

TDK-dolgozat

Kazup László

Marcsák Gábor Zoltán



MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR
Elektrotechnikai-Elektronikai Tanszék



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

Sorszámosztó rendszer fejlesztése háziorvosi rendelők részére

Kazup László

II. éves mechatronikai mérnök
MSc szakos hallgató

Marcásák Gábor Zoltán

I. éves logisztikai mérnök
MSc szakos hallgató

Konzulens:

Váradiné Dr. Szarka Angéla

egyetemi docens
Elektrotechnikai- Elektronikai tanszék

Miskolc, 2012

Köszönetnyilvánítás

A tanulmány/kutató munka a TÁMOP-4.2.1.B-10/2/KONV-2010-0001 jelű projekt részeként – az Új Magyarország Fejlesztési Terv keretében – az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Tartalomjegyzék

1	BEVEZETÉS	1
2	A SORSZÁMOSZTÓ RENDSZER HARDVERES FELÉPÍTÉSE	2
2.1	A váróteremben lévő hardverkomponensek	2
2.1.1	Sorszámosztó terminál	2
2.1.1.1	Nyomógomb	3
2.1.1.2	Hőpapíros nyomtató	3
2.1.1.3	Vázszerkezet, külső borítás	4
2.1.2	LED Kijelző	4
2.1.2.1	Működés, felépítés	6
2.1.2.2	A LED-ek típusai	6
2.1.2.2.1	Hagyományos LED-ek	6
2.1.2.2.2	Nagy fényerejű LED-ek	7
2.1.2.2.3	Fehér LED, teljesítmény LED-ek	7
2.1.2.3	LED kijelzők felépítése	8
2.1.2.3.1	Numerikus/alfanumerikus szegmenses LED kijelzők	8
2.1.2.3.2	Pont-mátrix LED kijelzők	10
2.1.2.4	LED-kijelzők üzemmódjai	11
2.1.2.4.1	Folytonos üzem	11
2.1.2.4.2	Multiplex-üzem	11
2.1.2.4.3	Moduláris felépítésű pont-mátrix kijelzők	12
2.1.2.5	Az alkalmazott pont-mátrix LED kijelző	12
2.1.2.5.1	A LED-kijelző főbb adatai	13
2.1.3	Hangszóró	13
2.2	A fogadóhelységben lévő hardverkomponensek	13
2.2.1	Személyi számítógép	14
2.2.2	USB- RS232 Hub	14
2.2.3	Behívó gomb	15
3	A SORSZÁMOSZTÓ RENDSZER SZOFTVERES FELÉPÍTÉSE	16
3.1	JavaFX vezérlőprogram	16
3.1.1	JavaFX Fejlesztői környezet	17

3.1.2	Verziókövető rendszer	17
3.2	Adatbázis	18
3.2.1	Felhasználók	18
3.2.2	Páciensek jegyzéke	18
3.2.3	Kiosztott sorszámok és előfoglalások	19
3.3	Beszédszintetizátor	19
3.4	A rendszer telepítése	19
4	A JAVA FX VEZÉRLŐPROGRAM MŰKÖDÉSE	20
4.1	A program indítása	20
4.2	Többszálú architektúra	22
4.3	Hardverelemek vezérlése	23
4.3.1	Nyomógombok	23
4.3.2	Hőpapíros nyomtató	24
4.3.3	LED kijelző	26
4.3.3.1	JNA, JNI	27
4.3.3.2	.DLL függvénykönyvtár használata Java környezetben	27
4.3.3.3	Karaktertömb konvertálása képpé	28
4.3.3.4	Megjeleníthető adatok	29
4.4	Az adatbázis kezelése	29
4.5	Időpont előfoglalás	30
4.5.1	Páciensek jegyzéke	30
4.5.2	Sorszámok és időpont előfoglalások jegyzéke	32
4.5.2.1	Ütemezett időpont	32
4.5.2.2	Státusz	32
4.6	Beszédszintetizátor	33
4.7	Főmenü	34
4.8	Beállítások képernyő	34

5	A RENDSZER TESZTELÉSE	35
6	MIÉRT JOBB A MI RENDSZERÜNK A JELENLEG FORGALOMBAN LÉVŐKNÉL?	35
7	ÖSSZEFOGLALÁS	36
8	IRODALOMJEGYZÉK	37

Ábrajegyzék

1. ÁBRA: HARDVERKOMPONENSEK A VÁRÓTEREMBEN	2
2. ÁBRA: RS-232 NYOMÓGOMB	3
3. ÁBRA: SAM4S ELLIX 10 HŐPAPÍROS NYOMTATÓ	4
4. ÁBRA: „FLIP DISC” KIJELZŐ	5
5. ÁBRA: HÉTSZEGMENSES LED KIJELZŐ FELÉPÍTÉSE, BEKÖTÉS	9
6. ÁBRA: KÉT DIGITES HÉTSZEGMENSES LED KIJELZŐ	9
7. ÁBRA: TIZENHAT SZEGMENSES EGYKARAKTERES LED KIJELZŐ	9
8. ÁBRA: KÜLÖNFÉLE KARAKTEREK MEGJELENÉSE TIZENHAT SZEGMENSES KIJELZŐN	10
9. ÁBRA: 8X8 PIXEL MÉRETŰ PONT-MÁTRIX KIJELZŐ	10
10. ÁBRA: HÉTSZEGMENSES LED KIJELZŐ FOLYTONOS ÜZEMBEN	11
11. ÁBRA: PÉLDA PONT-MÁTRIX KIJELZŐ MULTIPLEX ÜZEMÉRE	12
12. ÁBRA: SHENZEN YUHONG LCD-KIJELZŐ	12
13. ÁBRA: A LED-KIJELZŐ FŐBB ADATAI	13
14. ÁBRA: HARDVERKOMPONENSEK A FOGADÓHELYSÉGBEN	14
15. ÁBRA: AZ ÁLTALUNK HASZNÁLT USB- RS232 HUB	15
16. ÁBRA: JAVAFX SCENE BUILDER	17
17. ÁBRA: A JAVAFX VEZÉRLŐPROGRAM KEZDŐKÉPERNYŐJE	21
18. ÁBRA: VEZÉRLŐK AZ IDŐ KIJELZÉSÉRE	21
19. ÁBRA: A NAPTÁR VEZÉRLŐ	22
20. ÁBRA: DATACARRIERDETECT (DCD) LÁB A SOROS PORTON	24
21. ÁBRA: KINYOMTATOTT SORSZÁM ÉKEZETES KARAKTEREKKEK	26
22. ÁBRA: SORSZÁM KÜLDÉSE A KIJELZŐRE	29
23. ÁBRA: RÖVIDEBB NÉV KÜLDÉSE A KIJELZŐRE	29
24. ÁBRA: HOSSZABB NÉV KÜLDÉSE A KIJELZŐRE	29
25. ÁBRA: A PÁCIENSEK JEGYZÉKE, IDŐPONTFOGLALÁS	30
26. ÁBRA: FIGYELMEZTETÉS ÉRVÉNYTELEN SZÜLETÉSI IDŐ ESETÉN	31
27. ÁBRA: FELUGRÓ MENÜ A PÁCIENSEK LISTÁJÁN	31
28. ÁBRA: IDŐPONTFOGLALÁS A PÁCIENS SZÁMÁRA	31
29. ÁBRA: AZ ADOTT RENDELÉS IDEJÉRE FOGLALT IDŐPONTOK ÉS SORSZÁMOK TÁBLÁZATA	32
30. ÁBRA: STÁTUSZ JELZÉSE A LEFOGLALT IDŐPONTOK ÉS SORSZÁMOK TÁBLÁZATÁN	33
31. ÁBRA: A PROGRAM FŐMENÜJE	34
32. ÁBRA: A PROGRAM BEÁLLÍTÁSOK KÉPERNYŐJE	34

1 Bevezetés

A várakozás modern világunk egyik leggyakoribb és sajnos elkerülhetetlen kényszerűsége. Várakozunk közlekedés közben, a piros lámpánál, vásárlás során a pénztárnál, az étteremben, a patikában és a bankban. Egyes becslések szerint egy átlagos ember napi 45-60 perc közötti időt tölt várakozással. A hivatali ügyek intézésekor, egészségügyi intézményekben gyakran ennél is hosszabb, több óras sorban állással kell számolni. A helyzetet tovább rontja, hogy gyakran azt sem lehet megbecsülni, előreláthatólag mennyi ideig kell még várakozni. Ez a bizonytalanság az, ami sokakból komoly stresszt, frusztrációt vált ki [1].

A társadalmi jelenséget tanulmányozva terveztünk meg a Miskolci Egyetem Elektrotechnikai- Elektronikai tanszékén egy minden tekintetben korszerű, komplett sorszámosztó rendszert, mellyel egyszerűen és hatásosan koordinálható az ügyfelek várakoztatása. A kapott sorszám egyértelműen jelzi az ügyfelek számára, hogy mennyien várakoznak előttük, így csökkentve a bizonytalanság érzetét. A rendszer a kiszolgálás sorrendjét is egyértelműen meghatározza, ezzel kiszűrve a koordinálatlan sorbanállás során gyakran megfigyelhető félreértéseket. A várakozási idő jelentős csökkentése érdekében az időpont előfoglalás is biztosított, mely funkció szerves részét képezi az általunk készített sorszámosztónak.

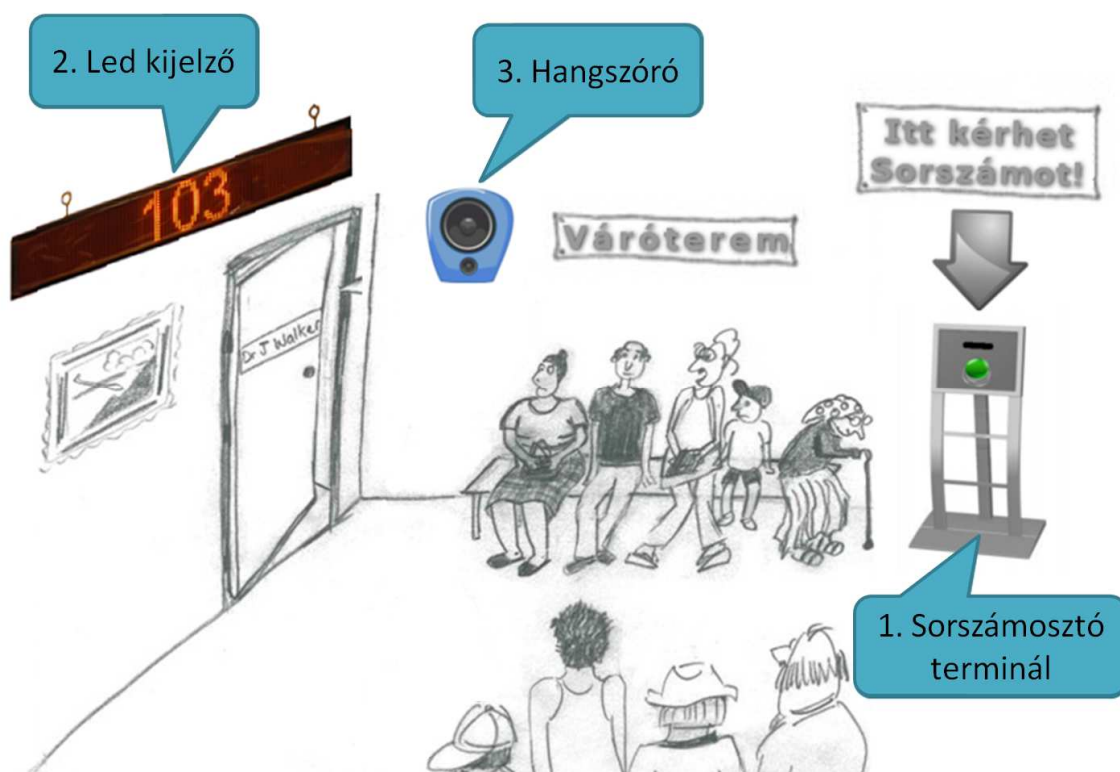
A rendszer tervezésekor nagyon fontos szempont volt, hogy a jelenleg kereskedelmi forgalomban lévő alternatíváknál jóval kedvezőbb áron értékesíthető, komplett megoldást hozzunk létre. A kedvező ár elérhetővé teszi megoldásunkat az alacsonyabb költségvetéssel gazdálkodó felhasználók számára. Külön figyelmet fordítottunk továbbá arra, hogy a sorszámosztó és ügyfélbehívó a lehető legnagyobb mértékben akadálymentesített legyen. A dolgozat következő részében a sorszámosztó rendszer architektúrája, szerkezeti sémája kerül bemutatásra. Utána külön részletezésre kerülnek a hardveres és szoftveres komponensek, valamint a fejlesztés lépései. A rendszert alapos laboratóriumi tesztelésnek vetettük alá, a dolgozat zárásaként a tesztelés során szerzett tapasztalatokat, illetve a kijavítandó hibákat és továbbfejlesztési lehetőségeket részletezzük.

2 A sorszámozító rendszer hardveres felépítése

A sorszámozító rendszer hardverkomponensei alapvetően két jól elkülöníthető csoportra bonthatók. Az egyik csoport azon részegységek együttese, melyek a váróteremben vannak elhelyezve. A másik csoport közvetlenül az asszisztens, vagy az orvos mellett található a fogadóhelységben.

2.1 A váróteremben lévő hardverkomponensek

Funkcionálisan két alapvető elvárás lehet megfogalmazni egy sorszámozító rendszerrel kapcsolatban. Mindenekelőtt a betegek könnyen és egyszerűen tudjanak sorszámot kérni. Ezután a páciensek behívása egyértelműen és jól észrevehető módon történjen meg. Ezen két ember-gép interakció sikeres végrehajtásához speciális eszközökre van szükség.



1. ábra: Hardverkomponensek a váróteremben

2.1.1 Sorszámozító terminál

A sorszámozító terminál feladata rendkívül egyszerű, és egyúttal kritikus fontosságú. Az érkező betegek számára kell sorszámot nyomtatnia. Mindenképpen a bejáráshoz közel,

jól észrevehető módon kell elhelyezni. Célszerű külön felhívni a figyelmet, hogy a rendelőben sorszámosztó rendszer működik. A terminál három kulcsalkatrészből épül fel.

2.1.1.1 Nyomógomb

A tervezés során fontos szempont volt, hogy a lehető legegyszerűbben lehessen sorszámhoz jutni. Napjainkban egyre inkább terjednek a bonyolult menürendszerrel rendelkező, érintőképernyős berendezések. Az általunk fejlesztett rendszer elsődleges célcsoportja a háziiorvosok. Gondolnunk kellett tehát az idősebb, károsult emberekre, és mindenképpen elkerülni a felhasználói felület indokolatlan túlbonyolítását. A rendszerünk esetében sorszám igényléséhez mindössze a terminálon található egyetlen, nagy méretű nyomógombot kell lenyomni. A nyomógomb RS-232 soros porton csatlakozik a rendszerhez.



2. ábra: RS-232 Nyomógomb

2.1.1.2 Hőpapíros nyomtató

A nyomógomb lenyomását a később bemutatásra kerülő központi vezérlő szoftver érzékeli. A szoftver kikeresi az adatbázisból a legközelebbi kiosztható sorszámot, majd elküldi a terminálba épített hőpapíros nyomtatónak. A hőpapíros nyomtatók egyik legelőnyösebb tulajdonsága, hogy nincs szükség tintapatronra működésükhöz. Helyette szó szerint a papírba égetik az információt, így karbantartásuk jóval egyszerűbb, üzemeltetésük pedig gazdaságos. Az egyetlen szükséges karbantartási feladat a papírtekercs cseréje. A kísérleti rendszerben használt Sam4s gyártmányú Ellix 10 típusú nyomtató további fontos

jellemzője, hogy rendelkezik egy úgynevezett „Guillotine” vágószerkezettel, melyet vezérelve automatikusan a megfelelő méretűre vágja a sorszám cetliket. A vágószerkezet élettartama 1.000.000 vágásra van hitelesítve. Napi 100 darab kiosztott sorszámmal számolva, és egész éves rendelést feltételezve is több, mint 27 éves élettartam várható el. A specifikáció szerint a nyomtatófej átlagosan 30.000.000 sor nyomtatását teszi lehetővé. A nyomtató a nyomógombhoz hasonlóan RS-232 soros porton csatlakozik a rendszerhez.



3. ábra: Sam4s Ellix 10 hőpapíros nyomtató

2.1.1.3 Vázszerkezet, külső borítás

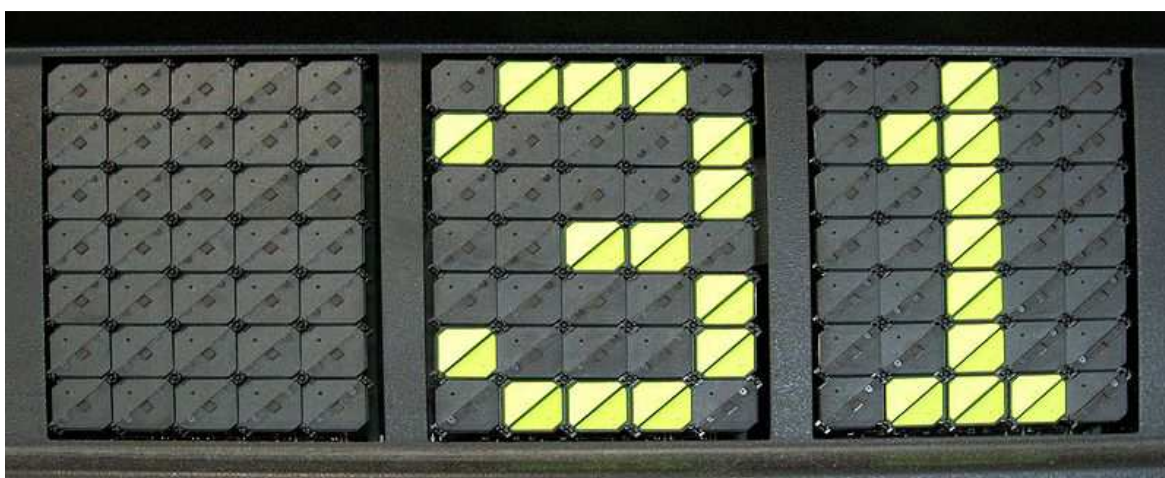
A sorszámosztó terminál vázszerkezetének tervezésekor különböző ergonómiai szempontokat kellett figyelembe vennünk. Fontos hogy a kezelőszervek a megfelelő magasságban legyenek elhelyezve. A szerkezetnek megfelelően stabilnak kell lennie a hőpapíros nyomtató és a nyomógomb elhelyezéséhez. Az eszközök esetleges illetéktelen eltulajdonítását elkerülendő a terminálon a környezethez való rögzítési pontokat kell kialakítani. A külső borítás kiválasztásánál a strapabíróság, az esztétikus megjelenés és a kedvező anyagköltség voltak a fő szempontok. A nyomtató eléréséhez, papírtekercs cseréjéhez a borítás megfelelő pontján zárható hozzáférési pontot kellett biztosítani.

2.1.2 LED Kijelző

A betegtájékoztató- és előjegyző rendszer kulcsfontosságú elemei az információk megjelenítését végző eszközök. Az alapkoncepció szerint szükség volt egy megfelelő

méretű optikai kijelzőre, mely a betegek számára fontos, folyamatosan frissülő információk megjelenítését végzi. Esetünkben egyik fő szempont volt a jól olvashatóság, mivel egyes esetekben a betegek csökkent látóképességgel rendelkezhetnek. A kifejezetten köztéri információ-megjelenítés céljára készült kijelzőknek két fő típusa létezik.

Az elektromechanikus kijelzők (elterjedt angol nevén „Flip Disc” Display), melyek hétszegmenses illetve pont-mátrixos kivitelben is készülnek, és manapság is használatosak pályaudvarokon, repülőtereken, valamint egyes közlekedési járműveken. Működési elvük a következő: minden egyes szegmens/ pixel egy egyik oldalon sötét, másik oldalán fehér vagy zöld színnel festett, szimmetriatengelye körül elforduló, tárcsa- vagy szögletes alakú, ferromágneses anyagú lemezből áll [2].



4. ábra: „Flip Disc” kijelző

A műanyag fészekben, melyben a pixel elhelyezkedik, két úgynevezett rögzítő mágnes van, mely sötét illetve világos pozícióban rögzíteni tudja a tárcsát. A fészek alatt egy elektromágnes található, melyre megfelelő irányú feszültséget kapcsolva egyik vagy másik pozícióba billenthető a lemez. Nagy előnye, hogy csak a képpont világos/sötét illetve sötét/világos váltásához igényel energiát, így fogyasztása rendkívül alacsony lehet. Hátránya azonban, hogy csak megfelelő fényviszonyok mellett olvasható, kiegészítő fényforrást igényel. Valamint az apró mechanikai szerkezetek porra és rázkódásra érzékenyek, ezért járműveken használt példányaiknál tipikus hiba a beragadt pixel jelensége.

A nyilvános kijelzők másik gyakori, és manapság egyre elterjedtebb típusa az úgynevezett LED-kijelzők, melyek nagy előnye, hogy nem igényelnek külső megvilágítást, és a félvezető technika elterjedésével manapság az áruk is egyre kedvezőbb. A dolgozatban ismertetett orvosi behívó rendszerben is LED-kijelzőt alkalmaztunk.

2.1.2.1 Működés, felépítés

A LED-kijelzők számos változatban, kivitelben és színben kaphatók. Léteznek szegmenses illetve pont-mátrixos kivitelben, illetve színes képi vagy mozgóképi információk megjelenítésére alkalmas, RGB LED-eket tartalmazó, úgynevezett LED-es kivetítők, LED-falak formájában. Kiviteltől eltekintve a kijelző alap építőköve minden esetben a fényemittáló dióda (LED), mely egy félvezető alapú eszköz.

A fényemittáló diódák működése az optikai lumineszcencia jelenségén alapul, amikor is a gerjesztett elektron alapállapotba való visszatérése során elektromágneses sugárzást (esetünkben fotont) bocsát ki. A fotont kibocsátó félvezetőket direkt félvezetőknak, míg az elsősorban hőtartományban sugárzó (fononokat kibocsátó) félvezetőket indirekt félvezetőknak nevezik. Direkt félvezetőnek minősül a GaAs (gallium-arzenid) és a GaN (gallium-nitrid) vegyület.

A fényt emittáló diódákat a kibocsátott fotonok minősége szerint az alábbi csoportokra bonthatjuk:

- IRED (InfraRed Emitting Diode): infravörös tartomány
- LED (Light Emitting Diode): látható fény tartománya
- SSDL (Solid State Diode Laser): indukált emisszió, lézerefény

Mivel az elektronikus megjelenítő eszközök a látható fény tartományában sugároznak, a további fejezetek kizárólag a LED-ekkel foglalkoznak.

2.1.2.2 A LED-ek típusai

Mivel a GaAs vegyület az infravörös tartományban sugároz, a látható fény eléréséhez a spektrumot különféle szennyezőkkel tolják el. A kibocsátott fény intenzitása szempontjából a LED-ek tovább osztályozhatók az alábbiak szerint.

2.1.2.2.1 Hagyományos LED-ek

Ezeket a típusokat általában információkijelzésre használjuk, mivel hatásfokuk meglehetősen alacsony, a hideg színek felé haladva fokozatosan csökken. Míg egy IR tartományban sugárzó IRED hatásfoka 1-5%-ig terjed, addig egy hagyományos zöld LED hatásfoka 0,05% körül mozog. A különböző színek eléréséhez különböző szennyezőket alkalmaznak: a GaAsP vegyület vörös, a GaAsP:N vegyület sárga, míg a GaP összetétel zöld színt eredményez.

Ahhoz, hogy az egyes típusoknak azonos gerjesztőáram mellett azonos legyen a fényárama, egyrészt a szem természetes érzékenysége gondoskodik (a legérzékenyebb tartomány a zöld/sárga szín határán van), ezen kívül a jobb hatásfokú vörös LED-ekbe abszorpciós réteget (epoxy) helyeznek el, míg a sárga és zöld LED-eket reflexiós réteggel látják el.

A hagyományos LED-ek nyitóirányú feszültsége a hideg színek felé haladva növekszik. Míg egy vörös LED tipikus nyitófeszültsége 1,8-1,9V, egy zöld LED esetén ez az érték 2,2-2,4V között van.

A hagyományos LED-ek névleges gerjesztőárama 20mA. A katalógusok ehhez az értékhez adják meg a fényáram, és a nyitófeszültség értékeket.

Megjegyzendő, hogy a LED-ek nyitóirányú karakterisztikája nagyon hasonló egy hagyományos diódáéhoz, így dinamikus ellenállásuk is viszonylag alacsony, $\sim 10 \Omega$. Emiatt korrekt meghajtásuk a legtöbb esetben áramgenerátorral valósítható meg.

2.1.2.2.2 Nagy fényerejű LED-ek

A közelmúltban kifejlesztett, nagy fényerejű LED-ek előnye, hogy fényáramuk azonos gerjesztőáram esetén jelentősen (40-50-szeresen) meghaladja a hagyományos LED-ek fényáramát. Míg egy hagyományos vörös LED fénykibocsátása 20mA gerjesztőáram esetén 150mCd, ugyanezen meghajtó áram mellett egy új típusú, nagy fényerejű vörös LED fénykibocsátása 5300mCd. Anyaguk jellemzően AlInGaP, valamint ezek szennyezett változatai.

A nagy fényerejű LED-ek közül legkésőbb, a 90-es években jelent meg a kék LED, mely alapanyaga InGaN.

2.1.2.2.3 Fehér LED, teljesítmény LED-ek

A fehér szín előállítása kezdetben korlátokba ütközött, mivel a fehér fény spektruma összetett, szélesebb tartományú, mint amit egy LED-del elő lehet állítani. Jelentős előrelépés volt a fehér LED-ek fejlesztése során a nagy fényerejű kék LED-ek hatékonyságának növekedése, és előállítási költségeinek csökkenése.

A fehér szín előállításának egyik módja, hogy egy kék színű LED-et egy foszfor jellegű abszorpciós réteggel borítanak, az ez által kibocsátott sárgás fény a kék fénnel keveredve a nappali fényviszonyokhoz hasonló fehér fényt eredményez. A másik lehetséges előállítási mód RGB LED-ek alkalmazása, amikor is a három alapszínből alakul

ki a fehér fény. A hétköznapi alkalmazásban azonban elterjedtebb az előbbi technológia, annak olcsósága miatt. Egy nagy fényerejű kék LED fénykibocsátása 20mA gerjesztőáram esetén 700mcd, a fehér LED-nél pedig 1200mCd, mely értékek lényegesen elmaradnak a nagy fényerejű vörös LED-ektől.

A teljesítmény LED-ek nem csak összetételbeli, hanem konstrukciós különbségeket is hordoznak a hagyományos LED-ekhez képest. Bár hatásfokuk jónak mondható (kb. 65%, 50-150lumen/W), kis térfogaton viszonylag nagy hőenergia szabadul fel, ezért hűtésükről gondoskodni kell. Kisugárzási karakterisztikájuk alkalmazásfüggő. Napjainkban egyre több típusú fényforrásban tűnnek fel, kiváltva a hagyományos izzókat.

2.1.2.3 LED kijelzők felépítése

A LED kijelzők már említett nagy előnye aktív fényforrás jellegük, mely jó konstrukció esetén a fényviszonyoktól közel független láthatóságot, olvashatóságot eredményez. Előnyeik közé sorolható még hosszú élettartamuk, illetve, hogy nem igényelnek karbantartást.

Alapvetően négy fő típusa létezik:

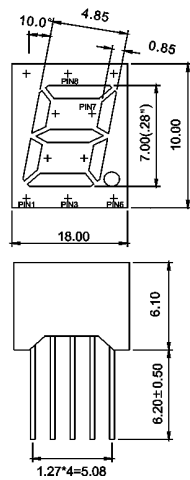
- pontszerű kijelzők (állapotjelzők)
- skála/vonal kijelzők
- numerikus/alfanumerikus szegmenses kijelzők
- pont-mátrix kijelzők

2.1.2.3.1 Numerikus/alfanumerikus szegmenses LED kijelzők

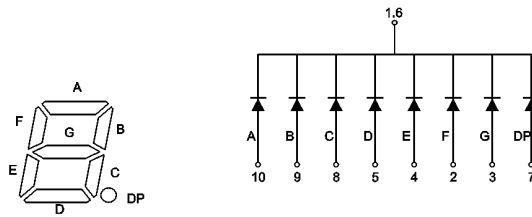
A szegmenses LED kijelzők leggyakrabban számszerű információk megjelenítésére használatosak (7 szegmenses kijelzők), illetve korlátozottan szöveges információk megjelenítését is lehetővé teszik (16 szegmenses kijelzők). Alkotóelemeikként többnyire a hagyományos LED típusok használatosak, azonban újabban megjelentek a nagy fényerejű LED-ekkel szerelt kijelző típusok is, melyek alkalmasabbak a multiplex kijelzési üzemmódra. A kivezetések számának csökkentése érdekében a LED-ek anódjai illetve katódjai közösítve vannak, ezáltal beszélhetünk közös anódos valamint közös katódos kijelzőkről. Az alábbi ábrán egy hagyományos, 7 szegmenses LED kijelző felépítése látható. A szegmensek nyitóirányú feszültsége 3,8-4,6V között változik típustól függően, mivel többnyire egy szegmens megvilágítására két, egymással sorosan kapcsolt LED-et

használnak, azonban ettől eltérően léteznek szegmensenként egy LED-et tartalmazó kijelzők is.

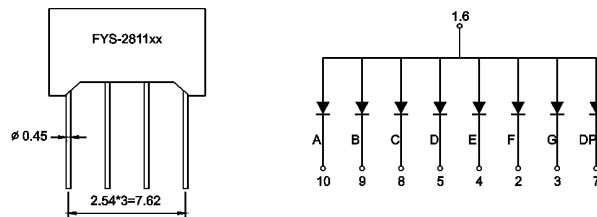
FYS-2811 Series



FYS-2811Ax

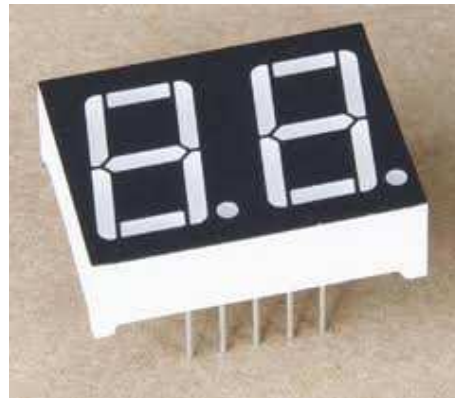


FYS-2811Bx

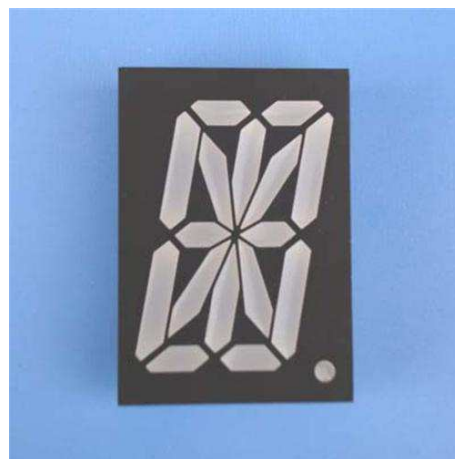


5. ábra: Hétszegmenses LED kijelző felépítése, bekötés

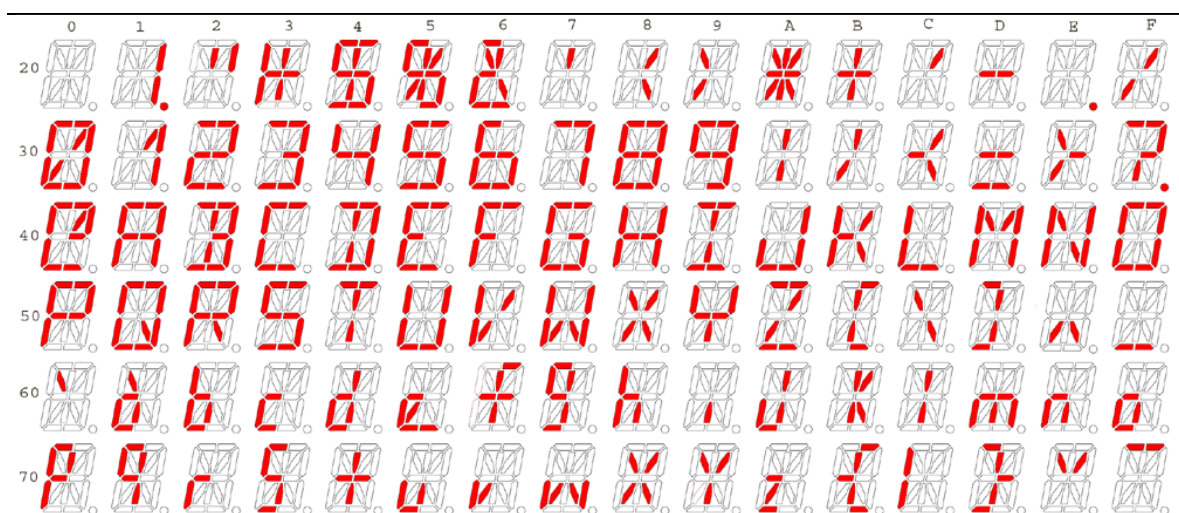
Egyes esetekben a kijelző egység több szimbólum megjelenítésére is alkalmas. Az egymástól függetlenül megjeleníthető szimbólumokat numerikus kijelzők esetén „digiteknek”, míg alfanumerikus kijelzőknél „karaktereknek” nevezik.



6. ábra: Két digités hétszegmenses LED kijelző



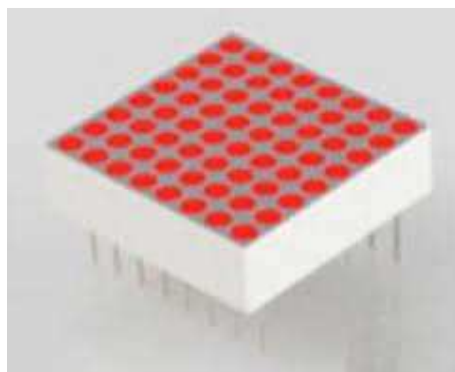
7. ábra: Tizenhat szegmenses egykarakteres LED kijelző



8. ábra: Különféle karakterek megjelenése tizenhat szegmenses kijelzőn

2.1.2.3.2 Pont-mátrix LED kijelzők

Mivel a szegmenses kijelzők csak korlátozottan alkalmasak szöveges információ megjelenítésére, ezért megjelentek a pont-mátrix kijelzők, melyek meghatározott számú pixelből állnak. Minden pixelnek egy LED dióda felel meg. A pixelek nagy száma miatt kizárólag a multiplexelt megjelenítés jöhet szóba ennél a típusnál, mivel a folytonos üzemmód rendkívül sok alkatrészt, adatportot igényelne.



9. ábra: 8x8 pixel méretű pont-mátrix kijelző

A szöveges információk mellett elegendő pixelszám esetén képi információkat is megjeleníthetünk ezeken a kijelzőkön. Léteznek monokróm típusú pont-mátrix kijelzők, melyek egy színnel képesek az információt megjeleníteni, valamint úgynevezett több színű, illetve „true-color” kijelzők (RGB LED-ekből felépítve), melyek akár színes képi információ megjelenítésére is alkalmasak.

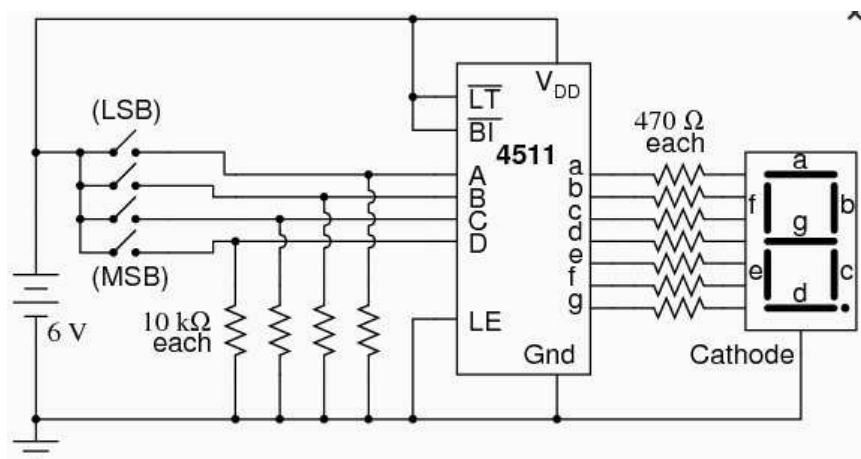
A pontmátrix kijelző pixelenkénti nyitófeszültsége általában 1,8-2,4V között van, mivel egy pixel megvalósításához többnyire egy LED kerül beépítésre. Ezen típusokban is egyre gyakrabban alkalmazzák a nagy fényerejű LED-eket.

2.1.2.4 LED-kijelzők üzemmódjai

Attól függően, hogy hány független elemet (szegmenst, pixelt) tartalmaz egy kijelző, a LED-ek üzemmódjait az alábbi módon csoportosítjuk:

2.1.2.4.1 Folytonos üzem

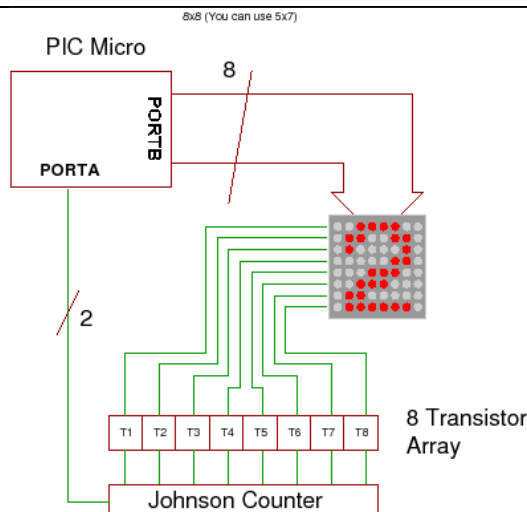
Folytonos üzemben a LED vagy folyamatosan kikapcsolt-, vagy folyamatosan bekapcsolt állapotban van. A váltás frekvenciája megegyezik a kijelezni kívánt információ frekvenciájával. A kijelző fényerejét a LED áramának változtatásával lehet növelni illetve csökkenteni, azonban megjegyzendő, hogy a kialakuló fényáram egy adott áramerősség felett nem nő arányosan a gerjesztőárammal. Folytonos vezérlés esetén a gerjesztőáram beállítása egyedileg történik minden LED-nél.



10. ábra: Hétszegmenses LED kijelző folytonos üzemben

2.1.2.4.2 Multiplex-üzem

Ebben az esetben a szem tehetetlenségét kihasználva a kijelző szegmenseit/pixeleit nem egyszerre, hanem részenként, a részeket folyamatosan váltogatva működtetjük. Ez esetben minden részegység azonos ideig kap vezérlést. A váltogatás frekvenciájának elegendően nagynek kell lennie ahhoz, hogy a szem a kialakult képet folytonosnak lássa, ne érzékeljen vibrálást. Multiplex üzemben azonban a szem az átlagos fényerőt érzékeli, vagyis a bekapcsolás idejére ennek a többszörösét kell biztosítani a LED-eknek. Az a tény, hogy egy LED impulzus üzemben rövid időre a névleges áramának többszörösével is vezérelhető, valamint a manapság elterjedt nagy fényerejű LED-ek jelenléte egyaránt alkalmassá teszi a LED-et a feladatra. A multiplex üzem azonban bonyolultabb elektronikát igényel, mint a folytonos üzem.



11. ábra: Példa pont-mátrix kijelző multiplex üzemére

2.1.2.4.3 Moduláris felépítésű pont-mátrix kijelzők

Mivel a multiplex vezérlés a LED-ek korlátozott csúcsárama miatt csak egy meghatározott pixelszámig lehetséges, a nagyobb méretű és pixelszámú kijelzőket modulárisan építik fel. Az egyes modulok meghatározott számú pixel megjelenítésére alkalmasak, melyek sorba fűzhetők, szinkronizálásukról és a megfelelő információ-áramlásról egy vezérlő egység gondoskodik. A modul a bemenetén a vezérlőtől csupán az aktuálisan megjelenítendő információt (kép/szöveg) kapja meg, a megjelenítés lebonyolításáért, az egyes kijelzők meghajtásáért saját maga felel.

2.1.2.5 Az alkalmazott pont-mátrix LED kijelző

Az általunk kiválasztott kijelző a Shenzhen Yuhong Electronics Ltd. terméke, egy modulárisan felépített pont-mátrix megjelenítő. Kedvező ára, valamint az általa nyújtott szolgáltatások széles köre miatt széles körben alkalmazható a nyilvános információközlésben. Az általunk választott típus CPower-1200 típusú vezérlővel rendelkezik, mely számos kijelző típus meghajtására alkalmas.



12. ábra: Shenzhen Yuhong LCD-kijelző

2.1.2.5.1 A LED-kijelző főbb adatai

Típusazonosító	F16X128R-P7.62
Felbontás	16x128 pixel
Kijelző mérete	122mmx975mm
Megjelenítés színe	vörös
Megjeleníthető elemek	szöveg, szám, kép, stb.
Kommunikációs interfész	RS232
Tápellátás	DC 5V/8A (szállítva 230VAC/5VDC hálózati adapterrel)

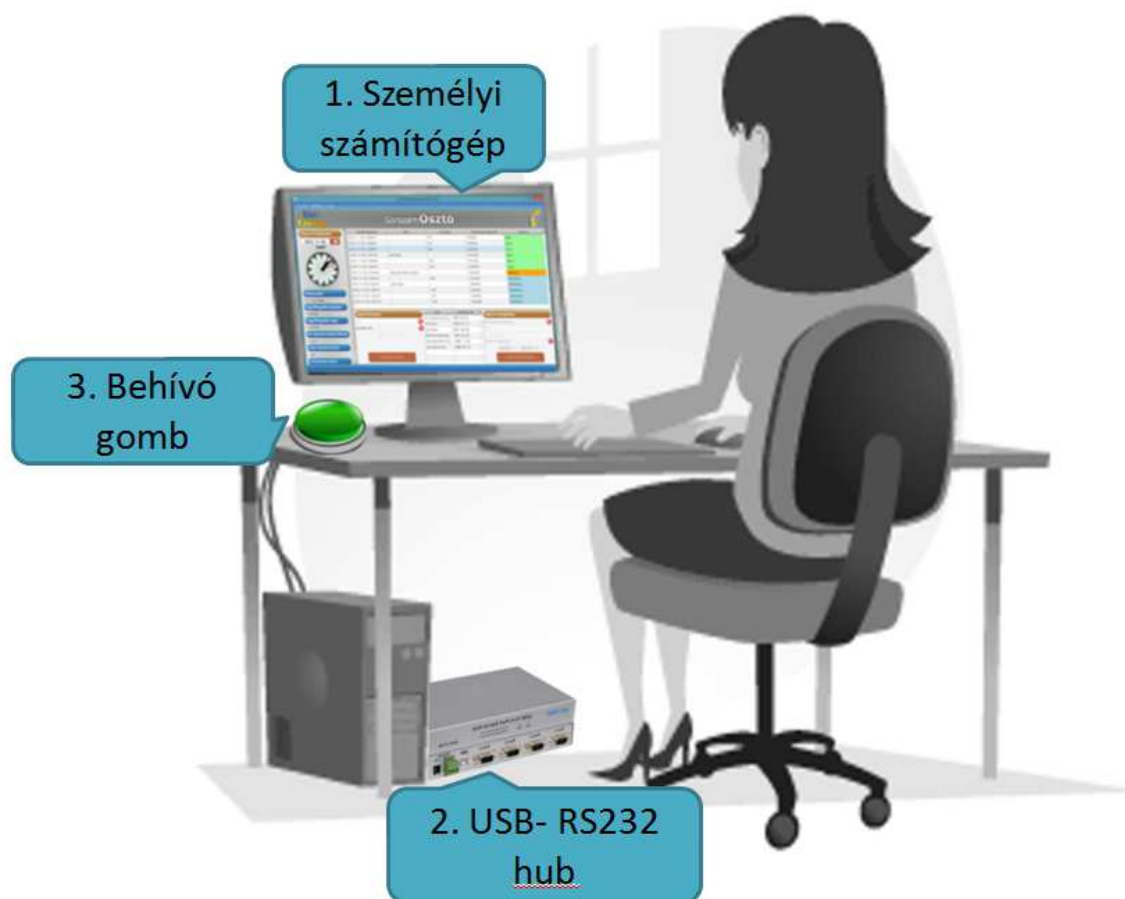
13. ábra: A LED-kijelző főbb adatai

2.1.3 Hangszóró

A rendszer a betegek behívásakor a LED-kijelzőn való vizuális megjelenítés mellett hanggal is jelzi a behívás tényét. Ekkor egy figyelemfelkeltő rövid dallamot játszik le az elsődleges hangkimeneten. A váróteremben egy hangszórót helyezünk el, melyet a PC hangkimenetre kötünk. A sorszámosztó és betegbehívó tervezésekor nagyon fontos szempont volt az akadálymentesítés. A látáskárosult emberek esetében a LED-kijelző alkalmazása sajnos nem elégséges. Ezen speciális igényt szem előtt tartva beépítettünk a rendszerbe egy magyar nyelvű beszédszintetizátort, mely valós időben képes az írásbeli adatok beszéddé alakítására (TTS- Text To Speech). Segítségével a rövid hangjelzés után a soron következőt név vagy sorszám alapján audiovizuális módon lehet behívni. A beszédszintetizátor által előállított beszéd tiszta és érthető annak ellenére, hogy a magyar dialektus az egyik legbonyolultabb és legnehezebben utánozható nyelv a világon [3].

2.2 A fogadóhelységben lévő hardverkomponensek

A rendszer hardveres tervezésekor kiemelt figyelmet fordítottunk arra, hogy egyszerűen csatlakoztatható legyen a már meglévő informatikai struktúrához.



14. ábra: Hardverkomponensek a fogadóhelységben

2.2.1 Személyi számítógép

A legtöbb orvosi rendelőben van személyi számítógép. A hardverkomponensek csatlakoztatásához mindössze egy szabad USB portra, illetve egy 3,5 mm-es hangkimenetre van szükség. Sok esetben tehát a rendszer beszereléséhez nincs szükség külön PC beszerzésére. Természetesen az sem jelent nagy problémát, ha a rendelőben nincs számítógép. A napjainkban kapható legegyszerűbb PC is bőségesen elegendő teljesítménnyel rendelkezik a rendszer kiszolgálásához, így a beszerelés költsége csak minimálisan növekszik.

2.2.2 USB- RS232 Hub

A nyomógombok, LED kijelző és hőpapíros nyomtató egyaránt RS232 soros interfészen keresztül csatlakoztatható. Egy nagyon kiforrott és megbízható kommunikációs szabványról van szó. Azonban egyre kevesebb új számítógépen található meg natív módon beépítve a soros port. Bizonyára szerepe van ebben a modernebb és univerzális USB

szabvány rohamos elterjedésének. Léteznek USB- RS232 átalakítók, melyek lehetővé teszik, hogy RS232 eszközt csatlakoztassunk a számítógép USB portjára.

A sorszámosztó rendszer esetében azonban alapkiépítésben is 4 darab soros eszközt kell csatlakoztatni. Ha négy külön USB- RS232 átalakítót alkalmaznánk, az nem lenne gazdaságos, mivel a számítógép 4 USB portját foglalná le a rendszer. Szerencsére léteznek úgynevezett „Hub”-ok, melyek transzparens módon képesek virtuálisan megsokszorozni egy portot. Az általunk használt 4 portos USB- RS232 Hub lehetővé teszi, hogy a számítógép egyetlen USB portjáról mind a négy hardvereszközt kezelni tudjuk.



15. ábra: Az általunk használt USB- RS232 Hub

2.2.3 Behívó gomb

Nemcsak a betegek számára szeretnénk volna a lehető legegyszerűbbé tenni a rendszer használatát, hanem a háziorvosok és asszisztensek számára is. A soron következő beteg behívásához semmi egyéb nem kell tenniük, mint megnyomni az erre szolgáló nagyméretű nyomógombot. A vezérlőszoftver a gomb lenyomásának hatására kikeresi a soron következő beteg nevét vagy sorszámát, majd kiküldi az adatot a LED-kijelzőre, valamint a beszédszintetizátor legenerálja és a hangszórón lejátsza a megfelelő hangfájlt a figyelemfelkeltő dallam után.

3 A sorszámosztó rendszer szoftveres felépítése

A szoftveres megvalósításhoz kizárólag nyílt forráskódú, szabadon felhasználható technológiákat alkalmaztunk. A sorszámosztó és behívó rendszerünk meglehetősen összetett hardveres szempontból, több különféle eszköz hibátlan és biztonságos működését kell garantálnunk. Nyilvánvaló, hogy bármilyen apró hiba hatalmas fennakadásokat okozhat az ügyfelek kiszolgálásában. A hardverkomponensek működését összehangoló vezérlőprogram JavaFX technológián alapul, mely napjainkban viszonylag újdonságnak számít.

3.1 JavaFX vezérlőprogram

A JavaFX az amerikai szoftveróriás Oracle Corporation fejlesztése, egy újfajta Java-alapú szoftver platform. A JavaFX drasztikusan lerövidült termelési ciklust kínál a Java-fejlesztők és a webfejlesztők számára egyaránt, valamint megkönnyíti a grafikát, videót, animációt és gazdag szövegfunkciókat tartalmazó alkalmazások létrehozását [4]. Kifejlesztésének egyik fő mozgatórugója az volt, hogy leváltsa az egyre inkább korosodó és elavuló Java felhasználói felület technológiákat (Swing, AWT). A modern kor igényeit szem előtt tartva az asztali programok mellett az úgynevezett gazdag internetes alkalmazások (RIA- Rich Internet Application) fejlesztése is támogatott. További előny az alacsony hardverigény, valamint a Java technológiából eredő platformfüggetlenség.

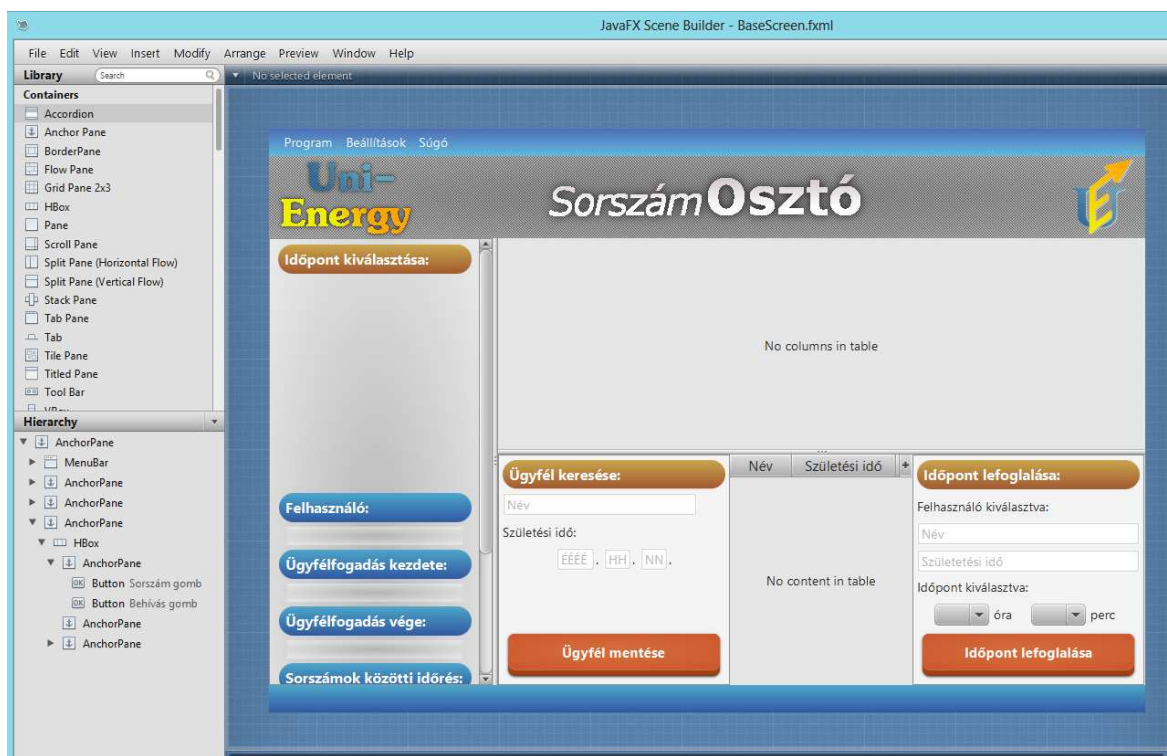
Nagyban egyszerűsíti a fejlesztést azon napjainkban egyre több fejlesztői környezetben (Android, .NET) megjelenő szemléletmód, mely szerint különválasztják a felhasználói felület létrehozását a kódolástól. A JavaFX technológia esetén a felhasználói felület elemeit egy XML alapú, úgynevezett FXML leíró nyelv segítségével definiálhatjuk [5]. Az elemek kinézetét, viselkedését a webfejlesztők számára jól ismert CSS (Cascading Style Sheet) technológia segítségével adhatjuk meg [6]. A jelenlegi 2.2-es verzióban lehetőségünk van asztali, valamint webalkalmazások fejlesztésére Windows, Linux és Mac OS X platformokon, tehát a legnépszerűbb operációs rendszerek kivétel nélkül támogatottak. A JavaFX ezen tulajdonsága kiemelten fontos a projektünk

szempontjából. A technológia használatához mindössze a legújabb JRE (Java Runtime Environment) telepítése szükséges.

3.1.1 JavaFX Fejlesztői környezet

A JavaFX platformon való fejlesztés alapeszköze a méltán népszerű NetBeans IDE (Integrated Development Environment). Használata nagy mértékben gyorsítja és egyszerűsíti a fejlesztést. Az elkészült programot telepíthető formában tehetjük közzé használatával.

A felhasználói felület létrehozását a JavaFX SceneBuilder nevű fejlesztői eszköz nagyban egyszerűsíti. A fejlesztő egy grafikus szerkesztőben hozhatja létre a felületet, ezáltal nem kell kódolnia. A SceneBuilder automatikusan generálja a felhasználói interfészt leíró FXML kódot.



16. ábra: JavaFX Scene Builder

3.1.2 Verziókövető rendszer

A sorszámosztó rendszer fejlesztése egy összetett projekt, ezért különösen indokolt volt valamilyen verziókövető szolgáltatás használata. Választásunk a Subversion (SVN) rendszerre esett, melyet a NetBeans fejlett módon támogat. A forráskódot és a dokumentációt egy internetes tárhelyre (Assembla) szinkronizáltuk, így a kutatócsoport

minden tagja kényelmesen és egyszerűen követhette nyomon a munkát. A felhasznált fejlesztői szoftvereknek nincs licenszdíja, így szoftveres oldalon biztosított az alacsony fejlesztési költség.

3.2 Adatbázis

A sorszámosztó és behívó rendszer működése során rengeteg olyan adat keletkezik, melyet megfelelő módon tárolnunk kell a zavartalan és gyors működés biztosítása érdekében. A fejlesztés kezdetekor úgy döntöttünk, hogy a rendszerhez egy modern és gyors relációs adatbázis kezelőt kell használnunk. A legtöbb rendszernek azonban magas licensz díja van, mely extra kiadást mi mindenképpen szerettünk volna elkerülni. Az adatbázis háttér és konzisztens adattárolás biztosítására az üzleti célokra is szabadon felhasználható PostgreSQL RDBMS-t (Relational Database Management System) választottuk [7]. Az alábbiakban az adatbázisban tárolandó legfontosabb adatokat ismertetem.

3.2.1 Felhasználók

A sorszámosztó rendszer felhasználói maguk a háziiorvosok. Az adatbázis tárolja nevüket, rendelési idejüket napi bontásban, továbbá a sorszámokra vonatkozó egyéb beállításokat, például a sorszámon megjelenítendő adatokat. A sorszámosztó és behívó tehát több felhasználót támogat. Ez nagyon hasznos tulajdonság, mivel a háziiorvosoknál gyakran előfordul, hogy délelőtti és délutáni műszak szerint váltják egymást. Fontos megkötés azonban, hogy egy időben csakis egy felhasználó használhatja a rendszert.

3.2.2 Páciensek jegyzéke

Az általunk fejlesztett rendszer egyik nagy újítása, hogy a sorszámosztás mellett támogatja az időpont előfoglalást. A pácienseknek tehát lehetőségük van előzetes egyeztetéssel időpontot foglalni a rendelési időben, ezzel minimalizálva a várakozási időt. A betegbehívó ezen pácienseket kiemelt prioritással kezeli, behívási sorrendjüket pedig intelligens módon képes összefésülni a sorszámmal rendelkező betegekkel. A funkció működéséhez el kell tárolnunk a páciensek nevét és születési idejét.

3.2.3 Kiosztott sorszámok és előfoglalások

A legfontosabb, hogy a kiosztott sorszámokat biztonságos és konzisztens módon tároljuk el. Ellenkező esetben egy esetleges rendszerhiba vagy áramszünet hatalmas problémákat okozna. A legkézenfekvőbb megoldás az adatbázis használata volt.

3.3 Beszédszintetizátor

Magyar nyelvű beszédhang előállítására a nyílt forráskódú eSpeak beszédszintetizátort használjuk fel. A JavaFX technológiához és a PostgreSQL adatbáziskezelőhöz hasonlóan szintén használható a legnépszerűbb operációs rendszereken, azaz a legtöbb Linux disztribúción, Windows és Mac OS X platformon [8].

3.4 A rendszer telepítése

A szoftverek telepítése előtt gondoskodni kell a hardverelemek megfelelő összekötéséről. A két nyomógombot, LED-kijelzőt és hőpapíros nyomtatót az USB- RS232 Hub soros portjaira kell csatlakoztatni. A Hub Windows 7 és Windows 8 operációs rendszereken Plug&Play módon működik, azaz a számítógép USB portjára való csatlakoztatás után a rendszer automatikusan telepíti a szükséges eszközöket. Windows XP, Linux és Mac OS X operációs rendszerekhez a gyártó mellékelte a szükséges meghajtó programokat.

A Hub az USB- RS232 konverziót, a portok elosztását és kezelését transzparens módon végzi. A felhasználó úgy használhatja a soros portokat, mintha azok natív módon a számítógép részei lennének. Az operációs rendszer is ennek megfelelően érzékeli ezeket az eszközközelőben, ahol sikeres telepítés után azok megjelennek a hozzájuk automatikusan rendelt névvel (COM*) együtt. A hangszórót a számítógép 3,5 mm-es (Jack) hangkimenetére kell kötni.

A következő lépés a JRE (Java Runtime Environment) telepítése, mely lehetővé teszi JavaFX alkalmazások futtatását. Következhet a JavaFX vezérlőprogram, mely egyszerűen kezelhető önkicsomagoló telepítő (Setup.exe) formájában telepíthető.

A rendszer megfelelő működéséhez már csak két fontos komponens hiányzik, a beszédszintetizátor, illetve a PostgreSQL adatbázis. A beszédszintetizátor telepítése magától értetődő folyamat. Az adatbázis esetében telepítéskor létre kell hozni az

úgynevezett "superuser" felhasználót, illetve be kell állítani a port számát, melyen keresztül kapcsolat létesíthető. A fejlesztés során a 7767-es portot választottuk. A JavaFX vezérlőprogram számára dedikált, korlátozott jogokkal rendelkező felhasználót kell létrehozni, továbbá definiálni kell az adatbázis struktúrát, a táblák szerkezetét, valamint a hozzáférési jogokat. Mindezt egyetlen script fájlban végezzük el, melynek lefuttatásához a telepítéskor létrehozott superuser-ként kell bejelentkeznünk a PostgreSQL DBMS-be. Ezt a lépést nagyban leegyszerűsíti, ha az adatbázishoz mellékelt pgAdmin III grafikus alkalmazást használjuk.

4 A JavaFX vezérlőprogram működése

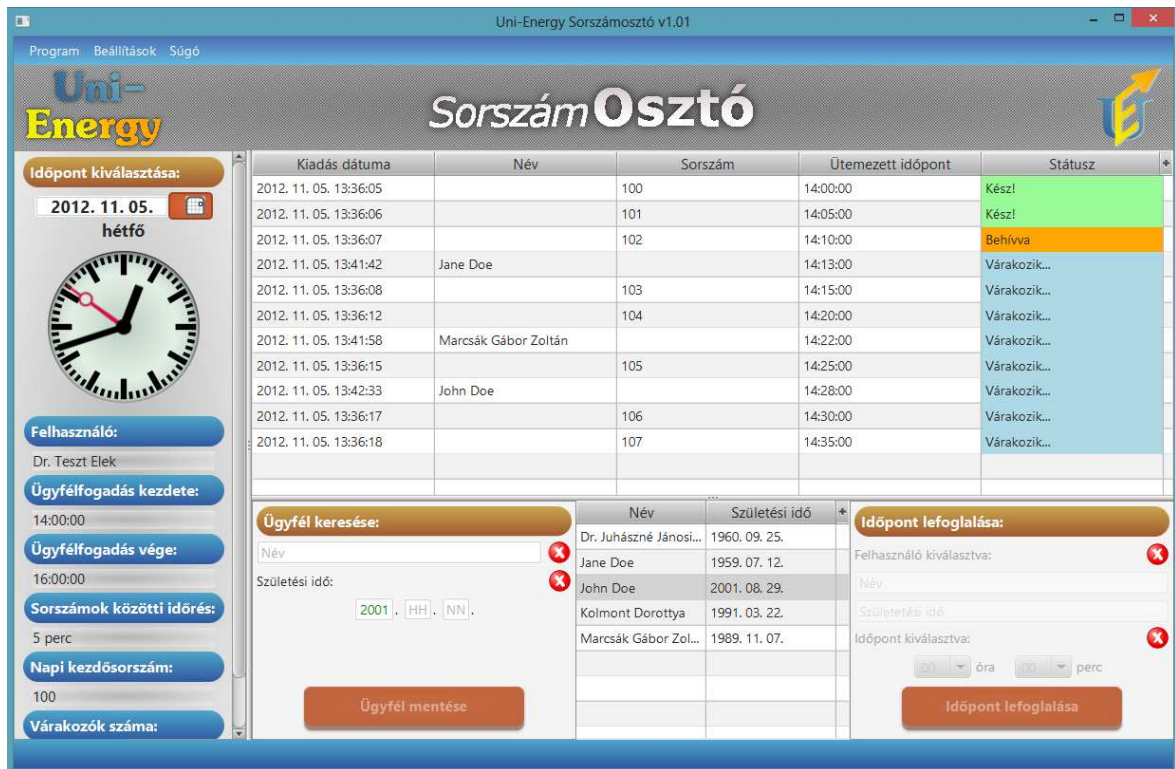
A vezérlőprogram a sorszámosztó és beteghívó legfontosabb eleme, tulajdonképpen a rendszer agya, mely összehangolja a hardver és szoftver komponensek működését. A rendszer összetettsége és bonyolultsága miatt a program forráskódja több mint negyven funkcionálisan jól elkülöníthető osztályból épül fel, valamint számos külső függvénykönyvtárt használ a különböző funkciók biztosításához. Az alábbiakban a fejlesztés főbb lépéseit, a program általános működését fogom ismertetni.

4.1 A program indítása

Indításkor több fontos művelet történik, mindenekelőtt a program teszteli, hogy tud-e csatlakozni az adatbázishoz. Amennyiben valamilyen okból nem sikerül, egy felugró ablakban tájékoztatja erről a tényről a felhasználót, valamint ajánlást tesz a PostgreSQL szolgáltatás működésének ellenőrzésére. Az adatbázishoz való kapcsolódást és a műveletek végrehajtását a JDBC (Java Database Connectivity) függvénykönyvtár segítségével valósítjuk meg.

Az adatbázis kapcsolat kiépülte után a regisztrált felhasználók listáját kérdezi le a program, melyet külön táblában tárolunk. Ha a lekérdezés eredményeként üres listát kapunk, akkor nincs regisztrált felhasználó. Ekkor a vezérlőprogram egy felugró ablakban figyelmeztet, hogy hozzunk létre egy felhasználót, különben a sorszámosztás nem kezdhető el. Abban az esetben, ha a lista nem üres, mindig az utoljára a programot bezáró felhasználó beállításai töltődnek le az adatbázisból. Természetesen lehetőség van felhasználóváltásra, az erre a célra szolgáló ablakot a későbbiekben fogom részletesen ismertetni.

Az idő egy nagyon fontos tényező a rendszer pontos működésének szempontjából. A program indításkor minden esetben példányosít egy Date típusú objektumot, mely az aktuális rendszeridőt tartalmazza. Az idő, valamint a felhasználó beállításainak ismeretében lekérdezhetők az adatbázisból az adott napon beállított rendelés idejére esetlegesen már lefoglalt időpontok és sorszámok. Az eredménylistát egy jól áttekinthető táblázatban jelenítjük meg.



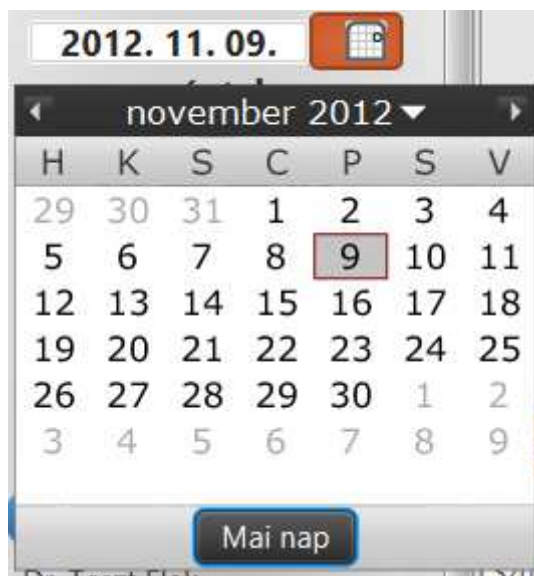
17. ábra: A JavaFX vezérlőprogram kezdőképernyője

Az idő kijelzésére két vezérlőt alkalmazunk. Egyrészt egy naptárt a nagyobb időintervallumok megjelenítésére, valamint egy analóg órát.



18. ábra: Vezérlők az idő kijelzésére

A naptár alapesetben el van rejtve, csak a dátumkijelző mellett található narancssárga gomb hatására jelenik meg a felhasználói felületen. Mivel a JavaFX viszonylag új technológiának számít, nem minden, más platformon alapvetőnek számító vezérlő található meg benne. Ilyen például a naptár is, melyet ezen okból kifolyólag nekünk kellett létrehoznunk. A munka során kiemelt figyelmet fordítottunk a magyar nyelvű lokalizáció használatára, valamint az egyszerű és gyors kezelhetőségre. A program indulásakor a naptár és az analóg óra is a rendszeridőnek megfelelő dátumra állítja be magát.



19. ábra: A Naptár vezérlő

A felhasználó adatait és beállításait az időkijelzők alatt jelenítjük meg. A legfontosabb adatok az adott napra beállított rendelési idő, a napi kezdő sorszám és az éppen várakozók száma. A program indulásakor az utolsó adatbázis lekérdezés a regisztrált páciensek nevét és születési idejét tölti fel egy listába, melyet szintén egy táblázatban jelenítünk meg.

4.2 Többszálú architektúra

A fejlesztés során fontosnak tartottuk, hogy egy gyors és reszponzív szoftvert hozzunk létre. Ez csak úgy lehetséges, ha az időigényes hardver és adatbázis műveleteket rendre külön dedikált szálon hajtjuk végre. A felhasználói interfész száljában kizárólag az ahhoz kapcsolódó programrészek futhatnak. A programozási szemlélet betartásával véleményem szerint sikerült egy a mai kor igényeinek megfelelő, látványosan és gyorsan működő felhasználói felületet létrehoznunk.

4.3 Hardverelemek vezérlése

A sorszámosztó és behívó rendszer fő hardverelemei a nyomógombok, a hőpapíros nyomtató és a LED kijelző. A JavaFX alkalmazásnak mindezen eszközöket megbízhatóan kezelnie kell. A csatlakozói interfész minden esetben az USB- RS232 Hub által biztosított soros port.

4.3.1 Nyomógombok

Java környezetben a soros port kezelésére számos külső függvénykönyvtár létezik. A projektben az RXTX [9] nevű függvénykönyvtárt használtuk fel erre a célra. Tanulmányoznunk kellett a nyomógombok soros portra kötésének lehetőségeit. Szoftveres szempontból egy olyan eseményre volt szükségünk, mely a nyomógomb lenyomásakor aktivizálódik. Az RXTX úgynevezett „Listener” használatával teszi lehetővé a soros port monitorozását. Egy olyan osztályt kell létrehozni, mely implementálja a „SerialPortEventListener” interfészt.

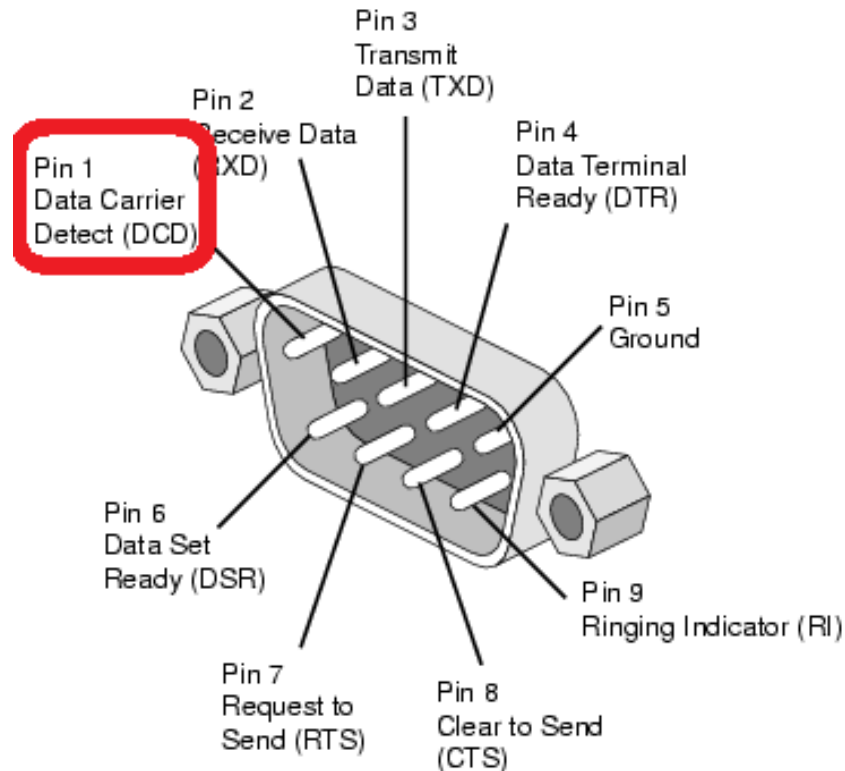
```
public class BehivasGombListener implements Runnable, SerialPortEventListener {...}
```

Az osztály konstruktorában kell beállítani azon port nevét, melyet monitorozni szeretnénk a Listener segítségével, valamint a setSerialPortParams() függvény segítségével meg kell adni a különböző soros port specifikus beállításokat.

```
CommPortIdentifier portId1 = CommPortIdentifier.getPortIdentifier("COM4");  
SerialPort serialPort1 = (SerialPort) portId1.open("ComControl", 2000);  
serialPort1.setSerialPortParams(9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,  
SerialPort.PARITY_NONE);
```

A nyomógombokat úgy kötöttük be, hogy lenyomásakor zárják a soros port DataCarrierDetect (DCD) lábát. Ez a kulcslépés a nyomógombok lenyomásának érzékeléséhez. Az úgynevezett „EventListener”-t úgy hozzuk létre, hogy figyelje ezen esemény bekövetkezését.

```
serialPort1.notifyOnCarrierDetect(true);  
serialPort1.addEventListener(this);
```



20. ábra: DataCarrierDetect (DCD) láb a soros porton

Ha valaki lenyomja a nyomógombot, bekövetkezik az esemény és megváltozik a DCD flag értéke (az RXTX a DCD flag-re CD flag-ként hivatkozik). A Listener lefuttatja az eseményhez tartozó programkódunkat.

```
public void serialEvent(SerialPortEvent event) {
    switch (event.getEventType()) {
        case SerialPortEvent.CD:
            System.out.println("CD changed");
    }
}
```

A Listener használatakor ügyelnünk kell arra, hogy folyamatosan fut egy saját számban. Ha a felhasználó kilép a programból, a szálat meg kell szüntetnünk, és fel kell szabadítanunk az erőforrásokat. Minden porthoz saját Listener-t hozhatunk létre, nagyon fontos azonban, hogy egy portot egyszerre csak egy Listener monitorozhat. Ellenkező esetben „TooManyListenersException” kivétel fog dobódni.

4.3.2 Hőpapíros nyomtató

A hőpapíros nyomtató vezérléséhez soros porton kell különböző bájtömböket kiküldeni. A nyomtató leírása hosszasan taglalja, hogy bizonyos funkciók elérése pontosan milyen karakterkombinációval történik.

Ahhoz hogy adatot tudjunk kiküldeni soros porton, a nyomógomboknál látott módszerrel megegyezően definiálni kell a portot neve (COM*) alapján, valamint be kell állítani a különböző kommunikációs paramétereket.

```
CommPortIdentifier portId2 = CommPortIdentifier.getPortIdentifier("COM3");  
SerialPort serialPort2 = (SerialPort) portId2.open("ComControl", 2000);  
serialPort2.setSerialPortParams(38400, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,  
SerialPort.PARITY_NONE);
```

Ezután egy kimenő csatornát kell létrehozni, az úgynevezett „OutputStream” típus példányosításával.

```
OutputStream out;  
out= serialPort2.getOutputStream();
```

A kiküldéshez egyszerűen az OutputStream-be írjuk a bájtömböt. A flush() függvénnyel biztosítjuk, hogy a teljes tömb kiküldésre került.

```
out.write(bytes);  
out.flush();
```

A nyomtató kezelése, ezáltal a sorszámok nyomtatása a fent vázolt két függvény hívásának ismételtetésével lehetséges. Az általunk használt hőpapíros nyomtató fejlett szövegformázási lehetőségekkel rendelkezik. Beállítható a karakterek betűkészlete, mérete, típusa. Lehetőség van a szöveg igazításának beállítására, sorköz és térköz használatára. A funkciókat a megfelelő bájtömb kiküldésével tudjuk beállítani. Gyakorlati példával szemléltetve szöveg középre igazításához az alábbi tömböt kell elküldeni a nyomtatónak.

```
byte[] kozeprelgazitas = new byte[]{27, 29, 97, 1};
```

Minden olyan karaktert, amit szeretnénk a sorszámra nyomtatni, szintén bájtömbként kell kiküldeni. Problémát jelenthet a speciális, ékezetes karakterek használata. A nyomtatóon lehetőség van a karakterkészlet kiválasztására, így a kapott bájtűkódokat helyesen tudja értelmezni, és a megfelelő karaktert papírra nyomtatni. A Java nyelvben a karaktertömböket String típusként szokás megadni. A getBytes() függvénnyel lehetőség van a String-ek bájtömbbé konvertálására. Az általunk a nyomtatóon használt "Latin2" karakterkészlet csak akkor nyomtatta a megfelelő karaktereket, ha a getBytes(Cp852) paraméterrel konvertáltuk a bájtömböket. Utóbbi esetben viszont az ékezetes és speciális karakterek is kifogástalanul jelennek meg.



21. ábra: Kinyomtatott sorszám ékezetes karakterekkel

A sorszám nyomtatásának befejeztével érdemes elvágni a tekercset, mely funkciót a korábban ismertetett „Guillotine” látja el. Működtetéséhez az alábbi bájttömböt kell kiküldeni

```
byte[] papirVagas = new byte[]{27, 'd', 1};
```

4.3.3 LED kijelző

Az általunk kiválasztott kijelző CPower-1200 típusú vezérlővel rendelkezik, mely egy igen elterjedt és megbízható típus. Használatához a gyártó mellékel egy SDK-t

(Software Development Kit), mely egy .DLL függvénykönyvtárat, és a különböző programnyelveken implementált példákat tartalmaz. A kijelző vezérléséhez a mellékelt .DLL fájl függvényeit kell használnunk.

4.3.3.1 JNA, JNI

Java fejlesztői környezetben a .DLL fájlok támogatására két elterjedt függvénykönyvtár használható, az egyik a JNI (Java Native Interface), a másik pedig a JNA (Java Native Access). A projekt során JNA használata mellett döntöttünk.

4.3.3.2 .DLL függvénykönyvtár használata Java környezetben

A .DLL fájlban található függvényeket C++ nyelven fejlesztették. A gyártó mellékelte a függvények leírását. Java JNA környezetben úgy lehet külső .DLL könyvtárat használni, ha létrehozunk egy interfészt a könyvtárban definiált függvények prototípusával.

```
public interface CP5200 extends Library {
    public int CP5200_RS232_InitEx(String fName, int nBaudrate, int dwTimeout);
    public int CP5200_RS232_SendText(int nCardID, int nWndNo, String pText, int crColor, int nFontSize, int nSpeed, int nEffect, int nStayTime, int nAlignment);
    public int CP5200_RS232_SendPicture(int nCardID, int nWndNo, int nPosX, int nPosY, int nCx, int nCy, String pPictureFile, int nSpeed, int nEffect, int nStayTime, int nPictRef);
}
```

Ezután be kell töltenünk a DLL fájlt a Java kódban. A Java Virtuális Gép akkor fogja megtalálni a fájlt, ha az aktuális Java osztállyal egy csomagban található meg.

```
CP5200 cp5200 = (CP5200) Native.loadLibrary("CP5200", CP5200.class);
```

A függvénykönyvtár sikeres betöltése után használhatjuk az interfészben definiált függvényeket. Elsőként a nyomógombokhoz és a hőpapíros nyomtatóhoz hasonlóan a megfelelő portot kell inicializálni.

```
int nRet = 0;
String strPort = "COM6";
nRet = cp5200.CP5200_RS232_InitEx(strPort, 115200, 600);
return nRet;
```

A port inicializálása után lehet adatot küldeni a LED kijelzőre. A kijelző karaktereket és képeket is meg tud jeleníteni. Eredetileg karakteres módban szeretnénk volna használni, azonban sajnálatos módon azt tapasztaltuk, hogy csak az „extended ASCII”

karakterkészletet ismeri. Sajnos ebből az következett, hogy a magyar ékezetes karakterek megjelenítése nem lehetséges ilyen módon.

4.3.3.3 Karaktertömb konvertálása képpé

A megoldást az jelentette, hogy képet küldünk a kijelzőre. A Java fejlett képmanipulációs lehetőségekkel rendelkezik. Létrehoztunk egy olyan függvényt, melynek paraméterében tetszőleges szöveget átadva visszatérési értékeként egy, a kijelző méretével megegyező 128*16 képpontos képet állít elő. A kép két színből tevődik össze, ha azt szeretnénk, hogy az adott LED világítson, a képpontnak pirosnak (RGB(255,0,0)), ha pedig nem szeretnénk hogy világítson, a képpontnak feketének (RGB(0,0,0)) kell lennie. Ezzel a módszerrel az ékezetes és egyéb speciális karakterek is használhatók.

```
private BufferedImage processImageText(String s, int fontSize) {
    int w = 128;
    int h = 16;
    BufferedImage img = new BufferedImage(
        w, h, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2d = img.createGraphics();
    g2d.setPaint(Color.red);
    g2d.setFont(
        new Font("Dialog", Font.PLAIN, fontSize));
    FontMetrics fm = g2d.getFontMetrics();
    int stringWidth = fm.stringWidth(s);
    int stringHeight = fm.getHeight();
    int x = (img.getWidth() - stringWidth) / 2;
    g2d.drawString(s, x, y);
    g2d.dispose();
    return img;
}
```

A függvény visszatérési értékeként kapott képet a LED kijelző használatához definiált interfészben lévő CP5200_RS232_SendPicture() függvénnyel küldhető ki a kijelzőnek.

```
ret = cp5200.CP5200_RS232_SendPicture(1, 0, 0, 0, 128, 16, pathSendImage, 1, 63, 0, 0);
```

A függvény paraméterezésekor meg kell adnunk a kép méreteit, magát a kiküldendő képet, illetve több egyéb beállítás mellett azt, hogy milyen animációval jelenjen meg a kijelzőn.

4.3.3.4 Megjeleníthető adatok

A kijelzőn az általunk használt módszerrel gyakorlatilag bármilyen karakterkészletű szöveg megjeleníthető. A projekt során sorszámokat és neveket küldünk a kijelzőre. Ügyelnünk kellett arra, hogy a nevek esetében változó karakterhosszúsággal kell számolni. Beépítettünk a programba egy automatikus skálázási függvényt, mely a karaktertömb hosszától függően változtatja a képre rajzolt karakterek nagyságát. Úgy határoztunk, hogy olvasható módon legfeljebb 30 karakter küldhető ki a kijelzőre. Az ennél hosszabb karaktertömbök esetében a túllógó részt nem vesszük figyelembe.



22. ábra: Sorszám küldése a kijelzőre



23. ábra: Rövidebb név küldése a kijelzőre



24. ábra: Hosszabb név küldése a kijelzőre

4.4 Az adatbázis kezelése

A rendszer megbízható és biztonságos működésének alapfeltétele, hogy bármilyen külső hibatényező esetén az aktuális futási állapot visszaállítható legyen. A sorszámosztó ezért teljesen szinkronban működik a PostgreSQL relációs adatbázissal. Ha egy páciens sorszámot kér a terminálon lévő gomb lenyomásával, a JavaFX vezérlőprogram lefoglalja

azt a sorszámot a páciens számára, és mire a hőpapíros nyomtató kinyomtatja, az adatbázisba is bekerül a tranzakció. Hirtelen áramszünet utáni újraindulás esetén a program futási állapota visszaállítható az adatbázisból, és onnan folytatódhat a sorszámosztás, ahol a hibajelenség folytán megszakadt. Külön táblában kerülnek letárolásra a program felhasználói, a felhasználókhoz tartozó beállítások, a páciensek adatai és a kiosztott sorszámok. A JDBC függvénykönyvtár PreparedStatement, CallableStatement és ResultSet eszközeit használjuk az adatbázissal való kommunikációhoz.

4.5 Időpont előfoglalás

Az időpont előfoglalás lehetősége újdonságnak számít a sorszámosztó rendszerek világában. Használatával együtt lehet kezelni a rendelési időben érkező, előzetes időponttal nem rendelkező pácienseket, illetve azokat, akik előzetesen időpontot egyeztettek. A rendszer garantálja, hogy utóbbi betegek pontosan a lefoglalt időpontban, vagy ha ez nem lehetséges, az ahhoz legközelebb eső lehetséges időpontban jussanak be az orvoshoz.

Az előfoglalás használatához regisztrálni kell a pácienseket a sorszámosztó rendszerben, hiszen ekkor a behívás nem sorszám, hanem név alapján történik.

4.5.1 Páciensek jegyzéke

A páciensek jegyzéke az előzetesen regisztrált betegek nevét és születési évét tárolja.

25. ábra: A páciensek jegyzéke, időpontfoglalás

Természetesen mentés előtt a rendszer ellenőrzi, hogy a születési idő létező dátum-e, ellenkező esetben nem engedélyezi a műveletet. Az esetleges hibára úgynevezett "Tooltip", azaz egy figyelemfelkeltő grafikai elem segítségével hívja fel a felhasználó a figyelmét. A "Tooltip"-eket a programban több más helyen is alkalmaztuk.

26. ábra: Figyelmeztetés érvénytelen születési idő esetén

Sikeres regisztráció után a páciensek jegyzéke egy táblázatban kerül megjelenítésre. A regisztrációs felület egyben keresésre is szolgál, azaz ha egy regisztrált páciensnek időpontot szeretnénk foglalni, és ismerjük a nevét vagy születési évét, a megfelelő szövegdobozba beírva a táblázatban automatikusan a találatok jelennek meg.

A táblázat rendelkezik felugró menüvel, mely a kiválasztott páciens sorára jobb egérgombbal kattintva jelenik meg.

27. ábra: Felugró menü a páciensek listáján

A „Törlés” menüpontot választva a regisztrált beteg törlésre kerül. Az „Időpont foglalása” menüpont kiválasztásával időpontot jegyezhetünk elő az aktuális páciens számára.

28. ábra: Időpontfoglalás a páciens számára

Az időpont az aktuális kiválasztott napra kerül lefoglalásra, amit a naptár vezérlőn változtathatunk meg. Az „Időpont lefoglalása” gomb hatására a lefoglalt időpont bekerül az adatbázisba.

4.5.2 Sorszámok és időpont előfoglalások jegyzéke

A felhasználói felület egyik központi eleme azon táblázat, melyben az adott rendelés idejére foglalt, időponttal vagy sorszámmal rendelkező pácienseket jeleníti meg a program. A táblázat oszlopai olyan információkat tartalmaznak, mint a kiadás dátuma, név/sorszám, ütemezett időpont és státusz.

Kiadás dátuma	Név	Sorszám	Ütemezett időpont	Státusz
2012. 11. 05. 13:36:05		100	14:00:00	Kész!
2012. 11. 05. 13:36:06		101	14:05:00	Kész!
2012. 11. 05. 13:36:07		102	14:10:00	Kész!
2012. 11. 05. 13:41:42	Jane Doe		14:13:00	Behívva
2012. 11. 05. 13:36:08		103	14:15:00	Várakozik...
2012. 11. 05. 13:36:12		104	14:20:00	Várakozik...
2012. 11. 05. 13:41:58	Marcák Gábor Zoltán		14:22:00	Várakozik...
2012. 11. 05. 13:36:15		105	14:25:00	Várakozik...
2012. 11. 05. 13:42:33	John Doe		14:28:00	Várakozik...
2012. 11. 05. 13:36:17		106	14:30:00	Várakozik...
2012. 11. 05. 13:36:18		107	14:35:00	Várakozik...
2012. 11. 05. 13:57:12		108	14:40:00	Várakozik...

29. ábra: Az adott rendelés idejére foglalt időpontok és sorszámok táblázata

4.5.2.1 Ütemezett időpont

Az ütemezett időpontot a program automatikusan számolja ki. A felhasználónak be kell állítania egy általunk „sorszámok közötti időrésnek” nevezett tulajdonságot. Ez egy becsült érték, mely azt adja meg, hogy átlagosan mennyi időt töltenek bent a páciensek az orvosnál. Ha a rendelés például délután 14.00-kor kezdődik, akkor az első sorszámos beteget 14.00-ra fogja ütemezni a rendszer. Amennyiben a sorszámok közötti időrés 5 percre van beállítva, a következő sorszámos beteg 14.05-re lesz beütemezve. Az ütemezett időpont tehát támpontot, tájékoztatást nyújt a felhasználónak.

4.5.2.2 Státusz

Az oszlop segít az orvosnak vagy asszisztensnek nyomon követni a sorszámosztás és behívás aktuális állapotát, azaz hogy ki az aki már végzett, ki van éppen behívva és ki az aki még várakozik. A kijelzésre a könnyebb áttekinthetőség érdekében színekódokat vezettünk be. A betegbehívó gomb lenyomásával a státuszok automatikusan frissülnek. A gombbal történő behívás soros elven működik, azaz mindig a soron következő páciens hívható be vele. Lehetőség van a betegek soron kívüli vagy ismételt behívására. A funkciót a páciensek jegyzékénél ismertetett megoldáshoz hasonló menüből érhetjük el.

Kiadás dátuma	Név	Sorszám	Ütemezett időpont	Státusz	
2012. 11. 05. 13:36:05		100	14:00:00	Kész!	
2012. 11. 05. 13:36:06		101	14:05:00	Kész!	
2012. 11. 05. 13:36:07		102	14:10:00	Kész!	
2012. 11. 05. 13:41:42	Jane Doe		14:13:00	Kész!	
2012. 11. 05. 13:36:08		103	14:15:00	Kész!	
2012. 11. 05. 13:36:12		104	14:20:00	Kész!	
2012. 11. 05. 13:41:58	Marcák Gábor Zoltán		14:22:00	Behívva	
2012. 11. 05. 13:36:15		105	14:25:00	Várakozik...	
2012. 11. 05. 13:42:33	John Doe		14:28:00	Várakozik...	
2012. 11. 05. 13:36:17			14:30:00	Várakozik...	
2012. 11. 05. 13:36:18			14:35:00	Várakozik...	
2012. 11. 05. 13:57:12			14:40:00	Várakozik...	

30. ábra: Státusz jelzése a lefoglalt időpontok és sorszámok táblázatán

4.6 Beszédszintetizátor

Az eSpeak beszédszintetizátor egy nyílt forráskódú parancssoros program. Több különböző nyelvet támogat, köztük a magyart is. Használatához a parancssorban ki kell választani azt a könyvtárat, ahol az espeak.exe található, majd a fájlt a leírásban megadott paraméterekkel futtatva tetszőleges beszédhangot állíthatunk elő TTS (Text To Speech) konverzió által.

Lehetőség van azonban Java alkalmazáson belül is parancssoros programok futtatására. A platform ezen képességét kihasználva tudtuk beépíteni a beszédszintetizátort JavaFX programunkba. Először a Runtime objektumot kell példányosítanunk, majd az exec() függvény segítségével ugyanúgy tudunk programokat futtatni, mintha parancssorban dolgoznánk.

```
Runtime rt = Runtime.getRuntime();
try {
    Process p = Runtime.getRuntime().exec("d:\\Program Files\\eSpeak\\command_line\\espeak -vhu -s100 -g10 -a150 \"" + eredménySorszamLekerdezes.get(i).getSSorszam() + "\""); } catch
(IOException e1) {}
```

Az eSpeak beszédszintetizátor sokrétű paraméterezési lehetőségekkel rendelkezik. Beállítható például a beszélő hangja, a beszéd gyorsasága, hanglejtése és számos egyéb jellemző. Megfelelő beállítások mellett nagyon jó eredmények érhetők el, a beszéd teljes mértékben felismerhető. Természetesen az eSpeak nem képes olyan élethű beszédhang létrehozására, mint az előre felvett hangmintákból dolgozó szintetizátorok (pl. Google Speech Api). Leginkább egy frissen olvasni tudó gyermek felolvasásához hasonlítható a működés, tehát a szöveg nem teljesen folyamatos, folyékony, de minden esetben érthető.

4.7 Főmenü

A program rendelkezik egy hagyományos menüvel, ahol a felhasználó kiválaszthatja megtekintésre a fejlesztő névjegyét, a súgót, illetve átválthat az általános beállítások képernyőre.



31. ábra: A program főmenüje

4.8 Beállítások képernyő

A beállítások képernyőn van lehetőség felhasználó létrehozására illetve kiválasztására, a rendelési idő beállítására és a sorszámokhoz kapcsolódó paraméterek megadására. Itt lehet továbbá beállítani és tesztelni a csatlakoztatott eszközöket, valamint a beszédszintetizátort.



32. ábra: A program Beállítások képernyője

5 A rendszer tesztelése

A sorszámosztó és betegbehívó rendszert alapos laboratóriumi tesztelésnek vetettük alá. A felhasználás jellegéből adódóan alapvető elvárás a hibátlan működés és kiemelkedő üzembiztosság. Az egész napos, délelőtti és délutáni rendelést szimuláló teszt eredményei alapján elmondhatjuk, hogy a rendszer jelen állapotában megfelel a fenti elvárásoknak. A JavaFX vezérlőprogram megbízhatóan működött, kritikus hiba, nem várt leállás egyszer sem következett be. A hardverkomponensekkel való kommunikáció is zökkenőmentes volt. Az iparban is gyakran használt RS-232 szabvány a mi projektünk esetében is bizonyította megbízhatóságát, kiforrottságát. A PostgreSQL adatbázis szintén az elvárt szinten teljesített. Az adatbázis műveletek kivétel nélkül a másodperc tört része alatt zajlottak le. A hardverelemek működésében hibát nem tapasztaltunk, a nyomógombok, hőpapíros nyomtató és LED-kijelző megbízhatóan funkcionált.

A teszt során egy rövid áramszünetet is szimuláltunk, ezzel vizsgálva a rendszer viselkedését előre nem látható, fatális hiba esetén. Az áram visszakapcsolása után a személyi számítógépen futó Windows 7 operációs rendszer „A Windows súlyos hiba után állt helyre” rendszerüzenettel fogadott minket, a JavaFX vezérlőprogram azonban a megszokott módon indult, és azonnal végrehajtotta a szinkronizációt az adatbázissal. Ezután pontosan onnan tudtuk folytatni a munkát, ahol az áramszünet előtt megszakadt, így a rendszer ebben a tekintetben is megfelelően vizsgázott.

6 Miért jobb a mi rendszerünk a jelenleg forgalomban lévőknél?

Kereskedelmi forgalomban jelenleg nem kapható olyan sorszámosztó és ügyfélhívó rendszer, amit kifejezetten a háziorvosok és betegek igényeit szem előtt tartva fejlesztettek ki. Rendszerünk egyik legfontosabb jellemzője az akadálymentesített működés, mely lehetővé teszi használatát a fogyatékkal élő, csökkent látó, hallóképességgel rendelkező páciensek számára is. Beépítettük továbbá az időpont előfoglaláshoz szükséges modulokat, ami alapvető elvárás a háziiorvosi rendelők esetében. A rendszert egy kifejezetten érzékeny célcsoport számára terveztük meg. A mai viszonyok között sajnos a háziiorvosi rendelőknek alaposan meg kell fontolniuk, hogy milyen eszközökre ruháznak be.

Elterjedt és viszonylag olcsó hardverelemekből építjük fel a rendszert, mely egyfajta moduláris architektúrát követ. Könnyen bővíthető, testre szabható az adott igényekhez igazodva. A rendszer kiépítése minimális költséggel és munkaigénnyel megvalósítható.

Törekedtünk a lehető leggazdaságosabb üzemeltethetőségre. Az egyetlen fogyóeszköz a hőpapír tekercs, melynek pótlása egyszerű, és nagyobb mennyiségben igen kedvező áron beszerezhető. A fő rendszerkomponensek, legfőképp a kijelző áramfogyasztása kedvező, köszönhetően a korszerű LED technológiának. Esetleges hiba esetén a hardverelemek egyszerűen és gyorsan cserélhetők.

A jelenleg forgalomban lévő rendszerekhez viszonyítva a mi megoldásunk rendelkezik egy hatalmas előnnyel, ez pedig az összköltség. A piaci árak tanulmányozása után arra a következtetésre jutottunk, hogy a teljes sorszámozító és betegbehívó rendszerünk kiépítése nagyságrendileg ugyanannyiba kerül, mint más megoldások esetén egyetlen főkomponens, például a sorszámozító terminál, vagy kijelző megvásárlása. Következésképpen más rendszerek árának töredékéből tudjuk biztosítani ugyanazon funkcionalitást, kiegészítve az időpont előfoglalással, akadálymentesített működéssel és a magyar nyelvű beszédszintetizátorral. A rendszerkomponensek és telepítés összköltsége egy középkategóriás hordozható számítógép árának felel meg, mely véleményünk szerint megfontolandó alternatívát jelent a háziiorvosi rendelők részére.

7 Összefoglalás

Összefoglalásképpen elmondhatjuk, hogy a sorszámozító és betegbehívó rendszer tervezésekor kitűzött céljainkat teljes mértékben sikerült elérni. Kifejlesztettünk egy működő kísérleti berendezést, mely képes sorszámozításra, valamint a betegek koordinált behívására. Egyedülálló módon a páciensek számára az időpont előfoglalás opciója is biztosított. A behívás során audiovizuális interakciót alkalmazunk, amit a LED-kijelző és a magyar nyelvű beszédszintetizátor együttes alkalmazásával érünk el.

Rendelkezünk tehát egy laboratóriumi körülmények között működő prototípussal, mely a tervezett sorszámozító rendszertől elvárt alapfunkciókat képes ellátni. A célunk azonban az, hogy háziorvosok számára elérhetővé tegyük a végleges, minden tekintetben megbízhatóan funkcionáló rendszert, mely a piacon kiemelkedően költséghatékony, és az Európai Unió elvárásainak megfelelően teljes mértékben akadálymentesített lesz.

8 Irodalomjegyzék

-
- [1] S. TAYLOR: Waiting for Service: The Relationship between Delays and Evaluations of Service (Journal of Marketing Vol. 58, No. 2) 56-69 p.
- [2] Flip-disc display (Wikipedia, a szabad enciklopédia)
http://en.wikipedia.org/wiki/Flip-disc_display
- [3] T. DUTOIT: An Introduction to Text-to-Speech Synthesis (Springer, 2001, ISBN 978-1402003691)
- [4] J WEAVER: Pro JavaFX2 Platform (Apress, 2012, ISBN 978-1430268727)
- [5] Oracle Documentation: Introduction to FXML
http://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html
- [6] E. A. MEYER: Cascading Style Sheets: The Definitive Guide (O'Reilly Media, Inc., 2004, ISBN 978-0596005252)
- [7] G. SMITH: PostgreSQL 9.0 High Performance (Packt Publishing, 2010, ISBN 978-1849510301)
- [8] eSpeak: Speech Synthesizer
<http://espeak.sourceforge.net/>
- [9] RXTX
http://rxtx.qbang.org/wiki/index.php/Main_Page