

**ORSZÁGOS TUDOMÁNYOS  
DIÁKKÖRI KONFERENCIA  
DOLGOZAT**

**Tráser Miklós László**

**2023.**



**SZEGEDI TUDOMÁNYEGYETEM**  
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS  
**MÉRNÖKI KAR**

***FUZZY LOGIKÁN ALAPULÓ HŐMÉRSÉKLET  
SZABÁLYOZÓ RENDSZER MEGVALÓSÍTÁSA***

***ORSZÁGOS TUDOMÁNYOS DIÁKKÖRI KONFERENCIA  
DOLGOZAT***

***Tráser Miklós László***

*Mechatronikai mérnök BSc*

*hallgató*

*Témavezető:*

***Dr. Sarcevic Péter***

*főiskolai docens*

***SZEGED***

***2023.***

## ***ÖSSZEFOGLALÁS***

A dolgozat célja, egy olyan, a fuzzy logikán alapuló hőmérséklet szabályzó rendszer alkotása, mely képes mikrokontrolleren futni. A rendszer figyeli a külső és belső környezetet is és ezek alapján változtatja az ablak és a fűtő-hűtő rendszer állapotát, a megfelelő belső körülmények eléréséhez. Az alap döntéshozó logikát a MATLAB Fuzzy Toolbox használatával hoztam létre, majd ezzel a logikával megegyező működésű kódot készítettem Arduino IDE platformon. Ilyen módon a döntéshozó rendszer futtatható volt mikrokontrolleren. A kódnak a helyes működését LabVIEW környezetet használva bizonyítottam. Maga a logika viselkedését MATLAB Simulink szimuláció segítségével vizsgáltam, melyhez viszonyítási méréseket is végeztem. A végeredmény azt mutatja, hogy a rendszer megfelelően működik és képes tartani a kijelölt hőmérsékletet, amellet, hogy a levegő minősége is megfelelő a szobában, a legkülönbözőbb külső hőingások ellenére is.

## ***ABSTRACT***

The goal of this thesis is to develop a fuzzy logic-based thermal control system that can operate on a microcontroller. The system monitors the indoor and outdoor environments, and based on this information, it controls the heating-cooling system and the state of the windows. I developed the fundamental decision-making logic using the MATLAB Fuzzy Toolbox, and subsequently, I wrote an Arduino IDE code that operates in a similar manner. This enables me to execute the decision-making logic on a microcontroller. I used the LabVIEW platform to verify that the code is working correctly. For the examination of the base logic, I conducted a simulation in MATLAB Simulink, for which comparative measurements were made. The results show that the system works correctly and can maintain the designated temperature and air quality, despite various external temperature fluctuations.

## **TARTALOMJEGYZÉK**

|             |                                                                              |           |
|-------------|------------------------------------------------------------------------------|-----------|
| <b>1.</b>   | <b>BEVEZETÉS .....</b>                                                       | <b>3</b>  |
| <b>2.</b>   | <b>SZAKIRODALMI ÁTTEKINTÉS.....</b>                                          | <b>4</b>  |
| <b>2.1.</b> | <b>A hőmérséklet szabályozásra alkalmazható technológiák és módszerek ..</b> | <b>4</b>  |
| 2.1.1.      | PID szabályzó.....                                                           | 4         |
| 2.1.2.      | Mesterséges neurális hálózat .....                                           | 5         |
| 2.1.3.      | A modell alapú prediktív szabályozás .....                                   | 5         |
| 2.1.4.      | A genetikus algoritmusok .....                                               | 5         |
| 2.1.5.      | Összegzés .....                                                              | 6         |
| <b>2.2.</b> | <b>A fuzzy következtetési rendszerek alapjai és lehetőségei.....</b>         | <b>6</b>  |
| 2.2.1.      | Fuzzy logika bevezetés.....                                                  | 6         |
| 2.2.2.      | Boole algebra.....                                                           | 7         |
| 2.2.3.      | Fuzzy logika története .....                                                 | 7         |
| 2.2.4.      | Fuzzy logika alapgondolata.....                                              | 7         |
| 2.2.5.      | Boole algebra és fuzzy halmazok összehasonlítása .....                       | 8         |
| <b>2.3.</b> | <b>Fuzzy logika működési lépései .....</b>                                   | <b>8</b>  |
| 2.3.1.      | Bemeneti függvények.....                                                     | 8         |
| 2.3.2.      | T-operátorok .....                                                           | 9         |
| 2.3.3.      | Szabályok .....                                                              | 10        |
| 2.3.4.      | Defuzzifikáció .....                                                         | 12        |
| <b>2.4.</b> | <b>Fuzzy logika használta hőmérséklet szabályzásra .....</b>                 | <b>12</b> |
| <b>3.</b>   | <b>MEGVALÓSÍTÁS.....</b>                                                     | <b>14</b> |
| <b>3.1.</b> | <b>Fuzzy logika elkészítése MATLAB segítségével .....</b>                    | <b>14</b> |
| 3.1.1.      | Bevezetés.....                                                               | 14        |
| 3.1.2.      | Alapbeállítások.....                                                         | 15        |
| 3.1.3.      | Bemenetek .....                                                              | 17        |
| 3.1.4.      | Kimenetek .....                                                              | 19        |
| 3.1.5.      | Szabályok .....                                                              | 20        |
| 3.1.6.      | Felületek.....                                                               | 20        |
| <b>3.2.</b> | <b>Fuzzy logika leprogramozása, mikrokontrolleren futtatása .....</b>        | <b>21</b> |
| 3.2.1.      | Mikrokontrollerek .....                                                      | 21        |
| 3.2.2.      | Arduino.....                                                                 | 22        |
| 3.2.3.      | Programozás .....                                                            | 22        |
| 3.2.4.      | Ki és bemeneti függvények .....                                              | 23        |
| 3.2.5.      | Az y érték számítás.....                                                     | 24        |

|             |                                                          |           |
|-------------|----------------------------------------------------------|-----------|
| 3.2.6.      | Szabályok .....                                          | 25        |
| 3.2.7.      | Kimeneti fuzzy, majd számbeli eredmény megalkotása ..... | 26        |
| 3.2.8.      | Használt funkciók.....                                   | 27        |
| <b>3.3.</b> | <b>LabVIEW szimuláció .....</b>                          | <b>28</b> |
| 3.3.1.      | Hardware in the loop .....                               | 28        |
| 3.3.2.      | LabVIEW .....                                            | 29        |
| 3.3.3.      | Block Diagram .....                                      | 29        |
| 3.3.4.      | Front Panel .....                                        | 31        |
| <b>3.4.</b> | <b>MATLAB szimuláció.....</b>                            | <b>33</b> |
| 3.4.1.      | A hőtani modell .....                                    | 34        |
| 3.4.2.      | Külső hőmérséklete .....                                 | 35        |
| 3.4.3.      | Levegő minőség .....                                     | 35        |
| 3.4.4.      | Várható eső.....                                         | 37        |
| 3.4.5.      | Eredmények.....                                          | 37        |
| 3.4.6.      | Nyári nap .....                                          | 38        |
| 3.4.7.      | Téli nap.....                                            | 39        |
| 3.4.8.      | Mérés.....                                               | 40        |
| <b>4.</b>   | <b>ÖSSZEFOGLALÁS .....</b>                               | <b>45</b> |
| <b>5.</b>   | <b>FELHASZNÁLT IRODALOM.....</b>                         | <b>47</b> |

## 1. BEVEZETÉS

Az ember számára megfelelő környezeti körülmények megteremtése létfontosságú a fizikai és mentális egészség, valamint a produktivitás szempontjából rövid és hosszútávon is. Ezen körülmények például a megfelelő hőmérséklet, páratartalom, tiszta levegő és egyéb kényelmi tényezők biztosítása.

A kutatás célja egy fuzzy logikán alapuló hőmérséklet és levegőminőség szabályzó rendszer alkotása, mely egy mikrokontrolleren képes futni. A rendszer figyeli a kinti és a benti hőmérsékletet, a benti levegő minőségét, valamint a legközelebbi várható esőt. Ezen paraméterek alapján határozza meg az ablak kinyitását és a fűtő-hűtő rendszer állapotát, olyan módon, hogy a benti hőmérséklet egy adott érték körül mozogjon, és a levegő minőség is megfelelő legyen. A tervezés során alapvető cél volt, hogy a benti célhőmérséklet elérése, ha erre lehetőséget adnak a kinti és a benti hőmérséklet értékek, az ablaknyitással legyen elérve. Valamint, hogy ne történjen az ablak nyitása és a fűtő-hűtő rendszer bekapcsolása egyszerre.

A dolgozat első felében a témával kapcsolatos szakirodalmi háttér áttekintése történik. Ezt követően bemutatom a fuzzy logika elkészítését a MATLAB Fuzzy Toolbox segítségével. Ezzel a logikával megegyező eredményt adó Arduino IDE-ban megírt kódot részletezem a következő egységben. Ezután a kód helyes működését egy LabVIEW környezetben létrehozott szimulációval bizonyítom. Ezt követően a logika valós környezetben történő viselkedésének vizsgálatához egy MATLAB Simulink funkcióban létrehozott szimulációt készítettem, amely során használt hőtani modell elkészítéséhez méréseket is végeztem. Ennek a mérésnek a leírásával fejezem be a dolgozatot, mely után az összegzés következik.

## 2. SZAKIRODALMI ÁTTEKINTÉS

### 2.1. *A hőmérséklet szabályozásra alkalmazható technológiák és módszerek*

Napjainkra számtalan szabályzási technológia elérhető szinte minden rendszerhez, így a fűtő - hűtő rendszerekhez is. Mindegyik módszernek meg van az előnye és hátránya a többihez képest, ezért nincs tökéletes módszer a probléma megoldására. Legfőképpen az adott szituációhoz leginkább illő módszert kell kiválasztani és alkalmazni.

A HVAC rendszerek, melyek magukba foglalják a fűtésért, szellőztetésért és légkondicionálásért felelős épületgépészeti berendezéseket. A rövidítés is ezen tulajdonságok angol megfelelőiből adódik. Ezen eszközöket az alkalmazás célja szerint két csoportba lehet sorolni. Az első csoportban olyan eszközök találhatók, melyek az ember számára kellemesebb környezetet teremtenek, míg a második csoportban az ipari felhasználást lehet sorolni. Számtalan kutatás bizonyítja, hogy megfelelő környezet pozitívan befolyásolja az ember egészségét és produktivitását is. A legfontosabb környezeti paraméterek a hőmérséklet és a levegő minőség, utóbbiba többek között beletartozik a levegő oxigén szintje és a páratartalom is. (Zhang, 2014)

#### 2.1.1. *PID szabályzó*

A HVAC rendszerek, legegyszerűbb és legmegbízhatóbb vezérlését PID szabályzóval lehet elérni. Az 1930-as években megjelent technológiát ma is számtalan eszközben használják, ugyanakkor ma már számos, újabb szabályzási módszer létezik. Ezek képesek a nem lineáris karakterisztikájú vagy bizonytalan zavar tényezővel rendelkező rendszerek irányítására is. (Homod, Mahlia, & Mohamed, 2009)

A komplex több változós rendszerek szabályzására hoztak létre az úgynevezett self-tuning, magyarul önbeállító szabályzókat, melyek képesek a dinamikusan változó körülményekhez igazodni. Ezen eszközök hátrányai, hogy a valós komplex rendszer alapján pontos modellt kell készíteni a megfelelő működéshez, amelyhez a rendszer pontos ismerete szükséges. (Marzuki , Omatu, & Yusof, 1995)

Az újabb kutatások arra irányulnak, hogy olyan irányítástechnikai rendszereket hozzanak létre, melyekhez nem szükséges komplex modell elkészítése. Ilyen rendszerek a mesterséges intelligenciára épülhetnek, például a neurális hálóra vagy akár a fuzzy logikára. A dolgozat során az utóbbi módszert alkalmazom, de először bemutatom a több lehetséges szabályzási logikát is.

### **2.1.2. *Mesterséges neurális hálózat***

A mesterséges neurális hálózat, angolul artificial neural networks (ANN), egy olyan rendszer, amely az emberi idegrendszer működéséből merített inspirációt alkalmaz az alpműködési elvében. Szerkezete sok szempontból hasonlít a biológiai rendszerre. Például, a mesterséges neurális hálózat nagyszámú alapegységből áll, amelyeket neuronoknak nevezünk. A neuronok rétegekbe rendeződnek, az egy rétegben lévő neuronok nem kapcsolódnak egymáshoz, csak a közvetlen szomszédságban lévő réteg neuronjaival lépnek kapcsolatban.

Fontos tulajdonsága a rendszernek, hogy nem szükséges az irányítani kívánt rendszer pontos ismerete, csak a bemeneti paraméterek és a változók közötti kapcsolatot vizsgálja a korábbi eredmények alapján. Ennek következtében a rendszer képes működése során szerzett tapasztalatok alapján javítani saját hatékonyságát. Egy megfelelően beállított PID szabályzóhoz képest is jól tud működni a ANN alapuló szabályzó rendszer, de a tanulási sebességét jól kell megválasztani. (Molnár, 2019) (Hu és mtsai. 2005)

### **2.1.3. *A modell alapú prediktív szabályozás***

A modell alapú prediktív szabályozás vagy angolul model predictive control (MPC) egy megfelelő irányítástechnika módszer, olyan esetekben, mikor a szabályozni kívánt rendszer matematikai modellje ismert. A módszer alapgondolata, hogy egy rendszer pontos dinamikai modelljét és a rendszer pillanatnyi állapotát ismerve, bármelyik pontra vetítve meg lehet határozni a rendszer állapotát. Az MPC működésének lényege, hogy egy adott pontban valamekkora idő intervallumra előre meghatározza a rendszer várható állapotát, majd ezen intervallumon optimalizálja a szabályzást. Ezen optimalizált szabályzás első lépését végrehajtja és a következő pontba lép, ahol újrakezdi a folyamatot. Az MPC legjobb lineáris modellel vagy korlátokkal rendelkező rendszerek szabályzására, de az alap logika továbbfejlesztett változatai képesek komplexebb rendszerek szabályzására is. (Kardos & Kutasi, 2019) (Vörös & Várszegi, 2017)

### **2.1.4. *A genetikus algoritmusok***

A genetikus algoritmus, angolul genetic algorithm, (GA) egy olyan optimalizálási módszer, melynek működése hasonlít a természetben megtalálható

evolúcióhoz. Lényegében egy irányított véletlenszerű keresés, mely során a logika a jobb eredményt adó eredmények irányába irányítja a keresést. Minden egyes újabb generációval jobb eredményt ad a módszer. Az első ilyen típusú rendszert 1970-es években John Holland hozta létre.

Egyszerűbb rendszerek esetén akár pár tucat generáció is megadhatja megfelelő közelítéssel a kívánt végeredményt, míg összetettebbekhez akár több száz is kellhet. A kellő generációk meghatározására nincs egyértelmű módszer. A módszer nagy előnye, hogy képes olyan rendszerek optimális szabályzását meghatározni, melyeknek nem ismert a struktúrája vagy más irányítástechnikai módszerrel nem kezelhető, mely ennél a módszernél gyorsabb egyszerűbb lenne. (Salamon , 2003) (Zhang, 2014)

### **2.1.5. Összegzés**

A legtöbb HVAC rendszer PID szabályzóval van felszerelve, mivel ez a legrégebbi, legegyszerűbb technológia. Ugyanakkor komplexebb rendszerek szabályzására már nem megfelelően alkalmas, ilyenek lehetnek akár a több be- és kimenettel rendelkező rendszerek, angolul Multiple Input, Multiple Output (MIMO), vagy a nem lineáris karakterisztikájúak. Az is problémát jelent, ha a rendszer paraméterei idővel változnak.

A fentebb felsorolt különböző irányítástechnikai módszerek közül több is, és a következő fejezetben bemutatásra kerülő fuzzy logika is, a lágy számítási, angolul soft computing, (SC) módszerek közé sorolhatóak. Ezen módszerek együttes használatai nagymértékben elterjedtek, tehát elmondható, hogy inkább kiegészítik egymást mintsem kiszorítják. (Zadeh L. , 1996) (Yager & Zadeh, 1992) Gyakran használják a fuzzy logikát és a PID szabályzót együtt, például a szabályzó hibáinak kezelésére, vagy akár az genetikus algoritmusokat modell alapú prediktív szabályozók pontos hangolására. Mindegyik módszernek megvan az előnye a másikkal szemben, és az adott helyzeten múlik, hogy melyik szabályozásra alkalmazható technológia magában vagy más módszerrel együtt a legalkalmasabb a cél elérésére.

## **2.2. A fuzzy következtetési rendszerek alapjai és lehetőségei**

### **2.2.1. Fuzzy logika bevezetés**

Az emberiség régóta szeretne számtalan komplex feladatot automatizálni, például az autóvezetést. Érdekes módon szinte minden felnőtt ember meg tud tanulni

vezetni, de több évtizedes fejlesztés után sincs még teljesen önvezető autó napjainkban, amely a sofőr felügyelete nélkül teljesen magától képes vezetni bármilyen forgalomban. (Kóczy & Tikk, 2001) Az ilyen és ehhez hasonló összetett problémák megoldására merült fel az igénye, az emberi gondolkodásra hajazó döntéshozó rendszer megalkotásának. Erre a hagyományos bool-algebra nem alkalmas, ellenben a korábban említett lágy számítási módszerek. (Pratihari, 2013)

### **2.2.2. Boole algebra**

A logikai algebra alapjait már Arisztotelész fektette le és a századok során számtalan matematikus továbbgondolta. George Boole angol matematikus érte el ezen a téren a legfontosabb eredményeket, ezért őt szokás a matematikai logika megalkotójának nevezni. Ő építette fel a szimbolikus logika első rendszerét és bevezette ezek jelölését. A mai számítógépek is ezen logikán alapulva működnek. A Boole-algebra olyan mennyiségek közötti kapcsolatot írja le, melyek értéke csak kétféle értéket vehet fel. Ez az érték vagy nulla vagy egy, más értelenben vagy igaz vagy hamis az érték. (Hailperin, 1976)

### **2.2.3. Fuzzy logika története**

A fuzzy logika alapjait Zadeh fektette le, majd ezt követően 1975-ben Mamdani egyszerűsítette le Zadeh modellét. Így ugyan kisebb rugalmasságot biztosított, de a modell kezelésének számításiigénye drasztikusan csökkent. (Mamdani & Assilian, 1975) A végeredmény egy teljesen új irányítási módszer, mely számtalan előnnyel rendelkezik. Ezek között szerepel, hogy a fuzzy logika képes matematikai eszközökkel kezelni a bizonytalanságot, amely információhiányból, pontatlanságból vagy esetleg kétértelműségből származhat. (Russell & Norvig, 1995)

### **2.2.4. Fuzzy logika alapgondolata**

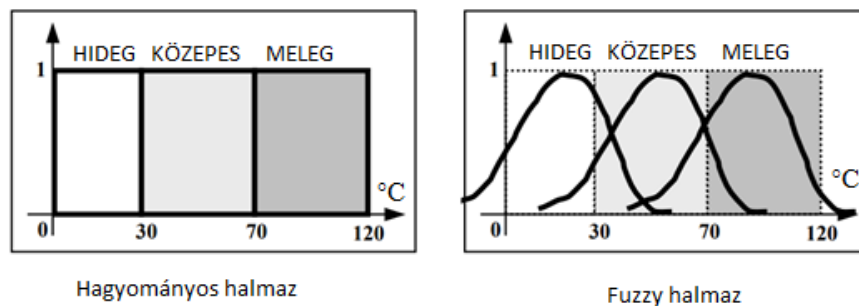
Ha belegondolunk a való életben még számtalan olyan probléma létezik, melynek megoldását a matematika nyelvére átfordítani a hagyományos boole algebra szerint nem lehet vagy nagyon körülményes. Ilyen módon automatizálni se lehet, pedig a felnőtt emberek többsége könnyűszerrel meg tudja oldani a feladatot. Erre a legegyszerűbb példa a már korábban említett gépkocsi vezetése. (Kóczy & Tikk, 2001) A Fuzzy logika működésének egyik fő eleme, hogy hasonlóan működik, mint az emberi agy. (Zilouchian & Jamshidi, 2001)

### 2.2.5. Boole algebra és fuzzy halmazok összehasonlítása

A fuzzy logika legnagyobb különbsége a boole algebrához képest, hogy részleges igazságokat is megenged, tehát eredménye nem csak nulla vagy egy lehet, lehet e két érték közötti eredmény is. Ez a tulajdonsága a nagy bonyolultságú műszaki problémák kezelését teszi lehetővé, ez volt a legfőbb motiváló erő a fuzzy logika és a fuzzy halmazelmélet kezdeti megalkotása során. (Zadeh L. , 1965)

## 2.3. Fuzzy logika működési lépései

### 2.3.1. Bemeneti függvények

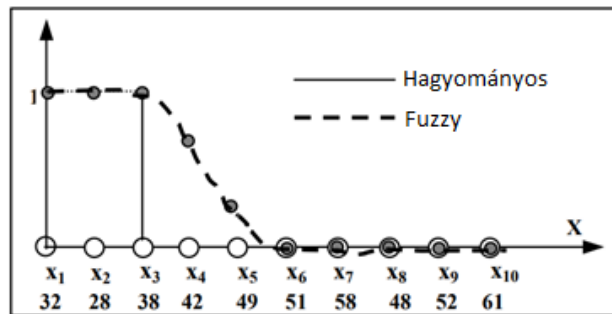


1. ábra Hagyományos halmaz és fuzzy halmaz

Hagyományos logikához képest a fuzzy logika lehetőséget ad a halmazok közötti átmenetre, más néven, hogy egy adott  $x$  érték csak részben tartozzon az egyik vagy másik halmazhoz. A hagyományos logika során például egy adott hőmérsékletet csak egy halmazba tudjuk beosztani egyszerre, tehát vagy hideg, vagy közepes, vagy meleg (1. ábra). Ellenben a fuzzy logikát használva a 40 °C-ra azt mondhatjuk, hogy 0,5 mértékben hideg és 0,7 értékben langyos. Valójában minden egyes bemeneti értékre leírható, hogy melyik halmazban milyen mértékig van benne, ez az érték 0 és 1 között változhat.

Például: 40 °C: 0,5 hideg, 0,7 langyos és 0,0 meleg

100 °C: 0,0 hideg, 0,0 langyos és 1,0 meleg



2. ábra Egy halmazba tartozó értékek

Ha hagyományos halmaz esetén csak a  $x_1$ ,  $x_2$  és  $x_3$  értékek esnek bele az adott halmazba, akkor a vele megegyező fuzzy halmaz esetén ezekhez az értékekhez, hozzá jön még az  $x_4$  és az  $x_5$  is, bár utóbbi kettő csak részben (2. ábra). Az  $x_6$  és az utána lévő pontok nem részei az egyik módon létrehozott halmaznak se. (Bai & Wang, 2006)

### 2.3.2. T-operátorok

A fuzzy halmazok metszetét és unióját, gyakran t-normának és t-konormának nevezik, az utóbbit használják s-norma néven is. Gyakran együtt kezelik a két fogalmat, ugyanis axiomatikus tulajdonságaik megegyeznek. A fuzzy logika során többféle módon lehet számolni ezen művelet pár eredményeit. Zadeh által javasolt MIN/MAX módszer az egyik legelterjedtebb a gyakorlati megvalósítások közül. Ugyanakkor léteznek más elméleti vagy kísérleti módszerek, melyek bizonyos szituációkban jobban működnek. Egy fuzzy döntéshozó rendszer megalkotása kezdetén mérlegelni kell többek között a modell pontosságát, egyszerűséget és a felhasználható számítási kapacitást is, és ezek alapján kell kiválasztani, hogy milyen módon kívánjuk számítani a T-operátorokat. A lehetséges számítási módok közül a Zadeh által javasolt módszer a legegyszerűbb, népszerűsége nagymértékben ebben rejlik, (1. táblázat). (Gupta & Theory, 1989)

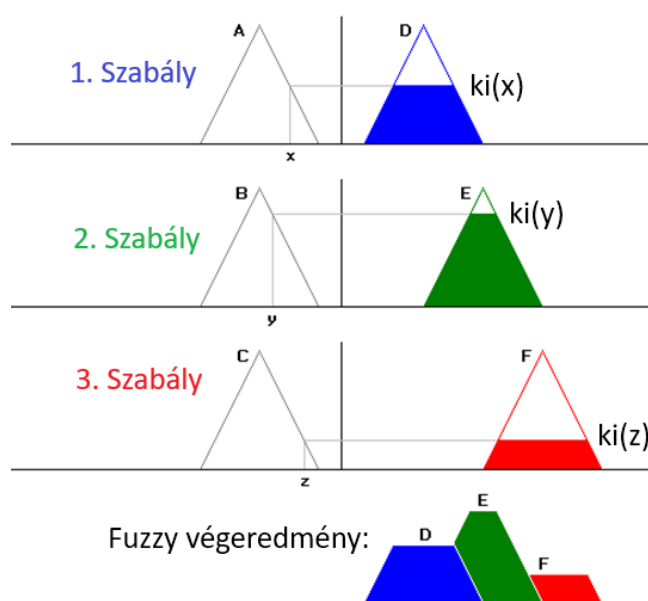
1. táblázat T-operátorok különböző számítási módjai  
Forrás: (Liu & Chen, 2016)

| Megnevezés       | t-norma                                                    | s-norma                                                               |
|------------------|------------------------------------------------------------|-----------------------------------------------------------------------|
| <b>Zadeh</b>     | $\min(x, y)$                                               | $\max(x, y)$                                                          |
| <b>Algebraic</b> | $xy$                                                       | $x + y - xy$                                                          |
| <b>Einstein</b>  | $\frac{xy}{1 + (1 - x)(1 - y)}$                            | $\frac{x + y}{1 + xy}$                                                |
| <b>Hamacher</b>  | $\frac{xy}{\gamma + (1 - \gamma)(x + y - xy)}, \gamma > 0$ | $\frac{x + y - xy - 1(1 - \gamma)xy}{1 - (1 - \gamma)xy}, \gamma > 0$ |

### 2.3.3. Szabályok

A szabályokat nyelvi módszerrel kell létrehozni, egy, „Ha..., akkor...” mondszerkezettel. Egy egyszerűsített példában három szabály szerepel, melyek egy kimeneti függvény egy-egy részhalmazának méretét határozzák meg (3. ábra). Látható, hogy a bemeneti x, y és z számértékek alapján, hogyan keletkezik a ki(x), ki(y) és ki(z) halmaz. Amilyen mértékben benne van x az A halmazban, olyan mértékben lesz a ki(x) halmaz része a D halmaznak.

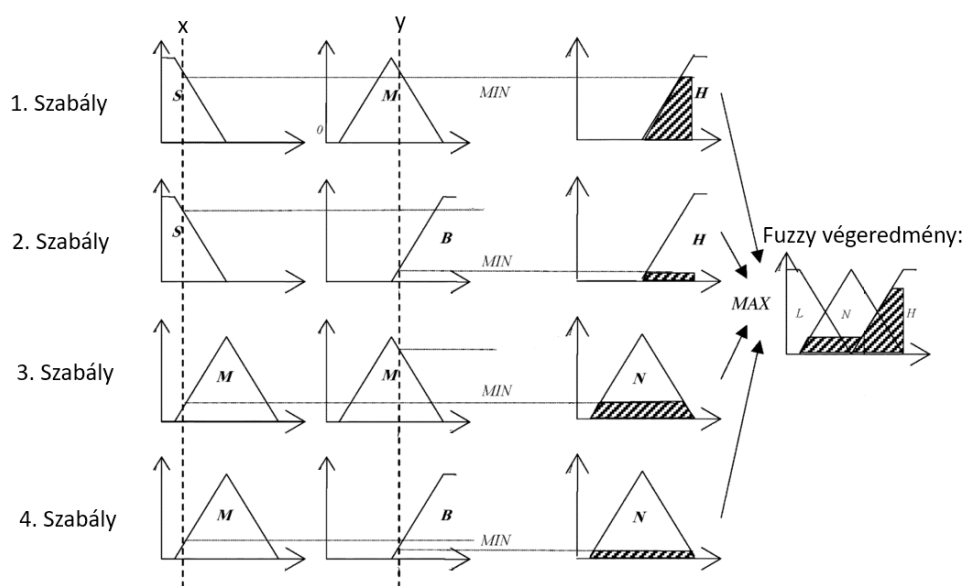
1. Szabály: **Ha** x része A-nak, **akkor** a kimenet D
2. Szabály: **Ha** y része B-nek, **akkor** a kimenet E
3. Szabály: **Ha** z része C-nek, **akkor** a kimenet F



3. ábra Fuzzy szabályok 1. példa

Ez a példa nagyon egyszerű volt, mivel egy kimeneti függvény halmazra csak egy szabály vonatkozott, ám tényleges fuzzy rendszereknél nem ez a helyzet. Az így felmerülő halmazelméleti problémát több módszerrel is meg lehet oldani, erről részletesebben a korábbi fejezetben írtam. Itt most csak a dolgozat során használt MIN/MAX módszer lépéseit részletezem, melyek a következők: ha egy szabályban több feltétel is van, mely alapján létre jön a kimeneti függvény részhalmaza, akkor a kisebb részhalmazt eredményező feltételt fogja figyelembe venni a logika. Ugyanakkor, ha egy kimeneti függvény részhalmazára két szabály is egyszerre akar hatni, a kettő közül a nagyobb halmazt eredményező szabály fog érvényesülni (4. ábra).

1. Szabály: **Ha** x része S-nek és y része M-nek, **akkor** a kimenet H
2. Szabály: **Ha** x része S-nek és y része B-nek, **akkor** a kimenet H
3. Szabály: **Ha** x része M-nek és y része M-nek, **akkor** a kimenet N
4. Szabály: **Ha** x része M-nek és y része B-nek, **akkor** a kimenet N



4. ábra Fuzzy szabályok 2. példa

Az ábrán szereplő példában összesen négy szabály van és mindegyik két feltételhez van kötve, közülük a MIN szabály alapján képződik a kimeneti függvény halmaza. Ugyanakkor az 1. és 2. szabály, valamint a 3. és 4. szabály ugyanarra a halmazra hat, itt a MAX szabály érvényesül. A végeredményt mindkét példa során, fuzzy formátumban kapjuk meg (Adrienn, 2019).

### 2.3.4. Defuzzifikáció

A folyamat fuzzy eredménye a logika alapbeállításai, a szabályok és a ki- és bemeneti függvények, valamint a bemeneti számértékek alapján adódik. Ebből többféle számítási módszer alapján lehet egy számértéket kapni. Ezek közül az egyik legnépszerűbb a center of gravity, vagy magyarul a súlyponti módszer, melynek lényege, hogy a kimeneti függvény halmazainak együttes súlypontját keresi meg és azt veszi eredménynek.



5. ábra Defuzzifikáció súlyponti módszerrel

Piros vonalak jelzik a korábbi példák során kapott fuzzy eredmények defuzzifikációját súlyponti módszerrel (5. ábra). Ahol a piros vonal metszi az x tengelyt az az érték lesz a számítási folyamat végeredménye.

A módszer nagy előnye, hogy háromszög és trapéz alakú halmazoknál könnyen számítható, ugyanakkor körültekintően kell használni. Adódhatnak olyan rossz defuzzifikáláshoz vezető szituációk, ahol az eredmény egyik halmaznak sem része. (Lukács, 2019)

### 2.4. Fuzzy logika használta hőmérséklet szabályzásra

A legtöbb fuzzy logikán alapuló hőmérséklet szabályzó rendszer, napjainkban üvegházak belső hőmérsékletét irányítja, feladatuk költségghatékony módon tartani a meghatározott belső körülményket. Ezen rendszerek több változósok és nem lineáris karakterisztikájúak lehetnek, így hagyományos szabályzási módszerekkel nehezen kezelhetőek.

A (Lafon & Balmat, 2002) által végzett tanulmányba a fuzzy logika a belső hőmérséklet és a páratartalom mellett bemeneti értéknek veszi a külső körülményeket is, míg a másik oldalon a hőmérséklet és a páratartalom állítás mellett az árnyékolás szabályzását is végzi a rendszer. A tanulmány során kétféle fuzzy szabályzót fejlesztettek, egy hagyományosan beállítottat és egy optimalizáltat. Az

eredmény alapján mindkét logikának voltak előnyei és hátrányai egymással szemben, de végső soron mind a kettő megfelelő munkát végzett az üvegház szabályzásával.

Hasonló tanulmányt végzett (Revathi & Sivakumaran, 2016), melyben fuzzy logikát használtak üvegház fűtésének szabályzására. Az általuk létrehozott alap fuzzy logikából készítettek egy részecske-raj alapú optimalizáción, angolul particle swarm optimization (PSO), átesett változatot. Az így keletkezett optimalizált fuzzy logika jobban teljesített, mint az alap változat.

Szintén ebben a témában (Benyezza és mtsai., 2018) által írt tanulmányban, páratartalom és hőmérséklet szabályzást végez a fuzzy rendszer. A logika kialakítása után azt egy Arduino mikrokontrollerre ültették át. A rendszer működését egy méretarányosan megépített üvegház makett segítségével vizsgálták. A modellt felszerelték többek között DHT11-es hőmérséklet és páratartalom mérő szenzorral, párasítóval és fűtő rendszerrel is. Az eredményük alapján a mikrokontroller képes volt a fuzzy logika futtatására és ezáltal a lekicsinyített üvegház megfelelő szabályzására.

### 3. MEGVALÓSÍTÁS

Az alapötlet egy szoba hőmérsékletének szabályozása volt mikrokontroller segítségével úgy, hogy a szabályozási folyamatba az ablak nyitását és csukását is bevonjuk. A fuzzy logikára, mint szabályzó rendszerre azért esett a választás, mert képes kellőképpen összetett több ki- és bemenettel rendelkező rendszert irányítani, ugyanakkor nem túl bonyolult a mikrokontroller számára. Az szakirodalom áttekintése során nem találtam olyan megoldást, mely az enyémhez hasonló alap logikára épülne. Emiatt először egy kezdetleges egyszerű döntéshozó rendszert hoztam létre, melyet lépésenként bővítettem, míg a célnak megfelelően nem működött.

Az alap logikának, amelyet a MATLAB-ban hoztam létre, az volt a célja, hogy a belső hőmérsékletet a lehető legtöbb esetben az ablak nyitásával próbálja szabályozni. Csak olyan körülmények között kapcsolja a fűtő-hűtő rendszert, amikor más módszerrel nem lehet elérni a kívánt belső hőmérsékletet. Egy ilyen rendszer képes költség és energia megtakarítására. Ezen alaplogika elkészítése után megállapítottam, hogy a valóságban egy átlagos nap során viszonylag kevés olyan időszak van, amikor pont úgy alakulnak a külső és belső hőmérsékletek, hogy az ablak nyitás a megoldás. Ezért gondoltam, hogy egy összetettebb rendszert készítek, mely a hőmérséklet szabályzáson kívül a belső levegő minőségére is odafigyel.

#### ***3.1. Fuzzy logika elkészítése MATLAB segítségével***

##### ***3.1.1. Bevezetés***

A MATLAB szoftvert kezdetben mátrix műveletek megoldására fejlesztették ki, de napjainkra már az egyik legelterjedtebb mérnöki számításokra használt program. Még a program első változatát Dr. Cleve Moler írta meg, az 1970-es években. (Kalechman, 2009) Manapság már az alap programon felül számos bővítmény is elérhető a szoftverhez, ilyen például a Fuzzy Logic Toolbox, mely lehetőséget ad fuzzy logika elkészítésére, valamint a MATLAB beépített grafikus szimulációs környezetében történő beillesztésére. Ezen bővítményen belül található a FIS editor, melyben a logikát létre lehet hozni. Először a fuzzifikációs modellt kell kiválasztani, mely Mamdani vagy Sugeno-ról elnevezett módszer lehet. Ezt követően a defuzzifikációs módszert, valamint a többi alapbeállítást kell kiválasztani. Miután meghatározásra kerül a fuzzy rendszer működési elve, a tényleges ki- és bemeneti függvények, valamint szabályok megalkotása a feladat. A FIS editor lehetőséget ad, a

szabályok nyelvi elemekből történő összerakására, valamint a logika működésének grafikus megjelenítésére. Az utóbbi során látszik, hogy az egyes bemeneti értékek változtatása, milyen módon változtatja a kimentei függvény geometriáját, melyből a számszerű végeredmény adódik. (Sivanandam, Sumathi, & Deepa, 2007)

### **3.1.2. Alapbeállítások**

Az emberi gondolkozás lemásolása a cél, tehát hogy az ember számára logikus döntéseket hozzon a rendszer. Ugyanakkor nem lehet túlbonyolítani a logikát számtalan szabállyal és különböző alakú bemeneti függvényekkel. A fuzzy logika alapján bármilyen görbét be tudtam volna állítani a ki- és bemeneti függvények halmazainak, alakjának, de akkor a görbe alatti terület és súlypontszámítások bonyolulttá váltak volna, amely megnehezítené a logika mikrokontrollerre való átültetését.

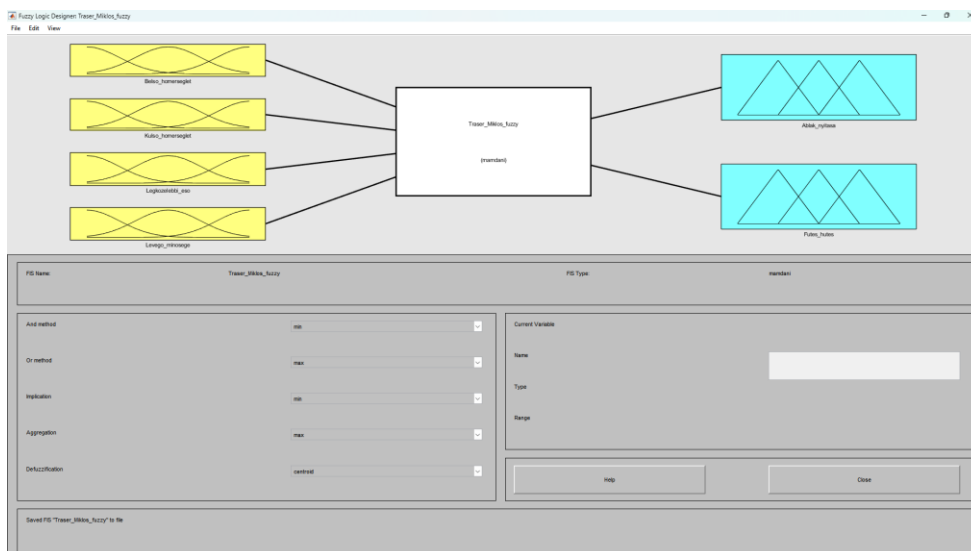
A MATLAB Toolbox többféle fuzzy logika elkészítését teszi lehetővé. Én a logika alapvető paramétereit annak figyelembevételével választottam ki, hogy lehetséges legyen a logikát leprogramozni az Arduino saját fejlesztő környezetében is, mely a C programozási nyelv alapjaira épül. Ez alapján választottam a Mamadani fuzzifikációs modellt, a Zadeh féle MIN/MAX t-operátorokat és a súlyponti defuzzifikálási módszert is. Ezeket és az összesen négy darab bemenetet és kettő darab kimenetet használtam, a fuzzy logika alap beállításához. (6. ábra).

Fuzzy logika bemenetek:

- Belső hőmérséklet
- Külső hőmérséklet
- Levegő minősége
- Legközelebbi eső

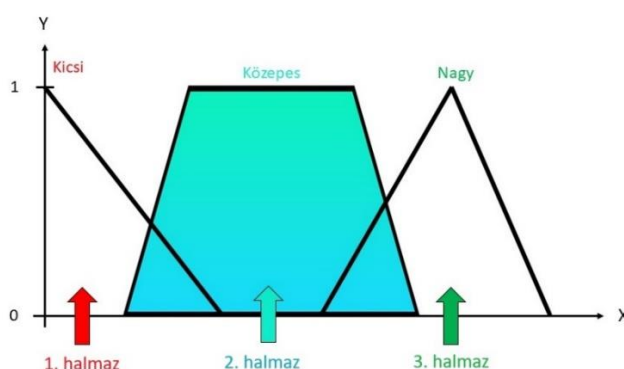
Fuzzy logika kimenetek:

- Ablak nyitásának mértéke
- Fűtés – hűtés állapota



6. ábra Fuzzy logika alapbeállításai

A legközelebb eső kivételével, mivel azt csak kettő, illetve a fűtés/hűtés mértékén, mert azt öt, az összes többi ki- és bemeneti függvényt három halmazra osztottam. Ezek alakja legtöbb esetben egy háromszögre vagy ritkább esetben trapézra emlékeztet. A dolgozat során mégis trapézként fogom kezelni őket. Ennek az oka az, hogy a trapézokat fel lehet bontani két darab derékszögű háromszögre és a köztük elhelyezkedő téglalapra. Ha az utóbbi rész területét 0-nak vesszük, akkor már meg is kapjuk a háromszöget. Így elég, ha minden ilyen halmaz alakját trapézként értelmezzük, és ez alapján végezzük el a számításokat.

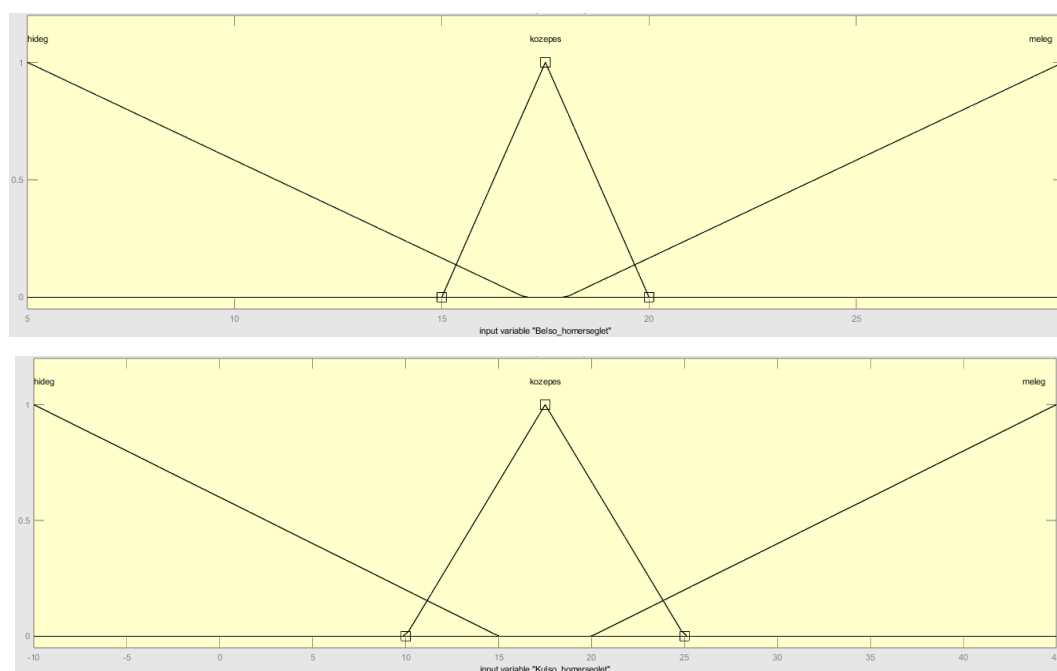


7. ábra Ki- és bemeneti függvények halmazokra osztása

A dolgozat során, mint az adott bemeneti vagy kimeneti függvény valahányadik halmazaként fogok hivatkozni rájuk (7. ábra). Méretüket, úgy választottam ki, a szabályokkal együtt, hogy megfelelően működjön a döntéshozó rendszer.

### 3.1.3. Bemenetek

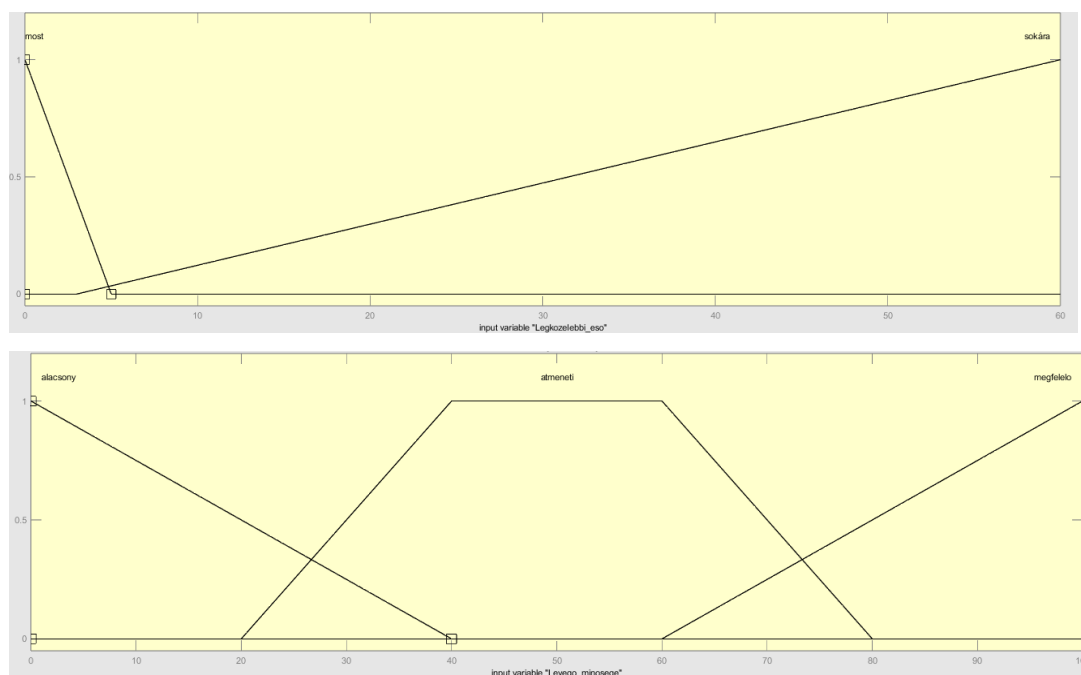
Az említett négy bemenet közül a kettő legfontosabb a külső és a belső hőmérséklet, melyeket három halmazra osztottam (8. ábra). A két bemeneti függvény halmazainak megnevezései megegyeznek „hideg”, „közepes” és „meleg”, viszont eltérő a két hőmérsékleti skála mérete. A kinti hőmérséklet a logika szerint -10 és 45 °C- fok között változhat, míg a benti érték esetében ez a skála csak 5 és 30 °C- fok közötti értéket vehet fel. Ugyanakkor ezen értékeket, úgy választottam meg, hogy mindkét esetben a középérték 17,5 °C legyen. Ennek köszönhetően az egész logika úgy lett kialakítva, hogy ezen középérték plusz mínusz körülbelül 2,5 °C tartása legyen a rendszer feladata, ha ennél melegebb van a szobában, akkor csökkenteni akarja a hőmérsékletet, ha ennél hidegebb, akkor növelni.



8. ábra Belső és külső hőmérséklet bemeneti függvénye

Fontos volt, hogy a három halmazt úgy alakítsam ki, hogy legyen a szomszédos részek között valamekkora átmenet. A pontos működés elérésének céljából lehetett volna még több halmazt létrehozni a skálán, például egy kicsit hideg vagy egy kicsit meleget. Ugyanakkor ezeknek csak akkor lenne értelme, ha szabályokba is belefoglalnám őket, amelyek még több szabályt eredményeznének, és nagymértékben bonyolítanák a számításokat, melyet a mikrokontrollernek valós időben kellene

elvégeznie. Valójában viszont a lényegi működésen igen kicsit finomítanak, ezért nem kerültek be a végső logikába.



9. ábra Legközelebbi eső és a levegő minőség bemeneti függvénye

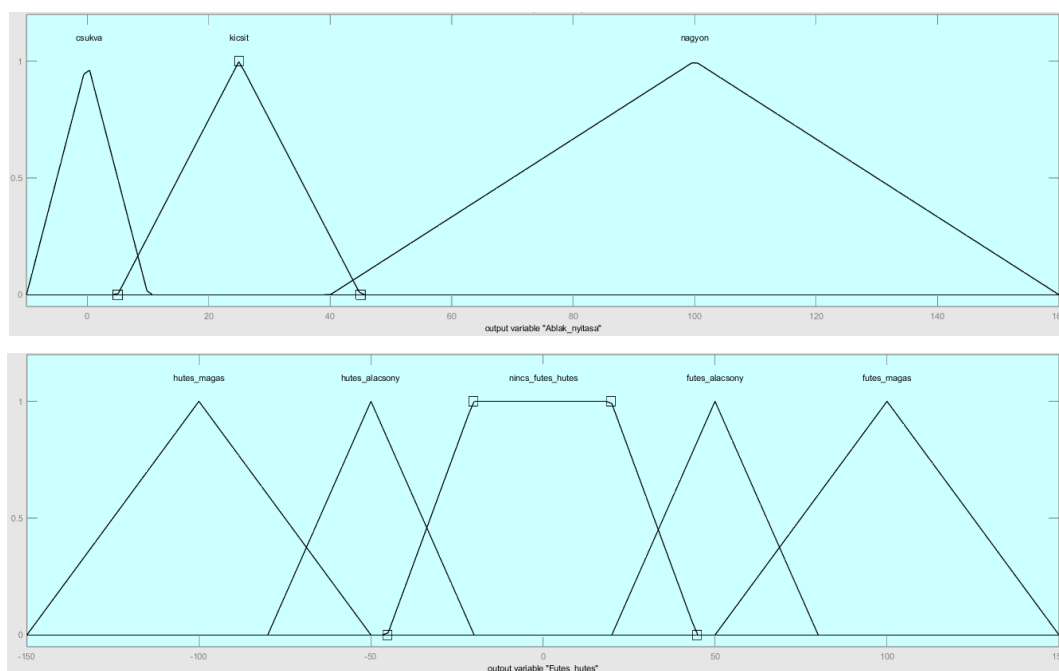
Két másik bemenet a hőmérséklet szabályzás kiegészítésére szolgál (9. ábra). A legközelebb eső skála 0-tól 60-ig terjed, ezt valójában percben kell érteni, két részre van osztva. A „most” rész 5 percig terjed a „sokára” 3 perctől 60-ig. Ez a bemenet azért fontos, mert a rendszer valós környezetben való megvalósításának legnagyobb nehézsége, hogy az ablak nyitásának mértékét is a mikrokontrollernek kell tudni állítani. Ilyen nyitórendszer kiépítése költséges és a legtöbb megoldás tetőablakok nyitására szolgál, ezért a szabályzó rendszert úgy alakítottam ki, hogy a kimeneti ablak egy tetőablak legyen. Ennek hátránya, hogy a rendszernek figyelni kell, hogy ne hagy beázzon a szoba egy eső következtében. Erre megoldást szolgál, hogy ha közeledik az eső, akkor az ablak becsukódik.

A levegő minőségét mintegy százalékos skálának kell elképzelni. Ezen a skálán a 100% érték jelenti, hogy a szobában található levegő oxigén és páratartama mennyisége tökéletes, míg a 0% esetén ezen értékek nem megfelelőek. A skála halmazokra bontását a következő módon határoztam meg, 40% alatt „alacsony”, 60% felett „megfelelo” és 20%-80% között „átmeneti” a levegő minősége. Ennek a

bemenetnek az eredménye, hogy ha alacsony a levegő minősége a szobában, akkor kinyílik az ablak, hogy friss levegőt juthasson be.

### 3.1.4. Kimenetek

A két kimenet az ablak nyitás és a fűtés vagy hűtés mértéke (10. ábra). Az előbbi egy 0-tól 100-ig terjedő skálán értelmezhető, ahol a 100-as értéket jelöli, ha teljesen nyitva van az ablak, ez esetünkben a tetőablak vízszintes állását jelenti. A nullás érték a csukott ablak, ám itt érdemes megjegyezni, hogy az esetünkben használt defuzzifikációs módszer egyik velejárója, hogy ezt az értéket sose érne el, ha skálát 0-tól 100-ig vennénk fel. Erre a problémára egyszerű megoldást jelent, ha a skálát kiszélesítjük, olyan módon, hogy a korábban szélső értéket jelölő „*nagyon*”, illetve „*csukva*” derékszögű háromszöget tükrözzük az y tengellyel párhuzamos oldalára. Az így kapott skála esetünkben -10 és 160 közé kerül. A végeredmény ugyanakkor nem lesz 0-nál kisebb vagy 100-nál nagyobb.



10. ábra Az ablak nyitás és fűtés/hűtés kimenetifüggvénye

A másik kimenet a fűtés vagy hűtés nagyságát adja meg. A korábbi módszert alkalmazzuk ezen kimenetifüggvény skáláján is, az így kapott skála -150 és 150 között fog húzódni. De a végeredmény itt is csak egy kisebb részen tud keletkezni -100 és 100 között, az előbbi a maximális hűtést az utóbbi a maximális fűtést jelenti. Ha ez a

kimeneti érték 0 akkor nem kapcsol be a legkondicionáló és így se fűtés, se hűtés nem lesz.

### 3.1.5. Szabályok

Összesen 13 darab szabályt használtam, ezek közül voltak fontosabbak és kevésbé lényegesek, melyek csak a logika finomhangolásába segítettek (11. ábra).

```

1. If (Belső_hőmérséklet is hideg) and (Külső_hőmérséklet is hideg) then (Ablak_nyitása is csukva)(Fűtés_hűtés is fűtés_magas) (1)
2. If (Belső_hőmérséklet is meleg) and (Külső_hőmérséklet is meleg) then (Ablak_nyitása is csukva)(Fűtés_hűtés is hűtés_magas) (1)
3. If (Belső_hőmérséklet is hideg) and (Külső_hőmérséklet is meleg) then (Ablak_nyitása is nagyon)(Fűtés_hűtés is nincs_fűtés_hűtés) (1)
4. If (Belső_hőmérséklet is meleg) and (Külső_hőmérséklet is hideg) then (Ablak_nyitása is nagyon)(Fűtés_hűtés is nincs_fűtés_hűtés) (1)
5. If (Belső_hőmérséklet is közepes) and (Külső_hőmérséklet is közepes) and (Legközelebbi_eső is sokára) then (Ablak_nyitása is kicsit)(Fűtés_hűtés is nincs_fűtés_hűtés) (0.05)
6. If (Levegő_minősége is megfelelő) then (Ablak_nyitása is csukva) (0.1)
7. If (Legközelebbi_eső is most) then (Ablak_nyitása is csukva) (1)
8. If (Belső_hőmérséklet is közepes) and (Külső_hőmérséklet is hideg) and (Levegő_minősége is alacsony) then (Ablak_nyitása is kicsit)(Fűtés_hűtés is fűtés_magas) (0.1)
9. If (Belső_hőmérséklet is közepes) and (Külső_hőmérséklet is meleg) and (Levegő_minősége is alacsony) then (Ablak_nyitása is kicsit)(Fűtés_hűtés is hűtés_magas) (0.1)
10. If (Levegő_minősége is alacsony) then (Ablak_nyitása is nagyon)(Fűtés_hűtés is nincs_fűtés_hűtés) (0.2)
11. If (Levegő_minősége is atmeneti) then (Ablak_nyitása is kicsit)(Fűtés_hűtés is nincs_fűtés_hűtés) (0.1)
12. If (Belső_hőmérséklet is közepes) and (Külső_hőmérséklet is hideg) then (Ablak_nyitása is csukva)(Fűtés_hűtés is fűtés_alacsony) (0.1)
13. If (Belső_hőmérséklet is közepes) and (Külső_hőmérséklet is meleg) then (Ablak_nyitása is csukva)(Fűtés_hűtés is hűtés_alacsony) (0.1)

```

11. ábra A fuzzy logika szabályai

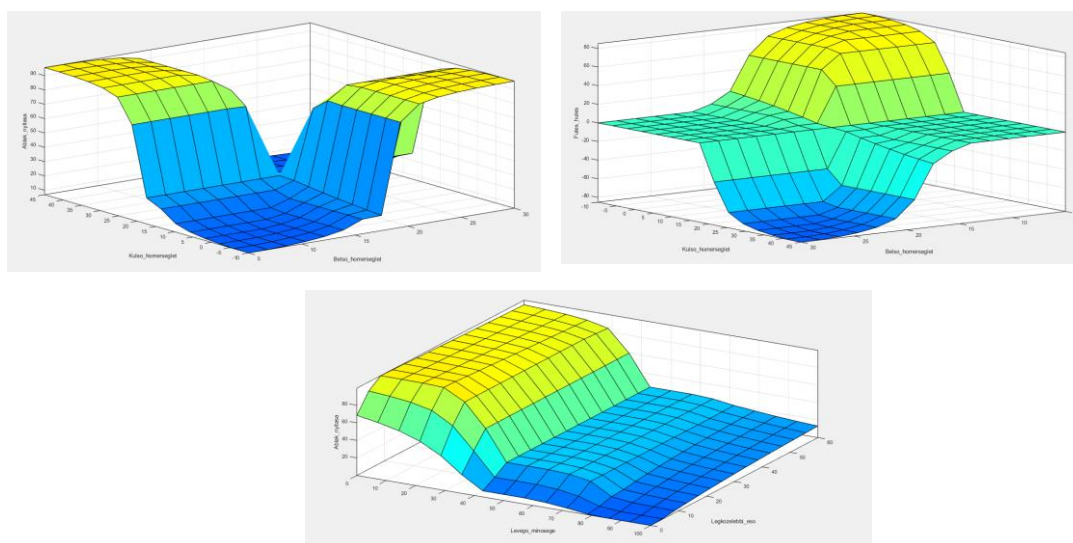
Minden szabályt 0 és 1 közötti fontossági értékkel láttam el, másnéven súlyoztam őket. Az egyessel jelölt szabályok a legfontosabbak. Ilyen például az első négy szabály, melyek a következőket tartalmazzák. Ha benti hőmérséklet hideg miközben a kint meleg van, akkor a rendszer az ablakot nyitja ki nagy mértékben és nem kapcsolja be a fűtő-hűtő rendszert. Ugyanez a folyamat zajlik le, ha fordítva van a külső és belső hőmérséklet tehát, bent van meleg is kint van hideg. Ezekkel szemben, ha kint is és bent is hideg vagy meleg van, akkor az ablak csukva marad és a fűtés vagy hűtés lesz nagy mértékű. A maradék kilenc szabály a megfelelő levegő minőséget biztosítja, figyeli a közelgő esőt és finom hangolja az alap logika működését.

A fuzzy logika előnye, hogy a szabályalkotást nyelvi eszközökkel lehet megtenni, mint ahogy ezt az ábra is mutatja.

### 3.1.6. Felületek

A MATLAB Toolbox képes az egyes bemeneti érték változását és az ennek következtében bekövetkező kimeneti értékek változásának függvényét leképezni. Ha nem csak egy bemenet hatását vizsgáljuk egy kimenetre, hanem két bemenet hatását egy kimenetre, akkor egy 3D-s felületet kapunk. A Fuzzy logika kialakítása során fontos szempont volt, hogy az egymással összefüggő be- és kimenetek által meghatározott felületek minél egyenletesebben változzanak a bemenetek változása következtében. Ilyen módon biztosítani lehet azt, hogy a kimeneti érték ne változzon

hirtelen, ugrásszerűen egyes bemenet kismértékű változása miatt. Az eredmény egy olyan irányítási rendszer, ahol a kimeneti értékek arányosan, fokozatosan változnak a bemeneti értékek változásának hatására. Ez a tulajdonság előnyös az irányított gépek, eszközök számára is, ugyanis nem hirtelen kell egy teljesen különböző állapotba lépniük. Esetünkben az ablak fokozatosan nyílik vagy csukódik, illetve a légkondicionáló lassan változtatja a fűtés vagy hűtés mértékét a hőmérséklet, a levegőminőség, továbbá a várható következő eső értékváltozása hatására.



12. ábra Felületek

A logika alapján, mint ahogy már korábban említettem, ha a kint is és bent is meleg van, akkor a rendszer a legkondicionáló segítségével hűti le a szobát, és nem nyitja az ablakot. Ugyanez történik, ha mindkét helyen hideg van, csak abban az esetben, fűtés a rendszer válasza. Ezekkel szemben, ha kint meleg és bent hideg vagy fordítva van, akkor a végeredmény az ablak kinyitása és a legkondicionáló használaton kívül hagyása (12. ábra). Az ablak nyitásának mértéke akkor csökken a várható eső hatására, ha az 5 percen belül fog elkezdni esni. Ezzel szemben, ha a levegő minőség 40% alá esik, akkor kezd kinyílni az ablak, hogy friss levegő jöhessen be a szobába.

### 3.2. Fuzzy logika leprogramozása, mikrokontrolleren futtatása

#### 3.2.1. Mikrokontrollerek

A mikrokontrollerek ideálisak olyan problémák megoldására, ahol szükség van az intelligens rendszerre, de felesleges egy teljes személyi számítógép használata, tárhellyel, perifériákkal és monitorral. A mikro szó a méretére, míg a controller az

egyik legjelentősebb felhasználási módjára utal. Manapság szinte mindenhol megtalálhatóak, szinte az összes olyan eszközben, amelyek mérnek, tárolnak, számolnak adatokat vagy információt jeleznek ki. (Axelson, 1994) Az egyszerű működéséből fakadóan a megbízhatósága kimagasló és az ára is nagyon kedvező, egy személyi számítógéphez képest, viszont programozni nem igen lehet őket magában, ehhez szükség van egy személyi számítógépre. (Panda és mtsai., 2006) Napjainkban számtalan gyártó készít különböző tulajdonságokkal, árral, és felhasználási céllal rendelkező mikrokontrollereket, ezek közül az egyik legnépszerűbb márka az Arduino.

### **3.2.2. Arduino**

Az Arduino egy nyílt forráskódú platform, mely kisebb elektronikai rendszerek létrehozására és programozására ad lehetőséget. Massimo Banzi és Casey Reas hozta létre az Arduino platformot 2005-ben, melynek alapjait az egy évvel korábban Hernando Barragan által készített szintén nyílt forrás kódú platform adta, a Wiring. A platform két fő részre osztható: a fejlesztő környezetre és a mikrokontrollerre. A kezdeti cél is egy könnyen tanulható és kezelhető szoftver és eszköz volt, mely így oktatási célokat is be tud tölteni. Ennek köszönhetően napjainkban széles körben elterjedt, és nagy népszerűségen örvend a platform. A mikrokontrollerre való új kód töltése egy USB kábelon keresztül történik, melyet a számítógépbe kell bedugni. A kódot az Arduino IDE-ban lehet elkészíteni, mely egy a C++ programozási nyelv egyszerűsített változatát használja. (Banzi, 2009)

Nagy népszerűségének köszönhetően, számtalan segédanyag elérhető az interneten, így szinte bármilyen problémára lehet találni megoldást. Az Arduino mikrokontroller és más programok összekötése is egy könnyen megoldható feladat, ilyen módon akár programozni is tudjuk más környezetek használásával. Ilyen környezet lehet a LabVIEW grafikus programozási felület. A programozáson kívül képes a grafikus környezetet megnyitni, írni és olvasni a soros kommunikációs portját az Arduinonak. (Schwartz & Manickum, 2015)

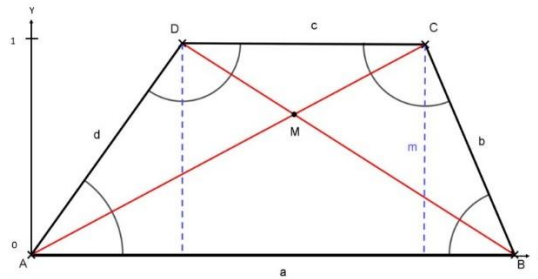
### **3.2.3. Programozás**

A fuzzy döntéshozórendszer valójában terület, és esetünkben súlypont számítások sokasága, ezt kihasználva le lehet programozni a logikát bármilyen programozási környezetben. Mivel az én célom Arduino mikrokontrolleren való futtatás, ezért én az Arduino IDE-t használtam programírási környezetnek.

A kódot 13 részre osztottam, minden rész egy lépést jelent a számítás menetében. Továbbá az alap számításon kívül a program elején és végén található egy-egy rész. Az előbbi a bemeneti szöveg formátumot választja szét a megfelelő bemeneti szám értékekre, az utóbbi a kimeneti értékeket rakja össze egy szöveggé. Ezekre azért van szükség, mert a LabVIEW program csak ilyen módon tud a soros porton keresztül kommunikálni a mikrokontrollerrel, és az én célom ezen program használata a program működésének ellenőrzésére.

Az Arduino program teljes kódjának mind a 624 sora az 1. mellékletben található meg.

#### 3.2.4. *Ki és bemeneti függvények*



13. ábra Halmaz a koordináta-rendszerben

Mint ahogy korábban említettem, minden egyes ki és bemeneti függvényt felbontottam halmazokra, melyek alakja trapéz. Ha egy koordináta-rendszerben képzeljük el az egyes trapézokat, akkor, az A és B pontjának az  $y$  értéke nulla, míg a D és C pontnak egy, továbbá azt is fontos megállapítani, hogy a pontok  $x$  koordinátája a következő módon növekedik: A, D, C és B (13. ábra). A két szabályszerűséget kihasználva minden trapézt meg tudunk határozni, csupán az  $x$  koordináták alapján. Továbbá, ha a bemeneti függvény háromszögre és nem trapézra van bontva, akkor azt is könnyűszerrel fel tudjuk írni, olyan módon, hogy a D és C pontnak azonos  $x$  koordinátát adunk meg.

```

1 // 1. rész bemenet:
2 float be1_msfc[3][4]={{5.0,5.0,5.0,17.0},{15.0,17.5,17.5,20.0},{18.0,30.0,30.0,30.0}}; //belső hőmérséklet
3 float be2_msfc[3][4]={{-10.0,-10.0,-10.0,15.0},{10.0,17.5,17.5,25.0},{20.0,45.0,45.0,45.0}}; //külső hőmérséklet
4 float be3_msfc[3][4]={{0.0,0.0,0.0,5.0},{3.0,60.0,60.0,60.0},{0.0,0.0,0.0,0.0}}; //eső
5 float be4_msfc[3][4]={{0.0,0.0,0.0,40.0},{20.0,40.0,60.0,80.0},{60.0,100.0,100.0,100.0}}; //oxigén
6
7 // 2. rész kimenet:
8 float ki1[3][4]={{-10.0,0.0,0.0,10.0},{5.0,25.0,25.0,45.0},{40.0,100.0,100.0,100.0}}; //Ablak
9 float ki2[5][4]={{-150.0,-100.0,-100.0,-50.0},{-80.0,-50.0,-50.0,-20.0},{-45.0,-20.0,20.0,45.0},{20.0,50.0,50.0,80.0},{50.0,100.0,100.0,150.0}}; //Fűtés/hűtés
10 float ki1_msfc[3][4]={{0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0}};
11 float ki2_msfc[5][4]={{0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0}};
12

```

14. ábra A ki- és bemeneti függvények felvétele

Angolul a ki- és bemeneti függvényeket membership function-nek hívják, innen jönnek a kódban szereplő rövidítések (14. ábra). A függvényeket mátrix formátumban tudjuk felírni, ahol a mátrix első eleme a halmaz sorszámát jelöli egy adott függvényen belül. A második elem a halmaz alakját határozza meg, a négy pont  $x$  koordinátája által.

```

45 // 2.1 rész kimenetek eltolasa
46 for (int i=0; i<3; i++){
47     for (int j=0; j<4; j++){
48         ki1_msfc[i][j] = ki1[i][j] + 10;}}
49
50 for (int i=0; i<5; i++){
51     for (int j=0; j<4; j++){
52         ki2_msfc[i][j] = ki2[i][j] + 150;}}
53

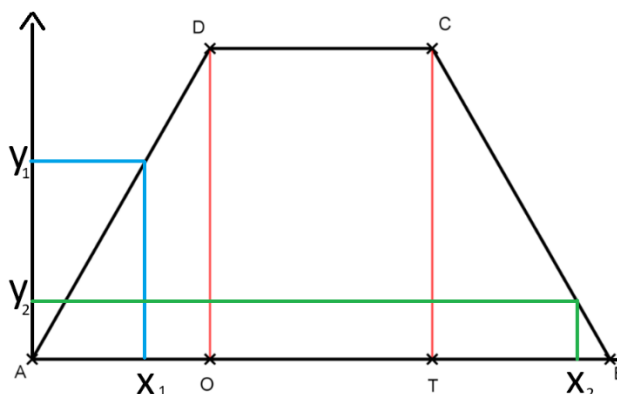
```

15. ábra A kimeneti függvények eltolása

A két kimeneti függvény halmazait eltoltam, olyan módon, hogy a teljes skála a pozitív tartományban legyen, erre a későbbi súlypont számítások miatt volt szükség (15. ábra). Az eltolás ellensúlyozása miatt ezt az értéket ki kell vonni a számítás legvégén, így a helyes eredményt kapjuk.

### 3.2.5. Az $y$ érték számítás

A 3. részben a szükséges változókat definiáltam, 4. részben a soros kommunikációs portot inicializáltam és beállítottam a sebességet 9600 szekundumra, majd a 5. részben a már korábban említett szöveg formátumot bontottam szét négy darab bemeneti szám értékre.



16. ábra Az  $x$  értékből következő  $y$  érték meghatározása a halmazokban

A 6. részben kezdődött a számítás. Először az adott bemeneti értékről kellett megállapítani, hogy milyen mértékben része az egyes halmazoknak. A bemeneti számértékek, amelyek valójában az  $x$  tengelyen elhelyezkedő pontok, alapján a hozzájuk tartozó  $y$  értéket határoztam meg. Lényegében a bemeneti függvények mátrixa segítségével határoztam meg, hogy az adott pont hol helyezkedik el a trapéz alakú halmaz  $x$  tengelyén. Ha az adott halmaz O és T pontja között helyezkedik, akkor a hozzá tartozó  $y$  érték az 1, ha A és O vagy T és B között, akkor egy arányossággal lehet számolni a 0 és 1 közötti értéket (16. ábra). Ezt az  $y$  értéket a kódban  $y_b$  változóval jelöltem és kiszámoltam mind a négy bemeneti függvény minden halmazára (17. ábra). A kapott  $y_b$  mátrix első eleme azt határozta meg, hogy hányadik bemenethez tartozik, míg a második, hogy az adott bemenet melyik halmazában milyen nagy ez a  $y$  érték. Mivel két szomszédos halmaznak van általában átfedésben levő része, ezért történik az, hogy egy bemeneti  $x$  értékhez az adott bemeneti függvényben több  $y$  érték is tartozhat.

```

94 // 6. rész y_b kiszámítása:
95 for(int i=0; i<3; i++){
96     if (be1_msf[i][0] <= xb[0] && xb[0] <= be1_msf[i][1]) {yb[0][i] = (xb[0] - be1_msf[i][0])/(be1_msf[i][1] - be1_msf[i][0]);}
97     else if (be1_msf[i][2] <= xb[0] && xb[0] <= be1_msf[i][3]) {yb[0][i] = (be1_msf[i][3] - xb[0])/(be1_msf[i][3] - be1_msf[i][2]);}
98     else if (be1_msf[i][1] < xb[0] && xb[0] < be1_msf[i][2]) {yb[0][i] = 1.0;}
99     else if (be1_msf[i][0] > xb[0] && xb[0] > be1_msf[i][3]) {yb[0][i] = 0.0;}

```

17. ábra Az  $y$  érték meghatározására példa

### 3.2.6. Szabályok

A 7. részben történik az összes 13 darab szabály, és a hozzájuk tartozó minimum számítás, valamint négy darab funkció meghívása (18. ábra). Valójában a bemeneti függvények halmazai egy-egy állapotot takarnak, például a belső

hőmérséklet bemenet első halmaza a hideg, második közepes, harmadik meleg, ugyanakkor mivel a tömbök és mátrixok első elem 0 és nem 1, így a számozás is eszerint alakul.

```

124 // 7. resz szabalyok:
125 //1. If (Belso_homerseget is hideg) and (Kulso_homerseget is hideg) then (Ablak_nyitasa is csukva)(Futes_hutes is futes_magas) (1)
126 if (be1_msf[0][0] <= xb[0] && xb[0] <= be1_msf[0][3] && be2_msf[0][0] <= xb[1] && xb[1] <= be2_msf[0][3]) {
127     ki_y[0] = (min(yb[0][0],yb[1][0]))*1;
128     Sp_ablak_kimenet[0] = Sp_ablak(ki_y[0], 0);
129     T_ablak_kimenet[0] = T_ablak(ki_y[0], 0);
130     Sp_fh_kimenet[0] = Sp_FH(ki_y[0], 4);
131     T_fh_kimenet[0] = T_FH(ki_y[0], 4);
132 }

```

### 18. ábra Szabályok számítására példa

A szabály első felében azt kell vizsgálni, hogy a bemeneti  $x$  érték az adott bemeneti függvény adott halmazának része-e, ha ez teljesül, akkor jön a szabály második fele. Itt először egy minimum keresés zajlik le, feltéve, ha több feltétel alkotja az adott szabályt, majd a kapott értékkel négy darab funkció van meghívva. Ezen funkciók határozzák meg a szabály következtében a két kimeneti függvényen megjelenő trapéz területét és súlypontját. A  $ki\_y$ -on kívül a funkciókhoz egy  $v$  értéket is meg kell határozni, ez valójában a kimeneti függvény halmazát határozza meg. Például a 18. 18. ábrán látható eset akkor fut le, ha az első bemeneti érték az első bemeneti függvény első halmazában található, valamint a második bemeneti érték a második bemeneti függvény szintén első halmazában található. Ha ez a két feltétel egyszerre teljesül, akkor jön a két korábban kapott  $yb$  érték összehasonlítása, amelyik a kisebb, azzal folytatódik a számítás. Ez a  $ki\_y$  érték fogja meghatározni a trapéz magasságát a kimeneti függvényben, melynek alapját a  $v$  érték adja meg. Ha ez az érték 0, akkor az adott kimeneti függvény első részén keletkezik a trapéz.

#### 3.2.7. *Kimeneti fuzzy, majd számbeli eredmény megalkotása*

Mind a 13 szabály elméleti síkon eredményezhet mind a két kimeneti függvényen egy különböző trapézt. Megállapítható, hogy ezek helyzete jelen esetben csak háromféle eredmény lehet az ablak nyitása kimenetnél és ötféle lehet a fűtés-hűtés kimenet esetében. Mivel az egyes kimeneti függvények ennyi halmazra vannak osztva. Maximum kereséssel meg kell állapítani, hogy az egyes kimeneti halmazok helyén keletkezett trapézok közül, melyek területe a legnagyobb, és ennek mekkora a súlypontja (19. ábra).

```

238 // 8. rész végeredény számítás
239
240 for(int i=0; i<13; i++){
241     if (isnan( T_ablak_kimenet[i])) { T_ablak_kimenet[i] = 0.0; }
242     if (isnan( Sp_ablak_kimenet[i])) { Sp_ablak_kimenet[i] = 0.0; } }
243
244     Teljes_T_ablak[0] = max(T_ablak_kimenet[11],max(T_ablak_kimenet[12],max(T_ablak_kimenet[6],max(T_ablak_kimenet[5],max(T_ablak_kimenet[0],T_ablak_kimenet[1])))));
245     if (Teljes_T_ablak[0] == 0.0) {Teljes_SP_ablak[0] = 0.0; }
246     else if (Teljes_T_ablak[0] == T_ablak_kimenet[11]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[11]; }
247     else if (Teljes_T_ablak[0] == T_ablak_kimenet[12]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[12]; }
248     else if (Teljes_T_ablak[0] == T_ablak_kimenet[6]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[6]; }
249     else if (Teljes_T_ablak[0] == T_ablak_kimenet[5]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[5]; }
250     else if (Teljes_T_ablak[0] == T_ablak_kimenet[0]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[0]; }
251     else if (Teljes_T_ablak[0] == T_ablak_kimenet[1]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[1]; }
252     else {Serial.print ("Hiba ablak_0!");}
253
254
255     Teljes_T_ablak[1] = max(T_ablak_kimenet[10],max(T_ablak_kimenet[8],max(T_ablak_kimenet[7],T_ablak_kimenet[4])));
256     if (Teljes_T_ablak[1] == 0.0) {Teljes_SP_ablak[1] = 0.0; }
257     else if (Teljes_T_ablak[1] == T_ablak_kimenet[10]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[10]; }
258     else if (Teljes_T_ablak[1] == T_ablak_kimenet[8]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[8]; }
259     else if (Teljes_T_ablak[1] == T_ablak_kimenet[7]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[7]; }
260     else if (Teljes_T_ablak[1] == T_ablak_kimenet[4]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[4]; }
261     else {Serial.print ("Hiba ablak_1!");}
262
263     Teljes_T_ablak[2] = max(T_ablak_kimenet[9],max(T_ablak_kimenet[3],T_ablak_kimenet[2]));
264     if (Teljes_T_ablak[2] == 0.0) {Teljes_SP_ablak[2] = 0.0; }
265     else if (Teljes_T_ablak[2] == T_ablak_kimenet[9]) {Teljes_SP_ablak[2] = Sp_ablak_kimenet[9]; }
266     else if (Teljes_T_ablak[2] == T_ablak_kimenet[3]) {Teljes_SP_ablak[2] = Sp_ablak_kimenet[3]; }
267     else if (Teljes_T_ablak[2] == T_ablak_kimenet[2]) {Teljes_SP_ablak[2] = Sp_ablak_kimenet[2]; }
268     else {Serial.println ("Hiba ablak_2!");}
269

```

### 19. ábra A fuzzy eredmény számítására példa

A számbeli végeredményhez már csak az előbb kiszámolt fuzzy végeredmény súlypontját kell meghatározni. Összeszorozzuk az azonos tömb számú trapézok súlypontját és területét, majd miután ezeket összeadjuk a végen elosztjuk az eredményt a területek összegével. Ezt elvégezve mindkét kimeneten meg is lesz a végeredmény.

#### 3.2.8. Használt funkciók

```

385 // 10. rész ablak kimenet reszeinek sp kiszamitas fugveny
386 float Sp_ablak(float y, int v) {
387     float P1[3]={0.0,0.0,0.0};
388     float P2[3]={0.0,0.0,0.0};
389     float T_1 = 0.0;
390     float T_2 = 0.0;
391     float L_1 = 0.0;
392     float L_2 = 0.0;
393     float A[3][3]={0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0}; // msf adott háromszögének adott trapézának adott területé
394     float S[3][3]={0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0}; // msf adott háromszögének adott trapézának adott sp
395     float Sp_a= 0.0;
396
397     if (v == 0){
398         int j = 0;
399         P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
400         P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
401         T_1 = (P1[j]-ki1_msf[j][0]);
402         T_2 = (T_1/3.0);
403         L_1 = (ki1_msf[j][3]-P2[j]);
404         L_2 = (L_1/3.0);
405         A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
406         A[j][1] = (P2[j]-P1[j])*y;
407         A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
408         S[j][0] = (P1[j]-T_2);
409         S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
410         S[j][2] = (P2[j]+L_2);
411         Sp_a = ((A[j][0] * S[j][0] + A[j][1] * S[j][1] + S[j][2] * A[j][2]) / (A[j][0] + A[j][1] + A[j][2])); ; }

```

### 20. ábra Terület és súlypont számító funkcióra példa

Ebben a részben a már korábban említett funkciók működéséről lesz szó (20. ábra). Először a „mozgó pontok” számítása történik meg, ezeket a kódban *P1* és *P2*-vel jelöltem, ez a két pont a trapéz D és C pontjai, *y* értékük a trapéz magasságával megegyező, amely a funkció egyik bemeneti értéke. Ennek és az alap háromszög

ismeretében arányossággal meg lehet határozni az  $x$  értéküket is, így megismerjük a trapéz négy pontját. A terület és súlypont számításához a trapézt három részre kell osztani, egy középső téglalapra és az ezt körbe fogó két derékszögű háromszögre. Ezen egyszerűbb testek területét és súlypontját matematikai szabályokkal könnyen ki tudjuk számítani. A teljes trapéz területét a rész területek összeadással, súlypontját a rész súlypontok területtel való súlyozott összegével kapjuk meg. A másik bemeneti értéke a függvénynek a  $v$  érték, amiről már korábban írtam. Mivel ebben a környezetben egy funkció csak egy eredményt adhat, ezért hiába számoltuk ki itt a területet is azt még egyszer ki kell egy másik, ehhez hasonló funkcióban számolni, ugyanis ez csak a súlypont értékét adja vissza.

### **3.3. LabVIEW szimuláció**

#### **3.3.1. Hardware in the loop**

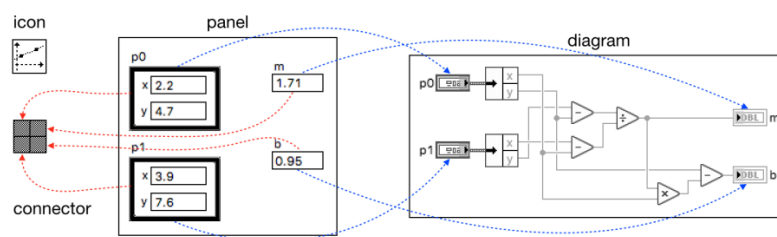
A hardware in the loop röviden HIL, egy olyan szimulációs technika, mely beágyazott rendszerek rendszerszintű tesztelését teszi lehetővé, olcsón és könnyen megismételhetően. Egy valós idejű szimulációt kell létrehozni, mely modellezni képes a rendszer bizonyos részeit, valamint a rendszer és környezete közötti kapcsolatot. Ilyen módon meg lehet vizsgálni, hogy a rendszer pontosan hogyan működne bizonyos szituációkban anélkül, hogy azokat valós környezetben állítanánk elő. (Ledin, 2001)

A technológiát még az 1950-es években fejlesztették ki Amerikában, legfőképpen az űrkutatás és hadiipar számára. A számítógépek fejlődése és elterjedése lehetővé tette, hogy több iparágban elterjedjen a módszer. Az űrkutatási sikere után az 1990-es évekre elterjedt az autóiparban is. (Nabi és mtsai., 2004) Napjainkban számos előnye miatt széles körben használják ezt a tesztelési módot. Előnyei között szerepel az olcsósága és megismételhetőségén kívül, a szimulációs sebessége, könnyű állíthatósága, biztonsága és automatizálhatósága. (Mihalič és mtsai., 2022)

A dolgozat során ezen szimulációs technológiát használom a mikrokontrolleren futó logika ellenőrzésére. A használt Arduino Nano úgy fog számolni minden esetben, mintha a bemeneti eredményeket szenzoroktól kapná és a kimeneti végeredményeket aktuátoroknak továbbítaná. Ugyanakkor nem kell megvalósítani a teljes rendszert szenzorokkal és aktuátorokkal, ugyanis azokat szimulálni is tudjuk LabVIEW környezetben, olyan módon, hogy a környezet lehetőséget ad a mikrokontroller soros kommunikációs portjának olvasására és írására.

### 3.3.2. LabVIEW

A LabVIEW program kezdeti célja nem egy új nyelv megteremtése volt, hanem hogy egy eszköz legyen, azan tudósok, mérnökök és technikusok számára, akik nem tudnak programozni, mérések és számítások automatizálására. A fejlesztő cég a National Instruments 1976-ban alapította James Truchard, William Nowlin és Jeffrey Kodosky. A szoftver két részre osztható az úgynevezett Front Panel-ra és a Block Diagram-ra. Az utóbbi felületen a grafikus programozás végezhető el, míg az előbbi az eredmények megjelenítésére szolgál. Ha a Front Panel-en elhelyezünk egy kijelzőt, vagy egy adat bevitelre szolgáló panelt, akkor a Block Diagram-on megjelenik ezen panelek megfelelője, amit így képesek vagyunk bekötni a grafikus programba (21. ábra). (Kodosky, 2020)



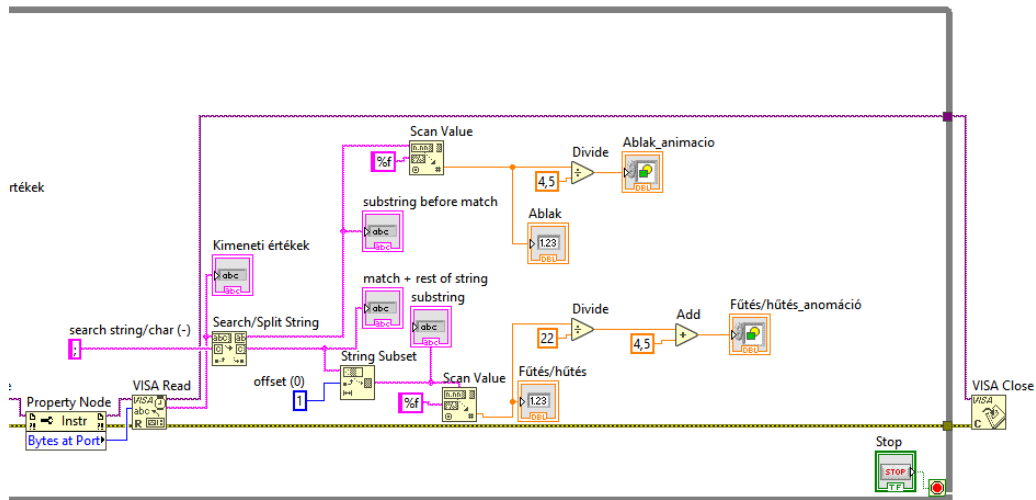
21. ábra LabVIEW működése

Forrás: (Kodosky, 2020)

### 3.3.3. Block Diagram

A program a négy bemeneti értékből pontosvesszővel elválasztva egy szöveget alkot, amit a soros porton keresztül kommunikál a mikrokontrollerrel. A folyamat végén az Arduino is egy pontosvesszővel elválasztott szöveget küld a porton keresztül, ezt választja szét a LabVIEW két kimeneti értékre.





23. ábra A Block Diagram második fele

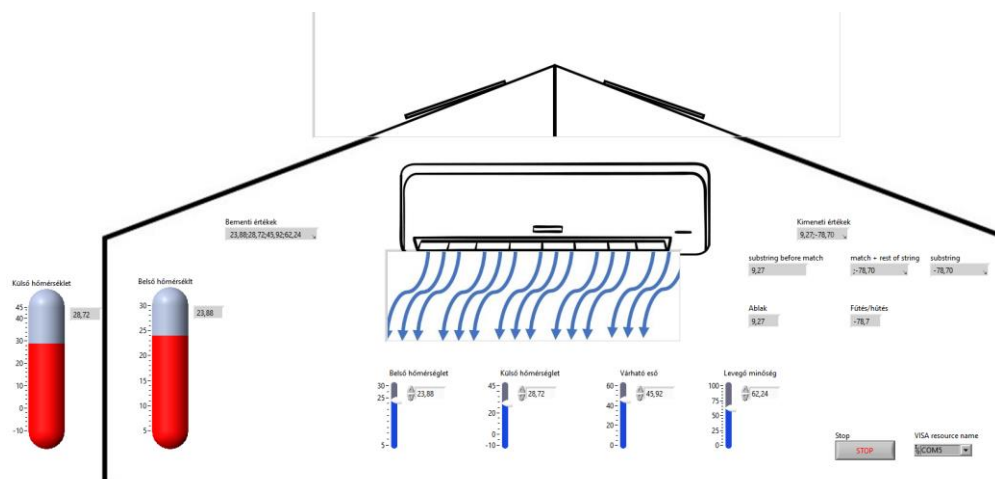
Miután megvan a két érték, azokat, hogy az animációk megfelelően működjenek, tovább kell alakítani (23. ábra). Az ablak animáció összesen 22 darab képből áll és minden egyes képen egy fokkal változik az ablakok nyitási szöge. A fűtés-hűtés animáció összesen 9 képből áll, ebből négy a fűtés és négy a hűtés erősségét jelzi. Az előbbieket a piros nyilak számát változtatják, míg az utóbbiak a kék nyilakét. A két állapot között van egy képkocka, amin nem szerepelnek nyilak, ilyenkor nincs bekapcsolva fűtő-hűtő rendszer.

A program alapesetben egyfolytában küldené és fogadná az adatokat az Arduinótól, ennek elkerülése érdekében egy időzítőt is elhelyeztem a while loop-ban, melynek köszönhetően 100 ms-ot vár a program két adat küldés és olvasás között. Továbbá még a grafikus környezetre, egy stop gombot is elhelyeztem, ami a program futásának vet véget megnyomás esetén, olyan módon, hogy kilép a loopból. A teljes Block Diagram-ot a 2. melléklet tartalmazza.

### 3.3.4. Front Panel

Háttérnek egy ház mintáját állítottam be, egy legkondicionáló berendezéssel a közepén, az animációk dobozát igyekeztem úgy beállítani, hogy azok illeszkedjenek ehhez. Helyet kapott még a négy bemeneti értéket állító csúszka és a két hőmérsékletet jelző hőmérő. Ezekről nem távol helyezkedik el a négy értéket magába foglaló szöveg

kiírása, melyet az Arduino fog kiolvasni. A ház másik felén az Arduinótól kapott értékek szétválasztásának lépéseinek értékei láthatóak, melynek eredménye a két alsó szám (24. ábra). Ezen számok befolyásolják az animációkat is, jól látható, hogy a 9,27-es ablak nyitási eredmény következtében épphogy egy kis résnyire van nyitva a két tetőablak. Továbbá az is megfigyelhető, hogy a -78,7-es eredmény miatt nagy értékű hűtést mutat a légkondicionáló animációja.



24. ábra A Front Panel működésére példa

A helyes működés ellenőrzésére a MATLAB Fuzzy Toolbox-ban is ugyanazokat a bemeneti értékeket alítottam, mint a LabVIEW-ban (2. táblázat Számítási eredmények). Az előbbi ablaknyitási eredménye 9,36, míg az az utóbbié 9,27, a fűtés-hűtés esetén ez az érték -78,8 és -78,7. Az eltérés kerekítési különbségekből adódik. Összesen 5 különböző bemeneti értékkel teszteltem a rendszert, melyeknek eredményét a 3. melléklet mutatja.

2. táblázat Számítási eredmények

| Számítást végző eszköz  |                        | MATLAB       | Arduino Nano |
|-------------------------|------------------------|--------------|--------------|
| <b>Bemeneti értékek</b> | Belső hőmérséklet [°C] | 23,88        | 23,88        |
|                         | Külső hőmérséklet [°C] | 28,72        | 28,72        |
|                         | Várható eső [min]      | 45,92        | 45,92        |
|                         | Levegő minőség [%]     | 62,24        | 62,24        |
| <b>Végeredmény</b>      | Ablak nyitás [%]       | <b>9,36</b>  | <b>9,27</b>  |
|                         | Fűtés / hűtés [+/- %]  | <b>-78,8</b> | <b>-78,7</b> |

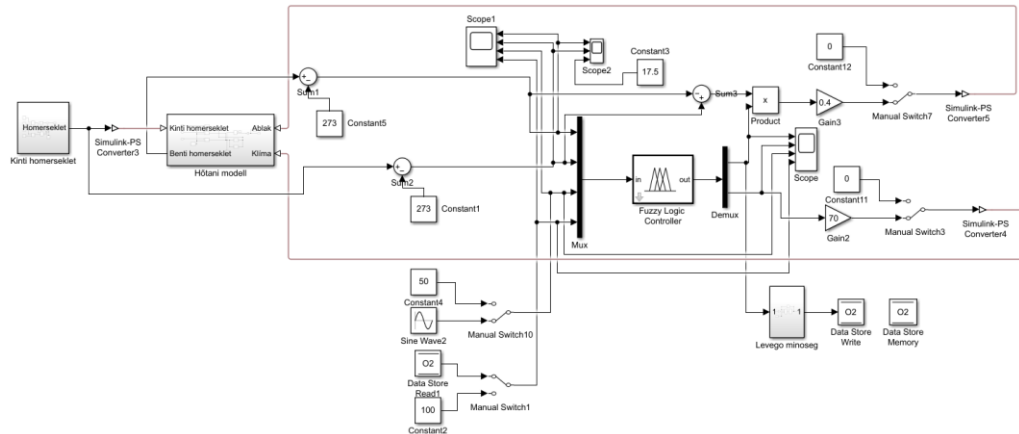
Összeségében megállapítható, hogy a mikrokontroller képes volt az általam tesztelt esetekben, közel megegyező eredményt adni, mint MATLAB Fuzzy Toolbox. A pontos hiba mértékének meghatározására vettem a két eredmény különbségét, majd ezt vetítettem az adott kimenet teljes skálájára, (3. táblázat). A legpontosabban számítások során is a hiba érték 1% alatt volt, de az átlagos hiba az ablak nyitására 0,29%, míg a fűtés/hűtésre 0,18% lett ezen öt eset során.

3. táblázat A mikrokontroller számítási hibája

| Számítás sorszáma | Ablak nyitás hiba | Fűtés / hűtés hiba |
|-------------------|-------------------|--------------------|
| 1. Számítás       | 0,09 %            | 0,05 %             |
| 2. Számítás       | 0,57 %            | 0,75 %             |
| 3. Számítás       | 0,66 %            | 0,00 %             |
| 4. Számítás       | 0,11 %            | 0,08 %             |
| 5. Számítás       | 0,03 %            | 0,00 %             |
| Átlag             | 0,29 %            | 0,18 %             |

### 3.4. MATLAB szimuláció

A szimulációnak a lényege, hogy a logika működését valós körülmények között lehessen vizsgálni, valamint, hogy a kapott eredmények alapján finom hangolni lehessen az eredeti fuzzy logikát. Az alap MATLAB Simulink könyvtárakon kívül még a Fuzzy és a Simcase bővítményt kellett használni a szimulációs rendszer összerakásához. Az előbbi a már korábban elkészített fuzzy logika blokk formában való beillesztését, az utóbbi az adott szoba hőtani viselkedésének megvalósítását tette lehetővé (25. ábra).

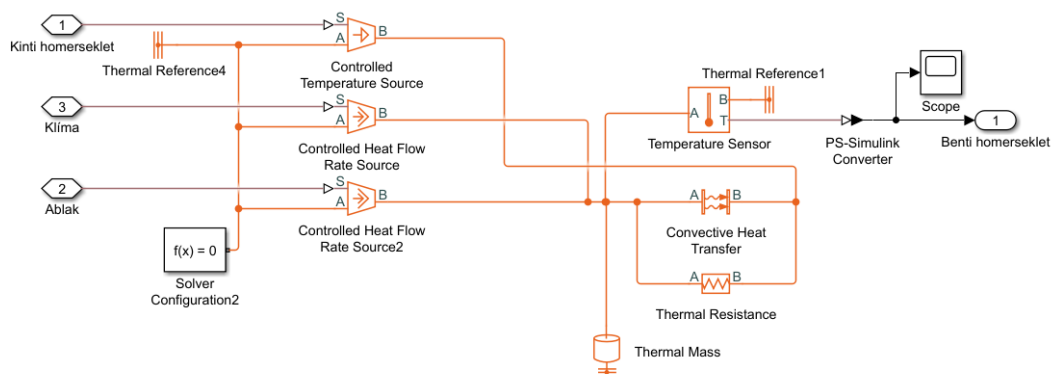


25. ábra Teljes szimulációs rendszer

### 3.4.1. A hőtani modell

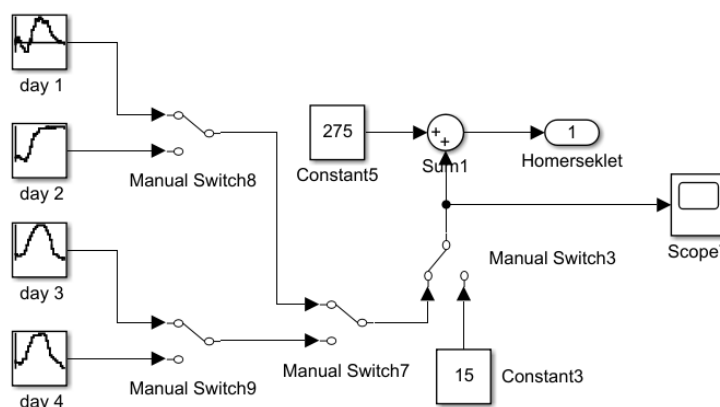
A hőtani modell elkészítése volt az egyik legnehezebb feladat a szimuláció során, mivel olyan modellt kellett alkotni, amely paramétereit képesek változni az ablak nyitásának mértéke alapján. Végül erre megoldást úgy találtam, hogy készítettem egy alapszintű hőtani modellt a szobáról, majd ehhez kapcsoltam egy hőforrást és két darab hőáramlatot (26. ábra). Az előbbi a külső hőmérsékletet szimulálta és a szoba falain történő hőveszteséget eredményezte, az utóbbi kettő volt a légkondicionáló berendezés, valamint az ablak modellje. Ezen blokkok hőtani beállításához becslési számításokat és a mérés során megismert karakterisztikákat használtam. A cél az volt, hogy ez a szimulált szoba belső hőmérséklete hasonlóan változzon, mint a mérés során mért értékek, azonos körülmények között. A rendszer eredménye, hogy a szoba belső hőmérsékletét szimulálni tudom a külső hőmérséklet, fűtés-hűtés és ablak nyitásának mértéke alapján. Ezen számolt érték lesz a fuzzy logika egyik bemenete.

### 3.4.2. Külső hőmérséklete



26. ábra Hőtani modell

A külső hőmérsékletet, egy napi hőingás segítségével állítom elő. Úgy készítettem el a szimulációt, hogy összesen négy különböző hőingású napot lehessen kiválasztani. A hőingás értékhez, alaptól hozzá kell adni 273-at, hogy az eredmény Kelvinben legyen és ne Celsiusba, ugyanakkor e konstans érték változtatásával lehet a napi hőingás értékeit feljebb vagy lejjebb mozgatni a hőmérséklet skálán (27. ábra).

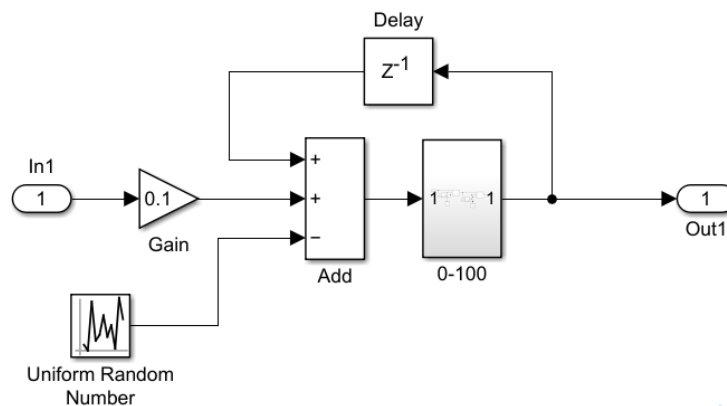


27. ábra Külső hőingás beállítása

### 3.4.3. Levegő minőség

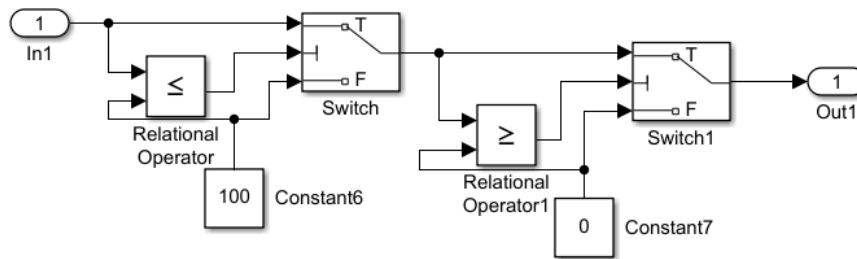
A szimuláció során nem csak a hőtani változásokat szerettem volna szimulálni, hanem a levegő minőségének változását is. Ehhez egy nagyon alap szintű modellt alkottam, amely az ablak aktuális nyitását veszi alapul, valamint egy véletlenszerűen generált számot, melynek értéke 3 és 6 között lehet. A folyamat során az ablak nyitási értékét

elosztom tízzel, majd ebből az értékből vonom ki a véletlen számot. Az eredményt késleltetés után visszakapcsolom a kivonást végző blokkba. Az így kapott jel nulláról indulva növekszik, ha az ablak 30%-nál jobban nyitva van és csökken, ha nincs. A véletlen számon múlik, hogy ez a csökkenés milyen mértékű, ez a szobában lévő emberek számának változását hivatott szimulálni. Ez a módszer a levegő minőségének szimulálására nagy mértékben leegyszerűsített, például egyenes arányosságot feltételez az ablak nyitása és a benti levegő minőségének javulása között, továbbá a nap 24 órája során egyfolytában azonos időközönként változik a szobában tartózkodó emberek száma. Ugyanakkor arra a célra megfelel, hogy lássuk, hogy hogyan oldja meg a döntéshozó rendszer ezen változó érték bizonyos szint fölött tartását (28. ábra).



28. ábra Levegő minőség változásának szimulálása

Elméleti síkon ezen jel akár nullánál kisebb vagy száznál nagyobb értéket is felvehet, mely értékeket a fuzzy logika nem tudja kezelni. Ennek kiküszöbölésére egy újabb alsóbb szinten elhelyezkedő rendszert hoztam létre, melynek feladata, hogy ha a bemeneti eredmény a skálán kívüli értéket akarna felvenni akkor a skála szélső értékét adja vissza a kimeneten, egyéb esetben nem változtat azon (29. ábra).



29. ábra Az eredmény 0 és 100 között tartása

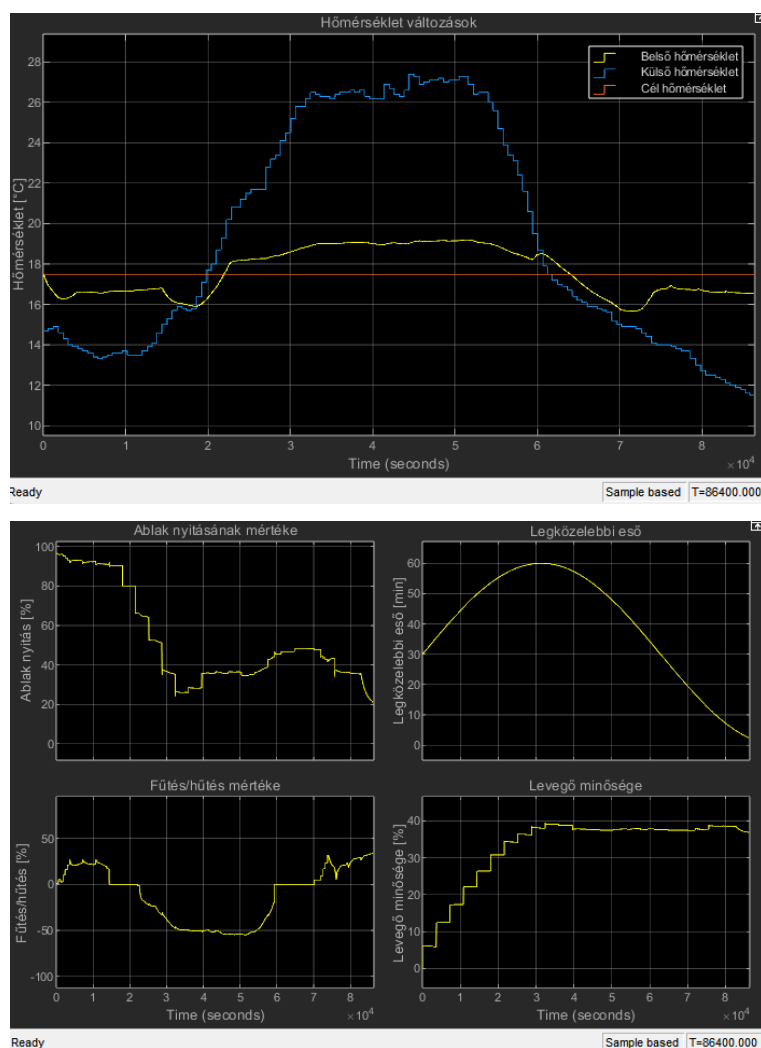
#### 3.4.4. Várható eső

Az utolsó fuzzy logika bemenetet, mely a következő várható esőt jelzi előre, egy szinuszhullámmal helyettesítettem, melyet úgy állítottam be, hogy 0 és 60 közötti érték között változzon.

Az egész szimulációs rendszert felszereltem kézi kapcsolókkal, melyek egy szabadon választható konstans érték és a szimulált érték közötti váltást teszik lehetővé. Ilyen módon le tudjuk futtatni a szimulációt bármilyen általunk választott értékekkel, vagy akár ki tudjuk kapcsolni a rendszer bizonyos részeit, egy-egy szimuláció erejéig. A jobb áltáthatóság érdekében az egy funkciót betöltő kapcsolási részeket egy alsóbb szinten elhelyezkedő rendszerbe vettem, így lett egy a kinti hőmérsékletingás beállítására, egy, amely a hőtani modellt tartalmazza és egy, amely a levegő minőséget számolja.

#### 3.4.5. Eredmények

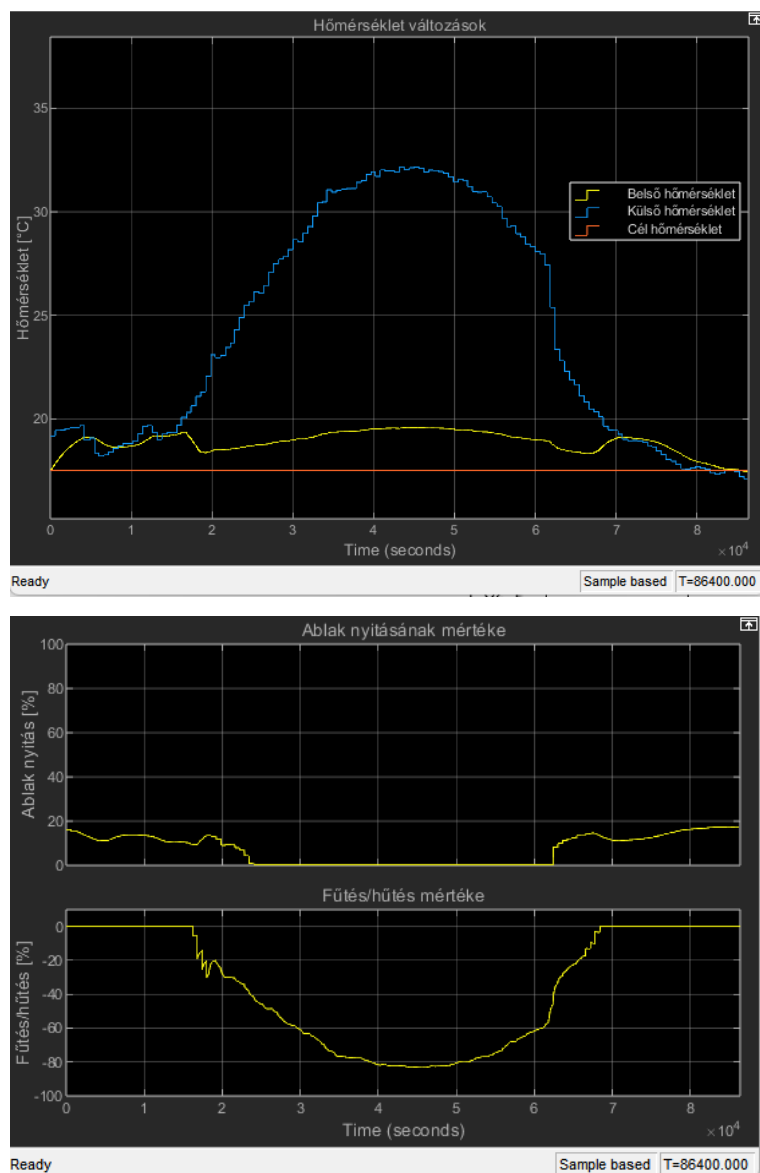
A szimulációt több külső hőmérsékleti beállítással is lefuttattam. Először egy közepes napi hőingású napot választottam, mikor a hőmérséklet 11 °C és 27 °C között alakult. Ezen szimuláció során mind a négy bementi értéket szimuláltam (30. ábra). A rendszer képes a 20 °C és 15 °C között tartani a belső hőmérsékletet, valamint 35% felett a levegő minőségét. Kezdetben az utóbbi érték 0-ról indul, ezért a nap elején nagy mértékű az ablak nyitása. A nap végére a megérkező eső miatt, az ablak csukódik. Ami a fűtés-hűtést illeti az a nap elején és végén kis mértékben fűt, a nap közepén nagyobb mértékben hűt, viszont a nap során van két több órás szakasz amikor nem kapcsolódik be. Ezen szakaszok során az ablak nagyobb mértékű nyitása miatt a belső hőmérséklet követi a külsőt. A levegő minőség kezdetben növekszik aztán 35% és 40% mozog a nap végéig.



30. ábra Eredmények: átlagos nap

### 3.4.6. Nyári nap

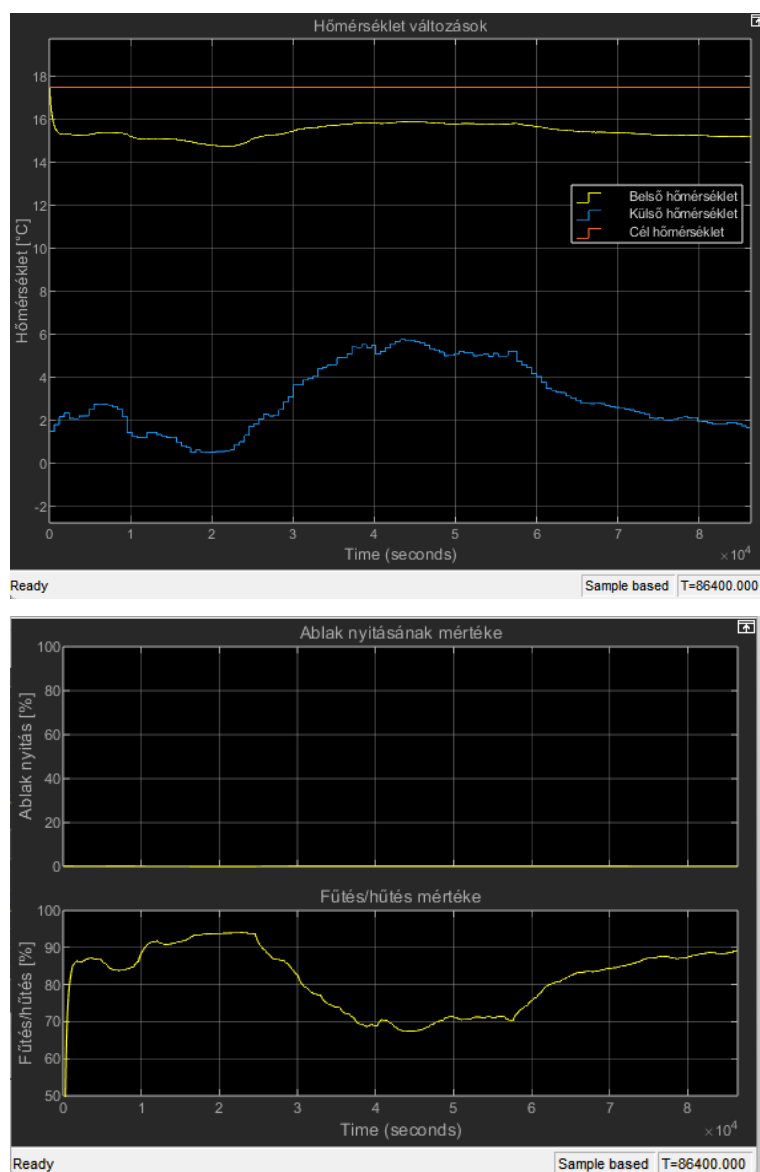
A nyári nap szimulálása során a két másodlagos fuzzy bemenetet fix értékre állítottam, olyan módon, hogy ne befolyásolják a logika működését. A rendszer képes tartani a célhőmérséklet közvetlen környezetét és nem engedi, hogy a belső hőmérséklet  $20^{\circ}\text{C}$ -ot elérje, miközben a kinti hőmérséklet a  $32^{\circ}\text{C}$ -ot is átlépi. A nap elején és végén a rendszer az ablak nyitásával igyekszik hűteni a szoba belső hőmérsékletét, ugyanis ilyenkor a kinti hőmérséklet alacsonyabb, mint a benti, ilyenkor a hűtő rendszer ki van kapcsolva. A nap többi részében az ablak van teljesen csukott állapotban, míg a hűtés van, hogy a 80%-ot is átlépi (31. ábra)



31. ábra Eredmények: nyári nap

#### 3.4.7. Téli nap

A téli nap során a külső hőmérséklet 0 és 6  $^{\circ}\text{C}$  között mozog végig, ugyanakkor a belső hőmérséklet nem esik 15  $^{\circ}\text{C}$  alá, ami a beállított cél hőmérséklet sáv alsó határa. A rendszer képes közel állandónak tartani a belső hőmérsékletet, a külső hőingás ellenére is. Mikor kint melegebb van a fűtés érték körülbelül 70% körül mozog, míg a nap leghidegebb pontján ez az érték a 90%-ot is átlépi (32. ábra).

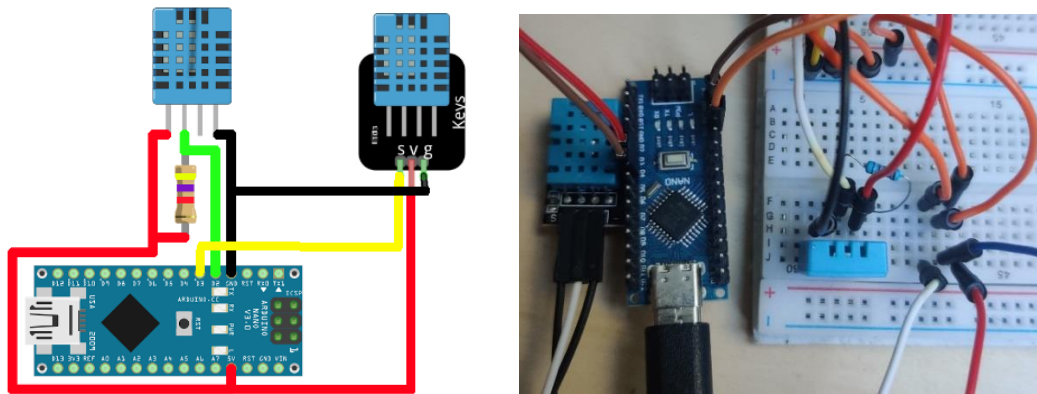


32. ábra Eredmények: téli nap

### 3.4.8. Mérés

A mérés célja egyrészt a dolgozat során használt két hőmérséklet szabályzó módszer, az ablak nyitása és a légkondicionáló rendszer működtetése, összehasonlítása, valamint a szoba termosztatikai megismerése volt. Ezek alapot nyújtottak a szimuláció paramétereinek a beállításához.

A méréshez felhasznált eszközök egy Arduino Nano mikrokontroller, két darab DHT11-es hőmérséklet szenzor és az adatok rögzítésére szolgáló személyi számítógép volt, valamint pár hosszabb vezeték. Az utóbbi segítségével képes voltam az egyik mérőszenzort az ablak külső felére rögzíteni, míg a másikat a szoba közepére elhelyezni.



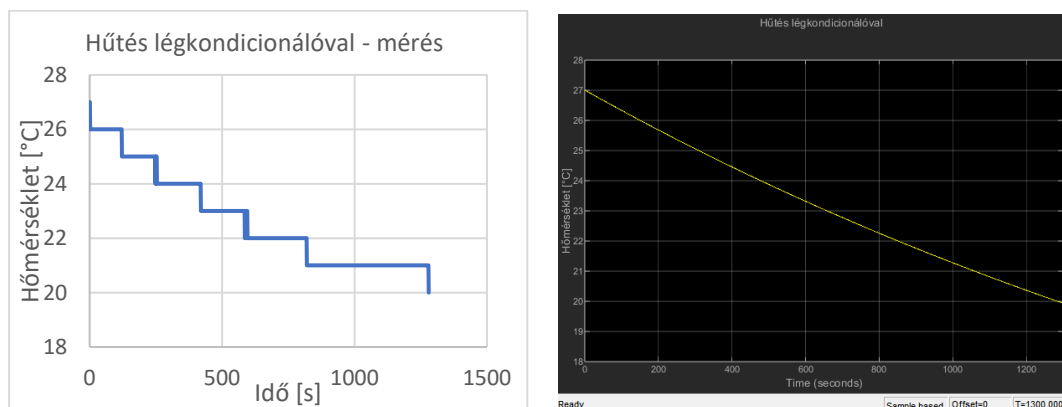
33. ábra A két darab hőmérséklet szenzor Arduinohoz kapcsolása

Két különböző típusú DHT11-es hőmérő állt rendelkezésemre a mérés során, az egyik már rendelkezett a szenzor elé szerelt felhúzó vagy másnéven pullup ellenállással, míg a másik nem. Az utóbbi esetében nekem kellett ezt a 10 k $\Omega$ -os ellenállást elhelyezni a kapcsolás során (33. ábra). Az előre szerelt szenzor, mivel annál csak a vezetékeket kellett meghosszabbítanom, került az ablak külső részére, míg a másik a belső hőmérsékletet mérte. A DHT11 az egyik legelterjedtebb hőmérséklet szenzor mikrokontrollerekhez, a hőmérséklet mérésen kívül, képes páratartalom mérésére is, ám ezt a tulajdonságát most nem használtam ezen mérés során. Nagy népszerűséget az árán kívül a könnyű használhatóságának köszönheti, ugyanis számtalan előre megírt könyvtár található az interneten, melyek a program megírását teszi egyszerűvé. Én is egy ilyen könyvtárat használva készítettem el az Arduinon futó programot, mely másodpercenként olvassa ki a két hőmérséklet értéket és írja ki azokat a soros monitorra, ezt a kódot a 4. melléklet tartalmazza. Az ide kiírt értékeket menti le egy a számítógépen futó program egy Excel fájlba. A használt hőmérséklet szenzor hátránya, hogy nem túl jó a pontossága és a felbontása sem, (4. táblázat). Ugyanakkor számomra csak viszonyítási értéknek volt fontos a mérés eredménye és erre a célra megfelel az eszköz.

4. táblázat A DHT11 szenzor adatai

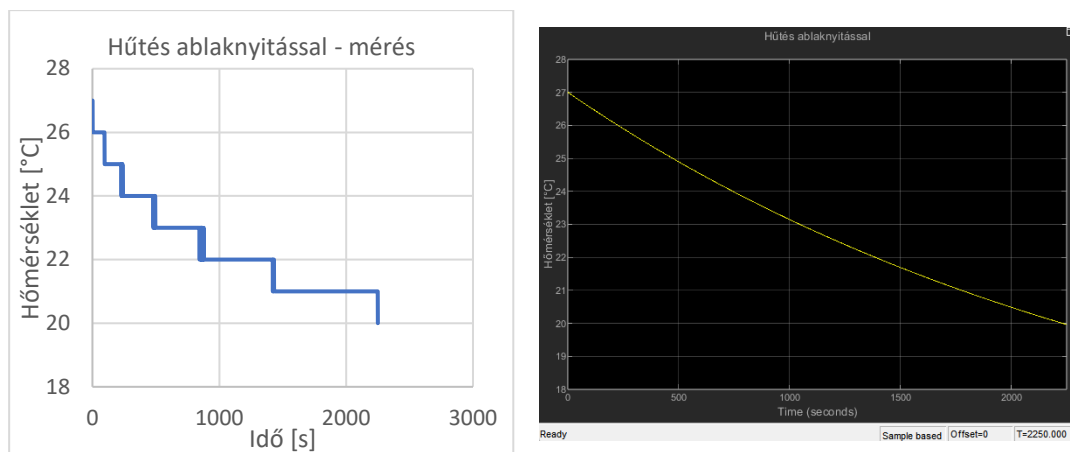
| Megnevezés                 | Érték    |
|----------------------------|----------|
| Mérés tartomány            | 0-50 °C  |
| Pontosság                  | +/- 2 °C |
| Mintavételezési frekvencia | 1 Hz     |
| Mérési felbontás           | 1 °C     |
| Működési feszültség        | 3-5 V    |

A mérés menete során kétféle módon hűtöttem le a szobát azonos hőmérsékletről, miközben mértem a belső és a külső hőmérsékletet. Kezdetben a szobában található légkondicionáló berendezés segítségével felfűtöttem a belső hőmérsékletet 27 °C-ra, majd először a korábban fűtésre szolgáló eszköz átállításával hűtöttem le a szobát 20 °C-ra. Ezt követően újra felmelegítés következett és másodjára az ablak kinyitásával értem el a lehűlést. Olyan időpontot választottam a mérés elvégzéséhez, amikor a kinti hőmérséklet közel állandóan 15 °C volt. Az első hűtés során a lehető leghidegebbre állítottam az eszközt, 17 °C-ra. A második esetben a két tetőablak teljes mértékben nyitva volt, vízszintes állásban.



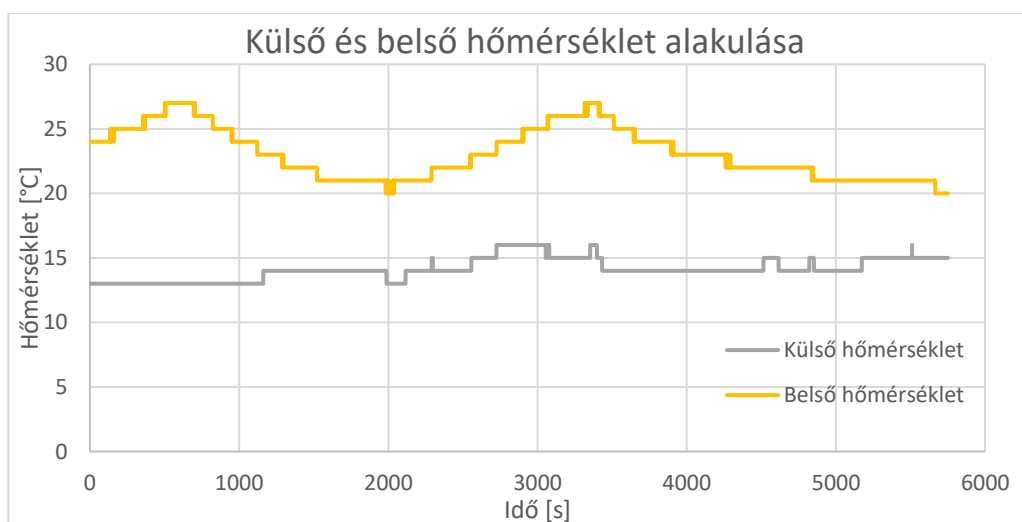
34. ábra Hűtés légkondicionálóval, mérés és szimuláció

A légkondicionálóval való lehűtés kicsit több mint 21,5 percet vett igénybe, míg a kezdeti 27°C-ról 20°C-ra hűlt a szoba belső hőmérséklete (34. ábra).



35. ábra Hűtés ablaknyitással, mérés és szimuláció

Az ablakok teljes nyitásával összesen 37,5 percnak kellett, hogy a szoba elérje a kívánt hőfokot (34. ábra). Hasonlóképpen a korábbihoz az első grafikon a mérést, míg a második a szimulációnak az eredményét mutatja azonos körülmények között. A két ábrán jól megfigyelhető, hogy hasonló a két hűtési technika karakterisztikája, ugyanakkor a légkondicionáló jelentősen, körülbelül 16 perccel, gyorsabban éri el a 20 °C-ot. A vártak megfelelően egyfajta hiperbolát ír le a két hűtési módszer karakterisztikája, mely során kezdetben gyorsabban hűlt egy fokot a szoba, aztán egyre több idő kellett ennek bekövetkeztéhez. A mérés mégis alátámasztja, hogy bizonyos körülmények között, ablak nyitással is lehet hőmérsékletet szabályozni még, ha nem is olyan gyorsan következik be a változás, mint egy légkondicionáló használata során.



36. ábra Külső és belső hőmérséklet alakulása a teljes mérés alatt

A teljes mérés összesen közel 96 percig tartott, melyet kettő darab hűtési és fűtési ciklusra lehet osztható. A folyamat során a két hőmérsékletet mérő szenzorból a program minden másodpercben kiolvasta és feljegyezte az éppen aktuális hőmérséklet értéket (36. ábra).

#### 4. ÖSSZEFOGLALÁS

A dolgozat célja egy olyan, fuzzy logikán alapuló hőmérséklet szabályzó, döntéshozó rendszer létrehozása volt, amely képes egy mikrokontrolleren futni. A rendszer figyeli a külső és belső környezetet is és ezek alapján változtatja az ablak és a fűtő-hűtő rendszer állapotát egy szobában. A végeredmény azt mutatja, hogy a rendszer jól működik és képes tartani a kijelölt hőmérsékletet, amellet, hogy a levegő minősége is megfelelő a szobában, a legkülönbözőbb külső hőingások ellenére is. Egy olyan irányítási rendszer jött létre, ahol a kimeneti értékek arányosan, fokozatosan változnak a bemeneti értékek változásának hatására.

Magát a logikát és külön a megírt kódot is szimulált környezetben teszteltem, utóbbit egy LabVIEW szoftverrel létrehozott grafikus felület segítségével. Ennek eredménye alapján az általam megírt program, a kerekítésből adódó hibát leszámítva, tökéletesebb azonos eredményt ad, mint a MATLAB Fuzzy Toolbox-ban összerakott logika. A fuzzy döntéshozórendszer valójában terület, és esetünkben súlypont számítások sokasága, ezt kihasználva le tudtam programozni a logikát Arduino IDE környezetet használva. A futtatható program összesen 600 sorát 13 részre tagoltam, minden rész egy lépést jelent a számítás menetében. Tehát a dolgozat fontos eredménye, hogy megfelelően megtervezett fuzzy döntéshozó rendszert át lehet ültetni egy mikrokontrollerre, így azt akár összetett irányítástechnikai rendszer szabályzására is lehet használni költségtakarékos és megbízható módon.

Ami a logika helyes működésének szimulálását illeti, azt MATLAB Simulink környezetben hoztam létre, több bővítményt, köztük a Simcape hőtani könyvtár használatával. Az ott lefutott szimulációk eredménye azt mutatja, hogy a rendszer képes tartani a megfelelő belső környezeti körülményeket, különböző külső napi hőingások mellett. Több különböző beállítással futtattam le a szimulációt, melyek alapján finomhangoltam a fuzzy logikát, amíg a végső beállítások ki nem alakultak. A dolgozatban három különböző karakterisztikájú nap szimulációs eredményét dokumentáltam. A logika tervezése során fontos volt, hogy úgy működjön a rendszer, hogy sose legyen egyszerre nagy mértékű fűtés vagy hűtés miközben nyitva van az ablak. A végeredményt mutató grafikonokat megfigyelve ez is teljesül a hőmérséklet tartás közben.

A kutatásom során a hőtani modell elkészítéséhez viszonyítási méréseket is végeztem. A mérés célja egyrészt a dolgozat során használt két hőmérséklet szabályzó módszer, az ablak nyitása és a légkondicionáló rendszer működtetése,

összehasonlítása, másrészt a szoba termosztatikai megismerése volt. Ezek alapot nyújtottak a szimuláció paramétereinek a beállításához. A méréshez felhasznált eszközök egy Arduino Nano mikrokontroller, két darab DHT11-es hőmérséklet szenzor és az adatok rögzítésére szolgáló személyi számítógép voltak.

A logika továbbfejlesztését én a szakirodalom áttekintése során bemutatott további lágy számítási módszerek felhasználásában látom, mint például a fuzzy logika pontosítását neurális hálóval vagy genetikus algoritmussal. A rendszert alkalmazni olyan területeken lehetne, ahol elengedhetetlen a megfelelő belső körülmények megteremtése és kívánt szinten tartása. Ilyen igény felmerülhet mezőgazdaságban például üvegházakban vagy zárt térben történő állattartás során, illetve akár az egészségügyi intézményekben is. A szakirodalom áttekintése során ehhez hasonló mikrokontrolleren is futó, több be- és kimeneteket használó (MIMO) döntéshozó logikát nem találtam, ezért újszerűnek mondható az általam megalkotott rendszer.

## 5. FELHASZNÁLT IRODALOM

- Adrienn, K. (2019. November 05). Fuzzy szakértői rendszerek lehetőségei megváltozott munkaképességű emberek foglalkozási rehabilitációjának kiválasztási folyamatában. *Alkalmazott Pszichológia*, old.: 103–127.
- Axelsson, J. (1994). *The Microcontroller Idea Book: Circuits, Programs & Applications Featuring the 8052-BASIC Single-Chip Computer*. Madison, WI: Lakeview Research.
- Bai, Y., & Wang, D. (2006). Fundamentals of Fuzzy Logic Control — Fuzzy Sets, Fuzzy Rules and Defuzzifications. In H. Zhuang, D. Wang, & Y. Bai, *Advanced Fuzzy Logic Technologies in Industrial Applications* (old.: 17-36). Springer.
- Banzi, M. (2009). *Getting Started with Arduino*. Maker Media.
- Benyezza, H., Bouhedda, M., Zerhouni, M., Boudjemaa, M., & Abu Dura, S. (2018). Fuzzy Greenhouse Temperature and Humidity Control based on Arduino. *International Conference on Applied Smart Systems (ICASS)* (old.: 1-6). Medea, Algeria: IEEE. Forrás: <https://ieeexplore.ieee.org/abstract/document/8652017/metrics#metrics>
- Gupta, M., & Theory, J. (1989). Theory of T-norms and fuzzy inference methods. *Fuzzy Sets and Systems*, 431-450. Forrás: <https://madangupta.com/pages/info/mmg/paper/RJ/RJ-051.pdf>
- Hailperin, T. (1976). *Boole's Logic and Probability: A Critical Exposition from the Standpoint of Contemporary Algebra, Logic, and Probability Theory*. North-Holland Publishing Company.
- Homod, R., Mahlia, T., & Mohamed, H. (2009). PID-Cascade for HVAC System Control. *The Second International Conference on Control, Instrumentation and Mechatronic Engineering (CIM09)*. Malacca, Malaysia,: Center for research in Applied Electronics (CRAE).
- Hu, Q., So, A., Tse, W., & Ren, Q. (2005). Development of ANN-based models to predict the static response and dynamic response of a heat exchanger in a real MVAC system. *Journal of Physics: Conference Series*, old.: 110–121.
- Kalechman, M. (2009). *Practical MATLAB applications for engineers*. CRC Press.
- Kardos, T., & Kutasi, D. (2019). HVAC rendszer modell alapú prediktív szabályozása. *Műszaki Tudományos Közlemények vol. 11* , 101–104.

- Kóczy, L. T., & Tikk, D. (2001). *Fuzzy rendszerek*. Typotex Elektronikus Kiadó Kft.
- Kodosky, J. (2020. June 12). LabVIEW. *Proceedings of the ACM on Programming Languages*, old.: 1–54.
- Lafon, F., & Balmat, J.-F. (2002). Optimized fuzzy control of a greenhouse. *Fuzzy Sets and Systems* 128, 47-59. Forrás: <https://www.sciencedirect.com/journal/fuzzy-sets-and-systems>
- Ledin, J. (2001). *Simulation Engineering: Build Better Embedded Systems Faster*. CRC Press.
- Liu, P.-D., & Chen, S.-M. (2016). Heronian aggregation operators of intuitionistic fuzzy numbers based on the Archimedean t-norm and t-conorm. *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*. Jeju, South Korea: IEEE. Forrás: <https://ieeexplore.ieee.org/document/7872971>
- Lukács, J. (2019). Defuzzifikációs módszerek megfelelőségének vizsgálata utastér zajszintjének meghatározására. *Bánki Közlemények*, old.: 11-15.
- Mamdani, H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 1-13.
- Marzuki, K., Omatu, S., & Yusof, R. (1995). Temperature Regulation with Neural Networks and Alternative Control Schemes. In *IEEE Transaction on Neural Networks*, Vol. 6 (old.: 572-582). IEEE.
- Mihalič, F, Truntič, M, & Hren A. (2022). Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges. *Electronics*, 11(15). Forrás: <https://www.mdpi.com/2079-9292/11/15/2462>
- Nabi, S., Balike, M., Allen, J., & Rzemien, K. (2004). An Overview of Hardware-In-the-Loop Testing Systems at Visteon. *SAE 2004 World Congress & Exhibition*. United States: SAE International.
- Panda, A., Wong, H., Kapila, V., & Lee, S.-H. (2006). Matlab Data Acquisition and Control Toolbox for Basic Stamp Microcontrollers. *Proceedings of the 45th IEEE Conference on Decision & Control*. San Diego, CA, USA,: IEEE.
- Pratihari, D. K. (2013). *Soft Computing Fundamentals and Applications*. Alpha Science International Ltd. : Oxford, U.K.
- Revathi, S., & Sivakumaran, N. (2016). Fuzzy Based Temperature Control of Greenhouse. *IFAC-PapersOnLine* 49, 549-554. Forrás: <https://www.sciencedirect.com/>

- Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc.
- Salamon , A. (2003). *Genetikus Algoritmusok*. ELTE. Forrás: [https://sala.melda.info/ga\\_speci\\_jegyzet.pdf](https://sala.melda.info/ga_speci_jegyzet.pdf)
- Schwartz , M., & Manickum, O. (2015). *Programming Arduino With LabVIEW*. Packt Publishing.
- Sivanandam, S., Sumathi, S., & Deepa, S. (2007). *Introduction to Fuzzy Logic using MATLAB*. Springer-Verlag Berlin Heidelberg.
- Vörös, I., & Várszegi, B. (2017). Sáv váltás prediktív modell alapú szabályozása. *25th Nemzetközi Gépészeti Konferencia*. Kolozsvár, Romania: Erdélyi Magyar Műszaki Tudományos Társaság (EMT). Forrás: [http://real.mtak.hu/70745/1/OGET\\_2017\\_voros.pdf](http://real.mtak.hu/70745/1/OGET_2017_voros.pdf)
- Yager, R., & Zadeh, L. (1992). *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Boston: Kluwer Academic Publishers.
- Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 338-353. Forrás: <https://www.sciencedirect.com/science/article/pii/S001999586590241X>
- Zadeh, L. (1996). The Evolution of Systems Analysis and Control: A Personal Perspective. *IEEE Control Systems Magazine*, 95-98.
- Zhang, F. (2014. January 1). Building Temperature Control with Intelligent Methods. *Electronic Theses and Dissertations*, old.: 733.
- Zilouchian, A., & Jamshidi, M. (2001). *Intelligent control systems using soft computing methodologies*. CRC Press.

## ***KÖSZÖNETNYILVÁNÍTÁS***

Köszönetemet fejezem ki mindazoknak, akiknek a segítségével idáig eljuthattam, és azoknak, akik mindvégig mellettem álltak. Elsősorban köszönöm, Dr. Sarcevic Péter főiskolai docens témavezetőmnek a hasznos útmutatást és építő jellegű kritikáit. Köszönettel tartozom a családomnak és közvetlen környezetemnek, akik rengeteg türelmet tanúsítottak velem szemben és tanácsaikkal segítettek dolgozatom elkészítésében.