

**ORSZÁGOS TUDOMÁNYOS  
DIÁKKÖRI KONFERENCIA  
DOLGOZAT MELLÉKLET**

**Tráser Miklós László**

**2023.**



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

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

***ORSZÁGOS TUDOMÁNYOS DIÁKKÖRI KONFERENCIA  
DOLGOZAT MELLÉKLET***

***Tráser Miklós László***

*Mechatronikai mérnök BSc*

*hallgató*

*Témavezető:*

***Dr. Sarcevic Péter***

*főiskolai docens*

***SZEGED***

***2023.***

## ***MELLÉKLETEK***

### **1. melléklet: Az Arduinon futó fuzzy logika teljes kódja**

```
// 1. rész bemenet:

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

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

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

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


// 2. rész kimenet:

float ki1[3][4]={{-10.0,0.0,0.0,10.0},{5.0,25.0,25.0,45.0},{40.0,100.0,100.0,160.0}}; //Ablak

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}}; //Futes/hutes

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}};

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}};


// 3. rész változók:

float A_1[3]={0.0,0.0,0.0}; // trapez első harmada területe
float A_2[3]={0.0,0.0,0.0}; // trapez középső harmada területe
float A_3[3]={0.0,0.0,0.0}; // trapez utolsó harmada területe
float X_1[3]={0.0,0.0,0.0}; // trapez első harmada sp
float X_2[3]={0.0,0.0,0.0}; // trapez középső harmada sp
float X_3[3]={0.0,0.0,0.0}; // trapez utolsó harmada sp


float yb[4][3]={{0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0}}; // számolt y érték
float xb[4]={0.0,0.0,0.0,0.0}; // bemeneti x érték


float ki_y[14]={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}; // kimeneti y érték
float Sp_ablak_kimenet[14]={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}; // ablak kimenet
sp

float Sp_fh_kimenet[14]={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}; // futes/hutes
kiemnet sp

float T_ablak_kimenet[14]={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}; // ablak kimenet
területe

float T_fh_kimenet[14]={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}; // futes/hutes kimenet
területe

float Teljes_SP_ablak[3]={0.0,0.0,0.0}; //ablak kimenet három részének súlypontjai
float Teljes_T_ablak[3]={0.0,0.0,0.0}; //ablak kimenet három részének területei
float Teljes_SP_FutesHutes[5]={0.0,0.0,0.0,0.0,0.0}; //futes/hutes kimenet három részének súlypontjai
float Teljes_T_FutesHutes[5]={0.0,0.0,0.0,0.0,0.0}; //futes/hutes kimenet három részének területei
float Ablak_kiemnet_eredmeny = 0.0; //ablak kimenet vegeredmenye
float FutesHutes_kiemnet_eredmeny = 0.0; //futes/hutes kimenet vegeredmenye


// 4. rész soros monitor:
```

```

void setup() {
    Serial.begin(9600);
}

void loop() {

    // 2.1 rész kimenetek eltolasa
    for (int i=0; i<3; i++){
        for (int j=0; j<4; j++){
            ki1_msfc[i][j] = ki1[i][j] + 10;}}

    for (int i=0; i<5; i++){
        for (int j=0; j<4; j++){
            ki2_msfc[i][j] = ki2[i][j] + 150;}}

    // 5. rész sting szétválasztás, bementi értékek beolvasása
    if (Serial.available() > 0) {
        String receivedString = Serial.readStringUntil('\n');
        int separatorIndex1 = receivedString.indexOf(';');
        if (separatorIndex1 != -1) {
            String strValue1 = receivedString.substring(0, separatorIndex1);
            receivedString = receivedString.substring(separatorIndex1 + 1);
            int separatorIndex2 = receivedString.indexOf(';');
            if (separatorIndex2 != -1) {
                String strValue2 = receivedString.substring(0, separatorIndex2);
                receivedString = receivedString.substring(separatorIndex2 + 1);
                int separatorIndex3 = receivedString.indexOf(';');
                if (separatorIndex3 != -1) {
                    String strValue3 = receivedString.substring(0, separatorIndex3);
                    String strValue4 = receivedString.substring(separatorIndex3 + 1);

                    // Konvertáld floatá
                    strValue1.replace(',', ' ');
                    strValue2.replace(',', ' ');
                    strValue3.replace(',', ' ');
                    strValue4.replace(',', ' ');
                    xb[0] = strValue1.toFloat();
                    xb[1] = strValue2.toFloat();
                    xb[2] = strValue3.toFloat();
                    xb[3] = strValue4.toFloat();
                }
            }
        }
    }

    // 6. rész yb kiszámítása:
    for(int i=0; i<3; i++){
        if (be1_msfc[i][0] <= xb[0] && xb[0] <= be1_msfc[i][1]) {yb[0][i] = (xb[0] -
be1_msfc[i][0])/(be1_msfc[i][1] - be1_msfc[i][0]);}

```

```

        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]);}
        else if (be1_msf[i][1] < xb[0] && xb[0] < be1_msf[i][2]){yb[0][i] = 1.0;}
        else if (be1_msf[i][0] > xb[0] && xb[0] > be1_msf[i][3]){yb[0][i] = 0.0;}
    }
    for(int i=0; i<3; i++){
        if (be2_msf[i][0] <= xb[1] && xb[1] <= be2_msf[i][1]) {yb[1][i] = (xb[1] -
be2_msf[i][0])/(be2_msf[i][1] - be2_msf[i][0]);}
        else if (be2_msf[i][2] <= xb[1] && xb[1] <= be2_msf[i][3]) {yb[1][i] = (be2_msf[i][3] -
xb[1])/(be2_msf[i][3] - be2_msf[i][2]);}
        else if (be2_msf[i][1] < xb[1] && xb[1] < be2_msf[i][2]){yb[1][i] = 1.0;}
        else if (be2_msf[i][0] > xb[1] && xb[1] > be2_msf[i][3]){yb[1][i] = 0.0;}
    }
    for(int i=0; i<3; i++){
        if (be3_msf[i][0] <= xb[2] && xb[2] <= be3_msf[i][1]) {yb[2][i] = (xb[2] -
be3_msf[i][0])/(be3_msf[i][1] - be3_msf[i][0]);}
        else if (be3_msf[i][2] <= xb[2] && xb[2] <= be3_msf[i][3]) {yb[2][i] = (be3_msf[i][3] -
xb[2])/(be3_msf[i][3] - be3_msf[i][2]);}
        else if (be3_msf[i][1] < xb[2] && xb[2] < be3_msf[i][2]){yb[2][i] = 1.0;}
        else if (be3_msf[i][0] > xb[2] && xb[2] > be3_msf[i][3]){yb[2][i] = 0.0;}
    }
    for(int i=0; i<3; i++){
        if (be4_msf[i][0] <= xb[3] && xb[3] <= be4_msf[i][1]) {yb[3][i] = (xb[3] -
be4_msf[i][0])/(be4_msf[i][1] - be4_msf[i][0]);}
        else if (be4_msf[i][2] <= xb[3] && xb[3] <= be4_msf[i][3]) {yb[3][i] = (be4_msf[i][3] -
xb[3])/(be4_msf[i][3] - be4_msf[i][2]);}
        else if (be4_msf[i][1] < xb[3] && xb[3] < be4_msf[i][2]){yb[3][i] = 1.0;}
        else if (be4_msf[i][0] > xb[3] && xb[3] > be4_msf[i][3]){yb[3][i] = 0.0;}
    }

    for(int i=0; i<4; i++){
        for(int j=0; j<3; j++){
            if (isnan(yb[i][j])) { yb[i][j] = 0.0;}}
    }

// 7. resz szabalyok:
//1. If (Belso_homerseget is hideg) and (Kulso_homerseget is hideg) then (Ablak_nyitasa is
csukva)(Futes_hutes is futes_magas) (1)
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])
{
    ki_y[0] = (min(yb[0][0],yb[1][0]))*1;
    Sp_ablak_kimenet[0] = Sp_ablak(ki_y[0], 0);
    T_ablak_kimenet[0] = T_ablak(ki_y[0], 0);
    Sp_fh_kimenet[0] = Sp_FH(ki_y[0], 4);
    T_fh_kimenet[0]= T_FH(ki_y[0], 4);
}
//2. If (Belso_homerseget is meleg) and (Kulso_homerseget is meleg) then (Ablak_nyitasa is
csukva)(Futes_hutes is hutes_magas) (1)
if (be1_msf[2][0] <= xb[0] && xb[0] <= be1_msf[2][3] && be2_msf[2][0] <= xb[1] && xb[1] <= be2_msf[2][3])
{

```

```

    ki_y[1] = (min(yb[0][2],yb[1][2]))*1;
    Sp_ablak_kimenet[1] = Sp_ablak(ki_y[1], 0);
    T_ablak_kimenet[1] = T_ablak(ki_y[1], 0);
    Sp_fh_kimenet[1] = Sp_FH(ki_y[1], 0);
    T_fh_kimenet[1]= T_FH(ki_y[1], 0);
}

//3. If (Belso_homerseglet is hideg) and (Kulso_homerseglet is meleg) then (Ablak_nyitasa is
nagyon)(Futes_hutes is nincs_futes_hutes) (1)
if (be1_msf[0][0] <= xb[0] && xb[0] <= be1_msf[0][3] && be2_msf[2][0] <= xb[1] && xb[1] <= be2_msf[2][3])
{
    ki_y[2] = (min(yb[0][0],yb[1][2]))*1;
    Sp_ablak_kimenet[2] = Sp_ablak(ki_y[2], 2);
    T_ablak_kimenet[2] = T_ablak(ki_y[2], 2);
    Sp_fh_kimenet[2] = Sp_FH(ki_y[2], 2);
    T_fh_kimenet[2]= T_FH(ki_y[2], 2);
}

//4. If (Belso_homerseglet is meleg) and (Kulso_homerseglet is hideg) then (Ablak_nyitasa is
nagyon)(Futes_hutes is nincs_futes_hutes)
(1)
if (be1_msf[2][0] <= xb[0] && xb[0] <= be1_msf[2][3] && be2_msf[0][0] <= xb[1] && xb[1] <= be2_msf[0][3])
{
    ki_y[3] = (min(yb[0][2],yb[1][0]))*1;
    Sp_ablak_kimenet[3] = Sp_ablak(ki_y[3], 2);
    T_ablak_kimenet[3] = T_ablak(ki_y[3], 2);
    Sp_fh_kimenet[3] = Sp_FH(ki_y[3], 2);
    T_fh_kimenet[3]= T_FH(ki_y[3], 2);
}

//5. If (Belso_homerseglet is kozepes) and (Kulso_homerseglet is kozepes) and (Legkozelebbi_eso is sokára)
then (Ablak_nyitasa is kicsit)(Futes_hutes is nincs_futes_hutes)
(0.05)

if (be1_msf[1][0] <= xb[0] && xb[0] <= be1_msf[1][3] && be2_msf[1][0] <= xb[1] && xb[1] <= be2_msf[1][3]
&& be3_msf[1][0] <= xb[2] && xb[2] <= be3_msf[1][3]) {
    ki_y[4] = (min(yb[2][1],min(yb[0][1],yb[1][1]))) * 0.05;
    Sp_ablak_kimenet[4] = Sp_ablak(ki_y[4], 1);
    T_ablak_kimenet[4] = T_ablak(ki_y[4], 1);
    Sp_fh_kimenet[4] = 0.0;
    T_fh_kimenet[4]= T_FH(ki_y[4], 2);
}

//6. If (Levego_minosege is megfelelo) then (Ablak_nyitasa is csukva)
(0.1)

if (be4_msf[2][0] <= xb[3] && xb[3] <= be4_msf[2][3]) {
    ki_y[5] = (yb[3][2])*0.1;
    Sp_ablak_kimenet[5] = Sp_ablak(ki_y[5], 0);
    T_ablak_kimenet[5] = T_ablak(ki_y[5], 0);
}

```

```

}

//7. If (Legkozelebbi_eso is most) then (Ablak_nyitasa is csukva)
(1)

if (be3_msf[0][0] <= xb[2] && xb[2] <= be3_msf[0][3]) {
    ki_y[6] = (yb[2][0])*1;
    Sp_ablak_kimenet[6] = Sp_ablak(ki_y[6], 0);
    T_ablak_kimenet[6] = T_ablak(ki_y[6], 0);
}

//8. If (Belso_homerseglet is kozepes) and (Kulso_homerseglet is hideg) and (Levego_minosege is alacsony)
then (Ablak_nyitasa is kicsit)(Futes_hutes is futes_magas)
(0.1)
if (be1_msf[1][0] <= xb[0] && xb[0] <= be1_msf[1][3] && be2_msf[0][0] <= xb[1] && xb[1] <= be2_msf[0][3]
&& be4_msf[0][0] <= xb[3] && xb[3] <= be4_msf[0][3]) {
    ki_y[7] = (min(yb[3][0],min(yb[0][1],yb[1][0])))*0.1;
    Sp_ablak_kimenet[7] = Sp_ablak(ki_y[7], 1);
    T_ablak_kimenet[7] = T_ablak(ki_y[7], 1);
    Sp_fh_kimenet[7] = Sp_FH(ki_y[7], 4);
    T_fh_kimenet[7]= T_FH(ki_y[7], 4);
}

//9. If (Belso_homerseglet is kozepes) and (Kulso_homerseglet is meleg) and (Levego_minosege is alacsony)
then (Ablak_nyitasa is kicsit)(Futes_hutes is hutes_magas)
(0.1)
if (be1_msf[1][0] <= xb[0] && xb[0] <= be1_msf[1][3] && be2_msf[2][0] <= xb[1] && xb[1] <= be2_msf[2][3]
&& be4_msf[0][0] <= xb[3] && xb[3] <= be4_msf[0][3]) {
    ki_y[8] = (min(yb[3][0],min(yb[0][1],yb[1][2])))*0.1;
    Sp_ablak_kimenet[8] = Sp_ablak(ki_y[8], 1);
    T_ablak_kimenet[8] = T_ablak(ki_y[8], 1);
    Sp_fh_kimenet[8] = Sp_FH(ki_y[8], 0);
    T_fh_kimenet[8]= T_FH(ki_y[8], 0);
}

//10. If (Levego_minosege is alacsony) then (Ablak_nyitasa is nagyon)(Futes_hutes is nincs_futes_hutes)
(0.2)

if (be4_msf[0][0] <= xb[3] && xb[3] <= be4_msf[0][3]) {
    ki_y[9] = (yb[3][0])*0.2;
    Sp_ablak_kimenet[9] = Sp_ablak(ki_y[9], 2);
    T_ablak_kimenet[9] = T_ablak(ki_y[9], 2);
    Sp_fh_kimenet[9] = Sp_FH(ki_y[9], 2);
    T_fh_kimenet[9]= T_FH(ki_y[9], 2);
}

```

```

//11. If (Levego_minosege is atmeneti) then (Ablak_nyitasa is kicsit)(Futes_hutes is nincs_futes_hutes)
(0.05)

if (be4_msf[1][0] <= xb[3] && xb[3] <= be4_msf[1][3]) {
    ki_y[10] = (yb[3][1])*0.1;
    Sp_ablak_kimenet[10] = Sp_ablak(ki_y[10], 1);
    T_ablak_kimenet[10] = T_ablak(ki_y[10], 1);
    Sp_fh_kimenet[10] = Sp_FH(ki_y[10], 2);
    T_fh_kimenet[10]= T_FH(ki_y[10], 2);
}

//12. If (Belso_homerseglet is kozepes) and (Kulso_homerseglet is hideg) then (Ablak_nyitasa is
csukva)(Futes_hutes is futes_alacsony)
(0.1)

if (be1_msf[1][0] <= xb[0] && xb[0] <= be1_msf[1][3] && be2_msf[0][0] <= xb[1] && xb[1] <= be2_msf[0][3])
{
    ki_y[11] = (min(yb[0][1],yb[1][0]))*0.1;
    Sp_ablak_kimenet[11] = Sp_ablak(ki_y[11], 0);
    T_ablak_kimenet[11] = T_ablak(ki_y[11], 0);
    Sp_fh_kimenet[11] = Sp_FH(ki_y[11], 3);
    T_fh_kimenet[11]= T_FH(ki_y[11], 3);
}

//13. If (Belso_homerseglet is kozepes) and (Kulso_homerseglet is meleg) then (Ablak_nyitasa is
csukva)(Futes_hutes is hutes_alacsony)
(0.1)

if (be1_msf[1][0] <= xb[0] && xb[0] <= be1_msf[1][3] && be2_msf[2][0] <= xb[1] && xb[1] <= be2_msf[2][3])
{
    ki_y[12] = (min(yb[0][1],yb[1][2]))*0.1;
    Sp_ablak_kimenet[12] = Sp_ablak(ki_y[12], 0);
    T_ablak_kimenet[12] = T_ablak(ki_y[12], 0);
    Sp_fh_kimenet[12] = Sp_FH(ki_y[12], 1);
    T_fh_kimenet[12]= T_FH(ki_y[12], 1);
}

// 8. resz vegeredeny szamitas

for(int i=0; i<13; i++){
    if (isnan( T_ablak_kimenet[i])) { T_ablak_kimenet[i] = 0.0;}
    if (isnan( Sp_ablak_kimenet[i])) { Sp_ablak_kimenet[i] = 0.0;} }

```

```

    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])))));
    if (Teljes_T_ablak[0] == 0.0) {Teljes_SP_ablak[0] = 0.0;}
    else if (Teljes_T_ablak[0] == T_ablak_kimenet[11]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[11];}
    else if (Teljes_T_ablak[0] == T_ablak_kimenet[12]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[12];}
    else if (Teljes_T_ablak[0] == T_ablak_kimenet[6]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[6];}
    else if (Teljes_T_ablak[0] == T_ablak_kimenet[5]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[5];}
    else if (Teljes_T_ablak[0] == T_ablak_kimenet[0]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[0];}
    else if (Teljes_T_ablak[0] == T_ablak_kimenet[1]) {Teljes_SP_ablak[0] = Sp_ablak_kimenet[1];}
    else {Serial.print ("Hiba ablak_0!");}

    Teljes_T_ablak[1] =
max(T_ablak_kimenet[10],max(T_ablak_kimenet[8],max(T_ablak_kimenet[7],T_ablak_kimenet[4])));
    if (Teljes_T_ablak[1] == 0.0) {Teljes_SP_ablak[1] = 0.0;}
    else if (Teljes_T_ablak[1] == T_ablak_kimenet[10]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[10];}
    else if (Teljes_T_ablak[1] == T_ablak_kimenet[8]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[8];}
    else if (Teljes_T_ablak[1] == T_ablak_kimenet[7]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[7];}
    else if (Teljes_T_ablak[1] == T_ablak_kimenet[4]) {Teljes_SP_ablak[1] = Sp_ablak_kimenet[4];}
    else {Serial.print ("Hiba ablak_1!");}

    Teljes_T_ablak[2] = max(T_ablak_kimenet[9],max(T_ablak_kimenet[3],T_ablak_kimenet[2]));
    if (Teljes_T_ablak[2] == 0.0) {Teljes_SP_ablak[2] = 0.0;}
    else if (Teljes_T_ablak[2] == T_ablak_kimenet[9]) {Teljes_SP_ablak[2] = Sp_ablak_kimenet[9];}
    else if (Teljes_T_ablak[2] == T_ablak_kimenet[3]) {Teljes_SP_ablak[2] = Sp_ablak_kimenet[3];}
    else if (Teljes_T_ablak[2] == T_ablak_kimenet[2]) {Teljes_SP_ablak[2] = Sp_ablak_kimenet[2];}
    else {Serial.println ("Hiba ablak_2!");}

for(int i=0; i<3; i++){
    if (isnan(Teljes_SP_ablak[i])) {Teljes_SP_ablak[i] = 0.0;}
    if (isnan(Teljes_T_ablak[i])) {Teljes_T_ablak[i] = 0.0;} }

for(int i=0; i<3; i++){
    if (abs(Teljes_T_ablak[i]) > 1000) {Teljes_T_ablak[i] = 0.0;}
    if (abs(Teljes_T_ablak[i]) > 1000) {Teljes_T_ablak[i] = 0.0;}}

    Ablak_kiemnet_eredmeny = - 10.0 + ((Teljes_SP_ablak[0] * Teljes_T_ablak[0] + Teljes_SP_ablak[1] *
Teljes_T_ablak[1] + Teljes_SP_ablak[2] * Teljes_T_ablak[2]) / (Teljes_T_ablak[0] + Teljes_T_ablak[1] +
Teljes_T_ablak[2]));

for(int i=0; i<13; i++){
    if (isnan( T_fh_kimenet[i])) { T_fh_kimenet[i] = 0.0;}
    if (isnan( Sp_fh_kimenet[i])) { Sp_fh_kimenet[i] = 0.0;} }

    Teljes_T_FutesHutes[0] = max(T_fh_kimenet[8],T_fh_kimenet[1]);
    if (Teljes_T_FutesHutes[0] == 0.0) {Teljes_SP_FutesHutes[0] = 0.0;}

```

```

else if (Teljes_T_FutesHutes[0] == T_fh_kimenet[8]) {Teljes_SP_FutesHutes[0] = Sp_fh_kimenet[8];}
else if (Teljes_T_FutesHutes[0] == T_fh_kimenet[1]) {Teljes_SP_FutesHutes[0] = Sp_fh_kimenet[1];}
else {Serial.print ("Hiba fh_0!");}

Teljes_T_FutesHutes[1] = T_fh_kimenet[12];
if (Teljes_T_FutesHutes[1] == 0.0) {Teljes_SP_FutesHutes[1] = 0.0;}
else if (Teljes_T_FutesHutes[1] == T_fh_kimenet[12]) {Teljes_SP_FutesHutes[1] = Sp_fh_kimenet[12];}
else {Serial.print ("Hiba fh_1!");}

Teljes_T_FutesHutes[2] =
max(T_fh_kimenet[9],max(T_fh_kimenet[10],max(T_fh_kimenet[7],max(T_fh_kimenet[4],max(T_fh_kimenet[3],T_fh_
kimenet[2])))));
if (Teljes_T_FutesHutes[2] == 0.0) {Teljes_SP_FutesHutes[2] = 0.0;}
else if (Teljes_T_FutesHutes[2] == T_fh_kimenet[9]) {Teljes_SP_FutesHutes[2] = Sp_fh_kimenet[9];}
else if (Teljes_T_FutesHutes[2] == T_fh_kimenet[10]) {Teljes_SP_FutesHutes[2] = Sp_fh_kimenet[10];}
else if (Teljes_T_FutesHutes[2] == T_fh_kimenet[7]) {Teljes_SP_FutesHutes[2] = Sp_fh_kimenet[7];}
else if (Teljes_T_FutesHutes[2] == T_fh_kimenet[4]) {Teljes_SP_FutesHutes[2] = Sp_fh_kimenet[4];}
else if (Teljes_T_FutesHutes[2] == T_fh_kimenet[3]) {Teljes_SP_FutesHutes[2] = Sp_fh_kimenet[3];}
else if (Teljes_T_FutesHutes[2] == T_fh_kimenet[2]) {Teljes_SP_FutesHutes[2] = Sp_fh_kimenet[2];}
else {Serial.print ("Hiba fh_2!");}

Teljes_T_FutesHutes[3] = T_fh_kimenet[11];
if (Teljes_T_FutesHutes[3] == 0.0) {Teljes_SP_FutesHutes[3] = 0.0;}
else if (Teljes_T_FutesHutes[3] == T_fh_kimenet[11]) {Teljes_SP_FutesHutes[3] = Sp_fh_kimenet[11];}
else {Serial.print ("Hiba fh_3!");}

Teljes_T_FutesHutes[4] = max(T_fh_kimenet[7],T_fh_kimenet[0]);
if (Teljes_T_FutesHutes[4] == 0.0) {Teljes_SP_FutesHutes[4] = 0.0;}
else if (Teljes_T_FutesHutes[4] == T_fh_kimenet[7]) {Teljes_SP_FutesHutes[4] = Sp_fh_kimenet[7];}
else if (Teljes_T_FutesHutes[4] == T_fh_kimenet[0]) {Teljes_SP_FutesHutes[4] = Sp_fh_kimenet[0];}
else {Serial.print ("Hiba fh_4!");}

for(int i=0; i<5; i++){
    if (isnan( Teljes_SP_FutesHutes[i])) { Teljes_SP_FutesHutes[i] = 0.0;}
    if (isnan( Teljes_T_FutesHutes[i])) { Teljes_T_FutesHutes[i] = 0.0;}
}

for(int i=0; i<5; i++){
    if (abs(Teljes_SP_FutesHutes[i]) > 1000) {Teljes_SP_FutesHutes[i] = 0.0;}
    if (abs(Teljes_T_FutesHutes[i]) > 1000) {Teljes_T_FutesHutes[i] = 0.0;}}
    FutesHutes_kiemnet_eredmeny = -150 + ((Teljes_SP_FutesHutes[0]*Teljes_T_FutesHutes[0]+
Teljes_SP_FutesHutes[1]*Teljes_T_FutesHutes[1]+ Teljes_SP_FutesHutes[2]*Teljes_T_FutesHutes[2]+
Teljes_SP_FutesHutes[3]*Teljes_T_FutesHutes[3]+ Teljes_SP_FutesHutes[4]*Teljes_T_FutesHutes[4])/

```

```
(Teljes_T_FutesHutes[0]+Teljes_T_FutesHutes[1]+Teljes_T_FutesHutes[2]+Teljes_T_FutesHutes[3]+Teljes_T_FutesHutes[4]));
```

```
if (isnan( Ablak_kiemnet_eredmeny)) { Ablak_kiemnet_eredmeny = 0.0;}
if (isnan( FutesHutes_kiemnet_eredmeny)) { FutesHutes_kiemnet_eredmeny = 0.0;}

// 9. resz eredmeny vissza alakitasa szovegge:
String Ablak_kiemnet_eredmeny_Str = String(Ablak_kiemnet_eredmeny, 2);
Ablak_kiemnet_eredmeny_Str.replace('.', ',');
Serial.print(Ablak_kiemnet_eredmeny_Str);
Serial.print(";");
String FutesHutes_kiemnet_eredmeny_Str = String(FutesHutes_kiemnet_eredmeny, 2);
FutesHutes_kiemnet_eredmeny_Str.replace('.', ',');
Serial.println(FutesHutes_kiemnet_eredmeny_Str);
//delay(1000);
}
```

```
// 10. resz ablak kimenet reszeinek sp kiszamitas fugveny
```

```
float Sp_ablak(float y, int v) {
    float P1[3]={0.0,0.0,0.0};
    float P2[3]={0.0,0.0,0.0};
    float T_1 = 0.0;
    float T_2 = 0.0;
    float L_1 = 0.0;
    float L_2 = 0.0;
    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 területe
    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
    float Sp_a= 0.0;

    if (v == 0){
        int j = 0;
        P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
        P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
        T_1 = (P1[j]-ki1_msf[j][0]);
        T_2 = (T_1/3.0);
        L_1 = (ki1_msf[j][3]-P2[j]);
        L_2 = (L_1/3.0);
        A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
        S[j][0] = (P1[j]-T_2);
        S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
        S[j][2] = (P2[j]+L_2);
```

```

    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]));
};

if (v == 1){
    int j = 1;
    P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
    P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
    T_1 = (P1[j]-ki1_msf[j][0]);
    T_2 = (T_1/3.0);
    L_1 = (ki1_msf[j][3]-P2[j]);
    L_2 = (L_1/3.0);
    A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
    A[j][1] = (P2[j]-P1[j])*y;
    A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
    S[j][0] = (P1[j]-T_2);
    S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
    S[j][2] = (P2[j]+L_2);
    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]));
};

if (v == 2){
    int j = 2;
    P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
    P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
    T_1 = (P1[j]-ki1_msf[j][0]);
    T_2 = (T_1/3.0);
    L_1 = (ki1_msf[j][3]-P2[j]);
    L_2 = (L_1/3.0);
    A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
    A[j][1] = (P2[j]-P1[j])*y;
    A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
    S[j][0] = (P1[j]-T_2);
    S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
    S[j][2] = (P2[j]+L_2);
    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]));
};

for(int i=0; i<3; i++){
    P1[i] = 0.0;
    P2[i] = 0.0;
    for(int j=0; j<3; j++){
        A[i][j] = 0.0;
        S[i][j] = 0.0;
    }
    T_1 = 0.0;
    T_2 = 0.0;
    L_1 = 0.0;
    L_2 = 0.0;

    return Sp_a;
}

```

```

}

// 11. rész futes/hutes kimenet reszeinek sp kiszamitas fugveny
float Sp_FH(float y, int v) {
    float P1[3]={0.0,0.0,0.0};
    float P2[3]={0.0,0.0,0.0};
    float T_1 = 0.0;
    float T_2 = 0.0;
    float L_1 = 0.0;
    float L_2 = 0.0;
    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 terulete
    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
    float Sp_fh= 0.0;

    if (v == 0){
        int j = 0;
        P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
        P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
        float T_1 = (P1[j]-ki2_msf[j][0]);
        float T_2 = (T_1/3.0);
        float L_1 = (ki2_msf[j][3]-P2[j]);
        float L_2 = (L_1/3.0);
        A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
        S[j][0] = (P1[j]-T_2);
        S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
        S[j][2] = (P2[j]+L_2);
        Sp_fh = ((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]));
    };

    if (v == 1){
        int j = 1;
        P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
        P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
        float T_1 = (P1[j]-ki2_msf[j][0]);
        float T_2 = (T_1/3.0);
        float L_1 = (ki2_msf[j][3]-P2[j]);
        float L_2 = (L_1/3.0);
        A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
        S[j][0] = (P1[j]-T_2);
        S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
        S[j][2] = (P2[j]+L_2);
        Sp_fh = ((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]));
    };
}

```

```

if (v == 2){
int j = 2;
P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
float T_1 = (P1[j]-ki2_msf[j][0]);
float T_2 = (T_1/3.0);
float L_1 = (ki2_msf[j][3]-P2[j]);
float L_2 = (L_1/3.0);
A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
A[j][1] = (P2[j]-P1[j])*y;
A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
S[j][0] = (P1[j]-T_2);
S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
S[j][2] = (P2[j]+L_2);
Sp_fh = ((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]));
};

```

```

if (v == 3){
int j = 3;
P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
float T_1 = (P1[j]-ki2_msf[j][0]);
float T_2 = (T_1/3.0);
float L_1 = (ki2_msf[j][3]-P2[j]);
float L_2 = (L_1/3.0);
A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
A[j][1] = (P2[j]-P1[j])*y;
A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
S[j][0] = (P1[j]-T_2);
S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
S[j][2] = (P2[j]+L_2);
Sp_fh = ((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]));
};

```

```

if (v == 4){
int j = 4;
P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
float T_1 = (P1[j]-ki2_msf[j][0]);
float T_2 = (T_1/3.0);
float L_1 = (ki2_msf[j][3]-P2[j]);
float L_2 = (L_1/3.0);
A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
A[j][1] = (P2[j]-P1[j])*y;
A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
S[j][0] = (P1[j]-T_2);
S[j][1] = (P1[j]+(P2[j]-P1[j])/2);
S[j][2] = (P2[j]+L_2);

```

```

    Sp_fh = ((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]));
};

```

```

for(int i=0; i<3; i++){
    P1[i] = 0.0;
    P2[i] = 0.0;
    for(int j=0; j<3; j++){
        A[i][j] = 0.0;
        S[i][j] = 0.0; }
    T_1 = 0.0;
    T_2 = 0.0;
    L_1 = 0.0;
    L_2 = 0.0;

    return Sp_fh;
}

```

// 12. rész ablak kimenet részeinek terület kiszamitas fugveny

```

float T_ablak(float y, int v) {
    float P1[3]={0.0,0.0,0.0};
    float P2[3]={0.0,0.0,0.0};
    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 területe
    float A_a= 0.0;

```

```

    if (v == 0){
        int j = 0;
        P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
        P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
        A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
        A_a = (A[j][0] + A[j][1] + A[j][2]); }

```

```

    if (v == 1){
        int j = 1;
        P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
        P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
        A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
        A_a = (A[j][0] + A[j][1] + A[j][2]); }

```

```

    if (v == 2){
        int j = 2;
        P1[j] = ((ki1_msf[j][1]-ki1_msf[j][0])*y)+ki1_msf[j][0];
        P2[j] = ((ki1_msf[j][3]-ki1_msf[j][2])*(1-y))+ki1_msf[j][2];
        A[j][0] = (P1[j]-ki1_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;

```

```

A[j][2] = (ki1_msf[j][3]-P2[j])*(y/2);
A_a = (A[j][0] + A[j][1] + A[j][2]); }

for(int i=0; i<3; i++){
    P1[i] = 0.0;
    P2[i] = 0.0;
    for(int j=0; i<3; i++){
        A[i][j] = 0.0; }}

return A_a;
}

// 13. resz futes/hutes kimenet reszeinek terület kiszamitas fugveny
float T_FH(float y, int v2) {
    float P1[3]={0.0,0.0,0.0};
    float P2[3]={0.0,0.0,0.0};
    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 területe

    float T_fhh= 0.0;

    if (v2 == 0){
        int j = 0;
        P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
        P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
        A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
        T_fhh = (A[j][0] + A[j][1] + A[j][2]); }

    if (v2 == 1){
        int j = 1;
        P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
        P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
        A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
        T_fhh = (A[j][0] + A[j][1] + A[j][2]); }

    if (v2 == 2){
        int j = 2;
        P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
        P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
        A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
        A[j][1] = (P2[j]-P1[j])*y;
        A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
        T_fhh = (A[j][0] + A[j][1] + A[j][2]); }

    if (v2 == 3){
        int j = 3;

```

```

P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
A[j][1] = (P2[j]-P1[j])*y;
A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
T_fhh = (A[j][0] + A[j][1] + A[j][2]); }

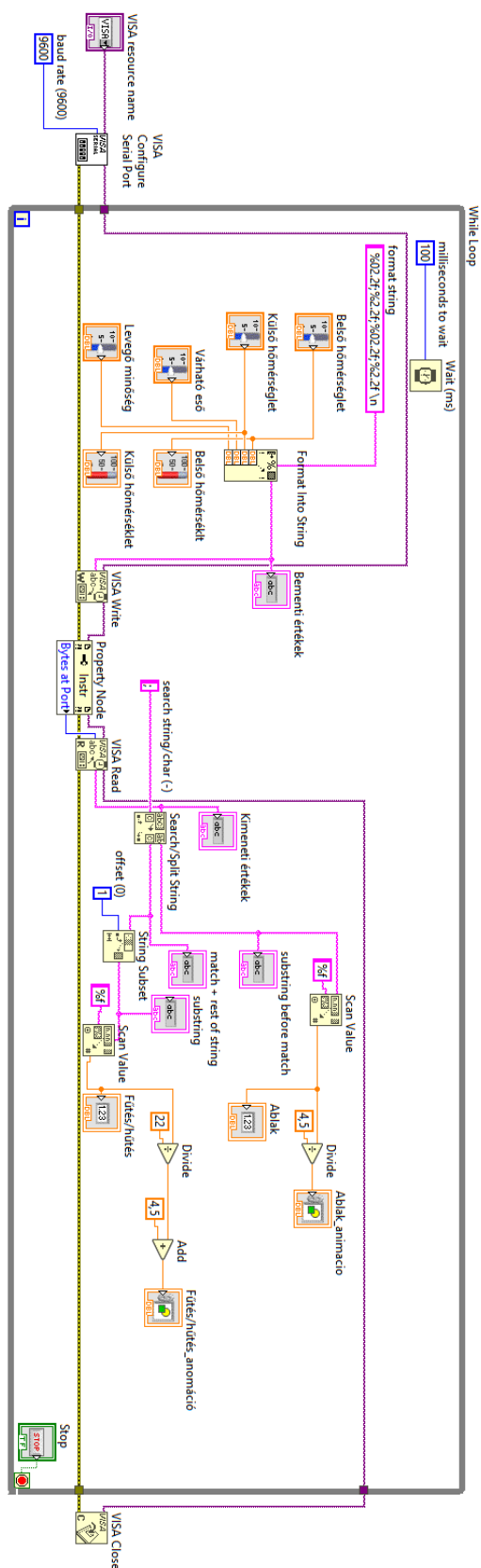
if (v2 == 4){
int j = 4;
P1[j] = ((ki2_msf[j][1]-ki2_msf[j][0])*y)+ki2_msf[j][0];
P2[j] = ((ki2_msf[j][3]-ki2_msf[j][2])*(1-y))+ki2_msf[j][2];
A[j][0] = (P1[j]-ki2_msf[j][0])*(y/2);
A[j][1] = (P2[j]-P1[j])*y;
A[j][2] = (ki2_msf[j][3]-P2[j])*(y/2);
T_fhh = (A[j][0] + A[j][1] + A[j][2]); }

for(int i=0; i<3; i++){
    P1[i] = 0.0;
    P2[i] = 0.0;
    for(int j=0; i<3; i++){
        A[i][j] = 0.0; }}

return T_fhh;
}

```

## 2. melléklet: A teljes LabVIEW Block Diagram



### 3. melléklet: Arduino és MATLAB által számított eredmények

| Számítást végző eszköz     |                        | MATLAB         | Arduino Nano   |
|----------------------------|------------------------|----------------|----------------|
| <b>1. 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>1. Végeredmény</b>      | Ablak nyitás [%]       | <b>9,36</b>    | <b>9,27</b>    |
|                            | Fűtés / hűtés [+/- %]  | <b>-78,80</b>  | <b>-78,70</b>  |
| <b>2. Bemeneti értékek</b> | Belső hőmérséklet [°C] | 17,24          | 17,24          |
|                            | Külső hőmérséklet [°C] | 8,52           | 8,52           |
|                            | Várható eső [min]      | 20,20          | 20,20          |
|                            | Levegő minőség [%]     | 54,08          | 54,08          |
| <b>2. Végeredmény</b>      | Ablak nyitás [%]       | <b>22,60</b>   | <b>22,03</b>   |
|                            | Fűtés / hűtés [+/- %]  | <b>5,96</b>    | <b>7,46</b>    |
| <b>3. Bemeneti értékek</b> | Belső hőmérséklet [°C] | 23,88          | 23,88          |
|                            | Külső hőmérséklet [°C] | 13,01          | 13,01          |
|                            | Várható eső [min]      | 45,92          | 45,92          |
|                            | Levegő minőség [%]     | 62,24          | 62,24          |
| <b>3. Végeredmény</b>      | Ablak nyitás [%]       | <b>79,70</b>   | <b>79,04</b>   |
|                            | Fűtés / hűtés [+/- %]  | <b>0,00</b>    | <b>0,00</b>    |
| <b>4. Bemeneti értékek</b> | Belső hőmérséklet [°C] | 23,88          | 23,88          |
|                            | Külső hőmérséklet [°C] | 27,01          | 27,01          |
|                            | Várható eső [min]      | 27,93          | 27,93          |
|                            | Levegő minőség [%]     | 75,27          | 75,27          |
| <b>4. Végeredmény</b>      | Ablak nyitás [%]       | <b>4,17</b>    | <b>4,06</b>    |
|                            | Fűtés / hűtés [+/- %]  | <b>-92,10</b>  | <b>-91,94</b>  |
| <b>5. Bemeneti értékek</b> | Belső hőmérséklet [°C] | 26,88          | 26,88          |
|                            | Külső hőmérséklet [°C] | 37,83          | 37,83          |
|                            | Várható eső [min]      | 0,00           | 0,00           |
|                            | Levegő minőség [%]     | 83,24          | 83,24          |
| <b>5. Végeredmény</b>      | Ablak nyitás [%]       | <b>0,00</b>    | <b>0,03</b>    |
|                            | Fűtés / hűtés [+/- %]  | <b>-100,00</b> | <b>-100,00</b> |

#### 4. melléklet: A mérés során használt kód

```
#include <dht11.h>

#define DHT11PIN_1 2 // Kinti szenzor lábainak csatlakozása
#define DHT11PIN_2 3 // Benti szenzor lábainak csatlakozása

dht11 DHT11_1;
dht11 DHT11_2;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int chk_1 = DHT11_1.read(DHT11PIN_1);
    int chk_2 = DHT11_2.read(DHT11PIN_2);
    Serial.print((float)DHT11_1.temperature, 0);
    Serial.print("]"); // karakter kiírása, mely mentén Excelbe szét lehet bontani több oszlopra
    Serial.println((float)DHT11_2.temperature, 0);

    delay(1000);
}
```