

---

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

**SISTEM INTERACTIV CU**  
**SENZORI**  
**PROIECT DE DIPLOMĂ**

**Coordonator științific:**

**Șef. lucr. Dr. ing. Turos László-Zsolt**

**Absolvent:**

**Csiki Krisztina**

**2023**

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Specializarea: **Calculatoare**

Viza facultății:



## LUCRARE DE DIPLOMĂ

Coordonator științific:  
**ș.l. dr. ing. Turos László-Zsolt**

Candidat: **Csiki Krisztina**  
Anul absolvirii: **2023**

**a) Tema lucrării de licență:**

SISTEM INTERACTIV CU SENZORI

**b) Problemele principale tratate:**

- Studiu bibliografic privind senzori și sisteme cu senzori
- Proiectarea protocolului serial de tip human readable
- Realizarea unei GUI pentru configurarea sistemului embedded și pentru vizualizarea datelor
- Realizarea aplicației Arduino responsabil pentru comunicația cu aplicația GUI, configurarea dinamică a senzorilor conectate la sistemul încorporat, logarea și vizualizarea datelor măsurate

**c) Desene obligatorii:**

- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.
- Imagini cu software-ul realizat

**d) Softuri obligatorii:**

- Aplicația Arduino de configurare dinamică a senzorilor
- Aplicația GUI de configurare a senzorilor și de vizualizare a datelor obținute

**e) Bibliografia recomandată:**

- Túrós László-Zsolt, Székely Gyula: Érzékelők és mérőhálózatok, Scientia kiadó, 2022
- <https://www.arduino.cc/en/software>
- Javaid, M., Haleem, A., Rab, S., Singh, R. P., & Suman, R. (2021b). Sensors for daily life: A review. *Sensors International*

**f) Termene obligatorii de consultații: săptămânal**

**g) Locul și durata practicii:** Universitatea „Sapientia” din Cluj-Napoca,  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Primit tema la data de: 31.03.2022  
Termen de predare: 28.06.2023

Semnătura Director Departament



Semnătura coordonatorului



Semnătura responsabilului  
programului de studiu



Semnătura candidatului

Csiki

---

## Declarație

Subsemnata/ul Csiki Krisztina, absolvent(ă) al/a specializării Calculatoare, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureș

Data: 28.06.2023

Absolvent

Semnătura.....Csiki.....

---

# Sistem interactiv cu senzori

## Extras

Utilizarea senzorilor a devenit o parte a vieții noastre de zi cu zi, avem în mod constant posibilitatea de a măsura caracteristicile fizice, chimice și biologice ale lumii din jurul nostru. Tema tezei mele este implementarea unui sistem de senzori interactivi într-un mediu Arduino, care facilitează integrarea senzorilor în sistem, ceea ce poate fi o provocare pentru un utilizator care abia începe să cunoască lumea Arduino și a microcontrolerelor.

Pentru funcționarea sistemului era necesar de un text, citibil de ochiul omenesc, un protocol pe bază ASCII cu ajutorul căruia senzorii pot fi configurate dinamic și datele pot fi transmise în mod uniform. În sistem au fost integrați mai mulți senzori diferiți, cu ajutorul cărora pot fi măsurate diferite valori și pot fi urmărite și schimbarea acestor valori.

Pentru utilizatori a fost realizată o interfață grafică, care include o aplicație desktop, prin care pot fi afișate datele în timp real măsurate de senzorii conectați la Arduino și este posibilă și salvarea și revizuirea datelor.

Între aplicație și microcontroler se realizează o comunicare serială. Am conceput aplicația pentru a fi utilizată în scopuri de laborator.

**Cuvinte cheie:** Arduino, protocol, senzori, desktop application

**SAPIENTIA ERDÉLYI MAGYAR**

**TUDOMÁNYEGYETEM**

**MAROSVÁSÁRHELYI KAR**

**SZÁMÍTÁSTECHNIKA SZAK**

**INTERAKTÍV**

**ÉRZÉKELŐRENDSZER**

**DIPLOMADOLGOZAT**

**Témavezető:**

**Dr. Turos László-Zsolt,**  
**egyetemi adjunktus**

**Végzős hallgató:**

**Csiki Krisztina**

---

# 2023

## Kivonat

Az érzékelők használata a mindennapjaink részévé vált, folyamatosan lehetőségünk van a minket körülvevő világban lévő fizikai, kémiai, biológiai jellemzők mérésére. Dolgozatom témája egy interaktív érzékelőrendszer megvalósítása Arduino környezetben, ami megkönnyíti az érzékelők integrálását a rendszerbe, ami kihívást jelenthet egy felhasználó számára, aki még csak ismerkedik az Arduino és a mikrovezérlők világával.

A rendszer működéséhez szükség volt egy szöveges, emberi szem által olvasható ASCII alapú általános protokoll megírására, ami segítségével az érzékelők dinamikusan konfigurálhatóak és az adatok egységes adatformában továbbíthatóak. Az érzékelőrendszerbe több különböző szenzor is integrálva lett, melyek által különböző fizikai mennyiségek mérésére van lehetőség és az értékek változásának a nyomon követésére.

A felhasználók számára készült egy felhasználói felület, ami egy asztali alkalmazást foglal magába, amin keresztül megjeleníthetők az Arduinohoz kötött szenzorok által mért valós idejű adatok, illetve lehetőség nyílik az adatok mentésére és visszanézésére is. Az alkalmazás és mikrovezérlő között soros kommunikáció valósul meg.

Az alkalmazást laboratóriumi célokra való felhasználásra terveztem meg.

**Kulcsszavak:** Arduino, protokoll, érzékelők, asztali alkalmazás

---

# Interactive sensor system

## Abstract

Nowadays, using sensors has become so popular that we do not even notice how easy it is for us to measure the world around us. Today we have the possibility and the technology to measure the physical, chemical and the biological characteristics of the world around us.

The goal of my thesis was to build a scalable system for sensors, created with the help of an Arduino, which helps in the integration of new sensors in the system for users, who may face a challenge when integrating a new sensor in an environment with Arduino and microcontrollers.

At the beginning I had to create a text-based protocol, written with ASCII code, which provides for the user an interface for dynamic configuration of the sensors and for the collective data forwarding.

Several different sensors have been integrated into the sensor system, with which it is possible to measure different physical quantities and monitor changes in values, but also an interface for the collected data.

For the better understanding of the real-time data collected by the sensors with the help of the Arduino I have also created a desktop application, , through which the real-time data measured by the sensors connected to the Arduino can be displayed, and it is also possible to save and review the data. The communication between the application and the sensors is realized by serial communication.

I designed the application to be used for laboratory purposes.

**Keywords:** Arduino, protocol, sensors, desktop application

## Tartalomjegyzék

1. Bevezető.....	11
1.1. Bevezető .....	11
1.2. Célkitűzések.....	11
2. Elméleti megalapozás .....	13
2.1. Arduino.....	13
2.2. Arduino Uno .....	13
2.3. Arduino IDE .....	15
2.4. Soros kommunikáció .....	16
2.5. ASCII protokoll .....	17
2.6. Érzékelők .....	18
2.6.1. Akadályelkerülő érzékelő.....	19
2.6.2. Analóg hőmérséklet érzékelő .....	20
2.6.3. Analóg mágneses Hall érzékelő .....	20
2.6.4. Digitális hőmérséklet érzékelő .....	21
2.6.5. Fényblokkolás érzékelő.....	22
2.6.6. Hőmérséklet és páratartalom érzékelő .....	22
2.6.7. Vonalkövető érzékelő .....	23
2.7. Asztali alkalmazás.....	24
3. A rendszer specifikációi és architektúrája .....	25
3.1. Követelmény specifikáció .....	27
3.1.1. Felhasználói követelmények .....	28
3.1.2. Rendszerkövetelmények .....	29
4. Részletes tervezés és gyakorlati megvalósítás .....	32
4.1. Arduino kód.....	32
4.1.1. Protokoll működésének bemutatása.....	33
4.2. Asztali alkalmazás.....	37
4.2.1. Kezdőoldal.....	39
4.2.2. Érzékelő adatlapja oldal .....	40
4.2.3. Konfigurációs oldal.....	41
4.2.4. Adatok megjelenítése oldal .....	43
4.2.5. Folytonos olvasás oldal .....	45
4.2.6. Korábbi mérések oldal .....	47
5. Következtetések.....	50
5.1. Megvalósítások .....	50
5.2. Továbbfejlesztési lehetőségek .....	50
6. Irodalomjegyzék .....	52



## Ábrák jegyzéke

1. ábra - Arduino Uno áramköri lap .....	13
2. ábra - Arduino Uno perifériáinak lábkiosztása .....	14
3. ábra - Arduino IDE .....	16
4. ábra - Soros kommunikáció .....	16
5. ábra - ASCII kódtábla .....	17
6. ábra - Érzékelő működési elve .....	18
7. ábra – Breadboard.....	19
8. ábra - HW-488 érzékelő.....	19
9. ábra - HW-498 érzékelő.....	20
10. ábra - HW-495 érzékelő.....	21
11. ábra - HW-506 érzékelő.....	21
12. ábra - HW-487 érzékelő.....	22
13. ábra - DHT11 érzékelő.....	23
14. ábra - HW-511 érzékelő.....	23
15. ábra - .NET MAUI.....	24
16. ábra – A rendszer vázlata.....	25
17. ábra - Tömbvázlat .....	26
18. ábra - Arduinohoz kötött érzékelők .....	27
19. ábra - Alkalmazás Használat-Eset diagramja.....	28
20. ábra - Digitális bemenet konfigurálása .....	34
21. ábra - Hibás konfigurációs parancs küldése.....	34
22. ábra - Olvasási parancs küldése.....	34
23. ábra – Érzékelők nevei és a hozzá tartozó függvények map tárolója .....	35
24. ábra - hibás olvasási parancs küldése .....	35
25. ábra - Hibás olvasási parancs küldése.....	35
26. ábra - Folyamatos olvasási parancs küldése.....	36
27. ábra - Folyamatos olvasás leállítása .....	36
28. ábra - Ismeretlen parancs küldése.....	36
29. ábra – Soros kommunikáció elindítása .....	37
30. ábra - Függvény a kommunikációs kapcsolat létrehozására.....	38
31. ábra - Hivatkozás könyvtár csomagokra.....	38
32. ábra - MVVM tervezési minta.....	39
33. ábra – Alkalmazás kezdőoldala .....	40
34. ábra - Kiválasztott bekötési rajz megjelenítése .....	40
35. ábra - Érzékelő adatlap oldala .....	41
36. ábra - Kivezetés konfigurálása .....	42
37. ábra - Konfigurálás után kapott visszajelzések .....	42
38. ábra - Nincs csatlakoztatva az Arduino áramköri lap .....	42
39. ábra - Érzékelő adatainak beolvasása .....	43
40. ábra - Hibaüzenet, ha nem volt érzékelő kiválasztva .....	43
41. ábra – Szenzor adatok megjelenítése oldal .....	44
42. ábra - Üres oldal .....	45
43. ábra - Folytonos olvasás megjelenítése .....	46
44. ábra - Folytonos olvasás folytatása.....	46
45. ábra – Jelmagyarázat.....	47
46. ábra - Mérési adatok listázása .....	47

47. ábra - Mentett mérési adatok.....	48
48. ábra - Sikeres mentés .....	48
49. ábra - Navigációs függvény .....	49
50. ábra - Gomb adattagjainak megadása .....	49

### **Táblázatok jegyzéke**

1. táblázat - Arduino Uno paraméterei .....	15
2. táblázat - HW-488 paraméterei .....	19
3. táblázat - HW-498 paraméterei .....	20
4. táblázat - HW-495 paraméterei .....	21
5. táblázat - HW-506 paraméterei .....	22
6. táblázat - HW-487 paraméterei .....	22
7. táblázat - DHT11 paraméterei .....	23
8. táblázat - HW-511 paraméterei .....	24

# 1. Bevezető

## 1.1. Bevezető

Az információk világában élünk, amikor azt szeretnénk, hogy minél több információhoz hozzáférhessünk a környezetünkben. A különböző érzékelők mára már nagyon elterjedtek, a mindennapjaink részévé váltak. Egyre több elektronikus készülék kerül a piacra, amik működéséhez elengedhetetlen az érzékelők használata és ezáltal azok folyamatos fejlődése. Szenzorok használata által lehetőség van a környezetünk fizikai, kémiai, biológiai tulajdonságainak mérésére és ezek villamos mennyiséggé alakítására.

Az érzékelők megfelelő összekötése, több érzékelő integrálása egyetlen rendszerbe kihívást jelenthet, mivel a szenzorok különböző típusúak, az adatokat nem egységes formában továbbítják. Az interaktív érzékelőrendszer használatával ez tehető könnyebbé, egy olyan felhasználó számára, aki szeretne kicsit jobban elmerülni az érzékelők és az Arduino, mikrovezérlők világában.

Egy protokoll segítségével egységes formában érhetőek el az érzékelők adatai, lehetőséget teremtve az adatok elemzésére, azok összehasonlítására. A rendszer segítségével nincs szükség arra, hogy minden, a rendszerben megtalálható szenzor kódját külön-külön megírja, hanem ehhez kap egy egységes felületet a felhasználó. A rendszer lehetőséget teremt különböző típusú érzékelők egyszerű és hatékony integrálására. A szenzorok használatával a rendszer felhasználói követhetik a hőmérséklet, páratartalom változását a környezetükben, de más mérések elvégzése is lehetséges.

Ha adott a felhasználó számára egyetlen felület, amelyen keresztül kezelhető egy rendszer az a felhasználóban is pozitív benyomást kelt, hiszen megkönnyíti számára egy feladat elvégzését, könnyedén hozzájuthat minden információhoz az érzékelőkkel kapcsolatban. Az egységes felület előnyei a hatékonyság, felhasználói élmény javítása, idő spórlása. A dolgozatban bemutatott interaktív érzékelőrendszer egy könnyen kezelhető felületet nyújt a felhasználó számára az érzékelők világában, így elősegítve, hogy minél jobban megismerhesse azt.

## 1.2. Célkitűzések

A dolgozat elkészítése során a következő célok voltak kitűzve:

- Szöveges parancsértelmező kódjának a megírása, általános protokoll kivitelezése: egy olyan kód megírása, ami segítségével szöveges parancsokat lehet küldeni és fogadni a mikrovezérlőn, a szenzorok dinamikus konfigurálása, adatainak továbbítása is ezen keresztül valósul meg. A protokollnak biztosítania kell, hogy az adatok egységes formában legyenek továbbítva, valamint minden elküldött parancsra érkezzen egy válasz.
- Különböző érzékelők a rendszerbe való integrálása és programozása: a különböző szenzorok kódjának összegyűjtése és azok olyan formában történő átírása, amelyek a protokoll alapján képesek megvalósítani az adatgyűjtést. Az érzékelők dinamikus konfigurálása, ami azt jelenti, hogy a rendszer használata közben választható ki, hogy melyik kivezetésre kötjük a szenzort, nem kell ezt már a kódban előre meghatározni. Lehetőség legyen arra, hogy egy rendszerhez egyszerre több érzékelőt is lehessen csatlakoztatni, így egy időben több különböző szenzor által küldött adat is rögzíthető.
- Felhasználói felület készítése: egy olyan alkalmazás elkészítése, ami lehetővé teszi a felhasználó számára, hogy a szenzorokat dinamikus konfigurálja és az azoktól kapott adatokat megjelenítse. A felület tervezése során fontos figyelembe venni, hogy a felület olyan legyen, aminek használatához nincs szükség hosszú tanulási időre, átlátható, felhasználóbarát. Legyen lehetősége a felhasználónak többet megtudni a kiválasztott érzékelőről, anélkül, hogy külön keresést kellene ehhez végeznie, így az alkalmazáson belül legyenek megjeleníthetők a kapcsolási rajzok, megnyithatóak a szenzorok adatlapjai. A mérési adatok rögzítése és listázása, így összehasonlíthatók a különböző időpontokban végzett mérések.

## 2. Elméleti megalapozás

### 2.1. Arduino

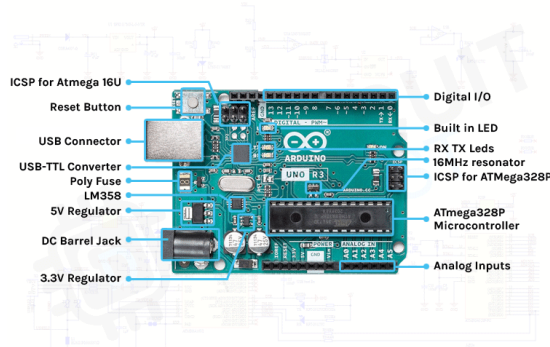
Az Arduino egy elektronikus eszközöket és szoftvereket tervező, gyártó, támogató hardver- és szoftvercég, ami lehetővé teszi, hogy az emberek hozzáférhessenek azokhoz a fejlett technológiákhoz, amelyek által kapcsolatba lépnek a fizikai világgal. Termékeik egyszerűek, nagy teljesítményűek és képesek arra, hogy a felhasználók igényeinek széles skáláját kielégítsék, kezdve a diákoktól a professzionális felhasználókig. [1]

Az Arduino projekt 2005-ben, Olaszországban indult, azzal a céllal, hogy olcsó és egyszerű megoldást nyújtson olyan eszközök létrehozására, amelyek érzékelők használatának segítségével kapcsolatba lépnek környezetükkel. [2]

Az Arduino két részből tevődik össze: az Arduino IDE-ből, vagyis egy nyílt forráskódú fejlesztői környezetből és az Arduino lapból. Az Arduino lapoknak számos fajtája létezik, amik méretben, mikrovezérlő típusában, ki- és bemenetek számában különböznek egymástól.[3]

### 2.2. Arduino Uno

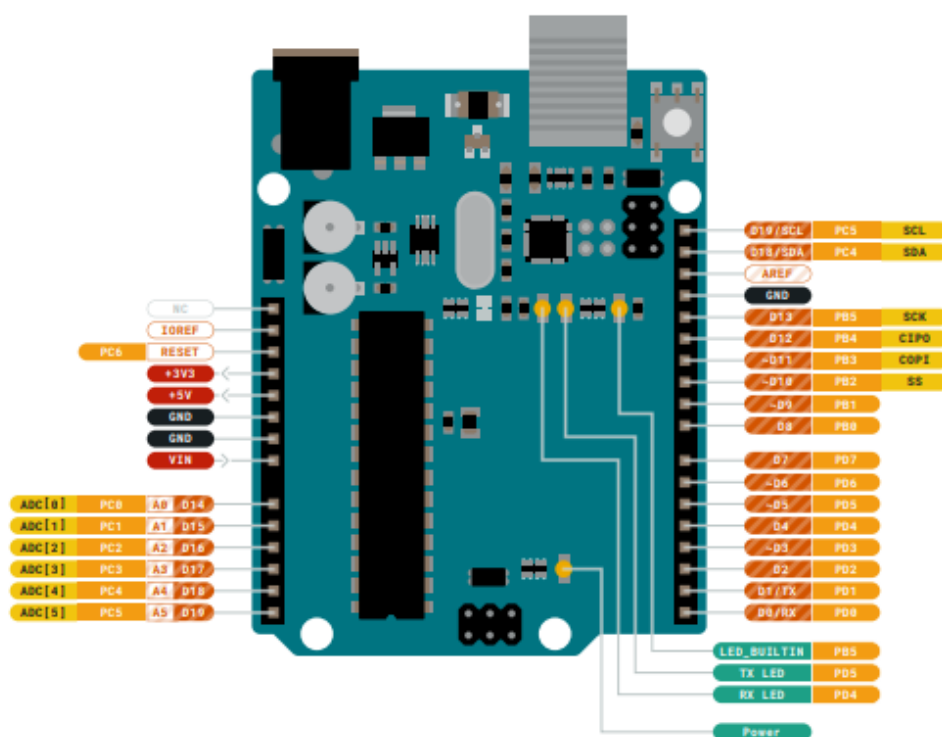
A rendszer kialakításánál egy Arduino Uno (1. ábra) típusú lapot használtam. Az érzékelőrendszer kialakításánál azért esett a választás, Arduino Uno mikrovezérlőre, mivel ez egy széles körben elterjedt típus, sok dokumentáció van, amelyek segítségével szolgálhatnak a megvalósítás során, valamint rengeteg könyvtár megtalálható, amelyek a fejlesztésben felhasználhatóak. Ez a mikrovezérlő rendelkezik elég sok kivezetéssel, így a rendszer jól bővíthető, egy időben több szenzor is rácsatlakoztatható.



1. ábra - Arduino Uno áramköri lap<sup>1</sup>

<sup>1</sup> Forrás: <https://circuitdigest.com/article/everything-you-need-to-know-about-arduino-uno-board-hardware>

Megtalálható benne egy ATmega328/P típusú, 5V feszültségen működő, 8 bites mikrovezérlő. Rendelkezik 2 KB SRAM memóriával, 1 KB EEPROM memóriával a paraméterek tárolására és 32 KB flash memória áll rendelkezésre a programok tárolására. A lapon még megtalálhatóak az analóg és digitális ki/bemenetek, amiken keresztül kapcsolhatóak az érzékelők az Arduinohoz. A lap két oldalán található tűksor aljzat melletti számokkal (2. ábra) hivatkozhatunk a kódban a megfelelő kivezetésre. [4]



2. ábra - Arduino Uno perifériáinak lábkiosztása<sup>2</sup>

Az Arduino egyik érdekessége, hogy a megírt kód feltöltés után az azonnal fut rajta, ez a bootloader segítségével valósul meg. Az USB csatlakozónak kettős szerepe van, ezen keresztül tölthető fel rá a program valamint biztosítható a működéséhez szükséges 5V feszültség, ez megoldható a külső tápcsatlakozó használatával is. [5]

Az alábbi táblázatban (1. táblázat) megtalálhatóak az Arduino Uno legfőbb jellemzői:

<sup>2</sup> Forrás: <https://docs.arduino.cc/retired/boards/arduino-uno-rev3-with-long-pins>

Mikrovezérlő	ATMega328/P
Órajel	16MHz
Működési feszültség	5V
Analóg ki/bemenetek száma	6
Digitális ki/bemenetek száma	14
SRAM	2KB
EEPROM	1KB
Flash memória	32KB
Maximális tápfeszültség (nem ajánlott)	20V

*1. táblázat - Arduino Uno paraméterei*

### **2.3. Arduino IDE**

Az Arduino IDE az a fejlesztői környezet, amelyben programozható az Arduino Uno. A kód C/C++ nyelven írható meg. Sok könyvtár is elérhető, amik nagy segítséget nyújtanak a programkód megírásában. A környezet áll egy szövegszerkesztőből és egy fordítóból, az Arduinotól kapott válaszokat a Serial monitor és Serial plotter segítségével lehet követni, megjeleníteni.

Egy új program fájl létrehozásakor (3. ábra) egy sablon jelenik meg, aminek két fő része van: a setup() és a loop() függvény. A setup függvényben történik az inicializáció, ez egyszer fut le, itt adható meg például az adatok átviteli sebessége (baud rate). A loop függvény az egy végtelen ciklus, ami addig fut, amíg a mikrovezérlő feszültség alatt van.

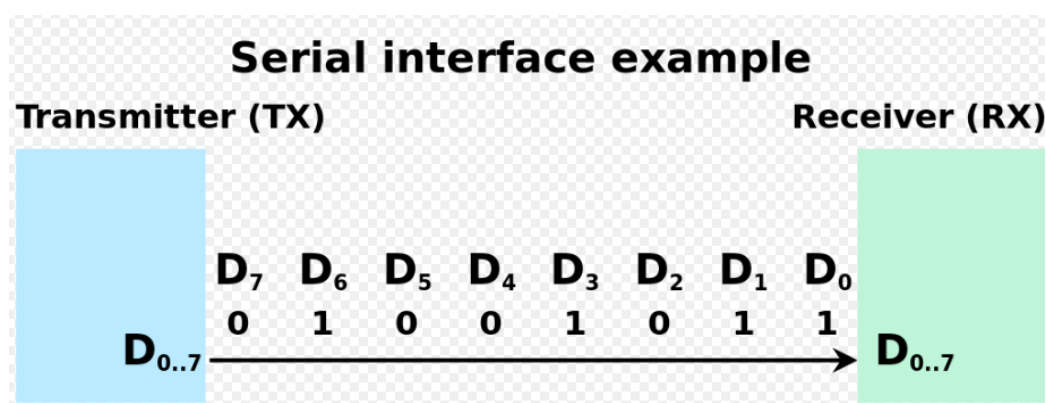
A megírt kód helyessége ellenőrizhető még az Arduinora való feltöltés előtt a Verify gomb megnyomásával, majd utána ha ki lett választva a megfelelő áramköri lap, esetemben ez az Arduino Uno, és a megfelelő port az Upload gomb megnyomásával már feltölthető a kód, ami azonnal fut.



3. ábra - Arduino IDE

## 2.4. Soros kommunikáció

Soros kommunikáció során a küldött üzenet minden bitje egymás után kerül elküldésre egy soros vonalon keresztül (4. ábra) és abban a sorrendben is érkeznek meg a vevőhöz, ahogyan el lettek küldve. Az Arduino Uno USB csatlakozója lehetővé teszi a soros kommunikáció megvalósulását. A kommunikációs csatornán egy időpillanatban mindig csak egy bit halad át, a kommunikáció sebességét határozza meg az Arduino kód inicializálásakor megadott adatátviteli sebesség. [7] Arduino kód írása során a Serial osztály segítségével valósítható meg az áramkörti lap és a számítógép közötti soros kommunikáció, minden típusú Arduino lap rendelkezik legalább egy soros porttal.



4. ábra - Soros kommunikáció<sup>3</sup>

<sup>3</sup> Forrás: [Serial and Parallel Data Transmission - Serial communication - Wikipedia](#)



## 2.5. ASCII protokoll

Az ASCII<sup>4</sup> kód 128 különböző karakterből épül fel, ami az angol abc betűit, számokat, írásjeleket tartalmaz (5. ábra). Az ASCII protokoll egy kérdés-válasz alapú kommunikációs protokoll, ami abból áll, hogy a számítógép ASCII karaktereket használva küld el egy parancsot az eszköznek, amelyre majd választ kap az eszköztől. Eszközök konfigurálására, adatok küldésére, azok olvasására használják az ASCII parancskészletet. A protokollban minden parancs ASCII karakterekből épül fel, minden parancs egy előtag elválasztó karakterrel kezdődik és egy záró karakterrel végződik. Minden használt karakter beírható egy számítógép billentyűzetéről. Az ASCII protokollt először az 1980-as években vezették be az RS-485 elektronikai szabványon alapuló eszközök közötti kommunikáció során.[8]

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	\$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

5. ábra - ASCII kódtábla<sup>5</sup>

Általában három lezáró karaktert használnak:

- visszatérítési karakter: a terminálban az Enter leütése után ez a karakter kerül elküldésre, karakterértéke a 13 (0D), <CR> formában gyakran használják, ami a Carriage Return rövidítése, programkódban '\r' formában adható meg

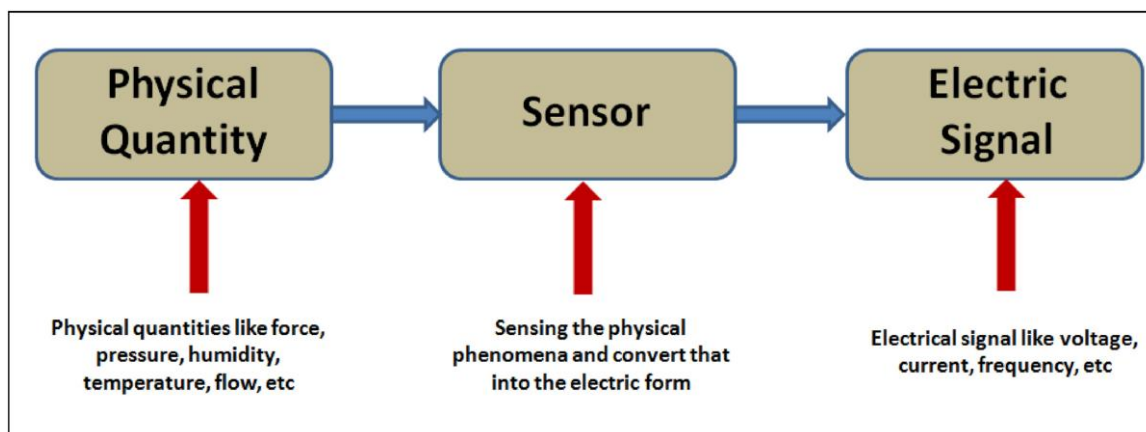
<sup>4</sup> American Standard Code for Information Interchange

<sup>5</sup> Forrás: <https://www.sciencebuddies.org/cdn/references/ascii-table.png>

- új sor karakter: ez az <LF> vagyis Line Feed karakter, ez a '\n' alakban írható le, az ASCII táblázatban kódja 10 (0A)
- null karakter: ez a táblázat első eleme, <NUL> formában használják [9]

## 2.6. Érzékelők

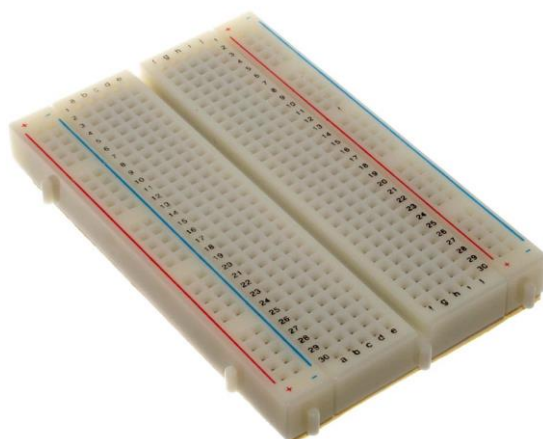
A szenzorok olyan eszközök, amelyek eseményeket, változásokat érzékelnek a környezetükben és ezt az információt továbbítják (6. ábra). Az érzékelők a fizikai jelenségeket mérhető villamos digitális jellé alakítják.[6] Az érzékelők statikus jellemzői közé tartozik a mérési tartomány, érzékenység, kimeneti jeltartomány, túlterhelési tartomány, ismétlőképesség, hibasáv, a dinamikus tulajdonságokhoz sorolható a holtidő, beállási idő, növekedési idő, túllövés. [10]



6. ábra - Érzékelő működési elve<sup>6</sup>

Az Arduino Uno áramköri laphoz egyaránt csatlakoztatható analóg és digitális érzékelő is, így ennek a köszönhetően az érzékelőrendszerben is megtalálható mindkettő. Ahhoz, hogy egyszerre több szenzort is lehetőség legyen az Arduinohoz kötni szükség van egy breadboard-ra (7. ábra). Ennek segítségével egyszerűen módosítható az áramkör, forrasztásmentesen összeköthetőek az elemek jumperek használatával. Fontos megjegyezni, hogy 5V tápfeszültség esetén, a csatlakoztatott érzékelők össz áramfelvétele nem haladhatja meg a 200mA értéket.

<sup>6</sup> Forrás: <https://www.sciencedirect.com/science/article/pii/S2666351121000425#sec2>



7. ábra – Breadboard

### 2.6.1. Akadályelkerülő érzékelő

Az alábbi ábrán látható (8. ábra) érzékelő érzékeli egy akadály jelenlétét a szenzor előtt, mindezt infravörös jel használatával. A beépített potenciométer segítségével állítható az érzékelési tartomány nagysága. Működési elve: található benne egy infravörös LED adó és vevőegység, ha az adó által kibocsátott fény egy akadályba ütközik, akkor az visszaverődik a vevőre, így érzékelve az előtte lévő tárgyakat.



8. ábra - HW-488 érzékelő<sup>7</sup>

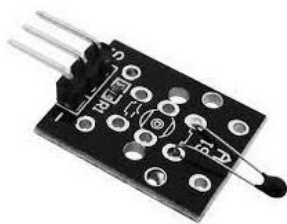
Működési feszültség	3.3V-5V
Áramfelvétel	$\geq 20\text{mA}$
Működési hőmérsékleti tartomány	$-10^{\circ}\text{C} \div 50^{\circ}\text{C}$
Érzékelési távolság	2cm-40cm

2. táblázat - HW-488 paraméterei

<sup>7</sup> Forrás: <https://shop.tavir.hu/wp-content/uploads/sen-ir-contact-3.jpg>

### 2.6.2. Analóg hőmérséklet érzékelő

A szenzor (9. ábra) áll egy NTC<sup>8</sup> termisztorból valamint egy 10k $\Omega$  ellenállásból. A termisztor ellenállása a környező hőmérsékletnek megfelelően változik és az ellenállás értéke felhasználható a pillanatnyi hőmérséklet kiszámításához.



9. ábra - HW-498 érzékelő<sup>9</sup>

Működési feszültség	5V
Mérési tartomány	-40°C÷125°C
Pontosság (0°C÷70°C között)	±0.2°C
Válaszidő	~10s

3. táblázat - HW-498 paraméterei

### 2.6.3. Analóg mágneses Hall érzékelő

A HW-498 (10. ábra) szenzor egy kisméretű Hall érzékelő, amit egy állandó mágnes vagy elektromágnes mágneses mezője működtet. A kimenő feszültség a tápfeszültségtől függ és arányosan változik a mágneses mező erősségével. A kimenő feszültség alapértelmezetten a tápfeszültség fele (a bemutatott rendszer esetében ez 2.5V körüli érték), ami a polaritás függvényében változik.

---

<sup>8</sup>NTC: Negative Temperature Coefficient

<sup>9</sup> Forrás: <https://m.media-amazon.com/images/I/51mbaAbnQKS.jpg>



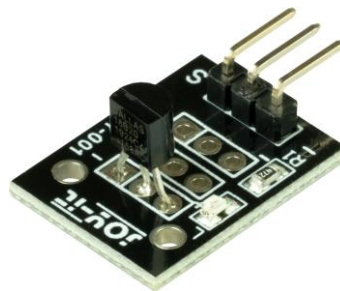
10. ábra - HW-495 érzékelő<sup>10</sup>

Működési feszültség	4.5V-6V
Működési hőmérsékleti tartomány	-40°C÷80°C
Áramfelvétel	3.5mA

4. táblázat - HW-495 paraméterei

#### 2.6.4. Digitális hőmérséklet érzékelő

A 1-Wire interfész csak egyetlen áramköri kivezetést igényel a kommunikációhoz, ez a hőmérséklet érzékelő (11. ábra) is ez szerint működik. A mért jelet 12 bitre képes felbontani, ezért magas a pontossága, használják ipari rendszerekben, termosztatikus vezérlésben.



11. ábra - HW-506 érzékelő<sup>11</sup>

<sup>10</sup> Forrás: <https://www.electronics-lab.com/wp-content/uploads/2018/04/hall-effect.jpg>

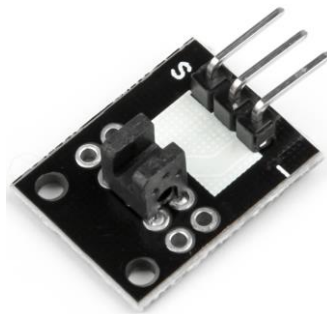
<sup>11</sup> Forrás: [https://cleste.ro/2739-superlarge\\_default/modul-senzor-de-temperatura-ds18b20.jpg](https://cleste.ro/2739-superlarge_default/modul-senzor-de-temperatura-ds18b20.jpg)

Működési feszültség	3V-5.5V
Mérési tartomány	-55°C÷125°C
Pontosság (-10°C÷80°C között)	±0.5°C
Programozható felbontóképesség	9-12 bit
Konverziós idő 12 biten	750ms
Áramfelvétel	1.5mA

5.táblázat - HW-506 paraméterei

### 2.6.5. Fényblokkolás érzékelő

A 12. ábra egy fényblokkoló szenzort ábrázol, az érzékelő egyik oldalán egy optikai emitter és kollektor található, a másik oldalán két ellenállás (1Ω és 33Ω). Az emitter egy fénysugarat bocsát ki, ha ez nem jut el a kollektorhoz akkor egy tárgy elzárja a köztük lévő utat.



12. ábra - HW-487 érzékelő<sup>12</sup>

Működési feszültség	3.3V-5V
Működési hőmérsékleti tartomány	-25°C÷85°C
Áramfelvétel	30mA
Emitter-kollektor közti távolság	0.2mm

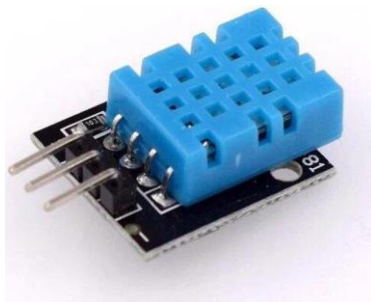
6. táblázat - HW-487 paraméterei

### 2.6.6. Hőmérséklet és páratartalom érzékelő

Ez egy digitális érzékelő (13. ábra), amely egyidőben képes a hőmérséklet és a levegő nedvességtartalmának mérésére, ez annak köszönhető, hogy a szenzor két elemből épül fel: egy

<sup>12</sup> Forrás: [https://cdn.shopify.com/s/files/1/2285/0583/products/56\\_480x480.jpg?v=1608198336](https://cdn.shopify.com/s/files/1/2285/0583/products/56_480x480.jpg?v=1608198336)

kapacitív páratartalom érzékelőből és egy termisztorból. Megtalálható benne egy A/D konverter, ami segítségével a mért értékek átalakíthatóak.



13. ábra - DHT11 érzékelő<sup>13</sup>

Működési feszültség	3V-5.5V
Mérési tartomány (páratartalom)	20-90%
Mérési tartomány (hőmérséklet)	0-50°C
Pontosság (páratartalom)	±5%
Pontosság (hőmérséklet)	±2°C
Áramfelvétel	1.5mA

7. táblázat - DHT11 paraméterei

#### 2.6.7. Vonalkövető érzékelő

A vonalkövető szenzor (14. ábra) működése egy infravörös fény segítségével valósul meg, pontosabban, hogy a vizsgált felület elnyeli vagy visszaveri a LED által kibocsájtott infravörös fényt.



14. ábra - HW-511 érzékelő<sup>14</sup>

<sup>13</sup> Forrás: <https://c1.neweggimages.com/productimage/nb640/AXF6D2202080NUOZ024.jpg>

<sup>14</sup> Forrás: <https://shop.tavir.hu/wp-content/uploads/robot-vonalszenzor-1ch-digit-90fok-2.jpg>

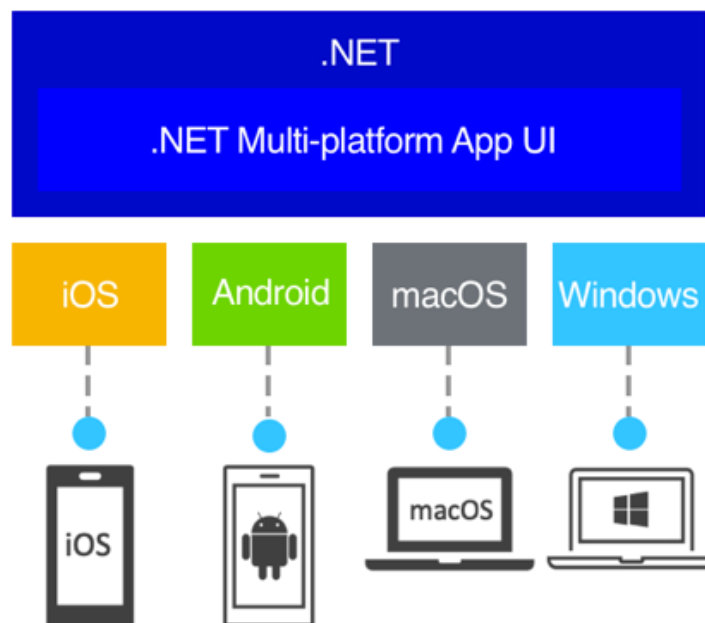
Működési feszültség	3.3V-5.5V
Áramfelvétel	20mA
Működési hőmérsékleti tartomány	-25°C÷75°C
Érzékelési távolság	10mm-12mm

8. táblázat - HW-511 paraméterei

## 2.7. Asztali alkalmazás

A rendszer felhasználói felülete egy .NET MAUI<sup>15</sup> desktop alkalmazás. A .NET MAUI (15. ábra) egy cross-platform keretrendszer, ami használatával natív mobil és asztali alkalmazások fejlesztésére van lehetőség. Ezáltal az alkalmazás futtat Android, iOS, macOS, Windows rendszereken is egyetlen közös kódbázisból. A kód megírása C# és XAML nyelvek használatával történik. [11]

A XAML<sup>16</sup> egy deklaratív leíró nyelv, ami azért lett létrehozva, hogy a .NET keretrendszer használata közben megkönnyítse a grafikus felhasználói felületek fejlesztését.



15. ábra - .NET MAUI<sup>17</sup>

<sup>15</sup> MAUI: Multi-platform App UI

<sup>16</sup>XAML: Extensible Application Markup Language

<sup>17</sup> Forrás: <https://learn.microsoft.com/en-us/dotnet/maui/media/what-is-maui/maui-overview.png>



### 3. A rendszer specifikációi és architektúrája

A rendszer két fő komponensből épül fel:

1. fizikai rendszer: az Arduino áramköri lap és az érzékelők összekapcsolása
2. asztali alkalmazás: az elkészített felhasználói felület

A rendszer leegyszerűsített vázlata az alábbi ábrán (16. ábra) látható:

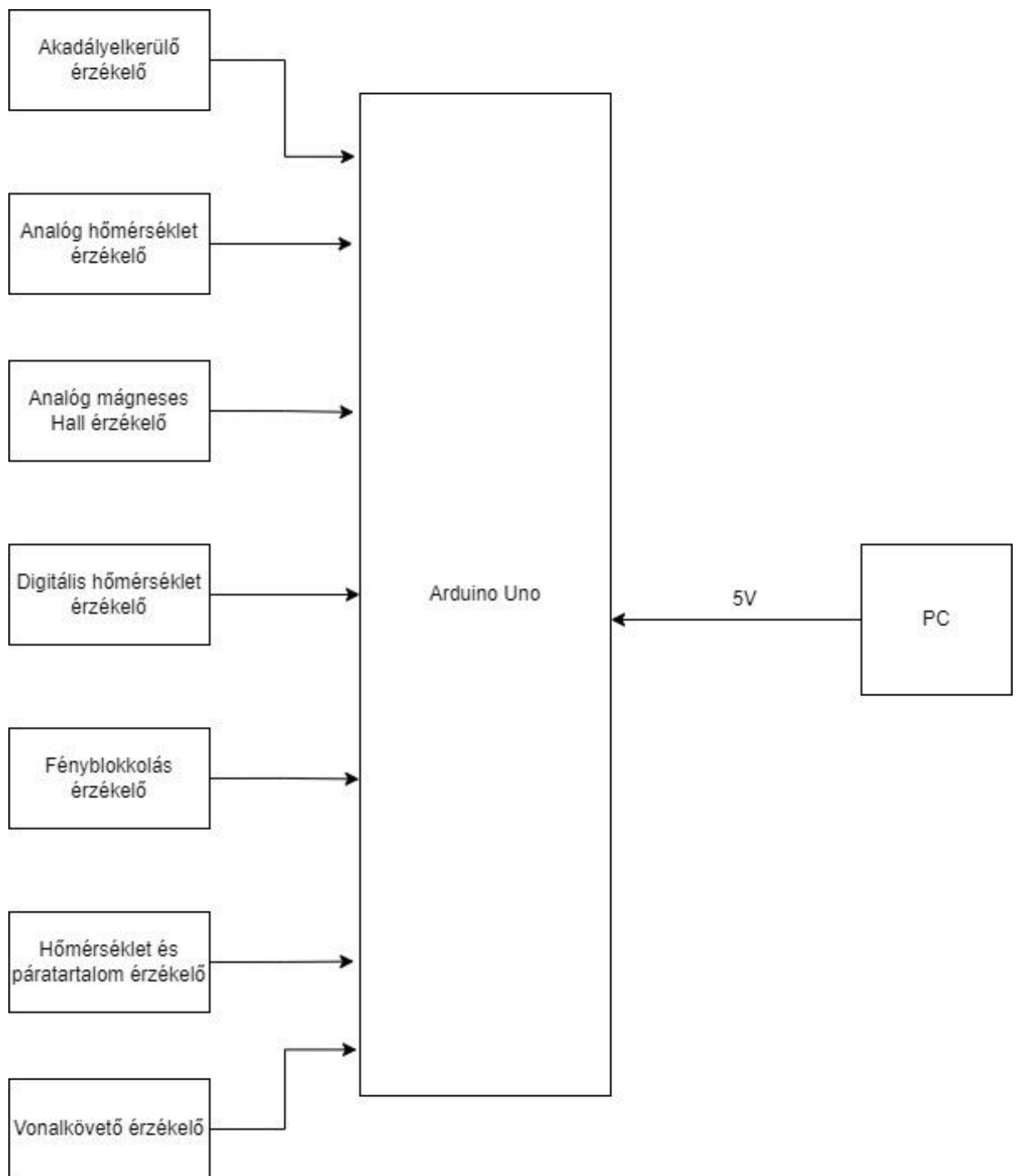


*16. ábra – A rendszer vázlata*

A rendszer hardver része, vagyis a fizikai rendszer áll egy Arduino Uno áramköri lapból és a hozzá csatlakoztatható érzékelőkből, a rendszer jelenlegi állapotában azok a szenzorok csatlakoztathatóak, amik az Érzékelők fejezetben kerültek bemutatásra. A rendszer másik összetevője, az asztali alkalmazás, ahol a felhasználó dinamikusan tudja konfigurálni a szenzorokat és a kapott adatokat leolvasni.

Az áramköri lap egy USB csatlakozóval van bekötve így azon keresztül kapja meg a működéséhez szükséges 5V tápfeszültséget, valamint a szenzorokat is ez látja el a szükséges feszültséggel.

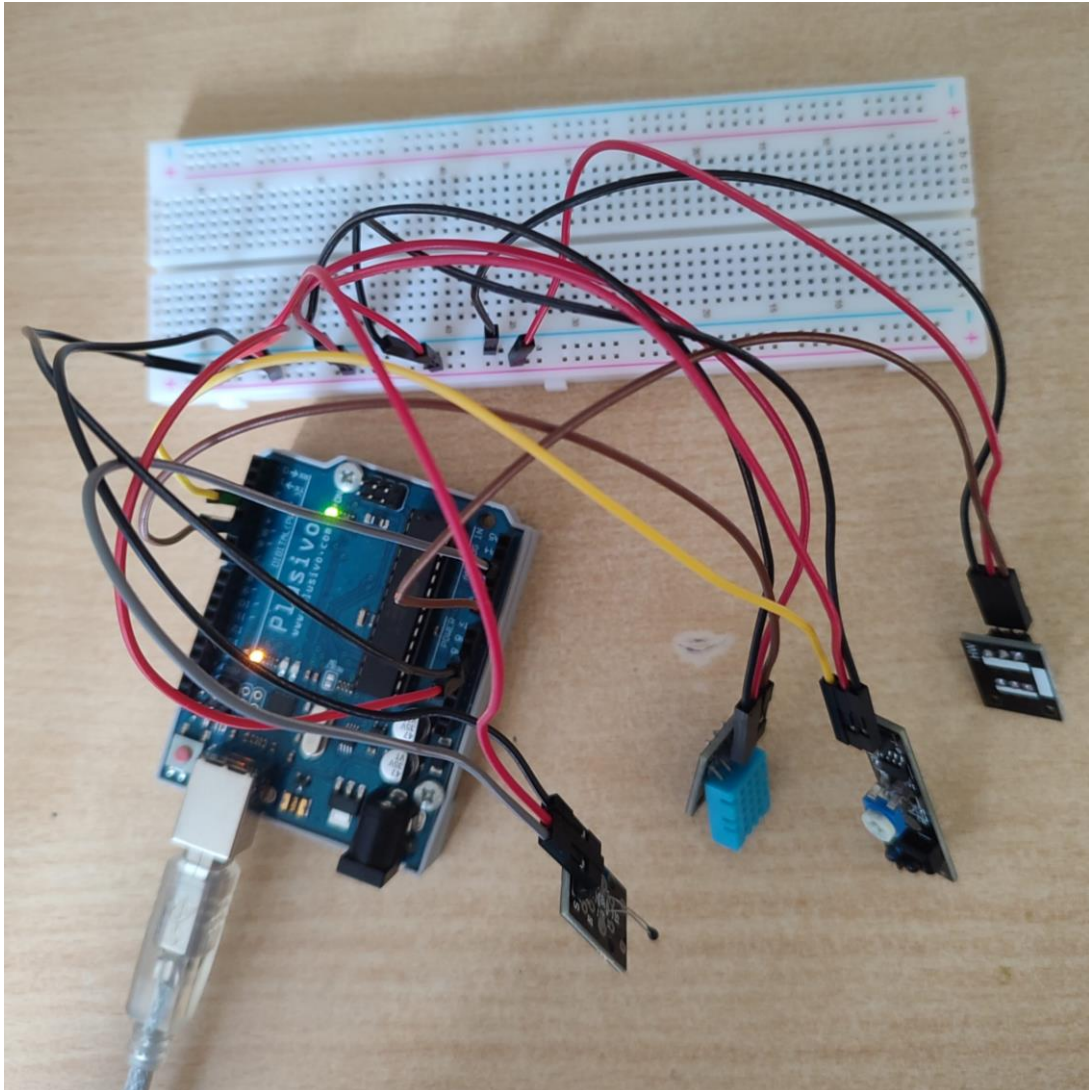
A 17. ábra mutatja be a rendszer tömbvázlatát, megjelenítve, a fizikai összetevőit a rendszernek.



17. ábra - Tömbvázlat

A 18. ábrán látható a rendszer, amikor az Arduino Uno áramköri laphoz csatlakoztatva van néhány szenzor, amelyek párhuzamosan vannak kötve. További érzékelőket lehet még hozzáadni vagy ki lehet cserélni az szerint, hogy éppen melyik szenzor adataira van szükség. A képen szereplő érzékelők mindegyike 5V-os feszültséggel üzemel, a breadboard segítségével kötöttem minden szenzort rá erre a kimenetre, valamint a GND-re, hiszen az Arduinonak csak

egy 5V feszültségű kivezetése van és két GND, így az érzékelők párhuzamos bekötésére van szükség, hogy mindegyik megkapja a működéséhez szükséges feszültséget. Az érzékelők bekötésénél nincs szükség külön ellenállások bekötésére, mert az általam használt érzékelők már kompakt modulok, amelyek tartalmazzák a szükséges ellenállást.



*18. ábra - Arduinohoz kötött érzékelők*

### **3.1. Követelmény specifikáció**

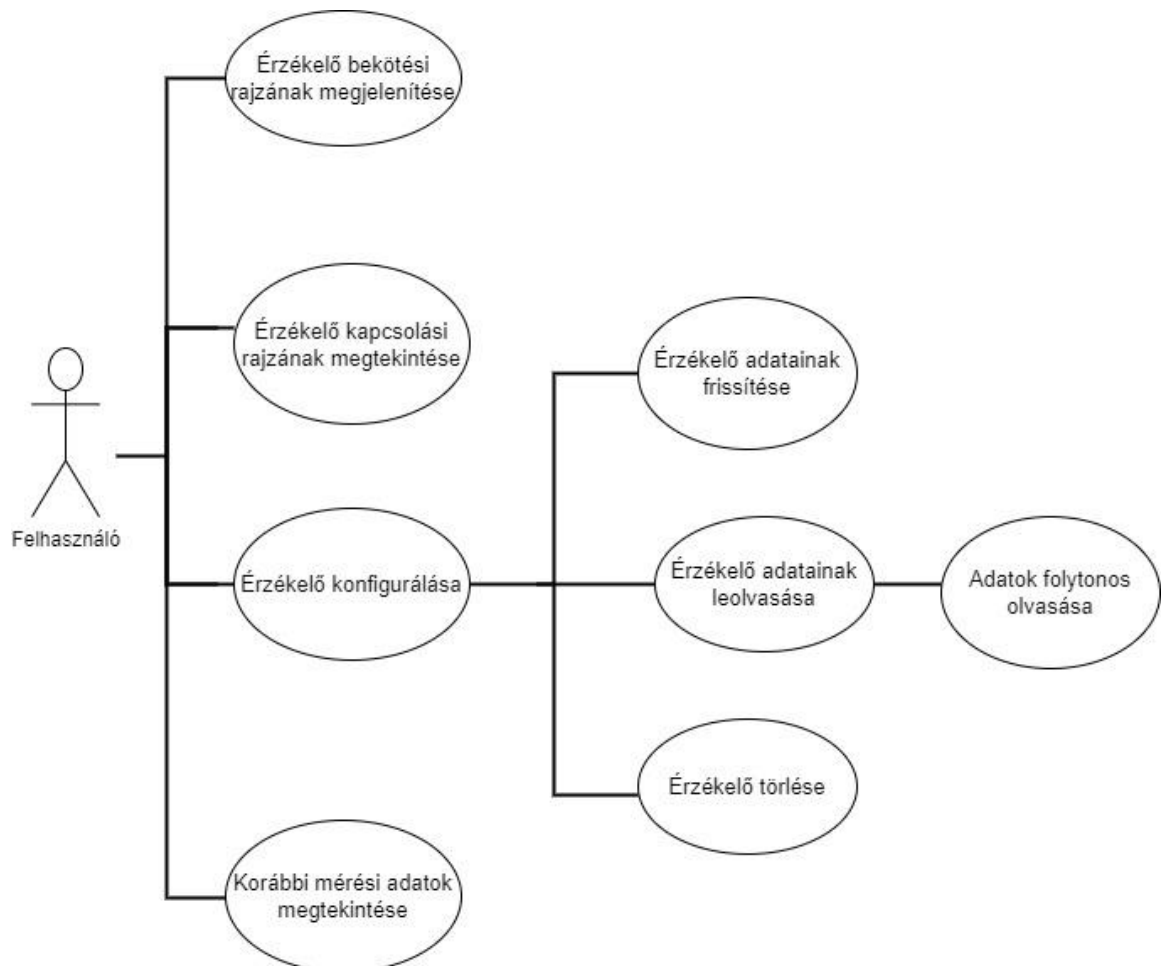
Egy szoftver megtervezésekor gyakran nehéz pontosan megfogalmazni, hogy hogyan épüljön fel a rendszer, miképp működjön a rendszer, hogyan nézzen ki a felhasználói felület. Ezek megtervezéséhez nyújtanak segítséget a követelmények, amik alapján aztán felépíthető a rendszer, segítségükkel meghatározhatóak azok a szolgáltatások, megszorítások amelyek szükségesek a rendszer működéséhez. A követelmény maga egy eléggé tág fogalom az iparban,

ezért általában két nagy részre osztva jelenik meg: felhasználói követelmények és rendszerkövetelmények, a továbbiakban ezek kerülnek bemutatásra. [12]

### 3.1.1. Felhasználói követelmények

A felhasználói követelményekben kerül megfogalmazásra az, hogy pontosan milyen szolgáltatások várhatóak el a rendszertől. A felhasználói követelmények elsősorban a felhasználóknak, ügyfeleknek íródnak, ezért nem tartalmaznak részletes technikai paramétereket a rendszerről, ezért tartalmaz diagramokat, természetes nyelven megírt kijelentéseket. [12]

A 19. ábra mutatja be az érzékelőrendszer használat-eset diagramját, ami egy online szerkesztő<sup>18</sup> segítségével készült. A diagramon megjelenik a felhasználó és a számára elérhető szolgáltatások.



19. ábra - Alkalmazás Használat-Eset diagramja

<sup>18</sup> <https://app.diagrams.net/>

A továbbiakban részletesebb bemutatásra kerülnek a felhasználói követelmények:

- Érzékelő bekötési rajzának megjelenítése: a rendszer elindulása után az érzékelők listájából kiválasztható egy érzékelő és akkor megjelenik az adott szenzor Arduino laphoz való csatlakoztatását bemutató kép
- Érzékelő adatlapjának/kapcsolási rajzának megtekintése: kiválasztva egy adott szenzort látható annak a belső kapcsolási rajza, megnyitható egy PDF dokumentum, ami az érzékelő adatlapját tartalmazza
- Érzékelő konfigurálása: kiválasztható, hogy az adott számú analóg vagy digitális ki/bemenetet milyen módba konfigurálja
- Érzékelő adatainak frissítése: újra lekérheti az adott szenzor által mért pillanatnyi értéket
- Érzékelő adatainak leolvasása: megjelenik az érzékelő által küldött adat a képernyőn
- Érzékelő törlése: ha egy érzékelő adatait már a felhasználó nem szeretné tovább figyelemmel kísérni, akkor az eltávolítható
- Érzékelő adatainak folytonos olvasása: az adatok egy grafikonon jelennek meg, ami folyamatosan frissül
- Korábbi mérések adatainak megtekintése: kiválasztva egy szenzort látható a rendszerben korábban végzett mérések eredményei

### **3.1.2. Rendszerkövetelmények**

A felhasználói követelményekkel ellentétben a rendszerkövetelmények részletesen bemutatják a rendszer funkcióit, szolgáltatásait. A funkciók alapján két csoportba sorolhatjuk ezeket a követelményeket: funkcionális és nem-funkcionális követelmények. [12]

#### **3.1.2.1 Funkcionális követelmények**

A funkcionális követelmények részletesen bemutatják, hogy a rendszer milyen választ adjon az egyes bemenetekre.

A felhasználói felület indítása után a felhasználónak lehetősége van egy legördülő listából kiválasztani egy szenzort, a kiválasztás után megjelenik a képernyőn az adott érzékelő bekötési rajza, ami segítséget nyújt a felhasználónak ahhoz, hogy azt hogyan kötheti be a rendszerbe. A Next gombra kattintva a következő oldalra jut a felhasználó, ahol az érzékelők konfigurálása történik.

A konfigurációs oldal betöltésekor a felhasználó az első legördülő listából tudja kiválasztani, hogy analóg vagy digitális be/kimenet szeretne konfigurálni. A másodikként

megjelenő legördülő lista opciói függenek az előzőtől, ha ott nem történt meg a kiválasztás, akkor itt a felhasználó egy üres listát talál. Ezután kiválasztható a konfigurációs mód, majd a Configure gombra kattintva megjelenik egy üzenet a felhasználó számára: sikeres konfiguráció esetén a „Successful configuration” üzenetet kiírva, sikertelen esetén pedig a „Configuration failed” üzenet olvasható. A konfigurálás után, kiválasztható az érzékelő majd a Read Data gomb megnyomásával a felhasználó átnavigál a következő oldalra.

A Sensors Data oldalon megjelenik a korábban kiválasztott szenzor által küldött adat, valamint három gomb. Az Update gomb megnyomásával a felhasználó frissíti az érzékelő által mért mennyiséget. A Remove gomb megnyomásával a felhasználó törli az érzékelőt és ahhoz, hogy ismét láthassa annak adatait újra kell azt konfigurálnia. A harmadik gomb, a Read Continuously kiválasztásával a felhasználó egy új oldalra kerül, ahol grafikonon megjelenítve követhetőek a szenzor valós idejű adatai. A felhasználónak lehetősége van több szenzor hozzáadására, ehhez az szükséges, hogy a Configure oldalon ismét konfigurálja az érzékelőt. A Sensor Data oldalon a több különböző szenzor által küldött adat egymás alatt, listászerű elrendezésben jelenik meg.

A megjelenő grafikon alatt megtalálható még két gomb, az első a Stop, amely megnyomásával a folyamatos frissítése az adatoknak leáll, ekkor megjelenik a Continue gomb, ha a felhasználó erre kattint, akkor folytatódik a szenzor adatainak folyamatos megjelenítése. A másik képernyőn megtalálható gomb a Back, ami megnyomásával a felhasználó visszajut az előző oldalra.

A Schematic oldalra lépve a felhasználó ismét egy legördülő listát lát, ahol miután egy érzékelő kiválasztásra került megjelenik két kép a képernyőn, az első a fizikai érzékelőt ábrázolja, a második pedig a szenzor kapcsolási rajzát. Itt még a felhasználónak lehetősége van az érzékelő adatlapjának a megnyitására, a Datasheet felirat mellett található Click here felírra kattintva megnyílik a szenzor adatlapja.

A Track Data lehetőséget kiválasztva a menüből a megjelenő oldalon szintén látható egy legördülő lista, ahol kiválasztva egy érzékelőt egy lista formájában megjelennek a korábbi mérések eredményei és a mérés pontos dátuma. A lista a dátum szerint rendezett, a legelső elem a legfrissebb érzékelő által küldött adat. A képernyőn még megtalálható az Export gomb, a gomb megnyomásával az adatok lementhetőek lokálisan a számítógépre, a fájl neve a mentés időpontja és a szenzor neve, helye a kódban megadott mappa.

A képernyő bal oldalán található egy menü, ami megkönnyíti a felhasználó számára az oldalak közti navigációt. Ha a menüből a Sensor Data oldalra navigál, még azelőtt, hogy akár egy érzékelőt is sikeresen konfigurált volna a rendszerbe, akkor ezt az oldalon megjelenő üzenet jelzi a felhasználónak.

### **3.1.2.2 Nem-funkcionális követelmények**

A rendszer használatához szükség van egy laptopra vagy asztali számítógépre, aminek az USB portjához csatlakoztatható az Arduino áramköri lap.

A felhasználói felület elkészítése a .NET 7 verziójú keretrendszerben valósult meg, ezért a rendszer ez vagy annál frissebb verziójú .NET-t kell tartalmazzon, a MAUI alkalmazás futtatásához Windows 10 vagy Windows 11 operációs rendszerre van szükség. Szükség van még az érzékelők adatlapjaink a megnyitásához internet hozzáférésre, mivel ezek nem lokálisan tárolt dokumentumok.

## 4. Részletes tervezés és gyakorlati megvalósítás

### 4.1. Arduino kód

A megvalósítás első lépése az Arduino kód megírása volt, a cél egy olyan protokoll megírása, amely által az érzékelőket dinamikusan lehet konfigurálni és az adatokat egységes adatformában továbbítani. A protokoll egy szabvány, szabályok halmaza ami meghatározza, hogy a kommunikáció, adatok küldése hogyan valósul meg a hálózaton belül.

A megírt protokollban az elküldhető parancsok formátuma jól meghatározott, ahogyan az ezekre kapott válasz is egységesített, a parancsok neve beszédes, magába foglalja, hogy milyen feladatot végez.

Amikor egy üzenet érkezik az első annak a részekre bontása, a protokoll szerint az üzenet első paramétere mindig a parancs neve, a második a ki/bemenet száma, ez után már parancstól függően változik. Ha egy olyan üzenet érkezik, amelyben a küldött parancsnév ismeretlen egy hibaüzenet érkezik: **ERR: parancsnév + unknown command** formában.

A konfigurációs parancs esetén az elvárt parancs a **parancsnév + Arduino kivezetés száma + kivezetés üzemmódja**. Ha az üzenet nem ebben a formában érkezik, vagy az nem teljes, akkor a kapott üzenet egy hibaüzenet. A hibaüzenetek egységes formája: **ERR: hiba oka**, ebben az üzenetben olvasható, hogy pontosan mi nem volt a protokoll előírásainak megfelelő a küldött üzenetben. A helyes konfigurációs üzenet a elküldése után kapott válasz **OK: parancsnév + Arduino kivezetés száma + kivezetés üzemmódja**. Helyes formátumú parancs elküldésekor meghívódik egy a parancsnévvel megegyező nevű függvény, ami végrehajtja magának a kivezetésnek a megfelelő módra való beállítást. Az elfogadott üzemmódok az input, output, input\_pullup, input módban az Arduino adott lába bemenetként viselkedik, output módban kimenetként és input\_pullup üzemmódban pedig szintén bemenetként, annyi különbséggel, hogy a láb felhúzó ellenállása magas lesz. A konfiguráló parancs neve: **configSensor**.

A szenzor által küldött adatok beolvasására két lehetőség van: az érzékelő adott időpillanatban mért értéke vagy az adatok folyamatos olvasása, amíg az nincs leállítva. Az első opciót választva és elküldve az ennek megfelelő parancsot, válaszként erre egyetlen üzenet érkezik az Arduinótól, ami tartalmazza a szenzor által mért adatot. A helyes olvasási parancs formája: **parancsnév + Arduino kivezetés száma + érzékelő neve**. Az Arduino kivezetés számánál azt a digitális ki/bemenetet kell megadni, amihez az érzékelő be van kötve valamint



előzőleg konfigurálva volt a megfelelő üzemmódba. A megfelelő és teljes formában küldött parancs esetén a válasz az a szenzor által abban az időpontban mért pillanatnyi érték. A helytelen parancs küldése esetén szintén hibaüzenet érkezik **ERR: hibaüzenet** formájában. Hibaüzenet érkezik, ha a küldött parancs nem tartalmazott minden szükséges részt, esetleg olyan szenzor név lett elküldve, ami nem a rendszer része. A rendszerben használható érzékelők nevei egy map típusú tárolóban vannak, amelyben a kulcs az érzékelő neve és hozzá tartozik a szenzor működéséhez szükséges függvény. Az adatok egyszeri olvasáshoz szükséges parancs neve: **readSensor**.

A folyamatos olvasás elindításához szükséges parancs hasonló felépítésű, mint ami az előbb említett egyszeri olvasást illeti: **parancsnév + Arduino kivezetés száma + érzékelő neve**. Itt a kiválasztott érzékelőtől folyamatosan érkezik az adat, egészen addig amíg egy megállító üzenet nem lesz küldve. Helyes parancs írásakor folyamatosan érkezik a mért adat az érzékelőtől, ezek értékek olvashatóak a Serial monitoron is. Hibás parancs esetében itt is hibaüzenet várható, ha nincs minden paraméter megadva vagy az adott szenzor nem található. A folyamatos olvasás leállítása után érkezik egy megerősítő üzenet, de ezt már a leállító parancs küldi. A folyamatos olvasásért felelős parancs neve: **readSensorContinuously**.

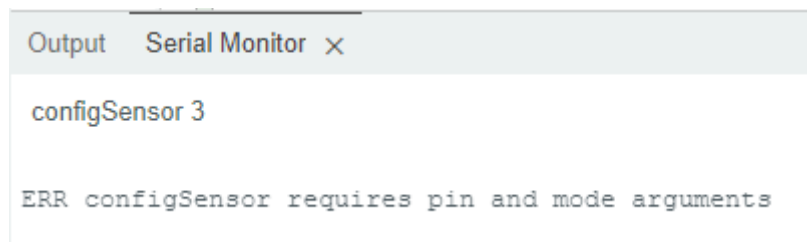
A folyamatos olvasás megállításához és ahhoz, hogy más parancsot is legyen lehetőség küldeni az Arduino fele, a **stop** parancs elküldésére van szükség, ennek elküldése után egy **OK: reading stopped** üzenet várható, ez után újra bármilyen parancs küldhető.

#### 4.1.1. Protokoll működésének bemutatása

Konfigurációs parancs küldésének módja és az arra kapott választ mutatja be az alábbi (20. ábra) ábra. A képen látható, hogy ahhoz, hogy az áramköri lap 3-as számmal jelzett digitális kivezetése bemenetként működjön, a “configSensor 3 input” parancs elküldésére van szükség és alatta olvasható az Arduino által küldött válasz, ami jelzi, hogy sikeres volt a konfiguráció. A következő ábra (21. ábra) szemlélteti egy nem teljes parancs elküldése esetén kapott választ, az elküldött parancsból hiányzik, hogy milyen módba kellene a kivezetést állítani, emiatt a kapott válasz egy hibaüzenet, így nem hajtódik végre a konfiguráció.

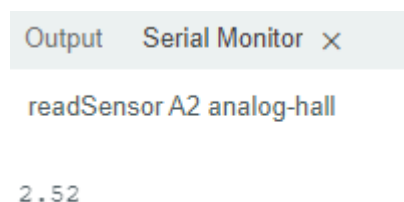


20. ábra - Digitális bemenet konfigurálása



21. ábra - Hibás konfigurációs parancs küldése

Az érzékelő által mért adat olvasásának módját mutatja be a 22. ábra. A megfelelő formában elküldött olvasási parancsra a kapott válasz a Hall érzékelő valós idejű mért értéke. A küldött parancsban a parancsnév után a bemenet száma van, amire az érzékelő van kötve és az érzékelő neve a küldött üzenet harmadik tagja. Minden érzékelőhöz tartozik egy függvény, ennek a függvénynek a meghívása történik az érzékelők nevei alapján, ezek a nevek és függvények láthatóak a 23. ábrán. A 24. ábra és 25. ábra egy-egy helytelen parancs elküldését mutatja meg, mindkét az Arduino által küldött üzenet egy hibaüzenet, ami tartalmazza, hogy miért nem volt sikeres az érzékelő adatának az olvasása. Az első esetben hiányzott a szenzor megnevezése, a második esetben olyan szenzor név lett megadva, ami nem található meg a rendszerben. Az érzékelőkhöz tartozó Arduino kódok megírása nem az én feladatom volt, mivel számos könyvtár elérhető a platformon, amelyek az egyes érzékelőkhöz tartoznak, így az én dolgom az volt, hogy ezeket integráljam a rendszerbe.



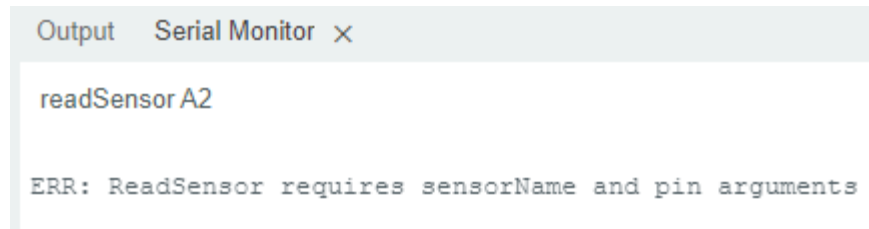
22. ábra - Olvasási parancs küldése

```

15 sensorNamesMap sensorNames = {
16   { "humidity-temp", &HW_507 },
17   { "analog-temp", &HW_498 },
18   { "analog-hall", &HW_495 },
19   { "line-tracker", &HW_511},
20   { "light-block", &HW_487},
21   {"digital-temp", &HW_506},
22   {"obstacle-avoid", &HW_488}
23 };

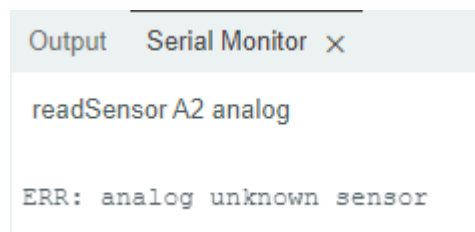
```

23. ábra – Érzékelők nevei és a hozzá tartozó függvények map tárolója



The screenshot shows the 'Serial Monitor' window in the Arduino IDE. The title bar includes 'Output' and 'Serial Monitor' with a close button. The text area contains the command 'readSensor A2' followed by an error message: 'ERR: ReadSensor requires sensorName and pin arguments'.

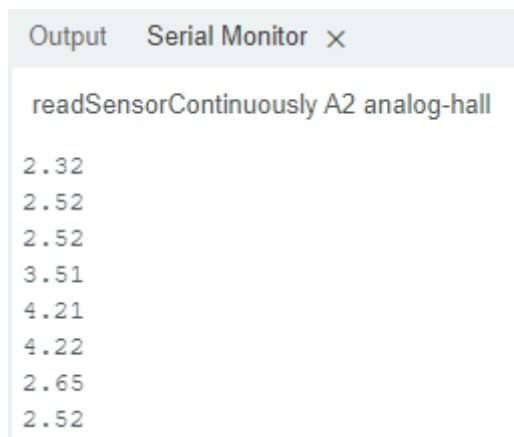
24. ábra - hibás olvasási parancs küldése



The screenshot shows the 'Serial Monitor' window in the Arduino IDE. The title bar includes 'Output' and 'Serial Monitor' with a close button. The text area contains the command 'readSensor A2 analog' followed by an error message: 'ERR: analog unknown sensor'.

25. ábra - Hibás olvasási parancs küldése

A folyamatos olvasás elindítása látható a 26. ábrán, az elküldött parancs felépítése megegyezik az egyszeri olvasási parancsával, csak a parancs nevek különböznek egymástól. Az üzenet elküldése után folyamatosan olvasható a Serial Monitoron az érzékelő által mért adat. Az olvasás leállítását szemlélteti a 27. ábra, ahol a parancs elküldése után már csak egy az olvasás leállítását jelző üzenet érkezik.



```
Output  Serial Monitor  X
readSensorContinuously A2 analog-hall
2.32
2.52
2.52
3.51
4.21
4.22
2.65
2.52
```

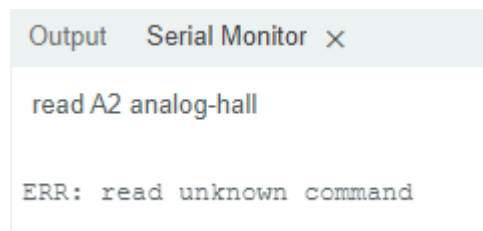
26. ábra - Folyamatos olvasási parancs küldése



```
Output  Serial Monitor  X
stop
...
4.22
2.65
2.52
2.52
2.52
2.52
2.52
OK: reading stopped
```

27. ábra - Folyamatos olvasás leállítása

Ha olyan parancs kerül elküldésre, amit nem tartalmaz a protokoll, akkor az arra kapott válasz egy hibaüzenet, ahogy a 28. ábrán is látható. A “read” kulcsszó nem része a protokollnak, így ezt a parancsot az Arduino nem tudja végrehajtani.



```
Output  Serial Monitor  X
read A2 analog-hall
ERR: read unknown command
```

28. ábra - Ismeretlen parancs küldése

Az alábbi ábrán (29. ábra) látható a soros kommunikáció elindítása, a setup() függvényben valósul meg az adatátviteli sebesség beállítása, ezzel elindítva a kommunikációt. Ez a függvény, csak egyszer fut le, utána következik a loop() függvény, ami egy végtelen ciklus

igazából, addig fut amíg az Arduino feszültség alatt van. Ezután folyamatosan kész arra az Arduino, hogy fogadja az üzenetet és amikor kap egy parancsot, akkor kezdetét veszi annak a feldolgozása. A `Serial.available()` függvény visszatéríti a soros porton olvasásra elérhető bájtok számát, a maximum megadható üzenet hossza 64 bájt.

```
1  #include "functions.h"
2
3  myFunctions f;
4  void setup() {
5      Serial.begin(9600);
6  }
7
8  void loop() {
9      if (Serial.available() > 0) {
10         String strLine = f.ReadLine();
11         if (strLine != "")
12             f.ParseLine(strLine);
13     }
14 }
```

29. ábra – Soros kommunikáció elindítása

## 4.2. Asztali alkalmazás

A felhasználói felület létrehozásakor az első lépés a kapcsolat létrehozásának megírása volt az Arduino és az alkalmazás között. Ez megoldható egy függvény megírásával (30. ábra), mert számos könyvtár megtalálható, ami segítségével megvalósul a soros kommunikáció. A függvény meghívásakor paraméterként a portot kell megadni, amin keresztül megvalósul a soros kommunikáció, ez esetben a COM3 és az átviteli sebességet, ami 9600. Fontos, hogy az itt megadott érték ugyanaz legyen, ami az Arduino kódban is szerepel, ellenkező esetben a kommunikáció nem lesz sikeres, hibák léphetnek fel, adatvesztéssel jár. Az Arduino alapértelmezetten a soros kommunikációra ezt a portot használja, van lehetőség ennek megváltoztatására, a Windows Eszközkezelőjét használva.

```

8      private SerialPort serialPort;
          0 references
9      public ConnectArduino(string portName, int baudRate)
10     {
11         serialPort = new SerialPort(portName, baudRate);
12         serialPort.Open();
13     }

```

30. ábra - Függvény a kommunikációs kapcsolat létrehozására

Az előbb említett könyvtárak közül az általam használt a System.IO.Ports, ezek a .NET keretrendszerhez tartozó könyvtárak elérhetőek a Visual Studio NuGet csomagkezelőjében. Ezen a felületen egyszerűen hozzáadható az alkalmazáshoz egy könyvtár, annak a nevére kell rákeresni és utána már telepíthető. Telepítés után a forráskódban fejlécében kell hivatkozni ezekre a csomagokra, hogy annak elemeit használni lehessen. Ez a hivatkozás mindig a kód elején kell megjelenjen, a könyvtár nevét a using kulcsszó előzi meg (31. ábra). Egy olyan fájl adattagjainak, függvényeinek használatakor ami ugyanabban a projektben van, de más mappában található, akkor arra hivatkozni szintén így lehet.

```

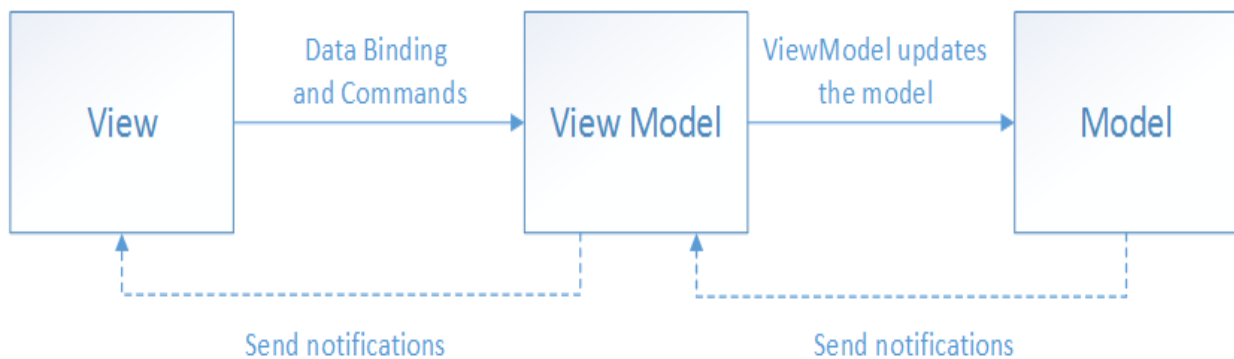
1  using CommunityToolkit.Mvvm.ComponentModel;
2  using CommunityToolkit.Mvvm.Input;
3  using System.Collections.ObjectModel;
4  using ui.Model;
5  using ui.View;

```

31. ábra - Hivatkozás könyvtár csomagokra

Az alkalmazás fejlesztése során az MVVM<sup>19</sup> tervezési mintát követtem, ami három részből tevődik össze: modell, nézet, nézetmodell (32. ábra). A minta használatával jól elkülöníthető egymástól az alkalmazás logikai része és a megjelenített felhasználói felület. A nézetben találhatóak meg a XAML fájlok, amik tartalmazzák a képernyők megjelenését, dizájnját. A modellben vannak az adattípusok, amik az alkalmazáson belül vannak használva, a nézetmodell köti össze az előző két elemet, tartalmazza az adatokat és logikát, ami a nézetben megjelenik. [13]

<sup>19</sup> MVVM: Model-View-ViewModel



32. ábra - MVVM tervezési minta<sup>20</sup>

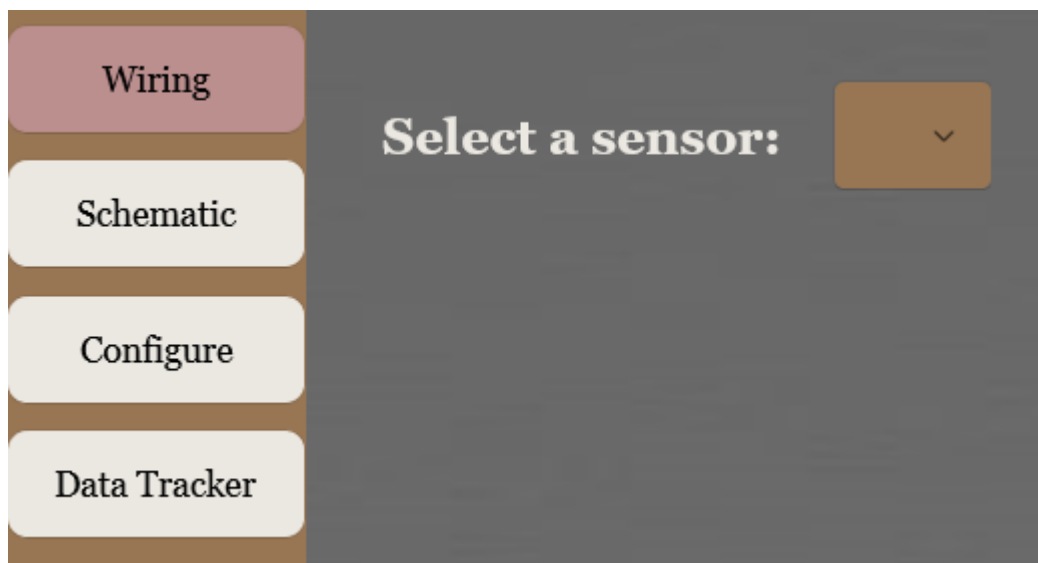
A három összetevő három külön mappában van elhelyezve, minden, az alkalmazás futása közben megjelenő oldalnak van egy külön View és ViewModel fájl létrehozva. A nézetben még a .xaml fájl mellett található egy .cs fájl is, de ennek szerepe csak az adatkötésben van, hogy a nézetmodell szerint frissüljön a nézet.

#### 4.2.1. Kezdőoldal

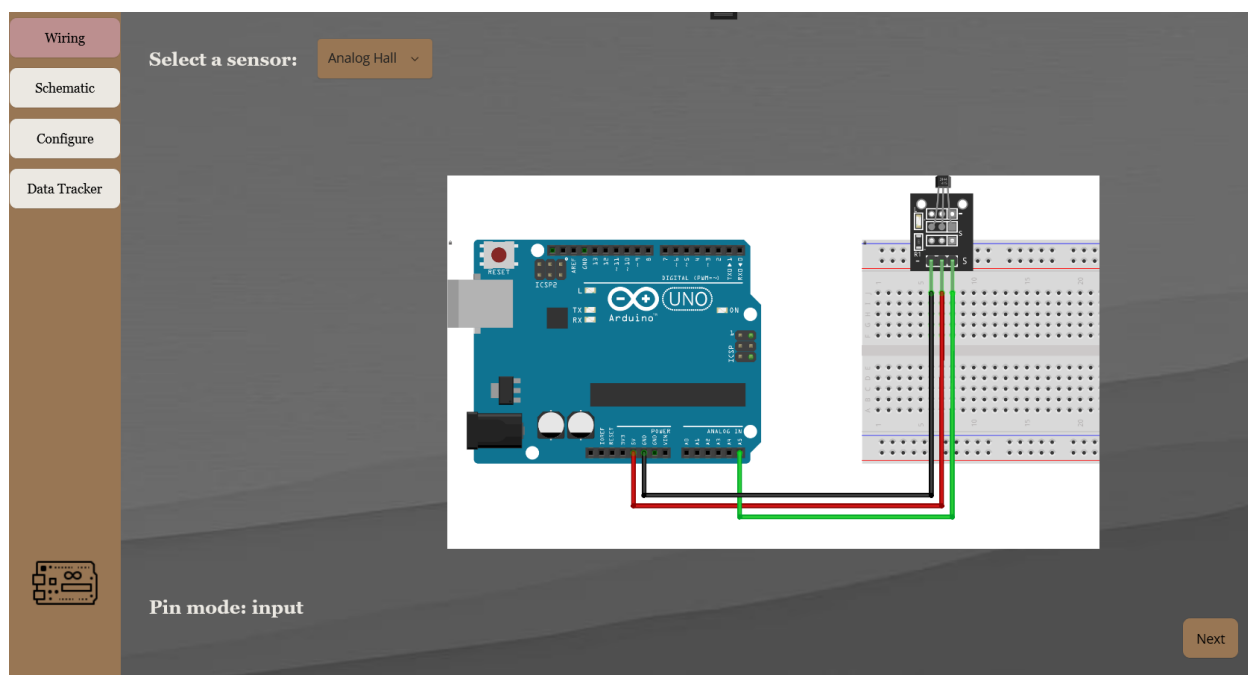
Az alkalmazás elindítása után a 33. ábrán látható oldal jelenik meg, bal oldalon a menü, ahol lehet navigálni az oldalak között, magán az oldalon pedig egy felirat jelzi, hogy hol választható ki az érzékelő a listából. A menüben látható, hogy melyik oldal van éppen megjelenítve képernyőn, mivel az aktuális oldal menüpontja mindig más színnel jelenik meg, mint a többi oldal neve. A listából való választás utána a 34. ábra szerinti kép lesz látható, ahol már megjelenik, a kiválasztott érzékelő bekötési rajzán kívül, a Next gomb is ami átnavigál a konfigurációs oldalra. Ahhoz, hogy a gomb és a felirat, ami jelzi, hogy milyen módba kell konfigurálni az Arduino kivezetését, csak azután legyen látható a xaml fájlban állítható be, a megjelenő elemek IsVisible tulajdonsága kezdetben false értékű, ami azt jelenti, hogy nem látható a képernyőn, majd ha megtörténik a kiválasztás ez az érték true lesz és így megjelennek a felületen. Az, hogy mikor lesz ez az érték igaz, annak változtatása a ViewModel-ben történik, egy bool típusú attribútum létrehozására van szükség, ami akkor true, ha a kiválasztott szenzor attribútuma nem null. Ez a bekötési rajz segítséget nyújt a felhasználónak, mutatja, hogy az érzékelő lábait pontosan hogyan kötheti össze az Arduino áramköri lappal. A + jellel jelölt kivezetés kerül mindig a tápfeszültségre, a – jellel jelölt láb az Arduino GND kivezetésére

<sup>20</sup> Forrás: <https://learn.microsoft.com/hu-hu/dotnet/architecture/maui/media/mvvm-pattern.png>

(földelési pont) kerül, a többi láb esetén az, hogy melyik kivezetésbe kerül azt a felhasználó dönti el és ennek a lábnak a konfigurálása történik meg a későbbiekben.



33. ábra – Alkalmazás kezdőoldala



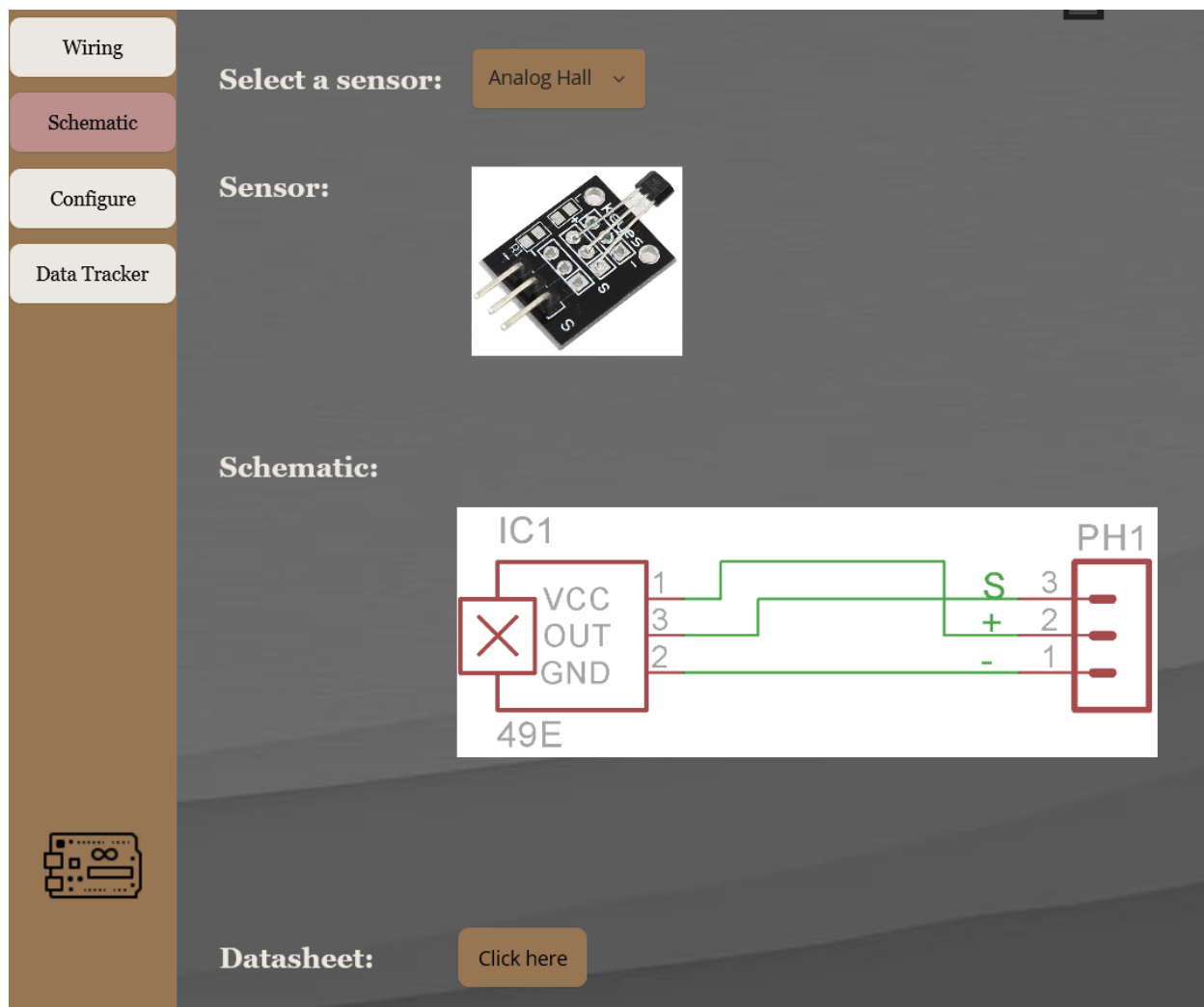
34. ábra - Kiválasztott bekötési rajz megjelenítése

#### 4.2.2. Érzékelő adatlapja oldal

A lista kiválasztása után megjelenik a kiválasztott érzékelő képe, hogy fizikailag hogyan néz ki az adott modul, alatta az érzékelő kapcsolási rajza, valamint a legalsó gomb



megnyomására megnyílik a böngészőben az érzékelő adatlapja, amiből minden paraméter megtudható a választott szenzorról.



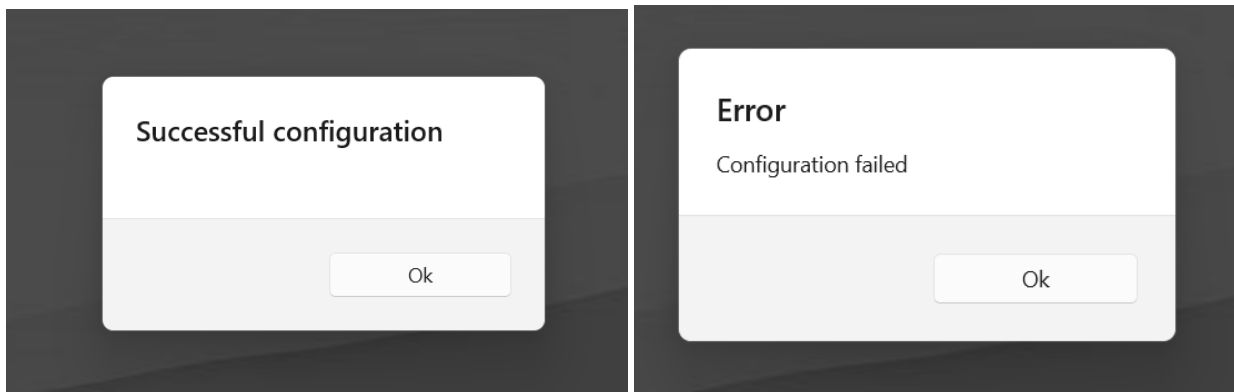
35. ábra - Érzékelő adatlap oldala

#### 4.2.3. Konfigurációs oldal

Az oldalra lépve a 36. ábrán látható elemek jelennek meg, ahol a kiválasztások után és a Configure gombot megnyomva egy üzenet jelenik meg, ami jelzi, hogy sikeres volt-e a konfiguráció vagy sem. Ez látható a 37. ábrán, a két üzenet közül az egyik jelenik meg a felhasználó számára, ha mindenhol nem volt kiválasztva egy opció, akkor a konfiguráció nem valósulhat meg.

The image shows a configuration interface on a dark grey background. It has two sections: 'Select pin:' and 'Select mode:'. Under 'Select pin:', there are two dropdown menus; the first is set to 'analog' and the second to 'A2'. Under 'Select mode:', there is one dropdown menu set to 'input'. Below these sections is a light grey button labeled 'Configure'.

36. ábra - Kivezetés konfigurálása



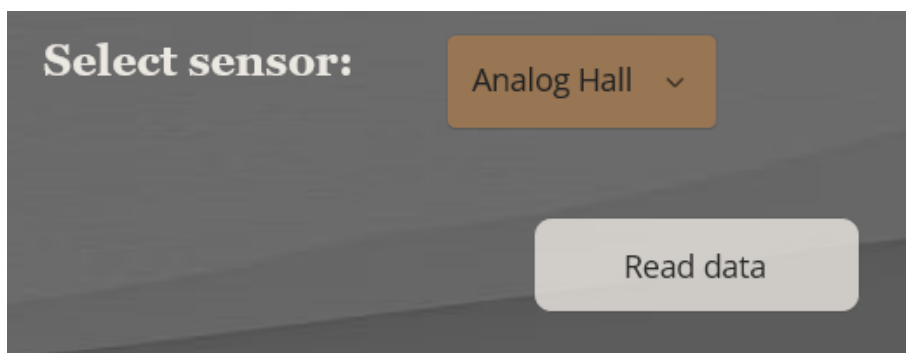
37. ábra - Konfigurálás után kapott visszajelzések

Ha a konfigurálás gomb megnyomásakor az áramköri lap nincs csatlakoztatva a számítógéphez, akkor egy hibaüzenet (39. ábra) jelzi ezt a felhasználónak. Az érzékelők konfigurálása, adatainak megjelenítése nem hajtható végre, ha nincs csatlakoztatva az Arduino. Csatlakoztatás nélkül megjeleníthetők a bekötési rajzok, a szenzorok adatlapjai valamint a korábbi mérések eredményei.

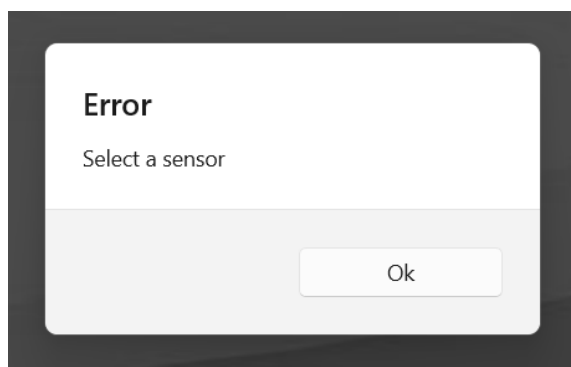


38. ábra - Nincs csatlakoztatva az Arduino áramköri lap

Ha sikeres volt a konfiguráció, akkor a képernyőn újabb elemek tűnnek fel (39. ábra), ahol ha a listából ki lett választva egy érzékelő a gomb megnyomása után a következő oldal jelenik már meg, ha viszont kiválasztás nélkül kattint a felhasználó a gombra akkor az előzőekhez hasonló üzenetdoboz ugrik fel, ami jelzi, hogy nem lett egyetlen érzékelő sem kiválasztva, ezt szemlélteti a 40. ábra.



*39. ábra - Érzékelő adatainak beolvasása*

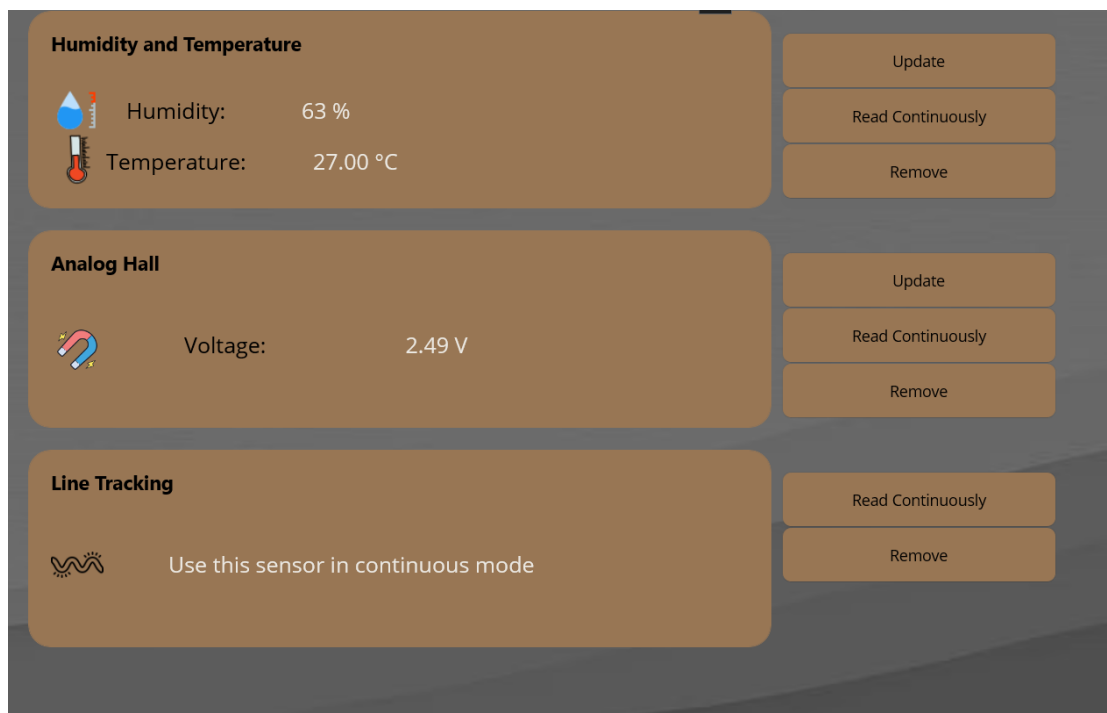


*40. ábra - Hibaüzenet, ha nem volt érzékelő kiválasztva*

#### **4.2.4. Adatok megjelenítése oldal**

Az alábbi ábra (41. ábra) szemlélteti azt az oldalt, ahol megjelennek az előzőleg konfigurált érzékelők adatai, valamint még minden érzékelő mellett három gomb, amikkel az adott érzékelővel kapcsolatos műveletek hajthatók végre. Az Update gombbal frissíthető az érzékelő pillanatnyi értéke, a következő segítségével elindítható a folytonos olvasása az adatoknak, a Remove gombbal pedig eltávolítható a rendszerből az érzékelő. Van olyan szenzor, ahol nem jelenik meg a mért pillanatnyi érték és ennél hiányzik a frissítés gomb is. Az ábrát nézve ez a vonalkövető szenzor esetében jelenik meg, mivel egy ilyen típusú érzékelő esetében egy egyszeri mérés nem mérvadó, ez folytonos módban hasznos.

Az oldalon még megjelennek kis színes ikonok, ezek kiválasztásánál próbáltam olyan szimbólumot választani, ami köthető az adott érzékelő típusához, például egy hőmérséklet érzékelő mellett egy hőmérő jelenik meg. Ezeket az ikonokat egy internetes oldalról<sup>21</sup> töltöttem le, ahogy nagyon sok elem közül lehet válogatni, ezek személyre szabhatóak, megváltoztatható a háttérszínük és ingyenesen letölthetőek. Ezek az elemek, a háttérkép a projekt Resources mappájában találhatóak meg.

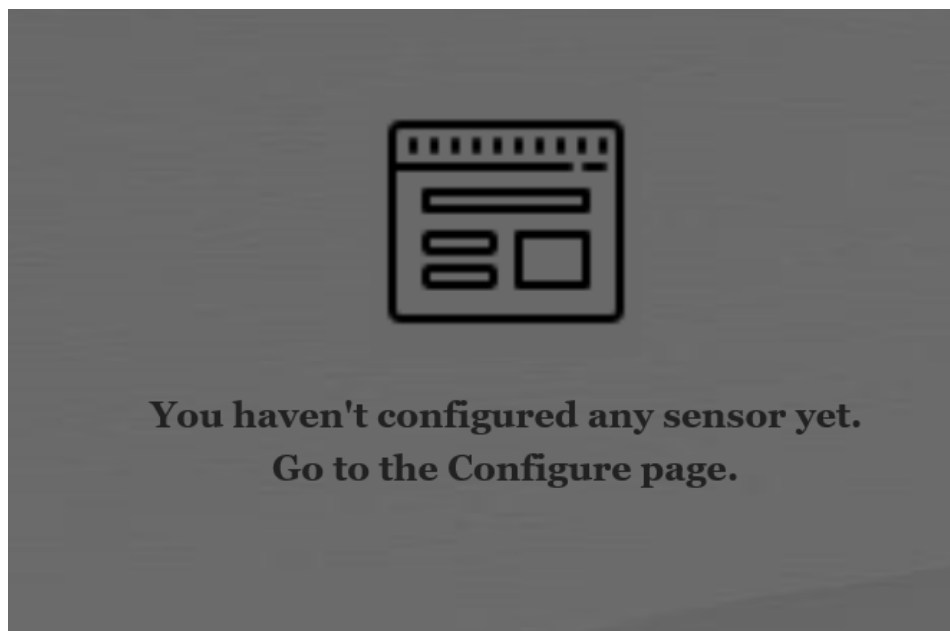


41. ábra – Szenzor adatok megjelenítése oldal

Ha minden érzékelő törölve lett vagy a felhasználó már azelőtt erre az oldalra lép, hogy konfigurált volna egyetlen érzékelőt is, akkor egy üzenet (42. ábra) jelzi számára, hogy ahhoz, hogy ezen az oldalon megjeleníthesse a szenzorok adatait vissza kell térnie előbb a konfigurációs oldalra.

---

<sup>21</sup> Forrás: <https://icons8.com/>



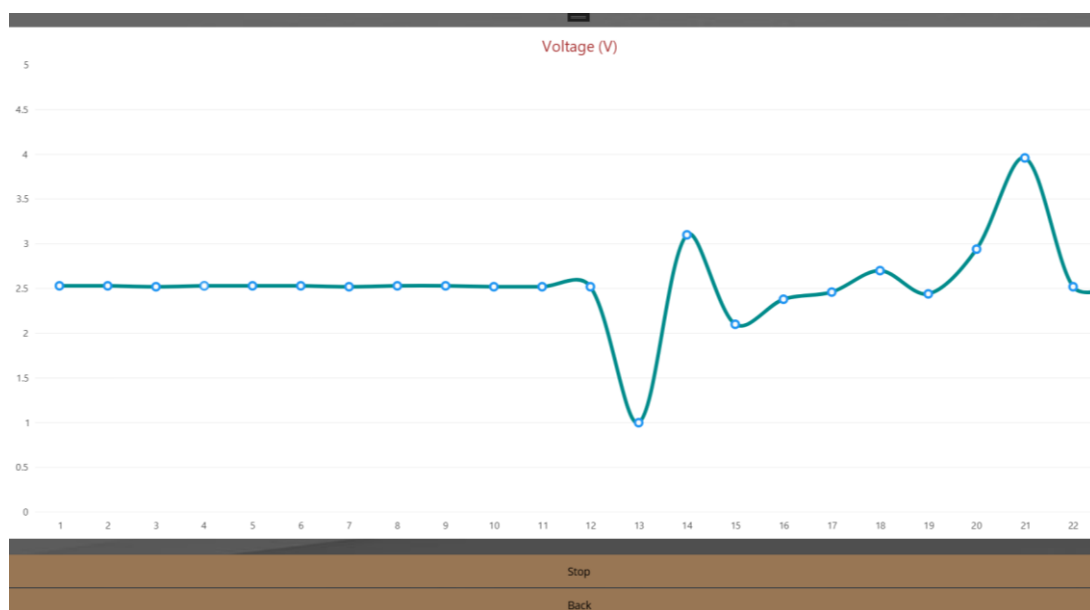
42. ábra - Üres oldal

#### 4.2.5. Folytonos olvasás oldal

A Read Continuously gomb megnyomása után megjelenő oldalon (43. ábra) látható egy grafikon, ami az érzékelő által mért adatokat jeleníti meg. Található még két gomb, a Stop megnyomása után leáll az adatok olvasása, megjelenik a Continue gomb (44. ábra), ami megnyomása után folytatódik az olvasás. A Back gombbal van lehetősége a felhasználónak visszatérni a előző oldalra. A grafikon x tengelyén mindig a vett minták száma jelenik meg, az Y tengelyen az értékek az adott szenzor által mérhető minimum és maximum értékek között mozognak.

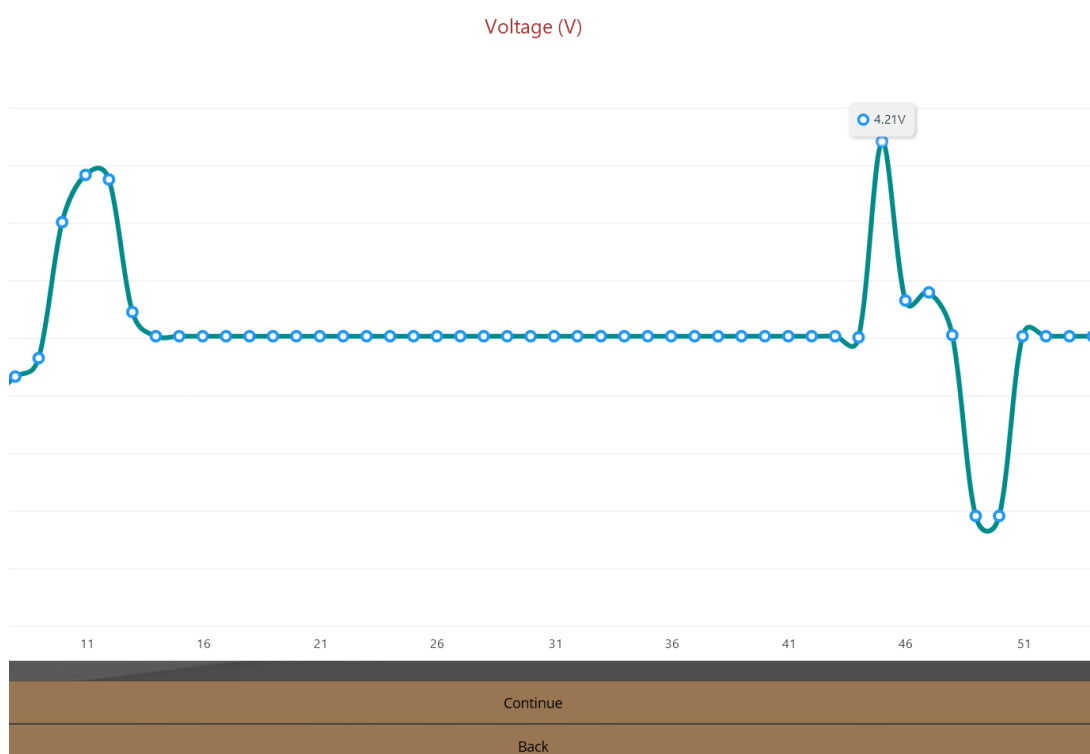
Visszatérve az előzőekben említett vonalvezető érzékelőhöz az csak 0 és 1 értéket küld, ezért a megjelenő grafikon mellett helyet kapott még egy jelmagyarázat is (45. ábra), ez a megoldás látható a hasonló szenzorok esetében is.

A grafikonok megjelenítése a LiveCharts2 könyvtár használatával valósult meg, ami segítségével megjeleníthetők interaktív grafikonok és diagramok.

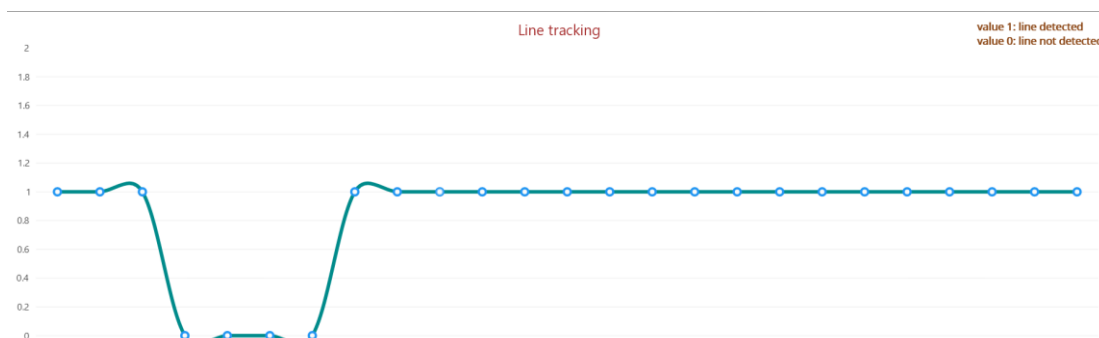


43. ábra - Folytonos olvasás megjelenítése

Kiválasztva egy adott adat pontot olvasható, hogy ott mennyi a pontos mért érték (44. ábra).



44. ábra - Folytonos olvasás folytatása



45. ábra – Jelmagyarázat

#### 4.2.6. Korábbi mérések oldal


A menü utolsó pontja a Track Data oldal (46. ábra - Mérési adatok listá), itt láthatóak a korábbi mérések eredményei szenzorokra lebontva. Látható a pontos időpont, hogy a mérés mikor valósult meg valamint, hogy az adott időpillanatban milyen értéket kaptunk. Minden mérési adat egy lokális adatbázisban mentésre kerül, kivéve azoknál a szenzoroknál, amelyek csak 0 és 1 értékeket küldenek. Ez az oldal hasznos, hogy össze lehessen hasonlítani különböző időpontokban kapott mérési adatokat.

Select a sensor:	Analog Hall	Export
6/24/2023 8:51:55 PM	2.66 V	
6/21/2023 2:47:56 AM	1.63 V	
6/21/2023 2:47:31 AM	1.56 V	
6/21/2023 2:46:33 AM	1.63 V	
6/21/2023 2:46:25 AM	1.63 V	
6/21/2023 2:46:19 AM	1.63 V	
6/21/2023 2:46:16 AM	1.63 V	
6/21/2023 2:46:16 AM	1.63 V	
6/21/2023 2:45:50 AM	1.63 V	

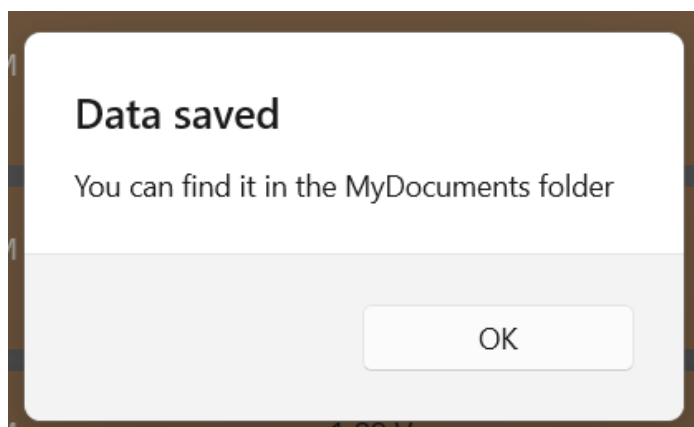
46. ábra - Mérési adatok listázása

A jobb oldalon található Export gomb megnyomásával az adatok lementhetők egy táblázatba, lokálisan a számítógépre, a mentett dokumentum a MyDocuments mappába kerül, ott létrejön egy fájl (47. ábra) aminek a neve az aktuális dátum és a kiválasztott érzékelő neve. Ha már egyszer mentettük az adatokat és létre lett hozva a fájl, a következő alkalommal ha ismét menteni szeretnénk az adatokat, akkor a már létező fájl fog frissülni és nem jön létre új. Abban az esetben ha ez a fájl törlésre került vagy egy más mappába lett helyezve, akkor a fájl újra létrejön, benne minden mérési adattal a kiválasztott érzékelőtől. Az adatok sikeres mentését egy, a képernyő közepén megjelenő, üzenet jelzi a felhasználó számára (48. ábra).

A	B
Date	Voltage(V)
6/25/2023 13:44	1.88
6/25/2023 13:44	2.72
6/25/2023 13:44	1.83
6/25/2023 13:44	1.74
6/25/2023 13:44	1.49
6/25/2023 13:44	2.10
6/25/2023 13:44	1.54
6/25/2023 13:44	1.49
6/25/2023 13:44	1.54
6/25/2023 13:44	2.16

 2023-06-25-analog-hall

47. ábra - Mentett mérési adatok



48. ábra - Sikeres mentés

Az eseménykezelés, például egy gomb megnyomása a ICommand interfész implementációjával valósul meg. Erre egy példa az oldalak közötti navigáció, a ViewModel



osztályban megíródik a `GoToConfigurePage` metódus, ami implementálja a `RelayCommand` osztályt, ami az  `ICommand` interfészhez tartozik. Utána ezt a függvényt a XAML fájlban a gomb elem `Command` adattagjához (50. ábra) kell kötni, így a gomb megnyomásakor meghívódik a függvény és végrehajtódik a navigáció. Az ábrán még látható, hogy hogyan történik meg a gomb háttérszínének, betűtípusának és színének is a beállítása.

```
53 [RelayCommand]
54 2 references
55 async void GoToConfigurePage()
56 {
57     await Shell.Current.GoToAsync(nameof(ConfigSensor));
58 }
```

49. ábra - Navigációs függvény

```
<Button Text="Configure" Command="{Binding GoToConfigurePageCommand}"
        BackgroundColor="#EBE8E2"
        TextColor="Black"
        FontFamily="Georgia"
        Margin="0, 0, 0, 10"/>
```

50. ábra - Gomb adattagjainak megadása

## 5. Következtetések

### 5.1. Megvalósítások

Végeredményben elmondható, hogy sikerült egy működő interaktív érzékelőrendszer kialakítása. A munkafolyamat közben többször is elakadtam, az elején apró figyelmetlenségek okozták a hibákat, amiket néha sok időbe telt megtalálnom, de minden ilyen hiba arra ösztönzött, hogy még több figyelmet fordítsak a feladatra.

Amit sikerült ténylegesen megvalósítani az a protokoll, amin keresztül dinamikusan, a program futása közben konfigurálhatók az érzékelők és egységes formában küldi el az adatokat a számítógép felé, minden a mikrovezérlőnek küldött parancsra érkezik egy válasz, aminek köszönhetően a felhasználó mindig tudja, hogy megtörtént-e a parancs végrehajtása. Hét érzékelő van jelenleg a rendszerbe integrálva, található köztük digitális és analóg érzékelő is. Ezeknek a szenzorok adatainak a megjelenítése és dinamikus konfigurálása a megvalósított asztali alkalmazáson keresztül történik, amit szintén sikerült létrehozni. Az alkalmazáson belül még van lehetőség az érzékelők adatlapjainak a megnyitására, kapcsolási rajzának a megtekintésére, valamint a korábbi adatok mérései is visszanézhetőek érzékelők szerint csoportosítva és a mérés időpontja szerint rendezve, ezeket az adatokat le is lehet menteni a számítógépre. Sikerült még megvalósítani azt, hogy az érzékelők adatait folytonosan is lehessen olvasni, az ilyen formában olvasott adatok egy idődiagramon jelenjenek meg.

A dolgozat elkészítése során nagyon sokat tanultam, az Arduinóról, annak programozásáról, a soros kommunikációról, arról, hogy hogyan kell az érzékelőket az áramkörü laphoz kötni. Itt nagyon figyelni kell, hogy helyesen történjen, ellenkező esetben, az érzékelők tönkre mehetnek vagy akár maga az Arduino is egy hibás bekötés miatt. A MAUI keretrendszert működését is jobban megismertem, hogy hogyan is lehet egy asztali alkalmazást létrehozni.

### 5.2. Továbbfejlesztési lehetőségek

Megvizsgálva a projektem látható, hogy sok lehetőséget rejt még a továbbfejlesztésre. Egyik ilyen lehetőség további szenzorok integrálása a rendszerbe így még több adatot lehet mérni a rendszer használata közben.

Egy másik lehetőség az Arduino Uno kicserélése egy olyan mikrovezérlőre, ami rendelkezik Wifi modullal vagy egy Ethernet modul hozzáadása a mikrovezérlőhöz, amin keresztül tud csatlakozni az internethez. Így az asztali alkalmazás mellett létrehozható egy

mobilalkalmazás is, ezáltal az IoT világába is beléphetünk, hiszen így megvalósulhat az, hogy távolról is vezérelhető az alkalmazás.

## 6. Irodalomjegyzék

- [1] Arduino [Online]: <https://www.arduino.cc/en/about#what-is-arduino>
- [2] Wikipedia [Online]: <https://en.wikipedia.org/wiki/Arduino>
- [3] Wikipedia [Online]: <https://hu.wikipedia.org/wiki/Arduino>
- [4] Az Arduino bemutatása [Online]: <http://www.inf.u-szeged.hu/~makan/tananyagok/elektronikai-alapok-programozoknak/az-arduino-bemutatasa/>
- [5] Deep, A., Singh, J. P., Narayan, Y., Chatterji, S., & Mathew, L. (2015). *Robotic arm controlling using automated balancing platform*. <https://doi.org/10.1109/ccintels.2015.7437924>
- [6] Javaid, M., Haleem, A., Rab, S., Singh, R. P., & Suman, R. (2021b). Sensors for daily life: A review. *Sensors International*, 2, 100121. <https://doi.org/10.1016/j.sintl.2021.100121>
- [7] Wikipedia [Online] [https://en.wikipedia.org/wiki/Serial\\_communication](https://en.wikipedia.org/wiki/Serial_communication)
- [8] *ASCII Protocol and Commands - Brainboxes*. [Online]: <https://www.brainboxes.com/faq/ascii-protocol-and-commands>
- [9] *Pololu - 6.3. ASCII Commands*. [Online]: <https://www.pololu.com/docs/0J44/6.3>
- [10] Túrós, L. Z., & Székely, G. (2022). Érzékelők és mérőhálózatok= Sensors and Measuring Networks.
- [11] Davidbritch. (2023, January 30). *What is .NET MAUI? - .NET MAUI*. Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui>
- [12] Ficsor, L., Krizsán, Z., & Mileff, P. (2011). Szoftverfejlesztés. *Miskolci Egyetem, Miskolc*.
- [13] Michaelstonis. (2022, November 4). *Model-View-ViewModel*. Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA CALCULATOARE

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

Ș.l. dr. ing. Szabó László Zsolt