

**PÉCSI TUDOMÁNYEGYETEM
MŰSZAKI ÉS INFORMATIKAI KAR
VILLAMOSMÉRNÖKI SZAK**

Stercz György

**Méhviasz sterilizáló berendezés
hőmérséklet szabályozó áramkör**

Pécs, 2017

Tartalomjegyzék

Köszönetnyilvánítás.....	3
1. Bevezetés.....	4
2. Irodalmi áttekintés.....	5
2.1. A méhviasz hőkezelése.....	5
2.2. Fuzzy elvű szabályozó.....	6
2.2.1. Fuzzy logika.....	6
2.2.2. A szabályozó felépítése.....	8
2.3. Váltakozó áramú szaggatók.....	9
2.4. Teljesítmény elektronikai félvezető kapcsolóelem - TRIAC.....	10
2.5. Impulzusszélesség-moduláció.....	11
2.6. Kommunikációs protokollok.....	11
2.6.1. Az I2C kommunikáció.....	11
2.6.2. Asszinkron soros kommunikációs protokollok.....	12
3. Tervezés, hardver és szoftver választás.....	14
3.1. A hőmérséklet szabályozó áramkör blokk diagramja.....	14
3.2. A fuzzy alapú hőmérsékletszabályozó megalkotásának lépései.....	15
3.3. Alkalmazható hőtermelő eszközök.....	20
3.4. Beágyazott rendszerek.....	20
3.4.1. Mikrovezérlők.....	20
3.4.2. A fejlesztői kártya.....	22
3.4.3. A ChibiOS.....	23
3.4.4. A µGFX.....	24
3.5. ADT7410 digitális hőmérséklet szenzor.....	25
4. Megvalósítás.....	26
4.1. A vezérlő egység.....	26
4.2. A teljesítmény szabályozó egység.....	28
4.3. A tápegység.....	30
4.4. Fejlesztői környezet.....	30
4.5. A méhviasz sterilizáló applikáció.....	32
4.5.1. Taszkok kialakítása.....	32
4.5.2. Pufferkezelés.....	34
4.5.3. Hőmérséklet mérés.....	35
4.5.4. Fuzzy elvű szabályozás.....	37
4.5.5. Sterilizálás.....	42
4.5.6. Fájlkezelés.....	44
4.5.7. Nyomtató kezelés.....	45
4.5.8. Hibakezelés.....	45
4.5.9. Grafikus felhasználói felület.....	46
4.5.10. A main függvény.....	48
5. A tesztberendezés és próbaüzemének eredményei.....	48
6. Összefoglalás.....	51
7. Irodalom jegyzék.....	52
8. Mellékletek.....	54
1. számú melléklet.....	54
2. számú melléklet.....	55
3. számú melléklet.....	55

Köszönnetnyilvánítás

Ezúton fejezem ki köszönemet mindenek előtt konzulensemnek, Zidarics Zoltán tanár úrnak, aki segítségével, instrukcióival hozzájárult a szakdolgozatom elkészüléséhez. Bevezetett a Linux világába, mely nagy segítséget nyújtott a munka során.

Továbbá köszönöm családomnak, hogy türelmesek voltak velem és mindenben a rendelkezésemre álltak. Kiváltképp testvéremnek, aki tartotta bennem végig lelket és igyekezett a segítségemre lenni.

Végül, de nem utolsó sorban, köszönöm mindenknak akik legkisebb mértékben is hozzájárultak hogy elkészülhetett a szakdolgozatom.

1. Bevezetés

A méhviasz a méhészeti egyik fontos alapanyaga. A méhek ebből építik lépjeiket, melyben nevelkedik a fiasítás, tárolják élelmüket a mézet és a virágport. A méhészeti e építő munkát úgy segítik, hogy a méhkaptár kereteibe műlépet helyeznek. A műlép egy méhviaszból készült vékony lap, melynek méhsejt mintás felületét hengereséssel készítik, így tartalmazza a méhsejtek alapját, a méheknek csak a falait kell felépíteniük. Használatával a lépek ellenállóbbak lesznek a mézpörgetés során előforduló kiszakadásokkal szemben, gyorsabban kiépülnek, csökkentik az építő munkához szükséges mézfogyasztást. Méhegészségügyi okokból kifolyólag egy keret élettartama maximum 3 év, leteltével cserélni szükséges. Mivel a méhviasz mesterségesen nem állítható elő, így kivett lépek beolvásztásával állítják elő a műlépet. A nyúlós költésrothadásnak nevezet méhbetegséget okozó baktérium spórái rendkívül ellenállóak, sterilizálatlan műlép használatával számuk feldúsulhat olyan mértékűre, mely a méhcsalád pusztulásához, akár járványhoz vezethet. Ennek megelőzése érdekében, fontos a méhviasz műlépgyártás előtti sterilizálása. Jómagam hobbiméhész lévén felkeltette az érdeklődésem egy olyan eszköz megvalósítása, mellyel otthoni körülmények között lehet steril méhviaszhoz jutni.

A méhviasz sterilizálása a kellő idejű hőhatás révén valósul meg, melynek nem tudnak ellenállni a kórokozók és spóráik, így e cél eléréshez egy hőmérséklet szabályozást szükséges megvalósítani. Dolgozatom célja egy olyan mikrovezérlő alapú több csatornás hőmérséklet szabályozó tesztberendezés elkészítése, mely alkalmas a sterilizálás feladatának ellátására és alkalmazkodva a korunk igényeihez érintőképernyős felhasználói interfésszel rendelkezik. A soros porton történő nyomtatás mellett, az eredmények digitális adathordozón történő tárolása is a megvalósítandó célok között szerepel, így nem csak a sterilizálás eredményei kerülhetnek archiválásra, hanem a teljes folyamatról képet lehet alkotni. Nem elérendő cél egy komplett végtermék kifejlesztése, így a szabályozást fuzzy logika segítségével kísérlem megvalósítani, mert alkalmazása esetén a rendszer matematikai modellezése helyett, nyelvi modellezés használható, így az esetleges későbbi fejlesztések során könnyen módosítható.

2. Irodalmi áttekintés

2.1. A méhviasz hőkezelése

A méhviasz hőkezelésére vonatkozó előírásokat a 70/2003. (VI. 27.) FVM rendelet I. fejezet 12. § 2. bekezdése határozza meg a következő módon: „*A műlépet előállító üzem köteles a viasz megbízható fertőtlenítése céljából olyan berendezést üzemeltetni, amely a viaszt egy órán át legalább 112 °C-os hőmérsékleten tartja. A hőfok és a hőhatás idejének megtartását kifogástalanul működő műszerekkel ellenőrizni kell. A viasz sterilizálására (fertőtlenítő hőkezelésére) szolgáló, viaszfertőtlenítő berendezést el kell látni automatikus hőfokszabályozó és hőmérséklet-regisztráló műszerrel, amely legalább öt percenként mér, és a mért értékeket az idő függvényében rögzíti, valamint ellenőrzéskor kinyomtatja. Az önirő műszerrel regisztrált adatokat három évig kell megőrizni. A viaszt, illetve a műlépet értékesíteni csak abban az esetben lehet, ha a viaszt minimum egy órán át, legalább 112 °C-on hőkezelték és ezt a hőmérséklet-regisztráló berendezés igazolja.*”[1]

A méhviasz hőkezelése során a rendeletnek való megfelelés mellett, figyelmet kell fordítani a méhviasz egyik minőségi jellemzője, a színének megtartására. Ezt befolyásolhatja a melegítésre használt edény anyaga, valamit méhviasz szennyezettsége. Egyes fémek érintkezve az olvadt méhviaszzal elszínezik azt, hatásuk annál erősebben jelentkezik, minél magasabb hőfokon és minél hosszabb ideig történik a hevítés. Használható alumínium-, rozsdamentes acél-, ónozott vagy zománcoszt lemezből készült edény. Az elszíneződés másik okozója a méhviaszban található szennyeződések, amik érintkezve a forró fémmel megpörköldnek, barnítják a viaszt. Emiatt a sonkoly feldolgozatlanul hőkezelésre nem alkalmas. A sonkoly a méhviasz nyersanyagának, a kaptár ból származó viastartalmú anyagok gyűjtőneve. A sterilizálás szükséges előkészítése, hogy a sonkolyt vízben, vagy valamilyen viaszolvasztó berendezésben (általában gőzviaszolvasztó) egybeolvassjuk majd a szennyeződést eltávolítjuk. A szennyeződés eltávolítása történhet ülepítéssel, szűréssel vagy kettő együttes alkalmazásával. A ülepítés során a vízben felolvasztott méhviaszt lassan hűtjük ki. A lassú megszilárdulás alatt a szennyeződés leülepszik, amit utána eltávolíthatunk. A szűrés alkalmazásakor felolvadt, folyékony viaszt leszűrjük. A viasz az összeolvasztás során vízzel emulziót alkot, bár általában a viaszban eloszló víz csupán 0,1-2,5%, de előfordulhat ennél magasabb érték is. Szélsőséges esetben az emulzió magas víztartalma miatt a méhviasz akár puha szivacsossá válhat. Ennek okozója az egybeolvastáskor történő erős kavargatás, vagy a gyors lehűlés. A lassú lehűtésnek nem csak a viasz kisebb víztartalma a kedvező

hatása, hanem a szennyeződés is jobban leülepszik és könnyebben eltávolítható. A sterilizálás folyamata során a méhviasz hőmérőkletét a víz forráspontja fölé kell emelni, aminek hatására a víz gőzzé alakulva távozik, ezért további odafigyelést igényel a biztonságos munkavégzés elérése érdekében. A távozó gőz miatt az olvadt viasz kiszaladhat az edényből, a. megrekedt gőz viaszt messze kilökheti. Az edényt lefedni csak olyan módon szabad, hogy a keletkező gőz eltávozhasson és megtölteni csak annyira, hogy az olvadt viasz ne érjen magasabbra 3/4-nél. Fontos megemlíteni, hogy a méhviasz könnyen és jól éghető anyag, emiatt biztosítani kell, hogy semmilyen gyűjtőforrással ne kerüljön kapcsolatba. Nyílt láng használata és dohányzás a hőkezelés során tilos! A bármilyen okból meggyulladt viasz oltására víz nem alkalmazható. Alkalmas porral, szén-dioxiddal oltó készülék, homok.[2]

2.2. Fuzzy elvű szabályozók

2.2.1. Fuzzy logika

A fuzzy szó jelentése elmosódott, pontatlan életlen, homályos, tehát a fuzzy logika az életlen halmazok logikája. A hagyományos halmazelméletben egy halmaz a következő módokon adható meg:

- Felsorolással, ha elemeinek száma véges.

$$A = \{1,2,3,4,5,6\}$$

- Az elemeire teljesülő szabály megadásával, tetszőleges elemszám esetén.

$$A = \{x \in X | x = n^2, n \in \mathbb{R}\}$$

- Karakterisztikus függvényével. X_A karakterisztikus függvény értéke 1 azon alaphalmazbeli elemekre, melyek elemei A halmaznak, egyébként 0.

$$X_A = \begin{cases} 1, & \text{ha } x \in A \\ 0, & \text{ha } x \notin A \end{cases}$$

A fuzzy logikában a karakterisztikus függvény úgy kerül általánosításra, hogy az, az alaphalmaz minden eleméhez egy értéket rendel hozzá valamilyen rögzített intervallumból (általában ez a $[0,1]$ intervallum.) A hozzárendelt érték nagysága a halmazbeli tagság mértékével arányos. Ezt a függvényt nevezzük tagsági függvénynek, az általa definiált halmazt pedig fuzzy vagy életlen halmaznak.[3;I.1]

„Legyen: X az x elemeknek vagy objektumoknak a halmaza ($x \in X$), tekintsük az igazságértékét egy életlen kifejezést $\mu_A(x)$ -et, mint az x elem A életlen halmazba

tartozásának

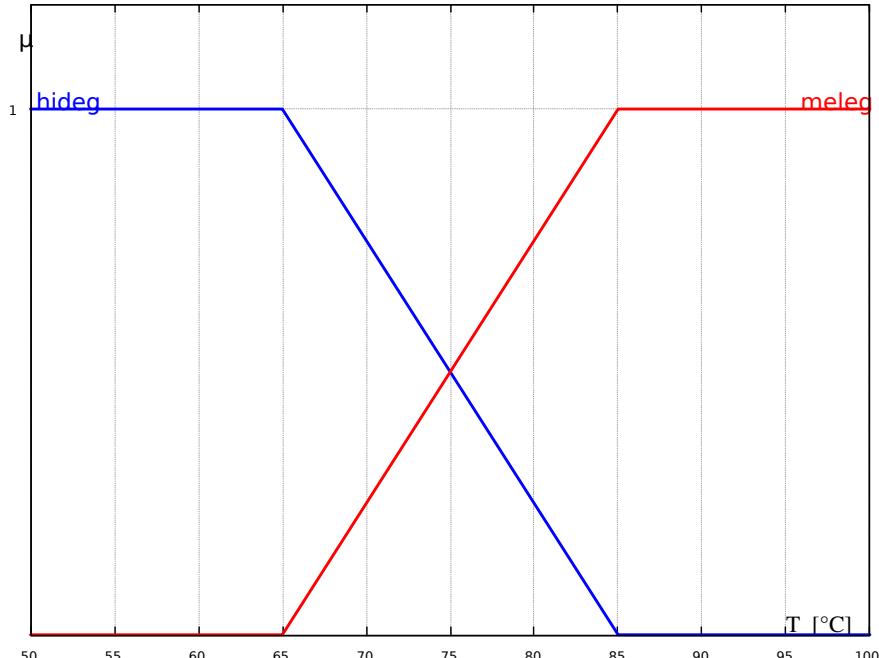
mértékét.

Az $(x, \mu_A(x))$ párok A halmaza: $A = \{(x, \mu_A(x)) | x \in X, \mu_A(x) \in \mathbb{R}\}$ életlen halmaz az X -en, $\mu_A(x)$ tagsági függvényvel. Az X halmazt az A életlen halmaz alaphalmazának vagy alaptartományának nevezzük.”[4]

Ez alapján a fuzzy halmazok lehetőséget biztosítanak a bizonytalan határokkal rendelkező nyelvi fogalmak ábrázolására. A hőmérséklet kapcsán ilyenek például a hideg és meleg fogalmak. Például a méhviasz hevítesi folyamata során nem egyértelműen megállapítható, hol van az a határ ami előtt azt mondhatjuk rá, hogy hideg és átlépése után hogy meleg. A két fogalmat definiálhatjuk egy-egy életlen halmazként:

- „hideg”: $H = \begin{cases} x, 1 & |x < 65 \\ x, \mu_A(x) & |65 \leq x \leq 85, \mu_A(x) = (85-x)/20 \\ x, 0 & |x > 85 \end{cases}$
- „meleg”: $M = \begin{cases} x, 0 & |x < 65 \\ x, \mu_A(x) & |65 \leq x \leq 85, \mu_A(x) = (x-65)/20 \\ x, 1 & |x > 85 \end{cases}$.

A halmazokat ábrázolva látható a két fogalom közötti folyamatos átmenet.(1. ábra)



1. Ábra: Hideg és meleg fuzzy halmazok

A hagyományos halmezelmélet alapműveletei, a komplementer képzés (negáció), metszet (konjunkció) és unió (diszjunkció) többféle módon is elvégezhető. Legelterjedtebben használatosak az alapvető fuzzy műveletek. Definiálásukhoz szükséges fogalom az életlen potencia halmaz.[I.1]

„Életlen potenciahalmaz: $P(X)$.

Az éles X alaphalmazon értelmezhető életlen halmazok összességét életlen potenciahalmaznak nevezzük, jele: $P(X)$

Komplementer képzés

Az $A \in P(X)$ életlen halmaz, (tagsági függvénye $\mu_A(x)$), komplementere A^C életlen halmaz, ha $\forall x \in X$ -re $\mu_{A^C}(x) = 1 - \mu_A(x)$.

Metszet és unió

Tekintsünk két életlen halmazt: $A, B \in P(X)$, ekkor $A \cap B$ életlen halmaz tagsági függvénye:

$$\forall x \in X \text{-re } \mu_{A \cap B}(x) = \min[\mu_A(x), \mu_B(x)] ,$$

$A \cup B$ életlen halmaz tagsági függvénye:

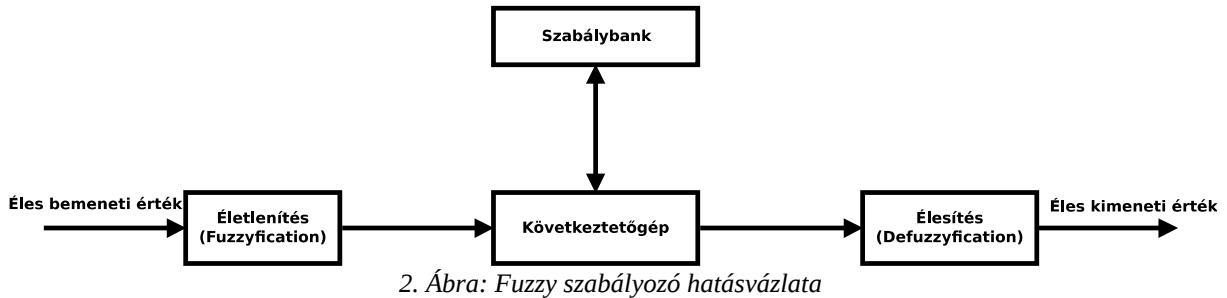
$$\forall x \in X \text{-re } \mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)] "[5]$$

Egy valóságos rendszer fizikális összefüggéseink feltárása és matematikai módszerekkel való leírása sok esetben bonyolult folyamat, de szavakkal könnyebben jellemzhető. A fuzzy logika segítségével egy rendszert leírhatunk jól megfogalmazott nyelvi kifejezésekkel, amelyek egy-egy életlen halmazként reprezentálják a fizikai mennyiségeket és a fuzzy műveletek segítségével közöttük kapcsolatokat hozhatunk létre. Ezáltal lehető válik a fuzzy elvű szabályozókban, hogy matematikai modellezés helyett nyelvi modellezést alkalmazzunk, ami növeli a rendszer áttekinthetőségét, elősegíti a hatékony módosítását és fejlesztését. A verbális vezérlési stratégiák által ez emberi tapasztalat bevhető szabályozási algoritmusba. A fuzzy elvű szabályozás eredményesen alkalmazható többparaméteres problémák, többértékű szabályozások, nagymértékű zavarok, erősen nemlineáris rendszerek esetén, valamint, ha a feladat matematikai modellje belátható időn belül nem meghatározható.[3]

2.2.2. A szabályozó felépítése

Amint a fuzzy szabályozó hatásvázlatán (2.ábra) látni lehet, a fuzzy szabályozó bemenetére az irányított folyamatra jellemző éles értékeket kapja, melyek a folyamat jellemzőinek mérésével állíthatóak elő. A szabálybázis az életlen rendszerleírást alkotó IF THEN szabályokból épül fel. Emiatt az első lépés a szabályozás során a bemenő jelek

életlenítése, fuzzyifikálása. A következtető gép az életlen bemeneti értékekhez a szabálybank segítségével életlen kimeneti értékeket rendel. Az irányított folyamathoz szükséges éles végrehajtó jeleket az életlen kimeneti értékek élesítésével, defuzzyifikálással állítja elő a szabályozó.[3]



Forrás: Szakonyi L. 2002 193. o.

2.3. Váltakozó áramú szaggatók

A hőkezelés során egy hőmérséklet szabályozást kell megvalósítani, aminek szükséges feltétele a fűtőtestek teljesítményének változtatása. A pillanatnyi teljesítményt leíró

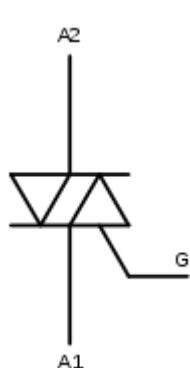
$$\text{összefüggés a következő: } P = \frac{U^2}{R} \text{ ahol:}$$

- P – fűtőtest hatásos teljesítménye;
- U – a fűtőtesten eső feszültség;
- R – a fűtőtest ohmos ellenállása.

Az összefüggést megvizsgálva 2 lehetséges megoldás van a teljesítmény megváltoztatására. Az egyik a fűtőtest ellenállásának megváltoztatása, ez csak a több fűtőellenállással rendelkező eszközök esetében lehetséges. A másik a feszültség megváltoztatása, aminek módszere az amplitúdószabályozás. Ez történhet változtatható értékű előtét ellenállással, változtatható légrésű, vagy változtatható egyenárammal előmágnesezett fojtó tekercssel, valamint szabályozó transzformátorral. Ezen módszerek mindegyikének nagy hátránya, hogy nem, vagy csak körülmenyesen automatizálható, így a feladat megoldásához nem használhatóak. Teljesítmény elektronikai félvezető eszközök segítségével megvalósíthatóak különféle váltakozó áramú szaggató áramkörök (AC Choppers). Ezek olyan elektronikus kapcsolók,

melyek segítségével váltakozó áramú körben az energiaáramlás megindítható és megszakítható. Ez történhet a egy fél periódus alatt egyszer, ekkor fázishasításos teljesítmény szabályozásról, ill. többször ebben az esetben vibrátoros teljesítmény szabályozásról beszélhetünk. Alkalmazásuk esetén a fogyasztón eső feszültség egy félperiódusra átlagolt értékkel jellemzhető. Előnyei hogy a kimeneti teljesítmény tetszőlegesen változtatható és minden periódusban történik áramvezetés, így gyors szabályozást tesz lehetővé. Hátrányuk a félperiódusok alatti egy vagy több kapcsolás miatt bekövetkező nagy áram és feszültség változások, melyek jelentős rádiófrekvenciás zavart okoznak. Ennek elkerülésére alkalmas a periódusszorzat-szabályozás. Ilyenkor a teljesítmény kapcsoló elem egész számú félperiódusnyi ideig van be és egész számú félperiódusnyi ideig van kikapcsolt állapotban, a kapcsolás mindenkor a nulla átmenetnél történik. A be- és kikapcsolt állapotok időtartamának változtatásával szabályozható a fogyasztó jutó átlagos teljesítmény. A módszer előnyei, hogy a fázishasítás miatti rádiófrekvenciás zavarok nem jelentkeznek, így a zavarszűrő egység megtakarítható, a kapcsoláshoz alkalmazott teljesítmény elektronikai félvezető igénybevétele csökken, bekapcsolásnál az áram csak lassan növekszik (kizárolag ohmos terhelés esetén, amikor az áram és feszültség nullpontja egybeesik). E szabályozási módszer használatakor a fogyasztón a tápfeszültség akár több periódusának idejéig is szünetelhet, így csak nagy időállandóval rendelkező folyamatok szabályozására alkalmazható. A méhviasz hevítése várhatóan nagy időállandóval fog rendelkezni, így ebben az esetben használható a periódusszorzat-szabályozás a fűtőtestek teljesítmény szabályozására.[6]

2.4. Teljesítmény elektronikai félvezető kapcsolóelem - TRIAC



3. Ábra: A triac kivezetései

A váltakozó áramú szaggatók megvalósításához legelterjedtebb a teljesítmény elektronikai félvezetők két típusát alkalmazzák a tirisztor és a triac. A tirisztor hátránya, hogy az áramot csak egy irányban képes vezetni, így váltakozó áramú alkalmazása esetén két tirisztor antiparalel bekötése szükséges valamint két vezérlőimpulzusról kell gondoskodni. A triac ezzel szemben az áramot mindkét irányban vezeti és elég egy vezérlőimpulzus használatához. Működése 4 üzemállapotban történhet a A₂ fő elektróda feszültségének és vezérlő impulzus polaritása függvényében, vonatkozási pontnak A₁ főelektródát tekintjük.[7]

Forrás:<https://hu.wikipedia.org/wiki/TRIAC>

Üzemállapot	U_{A2A1}	I_G
I.	Pozitív	Pozitív
II.	Pozitív	Negatív
III.	Negatív	Pozitív
IV.	Negatív	Negatív

1. Táblázat: A triac üzemállapotai

A triac begyújtása után csak az árama tartóáram alá való csökkentésével kapcsolható ki. Esetünkben ez a hálózati feszültség nullátmeneteinél teljesül, így minden félperiódusban eldönthető, hogy bekapcsoljon-e vagy sem. Használatával a fűtőtestek 1 másodpercre jutó átlagteljesítményét 0 és 100% között tudjuk változtatni 1%-os felbontással.[7]

2.5. Impulzuszélesség-moduláció

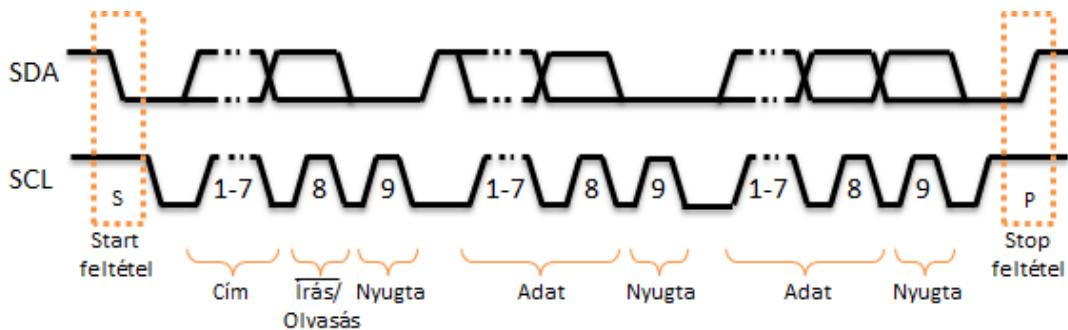
Az impulzuszélesség-moduláció (PWM – Pulse Width Modulation) a modulációs eljárások közül, az impulzusidő-modulációk csoportjába tartozik. Az impulzuszélesség-modulált jel esetén az impulzusok szélességét, azaz kitöltési tényezőjét (Duty Cycle) változtatjuk, állandó frekvencia és amplitúdó mellett. A kitöltési tényező $\gamma = \frac{t_{be}}{T} = \frac{t_{be}}{t_{be} + t_{ki}}$ [8]

2.6. Kommunikációs protokollok

2.6.1. Az I2C kommunikáció

Az Inter Integrated Circiut (I2C) protokoll a Philips cég által kifejlesztett cím alapú, mester-szolga (master-slave) szervezésű szinkronizált, soros, fél-duplex kommunikációs protokoll. Neve utal rá, hogy integrált áramkörök közötti adatcsere megvalósítására terveztek. Az I2C busz két vezetéket használ, egyet az órajel (SCA) és egyet az adatjel (SDA) számára. A buszon több és szolga is elhelyezkedhet, minden buszra csatlakozó eszköznek egyedi 7 vagy 10 bites címmel kell rendelkeznie. A kommunikációhoz szükséges órajelet a mester generálja és kommunikációt is csak ő kezdeményezhet. A busz inaktív állapotában minden magas szinten van. A buszon az adatforgalom módját a 4. ábra szemlélteti. A mester eszköz generál egy start bitet, azaz az SCL vonal magas állapotában generál egy magas-alacsony átmenet az SDA vonalon. A busz ekkor aktív állapotba kerül és lehetővé válik az adatcsere. Elsőként a mester küldi el a megszólítani kívánt eszköz címét, majd azt ezt követő bittel az

adatcsere irányát határozza meg (olvasás 1, írás 0). A byte vétele után a megcímzett eszköz felismerve saját címét egy nyugtázó jelet küld a sikeres vételről. Ez után következhet az írási vagy olvasási művelet. minden fogadott vagy elküldött byte után nyugtázás szükséges, ami úgy történik, hogy a szolga eszköz az SDA vonalat alacsony értékre húzza. A nyugtázás abban az esetben maradhat el, amikor a szolga eszköz az adó és a küldött byte sorozat végét az SDA vonal magas értéken tartásával jelzi a mester eszköznek, ezt hívjuk negatív nyugtázásnak (NACK). Az adatcsere végén a mester az SCL vonal magas állapotában generál egy alacsony-magas átmenetet az SDA vonalon, ekkor a busz szabaddá válik. A I2C többmesteres kialakítása miatt, szükséges annak kizárása, hogy egyszerre több mester is kommunikációt kezdeményezzen. Erre szolgál az arbitrációs folyamat. Abban az esetben, amikor egyszerre több mester kezdeményez kommunikációt, az a mester amelyik logikai 1 akar küldeni miközben a többek 0-át, elveszi az arbitrációt. Abban az esetben, ha a címbiteben nincs különbség az adatbiteken folytatódik tovább a folyamat, így nincs információ vesztés.[I.2;I.3]



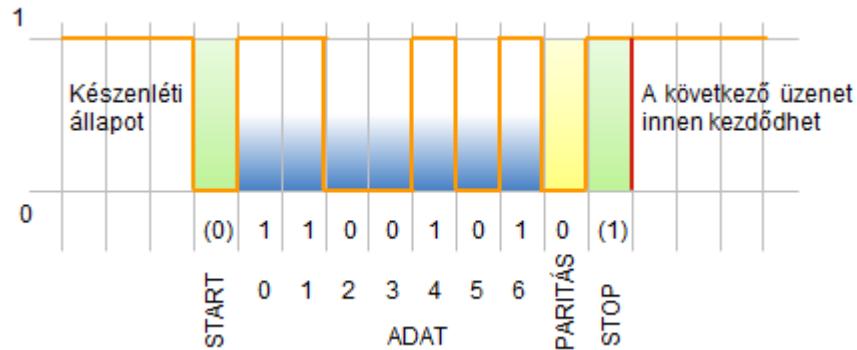
4. Ábra: I2C buszon történő adatküldés

Forrás: Ábra: http://moodle.autolab.unipannon.hu/Mecha_tananyag/autoipari_beagyazott_rendszerek/ch04.html#d0e705, 2017.05.06

2.6.2. Asszinkron soros kommunikációs protokollok

A aszinkron soros kommunikáció szinkron órajel nélkül adattovábbítást tesz lehetővé. Ebből kifolyólag alkalmazásához elég 3 vezeték, egy fogadó (RXD), egy küldő (TXD) és egy föld (GND) vezeték. Mivel órajelet nem használ, így a kommunikáció sebességét az egy bit időegységének hossza határozza meg. A szabványos értékek a következők: 150, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200 bit/s. Fontos, hogy az egymással kommunikációt folytató eszközök, azonos beállítást használjanak, a kommunikáció sebességét és üzenetformátumát tekintve egyaránt. Az adatvezetéken mikor nincs kommunikáció, minden logikai magas érték van. A kommunikáció a start bittel kezdődik, ami

egy logikai magas alacsony átmenet után egy bit ideig tartó logikai alacsony állapot. Ezt követik az adatbitek, melyek száma lehet 5, 6, 7, 8 és 9. Az adatbitek elküldésével a paritás bit következhet, melynek küldése beállítás függő, nem kötelező. Az üzenet a stop bit, vagy stop bitek záják, 1, 1,5, 2, majd ezt követően a vonal logikai magas értéken marad a következő start bitig. A definiált üzenet formátumot az alábbi ábra szemlélteti:

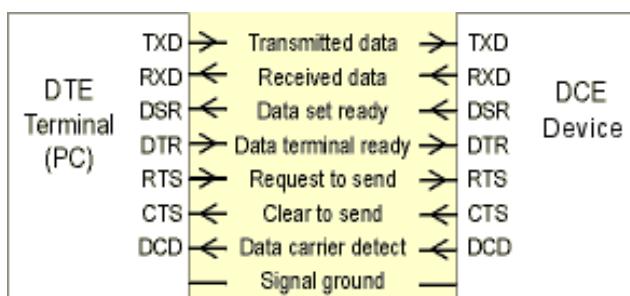


5. Ábra: UART és RS-232 üzenetformátum

Forrás: http://moodle.autolab.unipannon.hu/Mecha_tananyag/autoipari_beagyazott_rendszerek/ch04.html#d0e705

2017.05.12.

A mikrovezérlők körében népszerűen használ UART (Universal Asynchronous Receiver Transmitter – Univerzális Aszinkron Adóvevő) kommunikációs protokoll és például a nyomtatók esetében alkalmazott RS-232 kommunikációs szabvány is ezt az üzenetformátumot használja. A kettő közti különbség, hogy eltérő feszültség szinteket használnak, valamint az RS-232 további vonalakat is tartalmaz, melyekkel hardveres kézfogás valósítható meg. (6. ábra)[I.2][I.3]



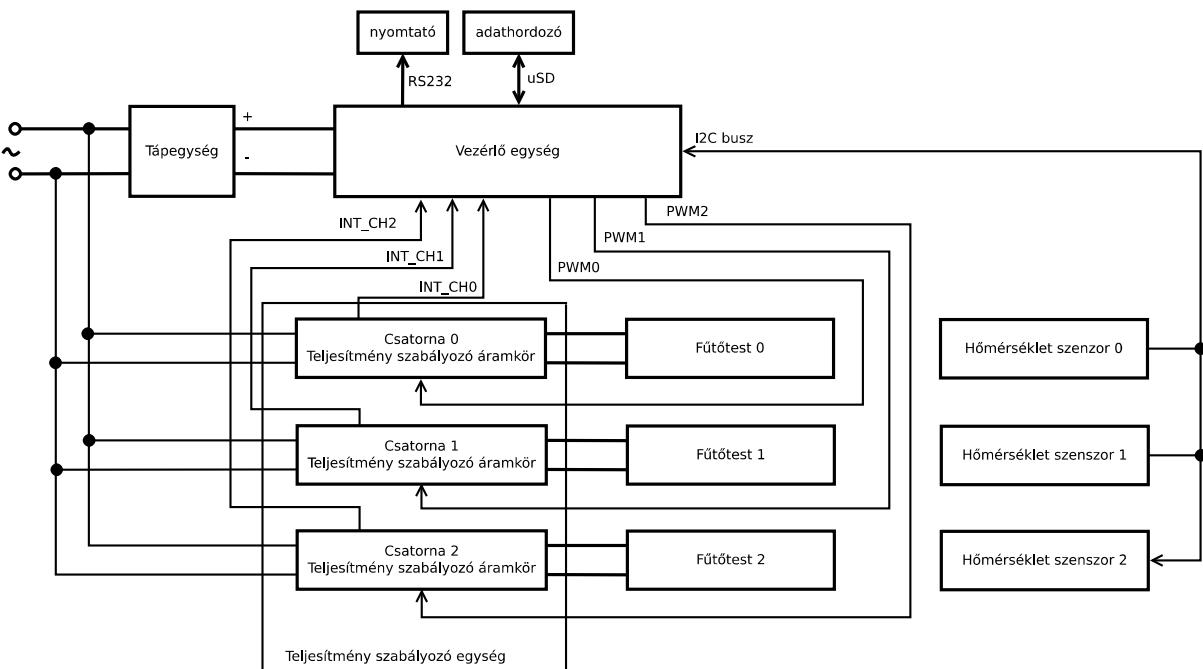
6. Ábra: RS-232 hardveres kézfogás

Forrás: <http://www.windmill.co.uk/handshaking.html>
2017.06.03.

3. Tervezés, hardver és szoftver választás

3.1. A hőmérséklet szabályozó áramkör blokk diagramja

A jogszabályban előírt hőkezelés megvalósításához egy olyan eszköz megalkotása szükséges, amely a méhviaszt a kívánt hőmérsékletre hevíti és afölött tartja az előírt ideig, ennek a tényét rögzíti az idő függvényében és képes kell legyen a mért adatok papír alapon történő archiválására. Továbbá rendelkezni kell valamilyen felhasználói felülettel, a kezelő személlyel történő kommunikációhoz. Érdemes az eszközt úgy kialakítani, hogy akár 3 csatorna kezelésére is képes legyen, mert így különböző teljesítmény igények és edényméretek kiszolgálására válik alkalmassá, valamint 1 és 3 fázisú hálózatról is üzemeltethető egyszerű módosításokkal. A teljes szabályozó áramkör a következő részegységekre osztható fel. Vezérlő egységre, teljesítmény szabályozó egységre és tápegységre.



7. Ábra: A szabályozó áramkör blokk diagramja

A vezérlő egység felelős a teljes folyamat irányításáért, célszerűen egy beágyazott rendszer. Periodikusan kiolvassa az egyes csatornákhöz tartozó digitális hőmérséklet szenzorokat, egy szabályozó algoritmus segítségével előállítja a teljesítményszabályozó egység bemeneti jeleit. Rendelkeznie kell RTC áramkörrel a hőmérséklet adatok archiváláshoz és kezelnie kell egy nyomtatót a rögzített eredmények kinyomtatásához. Előnyös, ha az eszköz képes valamilyen digitális adathordozón is rögzíteni az eredményeket,

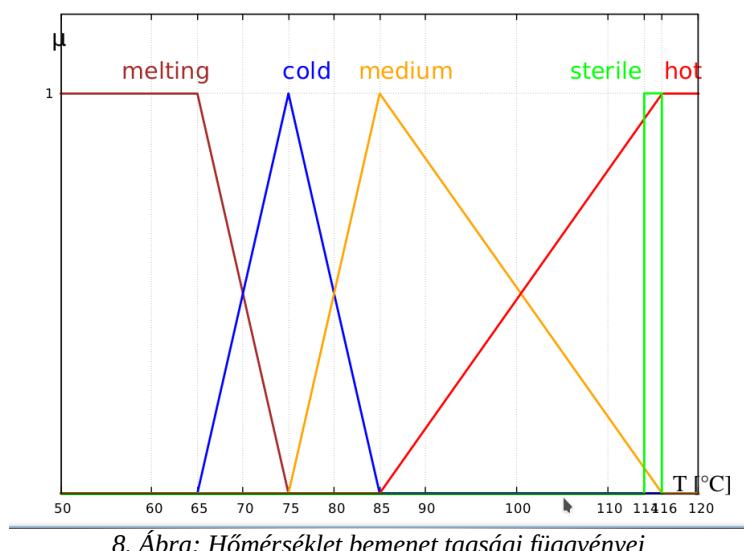
így azok digitális formában is eltárolhatóak, továbbá a fejlesztési folyamatot elősegítő logolás is megvalósíthatóvá segítségével. A vezérlő egység fontos feladata a HMI funkció megvalósítása. Ezen keresztül tud a felhasználó interakcióba lépni az eszközzel, elindítani, leállítani a hőkezelést és az eszköz ezen keresztül képes megjelenítheti információit a felhasználó számára (hőmérsékletek, a hőkezelés állapota, hibajelzések). Jól alkalmazható eszköz erre a feladatra egy érintő képernyős kijelző.

A szabályozó egység a vezérlő egységtől kapott jelek alapján szabályozza a fűtőtestek teljesítményét, a vezérlő egység által biztosított, csatoránkénti egy-egy bemeneten információt szolgáltat az olvadó biztosítók állapotáról, egy megszakítás generálásával.

A tápegység a hálózati váltakozó feszültségből egyenfeszültséget állít elő a vezérlő egység számára.

3.2. A fuzzy alapú hőmérsékletszabályozó megalkotásának lépései

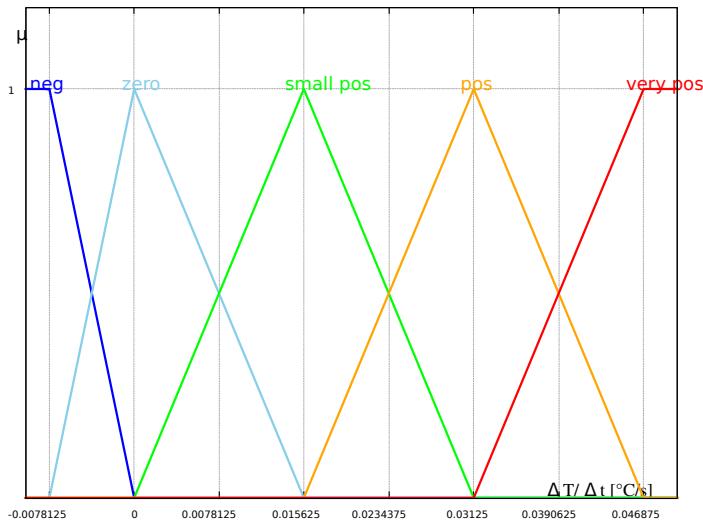
A szabályozó megalkotásának első lépése a be- és kimenő jelek, a hozzájuk tartozó nyelvi változók és terminusainak, valamint az egyes terminusokhoz tartozó tagsági függvények meghatározása. Az méhviasz hőkezelése során történő hőmérséklet szabályozás egyik bemenete a méhviasz hőmérséklete. Elérni kívánt cél, a szabályozó algoritmus adaptív viselkedése a különböző tömegű méhviasz mennyiségek hőkezelése esetén. Ennek érdekében a szabályozó másik a bemenete a hőmérséklet időegységre jutó változása. A fűtőtestek teljesítménye impulzusszélesség-modulált jel segítségével lesznek szabályozva, ezért a szabályozó kimenete e jelnek a kitöltési tényezője. A hőmérséklet bemenethez tartozó nyelvi változó legyen a Temp, terminusai: melting (olvadás), cold (hideg), medium (közepes), hot (forró) és sterile. A terminusokhoz tartozó fuzzy halmazok és ábrázolásuk:



8. Ábra: Hőmérséklet bemenet tagsági függvényei

A melting terminus a teljes hőmérséklet tartomány azon részét fedi le, amelyen a méhviasz teljesen, vagy döntő többségében még szilárd halmazállapotú. Amikor a hőmérséklet eléri a méhviasz olvadáspontját (kb. 63 °C) emelkedése megtorpan és elkezdődik az olvadási folyamat. A 85 °C elérve, feltehetően már a teljes olvadási folyamat lezajlott, nincsenek már az olvadt viaszban úszó szilárd darabok, a hőmérséklet ismét gyorsabb emelkedésnek indul. Az olvadási folyamatot lefedő terminus a cold. A medium terminus a hőmérséklet tartomány azon részét fedi le, amelyen a viasz teljesen vagy döntő többségében már folyékony halmazállapotú, de még nem érte el az állandósult állapotbeli hőmérsékletet. Az állandósult állapotbeli hőmérséklet meghatározáshoz figyelembe kell venni a korábban meghatározott, a szenzor mérési hibájával terhelt 114 °C-os hőmérsékletet és azt, hogy mekkora dinamikus ingadozás legyen engedélyezett a szabályozó számára. Ezt 2 °C-ban meghatározva az állandósult állapotbeli hőmérséklet 116 °C, ettől tekintjük a viasz forrónak, azaz a hot terminus itt éri el 1 igazságértéket. A hosszú átmenet a medium és hot szabály között, az állandósult állapotbeli hőmérséklet lassú megközelítését hivatott szolgálni, minimalizálva ezzel a legnagyobb túllendülés nagyságát. A sterile terminus a hőmérséklet dinamikus ingadozási tartományon belül tartását segíti.

A hőmérséklet változás bemenet nyelvi változója ΔT , terminusai: neg (negatív), zero (nulla), spos (small positive, kicsit pozitív), pos (pozitív), vpos (very positive, nagyon pozitív).



9. Ábra: Hőmérséklet változás bemenet tagsági függvényei

A ideális esetben, ha a veszteségektől eltekintünk a méhviasz belső energiájának megváltozása egyenlő a vele közölt hőmennyiséggel, $\Delta E_b = Q = P * t = c * m * \Delta T$ ahol:

- ΔE_b – a méhviasz belső energiájának megváltozása

- Q – a méhviaszal közölt hőmennyiséggel
- P – a fűtőtest teljesítménye
- t – az idő
- c – a méhviasz fajhője, 3,4 $\frac{kJ}{kg * ^\circ C}$ [I.4]
- m – a méhviasz tömege
- ΔT – a hőmérséklet változás

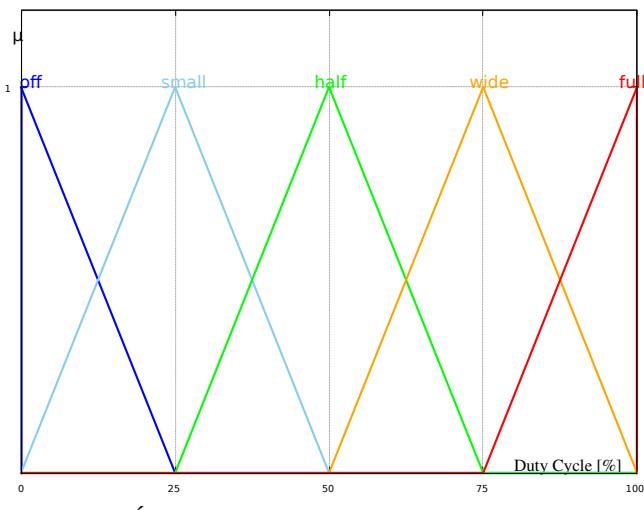
Az összefüggés alapján kiszámolható a méhviasz hőmérséklet változásának egy időegységre jutó elméleti maximum értéke. Ehhez a fűtőteljesítményt rögzítük 1,5 kW-ra, a hőkezelés során felhasználható méhviasz legkisebb tömegét 5 kg-ra. Ekkor

$$\frac{\Delta T}{\Delta t} = \frac{P}{c * m} = \frac{\Delta T}{1s} = \frac{1,5 \text{ kW}}{3,4 \frac{kJ}{kg * ^\circ C} * 5 \text{ kg}} = 0,088 \frac{^\circ C}{s}$$

értéket kapjuk a méhviasz 1 másodpercre

jutó legnagyobb hőmérsékletváltozására. Ezt az értéket a hőveszteség, az edényfalazat és a fűtőtestek hőtehetetlensége csökkenteni fogja, így a vpos terminológia legkisebb maximumhelye ennél alacsonyabb értékre került felvételre. Ez az érték és a 0 közötti tartomány a hőmérséklet szenzor 0,0078125 °C-os felbontásának egész számú többszöröseivel került felosztásra.

A szabályozó kimenete a PWM jel kitöltési tényezőjének értéke, nyelvi változója DC (Duty Cycle), terminusai: off (kikapcsolt), small (keskeny), half (fél), wide (széles), full (teljes).



10. Ábra: PWM kimenet tagsági függvényei

A tagsági függvények a 0 % és 100% közötti tartományt négy egyenlő részre felosztva kerültek meghatározásra.

A bemeneti változó és a hozzájuk tartozó tagsági függvények ismeretében a következő lépés a szabálybázis felállítása, mely segítségével a szabályozó az életlen bemeneti értékekhez életlen kimeneti értéket tud rendelni. Több bemenet esetén, a bemenetekhez tartozó életlen halmazok a NEM (NOT), ÉS (AND,) VAGY (OR) nyelvi összekapcsoló operátorok és a hozzájuk tartozó fuzzy halmazműveletek komplementer-, metszet-, és unióképzés segítségével kapcsolatba hozható. Így IF.....THEN szabályokkal megfogalmazható a szabályozó viselkedése. Jelen esetben a szabálybázis mátrix alakban a következő:

ΔTemp	neg	zero	spoz	pos	vpos
Temp					
melting	full	full	full	full	full
cold	full	full	wide	half	small
medium	full	wide	half	small	off
hot	off	off	off	off	off
sterile	wide	small	-	-	-

2. Táblázat: A fuzzy szabályozó szabályainak mátrixa

A szabálymátrix segítségével két szintaxisra illeszkedő szabálytípus állítható fel. Amikor a ΔTemp nyelvi változó terminusaitól független a kimenet, ekkor a szabály alakja IF (T) THEN (DC), egyébként IF (Temp AND ΔTemp) THEN DC. Ezzel a szabálymegfogalmazással összesen 14 szabály nyerhető ami befolyásolja a szabályozó viselkedését. Az egyes szabályok akkor válnak aktívvá, amikor a szabályhoz tartozó bemeneti terminusok vagy terminusok mindegyikének igazság értéke 0-tól különböző. A melting terminushoz egy szabály tartozik, így míg a viasz döntő többségében még szilárd halmazállapotú a fűtő teljesítmény maximális. A cold és medium terminusokhoz tartozó szabályok esetén az időegység alatti hőmérséklet változás függvényében kerül meghatározásra a kimeneti életlen érték. A hot terminushoz (melynek igazságértéke az állandósult állapotbeli hőmérséklet elérésével válik 1 értékűvé) tartozó szabály a fűtőteljesítményt 0-ra csökkenti. A sterile terminushoz csak két szabály tartozik. Szerepük csak abban van, hogy az állandósult állapotbeli hőmérsékletet elérve, a

fűtőteljesítmény kikapcsolása utáni visszahúlések esetén segítenek a hőmérsékletet a dinamikus ingadozási tartományon belül tartani.

A szabálybázis meglétével a szabályozó már elő tudja állítani az egyes aktív szabályokhoz tartozó kimeneti életlen értékeket. Ezen életlen értékekből a kimeneti éles érték előállítását nevezük defuzzyifikálásnak. Utolsó lépésben ennek a módszerét szükséges meghatározni. Real-time alkalmazások esetén a leggyakrabban alkalmazott a súlyozott maximum módszer, mely az aktív szabályok kimeneti maximumhelyének súlyozott átlaga, a súlyozó faktor az

$$\text{egyes szabályok aktivitási foka. N darab aktív szabály esetén } y_0 = \frac{\sum_{i=1}^n \beta_i * \bar{y}_i}{\sum_{i=1}^n \beta_i}, \text{ ahol:}$$

- y_0 – a kimenet éles értéke;
- β_i – az i-k szabály aktivitási foka;
- \bar{y}_i – a szabályhoz tartozó kimeneti terminus maximumhelye;
- n – az aktív szabályok száma.[3]

Az előző lépésekben felállított fuzzy szabályozó működési elve a következő:

1. A hőmérséklet éles értéke legyen 80 °C, a hőmérséklet időegységre jutó változása pedig 0,015625 °C/s. A fuzzyfikálás eredményeként megkapjuk hogy a két éles érték a nyelvi váltózók melyik terminusához mekkora igazság értékkal tartoznak. A hőmérséklet a cold terminusba és a medium terminusba is 0,5 igazságértékkal tartozik. A hőmérséklet változás a spoz terminusba t 1 igazságértékkel tartozik.
2. Ezen bemenetek esetén 2 szabály válik aktívvá. Az aktív szabályok igazságértéke a bemeneti terminusaikhoz tartozó igazságértékek metszete azaz minimuma.
 1. IF cold AND spoz THEN wide, igazság értéke: $\min[0,5, 1] = 0,5$;
 2. IF medium AND spoz THEN half, , igazság értéke: $\min[0,5, 1] = 0,5$.
3. A kimeneti maximumhely a wide terminus esetén 75 %, a half esetén 50%. A kimenet éles értéke a súlyozott maximum módszer szerint: $y_0 = \frac{0,5 * 75 + 0,5 * 50}{0,5 + 0,5} = 62,5\%$

3.3. Alkalmazható hőtermelő eszközök

A méhviasz hevítésére legkönnyebben és legegyszerűbben alkalmazható hőtermelő eszközök a csőfűtőtestek és főzőlapok. Mindkettő azonos elven működik, a rájuk kapcsolt feszültség hatására a fűtőellenállásukon áram folyik, ami hőt termel. Ezt a hőátadó felületükön keresztül a melegíteni kívánt anyagnak átadják. A csőfűtőtest előnyei a kis hőkapacitás és az, hogy az edénybe szerelve megtermelt hőjét közvetlenül a viasznak adhatja át. Hátrányaként a komplikáltabb beszerelése említhető meg, amihez a tároló edény falát át kell fúrni, így az itt keletkező tömítési elégtelenségek későbbi hibaforrásként jelentkezhetnek. A főzőlap alkalmazása egyszerűbb, viszont hátrányaként említhető meg a nagyobb hőkapacitása, valamint, hogy a hőt csak az edény falán keresztül közvetve adhatja át a viasznak.

3.4. Beágyazott rendszerek

Napjainkban egyre inkább népszerűvé válnak a különböző beágyazott rendszerek alkalmazásai. Ennek lehetőséget nyújt a gyártástechnológia rohamos fejlődése. Az egyre nagyobb mértékben elérhető miniatürizálással egyre nagyobb bonyolultságú áramköröket képesek a gyártók előállítani egy áramköri lapkán. Ezáltal növekszik a programozható digitális eszközök számítási teljesítménye, bővül a felhasználásuk lehetősége. Az irántuk táplált egyre növekvő igény megteremti nagyszámú sorozatgyártásuk lehetőségét, ami által az áruk csökken. Mind ezek mellett programozásukhoz szükséges szoftverek is egyre nagyobb számban elérhetőek. Mind hardver, (mikrovezérlők, FPGA-k, egy lapkás számítógépek, kijelzők, memóriák, érzékelők) mind szoftver (fejlesztői környezetek, beágyazott operációs rendszerek) tekintetében széles palettán lehet válogatni, így a beágyazott rendszerek az adott alkalmazáshoz könnyen testre szabhatóak, alkalmazásuk előnyös. A beágyazott rendszer egy adott cérla optimalizált hardver és szoftver kombinációja, mely képes autonóm vagy komplex alkalmazói berendezésekkel történő együttműködésre. Képesnek kell lennie környezetével folyamatosan kapcsolatban maradva érzékelők segítségével méréseket végezni, ha szükséges ezek alapján technológiai folyamatokba különböző aktuátorok segítségével beavatkozni, továbbá más rendszerekkel és/vagy felhasználókkal kommunikálni. [I.3]

3.4.1. Mikrovezérlők

A beágyazott rendszerek egyik fő alkotó eleme az alkalmazott programozható digitális eszköz, melynek egyik alternatívája a mikrovezérlő, ami nem más mint egy speciális mikroprocesszor. Különbség a kettő között abban rejlik, hogy a mikrovezérlő egy lapkára

integrálva tartalmazza a központi feldolgozó egységet(CPU), memóriát(RAM), háttértárat a rátöltött szoftver tárolására (FLASH RAM) és periférikus áramkörököt (I2C, SPI, SDRAM meghajtó áramkörök, A/D átalakító stb.) a külvilággal való kapcsolattartáshoz. Ebből fakadóan használatukhoz viszonylag kevés külső áramkör szükséges.



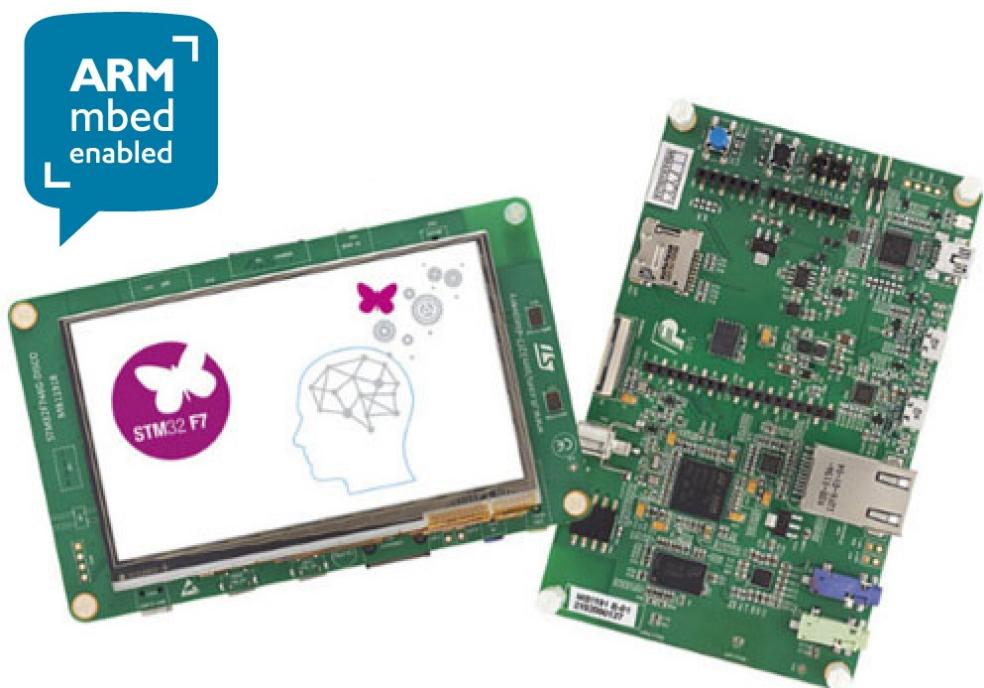
11. Ábra: Az stm32f746ng típusú mikrovezérlő áramköri diagramja

Forrás: Forrás:<http://www.st.com/en/microcontrollers/stm32f746ng.html>
2017.05.29.

A mikrovezérlő (11.ábra) áramköri diagramját megvizsgálva, megállapítható, hogy alkalmas a feladatra. Rendelkezik LCD kontroller, I²C, SDMMC (Secure Digital and MultiMediaCard), UART és SDRAM vezérlő periférikus áramkörökkel, továbbá legfőbb előnye, hogy gyártója fejlesztőkártyát kínál típusához.

3.4.2. A fejlesztői kártya

A mikrovezérlőket gyártó cégek jóvoltából különböző tudású fejlesztői kártyák érhetőek el a mikrovezérlőkhöz, melyek nagy mértékben megkönnyítik a fejlesztést. Ezen kártyák csatlakozó felületet biztosítanak a mikrovezérlő kivezetéseihez, valamint rendelkeznek programozó és hibakereső interfésszel. A felszerelt mikrovezérlő tudásához márten további eszközök is megtalálhatóak rajtuk, mint kijelző, memória, különböző csatalakozók (SD kártya foglalat, USB, Ethernet stb.). Jelen esetben a választott kártya az STM által gyártott STM32F746-DISCOVERY.



12. Ábra: STM32F746-DISCOVERY fejlesztői kártya

Tulajdonságai:

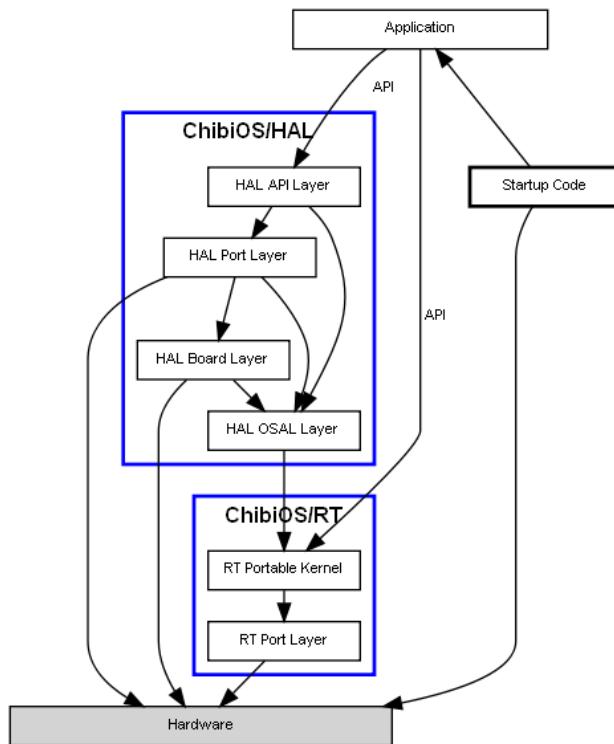
- A kártyán az STM32F746NG ARM Cortex-M7-es maggal rendelkező mikrovezérlő helyezkedik el, ami 1 Mbyte flashmemóriával és 340 kbyte RAM-mal rendelkezik.
- Egy USB-miniB csatlakozó keresztül elérhető az ST-LINK/V2-1 programozó és hibakereső áramkör, így a lefordított szoftverek feltöltéséhez csak egy megfelelő kábelrel kell rendelkezni.
- Felszerelték a kártyát egy Rocktech 4,3" méretű, 480x272 pixel felbontású színes, kapacitív érintő képernyős LCD-TFT kijelzővel.

- Rendelkezésünkre áll a kártyán egy a Micron által gyártott 128 Mbit-es SDRAM, amelyből 64 Mbit elérhető.
- Csatlakozó felületek tekintetében kielégíti a szükséges igényeket, biztosít két USB-microB csatlakozót, egy µSD kártya foglalatot, valamit a mikrovezérlő egyes lábai, a kártya hátsó oldalán, egy Arduino Uno V3 csatlakozón kerültek kivezetésre. Ennek raszter mérete 2,54 mm, így egy ilyen méretű tüskesor használatával elérhetőek.
- A kártya táp ellátása történhet 5V-ról mindhárom USB csatlakozóján keresztül, illetve külső tápegységről. Ennek kiválasztására egy jumper szolgál. Külső táplálás esetén két lehetőséget ad, egy 5 V tápegység számára szolgáló csatlakozási lehetőséget, valamit az Arduino csatlakozó felületen. Az utóbbi esetben 7-12 V-os tápfeszültség használható.[A.1]

3.4.3. A ChibiOS

A ChibiOS egy rétegelt felépítésű multitaszkos beágyazott operációs rendszer. Rétegei két nagy csoportra oszthatóak. A ChibiOS/RT ami az operációs rendszer valós idejű kernel része. Preemptív ütemezéssel rendelkezik, azaz minden programszál csak meghatározott ideig futhat, az idő lejárával másik szál kapja meg a futás jogát. A szálakhoz 128 prioritási szintet lehet társítani. Az ütemező e prioritás érték alapján állítja sorba a futásra várakozó programszálakat. Az azonos prioritási szinttel rendelkező szálak esetén Round Robin (körbeforgó) ütemezést használ. A kernel lehetőséget nyújt virtuális időzítők, bináris és számláló szemaforok, mutexek, kondíciós változók, esemény források és esemény flag-ek, üzenetek és mailboxok használatához. Alacsony erőforrás igényű, ami tovább csökkenthető a nem használt alrendszerek kikapcsolásával. A másik nagy csoport a ChibiOS/HAL, ami operációs rendszer hardver absztrakciós rétege. A modern mikrovezérlők leggyakrabban előforduló perifériáinak eszközvezérlőit tartalmazza. Az 13. ábrán a hardver és a szoftver kapcsolata látható a ChibiOS erősen rétegelt struktúráján keresztül. Nem javasolt az applikációból közvetlenül megszólítani a hardvert, mert az a hordozhatóság rovására megy. Köszönhetően a ChibiOS Port rétegeinek a hardver az applikáció módosítása nélkül kicserélhető. A ChibiOS, mint beágyazott operációs rendszer kiválasztása mellett, legfőképpen a következő szempontok szólnak. A ChibiOS egy nyílt forráskódú, szabad szoftver GPLv3 licencccel. Oktatási és otthoni célra ingyenesen használható. Legnagyobb részét C nyelven írták, így az általam legjobban ismert nyelven tudtam rá fejleszteni. Támogatja a kiválasztott STM32F746G Discovery Board fejlesztő kártyát, hardver

absztraktiós rétege tartalmazza a szükséges eszközmeghajtókat, többek között pl. I²C, EXT, GPT, PWM, RTC, SDC, SERIAL.[I.5;I.6;I.7]



13. Ábra: A ChibiOS rétegelt felépítése

Forrás <http://www.chibios.org/dokuwiki/doku.php?id=chibios:book:architecture> 2017.05.19.

3.4.4. A µGFX

A µGFX egy modulárisan felépülő grafikus keretrendszer. Különböző modulok segítségével lehetőséget nyújt, számos különböző feladat megoldására, így grafikus felhasználói interfések megvalósítására, érintőképernyők kezelésére, LCD kijelző vezérlése, felhasználói bementek olvasása vagy fájlok kezelése. A keretrendszer az alábbi modulokból épül fel:

- GDISP: egyszerű interfész szolgáltat kijelzők számára, függvények gazdag készletét tartalmazza egyszerű formák kirajzolásához, font rendereléshez és néhány egyéb funkcióhoz.
- GINPUT: interfész szolgáltat interaktív perifériák és érintőképernyők számára.
- GAUDIO: hangkezelésre szolgáló modul.

- GFILE: különböző fájlrendszerek ugyanazon API-n keresztül történő kezelésére.
- GEVENT: eseménykezelő modul a GWIN és GINPUT modulok számára.
- GTIMER: szoftveres időzítést tesz lehetővé.
- GQUEUE: különböző queue-kat és puffereket szolgáltat.
- GOS: absztrakciós réteg az operációs rendszer számára.
- GADC: közös API-t szolgáltat analóg interfések számára.
- GMISC: egyéb hasznos funkciókat tartalmazó modul..
- GWIN: komplett grafikus felhasználói felület készítésére alkalmas eszköztár, számos grafikus elemet biztosít (gombok, checkboxok, listák stb.).[I.8]

A μ GFX választásában a következő főbb szempontokat vettet figyelembe. Otthoni, hobbi és oktatási célokra ingyenesen használható, nyílt forráskódú szoftver. Támogatja az általam választott STM32F746G Discovery fejlesztőkártya használatát. Grafikus felületek a moduljai segítségével egyszerűen létrehozhatóak. A grafikus megjelenítés és az érintőlépernyő vezérlése egy keretrendszerben megoldható.

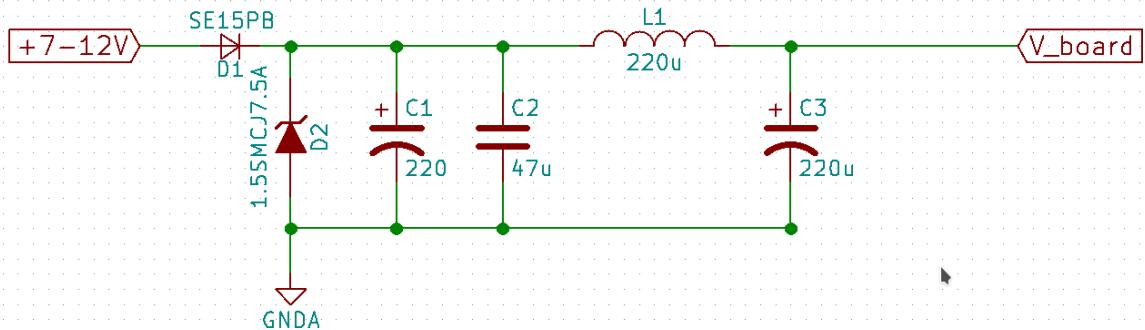
3.5. ADT7410 digitális hőmérséklet szenzor

A szenzor kiválasztásánál fontos szempont a méhviasz hőkezelése kapcsán előírt $112\text{ }^{\circ}\text{C}$ -os hőmérséklet. Ezt a szenzorok csak egy bizonyos hibával tudják kielégíteni, ami hozzáadódik az előírt hőmérséklethez. Ebből látszik, hogy a legtöbb maximum $125\text{ }^{\circ}\text{C}$ -ig működő szenzor nem alkalmas, mert a hőmérséklet szabályozás során keletkező esetleges túllendülések során tönkremehet. Az Analog Devices által gyártott ADT7410-es digitális hőmérséklet szenzor működési tartománya $-55\text{ }^{\circ}\text{C}$ és $150\text{ }^{\circ}\text{C}$ között van, így megfelelő választás a feladat megoldásához. A szenzor mérési hibája $+/-1\text{ }^{\circ}\text{C}$, 5 V-s tápfeszültség esetén. Annak érdekében, hogy ne a hibahatár közelében maradjunk, ennek a dupláját hozzáadva az előírt hőmérséklethez $114\text{ }^{\circ}\text{C}$ -ra szükséges hevíteni a viaszt. A szenzor működtethető 13 bites és 16 bites felbontással. Az előbbi $0,0625\text{ }^{\circ}\text{C}$ -os, az utóbbi $0,0078125\text{ }^{\circ}\text{C}$ -os hőmérséklet felbontást tesz lehetővé. Hőmérséklet konverziós ideje 240 ms. Kommunikálni I2C protokollon keresztül lehet vele, rendelkezik két címző lábbal, ami segítségével az I2C buszon használt címének utolsó két bitje állítható be, ezáltal 4 eszköz használható azonos buszon.[A.2]

4. Megvalósítás

4.1. A vezérlő egység

A vezérlő egység a fejlesztői kártyából és néhány kiegészítő áramkörből áll. A mikrovezérlő A kártya védelméért és a tápegység szűrését felelős áramkör:

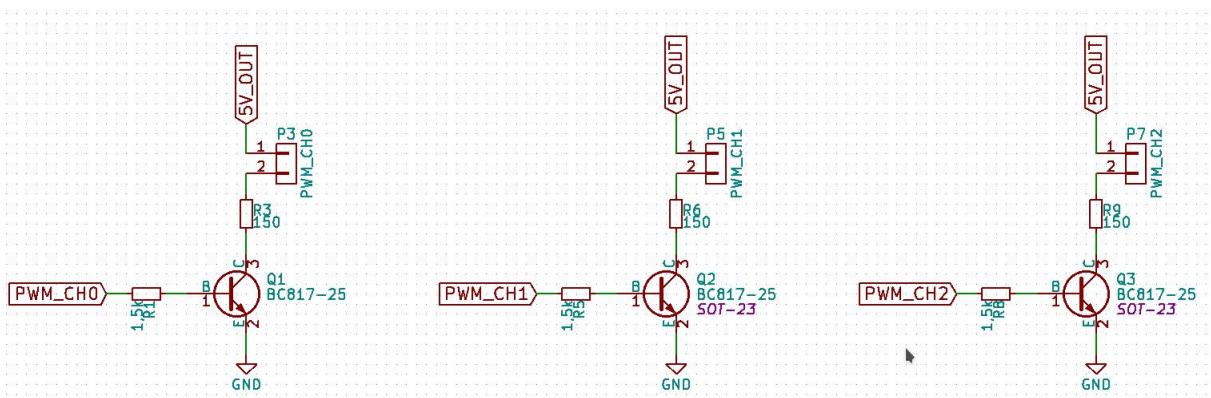


14. Ábra: Szűrő és védő áramkör kapcsolási rajz

A D1-es jelű dióda a tápfeszültség fordított polaritású bekötése, a D2 jelű 7,5 Voltos tranziens szupresszor dióda pedig a rövid ideig tartó túlfeszültség ellen nyújt védelmet. A két elektrolit kondenzátor a feszültség simítását végzi, valamint az áramkör tartalmaz egy LC szűrőt a tápegység felől érkező zajok kiszűrésére. Határfrekvenciája:

$$f_0 = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{220\mu H * 47\mu F}} = 1,56 \text{ kHz}$$

Az szabályozó egység csatornáihoz tartozó optotriakokat meghajtó áramkörök:



15. Ábra: A optotirák meghajtó áramkörök

A meghajtó áramkörök a P3, P5 és P7 jelű csatlakozókon keresztül kapcsolódnak a MOC3062 optotriak fotodiódáihoz. A tranzisztorok bázisaira csatlakoznak a mikrovezérlő PWM jelet szolgáltató lábai, melyeknek magas értéke esetén a tranzisztorok kinyitnak és a kollektorukra kapcsolt korlátozó ellenállásokon keresztül meghajtják a fotodiódákat. A méretezéshez szükséges adatok:

- $U_{BE}= 1,2V$ – a tranzisztor bázis-emitter feszültsége
- $U_{CE}=0,7V$ – a tranzisztor kollektor-emitter feszültsége
- $\beta=160$ – a tranzisztor egyenáramú erősítési tényezője
- $I_D=20mA$ – a fotodióda árama
- $U_D=1,2V$ - a fotodiódán eső feszültség
- $U_{pin}=3V$ – a mikrovezérlő kimenetének feszültsége magas állapotban[A.3][A.4]

A tranzisztor gyors nyitása érdekében legyen a bázisáram a szükséges 10-szerese:

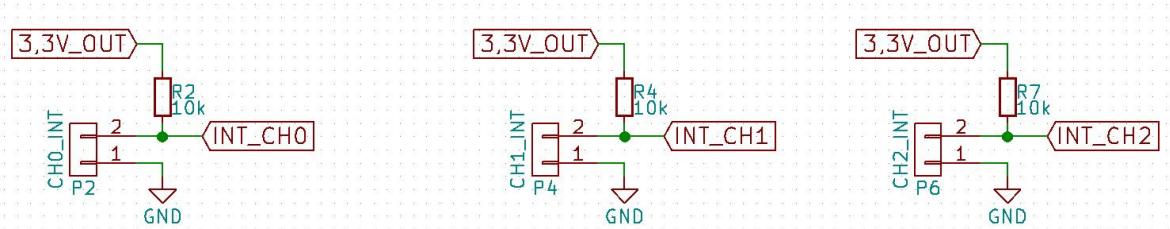
$$I_b = \frac{I_D}{\beta} * 10 = \frac{20mA}{160} * 10 = 1,25mA$$

A szükséges bázis ellenállás: $R_b = \frac{U_{pin} - U_{BE}}{I_{B0}} = \frac{3V - 1,2V}{1,25mA} = 1140\Omega \approx 1,5k\Omega$

Az kollektor áramkorlátozó ellenállás:

$$R_c = \frac{5V - U_d - U_{CE}}{I_D} = \frac{5V - 1,2V - 0,7V}{20mA} = 155\Omega \approx 150\Omega$$

A szabályozó egység reléihez kapcsolódó megszakítás bemenetek áramkörei:

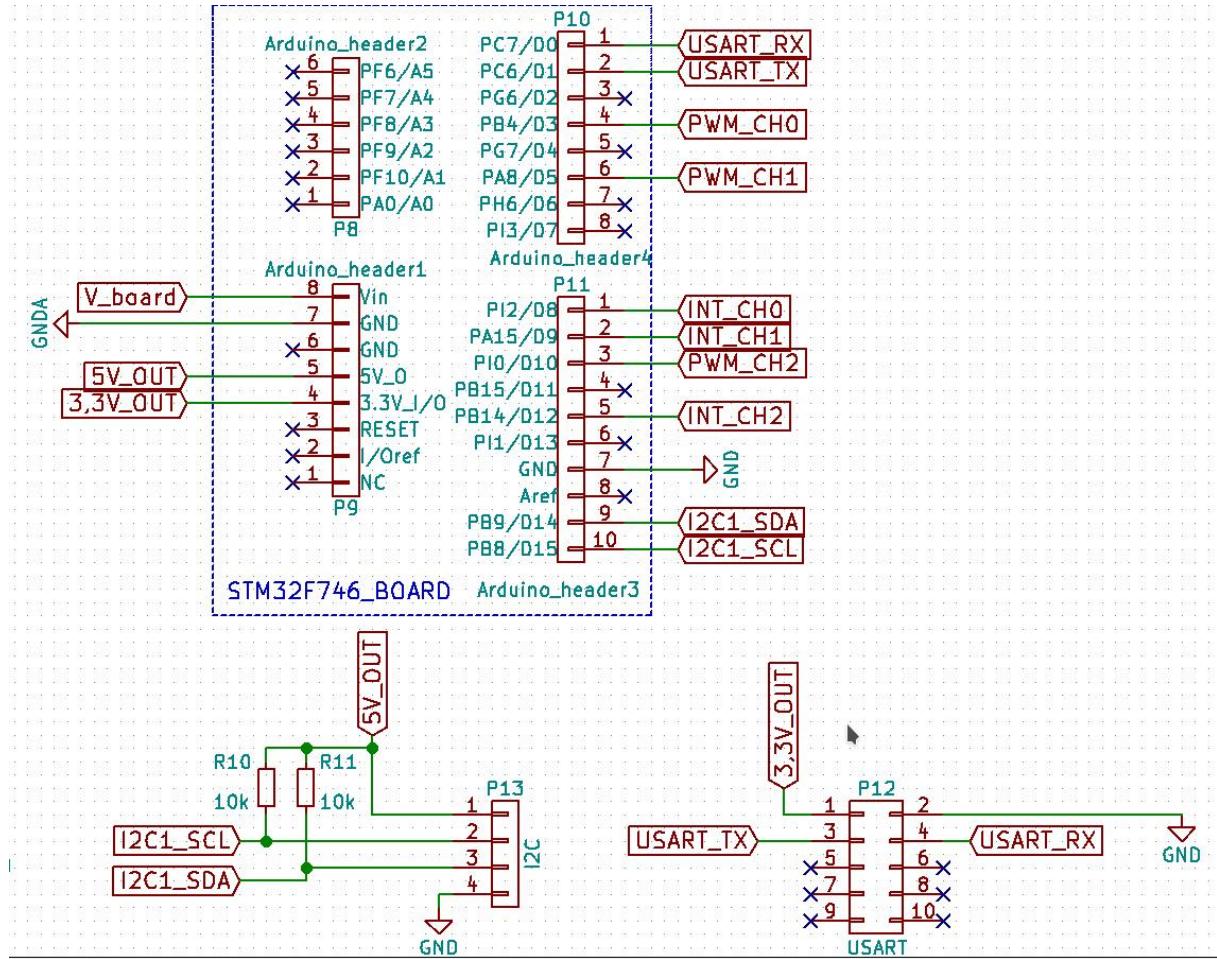


16. Ábra: Megszakítás bemenetek áramkörei

A P2, P4 és P6-os csatlakozón keresztül kapcsolódnak a relék érintkezői az áramkörbe. A relék meghúzott állapotában az érintkezők zárnak, így a megszakítás bemeneteket a földre

húzzák. a, elejtésükkor a bemenetek magas állapotba kerülnek, így egy felfutó él keletkezik, ami megszakítást generál. A relék pergésmentesítése szoftveresen kerül megoldásra.

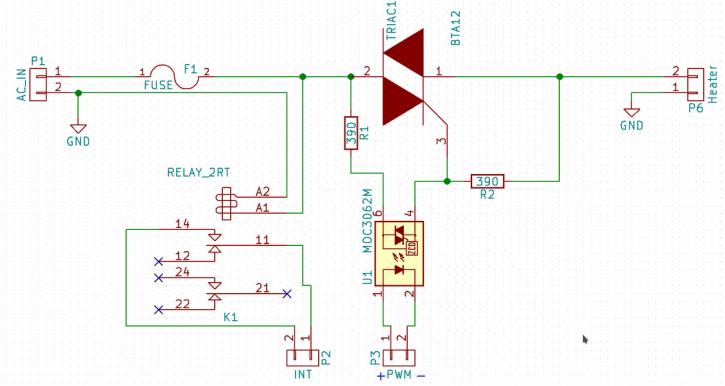
Az 17. ábrán a fejlesztői kártya csatlakozói valamint a soros porti és I²C csatlakozója látható.



17. Ábra: A kártya, az I²C és soros porti csatlakozók

4.2. A teljesítmény szabályozó egység

A fűtőtestek teljesítmény szabályozását csatornánként egy triac látja el, amely az optotriactól kapott vezérlő impulzus hatására gyújt be. A MOC3062-es típusú optotriac használatát indokolja, hogy rendelkezik null átmenet detektáló áramkörrel, így a periódusszorzat-szabályozás feltételeinek meg tud felelni az áramkör, valamit az optocsatolás révén a teljesítmény szabályozó egység galvanikus elválasztásra kerül a vezérlő egységtől. A fűtőtest és a triac-kal párhuzamosan kapcsolódik egy relé, mely az olvadó biztosíték kiolvadásakor elejt és az érintkezőinek bontásával szolgál információt erről a vezérlő áramkörnek. A kapcsolási rajza a 18. ábra.



18. Ábra: Teljesítmény szabályozó áramkör kapcsolási rajz

Fontos kérdés az áramkör működése során a triac hőmérséklete, hűtőborda igénye. Méretezése csatornánkénti 1 kW-os teljesítményre történt. A számításhoz szükséges adatok:

- $U_{be} = 230V$ – a hálózati feszültség effektív értéke;
- $U_T = 1,55 V$ -. a triacon vezető állapotban eső feszültség;
- $R_{thT}=2,3 \text{ }^{\circ}\text{C/W}$ – a triac és hűtőborda közötti termikus ellenállás;
- $T_k=25 \text{ }^{\circ}\text{C}$ – a környezet hőmérséklete;
- $T_{tmax}= 125 \text{ }^{\circ}\text{C}$ – a triac megengedett legnagyobb hőmérséklete.[A.5]

A triacon eső feszültség jelentősen kisebb, mint a fűtőtesten, így a rajtuk átfolyó áram kiszámításánál nem okoz nagy hibát, ha csak a fűtőtest teljesítményével történik a számolás:

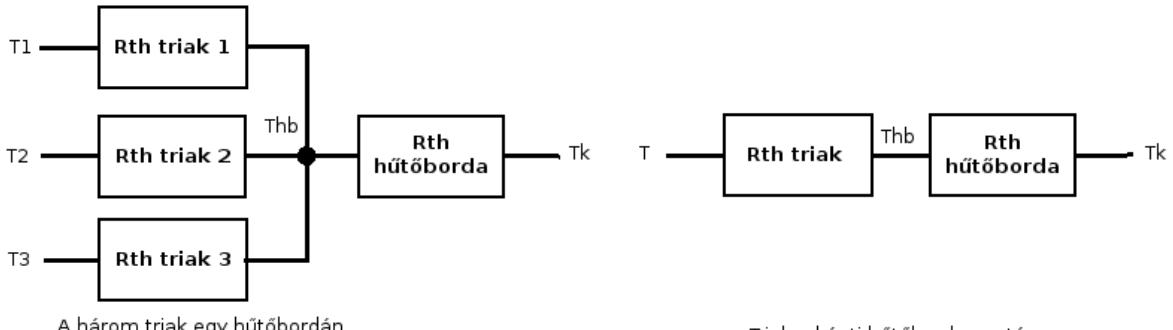
$$P_{fűtő} = U_{be} * I \rightarrow I = \frac{P_{fűtő}}{U_{be}} = \frac{1 \text{ kW}}{230 \text{ V}} = 4,35 \text{ A}$$

Ez alapján a triac disszipációs teljesítménye: $P_{DT} = U_T * I = 1,55 \text{ V} * 4,35 \text{ A} = 6,74 \text{ W}$

Ennek birtokában meghatározható, hogy a triac és a hűtőborda között mekkora hőmérséklet

$$\text{különbség alakul ki: } R_{thT} = \frac{\Delta T}{P_{DT}} \rightarrow \Delta T = R_{thT} * P_{DT} = 6,74 \text{ W} * 2,3 \text{ }^{\circ}\text{C/W} = 15,5 \text{ }^{\circ}\text{C}$$

Ebből következik hogy a triac működési hőmérsékletét $T_k + \Delta T < T_T < T_{tmax}$, azaz $40, 5 \text{ }^{\circ}\text{C}$ és $125 \text{ }^{\circ}\text{C}$ tartományon belül kell meghatározni. Legyen a $T_T=75 \text{ }^{\circ}\text{C}$. A szükséges hűtőborda meghatározásához szolgáló termikus modell (19. ábra).



19. Ábra: Triacok termikus modelljei

A hűtőborda maximális hőmérséklete: $T_{hb} = T_{max} - \Delta T = 75^\circ C - 15,5^\circ C = 59,5^\circ C$

A szükséges hűtőborda, ha minden három csatorna triacija rákerül:

$$R_{thhb} = \frac{T_{hb} - T_k}{P_{DT1} + P_{DT2} + P_{DT3}} = \frac{59,5^\circ C - 25^\circ C}{6,74 W + 6,74 W + 6,74 W} = 1,7^\circ C/W \quad \text{és abban az esetben, ha minden három külön hűtőbordán helyezkedik el:}$$

$$R_{thhb} = \frac{T_{hb} - T_k}{P_{DT}} = \frac{59,5^\circ C - 25^\circ C}{6,74 W} = 5,11^\circ C/W$$

4.3. A tápegység

Tápegység külön nem készült az áramkörhöz, hanem a Vigotronix által gyártott VTX-214-010-107 típusú kapcsoló üzemű tápegységet használtam, melynek jellemzői:

- Bemeneti feszültség: 90-264 VAC.
- Kimeneti feszültség: 7,5 V.
- Teljesítmény 10W.
- A bemenet a kimenettől galvanikusan elválasztott.
- Rövidzár, túlfeszültség és túlterhelés elleni védelemmel ellátott.[A.6]

4.4. Fejlesztői környezet

A szoftver megírása Debian alapú Linux operációs rendszeren történt, így ebben a fejezetben az ehhez szükséges eszközök kerülnek bemutatásra. Egy fejlesztői kártya használatakor, ez első felmerülő probléma, közte és a PC közötti kapcsolatteremtés megoldása. A kiválasztott kártya esetében a programozó és debugger áramkör a kártyán

megvalósított, így egy USB kábelre és egy PC-n futó szoftverre van szükség. Erre a cérla használható az *stlink* nevű szoftver, mely a *git* verziókezelővel letölthető és forrásból telepíthető. Debianon a forrás fájlok számára fenntartott hely a /usr/src mappa, így ide érdemes letölteni. A mappa tartalmának módosításához rendszergazdai jogosultsággal kell rendelkeznünk. Az *stlink* telepítéséhez szükséges a *Cmake* (*minimális verzió v2.8.7*), *biuld-essential* és a *Libusb 1.0-ás* (*minimális verzió v1.0.9*) csomagokkal. Az *stlink* letöltése valamint a szükséges programok telepítése a következő parancsokkal lehetséges:

```
#apt-get install git cmake biuld-essential libusb-1.0-0-dev  
#cd /usr/src  
#git clone https://github.com/texane/stlink  
#cd stlink  
#make release  
#cd biuld/Release  
#make install  
#ldconfig
```

A program működéséhez szükséges még a *stlink* mappájában lévő /etc/udev/rules.d mappa alatt található fájlok a saját /etc/udev/rules.d mappánkba másolni, majd ezt követően kiadni a következő két parancsot:

```
#udevadm control --reload-rules  
#udevadm trigger
```

Ezzel a megvan a szoftver telepítése, használatra kész. A telepítésről további információk érhetőek el <https://github.com/texane/stlink> oldalon.[I.9] A szoftver C nyelven kerül elkészítésre, amihez egy arm alapú mikrovezérlőkhöz használható fordító program a szükséges, amit a gcc biztosít. A kód szerkesztésére a *CodeBlocks*-ot használtam. Mindkettő tárolóból elérhető, telepítésük:

```
#apt-get install gcc-arm-none-eabi codeblocks
```

A megírt szoftver lefordítása a *make* fordításvezérlővel történik, ami külön telepítést nem igényel. Az applikáció létrehozásához kialakításra került egy mappa, *stm32f746* néven a következő tartalommal:

- /templates mappa. Ebben találhatóak a projekt létrehozásához szükséges mintafájlok;
- egy *create* script és a róla készült két soft link *createapp* és *createtask* néven;
- a *Makefile* számára a szükséges beállításokat tartalmazó *settings.mk*, az elérhető célokat tartalmazó *targets.mk* és az őket kilistázó *help.mk*.

A projekt létrehozása az stm32f746 mappából történik. A létrehozáshoz szükséges parancs:

```
$./createapp <appname>
```

Hatására létrejön a appname nevű mappa. Felépítése:

- doc mappa a *doxygen*nel készíthető dokumentáció számára;
- include mappa a header állományok számára;
- src mappa a források számára;
- Makefile, appname.cbp projekt fájl a *CodeBlocks* számára, valamint a chconf.h, halconf.h és mcuconf.h a ChibiOS konfigurációs fájljai.

Az applikáció mappájában kiadva, új taszk hozzáadására, az applikáció lefordítására és a mikrovezérlőre feltöltésére használható parancsok:

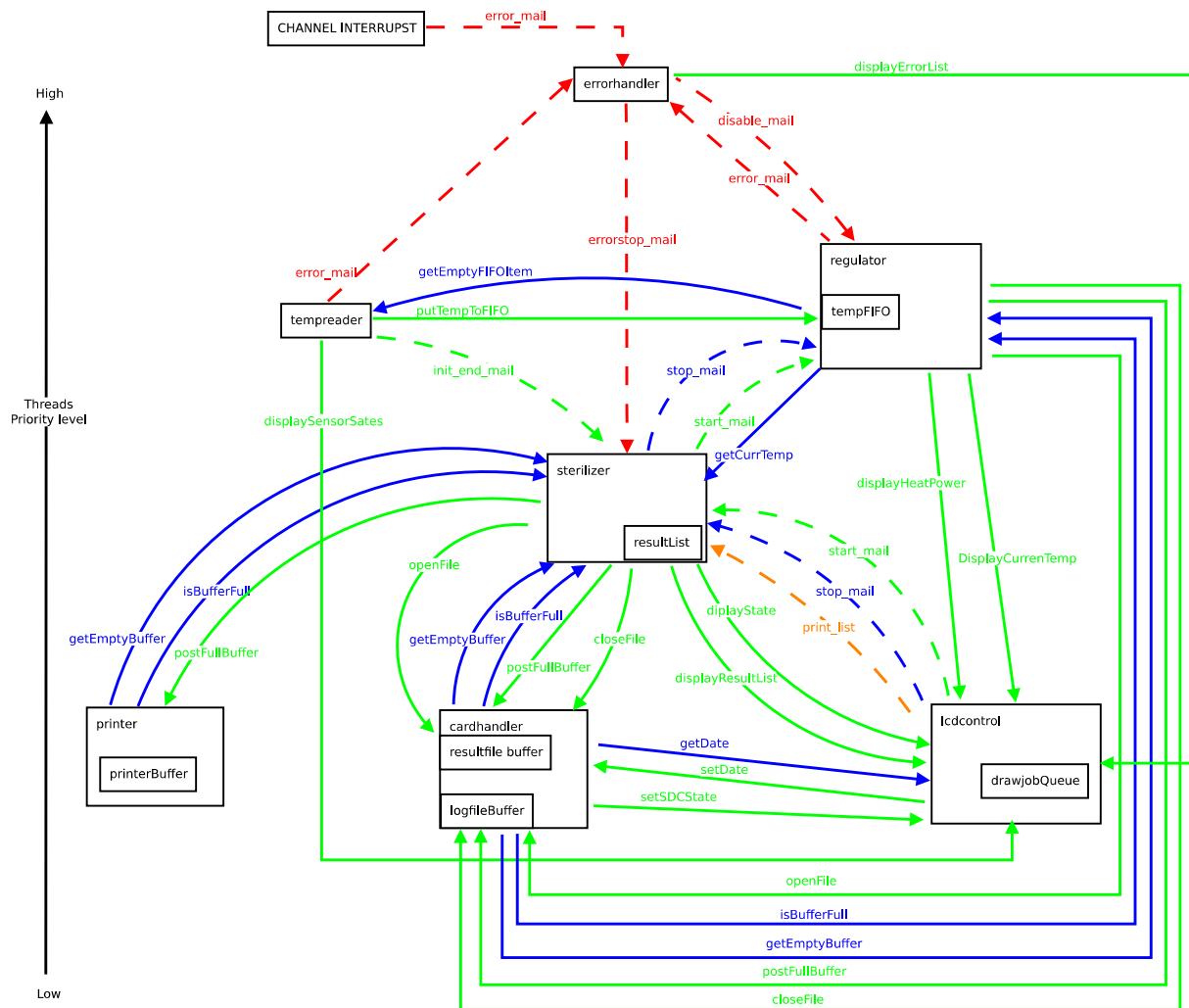
```
$make createtask  
$make  
$make flash
```

4.5. A méhviasz sterilizáló applikáció

4.5.1. Taszkok kialakítása

Mivel a ChibiOS egy multiaszkos operációs rendszer, így lehetőséget biztosít több programszál alkalmazására. Egy applikáció megvalósítása során lényeges kérdés a programszálak száma. Sok programszál alkalmazása növeli a szoftver modularitását, így elősegíti a kód újrahasználhatóságát, ugyanakkor több kontextus váltást eredményez, a szálak közötti kommunikáció is bonyolultabb, ami több rendszerhívással jár, valamint a szoftver memória igénye is növekszik. Emiatt szálak kialakításánál törekedni a kell a feladat megoldásához szükséges legkevesebb programszál kialakítására. További szempontok még a be- és kimenetek (pl.:I²C busz, soros port) és a több szál által megosztottan használt hardver elemek (pl.: kijelző, memóriakártya) kezelése, mely feladatok önálló programszálat igényelnek. Ezen szempontok figyelembe vételével a méhviasz sterilizáló applikáció esetén az önálló szálat igénylő feladatok: hőmérésklet szenzorok kezelése, PWM kimenetek kezelése, a sterilizáció folyamatának kezelése, fájl kezelés, nyomtató kezelés, hibakezelés és kijelző kezelés. A hőmérésklet olvasó szál kiolvassa az egyes csatornákhoz tartozó szenzorokat, majd átlagoló algoritmus segítségével előállítja az aktuális hőmérésklet értékeit. Ezt közli a

szabályozó szállal, ami kapott hőmérséklet értékek alapján meghatározza a PWM kimenetek kitöltési tényezőjét. A sterilizáló szál a szükséges időközönként (a jogszabály szerint legalább 5 percenként) lekérdezi a szabályozó száltól az aktuális hőmérséklet értékeit és gondoskodik a hőkezelés szükséges idejének (legalább 1 óra) megtartásáról. Ennek igazolására a lekérdezett hőmérséklet adatokból eredmény listát készít. Az elkészült listát egy fájl formájában a SD kártyára menti a kártyakezelő szál által, valamint kinyomtatja a nyomtatót kezelő szál segítségével. Célszerű kialakítani egy hibakezelő szálat, így a szálak a futásuk során detektált hiba esetén értesítik a hibakezelőt annak tényéről aki ennek hatására leállítja a hőkezelés folyamatát és a keletkező hibákat egy listába gyűjtö össze. A kijelző kezelő szál megjeleníti a szálak által közöl adatokat és kezeli az érintő panelt.



20. Ábra: Taszkok és egymással történő kommunikációjuk

A ki és bemeneti adatok szálak közötti cseréjét célszerű puffereken keresztül megvalósítani. Így szükséges egy puffer a beolvasott hőmérsékleteknek, a kezelt fájloknak, a kinyomtatni kívánt adatoknak. minden szál külön forrás fájllal rendelkezik melyben adatai

statikusan deklaráltak, így azok rejte vannak a többi szál elől, hozzáférést mutex védelme alatt, függvények segítségével engedélyeznek.

4.5.2. Pufferkezelés

Az előző fejezetben látható volt, hogy az applikáció megvalósításához több adatpufferre is szükség van. Ehhez érdemes kialakítani egy olyan pufferkezelő modult amely a szálak közötti belső adatcserét szolgálja. Elvárás vele szemben, hogy a puffer mérete, valamint elemeinek mérete tetszőlegesen meghatározható legyen és a puffer memória területe már a fordítási időben kerüljön lefoglalásra, így megelőzve a memória fragmentálódást. A ChibiOS ehhez eszközt biztosít a Memory Pools kernel moduljával. A memória pool (memória medence) elve az, hogy létrehozása során egyszer kerül lefoglalásra egy összefüggő memória terület a pool elemeinek számára. Használatkor a pool egyes elemeinek mutatója kerül kiadásra, használat után ezek a mutatók kerülnek visszavételre, így újra kiadhatóvá válnak. Ezáltal nincs szükség folyamatos memória foglalásra és felszabadításra. A memória pool létrehozásához szükség van egy tetszőleges típusú tömbre a pool elemeinek számára és egy memória pool leíróra, ami a *MEMORYPOOL_DECL(name, size, provider)* makróval tehető meg. Argumentumai a *name*, a leíró neve, *size* az elemeinek mérete, *provider* a pool elmeinek memória foglalásáért felelős függvény mutatója. A pufferkezelő esetében a deklarált tömb elemeinek mérete (*size_t* egységen) lesz a pool elemeinek mérete és a *provider* mutató mindenig NULL, mert a tömb deklarálásakor fordító által lefoglalásra kerül a memóriaterület számára, így nincs szükség további memória allokációra a szoftver futása során. A puffer kezelése az így kialakított poolhoz deklarált *inner_buffer_t* típusú változón keresztül valósul meg. A szükséges deklarációk általános alakban a következők:

```
static pool_item_type items[BUFFER_SIZE];
static MEMORYPOOL_DECL(poolname, sizeof(pool_item_type), NULL);
static inner_buffer_t buffer;
```

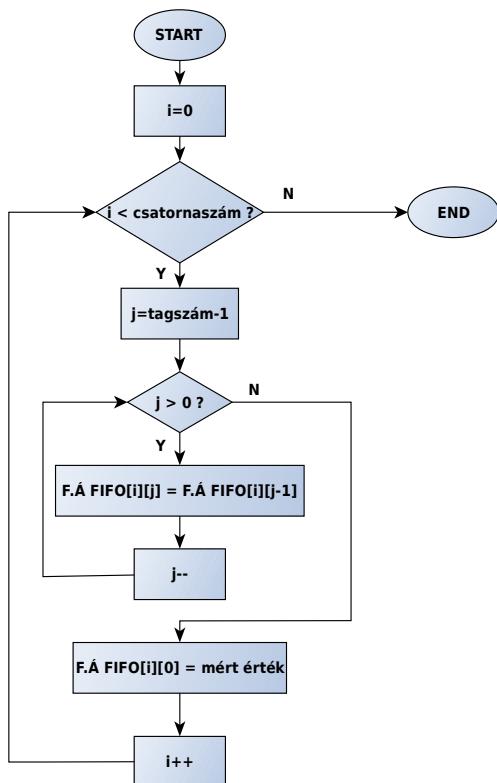
A pufferkezelő működése során két listát épít. Egyet az üres pool elemeknek és egyet az adatokkal feltöltött pool elemeknek. A listaelemek tartalma egy típus nélküli (*void**) mutató. Inicializáláskor a modul a listaelemek számára memóriát foglal és az üres lista a puffer számára deklarált pool elemeinek címével kitöltve kerül felépítésre. Mikor a termelő szél kér egy üres puffer elemet a pufferkezelő az üres lista első elemét kiveszi a listából és odaadja. A termelő szál az így kapott listaelem által tartalmazott mutató szerinti memóriaterületet feltöltheti adatokkal. A feltöltött pufferelement visszaadja a pufferkezelőnek aki beszúrja a feltöltött pufferelemek listájának végére. Amikor a fogyasztó szál kér egy tele puffereemet,

megkapja e lista első elemét. A tartalmazott mutató segítségével így feldolgozhatja az adatokat, majd a kiürült puffer elemet visszaadja a pufferkezelőnek, aki a visszakapott listaelemet az üres lista végére szúrja. Működése során a következő statisztikákat vezeti: underflow, értéke mindenig egyelőre növekszik, ha egy szál üres elemet szeretne kivenni, de nem rendelkezik vele a puffer; overflow, értéke mindenig egyelőre növekszik, ha egy szál üres elemet szeretne visszatenni, de az üres elemek száma maximális; postoverflow, értéke mindenig egyelőre növekszik, ha egy szál tele elemet akar a pufferbe tenni, de a tele elemek száma maximális. Továbbá lekérdezhető a puffer mérte, üres és tele elemek száma, valamint az inicializálás során történt hiba a listaelemek memória allokációja során történt hiba.

4.5.3. Hőmérséklet mérés

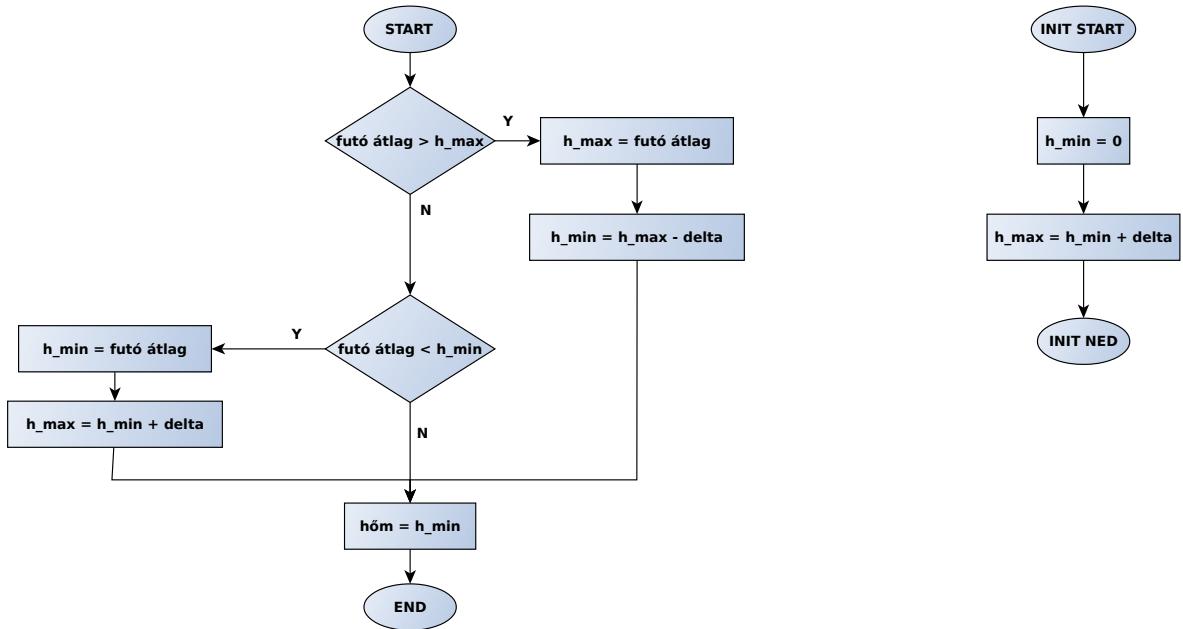
A csatornák aktuális hőmérsékletének meghatározása kettős zavarszűréssel ellátott. Első lépésben a mért hőmérsékletek futó átlagának kiszámítása történik. Az átlagoló algoritmus

eredményeként meghatározott hőmérséklet minden az utolsó n mérés átlaga: $T_n = \frac{\sum_{i=1}^n T_i}{n}$.



21. Ábra: Futó átlag FIFO léptetése
igazítja. Működését és kezdő értékeinek beállítását 22 ábra szemlélteti. A delta konstans érték, a hiszterézis intervallumának szélessége.

Megvalósítása egy kétdimenziós tömb segítségével történik, mely első indexének értéke a csatornák száma, másodiké pedig a futó átlag tagszáma. A újonnan mért hőmérsékletek ebbe a tömbbe kerülnek beléptetésre. Beléptetés előtt a tömb elmei az egyel magasabb indexű helyre kerülnek átmásolásra, az új elem pedig mindenig az első helyre kerül ábra. Ezáltal létrejön egy egyszerű FIFO, mely az utolsó n mérést tartalmazza (21. ábra). A FIFO-ban tárolt hőmérsékletek számtani átlagát mérésenként kiszámolva keletkezik a hőmérsékletek futó átlaga, mely a futó hiszterézis algoritmus bemenetére kerül, mely a hőmérsékletek értékek „ugrálását” hivatott megszüntetni. Az algoritmus a hiszterézis intervallumának az alsó értékét adja eredményül, miközben az intervallumot mindenig kapott értékekhez



22. Ábra: Futó hiszterézis algoritmus és inicializálása

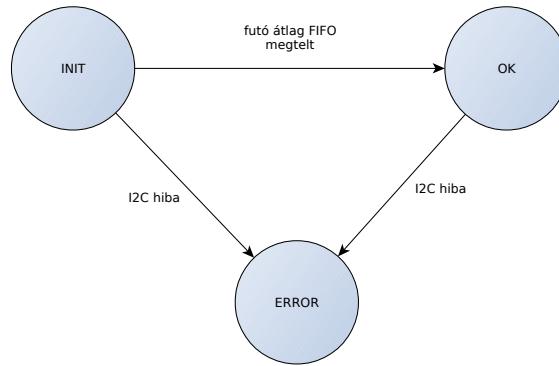
A hőmérséklet mérő szál két állapota van. Inicializálási állapotban indul, mely állapotban futó átlag FIFO feltöltése zajlik. Mikor lemérésre és eltárolásra került a FIFO tagszámának megfelelő mennyiségű adat, kiszámolhatóvá válik a csatornák aktuális hőmérséklete. Ekkor a szál OK állapotra vált és az állapotváltásról egy mailbox üzenetben értesít sterilizálást végző szálat. Ebben az állapotban kezdődik meg a mérések adatainak meghatározza és a hőmérséklet FIFO-ba töltése a szabályozást végező szál számára A hőmérsékletek számára definiált struktúra:

```

typedef struct{
    uint32_t timestamp;
    int16_t temp[CHANNEL_NUM];
    int16_t dtemp[CHANNEL_NUM];
    bool is_sterile;
}temperature_t;

```

Tartalmazza a mérés időbélyegét, a csatornák hőmérsékleteit és változásait az előző mintavételhez képest és a sterilizálási státusz logikai változóját, melynek értéke 1, ha minden csatorna hőmérséklete a szükséges 114 °C felett van.



23. Ábra: Szenzor állapotok

A szenzorok kiolvasása során detektálásra kerülnek I²C buszon keletkezett hibák és az időtúllépés, így egy szenzor a 23. ábra szerinti állapotokat veheti fel:

- INIT: A szenzor inicializálási állapotban van, amíg a futóátlag nem számítható, de a szenzor mér és üzemképes.
- OK: A futó átlag FIFO megtelt, a szenzor mér és üzemképes.
- ERROR: A szenzorral történő kommunikáció során hiba történt, a szenzor üzemképtelen. A hőméréséket mérést szál ekkor szenzorhiba üzenetet küld a hibakezelő szálnak és megjeleníti a hibás szenzor állapotát.

4.5.4. Fuzzy elvű szabályozás

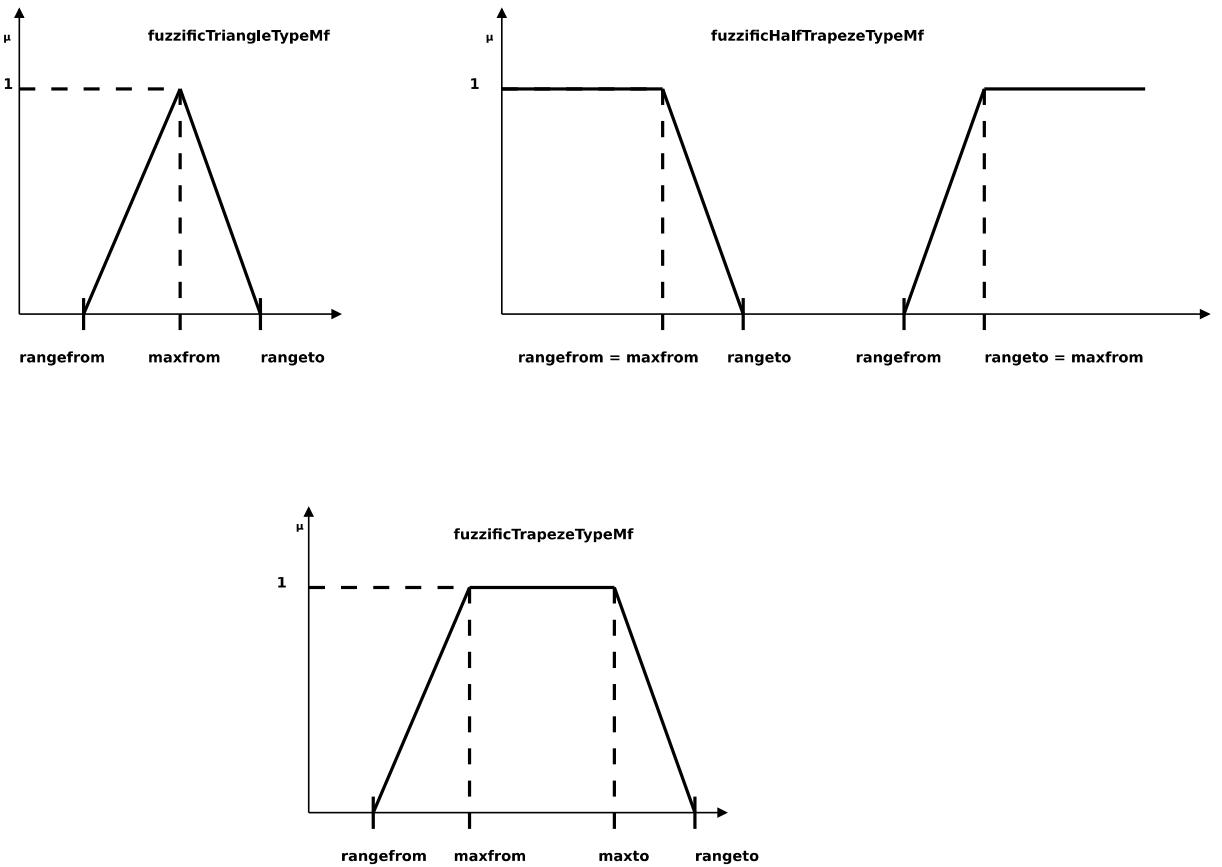
A fuzzy logika megvalósításához a következő típusok kerültek definiálásra:

```

typedef struct{
    int16_t rangefrom;
    int16_t rangeto;
    int16_t maxfrom;
    int16_t maxto;
    float(*fuzzyfic_func)(void *mfp, int16_t input);
}input_mf;

```

Tartalmazza a szabályozó bemeneteihez rendelt tagsági függvények töréspontjaihoz tartozó éles bemeneti értékeket és a fuzzyfikálást végző függvény mutatóját.



24. Ábra: Alkalmazható tagsági függvények alakjai és a fuzzyfikálást végző függvényeik neve

Az ábrán látható, hogy a *maxto* csak a trapéz alakú tagsági függvény esetében kerül használatra, többi fuzzyfikációs függvény figyelmen kívül hagyja. Ennek az oka, hogy így egy típus használható az összes tagsági függvény deklarálásra.

```
typedef struct{
    uint8_t maxpoint;
}output_mf;
```

A szabályozó kimenetéhez tartozó tagsági függvények leírásához csak a maximum pontjuk szükséges mivel a kimeneti tagsági függvények háromszög alakúak és a súlyozott maximum módszer alkalmazása miatt csak a maximum pontjuk kerül figyelembevételre.

```
struct Temp_mship{
    input_mf melting;
    input_mf cold;
    input_mf medium;
    input_mf hot;
    input_mf sterile;
};
```

```

struct DeltaTemp_mship{
    input_mf neg;
    input_mf zero;
    input_mf spos;
    input_mf pos;
    input_mf vpos;
};

struct PWM_mship{
    output_mf off;
    output_mf small;
    output_mf half;
    output_mf wide;
    output_mf full;
};

};

```

A három struktúra definíció, melyek az applikációhoz használt be és kimeneti tagsági függvényeket tartalmazzák. Értékkel való feltöltésükkor, arra kell figyelni, hogy a megvalósított fuzzy logika a szenzorok natív értékeivel számol, így a bemeneti függvények sarokpontjaihoz tartozó éles bemeneti érték a következő módon számolható a szenzorok 16

bites felbontása esetén: $T_{\text{natív}} = \frac{T}{T_{\text{felbontás}}} = \frac{75^{\circ}\text{C}}{0,0078125^{\circ}\text{C}} = 9600$. A kimeneti tagsági függvények maximum pontjainak értéke az adott tagsági függvény maximumához tartozó éles kimeneti érték ami a kitöltési tényező %-os értékét jelenti.

```

typedef struct{
    input_mf *if_side1;
    input_mf *if_side2;
    output_mf *then_side;
}fuzzy_rule;

```

A szabályok típusa. Tartalmazza a szabály IF oldalához tartozó tagsági függvény mutatóját. Az első minden a hőmérséklet, a második pedig a hőmérséklet változás. Azok a szabályok esetében, mikor a hőmérséklet változás bemenettől függetlenül történik a kimenet meghatározása, az *if_side2* mutatót NULL értékűre kell definiálni. A harmadik eleme a szabályhoz tartozó kimeneti tagsági függvény mutatója. A szabálybázis *fuzzy_rule* típusú struktúrák tömbje, melynek elemszáma a szabályok számával egyenlő.

A szabályozó első lépésben *static void fuzzyfication_input(int16_t temp, int16_t dtemp)* függvényt hívja meg melynek argumentumai a szabályozó két bemenetének értéke. A

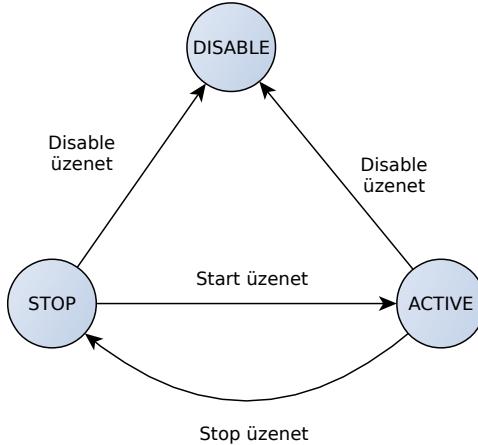
fuzzyfikálás során a függvény végighalad a szabálybázison és meghatározza bemenetek szabályokhoz tartozó fuzzy értékét. (A szabályban szereplő bemeneti tagsági függvény igazság értéke a bemeneti éles érték helyén). Az érvényes értékeknek a[0, 1] intervallumon helyezkednek el. Így a függvény hiba esetén a 2 értéket adja viasza, illetve ha a hőmérséklet változás bemenetet tagsági függvényének mutatója NULL, akkor 3-at. A kapott értékeket a két bemenet fuzzy értékei számára deklarált, a szabályok számával egyenlő méretű tömbök szabálynak megfelelő indexű helyre írja.

Az ezt követő lépés az aktív szabályok kiválasztása. Ezt a feladatot látja el a *static void evaluation_rules(void)* függvény. Futása során végig halad a két bemeneti fuzzy értékeket tartalmazó tömb elemein és kettő közül a kisebbet a szabályok aktivitási fokának tárolására deklarált tömb megfelelő indexű helyére másolja. A minimum kiválasztás a bemenetek ÉS összekapcsolását jelenti. Így előáll az egyes szabályok aktivitási foka, az inaktív szabályok esetén ez 0. Az összehasonlítás során, ha bárhol 2 értéket talál a függvény, tesz egy hibabejegyzést majd egygel növeli a hibaszámlálót, melyet a szabályozást végző szál folyamatosan figyel és hiba esetén hibaüzenetet küld.

Az utolsó lépés az életlen kimeneti értékek defuzzyifikálása. A feladat végrehajtásért felelős függvény *static uint32_t defuzzyification(void)*. Az abszolút maximum módszer használatával a szabályokhoz előállított aktivitási fok és a kimenti tagsági függvények maximum helyének alapján kiszámolja a szükséges kitöltési tényező értékét.

A szabályozó szál állapotai (25. ábra):

- FUZZYREG_STOP: A szál ebben az állapotban kezdi meg működését. A PWM csatornák letiltott állapotban vannak, de bármikor elindíthatóak.
- FUZZYREG_ACTIVE: A sterilizáló száltól érkező START üzenet hatására vált a szabályozó ebbe az állapotba. Ebben az állapotban a PWM kimentek aktívak.
- FUZZYREG_DISABLE: A hibakezelő szál által küldött DISABLE üzenet hatására történik a váltás ebbe az állapotba. Kilépni ebből az állapotból a szál nem tud, az eszközöt ilyenkor kikapcsolni szükséges és a hiba elhárítása után újraindítani.



25. Ábra: A szabályozó szál állapotai

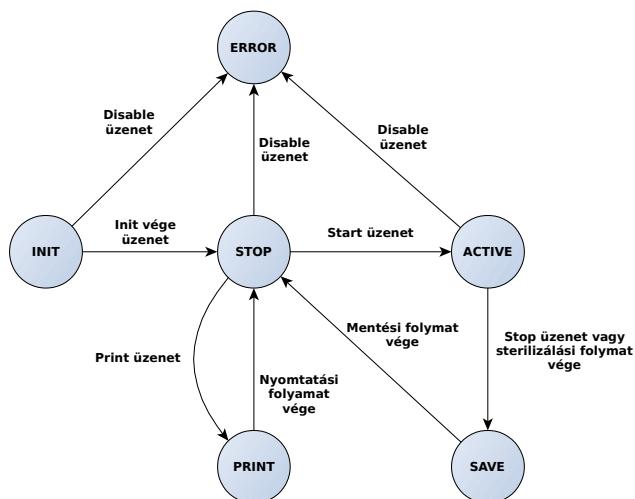
A szabályozó szál állapot váltásait elidéző üzenetek során végrehajtott műveletek a következőek:

- START: A PWM csatornák aktiválása, kezdeti hőmérsékletek elmentése, log fájl létrehozása, a szabályozó FUZZYREG_ACTIVE állapotba állítása.
- STOP: PWM csatornák inaktiválása, kitöltési tényező értékének nullázása, szabályozó FUZZYREG_STOP állapotba állítása.
- DISABLE: PWM csatornák inaktiválása, kitöltési tényező értékének nullázása, szabályozó FUZZYREG_DISABLE állapotba állítása.

A szabályozást végző szál állapotától függetlenül futásának minden ciklusában a kivesz egy elemet a hőmérséklet FIFO-ból (amennyiben nem üres) és megjeleníti. A szabályozó aktív állapotában a szál egy ciklusban elvégzett feladatai kiegészülnek. Ilyenkor megvizsgálja, hogy van-e olyan csatorna, melynek hőmérséklete a kritikus érték (125°C) felett van, ha igen akkor hibaüzenetet küld a hibakezelő szálnak. Meghatározza a fuzzy logika segítségével a csatornák PWM jeleinek kitöltési tényezőjét. Ellenőrzi a csatornák hőmérséklet-időfüggvényének tga értékét. Túl magas érték esetén hibaüzenet küld a hibakezelő szálnak. Kijelzi az aktuális kitöltési tényező értékét a kijelzőn. Log bejegyzést készít a csatornák hőmérsékleteiről, előző mintavételhez képesti változásukról, valamint a kitöltési tényezőkről. Majd amennyiben a fuzzy logika hibaszámlálója nem 0, hibaüzenet küld a hibakezelő szálnak.

4.5.5. Sterilizálás

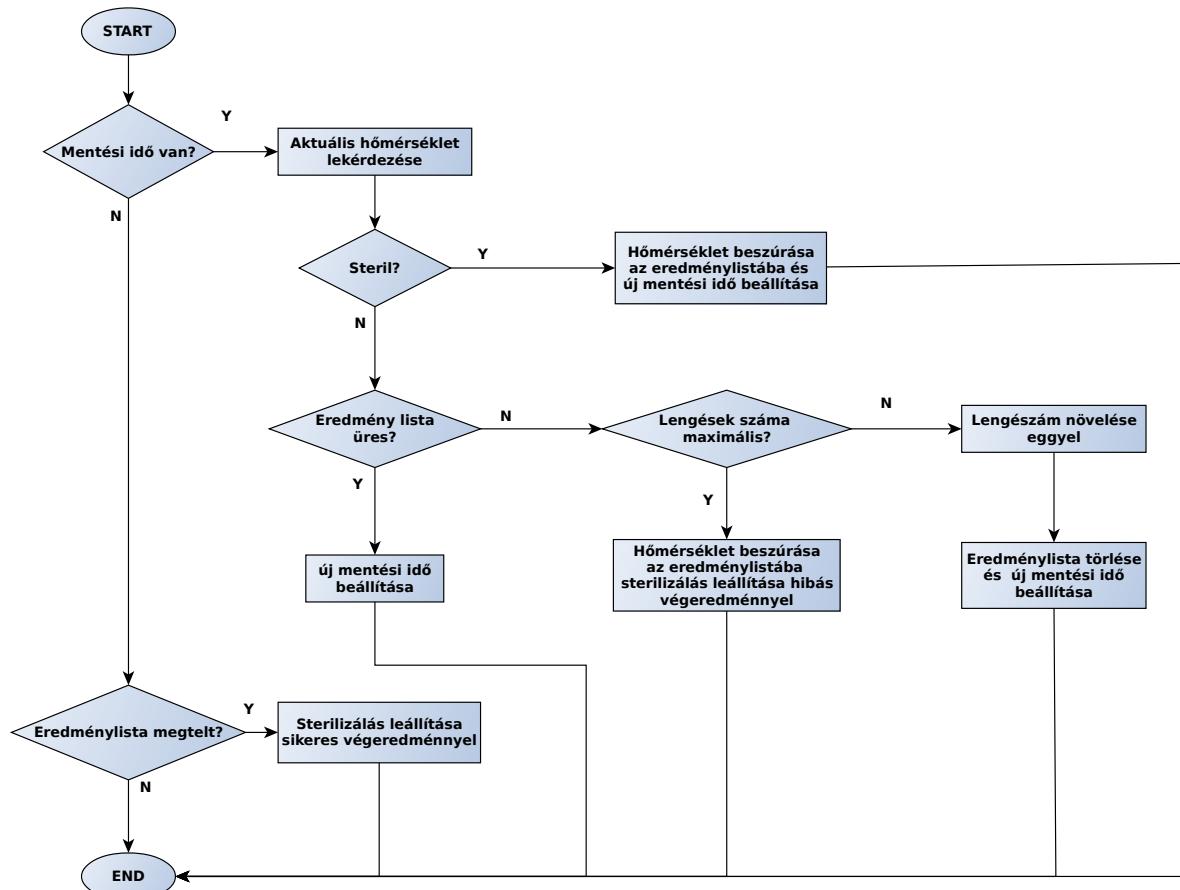
A sterilizálás feladatát ellátó programszál gondoskodik a mélviasz hőkezeléséhez előírt feltételek teljesüléséről. Ennek érdekében a sterilizálási folyamat alatt percenként leellenőrzi a mért hőmérsékleteket és egy eredmény listában rögzíti. A lista tartalma: a kezdés dátuma és ideje, a három csatornán mért hőmérsékletek a mérési idejükkel és a sterilizálási státuszukkal, a befejezés ideje, valamint a folyamat végeredménye. A sterilizáló szál folyamat állapotai 26. ábrán láthatóak.



26. Ábra: A sterilizáló szál állapotai

A sterilizáló szál inicializálás állapotban van a szoftver elindulásakor. Ez az állapot addig tart, míg a hőmérséklet olvasást végző száltól egy inicializálás vége üzenetet fogad, ekkor válik lehetővé a sterilizálás folyamatának megkezdése, így a programszál stop állapotba vált. A sterilizálás folyamata a kijelzőt kezelő szál start üzenetével indítható, melyre a start rutin kerül végrehajtásra. A rutin során az eredménylista kiürítésre kerül (amennyiben nem üres), tárolásra és kijelzésre kerül a dátum és a kezdő időpont, beállításra kerül a mentési idő, a lengések számának nullázása, valamint egy start üzenet küldése történik a fűtőelemek szabályozását végző programszálnak. A rutin végén a sterilizálás aktív állapotba kerül. A sterilizálási folyamat során, programszál minden ciklusában végrehajtja 27 ábra szerinti műveleteket. Az eredménylista 61 elemmel rendelkezik, így a percenkénti mentés mellett, megtelésekkor a viasz a szükséges egy órát töltött a steril hőmérsékleten. A szabályozás karakteristikájából eredően, a hőmérséklet az szabályozási idő alatt bekövetkezett lengések során a hőmérséklet, akár 114 °C alá is süllyedhet. Ez a sterilizálási folyamat behatási ideje alatt nem engedhető meg. Azért, hogy a folyamat egy vagy néhány hőmérséklet kilengés

következtében ne szakadjon meg, definiálható a sterilizálás folyamatához a eltűrt lengések maximális száma. Amennyiben a kilengések száma elérte a maximumot, és ezt követően a percenként, bármely szenzoron mért hőmérséklet 114 °C alá csökken, a hibás eredmény listában rögzítése után a hibás végeredménnyel zárul a sterilizálás folyamata.



27. Ábra: A sterilizálás ciklikusan végzett műveletei

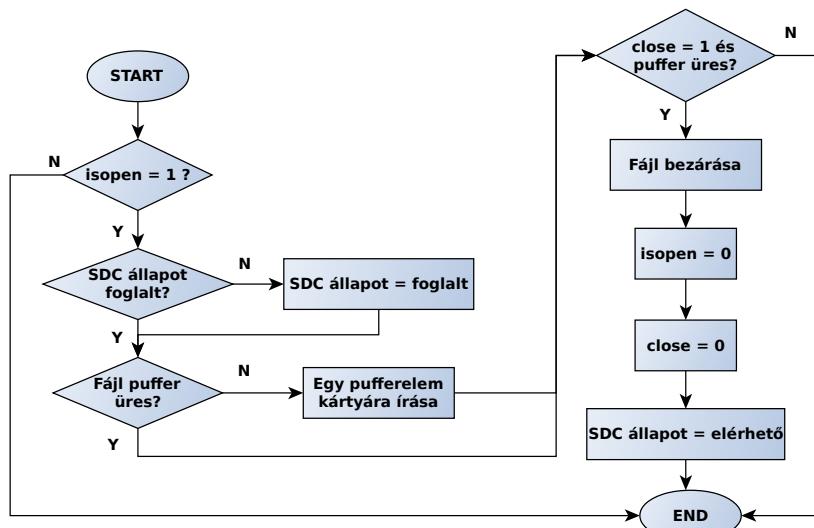
A folyamat a felhasználó által előidézett és a kijelző száltól érkező stop üzenet segítségével is bármikor leállítható. A folyamat leállása akármilyen okból történik (sikeres vagy sikertelen végeredmény, felhasználó által) a stop rutin végrehajtásával zárul. E rutin során a programszál stop üzenet küld a szabályozást végző szálnak, eltárolja és megjeleníti a folyamat befejezésének időpontját, majd a állapotát mentés állapotba állítja. A mentés folyamán, ha az eredménylista nem üres és a eredmények számára történő fájl létrehozása sikeres volt, akkor az eredmények a fájlhoz rendelt pufferen keresztül a microSD kártyán mentésre kerülnek, és a sterilizálás a mentés befejeztével ismét stop állapotba kerül. A print üzenet hatására a sterilizáló szál állapota nyomtatási állapotba kerül és az eredménylista tartalmát a nyomtatón

pufferbe helyezi. A hibakezelő szál disable üzenetének hatására hibaállapotba kerül, melyből nem léphet ki. Bekövetkezése esetén hibaelhárítás és újraindítás szükséges.

4.5.6. Fájlkezelés

Az fájlkezelés a ChaN-FatFs modul segítségével történik, így az applikáció futása során keletkező két fájl elmentéséhez egy FAT fájlrendszerrel ellátott µSD kártyára van szükség. Egy virtuális timer segítségével monitorozza a kártya jelenlétét. Segítségével a kártya behelyezésekor és eltávolításakor egy-egy esemény generálódik. A kártyakezelő a két esemény bekövetkeztekor elvégzi a kártya fel és lecsatolásának műveletét és a kártya állapotát megjeleníti a kijelzőn. A kártya által felvehető állapotok:

- SDC_NOTINSERTED: A kártya kivételre került.
- SDC_ERROR: A kártya fájl rendszerét nem sikerült felcsatolni.
- SDC_BUSY: Valamelyik fájl van nyitva.
- SDC_READY: A kártya használatra kész és nincs nyitott fájl.
- SDC_FULL: A kártyán felcsatoláskor nem volt szabad hely található.



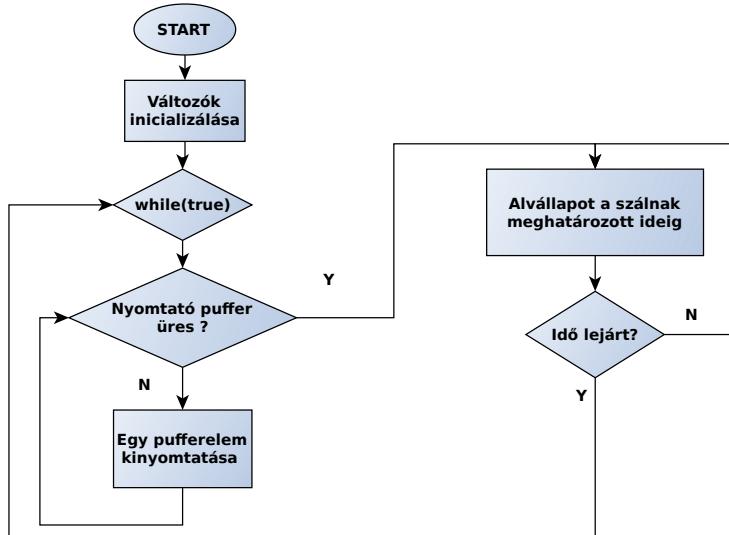
28. Ábra: Fájlok kezelése

A fájlíráshoz a két fájlnak külön puffere van, a szál ezt a kép puffert ürítve írja ki az adatokat a kártyára. minden fájlhoz tartozik két logikai változó. A *bool isopen* és *bool close*. Az adott fájl pufferébe termelő szál az adatközlés előtt létrehozza a fájlt és az *isopen* változót igaz, a *close* változót hamis értékre állítja. A kártyakezelő szál ennek hatására az adott fájl pufferét

elkezdi kiüríteni. Mikor befejezte a termelő szál a fájlközlést a *close* értékét igazra állítja. Ekkor a kártyakezelő szál tudja, hogy nem történik tovább adatközlés, így a puffer kiürülése után bezárhatja a fájlt. Bezárás után minden változót hamis értékre állítja.

4.5.7. Nyomtató kezelés

A nyomtatót kezelő szál feladata egyszerű. Futásának minden ciklusában megvizsgálja a nyomtató számára fenntartott puffert és ha van benne adat azt a soros porti kimenetre juttatja.



29. Ábra: Nyomtatókezelés

4.5.8. Hibakezelés

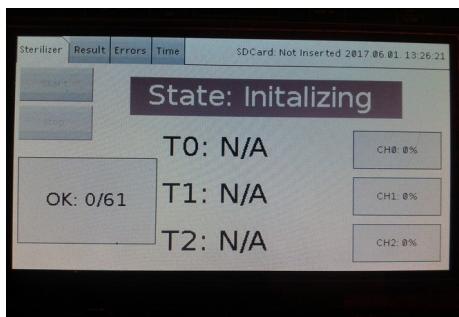
A működés során az applikáció szálai a detektál hibák esetén mailbox üzenetben értesítik a hibakezelőt. A hibakezelő a detektált hibákból egy listát épít, melyet megjelenít a kijelzőn. Az előfordulók hibák a következők lehetnek: szenzorhiba, túl magas hőmérséklet, túl gyors hőmérséklet emelkedés, fuzzy logika hiba, biztosíték hiba. Ezen hibák természete minden olyan, hogy keletkezésük után a hőkezelés nem folytatható, így a szabályozó és a sterilizáló szál is letiltásra kerül. A csatornák olvadó biztosítóhoz tartozó relék által generált megszakítás kezelését a ChibiOS EXT Drivere végzi. A driver csatornához GPIO (General Purpose Input Output -Általános célú be- és kimenet) lábakat lehet rendelni, melyeken keletkező fel és lefutó élek hatására megszakítás generálódik. Az applikáció esetében ez felfutó él. A pergésmentesítés úgy történik, hogy a generált megszakításkor az adott lábhoz tartozó csatorna letiltásra kerül és egy virtuális timer indul a relék pergési idejénél nagyobb időzítéssel. Így az időzítő lejártáig nem számít az adott bemenet állapota. Az időzítő lejártakor

az adott bemenet állapota megvizsgálásra kerül, magas értéke esetén a hibaüzenet küldés történik, majd az adott csatorna engedélyezése.

4.5.9. Grafikus felhasználói felület

A kijelzőt kezelő programszál az adatok megjelenítéséhez egy rajzolási műveleteket tartalmazó feladatsort üzemeltet. A feladatsor a rajzolást végző függvények mutatóiból álló láncolt lista. Amikor valamely programszál adatot szeretne megjeleníteni a kijelzőn, ezeket megosztja a kijelzőt kezelő szállal és az adatok megjelenítéséhez szükséges függvény mutatóját a lista végére helyezi. A kijelzőt kezelő szál, futásának minden ciklusában megjeleníti az időt és a feladatsorban összegyűjtött feladatokat elvégzi, valamint az érintő kijelző eseményeit kezeli.

A grafikus felület megvalósítása a µGFX GWIN modulja által nyújtott widgetek segítségével történt. A felhasználó felület fő alkotó eleme egy Tabset widget, melynek lapjai tartalmazzák az egyes funkciókhöz tartozó további widgeteket. Az első lap a sterilizálás folyamatához kapcsolódó információkat foglalja össze. A folyamat állapotát és az egyes csatornához tartozó hőmérséklet értékeit egy-egy címke (Lable widget) jeleníti meg. Az csatornák PWM jeleinek kitöltési tényezőit folyamatjelzők (Progressbar widget) szemléltetik grafikusan és numerikus értékkal is. Ugyanilyen widget tartozik a sterilizálás folyamata során az eredménylistában szereplő hőmérsékleteket számának kijelzéséhez, mely számszerűen megjeleníti, hogy aktuálisan mennyi „OK” eredmény van a szükséges 61-ből. Ezen a lapon helyezkedik el a Start és a stop gomb (Button widget), melyek segítségével indítható vagy leállítható a folyamat. A bekapcsoláskor az állapot szürke háttérrel és a „State: Initializing” felirattal jelenik meg továbbá hőmérsékletek értéke helyén az N/A felirat olvasható. Ekkor történik a futó átlag számításhoz szükséges adatgyűjtés, így nincs még megjeleníthető hőmérséklet. Ilyenkor a Start és Stop gombok letiltott állapotban vannak, sterilizálás nem indítható.(30. ábra) A futó átlag számítás megkezdődésével megjelennek a hőmérséklet értékek és „State: Stop” felirat. A Start gomb aktív állapotba kerül, megkezdhető a hőkezelés. (31. ábra)

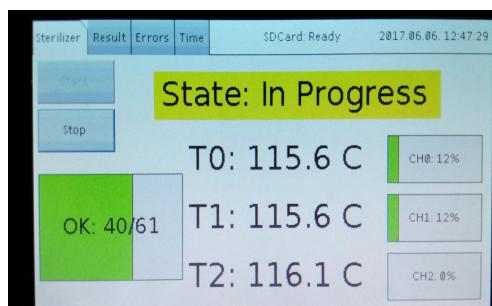


30. Ábra: Inicializálási állapot kijelzése

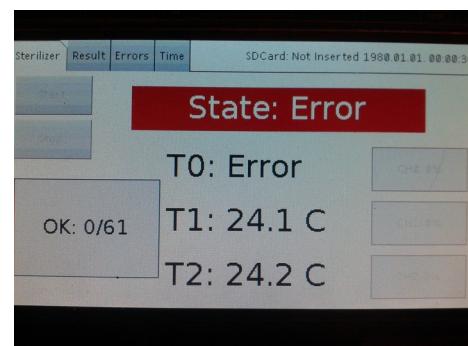


31. Ábra: Stop állapot kijelzése

A hőkezelés elindulásával az állapot jelző címke hátttere sárgára vált és a „State: In Progress” felirat tájékoztat, hogy a hőkezelés folyamatban van, a folyamatjelzők pedig informálnak az aktuális fűtőteljesítményről és a sikeres hőmérséklet vizsgálati eredmények számáról (32. ábra). Hiba keletkezése esetén a állapotjelző háttere piros színű és a „State: Error” felirat A Start és Stop gomb letiltásra kerül és a elszürkülnek a fűtőteljesítmény folyamatjelzői jelezve, hogy a kimenet letiltásra került. Szenzorhiba esetén, az érintett csatornához tartozó hőmérséklet érték helyett az „Error” felirat jelenik meg (33. ábra).



32. Ábra: Sterilizálás folyamatának kijelzése

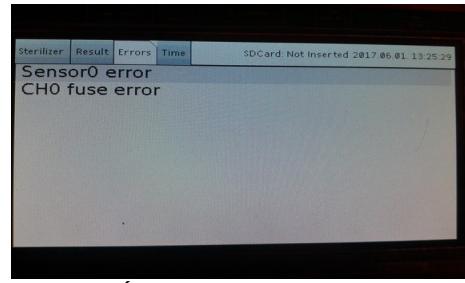


33. Ábra: Hibaállapot állapot kijelzése

A második lap tartalmazza a sterilizálási folyamat dátumát, kezdő és befejező idejét, valamint végeredményét címkék formájában. A percenként mért eredmények kiírásához egy lista objektum ad lehetőséget (List widget), mely a scrollolható listát jelenít meg. Ezen a felülben helyezkedik el a Print gomb, mely segítségével a listát lehet kinyomtatni (35. ábra). Az errors lap tájékoztat a detektál hibákról szintén egy lista formájában (34. ábra).

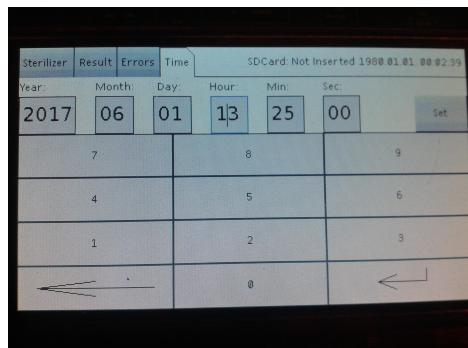
Sterilizer	Result	Errors	Time	SDCard: Ready	2017.06.06. 12:51:59
Date: 2017.06.06					
Start: 10:25:13	End:				
Nr.	Time	CH0	CH1	CH2	Status
00	12:08:13	114.1 C	114.0 C	115.3 C	Sterile
01	12:09:13	114.3 C	114.2 C	115.2 C	Sterile
02	12:10:13	114.6 C	114.6 C	115.4 C	Sterile
03	12:11:13	114.9 C	114.6 C	115.7 C	Sterile
04	12:12:13	115.2 C	114.9 C	115.9 C	Sterile
05	12:13:13	115.5 C	115.0 C	116.1 C	Sterile
06	12:14:13	115.6 C	115.3 C	116.4 C	Sterile
07	12:15:13	115.8 C	115.3 C	116.5 C	Sterile
08	12:16:13	116.0 C	115.4 C	116.6 C	Sterile
09	12:17:13	116.8 C	115.6 C	116.8 C	Sterile
10	12:18:13	117.1 C	116.9 C	117.0 C	Sterile
11	12:19:13	116.8 C	115.9 C	117.0 C	Sterile
12	12:20:13	116.8 C	116.2 C	116.9 C	Sterile
13	12:21:13	116.1 C	116.3 C	116.9 C	Sterile

35. Ábra: Eredménylista kijelzése



34. Ábra: Hibaüzenet kijelzése

A time fülre lépve jelenik meg a dátum és idő beállítására szolgáló felület. Tartalma egy numerikus billentyűzet (Keyboard widget) és az adatok megadására szolgáló szövegszerkesztő (TextEdit widget) elemek és egy Set gomb a beírt idő érvényesítésére. Váltani a lapok között tabset fejlécében megjelenő fülek megérintésével lehet. Az microSD kártya állapota, a dátum és idő is a fejléken olvashatóak (36. ábra).



36. Ábra: Dátum és idő beállító felület

4.5.10. A main függvény

A szoftver futását a main szál kezdi, ez az applikáció belépési pontja. Inicializálja a rendszert majd a beállítja a GPIO lábakat, majd elindítja az egyes programszálakat. Végtelen ciklusában az a Shell kezelését látja el, mely révén egy egyszerű parancssoros felületen lehet kommunikálni a szoftverrel. A szoftver teljes forráskódja a 3. számú mellékletben található.

5. A tesztberendezés és próbaüzemének eredményei

A szoftver működésének kipróbálása céljából elkészült egy három csatornával rendelkező tesztmodell. A modell a viasz hevíttésére 3 darab főzőlapot használ, melyek egyenként 500W névreleges teljesítményen üzemelnek. A viasz tárolását egy 20 literes zománcozott edény látja el. A készülék fém burkolata alatt üveggyapot szigetelés van, a hőveszteség és az égési sérelmek elkerülése céljából. (1. számú melléklet, 4.kép) Az áramkörök egy átalakított

kismegszakító házban kerültek felszerelésre a burkolatra. (1 számú melléklet, 1-2.kép) A szenzorok három rézcsőbe kerültek beépítésre, melyeken réz zárókupak van hőálló ragasztóval felragasztva, hogy a viasz ne juthasson be a cső belsejébe. A szenzorokat az edénybe juttatni az eszköz tetején kivágott lyukakon lehetséges, amihez a viasz darabokra kell törni. A szenzorok távolságát az edény aljától a réz csöveken található csavaros tömlő bilincs segítségével lehet beállítani. (1 számú melléklet, 5.kép) A próbák során ez kb. 1,5-2 cm volt. A nyomtatás feladatát egy Olivetti PR2E típusú mátrix nyomtató látta el, mely az Olimex által gyártott RS323 soros porton csatlakozott az eszközhöz. (1 számú melléklet, 3.kép) A hőkezelések során 5 és 10 kg viasz került sterilizálásra. A sterilizálási folyamatok során keletkező log fájlok kiértékeléshez készült egy awk script, melynek segítségével előállítható a szabályozás során keletkezett csatornánkénti legnagyobb túllövés és a szabályozási idő ami után a hőmérséklet már nem lép ki a dinamikus ingadozási tartományból. Az eredményeket egy szövegfájlba menti, melynek tartalma a következő:

File: log2017_05_09_17_25_12_5kg.dat

Total time: 9060 s

Time of regulation CH0: 5357 s

Time of regulation CH1: 5433 s

Time of regulation CH2: 5451 s

CH0 maximun temperature: 117.617 °C at 6417 s

CH1 maximun temperature: 117.492 °C at 6435 s

CH2 maximun temperature: 117.226 °C at 6492 s

File: log2017_05_10_08_16_32_10kg.dat

Total time: 11941 s

Time of regulation CH0: 8137 s

Time of regulation CH1: 8288 s

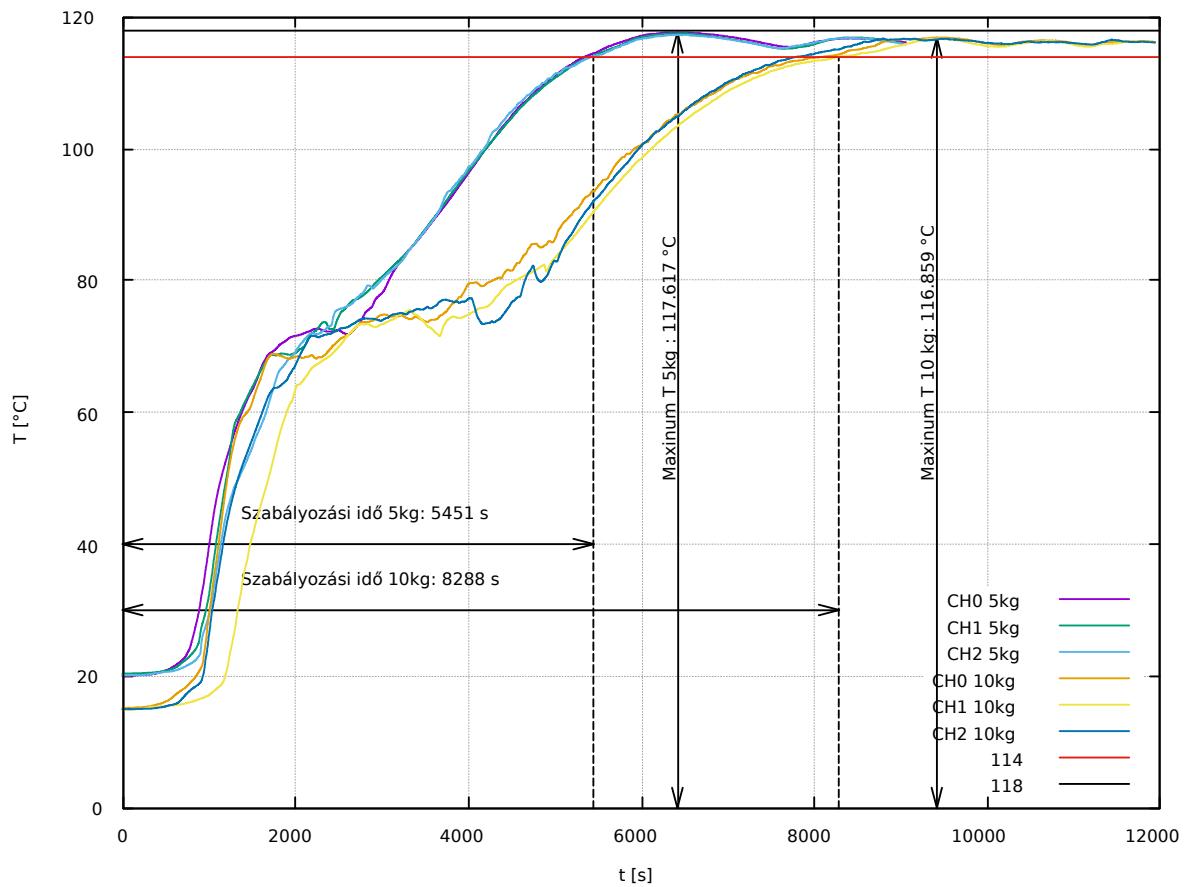
Time of regulation CH2: 7845 s

CH0 maximun temperature: 116.859 °C at 9417 s

CH1 maximun temperature: 116.695 °C at 9423 s

CH2 maximun temperature: 116.695 °C at 8795 s

A kapott fájl alapján a legnagyobb túllövés minden esetben a 0-dik sorszámú csatornán keletkezett, valamint a leghosszabb szabályozási idő 5 kg viasz esetén a 2. és 10kg esetén az 1-es számú csatornán volt tapasztalható, amit a 37. ábra szemléltet.



37. Ábra: A hőmérsékletek alakulása 5 és 10 kg viasz esetén

6. Összefoglalás

A szakdolgozatom célja volt egy olyan áramkör megalkotása, amely a jelenlegi jogszabálynak megfelelve képes ellátni, a méhviasz hőkezelésnek feladatát. Ehhez egy olyan szabályozó algoritmusra volt szükség, amely a méhviaszt legalább egy órán át 112 °C felett tartja. A fuzzy logika alkalmazásával ennek sikerült eleget tenni, melynek nagy előnye volt, hogy segítségével, nem volt szükség a rendszer pontos matematikai modelljének felállítására. Amint az a 37. ábrán látható volt, sikerült elérni hogy a hőmérséklet a 114 °C elérése után és a legnagyobb túllendülések alkalmával sem lépett ki a meghatározott +/-2 °C-os dinamikus ingadozási tartományból. Viszont ahogy a 1. számú melléklet 4.képén látni, sajnos méhviasz színe a sorozatos hőkezelések során a nem kívánatos sötétbarna lett. Ennek oka, hogy a méhviasz nem lett kellően megtisztítva a sterilizálások előtt. Így a jövőben erre sokkal jobban oda kell figyelni.

A hőkezelési folyamat során a méhviasz steril hőmérséklet felett töltött idejét percentként vizsgálja szoftver, így az előírt legalább 5 percentkénti vizsgálatnak is sikerült megfeleni. A kinyomtatás mellett, memória kártyán való eltárolás is megvalósult, segítségével a teljes folyamatról információhoz tudtam jutni a fejlesztés során, ami sokat segített a szabályozó hangolásában.

A szoftverhez megvalósult egy egyszerű grafikus felhasználói interfész, mely segítségével az érintő képernyőn keresztül lehet interakcióba lépni az eszközzel. Ezt a választott STM32F746 DISCOVERY fejlesztői kártya tette lehetővé és a µGFX grafikus keretrendszer által szolgáltatott lehetőségek. A szoftver alapját a ChibiOS valós idejű operációs rendszer képezte a fejlesztés során. Mindkettőről elmondható, hogy aktívan fejlesztett szoftver és kitűnő dokumentációval rendelkeznek, így használatuk előnyös.

A célt, hogy megvalósuljon egy tesztberendezés, mely alapját képezheti egy későbbi eszköznek sikerült elérni. Egy teljesen komplett késztermék még azonban sokkal több munkát igényel. További fejlesztések irányára lehet a felhasználói felület, részletesebb funkció gazdagabb kidolgozása. Pl.: a hőmérséklet alakulásának ábrázolása grafikonon a hőkezelés során, a fuzzy logika függvényeinek megjelenítése. Az edényzet tovább fejlesztése mindenkorban szükséges, egy leeresztő csappal ellátott edény és a szenzorok alulról történő beépítése célszerűbb volna, mert így a nagyobb mennyiségi viaszánál körülményesebb volt a sok viaszdarab között lejuttatni a szenzorokat az edény aljára.

7. Irodalom jegyzék

- 1: 70/2003. (VI. 27.) FVM rendelet a méhállományok védelméről és a mézelő méhek egyes betegségeinek megelőzéséről, I fejezet, 12 § (2), 2003
- 2: Örösi Pál Zoltán: Méhek között, Mezőgazdasági kiadó, Budapest, 1957, Viasz és viaszolvasztás
- 3: Dr. Szakonyi Lajos, Jancskárné Anweiler Ildikó: Szabályozások, Pécs, Pécsi Tudományegyetem Pollack Mihály Műszaki Főiskolai Kar Műszaki Informatika Tanszék, 2002, 10. Bevezetés a fuzzy-elvű szabályozásokba
- 4: Dr. Szakonyi Lajos, Jancskárné Anweiler Ildikó: Szabályozások, Pécs, Pécsi Tudományegyetem Pollack Mihály Műszaki Főiskolai Kar Műszaki Informatika Tanszék, 2002, 10. Bevezetés a fuzzy-elvű szabályozásokba, 169-170 p.
- 5: Dr. Szakonyi Lajos, Jancskárné Anweiler Ildikó: Szabályozások, Pécs, Pécsi Tudományegyetem Pollack Mihály Műszaki Főiskolai Kar Műszaki Informatika Tanszék, 2002, 10. Bevezetés a fuzzy-elvű szabályozásokba, 172-173 p.
- 6: Ferenczi Ödön: Teljesítmény szabályozó áramkörök, Műszaki könyvkiadó, Budapest, 1981
- 7: Hajdú Bálint: Teljesítményelektronika In Tatár József (szerk): *Elektronikus áramkörök és ipari elektronika*, Műszaki könyvkiadó, Budapest, 1991 126-192 p.
- 8: Ferenczi Ödön: Kapcsolóüzemű tápegységek, Műszaki könyvkiadó, Budapest, 1978, I. rész
2.1. Modulációs eljárások

Internetes források

- I.1: Kóczy T. László, Tikk Domonkos (2001): Fuzzy rendszerek [on-line] Oktatási Hivatal, Budapest, URL:
<http://www.tankonyvtar.hu/hu/tartalom/tkt/fuzzy-rendszerek-fuzzy/adatok.html> (2017.06.02.)
- I.2: Dr. Fodor Dénes, Speiser Ferenc (2014): Autóipari beágyazott rendszerek [on-line] Pannon egyetem, URL: http://moodle.autolab.unipi-pannon.hu/Mecha_tananyag/autoipari_beagyazott_rendszerek/index.html (2017.06.02.)
- I.3: Fodor Attila, Vörösházi Zsolt (2011): Beágyazott Rendszerek és programozható logikai eszközök [on-line], Typotex URL:
http://www.tankonyvtar.hu/hu/tartalom/tamop425/0008_fodorvoroshazi/Fodor_Voroshazi_Beagy_0903.pdf (2017.06.02.)

I.4: http://www.engineeringtoolbox.com/specific-heat-solids-d_154.html (2017.06.02.)

I.5: ChibiOS/RT 3.0 The Ultimate Guide Chapter 4 URL:

<http://www.chibios.org/dokuwiki/doku.php?id=chibios:book:architecture> (2017.06.02.)

I.6: ChibiOS/RT 4.0.0 Reference Manual

URL: <http://chibios.sourceforge.net/docs3/rt/index.html> (2017.06.02.)

I.7: ChibiOS/HAL 4.0.0 Reference Manual

URL: <http://chibios.sourceforge.net/docs3/hal/index.html> (2017.06.02.)

I.8: µGFX wiki URL: https://wiki.ugfx.io/index.php/Main_Page (2017.06.02.)

I.9: STLINK Compiling <https://github.com/texane/stlink/blob/master/doc/compiling.md> (2017.06.02.)

Adatlapok

A.1: UM1907 User manual Discovery kit for STM32F7 Series with STM32F746NG MCU

URL:

http://www.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf (2017.06.02.)

A.2: Analog Devices ADT7410 adatlap URL: <http://www.analog.com/media/en/technical-documentation/data-sheets/ADT7410.pdf> (2017.06.02.)

A.3: Diotec BC817 tranzisztor adatlap

URL: http://diotec.com/tl_files/diotec/files/pdf/datasheets/bc817 (2017.06.02.)

A.4: Isocom MOC3062 adatlap URL:

http://www.isocom.com/images/stories/isocom/isocom_new_pdfs/MOC3060-63_Data_Sheet.pdf (2017.06.02.)

A.5: ST BTA12 adatlap URL:

<http://www.st.com/content/ccc/resource/technical/document/datasheet/aa/8c/87/f6/18/fc/46/93/CD00002267.pdf/files/CD00002267.pdf/jcr:content/translations/en.CD00002267.pdf> (2017.06.02.)

A.6: Vigortronix VTX-214-010-107 adatlap URL:

<http://www.farnell.com/datasheets/1960105.pdf> (2017.06.02.) Mellékletek 1. számú melléklet

8. Mellékletek

1. számú melléklet

Az elkészült tesztberendezés fotói



1. kép Készülékház



2. kép Tesztberendezés



3. kép Olivetti mátrix nyomtató



4.kép Viasztároló edény



5. kép A szenzort tartalmazó rézcső

2. számú melléklet

Kinyomtatott sterilizálási eredmények

3. számú melléklet

CD melléklet tartalma:

- A dolgozat pdf formátumban.
- Forráskódok.
- Nyáktervek KiCAD formátumban.
- Az eredményeket feldolgozó script, eredmény és log fájlok.

Nyilatkozat

Alulírott Stercz György a PTE-MIK hallgatója kijelentem, hogy a szakdolgozatomat nem megengedett segítség nélkül, saját magam készítettem és csak az Irodalomjegyzékben megadott forrásokat használtam fel. minden olyan részt, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból vettettem, egyértelműen, a forrás megadásával megjelöltem.

Pécs, 2017.06.09.

.....

Stercz György