellett, hogy el lehet-e érni azt, hogy a szerver leálljon (ne tudjon fogadni beérkező kéréseket). Ez a már említett TDD teszteléshez hasonlított.

Ilyen tesztre példa minden értéknövelő nyomógomb 5 alkalommal történő lenyomása a felhasználói felületen és ellenőrizni, hogy nemcsak az OLED számára, hanem a felhasználói felületen is frissültek az értékek. Az eredmények az 5.3.1 és 5.3.2. ábrán láthatóak. Az értékek megegyeznek ezért kijelenthető, hogy a rendszer helyesen jeleníti meg a számolt értékeket, mindkét felületen.



5.3.1 ábra - OLED eredmény

Current Values	
Tidal Volume:	Plateau Pressure:
450	45
Breaths/Minute:	Cycle Ratio:
25	1::3.00

5.3.2 ábra - UI eredmény

Más manuális teszt alkalmával az volt vizsgálva, hogy milyen időbeli eltérések vannak a manuális és a távvezérelt irányítás között. A távvezérléshez Postman felhasználásával volt válaszidő mérve és össze volt adva a kód azon részével, amelyik az új paraméterek kiszámításáért felel. Itt befolyásoló oknak számított a vezeték nélküli kapcsolat erőssége. Manuális üzemmódnál a programkód gombnyomást figyelő részen *millisec()* függvény indításával volt lemérve. A cél az volt, hogy egy paraméter módosítás után, mennyi időnek kell

eltelnie, hogy az új érték kiszámítódjon. Az eredmény az 5.3.3. táblázatban látható. Távvezérelt üzemmódban átlagosan a manuális felénél kevesebb idő alatt lehet eljutni az új érték kiszámításához. Ennek főként az az oka, hogy a fő függvény (loop) csak bizonyos időközönként fut le, ráadásul a baud ráta (jelátviteli sebesség) értéket le kellett csökkenteni az eddigi 115200-ról 9600-ra, mert a szenzorok kiiratásánál olyan gyorsan érkezett az információ, hogy nem volt idő a konkrét értékek leolvasására a folyamatos változtatások miatt. Emellett manuális üzemmódnál még problémát jelentett az, hogy a gombok minden alkalommal vissza kell álljanak középpállásba és csak azt követően lehet újból paramétereket változtatni.

Remote Controlled (ms)	Manual (ms)
201	491
151	379
196	523
190	520
206	380
212	575
183	608
205	463
165	570
175	396

5.3.3. táblázat - Manuális és távvezérelt üzemmódok időbeli összehasonlítása

5.4. A megvalósult lélegeztetőgép

A mechanikai fejlesztésben résztvevő kolléga által elkészített teljes mechanizmus robbantott rajza az III. számú mellékletben található, ahol szerepel a két forgódugattyú, a körülvevő dugattyúház, a közös tengely, emellett két kerék páros és a hajtótengely.

A mechanikai részt és a vezérlőt összekötő elem a léptetőmotor, ezért lehetett egymástól függetlenül fejleszteni, viszont a mechanikai komponensek kialakításánál fontos volt figyelembe venni, hogy a vezérlő fizikai komponensei elférjenek és a külvilággal kommunikáló elemek (tológombok, maga az OLED) számára pontos méretű kivágások legyenek a hátlap oldalán.

Az elkészült lélegeztetőgép megtekinthető az 5.3.1. ábrán. A mechanizmus alkatrészei 3D nyomtatás útján, illetve CNC gép felhasználásával készültek.



5.3.1. ábra - A lélegeztetőgép funkcionális modellje a dugattyú felőli oldalról nézve

5.5. Mérési eredmények

A funkcionális lélegeztetőgépet felhasználva, olyan mérésekre volt szükség, amellyel igazolni lehetett azt, hogy az értékek valóban megfelelnek az RMVS001 előírásainak. A mérési eredményeket a II. melléklet táblázataiban lehet megtekinteni. Minden mérés során három rögzített érték volt meghatározva, és ezen három beállított érték közül egy volt megmérve a 3.3.4. ábra szerint. A rögzített értékek szürke, míg a mért értékek zöld színnel vannak jelölve a táblázatokban. Az első táblázatban a percenkénti belégzések számához öt többszöröse voltak felhasználva, illetve a térfogat számára ötvenesével voltak az értékek beállítva azért, mert távvezérelten gyorsabban lehet a paramétereket állítani, ahogy az 5.3.3. táblázat is bizonyítja. A másik két táblázatban az értékek manuális vezérlés útján voltak beállítva.

6. Következtetések

Az elkészült lélegeztetőgép vezérlője teljesíti a megfogalmazott célokat, és alternatívákat kínál a távvezérléses klinikai lélegeztetőgépek területén. A kivitelezés során alkalmazva voltak az elvégzett eredményei, melyek alapján számos továbbfejlesztési lehetőséget lehet megfogalmazni.

6.1. Megvalósítások

A felhasználói követelményben megfogalmazott megszorításokból sikerült a lehető legtöbbet kivitelezni. Így sikerült egy olyan lélegeztetőgépet megvalósítani amelynél lehetőség van nemcsak kézi, hanem távvezérelt szabályozásra is. A szabályozáshoz szükséges paraméterek

megfelelnek a RMVS001 előírásainak, tehát egy hivatalosan elismert és elfogadott leírást kellett szem előtt tartani a fejlesztés során. Habár sikerült három szempont szerint szabályozni, (ki/belégzési arány, percenkénti belégzések száma és belégzési térfogat) mégis kellő eszközök hiányában a plató nyomás szabályozása továbbfejlesztési lehetőség maradt. Ennek ellenére a programkódban, úgy van implementálva, mintha eddig is részt vett volna a motor működésében. Ezért amire szükség lesz az egy nyomás mérő szenzor, amely jóval nagyobb nyomást képes lemérni, mint az eddig felhasznált barometrikus nyomás mérő.

A felhasználói platform kinézetében és funkcionalitásának egyszerűségében könnyen használható nemcsak orvosok, de olyan emberek számára is akik otthon szeretnék lélegeztetést alkalmazni. A kísérletekből kiderült, hogy valóban az elvárt működést és megközelítőleg helyes eredményeket szolgáltatja.

A felhasznált alkatrészek többnyire az olcsó kategóriába sorolhatóak, hiszen az volt végig a cél, hogy könnyen megfizethető legyen bárki számára.

A programkód majdnem összes részét sikerült a verziók során modulokra bontani, ezáltal elősegítve a továbbfejlesztési lehetőségek implementálását, illetve növelve a kód átláthatóságát egyaránt.

6.2. Hasonló rendszerekkel való összehasonlítás

A [20] hivatkozásban található tudományos cikkben a lélegeztetőgép vezérléséhez egy PIC18F4550 típusú mikrovezérlőt használtak.

Előnye, hogy több szenzor értékeit is nyomon lehet követni, illetve mesterséges intelligenciát alkalmaztak a megfelelő vezérlés előállításához, ezért jobban alkalmazkodik a beteg lélegeztetési szükségleteihez.

Hátránya pedig abban áll, hogy nem felel meg az RMVS001 követelményeinek, így nem lehet klinikailag értékesíteni, illetve nem lehet nyomon követni a szenzorok változását sem egy felhasználói felületen. Nincs minimális védelem hozzárendelve a mikrovezérlős irányításhoz.

Másik rendszer, amely közelebb áll az itt bemutatott lélegeztetőgéphez a [21] cikk mutatja be. Előnye, hogy a mikrovezérlő, hogy, ha a légzés nem felel meg az elvártnak, egy olyan riasztást vált ki amelyik egy SOS üzenetet küld az érintett orvos telefonjára. Ezáltal gyorsabban tudja riasztani a személyzetet, ha probléma merülne fel. A rendszer emellett tartalmaz a beteget közvetlen mérő szenzorokat is. Hátránya, hogy nem lehet nyomást szabályozni, illetve ugyancsak nem felel meg az RMVS001 követelményeinek.

6.3. Továbbfejlesztési lehetőségek

6.3.1. Hardver fejlesztési lehetőségek:

- a grafikus kijelző méretének megnövelésével sokkal átláthatóbb lenne a felhasználói felület, emellett a felhasználók számára esetleges hibaüzeneteket is ki lehet írni, annak érdekében, hogy segítse a rendszer állapotáról.
- vezeték nélküli helyi hálózatok bővítése GPRS³⁵ hálózatra.
- bővíteni a memóriának a kapacitását azért, hogy olyan tanuló algoritmusokat lehessen alkalmazni amelyek szenzorok mért értékei alapján előre meghatároznak egy lélegeztetési formát a beteg számára
- több szenzor beépítése a rendszerbe, ezáltal egyéb értékeket is megjeleníteni (például SpO2)

6.3.2. Szoftver fejlesztési lehetőségek:

- bejelentkezési procedúrát létrehozni, azért, hogy a beteg lélegeztetését csak az a személy tudja nyomon követni, aki a kijelölt orvosa. A jelszavakat hash-ként eltárolni, ezáltal egy bizonyos szintű feltörés elleni biztonságot nyújtva. [22].
- hasznos volna egy adatbázis kapcsolatot létrehozni, hogy egy szakorvos naplózni tudja betegeinek a lélegeztetési változtatásait, a szenzorok és mérőműszerek értékeit, illetve egyszerre nyomon tudjon követni több beteget is [23].
- a beteg egyéb állapotának adatait is figyelembe véve előállítani, kiszámítani egy ajánlott lélegzés szabályozást, vagy ha éppen komplikációk lépnek fel, akkor ennek megfelelően módosítsa a belélegzett levegő térfogatát, ki és belégzési arányt, vagy épp más paramétereket [24].
- a felhasználói platformon a szenzorok mellé két mezőt helyezni, amelyben meg lehet adni, hogy milyen értékek túllépése után jelezze azt az orvos számára. Hibaüzenetet és esetleges végrehajtási javaslatot is csatolni lehet az üzenet után
- a felhasználói platform átírása egy modern webfejlesztő környezetbe (Angular³⁶)

³⁵ GPRS - csomagkapcsolt IP-alapú technológia

³⁶ Angular - <https://angular.io>

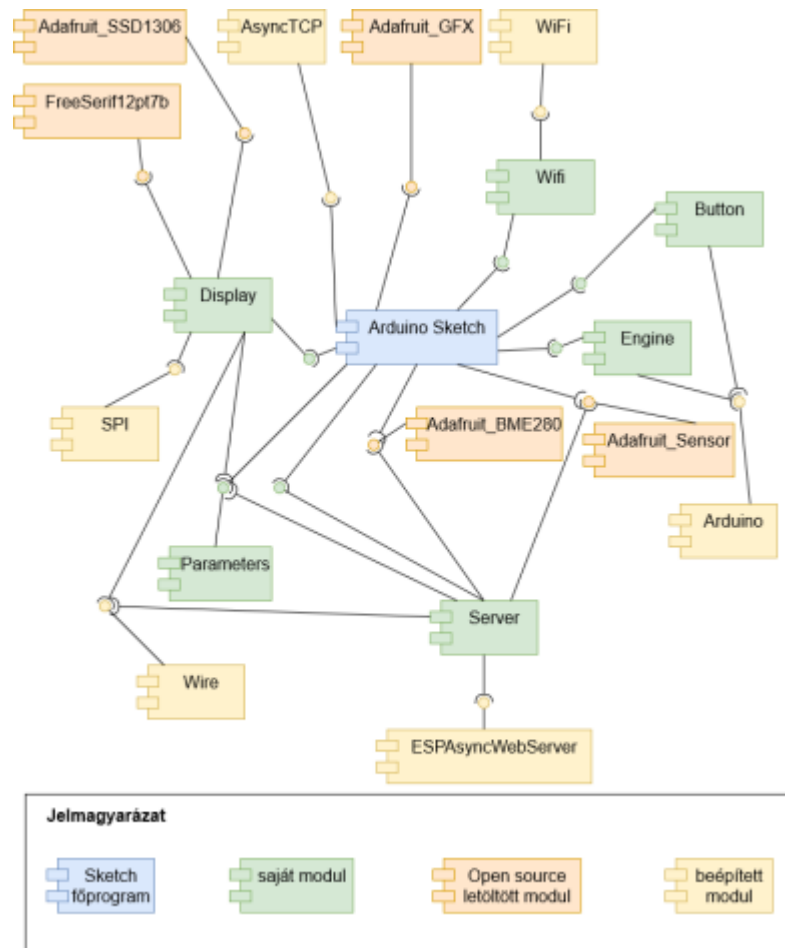
7. Irodalomjegyzék

- [1] Tulassay, Zs., 2011. A belgyógyászat alapjai, Medicina Könyvkiadó Zrt., Budapest
- [2] Tobin, M.J., 2006. Principles and Practice of Mechanical Ventilation, Ed. McGraw-Hill, New York
- [3] Péntes, I., Lörx, A.J., 2004. A lélegeztetés elmélete és gyakorlata, Medicina Kiadó, Budapest
- [4] Wozniak R, D., Rubino A., Tan, A., Jones L, N., Webb, S., Vuylsteke, A., Palas, E., Quinnell, T., Smith, I, Davies, M., 2020. Positive role of continuous positive airway pressure for intensive care unit patients with severe hypoxaemic respiratory failure due to COVID-19 pneumonia: A single centre experience. Cambridge
- [5] RMVS001 Rapidly Manufactured Ventilator System, Medicines & Healthcare Products Regulatory Agency, 2020 London
- [6] Palacka,P., Furka, S., Furka, D., Huzevka, T., Gallik, D., Janek, M., 2020. Q-VENT—A novel device for emergency ventilation of the lungs in patients with respiratory failure due to diseases such as COVID-19, Bratislava, [Online]:
<https://onlinelibrary.wiley.com/doi/10.1002/mds3.10124>
- [7] Hewing, L., Menner M., Tachatos, N., Daners, M., Pasquier, C., Lumpe, T., Shea, K., Carron, A., Zeilinger, M., 2020. Volume Control of Low-Cost Ventilator with Automatic Set-Point Adaptation, [Online]. Available: <https://arxiv.org/pdf/2009.01530.pdf>
- [8] Espressif Systems Copyright, 2019. ESP32 Series Datasheet, Espressif Inc., China
- [9] Toshiba Corporation, 2014. TB67S109AFTG Datasheet CLOCK-in controlled Bipolar Stepping Motor Driver, [Online]. Available:
<https://datasheet4u.com/datasheet-pdf/Toshiba/TB67S109AFTG/pdf.php?id=1387707>
- [10] Nanotech, Closed Loop Technologie, [Online].
Available:https://en.nanotec.com/fileadmin/files/Application_Notes/Closed_Loop/2012_06_21_nanotec_whitepaper_ClosedLoop_en.pdf
- [11] Leadshine Technology Co., Ltd, 57HS Series Hybrid Stepping Motors, [Online].
Available: <http://www.leadshine.com/UploadFile/Down/57HSxxd.pdf>
- [12] Solomon Systech, 2008. 128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller, [Online]. Available:
<https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>

-
- [13] ON Semiconductor, A Low-Dropout Positive Fixed and Adjustable Voltage Regulators, [Online]. Available: <https://pdf1.alldatasheet.com/datasheet-pdf/view/174810/ONSEMI/NCP1117.html>
- [14] Bosch, Humidity sensor measuring relative humidity, barometric pressure and ambient temperature, [Online]. Available: <https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/>
- [15] Mouser Electronics, Joystick Module, [Online]. Available: <https://components101.com/modules/joystick-module>
- [16] Santos R., Santos, S., 2019. Learn ESP32 with Arduino IDE, Randomnerdtutorials, Porto
- [17] Massimo, B., Shiloh, M., 2015. Getting Started with Arduino 3rd edition, USA
- [18] Baumer, MTA5 - MTX5 Industrial Capsule Pressure Gauges, [Online]. Available: https://www.elfadistelec.pl/Web/Downloads/_e/ng/mta5-mtx5_tds_eng.pdf
- [19] Kobold, Modular, compact Inline Flowmeter, [Online]. Available: <https://www.kobold.com/Modular-compact-inline-Flowmeter-KME>
- [20] Adem, G., Hakan, I., Inan, G., 2016. Design and Construction of a Microcontroller-Based Ventilator Synchronized with Pulse Oximeter
- [21] Sumanta, B., Sriram, K., India, 2012. Real-Time Breath Rate Monitor based Health Security System using Non-invasive Biosensor
- [22] Folláth, J., Huszti, A., Pethő, A., 2011. Informatikai biztonság és kriptográfia, Kempelen Hallgatói Információs Központ, Budapest
- [23] Pétery, K., 2015. Adatbázisok létrehozása, Mercator Stúdió Kiadó, Budapest
- [24] Russell, S., Norvig, P., 2015. Mesterséges intelligencia modern megközelítésben, II.

8. Függetlékek

I. Melléklet - Komponens diagram



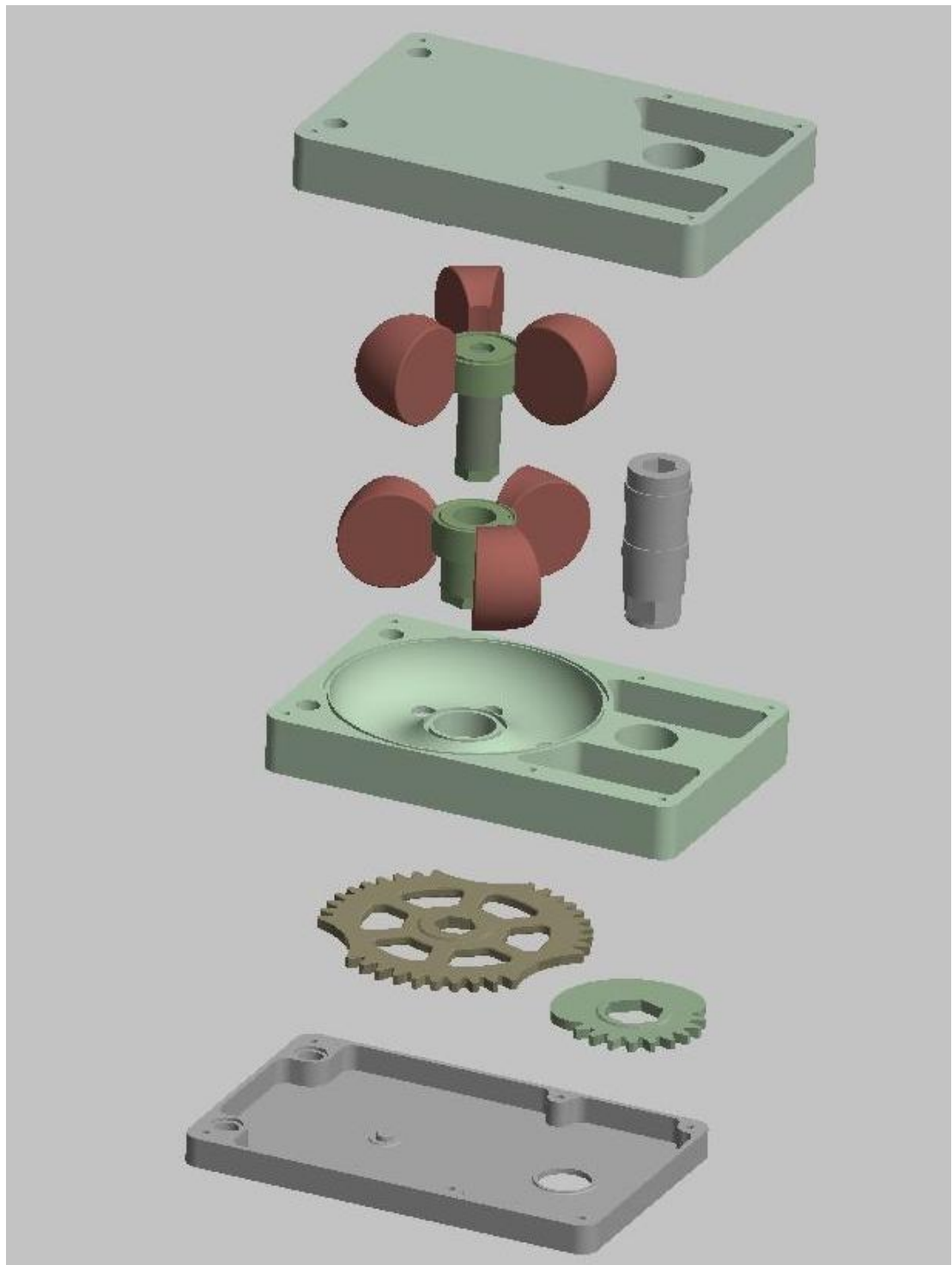
II. Melléklet - Elvégzett mérések eredményei

Test No.	Insp./Exp. Ratio		Respiratory Rate		Tidal Volume		Plateau Pressure	
	[I:E]		[b/m]		[ml]		[mB]	
	Prescribed	Measured	Prescribed	Measured	Prescribed	Measured	Prescribed	Measured
1	1:1		10		400	403		
2	1:2		15		400	402		
3	1:3		15		400	403		
4	1:2		20		350	349		
5	1:2		25		350	349		
6	1:3		20		400	400		
7	1:4		15		300	298		
8	1:4		20		300	299		
9	1:4		25		300	299		
10	1:1		20				30	29,7
11	1:2		25				25	25,8
12	1:3		20				25	25,1
13	1:2		25				20	20,6
14	1:2		30				25	25,8
15	1:3		20				25	25,2
16	1:4		15				20	20,3
17	1:4		20				25	25,2
18	1:4		25				30	30,3
19	1:2		25	25,07	300			
20	1:2		30	29,95	350			
21	1:3		25	25,02	300			
22	1:3		30	29,99	350			
23	1:4		25	25,03	300			
24	1:4		30	30,01	350			
25	1:2	1:2,03	25		300			
26	1:3	1:3,02	25		300			
27	1:3	1:3,01	25		350			
28	1:4	1:3,99	25		300			
29	1:4	1:3,99	25		350			
30	1:2	1:2,02	25				20	
31	1:3	1:3,01	25				20	
32	1:3	1:3,02	25				25	
33	1:4	1:3,99	25				20	
34	1:4	1:3,99	25				25	
35	1:1		30		600	599		
36	1:2		30		600	596		
37	1:3		30		600	598		
38	1:2		30				35	34,61
39	1:4		30				35	34,71
40	1:4		30				35	35,32

Test No.	Insp./Exp. Ratio		Respiratory Rate		Tidal Volume		Plateau Pressure	
	[i:E]		[b/m]		[ml]		[mB]	
	Prescribed	Measured	Prescribed	Measured	Prescribed	Measured	Prescribed	Measured
41	1:2		10		490	492		
42	1:3		11		500	499		
43	1:3		12		510	501		
44	1:4		12		550	453		
45	1:2		12		570	559		
46	1:3		15		600	593		
47	1:3		16		620	628		
48	1:4		16		630	637		
49	1:4		20		630	631		
50	1:2		30	29,91	300			
51	1:3		25	25,03	310			
52	1:3		25	25,16	320			
53	1:3		24	23,91	340			
54	1:1		23	23,21	350			
55	1:4		24	24,71	370			
56	1:1		20				35	35,68
57	1:2		27				20	20,92
58	1:3		28				25	25,47
59	1:2		27				20	20,04
60	1:2		26				25	24,98
61	1:3		26				25	25,08
62	1:2		25				20	20,13
63	1:2		24				25	24,86
64	1:4		23				35	35,02
65	1:2	1:2	35		400			
66	1:3	1:2,95	25		320			
67	1:3	1:3,11	25		330			
68	1:4	1:4	25		340			
69	1:4	1:4,02	25		360			
70	1:2	1:1,92	25				20	
71	1:3	1:3,13	25				25	
72	1:3	1:3,03	25				30	
73	1:4	1:4,08	25				35	
74	1:4	1:3,91	25				40	

Test No.	Insp./Exp. Ratio		Respiratory Rate		Tidal Volume		Plateau Pressure	
	[i:E]		[b/m]		[ml]		[mB]	
	Prescribed	Measured	Prescribed	Measured	Prescribed	Measured	Prescribed	Measured
75	1:1		24				30	29,99
76	1:2		23				25	25,05
77	1:2		24				25	25,08
78	1:2		26				30	30,13
79	1:2		27				25	25
80	1:1		28				25	25,44
81	1:1		29				20	19,92
82	1:4		29				25	25,01
83	1:2		28				30	30,07
84	1:1		10		400	398		
85	1:2		11		500	501		
86	1:3		12		600	603		
87	1:4		20		610	614		
88	1:2		21		620	630		
89	1:3		22		630	621		
90	1:4		22		640	639		
91	1:3		23		640	629		
92	1:3		24		700	711		
93	1:2		27	27,89	600			
94	1:2		26	25,98	610			
95	1:4		26	26,04	620			
96	1:3		25	25,01	630			
97	1:4		24	24,26	640			
98	1:4		23	23,02	650			
99	1:3	1:2,02	22		660			
100	1:3	1:3,04	22		670			
101	1:3	1:2,98	21		680			
102	1:4	1:4,05	20		690			
103	1:4	1:4,12	17		700			
104	1:2	1:1,98	16				15	
105	1:3	1:3,09	15				20	
106	1:3	1:3,1	14				25	
107	1:4	1:4	14				30	
108	1:3	1:3,02	13				35	

III. Melléklet - A lélegeztetőgép robbantott rajza



UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Vizat decan
Conf. dr. ing. Domokos József

Vizat director departament
Ș.l. dr. ing Szabó László Zsolt