

# **DEBRECENI EGYETEM**

## **INFORMATIKAI KAR**

### **SZAKDOLGOZAT**

**Intelligens Csibekeltető: Raspberry Pi és Android  
Integráció**

Témavezető:  
**Dr. Kocsis Gergely**  
Egyetemi docens

Készítette:  
**Márton Áron**  
Programtervező  
informatikus (BSc)

**DEBRECEN**  
**2025**

# Köszönetnyilvánítás

Ezúton is szeretném kifejezni köszönetemet témavezetőmnek, Dr. Kocsis Gergelynek, aki értékes tanácsaival és támogatásával hozzájárult a szakdolgozatom elkészítéséhez. Szakmai tapasztalata és segítőkészsége nélkülözhetetlen volt számomra a kutatás során. Hálás vagyok, hogy minden türelmesen és segítőkészen válaszolt a kérdéseimre, valamint irányt mutatott a felmerülő nehézségek megoldásában. Köszönöm, hogy szakmai útmutatásával hozzájárult sikeres dolgozatom megvalósításához.

Továbbá köszönettel tartozom minden olyan személynek, aki segített nekem megvalósítani a projektet és ezáltal hozzájárult a szakdolgozatom sikeréhez.

Köszönettel,  
Márton Áron

# Tartalomjegyzék

1. Bevezető .....	1
2. Hardware .....	2
2.1. Raspberry Pi 4 Model B .....	3
2.2. DHT20 hőmérő és páratartalom érzékelő .....	5
2.3. Relé modul.....	7
2.4. Fűtőelem .....	9
2.5. Tápellátás.....	11
2.6. DC Motor.....	13
2.6. Mikrokapszoló .....	16
2.8. Ventilátor .....	18
2.9. LED panel.....	20
3. Hardware összeállítása .....	22
3.1. Modulok összeszerelése .....	22
3.2. Fejlesztői környezet.....	23
3.2.1. Fejlesztői környezet beállítása .....	23
3.3. Raspberry Pi programozás.....	23
3.4. Csibekeltető .....	24
3.4.1. Hőmérő .....	26
3.4.2. Páratartalom .....	27
3.4.3. Tojásforgatási mechanizmus.....	27
3.4.4. Szellőzés .....	28
3.4.5. Áramfogyasztás és költségek .....	28
3.5. Csibekeltetés .....	30
3.5.1. Statisztika.....	31
3.5.2. Csibekeltetés összefoglalva .....	34
3.6. Elkészülés folyamata .....	36
4. Software összeállítása .....	37
4.1. API.....	37
4.2. Csibekeltető mobil applikáció .....	39

4.2.1. Android operációs rendszer .....	40
4.2.2. Kotlin és Jetpack Compose .....	40
4.2.3. Android Studio fejlesztői környezet .....	41
4.2.4. API használata .....	41
4.2.5. Navigáció .....	44
4.2.6. Kezdőlap .....	45
4.2.7. Vezérlés.....	46
4.2.8. Beállítások.....	47
4.3. Szoftver architektúrális ábra.....	48
5. Összegzés .....	49
6. Irodalomjegyzék és hivatkozások .....	51

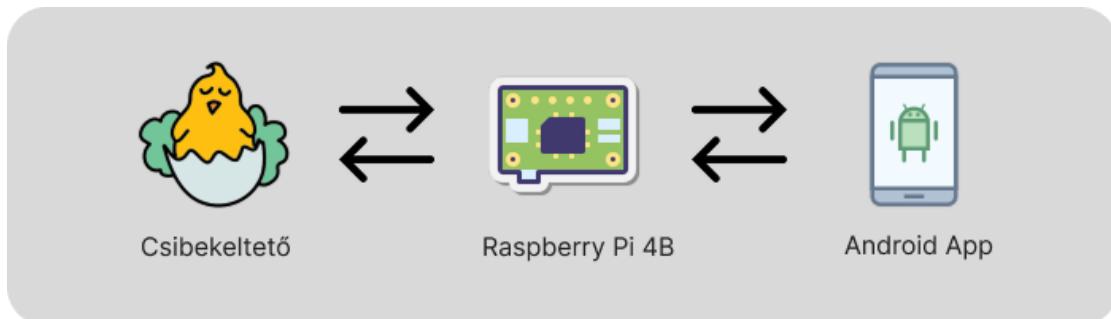
# 1. Bevezető

Gyerekkorom óta érdekeltek az elektronikai eszközök és az azokkal való barkácsolás, emellett pedig mindig is lenyűgözött a természet és az élőlények fejlődése. A szakdolgozatom témajául ezért egy olyan projektet választottam, amely ezt a két területet ötvözi: egy intelligens csibekeltető megépítését és vezérlését. A célom egy olyan automata rendszer létrehozása, amely Raspberry Pi segítségével képes megfelelő környezetet biztosítani a tojások keltetéséhez, miközben távolról is felügyelhető és irányítható.

A projekt kiválasztásában az is szerepet játszott, hogy egy gyakorlati, valós problémát szerettem volna megoldani. A keltetés precíz és állandó körülményeket igényel – mint például a megfelelő hőmérséklet, páratartalom és a tojások rendszeres forgatása – ezek automatikus megvalósítása ideális kihívás egy modern mikrokontrolleres rendszer számára. Emellett fontos szempont volt az is, hogy a rendszer egyszerűen újrahasznosítható vagy továbbfejleszthető legyen más állatok, vagy egyéb inkubációs folyamatok esetén is.

A keltető központi egysége egy Raspberry Pi 4B, amelyhez különböző hardverelemek csatlakoznak: egy 12 V-os fűtőelem, hőmérséklet- és páratartalom-érzékelők, mikrokapsolók, DC motor a tojásforgatáshoz, 5 V-os ventilátor a levegő keringetésére, valamint LED-ek a visszajelzéshez. Ezeket relémodulokon keresztül vezérlek, egy Python nyelven írt program segítségével. Továbbá egy Flask-alapú API-t is készítetek, amely lehetővé teszi az Androidos mobilalkalmazás általi távfelügyeletet és irányítást.

A dolgozat célja nem csupán a rendszer technikai megvalósításának bemutatása, hanem az is, hogy rávilágítson, miként lehet hétköznapi alkatrészekből és nyílt forráskódú szoftverekkel professzionális és költséghatékony megoldást készíteni valós feladatokra. Reményeim szerint a szakdolgozatom másokat is inspirál arra, hogy bátran kísérletezzenek és alkossanak saját rendszereket, akár oktatási, akár gyakorlati célból.



Szoftver és hardver kapcsolata

## 2. Hardware

A csibekeltető megépítéséhez olyan mikrokontrollert vagy egykártyás számítógépet kerestem, amely alkalmas volt a számomra ismert magasszintű programozási nyelveken való fejlesztésre. Fontos szempont volt továbbá, hogy rendelkezzen elegendő számítási kapacitással és megfelelő számú IO/GPIO porttal, hogy a későbbiekben tovább tudjam fejleszteni a csibekeltetőt.

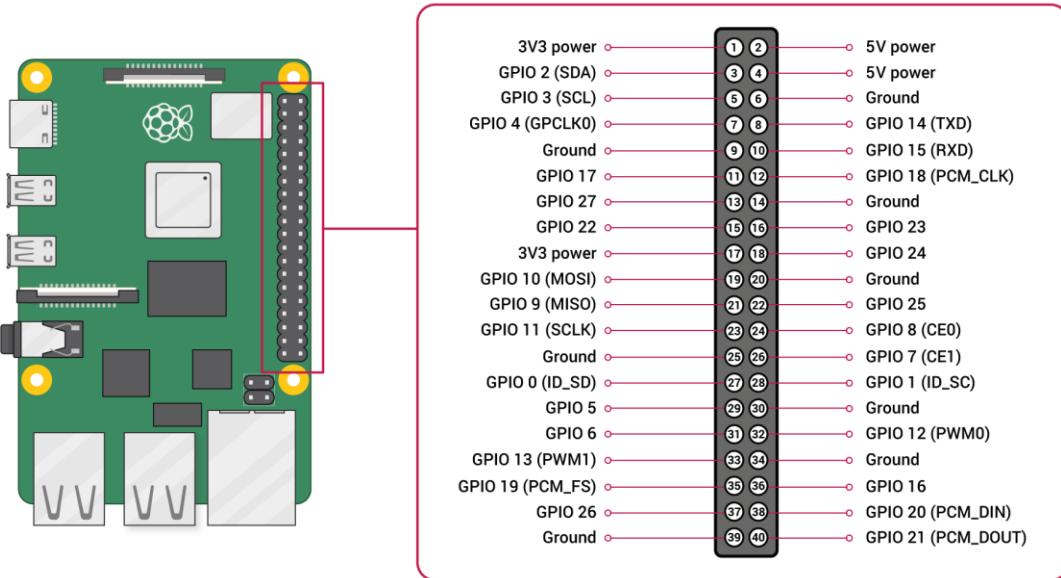
Ekkor találtam rá a Raspberry Pi 4 Model B (Raspberry Pi 4B, RasPi) egykártyás számítógépre, amely számos előnyvel rendelkezett. Ez az egykártyás számítógép lehetővé tette számomra, hogy könnyedén csatlakozzam az internethez, és a magas számítási kapacitása révén alkalmas volt az általam tervezett alkalmazásprogramozási felület (API, Application Programming Interface) futtatására. Emellett a modul megfelelő számú IO/GPIO porttal rendelkezett, amely lehetővé tette a különböző eszközök vezérlését és bővítését.

Kezdetben beszereztem pár eszközt, kezdve a RasPi-val, egy 12 VDC fűtőelemmel, egy 5 VDC ventilátorral, egy 4 csatornás relé modullal, egy hőméréklet és páratartalom érzékelővel, illetve anya-anyá és anya-férfi vezetékekkel.

A tervem az volt, hogy a csibekeltetőt a RasPi fogja irányítani teljesen autonóm módon vagy minimális emberi beavatkozással. A Raspberry Pi a gyári tápegységről kapná az áramot. A többi hardver komponenst, mint például a fűtőelem, illetve a DC motor külön-külön kapná az áramot. Az egész rendszer hozzájut az internethez, így tudok a csibekeltetővel kommunikálni az Android alkalmazásommal keresztül.

Továbbá beszereztem egy 12 VDC motort egy L298M motor meghajtó modullal. Ez a DC motor fogja a tojásokat megforgatni, ami nagy szerepet játszik a keltetésben.

## 2.1. Raspberry Pi 4 Model B



1. ábra: Raspberry Pi 4 Model B<sup>1</sup>

A Raspberry Pi 4 Model B (1. ábra) egy olyan egykártyás számítógép, amely változatos és erős erőforrásokkal rendelkezik és amellyel, lehetséges egy operációs rendszert futtatni. Ez az egykártyás számítógép, olyan fejlesztőknek való, akik IoT (Internet of Things, Tárgyak Internete) projektjeiken dolgoznak, olyan prototípus fejlesztőknek, akik gyorsan és hatékonyan megtudják valósítani a tervüket, illetve kezdetben az alapelvek volt, hogy a diákok és a gyerekek megismerkedjenek a programozással. A Raspberry Pi család nagyon elterjedt egykártyás számítógépnek számolódik.

Mivel ez egy egykártyás számítógép, így nem igényel PC vagy más külső hardvert a működéséhez, hiszen tud egy operációs rendszert futtatni egy SD kártyáról vagy egy SSD-ről, illetve az energiaellátását-át tudja egy adaptorról kezelni.

Főbb paraméterei<sup>2</sup>:

- Tápfeszültség: A RasPi ajánlott tápfeszültsége 5 V(DC). A megfelelő működéshez legalább 3 A áramerősséggű USB-C tápegység javasolt.
- I/O felület: A RasPi rendelkezik kettő darab 3.3 V portal, kettő darab 5 V portal, amik folyamat adják az áramot, több GPIO portal rendelkezik, amikkel lehetséges a többi hardverrel a kommunikáció, illetve sok más port is jelen van a RasPi-n.

<sup>1</sup> Kép forrása: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio>

<sup>2</sup> Bővebb információk az Irodalomjegyzékben találhatók. [1]

- Áramfelvétel: A RasPi üresjáratban körülbelül 600mA használ, teljes terhelésen pedig képes használni akár 2-3 A-t is.
- CPU: Egy Broadcom BCM2711, négymagos Cortex-A72 (ARM v8), 64 bites, 1.5 GHz processzor található a RasPi-n.
- RAM: A Raspberry Pi megvásárolható 2 GB, 4 GB vagy 8 GB LPDDR4 SDRAM változatban. A csibekeltőben egy 4GB verzió van.
- Méret: 85 mm × 56 mm × 17 mm (hitelkártya méretű)
- Operációs rendszer: Raspberry Pi OS (RaspbianOS, Linux alapú), de más rendszerek is futtathatók (Ubuntu, Android stb.)

A Raspberry Pi 4 Model B 40 tűs GPIO csatlakozókkal rendelkezik, amik lehetővé teszik a kommunikációt a külső hardverekkel, illetve azok irányítását.

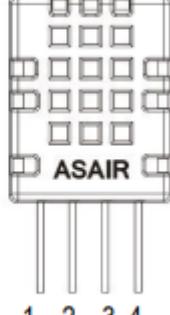
A GPIO-csatlakozók (General Purpose Input/Output, általános célú bemeneti és kimeneti portok) nagy jelentőséggel bírnak a beágyazott rendszerek fejlesztésében. Ezeket a csatlakozókat digitális bemenetként és kimenetként is konfigurálhatjuk.

Ezek a GPIO-lábak nélkülözhetetlenek olyan projekteknél, ahol fizikai eszközöket (pl. szenzorokat, LED-eket) kell közvetlenül vezérelni vagy olvasni. Szerepük kiemelten fontos robotikában, okosotthon-megoldásokban és automatizált rendszerekben, mivel rajtuk keresztül kommunikál a hardver a külvilággal. Oktatási környezetben is nélkülözhetetlenek, mert segítségükkel gyakorlati tapasztalat szerezhető elektronikai és programozási alapismeretkből.

A Raspberry Pi Alapítvány folyamatosan fejleszti hardveres és szoftveres ökoszisztemáját, hogy még szélesebb körben használható legyen oktatási és ipari célokra. Az újabb generációs Raspberry Pi modellek nagyobb teljesítményt és energiahatékonyságot kínálnak, így versenyképesek maradnak a beágyazott rendszerek piacán. A vállalat egyre nagyobb hangsúlyt fektet az ipari és vállalati felhasználásra, különösen az IoT (Internet of Things) és automatizálási megoldások területén. A szoftveres támogatás folyamatosan bővül, beleértve a 64 bites operációs rendszereket, a Raspberry Pi OS (Raspbian) fejlesztését és a kompatibilitás javítását más Linux-disztribúciókkal. A jövőbeli modellek várhatóan még fejlettebb interfészekkel és kommunikációs lehetőségekkel rendelkeznek majd, hogy könnyebben integrálhatók legyenek okoseszközökbe és ipari rendszerekbe. Az oktatás és a programozás népszerűsítésére a cég továbbra is elérhető árú eszközöket fejleszt, hogy minél több diák és hobbifejlesztő férhessen hozzá a technológiához.

## 2.2. DHT20 hőmérséklet és páratartalom érzékelő

Pins	Name	Describe	
1	VDD	Power supply(2.2v to 5.5v)	
2	SDA	Serial Data Bidirectional port	
3	GND	Ground	
4	SCL	Serial clock Bidirectional port	



2.ábra: DHT20 - AHT20 tűs modul - I<sup>2</sup>C hőmérséklet- és páratartalom-érzékelő<sup>3</sup>

A DHT20 (2. ábra) egy hőmérséklet és páratartalom szenzor, amely képes megfigyelni a körülötte lévő levegő hőmérsékletét és páratartalmát. A DHT20 a DHT11 új, továbbfejlesztett változata, amely dedikált ASIC-érzékelőchippel van felszerelve. Ezt az érzékelőt egy 2003-ban alapított, ASAIR nevű kínai cég gyártja.

A DHT20 szabványos I<sup>2</sup>C interfészen keresztül kommunikál, ami egyszerű integrációt tesz lehetővé különböző mikrokontrollerekhez. Az érzékelő gyors válaszidővel rendelkezik, és kiváló hosszú távú stabilitást biztosít, így ideális választás olyan alkalmazásokhoz, ahol megbízható és pontos hőmérséklet- és páratartalom-mérés szükséges.

Az érzékelő pontossága páratartalom mérésére ±3% RH, hőmérséklet mérésére pedig ±0,5°C. Az eszköz széles tápfeszültségtartományban működik (2,2 V és 5,5 V között), továbbá gyors reakcióidővel és kiváló hosszú távú stabilitással bír. Széles körben alkalmazható például klímatechnikai rendszerekben, háztartási készülékekben, autókban, orvosi és mérési berendezésekben, valamint IoT eszközökben.

A hőmérséklet és páratartalom szenzor kis mérete miatt ideális választás olyan alkalmazásokhoz, ahol korlátozott a hely, de pontos hőmérséklet- és páratartalom-mérésre van szükség. Kompakt kialakítása lehetővé teszi az egyszerű integrációt különböző eszközökbe, beleértve a csibekeltető rendszert is, ahol fontos a stabil és megbízható környezeti érzékelés.

Az olvasó paraméterei az alábbiak<sup>4</sup>:

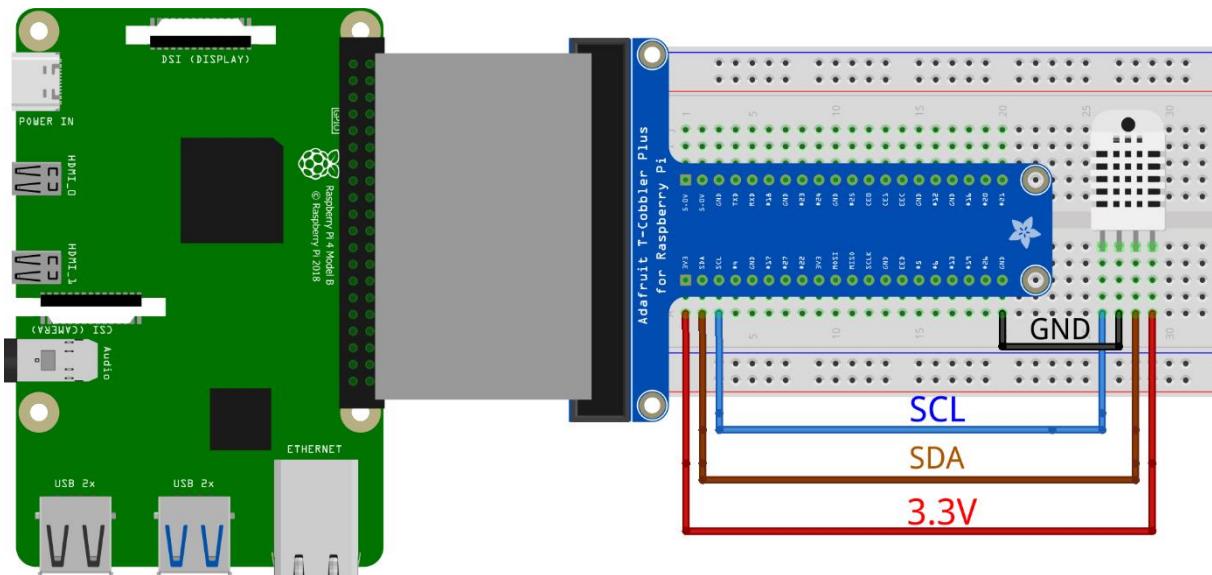
- Áramfelvétel: A szenzor készenléti állapotban körülbelül 250 nA használ, mérési állapotban pedig körülbelül 980 µA.

<sup>3</sup> Kép forrása: <https://aqicn.org/air/sensor/spec/asair-dht20.pdf>

<sup>4</sup> Bővebb információk az Irodalomjegyzékben találhatók. [2]

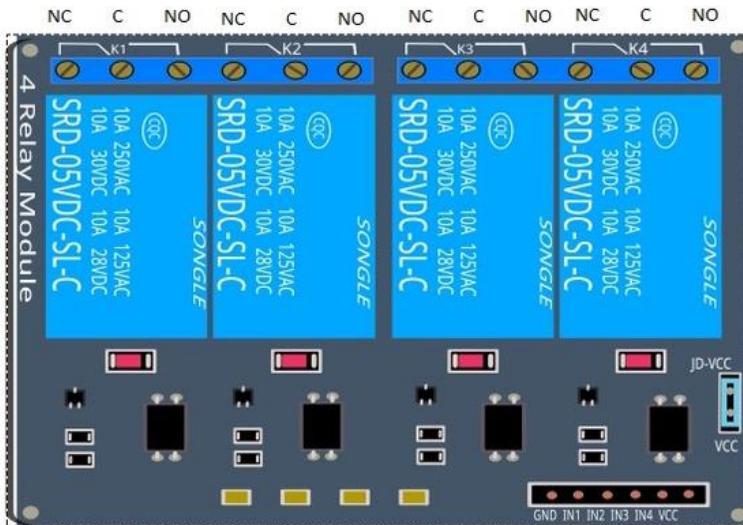
- Tápfeszültség: 2,2 V és 5,5 V között van. Az ajánlott 3,3 V.
- Mérési pontosság: Hőmérséklet  $\pm 0,5^{\circ}\text{C}$ , páratartalom  $\pm 3\%$  RH
- Kommunikációs protokoll: I<sup>2</sup>C (digitális)
- Működési frekvencia: I<sup>2</sup>C szabvány szerint (általában max. 400 kHz „fast mode”)

A Raspberry Pi 4B és a DHT20 összekapcsolása során, oda kell figyelnünk a helyes összekötésre, illetve az ajánlott tápfeszültségre. Az ilyen technikai részletek figyelembevétele nélkülözhetetlen az elektronikai projekt tervezése és megvalósítása során. A Raspberry Pi 4B és a DHT20 összeköttetését (3. ábra) az alábbi módon valósítottam meg:



3. ábra: DHT20 és Raspberry Pi kapcsolási rajz

## 2.3. Relé modul



4.ábra: Négy csatornás relé<sup>5</sup>

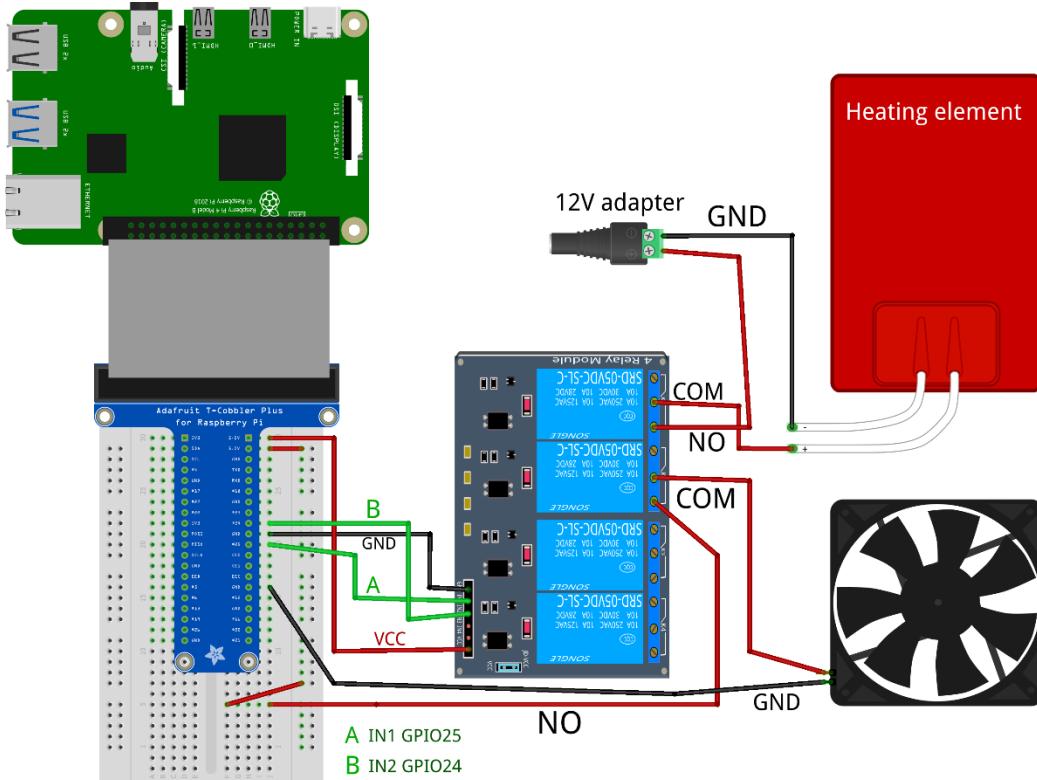
A 4 modulból álló relé (4. ábra) egy olyan kapcsolómodul, amely négy egymástól függetlenül vezérelhető reléből épül fel. Ezekkel a relékkal alacsony feszültségű jelekkel (pl. GPIO kimenetek) nagyobb feszültségű vagy nagyobb áramerősséggű fogyasztókat (fűtőelemek, motorok) kapcsolhatunk ki- vagy bekapcsolt állapotba. A modul általában LED-visszajelzőkkel van ellátva, és optocsatolós védelemmel rendelkezik, hogy biztonságosan levállassa a vezérlő elektronikát a nagyobb teljesítményű áramköröktől. Különösen hasznos házi automatizálási projektekben, például lámpák, elektromos eszközök vagy motorok távvezérlésére. A Raspberry Pi GPIO-csatlakozóival könnyen vezérelhető.

A modul reléi egyenáramú fogyasztóknál akár 30 V / 10 A-ig, váltóáramú fogyasztóknál pedig akár 250 V / 10 A-ig terhelhetők. Működésükhez 5VDC tápfeszültség szükséges, így könnyen használhatóak mikrokontrollerekkel vagy Raspberry Pi-vel, GPIO-portokon keresztül. Optocsatolós leválasztásuk révén biztonságosan elkülönítik a vezérlőáramkört a kapcsolt fogyasztóktól. Gyakran használt eszköz IoT, házi automatizálási, vagy robotikai projektekben, például világítás- és motorvezérlésben.

A relémodulok működéséhez három fő csatlakozási pontot használnak: COM (Common – közös pont), NC (Normally Closed – alaphelyzetben zárt) és NO (Normally Open – alaphelyzetben nyitott). A COM érintkező a közös pont, amely a relé állapotától függően az NC vagy az NO érintkezőhöz csatlakozik. Alaphelyzetben az NC és a COM között folyik az áram, míg a NO nyitott állapotban van. Amikor a relé aktívvá válik, az NC megszakad, és a

<sup>5</sup> Kép forrása: <https://yadavdharm.wordpress.com/2020/08/12/relay-interface-with-arduino/#jp-carousel-670>

COM az NO-hoz csatlakozik, így az áramkör záródik. Ez a működési elv lehetővé teszi különböző kapcsolási konfigurációk kialakítását, például világítás- vagy motorvezérlésben, ahol az áramkör állapota a vezérlőjelektől függően változik.



5.ábra: Relé kapcsolási rajz

## 2.4. Fűtőelem



6.ábra: 12 V PTC Fűtőelem<sup>6</sup>

A fűtőelem (6. ábra) egy 12 VDC PTC (Positive Temperature Coefficient, Pozitív Hőmérsékleti Együttható) amely képes akár 110 Celsius fokra felmelegedni. A PTC fűtőelem önszabályozó, így nem igényel bonyolult vezérlést a hőmérséklet szabályozására. Biztonságosabb, mint a hagyományos ellenállásos fűtőelemek, mert nem melegszik túl, így csökkenti a tűzveszélyt. Egyenletes hőeloszlást biztosít, ami elengedhetetlen a tojások megfelelő keltetéséhez. Gyorsan eléri az üzemi hőmérsékletet, így nem kell sok időt várnai a megfelelő környezet kialakítására. Stabilan tartja a hőmérsékletet a tojások körül, ami kritikus a sikeres keltetéshez.

A fűtőelem kis méretének és egyszerű szerelhetőségének köszönhetően könnyedén elhelyezhető a keltető belső falára. A fűtőelem kiválóan együttműködik a hőmérsékletérzékelővel, amely pontos visszajelzést ad a rendszernek a belső hőállapotról. A Raspberry Pi nem közvetlenül, hanem egy relén keresztül vezérli a fűtőelemet, mivel az 12V feszültségen működik, ami meghaladja a GPIO portok feszültségtűrését. A relé lehetőséget biztosít arra is, hogy biztonsági logikát építsünk be, például időzített kikapcsolást vagy túlmelegedés esetén lekapcsolást. A fűtőelem működését LED-ekkel is visszajelzem a felhasználó számára, így könnyen nyomon követhető, mikor aktív a fűtés. A fűtőelem hosszú élettartamú, nincs benne mozgó alkatrész, így karbantartást nem igényel.

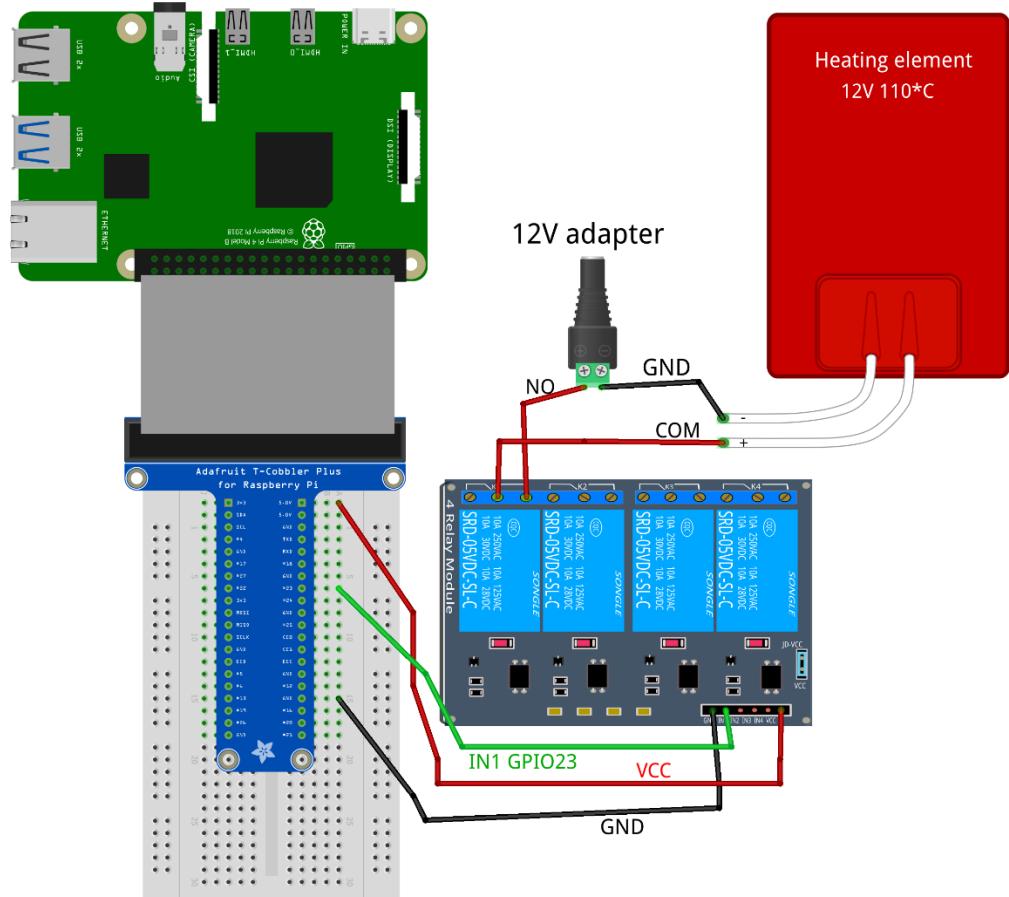
A keltető megfelelő hőmérsékletének biztosítása érdekében egy PTC önszabályozó fűtőelemet választottam, amelynek legfontosabb műszaki jellemzői az alábbiakban találhatók:

- Típus: PTC (Positive Temperature Coefficient, Pozitív Hőmérsékleti Együttható) kerámia fűtőelem
- Üzemi feszültség: 12 VDC

<sup>6</sup> Kép forrása: <https://www.aliexpress.com/item/1005007928878692.html>

- Teljesítmény: Több változat is létezik 5 W-tól egészen 120 W-ig. A keltetőben egy 24 W (12 V/2 A) fűtőelem található
- Felhasználási terület: Inkubátorok, 3D nyomtatók, akkumulátor-melegítők
- Kábelezés: Két kábel található rajta: egy pozitív és egy negatív, így a bekötése egyszerű.

A rendszer egyik legfontosabb része a fűtőelem, melynek kapcsolását az alábbi rajzon szemléltetem.



7. ábra: Fűtőelem és Raspberry Pi kapcsolási rajz

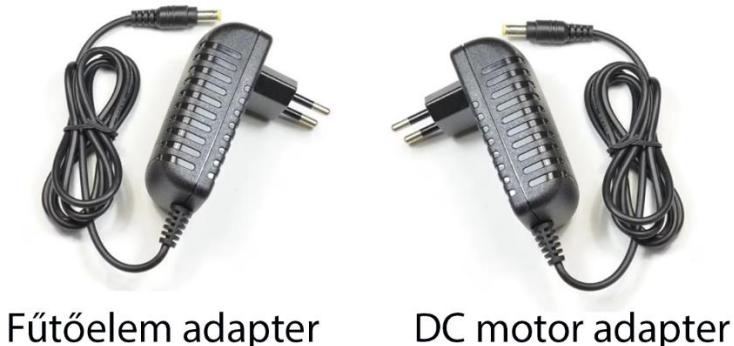
## 2.5. Tápellátás

Ahhoz, hogy a keltetőben működhessen a fűtőelem, amely melegíti a tojásokat, illetve a DC motor, amely forgatja a tojásokat, kénytelen voltam kettő darab külső adaptert beszerezni. Az adapterek (8. ábra) teljesen megfeleltek az elvárásaimnak. Az adapter egy műanyag házban helyezkedik el, amely védelmet nyújt a belsejében található elektronika számára. A projekt mobilitását is szem előtt tartva, olyan adaptereket választottam, amelyek könnyen cserélhetők, mozgathatók, és nem igényelnek fix beépítést.

A méretei is ideálisnak bizonyultak a projektben való használathoz, hiszen minden össze  $5.5 \times 2.5 \times 13$  mm-es volt. Ezek a kompakt méretek lehetővé tették a könnyű elhelyezést és mozgatást a projekten belül.

A keltető működéséhez elengedhetetlen a megbízható és stabil tápellátás, hiszen a hőmérséklet-szabályozás és a tojásforgatás folyamatos energiaellátást igényel. A kiválasztott 12V-os adapterek elegendő áramerősséget biztosítanak mind a fűtőszál, mind a DC motor számára. Fontos szempont volt, hogy a tápegységek túláram elleni védelemmel rendelkezzenek, így növelte a rendszer biztonságát. Az adapterek csendes működése előnyös, mivel nem zavarnak a keltetési időszak alatt.

A sorkapocs átalakítók (9. ábra) használatával a vezetékek cseréje vagy újraszerelése jelentősen leegyszerűsödött, így sok időt takaríthatok meg. A csatlakozások könnyen nyomon követhetők, így hibakeresés esetén gyorsan beazonosítható egy esetleges probléma forrása.



8.ábra: Tápegységek, Andowl 12 V 1-2 A<sup>7</sup>

<sup>7</sup> Kép forrása: <https://www.emag.hu/halozati-adapter-tapfeszultseg-ac-100-240v-dc-12v-2a-q-dc507/pd/DJS9TBYBM/>



9.ábra: DC – Sorkapocs átalakító<sup>8</sup>

A sorkapocs átalakító használata különösen előnyös volt a projektem során, mivel nagyban megkönnyítette a tápellátási vezetékek csatlakoztatását. Az átalakító egyik oldalán egy szabványos DC csatlakozó foglal helyet, amelybe az adapter vége könnyedén bedugható, míg a másik oldalon két csavaros kapocs található, amelyekbe a vezetékek befogathatók. Ez a kialakítás lehetővé tette, hogy forrasztás nélkül, gyorsan és stabilan lehessen rögzíteni az egyes tápvezetékeket. A csavaros rögzítésnek köszönhetően a vezetékek nem tudnak kilazulni, ami csökkenti a kontaktushibák és az időszakos megszakadások esélyét. A sorkapocs biztonságos és mechanikusan is stabil kapcsolatot biztosít, amely ellenáll a rezgéseknek és a mozgatásnak is.

A moduláris rendszer szempontjából ez a megoldás ideálisnak bizonyult, mivel a különböző eszközök – mint például a fűtőszál, motor vagy a ventilátor – könnyedén leválaszthatók vagy cserélhetők anélkül, hogy újra kellene forrasztani az egész rendszert. Emellett a sorkapocs használata segít rendszerezni a kábelezést is, így az egész keltető belső elrendezése letisztultabb és könnyebben áttekinthető marad. A csatlakozók polaritása jól láthatóan fel van tüntetve az átalakítón, így minimalizálható a hibás bekötés veszélye.

---

<sup>8</sup> Kép forrása: <https://www.villanyonline.hu/21x55-dc-aljzat-2p-sorkapocs-dca005-7024>

## 2.6. DC Motor



10. ábra: DC motor N20-12-75<sup>9</sup>

A tojásforgató mechanizmust egy 12 V-os, 75 RPM-es egyenáramú, fémházas motor (10. ábra) működteti, amelyet a Raspberry Pi 4B vezérel. A motor D-alakú tengelyéhez egy forgatókar kapcsolódik, amely közvetlenül mozgatja a tojásokat tartó keretet. A rendszer nem használ pozícióérzékelőt vagy mikrokapcsolót; a mozgatás időalapú vezérléssel történik. A forgatás vezérlése úgy van kialakítva, hogy a motor hat óránként egyszer aktiválódik. A programozott ciklus során a motor először 4 másodpercig forog az egyik irányba, majd 4 másodpercig az ellenkező irányba, így biztosítva a tojások oda-vissza mozgatását. A motor 20%-os PWM kitöltési tényezővel üzemel, ami lecsökkenti a fordulatszámát és lehetővé teszi a lassú, kíméletes forgatást. A 75 RPM-es motor ebben a konfigurációban nem okoz hirtelen mozgást vagy rázkódást, így kifejezetten alkalmas a tojások forgatására. Mivel az irányváltás is időzítve történik, a rendszer egyszerű felépítésű, mechanikus visszajelzés nélkül is megbízhatóan működik. A motor stabilan van rögzítve a keltető szerkezetéhez, hogy a rendszer működése során fellépő nyomaték ne okozzon elmozdulást. Ez az időzített, automatizált megoldás hatékonyan helyettesíti a manuális tojásforgatást, amely a csibék egészséges fejlődéséhez elengedhetetlen.

A használt DC motor műszaki jellemzői:

- Tápfeszültség: 12 VDC
- Méretek: 26 mm x 10 mm x 12 mm, ami számomra tökéletes a keltetőben
- Tengely kialakítása: D-alakú tengely (6 mm átmérő)
- Fordulatszám: 75 RPM, az alacsony fordulatszám megfelelő a folyamatos tojásforgatáshoz
- Üzemi hőmérséklet tartomány: 0–50 °C (általános használatra)
- Alkalmazási terület: robotika, automatizált mozgatás, keltetők, DIY projektek

---

<sup>9</sup> Kép forrása: <https://nfpshop.com/product/12mm-dc-gearmotor-24mm-type-model-nfp-jga12-n20>

A csibekeltető motorvezérléséhez egy L298N (*11. ábra*) típusú, kettős teljes H-híd motor meghajtó modult alkalmazok. Ez az eszköz lehetővé teszi egyenáramú motorok irányának és sebességének vezérlését külső mikrovezérlő, jelen esetben a Raspberry Pi 4B segítségével. A modul támogatja a PWM jelalapú vezérlést, így finoman szabályozható a motor fordulatszáma, ami elengedhetetlen a tojások kíméletes forgatásához. Az L298N két teljes H-híddal rendelkezik, így akár két motor független vezérlésére is alkalmas, bár jelen<sup>10</sup> rendszerben csak egyet használunk. A modul 12 V-os tápfeszültséggel működik, és megbízható teljesítményt nyújt hosszú távú, automatikus üzemelés során is.



*11. ábra: L298N-MOD motor meghajtó modul<sup>10</sup>*

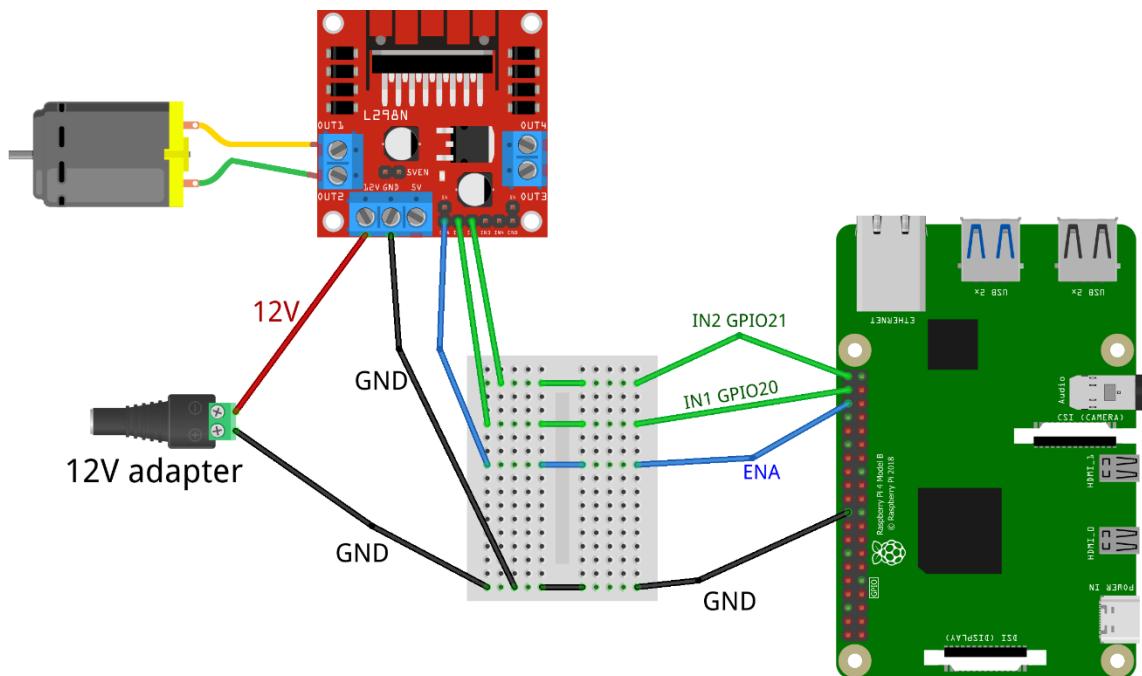
Az L298N modul műszaki jellemzői az alábbiak:

- Működési feszültség: 5 V – 35 V (motor tápfeszültség)
- Áramerősség csatornánként: max. 2 A (rövid ideig), ajánlott 1–1.5 A
- Csatornák száma: 2 (két DC motor vagy 1 léptetőmotor)
- Vezérlő bemenetek: IN1, IN2, IN3, IN4 (irány), ENA, ENB (sebesség/PWM)
- PWM támogatás: Igen (ENA és ENB lábakon keresztül)
- GND közösítés kötelező: Igen (RasPi GND és L298N GND össze kell legyen kötve)

A motor működtetéséhez szükséges elektronikai megoldást az alábbi kapcsolási rajzon szemléltetem, bemutatva a motor és vezérlőelemei közötti kapcsolatokat.

---

<sup>10</sup> Kép forrása: <https://www.minikits.com.au/l298n-mod-01>



12. ábra: DC motor és Raspberry Pi kapcsolási rajz

## 2.6. Mikrokapcsoló



13. ábra: Mikrokapcsoló V-156-1C25<sup>11</sup>

A keltető biztonságos működésének egyik fontos eleme a fedél helyzetének folyamatos ellenőrzése. Ennek megvalósításához egy V-156-1C25 típusú végálláskapcsolót (13. ábra) alkalmaztam, amely mechanikus elven működő mikrokapcsoló, hosszabbított karral ellátva. A kapcsoló úgy került elhelyezésre a szerkezetben, hogy zárt fedél esetén nyomás alatt tartva zárt állapotban van, míg a fedél nyitásakor kienged, és ezzel átvált nyitott állapotba. A kapcsoló jele közvetlenül a Raspberry Pi-hez csatlakozik, amely így azonnal érzékeli, ha a fedél nyitva van. Ilyen esetben a rendszer automatikusan leállítja a fűtőelemet, a tojásforgató motort, valamint a ventilátort is, ezáltal megelőzve a hirtelen hő- vagy párvaveszteséget. A nyitott állapot egyben riasztást is generál, amely lehet fényjelzés, így a felhasználó azonnal értesül a helyzetről.

A rendszer mindaddig nem áll vissza alapállapotba, amíg a fedél vissza nem záródik és a mikrokapcsoló ismét zárt állapotba nem kerül. E megoldás növeli a berendezés üzembiztonságát és hozzájárul az optimális keltetési környezet fenntartásához. A kapcsoló egyszerű felépítése és megbízhatósága révén ideális választás az ilyen típusú alkalmazásokhoz. A fedélpozíció ellenőrzése tehát szerves része a keltető automatizált vezérlésének.

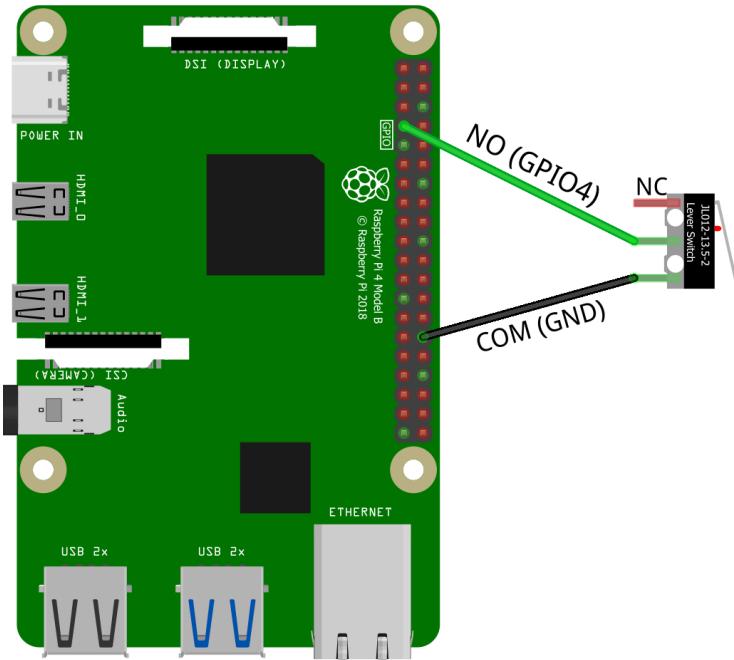
A végálláskapcsoló alkalmazásához elengedhetetlen a pontos műszaki paraméterek ismerete, ezért az alábbiakban összefoglalom a V-156-1C25 típusú kapcsoló legfontosabb specifikációit:

- Kapcsolási típus: SPDT (Single Pole Double Throw) – azaz három kivezetéssel rendelkezik: NO (normally open), NC (normally closed) és COM (közös)
- Mechanikai élettartam: 50 000 000 ciklus, ami megfelelő egy csibekeltetőben
- Elektromos élettartam: 100 000 ciklus
- Működési hőmérséklet-tartomány:  $-25^{\circ}\text{C}$ -tól  $+80^{\circ}\text{C}$ -ig

<sup>11</sup> Kép forrása: <https://techfun.hu/termek/v-156-1c25-vegallaskapcsolo-kiterjesztett-retesszel/>

- Szerelhetőség: Csalváros rögzítés 5.3 mm átmérőjű furatokkal
- Méret: 49.2 mm x 19.9 mm x 10.3 mm (kar nélkül)

A kapcsolási rajz az alábbiak szerint nézne ki, ahol a mikrokapcsoló egyik lába a Raspberry Pi egyik bemeneti GPIO lábára, a másik pedig földre (GND) van kötve:



14. ábra: Mikrokapcsoló és Raspberry Pi kapcsolási rajz

## 2.8. Ventilátor



15. ábra: 5 V ventilátor 50x50mm<sup>12</sup>

A rendszerben alkalmazott ventilátor (15. ábra) egy 5 V-os működtetésű, 2 tűs, 50 mm x 50 mm méretű légkeverő eszköz, amelynek elsődleges feladata a belső levegő áramoltatása. Kompakt méretének és kis energiaigényének köszönhetően kiválóan alkalmas mikrokontroller-alapú rendszerekhez, mint például a Raspberry Pi által vezérelt csibekeltető. A kétvezetékes kialakítás leegyszerűsíti a bekötést, mivel csupán a tápfeszültség (VCC) és a föld (GND) csatlakoztatása szükséges.

A ventilátor működése során cirkuláltatja a levegőt a zárt térben, ezáltal biztosítva az egyenletes hőmérséklet-eloszlást a keltető teljes területén. Ez a funkció különösen fontos, mivel a kelési folyamat során a hőeloszlás egyenetlensége negatívan befolyásolhatja az embriók fejlődését. A ventilátor a hőforrás, például a fűtőelem környezetéből elvonja a hőt, majd azt eloszlatja a keltető többi részében. A folyamatos, de alacsony zajszintű működés lehetővé teszi, hogy a készülék hosszabb ideig is üzemeljen anélkül, hogy megzavarná a kelési környezetet. A ventilátor relével vezérelhető, ezáltal az automatizált hőmérséklet-szabályozási rendszer részeként működik.

A megfelelő levegőkeringetés egyben hozzájárul a páratartalom stabilitásához is, mivel segíti a nedvesség egyenletes eloszlását. Összességében a ventilátor egy egyszerű, de a rendszer működése szempontjából nélkülözhetetlen komponens.

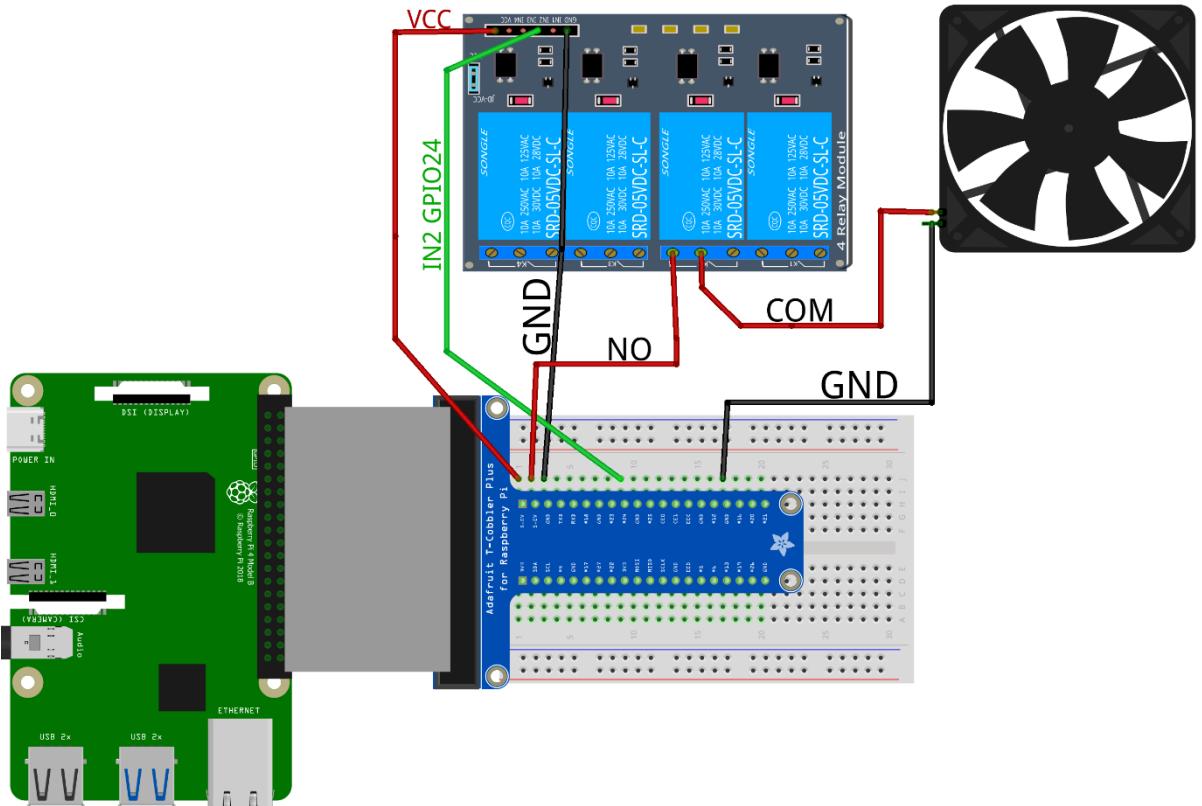
Az említett 5 V-os, 2 tűs ventilátor specifikációi az alábbiak szerint foglalhatók össze:

- Tápfeszültség: 5 VDC
- Csatlakozás: 2 tűs (piros – VCC, fekete – GND)
- Méret: 50 mm x 50 mm x 10 mm

<sup>12</sup> Kép forrása: <https://www.lojadopaineldeled.com.br/cooler-5v-030a-para-gabinete-de-led-outdoor-10-unidades>

- Fordulatszám: kb. 5000–7000 RPM
- Zajszint: alacsony, tipikusan 20–25 dB
- Anyag: műanyag ház és lapátok

A ventilátor a relé modul NO (Normally Open) kimenetére lett csatlakoztatva, így csak akkor kap 5 V tápfeszültséget, amikor a Raspberry Pi vezérlője aktiválja a relét, ahogyan az a kapcsolási rajzon is látható.



16. ábra: Ventilátor és Raspberry Pi kapcsolási rajz

## 2.9. LED panel



17. ábra: Státusz LED panel a csibekeltetőn

A csibekeltető működésének vizuális visszajelzését LED-ek segítségével valósítottam meg (17. ábra). Összesen öt darab állapotjelző LED-et alkalmaztam, amelyek mindegyike a csibekeltető egy-egy fontos funkcióját jelzi. Ez a megoldás lehetővé teszi, hogy a felhasználó gyorsan és egyszerűen információt kapjon az eszköz működéséről. A piros LED azt jelzi, ha a fedél nyitva van, ami hatással lehet a belső hőmérséklet stabilitására. A zöld LED világít, amikor a fűtőszál aktív, ezzel jelezve, hogy a rendszer melegíti a teret. A fehér LED a ventilátor működését mutatja, amely biztosítja a levegő megfelelő áramlását a belső térben. A sárga LED a DC motor aktiválását jelzi, ami a tojásforgatás funkcióért felelős. A hideghéjű LED kizárolag a rendszer indításakor, körülbelül 10 másodpercig világít, így visszajelzést ad arról, hogy a Raspberry Pi elindult.

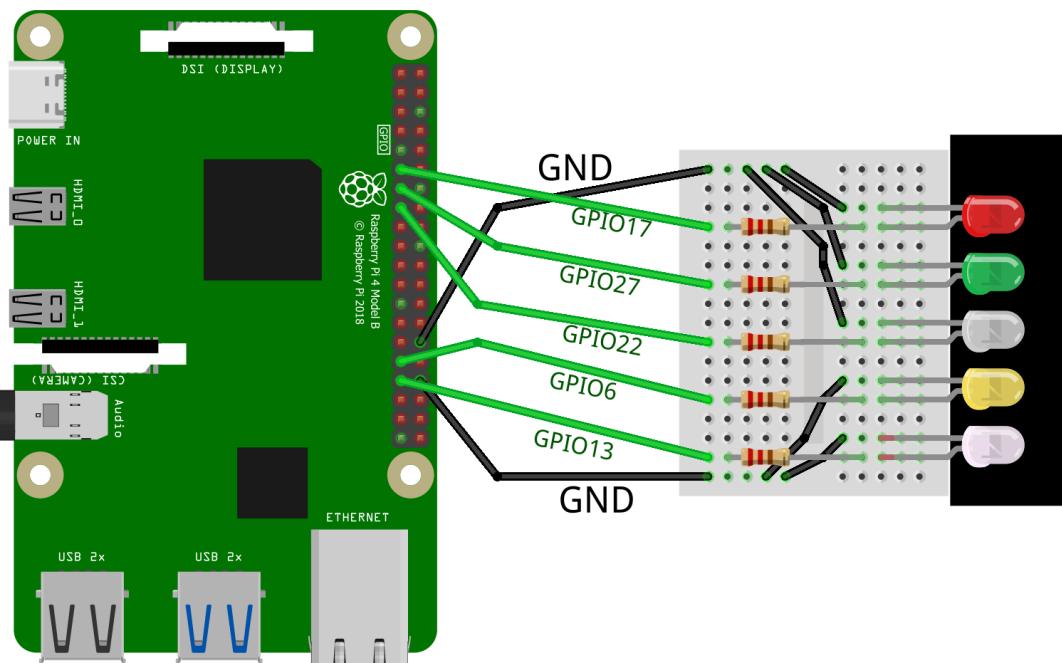
A LED-eket a Raspberry Pi GPIO lábairól vezérlek, amelyek 3.3V feszültséget biztosítanak. Annak érdekében, hogy megvédjem a LED-eket a túláramtól, minden LED előtt 330 ohmos előtétellenállást kööttem. Ezek az ellenállások korlátozzák az átfolyó áramot, így a LED-ek biztonságosan működhettek. Az ellenállások nélkül a LED-ek könnyen túlmelegedhetnek vagy akár ki is éghetnek. A 330 ohmos érték ideális kompromisszumot jelent a fényerő és a biztonságos működés között.

Az alábbiakban részletesen bemutatom az egyes LED-ek műszaki jellemzőit és funkcióját, amelyek a csibekeltető állapotának vizuális visszajelzését szolgálják.

- **Piros LED:** A fedél nyitott állapotát jelzi. Standard 5 mm-es piros LED, körülbelül 2.0 V nyitófeszültséggel és 20 mA névleges áramfelvétellel. A Raspberry Pi 3.3 V-os GPIO kimenetéről kap tápfeszültséget, és a megfelelő áramkorlátozást egy 330 ohmos előtétellenállás biztosítja.

- Zöld LED: A fűtőszál bekapcsolt állapotát mutatja. 5 mm-es zöld LED, nyitófeszültsége kb. 2.1 V, névleges árama 20 mA. A 3.3 V-os GPIO kimenet és a LED közé egy 330 ohmos ellenállás van bekötve, így az áram biztonságos szinten marad.
- Fehér LED: A ventilátor működését jelzi. 5 mm-es fehér LED, nyitófeszültsége 3.0–3.2 V, 20 mA áramfelvétel mellett. A Raspberry Pi 3.3 V-os GPIO kimenetéről kap tápot, egy 330 ohmos előtétellenálláson keresztül. A fényereje így kissé alacsonyabb lehet, de a visszajelzés céljára elegendő.
- Sárga LED: A tojásforgató DC motor aktív állapotát mutatja. 5 mm-es sárga LED, 2.0–2.2 V nyitófeszültséggel és 20 mA áramfelvétellel. A 3.3 V-os kimenet és a GND közé van bekötve, sorosan 330 ohmos ellenállással.
- Hideghéjér LED: A Raspberry Pi indulásakor világít körülbelül 10 másodpercig. 5 mm-es hideghéjér LED, nyitófeszültsége 3.0–3.2 V, áramfelvétele 20 mA. A LED a 3.3 V-os GPIO kimenetről kap feszültséget, 330 ohmos ellenállással sorba kötve, kizárálag a bootolás során vezérelten aktiválódik.

Az alábbi ábrán látható a LED-ek kapcsolási rajza a Raspberry Pi GPIO lábaira kötve, 330 ohmos előtétellenállásokkal.



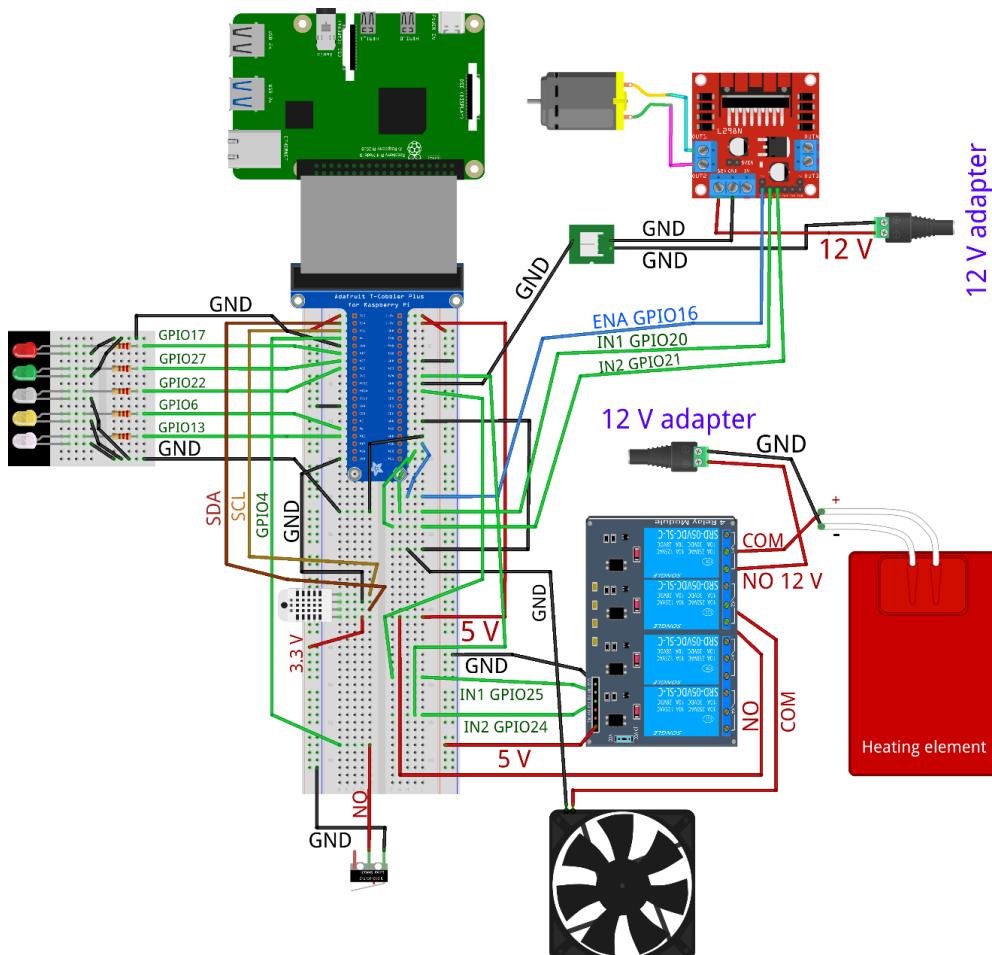
18. ábra: LED panel és Raspberry Pi kapcsolási rajz

# 3. Hardware összeállítása

A következő fejezetben a csibekeltető rendszer hardveres összeállítását mutatom be, amely a Raspberry Pi 4B központi vezérlőegység köré épül. Röviden ismertetem a rendszer kapcsolási rajzát, majd bemutatom az egyes elemek összeszerelésének és csatlakoztatásának folyamatát. Kitértem arra is, hogy a Raspberry Pi miként képes vezérelni és szabályozni a különböző komponenseket a GPIO portokon keresztül. Végül bemutatom, hogyan ellenőriztem és finomhangoltam a hardver és szoftver együttes működését.

## 3.1. Modulok összeszerelése

Az alábbi kapcsolási rajzon látható a Raspberry Pi 4B központi vezérlő és a hozzá kapcsolódó komponensek bekötési módja. A rajzon áttekinthető, hogyan csatlakoznak a GPIO pineken keresztül az egyes elemek, valamint hogyan biztosított a megfelelő feszültség és földelés. A kapcsolási rajz segítségével egyszerűen követhető a rendszer teljes hardveres felépítése.



19. ábra: Hardverek és Raspberry Pi kapcsolási rajz

## 3.2. Fejlesztői környezet

A fejlesztés során a Visual Studio Code<sup>13</sup> (VSCode) integrált fejlesztői környezetet (IDE) használtam. A VSCode egy népszerű, nyílt forráskódú szerkesztő, amelyet a Microsoft fejlesztett ki. Legfőbb előnyei közé tartozik a könnyű bővíthetőség, a felhasználóbarát kezelőfelület és a különféle programnyelvekhez nyújtott támogatás, beleértve a Python-t is, amellyel a Raspberry Pi-n futó programot készítettem. Mivel a Raspberry Pi-n futó operációs rendszer Linux-alapú, VSCode segítségével SSH-kapcsolaton keresztül közvetlenül csatlakoztam a Raspberry Pi-re, így a kódírást és a tesztelést egyszerre tudtam végezni anélkül, hogy külön fájlátvitelt kellett volna alkalmaznom. Az SSH<sup>14</sup> (Secure Shell) kapcsolat biztonságos, titkosított kommunikációt biztosított a számítógépem és a Raspberry Pi között, lehetővé téve a kód valós idejű szerkesztését, hibakeresését és futtatását. Ez a munkafolyamat jelentősen felgyorsította és leegyszerűsítette a fejlesztési ciklust, hozzájárulva a hatékonyabb munkavégzéshez.

### 3.2.1. Fejlesztői környezet beállítása

A fejlesztői környezet beállítása során telepítettem a Visual Studio Code (VSCode) integrált fejlesztői környezetet a számítógépemre, majd az SSH kiegészítővel közvetlen kapcsolatot hoztam létre a Raspberry Pi-vel, melyre jelszavas hitelesítéssel jelentkeztem be. A Pi-hez statikus IP-címet állítottam be, így nem kellett újra konfigurálnom minden egyes csatlakozáskor. A Raspberry Pi operációs rendszerén telepítettem a Python programozási nyelvet, valamint a Flask webes keretrendszerét, az RPi.GPIO könyvtárat, a requests modult API kommunikációhoz és a python-dotenv csomagot környezeti változók kezelésére. Ezekkel a csomagokkal egy stabil és hatékony fejlesztői környezetet alakítottam ki, ami jelentősen leegyszerűsítette a csibekeltető vezérlőszoftver fejlesztését.

## 3.3. Raspberry Pi programozás

Sikerült beállítani az IDE-t, így el lehet kezdeni a fejlesztést. A Raspberry Pi programozásához Python nyelvet használtam, és minden egyes hardveres komponenshez külön programokat készítettem a könnyebb hibakeresés és karbantarthatóság érdekében. Első

---

<sup>13</sup> Bővebb információk az Irodalomjegyzékben találhatók. [3]

<sup>14</sup> Bővebb információk az Irodalomjegyzékben találhatók. [4]

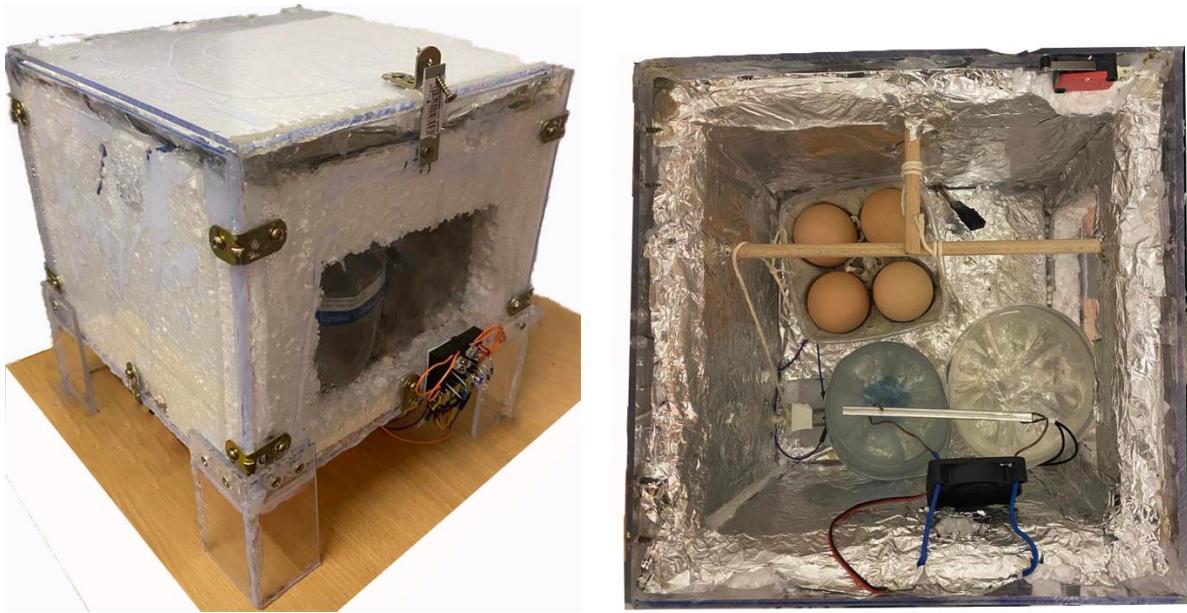
lépésként importáltam a szükséges Python-csomagokat, például az *RPi.GPIO*-t, amellyel a Raspberry Pi GPIO-pineket tudtam vezérelni. A GPIO pinek címét egy külön *csibekelteto\_utils.py* fájlban definiáltam, így minden egyes programban egyszerűen, kulcs alapján tudom beimportálni és használni a megfelelő pineket. Az egyes programokban először beállítottam a GPIO üzemmódot *GPIO.setmode(GPIO.BCM)*, majd az adott eszközhöz kapcsolódó pineket inicializáltam, általában kimenetként *GPIO.setup(pin, GPIO.OUT)*. A hőmérséklet- és páratartalom-érzékelő beolvasását végző program fájlba menti az aktuális értékeket, és ezek alapján egy másik program vezérli a fűtőelemet: ha a hőmérséklet nem éri el a kívánt szintet, akkor a fűtés bekapcsol, majd, ha megfelelő a hőmérséklet, automatikusan kikapcsol. A ventilátor vezérlése egy külön fájlban történik, ez a program folyamatosan működteti a ventilátort, amint elindul. A DC motor forgatja a tojásokat, és ezt egy időzítőre épülő külön program oldja meg, amely 6 óránként bekapcsolja a motort egy meghatározott időre, majd ismét kikapcsolja. A LED-ek működését szintén külön fájl kezeli, ezek a visszajelző fények különböző állapotokat jeleznek (például megy-e a fűtés, aktív-e a szenzor stb.), de nem befolyásolják a többi program működését.

Annak érdekében, hogy a Raspberry Pi biztonságosan leálljon, minden program végén egy biztonságos kilépést biztosítottam egy Python *try-except-finally* szerkezzettel. Ennek segítségével, ha bármilyen ok miatt megszakadna a program futása, a GPIO pineket automatikusan visszaállítom alaphelyzetükbe a *GPIO.cleanup()* függvény segítségével. A kész programokat alaposan teszteltem, és ellenőriztem, hogy minden hardverkomponens megfelelően reagál-e a vezérlésre.

### 3.4. Csibekeltető

A következő fejezetben a saját tervezésű és kivitelezésű csibekeltető berendezés kerül bemutatásra, melynek célja a tojások számára optimális körülmények biztosítása a teljes keltetési folyamat során. A projekt fő célkitűzése egy olyan automatizált rendszer létrehozása volt, amely valós időben képes érzékelni és szabályozni a környezeti tényezőket – mint például a hőmérsékletet és a páratartalmat – a keltetés sikeresége érdekében. A fejezet részletesen ismerteti a felhasznált hardverelemeket, a vezérlőlogika működését, valamint a berendezés fizikai felépítését, beleértve a szigetelést és a komponensek elrendezését is.

Az alábbi képen (20. ábra) a keltető külső és belső nézete látható. A felülnézeti képen jól megfigyelhető a keltetés folyamata, a tojásokat tartó forgatómechanika, továbbá a hőszigetelt belső tér és az elektronikusan vezérelt egységek elhelyezkedése.



20. ábra: Csibekeltető

A keltető dobozomat teljes egészében 4 mm vastag Gutta GGLISS hobbyüvegből (plexi)<sup>15</sup> építettem meg, hogy a belső teret könnyen megfigyelhessem, és elegendő fény jusson be a tojásokhoz. A dobozom 25 cm széles, 25 cm hosszú és 20 cm magas(lábakkal 28 cm), belső mérete tökéletesen megfelelt a kisebb létszámu tojások keltetéséhez. A plexi felületet egyszerű volt tisztán tartani, és jól bírta a nedvességet, így praktikusnak bizonyult az építés során.

Az oldalfalak belső részét 2 cm vastag expandált polisztirol<sup>16</sup> lapokkal szigeteltem le, hogy biztosítsam a megfelelő hőtartást. Nagyon figyeltem rá, hogy ezek a lapok szorosan illeszkedjenek, és ne keletkezzenek hőhidak, mivel ez kulcsfontosságú a stabil keltetési hőmérséklet szempontjából.

A plexi lapokat precízen vágtam és illesztettem egymáshoz, így lémmentesen záródott a doboz. A fedél szintén plexiből készült, és úgy alakítottam ki, hogy könnyen nyitható legyen. Ezáltal gyorsan hozzáértem a tojásokhoz, és ha szükséges volt, egyszerűen szellőztetni tudtam a keltetőt.

<sup>15</sup> Bővebb információk az Irodalomjegyzékben találhatók. [5]

<sup>16</sup> Bővebb információk az Irodalomjegyzékben találhatók. [6]

Ezzel a megoldással végül egy jól átlátható, jól szigetelt és egyszerűen tisztán tartható keltetőt hoztam létre, amely teljes mértékben bevált a gyakorlati próbák során.

A következő táblázat (21. ábra) a csirke keltetésének napokra bontott optimális környezeti feltételeit mutatja be, beleértve a hőmérsékletet, a párataartalmat, a szükséges hűtési időket, valamint a tojásforgatás gyakoriságát és módját<sup>17</sup>.

Nap	Hőmérséklet	Párataartalom	Hűtés	Forgatás	
				kézi	auto.
1	38,1 °C	55-65 %	nincs	nincs	2 óra
2	38,1 °C	55-65 %	nincs	nincs	2 óra
3	38,1 °C	55-65 %	nincs	nincs	2 óra
4	37,8 °C	55-65 %	5 perc	R/E	2 óra
5	37,8 °C	55-65 %	5 perc	R/E	2 óra
6	37,8 °C	55-65 %	5 perc	R/E	2 óra
7	37,8 °C	55-65 %	5 perc	R/E	2 óra
8	37,8 °C	55-65 %	10 perc	R/E	2 óra
9	37,8 °C	55-65 %	10 perc	R/E	2 óra
10	37,8 °C	55-65 %	10 perc	R/E	2 óra
11	37,5 °C	55-65 %	10 perc	R/E	2 óra
12	37,5 °C	55-65 %	15 perc	R/E	2 óra
13	37,5 °C	55-65 %	15 perc	R/E	2 óra
14	37,5 °C	55-65 %	15 perc	R/E	2 óra
15	37,5 °C	55-65 %	20 perc	R/E	2 óra
16	37,5 °C	55-65 %	20 perc	R/E	2 óra
17	37,5 °C	55-65 %	20 perc	R/E	2 óra
18	37,5 °C	55-65 %	20 perc	R/E	2 óra
19	37,2 °C	70-75 %	nincs	nincs	
20	37,2 °C	70-75 %	nincs	nincs	
21	37,2 °C	70-75 %	nincs	nincs	

21. ábra: Csirke keltetés táblázat

### 3.4.1. Hőmérséklet

A hőmérséklet a mesterséges keltetés egyik legkritikusabb paramétere, mivel közvetlen hatással van az embrió fejlődési ütemére és a kikelési arányra. A legtöbb baromfifajnál – így a tyúknál is – az optimális keltetési hőmérséklet 37,5 °C körül alakul. Ezt az értéket a keltetés teljes időtartama alatt lehetőség szerint stabilan kell tartani, mivel már néhány tized fokos eltérés is fejlődési rendellenességekhez, illetve a kelési arány csökkenéséhez vezethet. A túl magas hőmérséklet a fejlődés felgyorsulását, de egyben a halálozási arány növekedését is okozhatja, míg az alacsony hőmérséklet lassabb fejlőést, esetenként kikelési késedelmet eredményezhet.

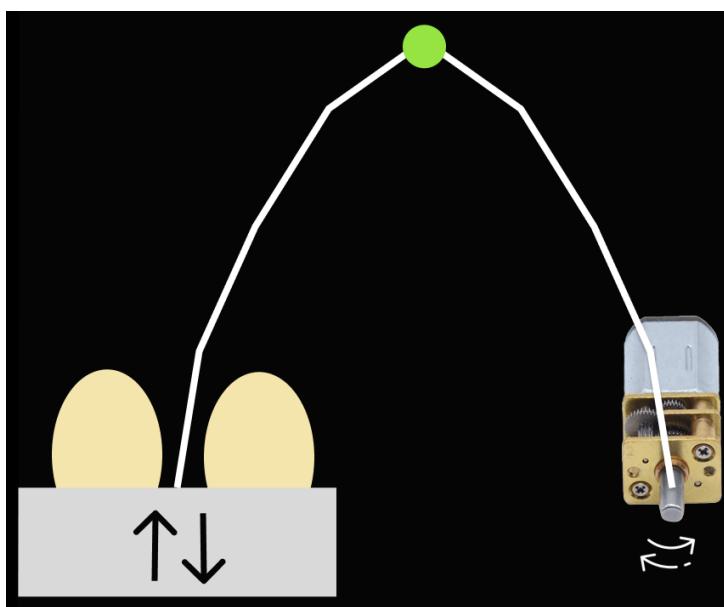
<sup>17</sup> Bővebb információk az Irodalomjegyzékben találhatók. [7]

### 3.4.2. Páratartalom

A páratartalom szabályozása a keltetés egyik alapvető tényezője, mivel befolyásolja a tojásban lévő folyadék párolgását, és ezáltal az embrió megfelelő fejlődését. A túl alacsony páratartalom túlzott nedvességesztéshez, a tojás belsejének kiszáradásához, és a csibe fejlődésének károsodásához vezethet. Ezzel szemben a túl magas páratartalom megakadályozhatja a megfelelő párolgást, ami oxigéniányt okozhat, illetve megnehezíti a csibe kikelését a tojáshéjból. A keltetési időszak nagy részében az ideális relatív páratartalom általában 50–55% körül alakul, míg az utolsó három napban – a kelési szakaszban – ez az érték 65–70%-ra emelkedik.

A szükséges páratartalmat a keltető belsejében elhelyezett víztartály biztosítja, melyből a párolgás révén jut a levegőbe a megfelelő mennyiségi nedvesség.

### 3.4.3. Tojásforgatási mechanizmus



22. ábra: Tojásforgatási mechanizmus

A mesterséges keltetés során elengedhetetlen a tojások rendszeres forgatása, amely elősegíti az embrió egyenletes fejlődését, valamint megelőzi, hogy a fejlődő csibe a tojáshéjhoz tapadjon. Ezt a megállapítást az amerikai Oklahoma State University mezőgazdasági tanszéke is alátámasztja, mely szerint a tojások napi többszöri forgatása elengedhetetlen a sikeres keltetéshez (Oklahoma State University Extension, 2023)<sup>18</sup>.

<sup>18</sup> Bővebb információk az Irodalomjegyzékben találhatók. [8]

A jelen dolgozatban bemutatott keltetőgép esetében egy egyszerű, mechanikus megoldás került alkalmazásra a tojások mozgatására (22. ábra). A rendszer lelke egy 12 V-os egyenáramú motor, amely egy madzagot húz meg. Ez a madzag egy fa darabon keresztül van elvezetve, majd rögzítésre kerül a tojásokat tartalmazó dobozhoz. A motor elindításakor a madzag megfeszül, így a fa elem közvetítésével a doboz finoman megemelkedik vagy süllyed. Ez a felle irányú mozgás elegendő ahhoz, hogy a tojások helyzete megváltozzon, biztosítva a megfelelő és rendszeres forgatást. A rendszer előnye, hogy egyszerűen kivitelezhető, megbízható, és nem igényel bonyolult mechanikai alkatrészeket.

#### 3.4.4. Szellőzés

A megfelelő szellőzés alapvető fontosságú a keltetés során, mivel biztosítja az embriók oxigénellátását, valamint segíti a szén-dioxid eltávolítását a zárt térből. A friss levegő folyamatos utánpótlása nélkül a tojásokban fejlődő csibék oxigéniányos állapotba kerülhetnek, ami fejlődési rendellenességekhez vagy akár az embrió pusztulásához is vezethet. A dolgozatban bemutatott keltetőgépen a szellőzést egy 5 V-os ventilátor biztosítja, amely a levegő áramoltatásával egyenletes hő- és párvízszonyokat teremt a belső térben. A ventilátor mögött kialakított szellőzőrések lehetővé teszik a friss levegő bejutását a doboz belsejébe, így biztosítva a megfelelő légcserét a keltetés teljes időtartama alatt.

#### 3.4.5. Áramfogyasztás és költségek

Ebben a fejezetben kiszámítom, hogy a csibekeltető rendszer az inkubáció 21 napja alatt összesen mennyi elektromos energiát fogyaszt. Részletesen bemutatom az egyes alkatrészek – a fűtőelem, a ventilátor és a motor – energiaigényét és működési idejét. A számítások alapján meghatározom a teljes fogyasztást kilowattórában, valamint ennek várható költségét forintban.

##### Fűtőelem:

Elsőként meghatározom a PTC (Positive Temperature Coefficient) fűtőelem teljesítményét:

$$P = U * I = 12 V * 2 A = 24 W^{19}$$

Ez azt jelenti, hogy a fűtőelem folyamatosan 24 watt teljesítményt ad le hő formájában.

- Bekapcsolási ciklus: 2 perc BE / 2 perc KI, tehát az idő 50%-ában működik.
- Napi működési idő:  $1440 \text{ perc} \times 50\% = 720 \text{ perc} = 12 \text{ óra}$
- Energia:  $0.024 \text{ kW} \times 12 \text{ h} = 0.288 \text{ kWh}$
- Költség:  $0.288 \text{ kWh} \times 70 \text{ Ft} = 20,16 \text{ Ft / nap}$

<sup>19</sup> Bővebb információk az Irodalomjegyzékben találhatók. [9]

### **Ventilátor:**

A ventilátor teljesítménye:

$$P = U * I = 5 V * 0.2 A = 1 W$$

Ez alapján a ventilátor folyamatosan 1 wattot, azaz 0.001 kilowattot fogyaszt a keltető szellőztetésére.

- Napi idő: 24 óra
- Energiafelhasználás:  $0.001 \text{ kW} \times 24 \text{ h} = 0.024 \text{ kWh}$
- Költség:  $0.024 \text{ kWh} \times 70 \text{ Ft} = 1,68 \text{ Ft / nap}$

### **Hőmérséklet és páratartalom szenzor:**

Az érzékelő energiafogyasztása és költsége elhanyagolható, de a számítás kedvéért az alábbi értékeket vesszük figyelembe:

$$P = U * I = 5 V * 0.001 A = 0.005 W = 0.000005 kW$$

- Energia:  $0.000005 \text{ kW} \times 24 = 0.00012 \text{ kWh}$
- Költség:  $0.00012 \text{ kWh} \times 70 \text{ Ft} \approx 0.0084 \text{ Ft / nap}$

### **DC motor:**

A tojásforgatás 6 óránként történik, alkalmanként 1 percig, így naponta összesen 4 percig működik.

$$P = U * I = 12 V * 1 A = 12 W = 0.012 kW$$

- Napi működési idő: 4 perc = 0.0667 óra
- Energiafogyasztás:  $0.012 \text{ kW} \times 0.0667 \text{ h} = 0.0008 \text{ kWh}$
- Költség:  $0.0008 \text{ kWh} \times 70 \text{ Ft} = 0.056 \text{ Ft / nap}$

### **Mikrokapsoló:**

A mikrokapsoló energiafogyasztása és költsége szintén elhanyagolható, mivel csak jeladóként funkcionál, és minimális áramot igényel.

### **Raspberry Pi 4B:**

A Raspberry Pi 4B 50%-os átlagos terheléssel működik. Az adapttere 5V / 3A kimenetet biztosít, ennek megfelelően:

$$P = U * I = 5V * 3A = 15W$$

- 50% terhelés:  $15 \text{ W} * 50\% = 7.5 \text{ W} = 0.0075 \text{ kW}$
- Napi fogyasztás:  $0.0075 \text{ kW} \times 24 \text{ h} = 0.18 \text{ kWh}$
- Költség:  $0.18 \text{ kWh} \times 70 \text{ Ft} = 12.6 \text{ Ft / nap}$

A fenti számítások alapján a rendszer napi energiafogyasztását 23. ábra, a 21 napos inkubációs ciklusra vetített fogyasztást pedig a 24. ábra szemlélteti.

### **1 nap:**

Eszköz	Fogyasztás (kWh)	Költség (Ft)
Fűtőelem	0.288	20.16
Ventilátor	0.024	1.68
Motor	0.0008	0.056
Raspberry Pi (50%)	0.18	12.6
Hő/pára szenzor	0.0001	0.0084
Összesen	0.4929	34.5044

23. ábra: A rendszer egy napi energiafogyasztása és költsége

### **21 nap:**

Eszköz	Fogyasztás (kWh)	Költség (Ft)
Fűtőelem	6.048	423.36
Ventilátor	0.504	35.28
Motor	0.0168	1.176
Raspberry Pi (50%)	3.78	264.6
Hő/pára szenzor	0.0025	0.1764
Összesen	10.3513	724.5924

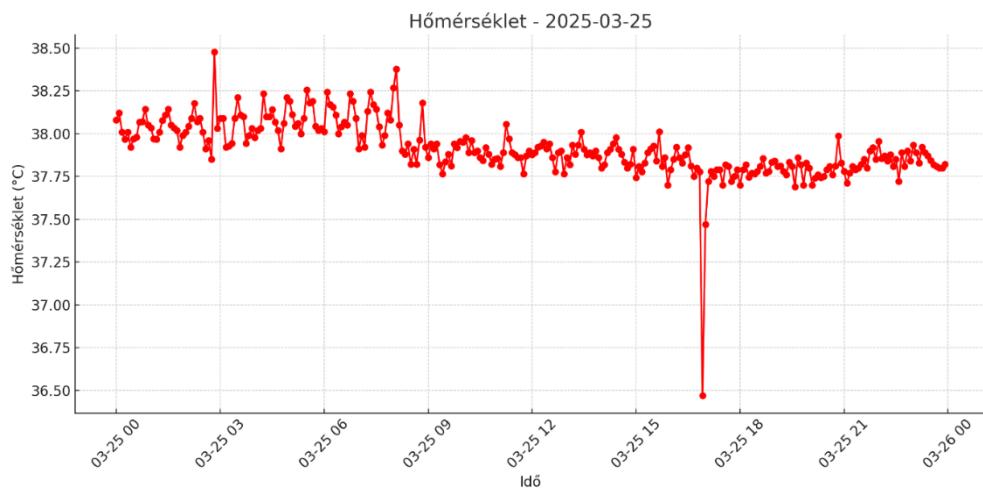
24. ábra: A rendszer 21 napos (teljes inkubációs idő) energiafogyasztása és költsége

## **3.5. Csibekeltetés**

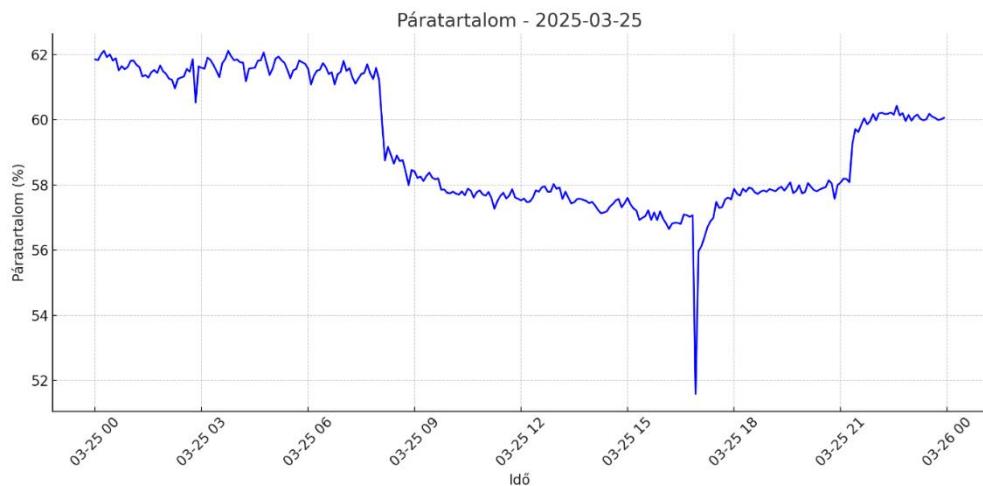
A keltetési folyamatot 2025. március 24. és április 15. között hajtottam végre. Az inkubátorba összesen négy tojást helyeztem el, és a teljes keltetési időszak alatt folyamatosan figyelemmel kísértem a hőmérsékletet és a páratartalmat. A megtermékenyülést követő napokban a tojásokon belüli fejlődés nyomon követhető volt. Két tojás esetében egyértelműen látható volt az embrió mozgása, a testük körvonala és időnként még a kis lábaik is felismerhetők voltak. A fejlődés során ezek az embriák aktívnak tűntek, ami bizakodásra adott okot. A keltetési idő vége felé azonban ezek a mozgások fokozatosan abbamaradtak. A 21 nap letelte után egyik csibe sem bújt ki a tojásból. A keltetés eredménye így végül sikertelennek bizonyult, mivel bár két embrió fejlődött, egyik sem tudott kikelni.

### 3.5.1. Statisztika

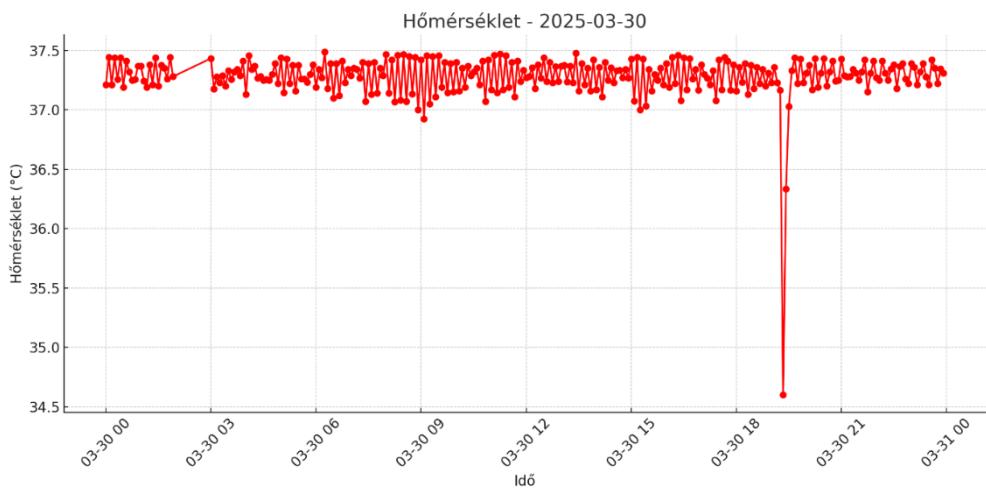
A szakdolgozat részeként statisztikai diagramokat készítettem a keltetés ideje alatt mért hőmérséklet és páratartalom értékekről. A mért adatokat négy külön napra bontva, 5 percenkénti átlagolással ábrázoltam, külön diagramokon megjelenítve a hőmérsékletet és a páratartalmat. Ezek a grafikonok szemléletesen mutatják a környezeti paraméterek alakulását a keltetési folyamat során.



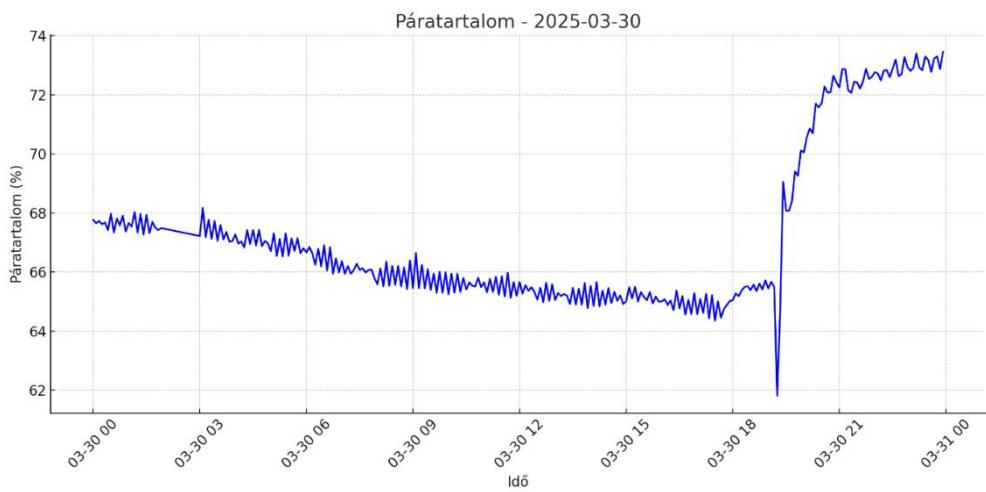
25. ábra: 2025 Március 25 hőmérséklet adatai



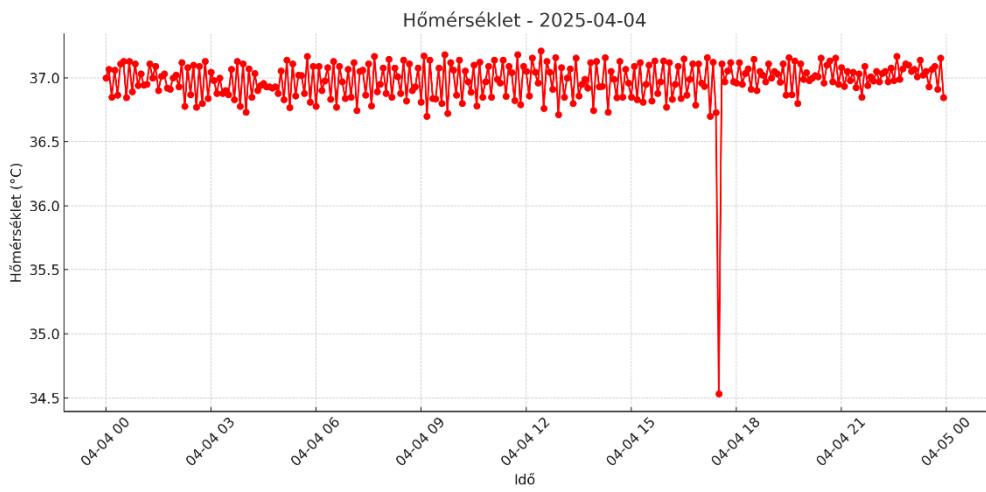
26. ábra: 2025 Március 25 páratartalom adatai



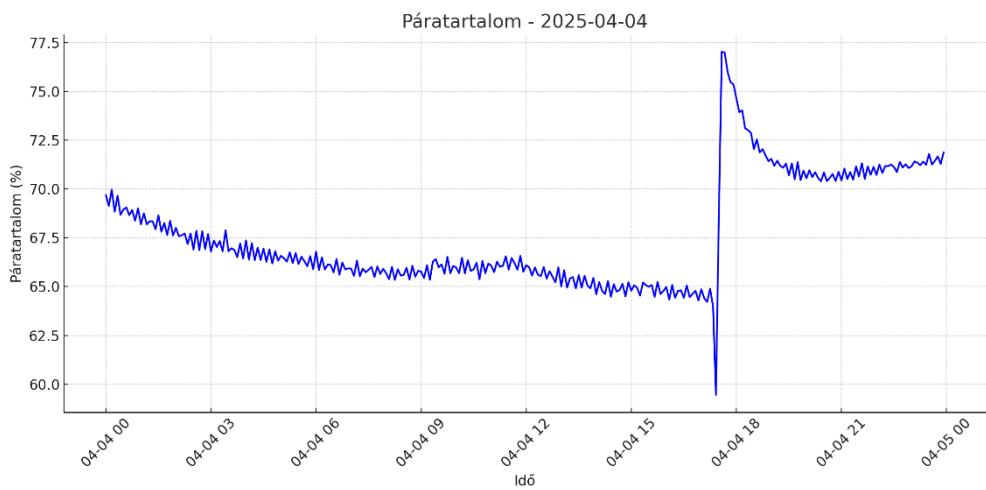
27. ábra: 2025 Március 30 hőmérséklet adatai



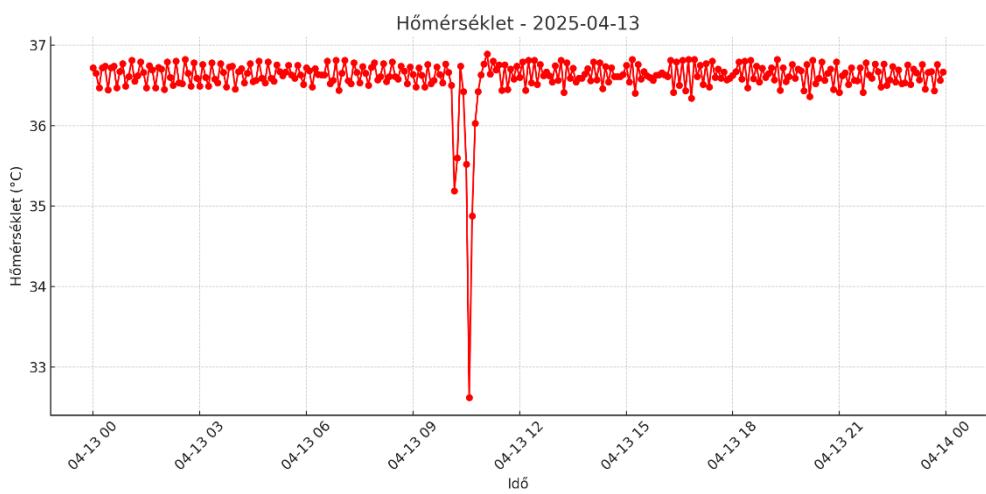
28. ábra: 2025 Március 30 páratartalom adatai



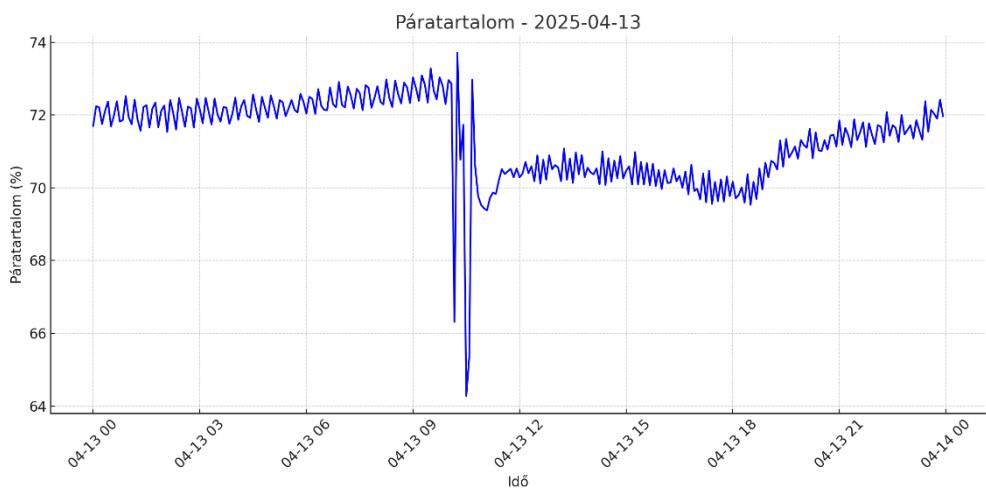
29. ábra: 2025 Április 4 hőmérséklet adatai



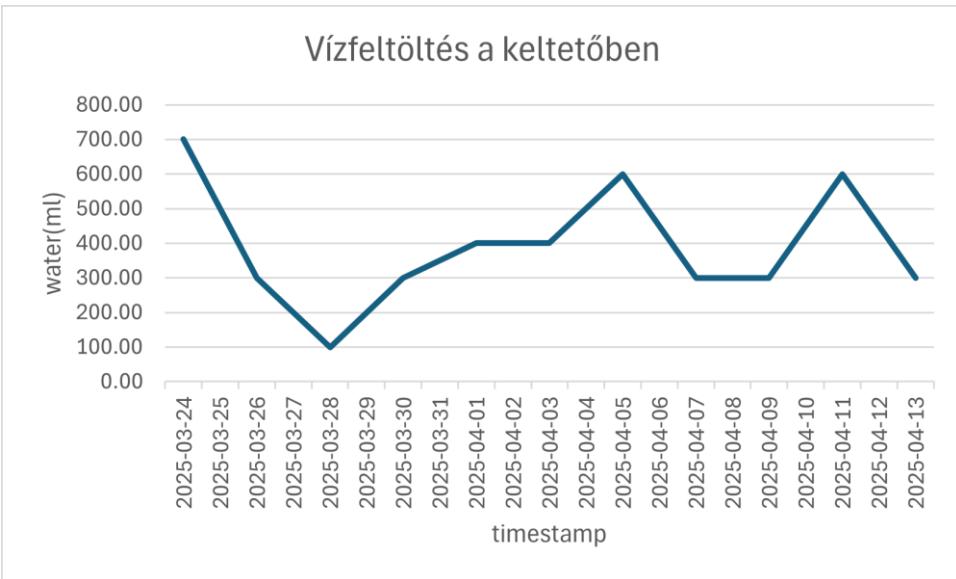
30. ábra: 2025 Április 4 páratartalom adatai



31. ábra: 2025 Április 13 hőmérséklet adatai



32. ábra: 2025 Április 13 páratartalom adatai



33. ábra: A keltetés során alkalmanként betöltött vízmennyiségek

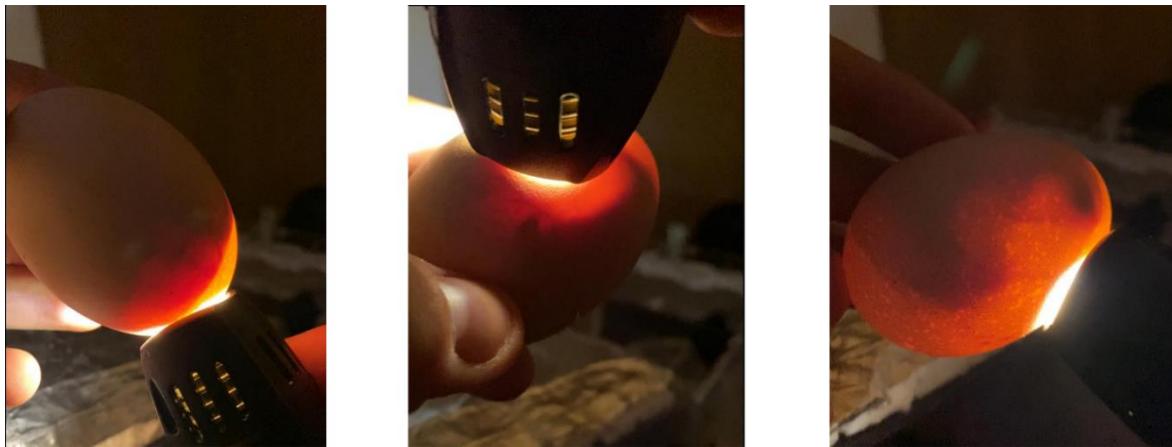
A diagramokon látható hirtelen ugrások a hőmérsékletben és páratartalomban abból adódnak, hogy ebben az időpontban nyitottam ki a keltető tetejét. A tető levételekor a belső környezet hőmérséklete gyorsan lecsökkent, a páratartalom pedig ingadozott a külső levegő beáramlása és a víz utántöltése miatt. A víz betöltésével a páratartalom kezdetben gyorsan csökkent, majd visszaállt a kívánt értékre, miközben a hőmérséklet is hamar stabilizálódott. Ezek az ugrások tehát nem rendszerhibák, hanem a normális karbantartási folyamat eredményei.

A teljes keltetési folyamat alatt összesen 4300 ml vizet használtam fel. Ez a mennyiség biztosította a szükséges páratartalom folyamatos fenntartását a keltető belsejében.

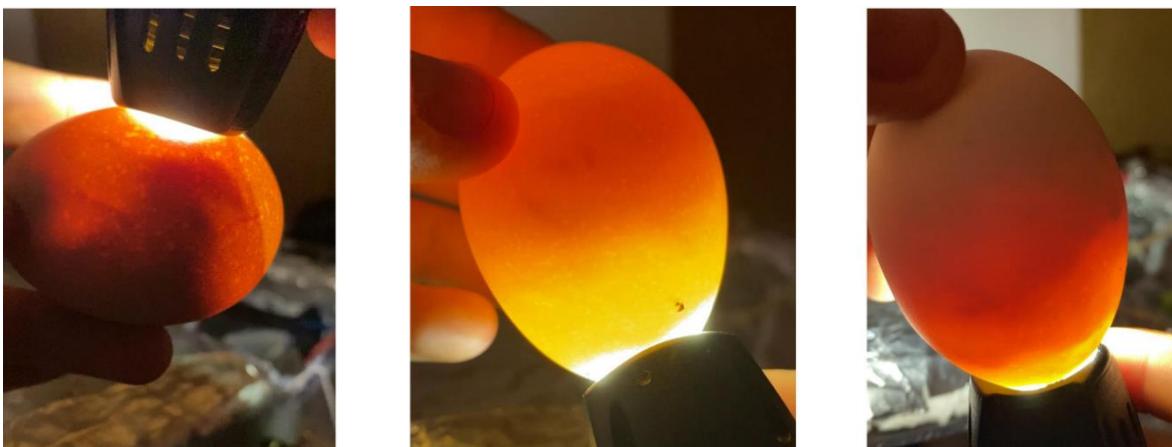
### 3.5.2. Csibekeltetés összefoglalva

A keltetés során egyetlen tojás sem kelt ki, azonban két tojásban megindult az embrió fejlődése, míg a többi tojásban egyáltalán nem volt tapasztalható fejlődés. A sikertelenség okai között szerepelhettek a tojások esetleges terméketlensége vagy nem megfelelő minősége, illetve a keltető működéséből adódó hőmérséklet-ingadozások is. A keltetőben használt relé időnkénti hibás működése miatt tapasztalt hőmérsékleti eltérések szintén negatívan befolyásolhatták a keltetés eredményét. A jövőbeni fejlesztések között szerepelhet a fűtőelemet vezérlő relé stabilabbra cserélése és a hatékonyabb szellőzőrendszer kialakítása, ezzel javítva a belső klíma stabilitását. Összességében a keltető berendezés és én magam is minden tölünk telhetőt megtettünk annak érdekében, hogy sikeres legyen a keltetés.

A képek a tojások állapotát mutatják a keltetési folyamat végén, látható rajtuk a fejlődésnek indult (34. ábra), illetve a fejlődést nem mutató tojások is (35. ábra).

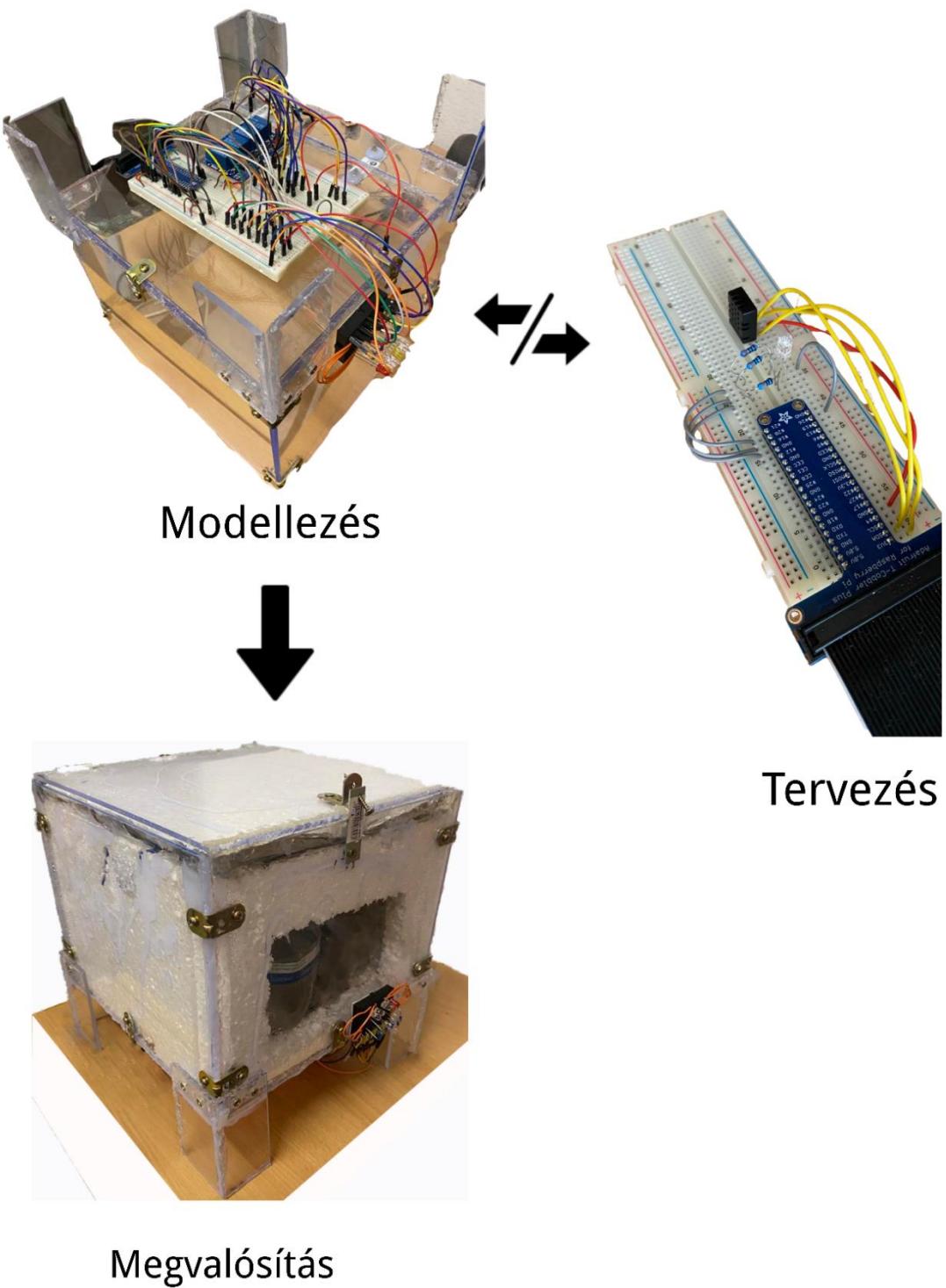


34. ábra: Fejlődésnek indulto tojások



35. ábra: Egy tojásban fejlődésnek indulto csibe és másik kettő fejlődést nem mutató tojások

### 3.6. Elkészülés folyamata



36. ábra: Elkészülés folyamata

# 4. Software összeállítása

A csibekeltető rendszer működésének egyik alapvető eleme a szoftveres vezérlés, amely több komponens együttműködésén alapul. A Raspberry Pi 4B egy Python nyelven írt vezérlőprogramot futtat, amely kezeli a különböző hardverelemekeket, mint a fűtőelem, ventilátor, szenzor, LED-ek és a tojásforgató motor. A Raspberry Pi ezen felül egy Flask alapú API-t is biztosít, amely lehetővé teszi a rendszer interneten keresztüli elérését. A rendszerhez tartozik egy Android alkalmazás is, amely ezen API-n keresztül kommunikál a Raspberry Pi-vel.

Az alkalmazás Kotlin nyelven, Jetpack Compose keretrendszerrel készült, amely modern és deklaratív megközelítést alkalmaz a felhasználói felület kialakításához. Az alkalmazás valós idejű hőmérséklet- és páratartalom-adatokat jelenít meg, grafikonos formában is, továbbá lehetőséget nyújt a felhasználónak a hőmérséklet beállítására és a hardverelemekek vezérlésére. A tojásforgatás is elindítható manuálisan az alkalmazásból. A szoftver kialakításánál fontos szempont volt a megbízhatóság, a biztonságos adatkezelés, valamint a könnyen használható, felhasználóbarát kezelőfelület kialakítása.

## 4.1. API

Az API-t Python nyelven írtam Flask keretrendszerrel, és a Raspberry Pi-n fut, ahol a hardverek vezérlését, valamint az adatok továbbítását végzi az Android alkalmazás felé. Az alkalmazás ezen az API-n keresztül kommunikál a rendszerrel, a kérések és válaszok JSON formátumban történnek. Az API végpontjai lehetővé teszik a hőmérséklet- és páratartalom-értékek lekérdezését, valamint a fűtőelem, ventilátor, LED-ek és a tojásforgató motor ki- és bekapcsolását. Az Android alkalmazás ezeken a végpontokon keresztül jeleníti meg az aktuális szenzoradatokat, és biztosít lehetőséget a felhasználónak a rendszer távoli vezérlésére. Az API kialakításánál figyeltem arra, hogy a végpontok logikusan strukturáltak és jól dokumentáltak legyenek, hogy a későbbi fejlesztések is egyszerűen megvalósíthatók legyenek. Annak érdekében, hogy az API ne csak helyi hálózaton, hanem globálisan is elérhető legyen, az ngrok szolgáltatást használtam<sup>20</sup>, amely egy statikus domain biztosít számomra, és ezen keresztül kapja meg az API a kéréseket. A kommunikáció így stabil internetkapcsolaton keresztül történik, a szoftver moduláris felépítése pedig egyszerű bővíthetőséget, könnyű karbantartást

---

<sup>20</sup> Bővebb információk az Irodalomjegyzékben találhatók. [10]

és valós idejű működést tesz lehetővé, ami különösen fontos a keltetési folyamat pontosságára szempontjából.

Az alábbiakban néhány példa segítségével bemutatom, hogyan működnek a lekérdezések az API-n keresztül, és hogy milyen választ kapunk ezekre — a példákban a *curl*<sup>21</sup> parancs segítségével hívom meg az egyes végpontokat, és megmutatom a válaszként érkező JSON formátumú adatokat is.

<https://harmless-toad-remarkably.ngrok-free.app/get-day>

- Funkció: Ez az API-végpont adja meg, hány napja tart a keltetés.
- Működés: A rendszer egy fájlból kiolvassa a keltetés kezdődátumát, majd ezt összeveti a mai nappal, és kiszámolja a különbséget napokban. Ez segít a felhasználónak nyomon követni, éppen hányadik napnál tart a folyamat, ami fontos a keltetés időzítéséhez és ellenőrzéséhez.
- Válasz: A végpont a „*response*” (37. ábra) változóban adja meg, hány napja tart a keltetés.



```
curl -X GET "https://harmless-toad-remarkably.ngrok-free.app/get-day"
{"other":null,"response":"13","status_code":200,"timestamp":"Sun, 06 Apr 2025 17:23:41 GMT"}
```

37. ábra: Curl parancs, amely lekérdezi, hány napja tart a keltetés

<https://harmless-toad-remarkably.ngrok-free.app/led-indication?val=on>

- Funkció: Ez az API-végpont lehetővé teszi a LED panel ki- és bekapcsolását
- Működés: A lényege, hogy a rendszer egy fájlban tárolja a LED állapotát, amelyben az érték vagy on, vagy off. Amikor az API-n keresztül érkezik egy lekérdezés — például „*val=on*”, a program ezt az értéket elmenti a fájlba, majd ennek megfelelően kapcsolja be vagy ki a LED panelt.
- Válasz: A végpont egy egyszerű választ ad, a „*response*” (38. ábra) változóban pedig visszaadja a beállított állapotot.



```
curl -X GET "https://harmless-toad-remarkably.ngrok-free.app/led-indication?val=on"
{"other":null,"response":"led is on","status_code":200,"timestamp":"Sun, 06 Apr 2025 17:25:15 GMT"}
```

38. ábra: Curl parancs, amely bekapcsolja a LED panelt

<sup>21</sup> Bővebb információk az Irodalomjegyzékben találhatók. [11]

<https://harmless-toad-remarkably.ngrok-free.app/on-cooler>

- Funkció: Ez az API-végpont lehetővé teszi a ventilátor bekapcsolását.
- Működés: A működése során a rendszer először ellenőrzi, hogy a ventilátor már be van-e kapcsolva, és ha még nem működik, akkor aktiválja azt.
- Válasz: Bekapcsolja a ventilátort, és a „*response*” 39. ábra) változóban egy szöveges üzenetet kapunk vissza, amely jelzi, hogy a ventilátort bekapcsoltuk.



```
curl -X GET "https://harmless-toad-remarkably.ngrok-free.app/on-cooler"  
{"other":null,"response":"cooler is on","status_code":200,"timestamp":"Sun, 06 Apr 2025 17:25:57 GMT"}
```

39. ábra: Curl parancs, amely bekapcsolja a ventilátort

## 4.2. Csibekeltető mobil applikáció



40. ábra: A csibekeltető applikációjának ikonja<sup>22</sup>

A csibekeltető rendszeréhez egy Android alkalmazás is készült, amely Kotlin nyelven, Jetpack Compose<sup>23</sup> használatával valósult meg. Az applikáció célja, hogy felhasználóbarát módon biztosítson hozzáférést a keltető működéséhez. A mobilalkalmazás lehetővé teszi az eszköz állapotának nyomon követését és távoli vezérlését. A fejlesztés során fontos szempont volt az egyszerű kezelhetőség és az azonnali visszajelzések megjelenítése. Ezzel a megoldással a felhasználók bárhonnan interakcióba léphetnek a rendszerrel, növelve ezzel a keltetési folyamat biztonságát és hatékonyságát.

<sup>22</sup> Kép forrása: <https://icons8.com/icon/81637/hatching-chicken>

<sup>23</sup> Bővebb információk az Irodalomjegyzékben találhatók. [12]

### **4.2.1. Android operációs rendszer**

Az Android egy nyílt forráskódú mobil operációs rendszer, amelyet a Google fejlesztett ki, és elsősorban érintőképernyős eszközökön, például okostelefonokon és táblagépeken használják. Világszerte az egyik legelterjedtebb mobilplatform, amelyre fejlesztők milliói készítenek alkalmazásokat. Az Android rendszer Java vagy Kotlin nyelven írt programokat futtat, és a felhasználói felület kialakításához modern eszközöket, például a Jetpack Compose-t is támogatja. Rugalmassága és testreszabhatósága miatt ideális választás olyan egyedi fejlesztésekhez is, mint például egy csibekeltetőt vezérlő alkalmazás.

### **4.2.2. Kotlin és Jetpack Compose**

A Kotlin és a Jetpack Compose választása nemcsak technikai szempontból volt előnyös, hanem szakmai fejlődésem szempontjából is tudatos döntés volt. Korábban nem dolgoztam ezekkel az eszközökkel, így kiváló lehetőséget jelentettek számomra az új technológiák elsajátítására. Szerettem volna kilépni a komfortzónámból, és egy számomra új megközelítést megtanulni a felhasználói felületek fejlesztésében. A projekt során így nemcsak egy működő alkalmazást készítettem, hanem értékes tapasztalatokkal is gazdagodtam.

#### **Kotlin<sup>24</sup>:**

A Kotlin egy modern, statikusan típusos programozási nyelv, amelyet a JetBrains fejlesztett ki. Kifejezetten Android-fejlesztésre optimalizálták, és hivatalosan támogatja a Google. Tiszta szintaxisa és fejlett nyelvi funkciói miatt gyorsan népszerűvé vált a fejlesztők körében.

#### **Jetpack**

#### **Compose:**

A Jetpack Compose a Google által fejlesztett deklaratív UI eszközkészlet Android alkalmazások felhasználói felületének egyszerűbb és hatékonyabb megvalósítására. Segítségével a felhasználói felület kódja olvashatóbb, modulárisabb és könnyebben karbantartható. A Compose lehetővé teszi, hogy a fejlesztők kevesebb kóddal valósítsanak meg interaktív és dinamikus felületeket. Mivel teljes mértékben Kotlin alapú, jól illeszkedik a modern Android fejlesztési ökoszisztémába. A Compose bevezetésével jelentősen csökkent a fejlesztési idő és növekedett a UI komponensek újrafelhasználhatósága.

---

<sup>24</sup> Bővebb információk az Irodalomjegyzékben találhatók. [13]

### 4.2.3. Android Studio fejlesztői környezet

Az Android Studio a Google által fejlesztett hivatalos integrált fejlesztői környezet (IDE) Android alkalmazások készítésére. Az IDE a JetBrains IntelliJ IDEA platformjára épül, és kifejezetten Android-alkalmazások fejlesztésére optimalizált eszközöket tartalmaz. A Kotlin nyelv és a Jetpack Compose teljes körű támogatásával modern, deklaratív felhasználói felületek is könnyedén létrehozhatók.

A legfontosabb funkciók közé tartoznak:

- Logcat: Valós idejű naplózási felület, amely a rendszerüzenetek és hibák megjelenítésére szolgál
- Android Virtual Device (AVD) Manager: Virtuális eszközök létrehozása és futtatása tesztelési célokra
- Gradle build rendszer: A projektfordítás, függőségkezelés és build konfigurációk alapja
- IntelliSense (kódkiegészítés): Gyors és intelligens javaslatok a kódírás során
- Verziókezelő támogatás (VCS): Beépített integráció például a Git vagy GitHub rendszerrel

### 4.2.4. API használata

A mobilalkalmazás és a Flask-alapú backend közötti kommunikációhoz a Retrofit könyvtárat használtam, amely egy típusbiztos HTTP kliens Androidhoz. A Retrofit segítségével az API-végpontokat annotációk segítségével lehet deklaratívan definiálni, ezáltal letisztult és karbantartható kódbázist eredményez. Az adatok feldolgozásához a Retrofit-et a GSON könyvtárral integráltam, amely automatikusan gondoskodik a JSON válaszok Kotlin adatosztályokká való deserializálásáról. Az API-hívások aszinkron végrehajtása coroutine-ökkel történt, így az alkalmazás nem blokkolja a fő szálat, ami elengedhetetlen a gördülékeny és reszponzív felhasználói élmény biztosításához.

A Retrofit és a GSON használatához a build.gradle (Module: app) fájlban hozzáadtam a szükséges függőségeket (41. ábra).



```
// build.gradle.kts (Module:app) file

implementation("com.squareup.retrofit2:retrofit:2.9.0") // <-- Retrofit alap könyvtár
implementation("com.squareup.retrofit2:converter-gson:2.9.0") // <-- GSON konverter Retrofithez
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.2") // <-- Coroutine Android
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.2") // <-- Coroutine alap
```

41. ábra: A Retrofit, GSON és Coroutine függőségek hozzáadása a build.gradle.kts fájlhoz

A mobilalkalmazás egyik fontos funkciója, hogy megjelenítse, hányadik napja tart a keltetés. Ezt az információt a Raspberry Pi-n futó backend szolgáltatja egy API-végponton keresztül, amely a <https://harmless-toad-remarkably.ngrok-free.app/get-day> URL alatt érhető el.

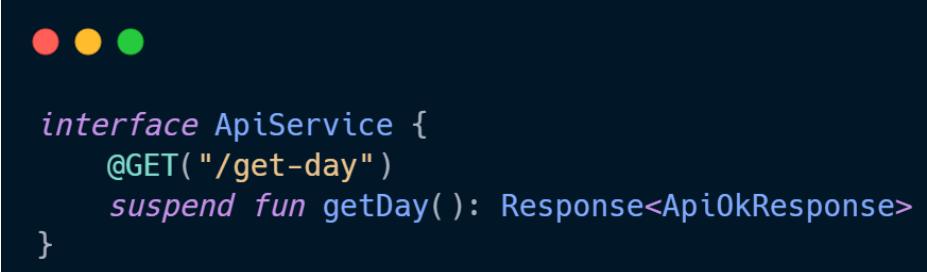
Első lépésként létrehoztam egy adatmodell osztályt az API válaszának kezelésére ApiOkResponse, amely a válasz kulcsait reprezentálja Kotlin adatszerkezzettel (42. ábra).



```
data class ApiOkResponse(
    val statusCode: Int = 0,
    val response: String = "",
    val timestamp: String = "",
    val other: List<String> = listOf()
)
```

42. ábra: Az API válaszát kezelő ApiOkResponse data class

Ezután létrehoztam az interfészét, amely definiálja a Retrofit végpontot és a suspend függvényt az adatok lekérésére (43. ábra).



```
interface ApiService {
    @GET("/get-day")
    suspend fun getDay(): Response<ApiOkResponse>
}
```

43. ábra: Retrofit interfész a keltetési nap lekérésére

A Retrofit példányt a RetrofitInstance objektumban inicializáltam, ahol megadtam a bázis URL-t és a JSON konvertert (44. ábra).



```
object RetrofitInstance {

    val api: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BaseUrl.API_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }
}
```

44. ábra: Retrofit példány létrehozása

A hálózati kérés végrehajtását egy általános célú `fetchApiResponse` függvénytel valósítottam meg, amely kezelni tudja a sikeres válaszokat és a különböző hibaforgatókönyveket is (45. ábra).



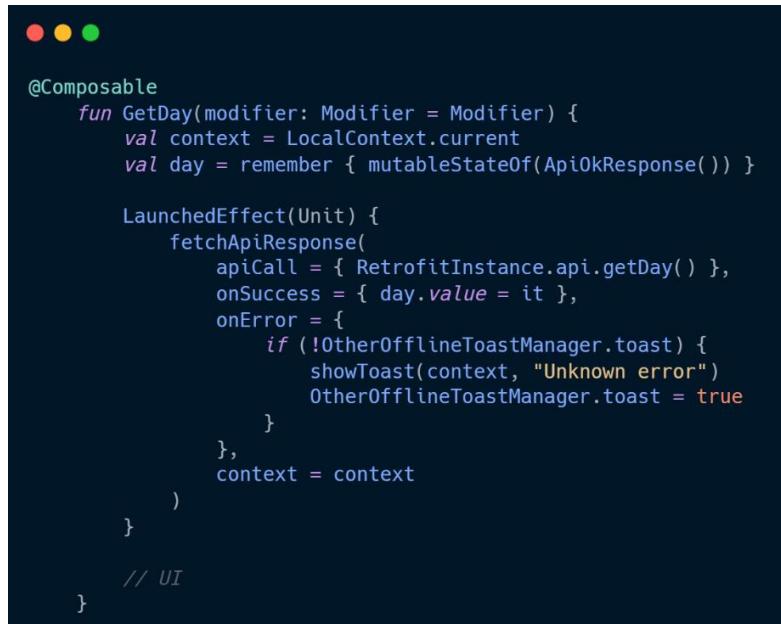
```

suspend fun <T> fetchApiResponse(
    apiCall: suspend () -> Response<T>,
    onSuccess: (T) -> Unit,
    onError: ((() -> Unit)? = null,
    context: Context
) {
    try {
        val response = apiCall()
        Log.d("API_CALL", "Response Code: ${response.code()}, Body: ${response.body()}")
        if (response.isSuccessful && response.body() != null) {
            withContext(Dispatchers.Main) {
                Log.d("API_SUCCESS", "Received data: ${response.body()}")
                onSuccess(response.body()!!)
            }
        } else {
            withContext(Dispatchers.Main) {
                Log.e("API_ERROR", "Error: ${response.errorBody()?.string()}")
                onError?.invoke()
                showToast(context, "Error fetching data")
            }
        }
    } catch (e: HttpException) {
        Log.e("API_ERROR", "HTTP Exception: ${e.message}")
        withContext(Dispatchers.Main) { showToast(context, "HTTP error") }
    } catch (e: Exception) {
        Log.e("API_ERROR", "General Exception: ${e.message}")
        withContext(Dispatchers.Main) { showToast(context, "Error occurred") }
    }
}

```

45. ábra: API-hívás lebonyolítása és hibakezelés coroutine-on belül

A tényleges adatlekérdezés a Jetpack Compose felhasználói felület egyik Composable függvényében történik a LaunchedEffect segítségével, így az API-hívás automatikusan lefut a komponens betöltődésekor (46. ábra).



```

@Composable
fun GetDay(modifier: Modifier = Modifier) {
    val context = LocalContext.current
    val day = remember { mutableStateOf(ApiOkResponse()) }

    LaunchedEffect(Unit) {
        fetchApiResponse(
            apiCall = { RetrofitInstance.api.getDay() },
            onSuccess = { day.value = it },
            onError = {
                if (!OtherOfflineToastManager.toast) {
                    showToast(context, "Unknown error")
                    OtherOfflineToastManager.toast = true
                }
            },
            context = context
        )
    }
    // UI
}

```

46. ábra: A `GetDay` Composable, amely meghívja az API-t és frissíti a felületet

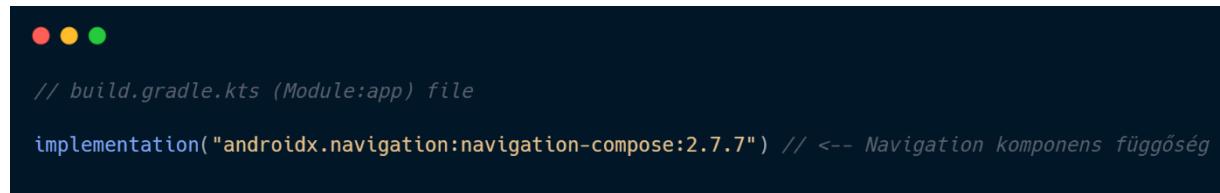
Ez a struktúra lehetővé teszi az aszinkron működést, a hibák kezelését, valamint azt is, hogy a felhasználói felület mindenkorán az aktuális információkat jelenítse meg, anélkül, hogy a fő szálat blokkolná.

#### 4.2.5. Navigáció

Az alkalmazás navigációs rendszerét a Jetpack Compose könyvtár Navigation komponense biztosítja. A különböző képernyők közötti átjárást egy NavHost-ba szervezett navigációs gráf valósítja meg, amelyhez egy NavController tartozik. A képernyők útvonalai deklaratív módon kerülnek definiálásra, így a navigáció logikája átlátható és könnyen karbantartható marad. Ez a megközelítés támogatja a modern, komponens-alapú Android fejlesztést, és jól illeszkedik a Compose alapú felhasználói felület struktúrájához.

Az alkalmazás három fő oldala a Home (Kezdőlap), a Control (Vezérlés) és a Settings (Beállítások), amelyek között a navigáció segítségével váltathat a felhasználó. A Home oldal az inkubátor aktuális állapotát mutatja, a Control oldalon az eszközök vezérlése történik, míg a Settings felületen általános információk és az alkalmazás nyelvi beállításai érhetők el.

Az alábbi ábrán látható, hogyan történik a Jetpack Compose Navigation könyvtár importálása a projekt build.gradle fájljába a szükséges függőség hozzáadásával.



```
// build.gradle.kts (Module:app) file
implementation("androidx.navigation:navigation-compose:2.7.7") // <-- Navigation komponens függőség
```

47. ábra: A Jetpack Compose Navigation komponens függőségének deklarálása a build.gradle.kts fájlban

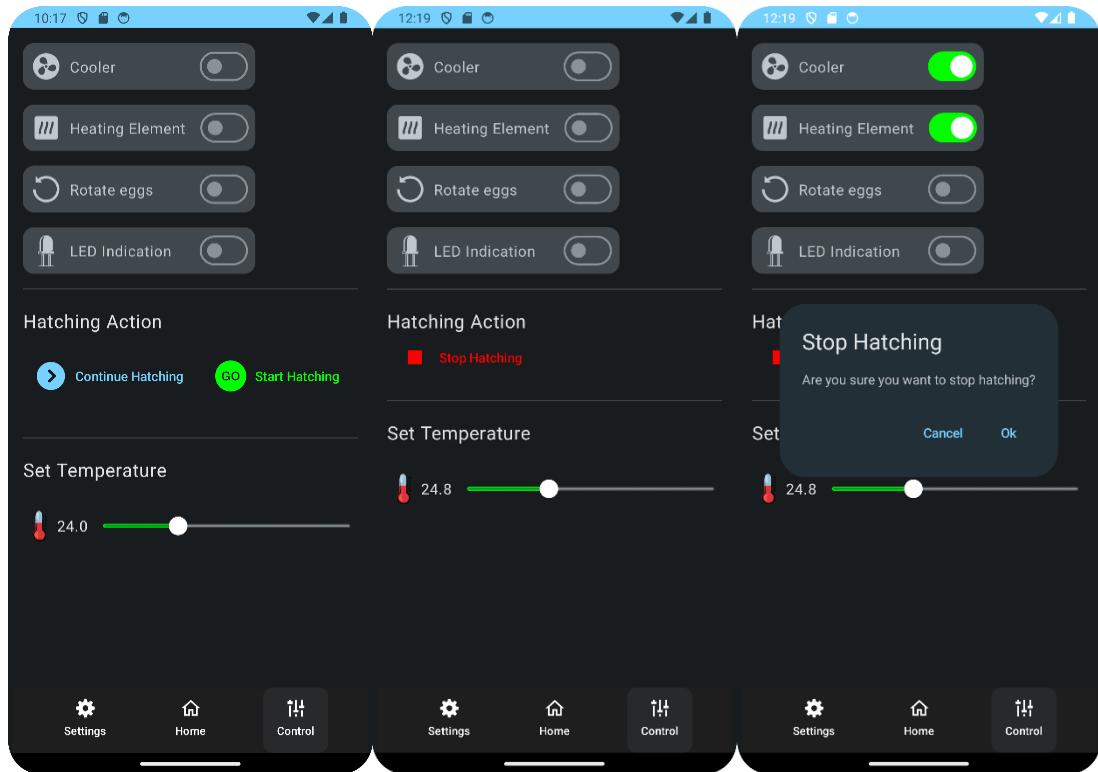
#### 4.2.6. Kezdőlap



48. ábra: Kezdőlap oldal

Az alkalmazás kezdőlapján (48. ábra) áttekinthető formában jelennek meg a csibekeltetés folyamatához kapcsolódó legfontosabb információk. Az aktuális hőmérsékleti és páratartalom-értékek mellett az oldal jelzi azt is, hogy a keltető teteje zárt állapotban van-e. Megtekinthető továbbá az utolsó tojásforgatás időpontja, illetve a jelenlegi keltetési állapot részletei is. Az alkalmazás hasznos tippekkel segíti elő a megfelelő keltetési körülmények fenntartását. Az adatok valós időben frissülnek, így mindenkorán pontos és aktuális információkat jelenítenek meg a keltetési folyamatról.

#### 4.2.7. Vezérlés



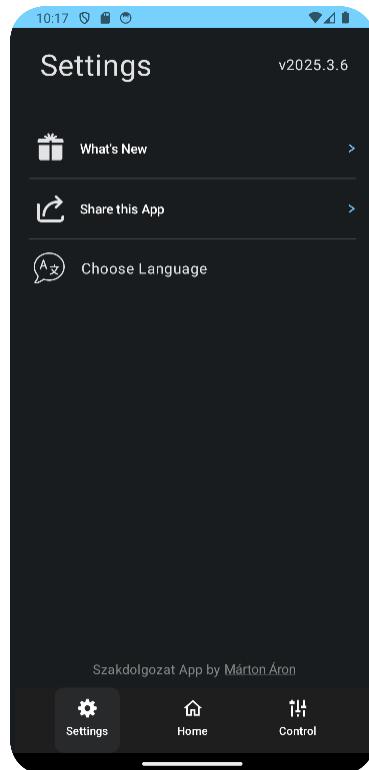
49. ábra: Vezérlés oldal

A vezérlés oldal (49. ábra) a keltető főbb hardverelemeinek manuális működtetésére szolgál. Innen kapcsolható be vagy ki a fűtőelem, amely a megfelelő hőmérséklet biztosításáért felel. A ventilátor szintén vezérelhető, ami a levegő áramoltatását segíti elő a belső térben. A tojásforgató motor működtetése is innen történik, így a forgatás igény szerint indítható vagy szüneteltethető. A LED panel is külön kapcsolható, amely vizuális visszajelzést ad az egyes hardverelemek állapotáról. A kívánt hőmérséklet pontosan megadható, ezt a rendszer figyelembe veszi az automatikus fűtésvezérlés során.

A vezérlés oldalon a keltetési állapot is szabályozható, így lehetőség van a folyamat elindítására, szüneteltetésére vagy újrakezdésére. Ha a keltetés éppen aktív, akkor az itt található vezérlővel leállítható, míg szüneteltetett állapotból folytatható vagy teljesen előlről indítható.

Az oldal kialakítása áttekinthető, így a beavatkozások gyorsan és hatékonyan elvégezhetők. Amennyiben valamelyik hardver már be van kapcsolva, az alkalmazás automatikusan frissíti az állapotát, és ennek megfelelően jeleníti meg a kapcsolót is, például, ha a ventilátor aktív, akkor a hozzáartozó kapcsoló bekapcsolt (on) állásban jelenik meg.

#### 4.2.8. Beállítások

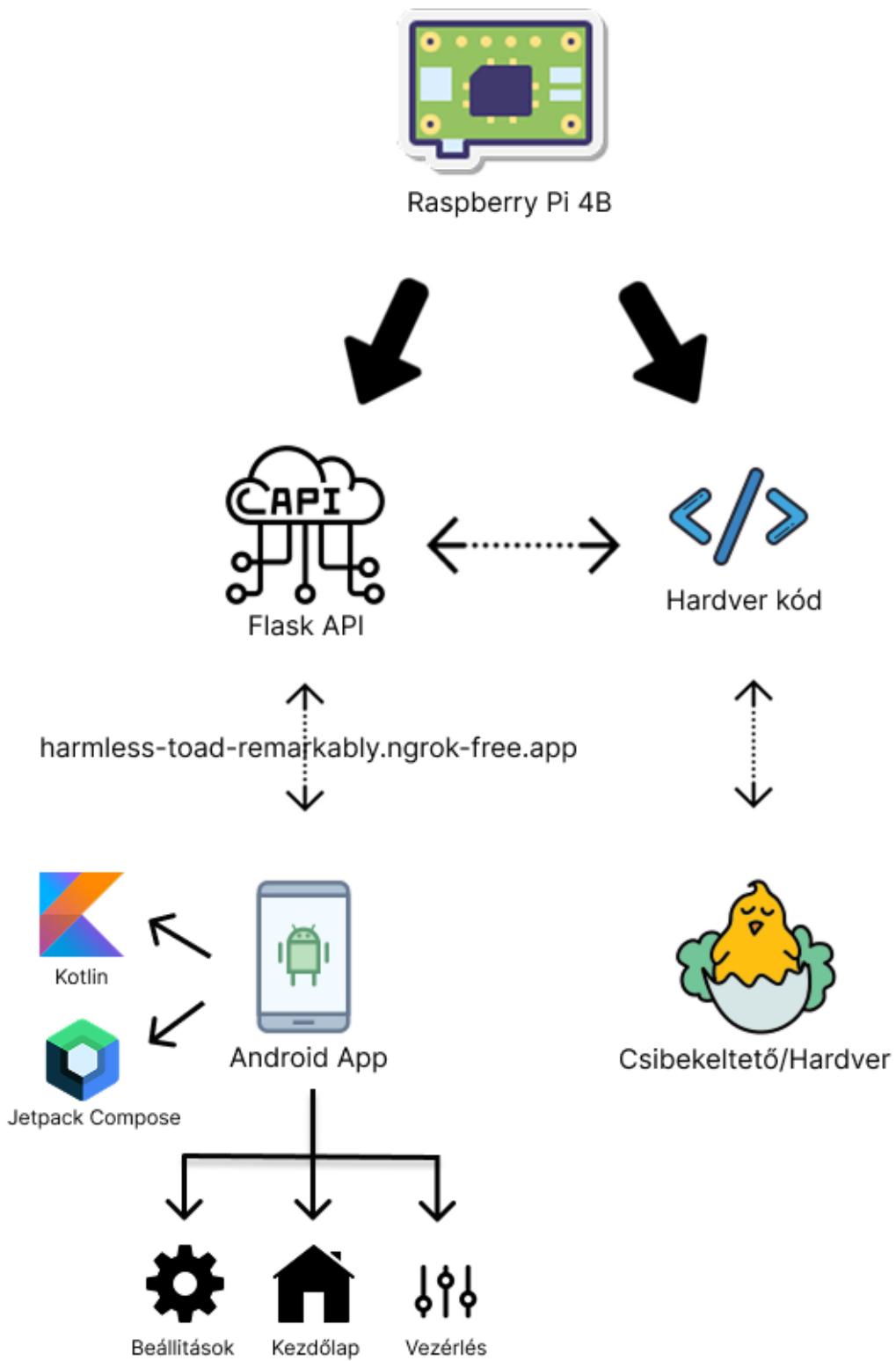


50. ábra: Beállítások oldal

A beállítások oldal (50. ábra) az alkalmazás testreszabására és az információk megtekintésére szolgál. Itt választható ki az alkalmazás nyelve, amely magyar vagy angol lehet. Megtekinthetők az aktuális verzióhoz tartozó új funkciók és fejlesztések is. Az alkalmazás egyszerűen megosztható másokkal egy dedikált megosztási lehetőségen keresztül. Az oldalon feltüntetésre kerül az aktuális alkalmazásverzió száma, valamint a fejlesztő neve is.

A beállítások célja, hogy a felhasználói élmény kényelmesen személyre szabható legyen. Az átlátható elrendezés segít a funkciók gyors megtalálásában.

### 4.3. Szoftver architektúrális ábra



51. ábra: Szoftver architektúrális ábra

## 5. Összegzés

A szakdolgozatom célja egy működő, automatizált csibekeltető rendszer létrehozása volt, amelyhez egy Android-alkalmazást is készítettem saját használatra, hogy a rendszer állapotát kényelmesen, távolról is figyelemmel kísérhessem. A projekt során egy Raspberry Pi 4B mikroszámitógépet használtam központi vezérlőként, amely különböző hardvereket vezérel reléken keresztül. A rendszer feladata a keltetéshez szükséges hőmérséklet, páratartalom és szellőzés precíz szabályozása, valamint a tojások időszakos forgatása. A fizikai eszközökön túl fejlesztettem egy Python alapú backend API-t, amely kommunikál a hardverrel, valamint egy Kotlin Compose-ban írt Android alkalmazást is, amely valós időben jeleníti meg a mért adatokat.

A fejlesztés során számos nehézségbe ütköztem. Először is, a Raspberry Pi GPIO-kezelésének elsajátítása és a relékkal való biztonságos munka komoly tanulási folyamatot igényelt. A Raspberry Pi GPIO-jain keresztüli relék vezérlése során nehézséget okozott a megfelelő kimeneti logika kialakítása, mivel a relék nem mindenkor a várt módon kapcsoltak. A keltetődoboz anyagának kiválasztása szintén problémás volt: hőszigetelés, nedvességállóság és megmunkálhatóság szempontjából is kompromisszumot kellett kötni. Emellett a kábelezés elrendezése is kihívást jelentett, hogy minden eszköz jól hozzáférhető és biztonságosan rögzített legyen.

A szoftveres oldalon a legnagyobb feladatot a keltetési logika kidolgozása jelentette: a fűtés és tojásforgatás időzítése pontos szinkronizálást igényelt, míg a ventilátor esetében nem volt szükség összehangolásra, mivel az folyamatosan működik a keltetés teljes ideje alatt.

Bár a Flask keretrendszerrel már korábban is dolgoztam, a Raspberry Pi használata, valamint a Kotlin nyelv és a Jetpack Compose fejlesztési környezet teljesen új volt számomra. A Jetpack Compose modern, deklaratív megközelítése lehetővé tette, hogy egy egyszerű, mégis jól használható mobilalkalmazást készítsek, amely valós időben megjeleníti a keltető állapotát. A projekt során sok kihívással szembesültem, amelyeket meg kellett oldanom, és ezáltal rendkívül sok gyakorlati tapasztalatot szereztem a hardvervezérlés, a szoftverfejlesztés és az új technológiák alkalmazása terén.

Továbbfejlesztési lehetőségeként érdemes lenne egy oxigénszint-mérőt is beépíteni a rendszerbe, amely segítene az optimális légösszetétel fenntartásában a keltetés teljes időtartama alatt. A jelenlegi megoldásban a levegő cseréje a keringető ventilátor működésén keresztül

történik, azonban ez nem teszi lehetővé a légáramlás pontos és célzott szabályozását. A jövőben érdemes lenne egy automatizált, időzíthető szellőztető rendszert kialakítani, amely a ventilátortól függetlenül képes friss levegőt bejuttatni a keltetőbe. Ez különösen fontos lehet a keltetés utolsó szakaszában, amikor a csibék oxigénigénye megnövekszik.

Ez a csibekeltető projekt megmutatta számomra, hogy informatikusként is képes vagyok az agrárium egy területén gyakorlati megoldást létrehozni. Bizonyította, hogy a technológiai tudás megfelelő kreativitással más szakterületeken is sikeresen alkalmazható.

Az alábbi QR-kód a projekt GitHub-tárházára vezet, ahol a csibekeltető rendszer teljes forráskódja elérhető. A repository tartalmazza a Raspberry Pi-hez írt Python vezérlőprogramot, a Flask-alapú API-t, az Android-alkalmazás forráskódját Kotlin Jetpack Compose-ban, valamint minden további dokumentációt és műszaki leírást is, amely a rendszer megértéséhez és újraépítéséhez szükséges.



## 6. Irodalomjegyzék és hivatkozások

[1]: Raspberry Pi leírás:

<https://www.raspberrypi.com/documentation/computers/getting-started.html>

[2]: DHT20 Humidity and Temperature Module

<https://aqicn.org/air/sensor/spec/asair-dht20.pdf>

[3]: Visual Studio Code docs:

<https://code.visualstudio.com/docs>

[4]: SSH (Secure Shell) leírás:

[https://hu.wikipedia.org/wiki/Secure\\_Shell](https://hu.wikipedia.org/wiki/Secure_Shell)

[5]: Plexi leírás:

[https://hu.wikipedia.org/wiki/Poli\(metil-metakril%C3%A1t\)](https://hu.wikipedia.org/wiki/Poli(metil-metakril%C3%A1t))

[6]: Expandált polisztirol:

<https://kp.hu/nikecell-hungarocell-polisztirolo-eps-mit-is-takar-a-nev/>

[7]: Hőmérséklet és páratartalom táblázata:

[https://tyukportal.com/csirke-keltetese-keltetogeppe/#4\\_Ezek\\_az\\_idealis\\_parameterek\\_a\\_csirke\\_keltetesehez](https://tyukportal.com/csirke-keltetese-keltetogeppe/#4_Ezek_az_idealis_parameterek_a_csirke_keltetesehez)

[8]: Oklahoma State University Extension. *Artificial Incubation of Poultry Eggs*:

<https://extension.okstate.edu/fact-sheets/artificial-incubation.html#position-and-turning-of-eggs>

[9]: Teljesítmény kiszámolása:

[https://en.wikipedia.org/wiki/Electric\\_power#Resistive\\_circuits](https://en.wikipedia.org/wiki/Electric_power#Resistive_circuits)

[10]: ngrok szolgáltatás:

<https://ngrok.com/docs/what-is-ngrok/>

[11]: curl leírás:

<https://hu.wikipedia.org/wiki/CURL#>

[12]: Jetpack Compose:

<https://developer.android.com/compose>

[13]: Kotlin:

[https://hu.wikipedia.org/wiki/Kotlin\\_\(programoz%C3%A1si\\_nyelv\)](https://hu.wikipedia.org/wiki/Kotlin_(programoz%C3%A1si_nyelv))

Kijelentem, hogy a dolgozat készítése során mesterséges intelligencia alapú eszközöket kizárolag a nyelvi megfogalmazás javítására és háttérinformációk keresésére használtam. A szakmai tartalom saját munkám eredménye.