

Debreceni Egyetem
Természettudományi és Technológiai Kar
Fizika Intézet
Villamosmérnök BSc-képzés

Intelligens kaptármérő rendszer fejlesztése és kivitelezése

Tóth Bálint Zoltán

2024

Tartalomjegyzék

Bevezetés3

1. Az intelligens kaptármérő rendszer bemutatása5

- 1.1. Áttekintés és alapfogalmak6*
- 1.2. Alkalmazott eszközök és technológiák9*
- 1.3. Firebase integrációja a projektben33*
- 1.4. MIT App Inventor: Történeti háttér és működés35*

2. Hardveres megvalósítás36

- 2.1. Kezdeti kísérletek breadboardon38*
- 2.2. NYÁK tervezés és kivitelezés40*
- 2.3. Tápegység és burkolat41*
- 2.4. Váz tervezése és kivitelezése42*
- 2.5. Globális helymeghatározó rendszer43*

3. Szoftveres felépítés46

- 3.1. Kód struktúrája és működése46*
- 3.2. Az adatküldés folyamata a Firebase-re47*
- 3.3. Kód részletek és funkciók47*
- 3.4. Globális helymeghatározás kódja52*
- 3.5. Eszközök inicializálása53*

4. Firebase Realtime Database működése és használata55

- 4.1. Firebase adatbázis működése56*
- 4.2. Az alkalmazás célja és felépítése56*
- 4.3. Második oldal – GPS helymeghatározó58*
- 4.4. Az alkalmazás működése59*

5. Mérés-eredmények60

- 5.1. A szenzorok által mért adatok elemzése60*
- 5.2. Eredmények kiértékelése62*

6. Összegzés és fejlesztési lehetőségek63

- 6.1. A rendszer eredményeinek értékelése63*
- 6.2. További fejlesztési irányok63*

Köszönetnyilvánítás65

Irodalomjegyzék66

Bevezetés

A szakdolgozat témája egy méhészeti gyakorlatban alkalmazható szenzoros rendszer fejlesztése, amely a kaptárak állapotának nyomon követésében és az intenzív hordási időszakban hasznos információkat nyújt. Témaválasztásom inspirációja onnan ered, hogy édesapám már több mint 25 éve méhészkedik, és 10 éves korom óta én is aktívan besegítek a családi méhészetben. A jövőben pedig én is szeretnék méheket tartani.

A méhészet legfontosabb szezonja a hordási időszak, amely során minden méhész kiemelt figyelemmel követi a méhcsaládok állapotát, a méztermelés intenzitását és az aktuális időjárást. A rendszeres ellenőrzés szükségessége, valamint az elérhető adatok távoli monitorozásának igénye vezetett arra, hogy egy komplex, különféle szenzorokkal ellátott rendszert hozzak létre, amely könnyen használható terepen és bármikor biztosítja a kaptárról és környezetéről származó információkat.

Emellett fontos figyelembe venni, hogy a hordási időszakban a méhészek gyakran vándorolni kényszerülnek. A növénytablák vagy erdők, amelyek nektárt biztosítanak a méhek számára, sokszor több száz kilométerre helyezkednek el a méhészek lakhelyétől. Az adott növény virágzásának idejére a méhcsaládokat az adott területre kell szállítani. Ilyen körülmények között rendkívül hasznos, ha távol az otthontól is valós időben követhetőek a kaptár állapotát és környezetét érintő adatok, biztosítva ezzel a méhcsaládok megfelelő gondozását és a termelés optimalizálását. A rendszer lehetővé teszi, hogy a méhészek időben értesüljenek a környezeti paraméterek változásairól, a kaptáron belüli eseményekről, valamint az extrém időjárási viszonyokról, illetve a tűz veszélyéről. Ezáltal biztosítja, hogy a méhész gyorsan reagálhasson a változó körülményekre.

A projekt elsődleges célja a kaptár tömegének mérésére szolgáló rendszer fejlesztése, hiszen a hordási időszakban a mézgyarapodás követése kiemelten fontos a termelés sikeressége és a méhcsaládok állapota szempontjából. Azonban a súlymérés mellett további öt paraméter figyelése is szükséges: a kaptár belső hőmérséklete, a külső környezeti hőmérséklet, a páratartalom, a méhek zajszintje, valamint egy gázérzékelő által mért füstszint mind hasznos információval szolgálnak a méhcsalád aktivitásáról és a kaptár biztonságáról. A hordási időszak alatt például a méhek reggel kirepülnek nektárt gyűjteni, amit a kaptár súlyának csökkenése is jelezhet. A hangérzékelő segítségével a bogarak zümmögési intenzitását is mérhetjük, ami

közvetett információval szolgál a gyűjtőtevékenység intenzitásáról. Ha a zajszint túl magas, az rajzásra utalhat, amit érdemes elkerülni, mivel ilyenkor a méhcsalád egy része elhagyná a kaptárt. Estére a méhek visszatérnek, és a napi súlygyarapodásból következtethetünk a méztermelés sikerességére: egy erős család akár 10 kg-os napi gyarapodást is elérhet.

A további hasznos funkciókból le lehet vonni a következtetést arra, hogy ha magas a páratartalom és a kaptár súlya nem növekszik, valamint a zajszint is alacsony, akkor nagy valószínűséggel aznap esős idő miatt nincs gyűjtőtevékenység. A gázérzékelő pedig figyelmeztet, ha füstöt érzékel, ami a kaptár esetleges felgyulladására utalhat. Sajnos volt már ilyen esetünk is, ezért tartottam fontosnak ezt a biztonsági funkciót.

A rendszert elláttam egy extra funkcióval, hogy pontosan tudjuk a rendszer hollétét. A GPS-nyomkövetés lehetővé teszi, hogy mindig pontosan tudjuk, hol helyezkedik el a kaptármérő rendszer. Ez különösen fontos azokban az esetekben, amikor a méhállományt olyan nehezen megközelíthető helyekre telepítjük a hordás idejére, ahol a tájékozódás kihívást jelenthet. Így biztosítható, hogy mindig pontosan megtaláljuk az adott családot, amelyre a rendszer telepítve lett. Emellett a GPS-nyomkövetés egy másik lehetséges problémára is megoldást nyújt: előfordulhat, hogy a méhállományok és kaptárrendszerek lopás áldozatává válnak. Ha ilyen eset történik, a GPS segítségével nyomon követhetjük a rendszer helyzetét.

Annak érdekében, hogy a távoli helyeken elhelyezett kaptárakról folyamatosan és valós időben kapjunk adatokat, a rendszer a mért értékeket egy SIM-modul segítségével GPRS-kapcsolaton keresztül továbbítja egy alkalmazásra. Az alkalmazásban azonnal látható a kaptár állapota, ami így lehetővé teszi a rendszeres helyszíni ellenőrzés minimalizálását és költséghatékony megoldást nyújt a méhészet számára.

A rendszer központja egy Arduino Nano mikrokontroller, amely a szenzorok által gyűjtött adatokat dolgozza fel és továbbítja a kijelzőre, valamint az alkalmazás számára. A rendszerhez tartozik egy LCD-kijelző, amely az aktuális adatokat mutatja, és egy gomb, amellyel az értékek között válthatunk. Tíz másodpercig tartó gombnyomás esetén a rendszer lenullázza a tehercellát, hogy a következő mérésnél pontos adatokat kaphassunk. A komponenseket egy összehegesztett fémvázra szereltem, mivel a kaptár akár 90 kg-ot is nyomhat, ezért szükséges a stabil, nagy teherbírású kialakítás.

A rendszert úgy alakítottam ki, hogy a szenzorokat könnyen el lehessen távolítani, így a felhasználó a mérendő paramétereket rugalmasan kiválaszthatja, például ha füstölünk, a gázérzékelőt ki lehessen venni, hogy ne jelezzen téves riasztást.

A szakdolgozatom célja e rendszer megtervezése, kivitelezése és annak bemutatása, hogyan segíti a méhkaptár valós idejű monitorozása a méhészeket a hordási időszak alatt a méhcsaládok állapotának követésében, valamint a váratlan helyzetek (pl. tűz vagy lopás) elkerülésében.

1. Az intelligens kaptármérő rendszer bemutatása

Az intelligens kaptármérő rendszer egy olyan szenzorokból és egy Arduino Nano mikrokontrollerből álló eszköz, amely előre beprogramozott kód alapján működik. A rendszer célja, hogy a kaptárak állapotáról és a környezeti paramétereiről megbízható adatokat biztosítson a méhészek számára, segítve ezzel a mindennapi munkát és a termelés hatékonyságát.

A legtöbb szenzor, mint például a DHT11, KY-038, MQ-2 és DS18B20 egyirányú kommunikációval működik, vagyis csak adatokat küld a mikrokontrollernek, amely ezeket feldolgozza. Kivételt képez a tehercella, amelyhez egy HX711-átalakítóra van szükség. Ez az átalakító felerősíti a tehercella által szolgáltatott analóg jelet, majd digitális formában továbbítja az Arduino felé. A HX711 csatlakoztatásához az adat- és órajel-lábakat (DT és SCK) a mikrokontroller megfelelő lábaira kell kötni.

Az adatokat egy LCD-kijelző jeleníti meg, amely I2C kommunikációs modullal csatlakozik az Arduinohoz, így csak két vezetéket igényel az SDA- és SCL-lábakon. A kijelzőn megjelenített adatok egy gomb megnyomásával váltogathatók a teher, a belső és külső hőmérséklet, illetve a páratartalom között. A DHT11-szenzor méri a külső hőmérsékletet és páratartalmat, míg a DS18B20-hőmérséklet-érzékelő a kaptáron belül van elhelyezve és tizedpontos adatokat szolgáltat.

A hangérzékelő a méhek kijáratánál található, így pontos képeket kaphatunk a méhek aktivitásáról a hangintenzitás alapján, míg a füstérzékelő a kaptár oldalára van rögzítve. A tehercella a kaptár alatt helyezkedik el, összetartva a vázat, valamint a kaptárt alátámasztva méri a kaptár teljes tömegét.

A rendszer az összegyűjtött adatokat GPRS-hálózaton keresztül küldi el a Firebase-szerverre egy SIM-modul segítségével, onnan egy mobilalkalmazásra továbbítva bármikor elérhetők. Az adatküldés meghatározott időközönként történik, így folyamatos nyomon követés biztosítható.

Az egész rendszer egy nyomtatott áramkörre épült, amelyre JST-csatlakozókat forrasztottam, hogy a szenzorokat könnyen ki lehessen cserélni vagy el lehessen távolítani. A szenzorokat hosszabb vezetékekkel kötöttem be, amelyeket kábelvédőbe húztam, hogy ellenálljanak az időjárási viszontagságoknak.

A rendszer kiegészítéséhez egy GPS-modult is hozzáadtam, amely a helymeghatározást szolgálja. A GPS-funkciót egy külön Arduino Uno vezérli, amelyre egy proto shieldet helyeztem, hogy a modulok könnyen rögzíthetők legyenek. Egy Neo 6M GPS-modul pontosan meghatározza a rendszer hosszúsági és szélességi koordinátáit. Az adatokat szintén egy SIM-modullal továbbítja a rendszer a Firebasebe, ahol a mobilalkalmazás térképen jelöli ki a pontos helyet.

Az áramellátást 6 darab 1,5 voltos elem biztosítja, amelyek sorosan kapcsolt elemfoglalatban helyezkednek el. Az Arduino Uno működéséhez külön egy 9 voltos lapos elemet használtam, mivel az nem üzemeltet egyszerre sok egységet.

A rendszert egy 20x15x6 cm-es műszerdobozba szereltem, amelyen megfelelő méretű nyílásokat alakítottam ki az antennáknak és a kábeleknek. A doboz három kapcsolót tartalmaz: az egyik a főkapcsoló, amely az egész rendszer áramellátását biztosítja, a második az LCD-kijelzőt vezérli, hogy csak akkor működjön, amikor szükséges, a harmadik pedig a GPS-rendszert kapcsolja ki és be, hogy csak terepen használjon energiát.

A mobilalkalmazás egyszerűen kezelhető: az egyik oldalon a mért adatok láthatók, míg a másik oldalon a GPS által meghatározott helyszín jelenik meg a térképen. Ezáltal a rendszer nemcsak a kaptár állapotának figyelemmel kísérésében segít, hanem könnyen hordozható és a terepi munkát is hatékonyabbá teszi.

1.1. Áttekintés és alapfogalmak

Az intelligens kaptármérő rendszer olyan IoT (Internet of Things) alapú megoldás, amely szenzorokkal figyeli a kaptár környezeti és belső állapotát, majd az adatokat továbbítja egy alkalmazásra, ahol a méhészek valós időben nyomon követhetik ezeket. A rendszer legfontosabb elemei közé tartoznak a különböző szenzorok (tehercella, hőmérséklet- és páratartalom-érzékelő, hangérzékelő és gázérzékelő), egy mikrokontroller, amely az adatok gyűjtését és feldolgozását végzi, valamint egy GSM kommunikációs modul, amely az adatokat egy távoli szerverre, aztán egy alkalmazásba továbbítja.

A rendszer által mért paraméterek fontossága a méhészetben

A kaptárban zajló folyamatok pontos figyelése több tényező mérését igényli. Az intelligens kaptármérő rendszer főbb mért paraméterei a következők: kaptár súlya, belső hőmérséklet, külső hőmérséklet, páratartalom, hangintenzitás és füstkoncentráció.

1. A kaptár súlya

A kaptár súlya a legfontosabb paraméter a hordási időszak alatt, hiszen a méztermelés mértékét közvetlenül mutatja. A méhek által gyűjtött nektárból idővel méz lesz, amely a kaptár súlyának gyarapodásában tükröződik. A kaptár súlyának nyomon követésével a méhészkövetkeztetni tud a méz mennyiségének napi növekedésére. A súlycsökkenés reggelente jelzi, hogy a méhek kirepültek, hogy nektárt gyűjtsenek, míg este, amikor visszatérnek, a súly újra megnövekszik. Ez a minta a hordási aktivitást jelzi, és segít a méhésznek felmérni, milyen intenzitással zajlik a méztermelés. A tehercella, amely akár 200 kg-ig is képes mérni, nemcsak a kaptár aktuális súlyát képes követni, hanem nagyobb terheléseknél, például mézes hordók mérésénél is használható.

2. A kaptár belső hőmérséklete

A kaptár belső hőmérséklete fontos mutatója a méhcsalád egészségének és aktivitásának. A méhek ideális hőmérsékleti tartományban érzik magukat a legjobban, általában 34–36°C között. A belső hőmérséklet a kaptár fűtésének és hűtésének eredménye, amelyet a méhek az egészséges állapot fenntartása érdekében folyamatosan szabályoznak. A belső hőmérséklet emelkedése jelezheti a méhek intenzív aktivitását, míg alacsonyabb hőmérséklet mutathatja, hogy a méhcsalád esetleg kevesebb energiát fordít a hőszabályozásra, ami alacsony aktivitást vagy hidegebb időjárást jelezhet.

Ha a kaptáron belüli hőmérséklet drasztikusan lecsökken, az annak a jele lehet, hogy a méhcsalád legyengült vagy jelentősen megfogyatkozott a létszámuk. Ennek oka lehet valamilyen betegség vagy más káros tényező, például atkák jelenléte. Az atkák – különösen a *Varroa destructor* – a méhcsaládok egyik legpusztítóbb kártevői. Ezek az apró élősködők a méheken élnek, testnedveikkel táplálkoznak, és legyengítik a szervezetüket. Az atkák nemcsak közvetlenül károsítják a méheket, hanem különböző vírusokat is terjesztenek, amelyek tovább rontják a méhcsalád egészségét.

Az atkákkal fertőzött méhcsaládok kevesebb hőt tudnak előállítani, mivel a legyengült méhek nem képesek megfelelően szabályozni a kaptár hőmérsékletét. Ez különösen veszélyes télen, amikor a hideg időjárás miatt a méhcsalád számára létfontosságú a megfelelő hőmérséklet fenntartása. Az atka elleni védekezés ezért kiemelten fontos a méhészetekben. A méhészek különböző módszereket alkalmaznak, például kémiai kezeléseket, biológiai megoldásokat vagy hőkezelést, hogy megfékezzék az atkák terjedését és biztosítsák a méhcsaládok egészségét és túlélését. Az atka elleni védekezés nemcsak a méhcsaládok számára létfontosságú, hanem a méhészeti termelés sikerességének is alapfeltétele.

3. Külső hőmérséklet

A külső hőmérséklet figyelemmel kísérése lehetővé teszi a méhésznek a méhek viselkedésének előrejelzését. Magasabb hőmérsékleten a méhek aktívabban gyűjtenek, míg alacsonyabb hőmérséklet, vagyis rossz időben kevesebb méh hagyja el a kaptárat. Ez a mutató közvetett információval szolgál a nektárgyűjtés sikerességéről és intenzitásáról is. Ha például hidegebb időjárás érkezik, a méhek a kaptárban maradhatnak, ami csökkentheti a napi gyarapodást. A külső hőmérséklet figyelése így segít a méhésznek a várható termelési adatok felmérésében és a szükséges intézkedések meghozatalában.

4. Páratartalom

A páratartalom fontos szerepet játszik a hordási időszakban, mivel az esős, párás időjárás megakadályozza a méhek repülését és a nektárgyűjtést. Ha a páratartalom magas, a méhészt feltételezheti, hogy aznap nem lesz jelentős gyarapodás, hiszen a méhek inkább a kaptárban maradnak. Az alacsony külső páratartalom segít a nektár átalakításában is, hiszen a méhek folyamatosan párologtatják a nedvességet a mézből, hogy az elérje a megfelelő állagot. Az alacsony páratartalom kedvezőbb körülményeket biztosít a nektár gyűjtésére, míg a magasabb páratartalom lassíthatja a mézkészítés folyamatát.

5. Hangintenzitás (zümmögés)

A méhek zümmögésének intenzitása fontos mutatója a kaptár belső aktivitásának és a hordási időszak intenzitásának. A hangérzékelő figyeli a kaptárban zajló zümmögést, amely a méhek mozgásának és munkavégzésének eredménye. Ha a méhek intenzíven zümmögnek, az erős gyűjtési tevékenységre utal. Ugyanakkor a nagyon erős, folyamatos hangos zümmögés rajzásra utalhat, amikor a méhcsalád egy része elhagyni készül a kaptárt, hogy új lakhelyet keressen. A rajzást megelőző zajszintváltozás segíthet a méhésznek abban, hogy időben beavatkozzon és megakadályozza a méhcsalád rajzását, ami csökkentheti a termelést.

6. Füstkoncentráció (gázérzékelő)

A füstérzékelés a kaptár biztonságának egyik alapvető eleme. Egy esetleges kaptártűz komoly veszélyt jelenthet a méhcsaládra és az anyagi értékekre is. A gázérzékelő segítségével a rendszer észleli a füstöt, amely riasztást vált ki, így a méhészt gyorsan reagálhat a veszélyre. Ez a funkció különösen fontos a méhészetek biztonságának megőrzése érdekében, mivel a

kaptárak igen könnyen lángra tudnak kapni száraz időszakban, a füstölés alatt egy félrement szikra komoly károkat okozhat.

Az intelligens kaptármérő rendszer ezen paraméterek figyelésével egy átfogó képet nyújt a méhcsalád mindennapi működéséről és a méztermelés alakulásáról. A rendszer által nyújtott adatokat valós időben lehet nyomon követni, így a méhészek azonnal értesülhetnek a kaptárban zajló eseményekről, és szükség esetén beavatkozhatnak. Ez nemcsak a méhészet hatékonyságát növeli, hanem a termelés és a méhcsalád egészségének hosszú távú fenntartásához is hozzájárul. Az intelligens kaptármérő rendszer alkalmazása lehetőséget biztosít a méhészek számára, hogy a modern technológia segítségével pontosabban megfigyeljék a környezeti feltételeket és a méhek által közölt fizikai jeleket, ezzel támogatva a kaptárak optimális állapotának fenntartását.

1.2. Alkalmazott eszközök és technológiák

Az intelligens kaptármérő rendszer megvalósításához különböző hardverek és szoftverek integrálása szükséges, amelyek lehetővé teszik a szenzorok adatainak gyűjtését, feldolgozását és távoli elérését. A következőkben az Arduino-platform és a SIM800L-modul mellett a különböző szenzorok és kijelzők funkcióját, valamint a Firebase és az MIT App Inventor szerepét részletezem.

Arduino platform ismertetése

Történeti áttekintés

Az Arduino egy nyílt forráskódú, könnyen használható hardver- és szoftverplatform, amelyet 2005-ben alkottak meg Olaszországban azzal a céllal, hogy megkönnyítse a kezdő és haladó fejlesztők számára az elektronikai projektek megvalósítását. Az Arduino-projektek középpontjában a könnyű kezelhetőség és az alacsony költségek állnak, így népszerű választás oktatási intézményekben, hobbiprojektekben és professzionális alkalmazásokban is. A platform rengeteg lehetőséget kínál, kezdve az egyszerű LED-vezérléstől egészen komplex IoT-alkalmazásokig.

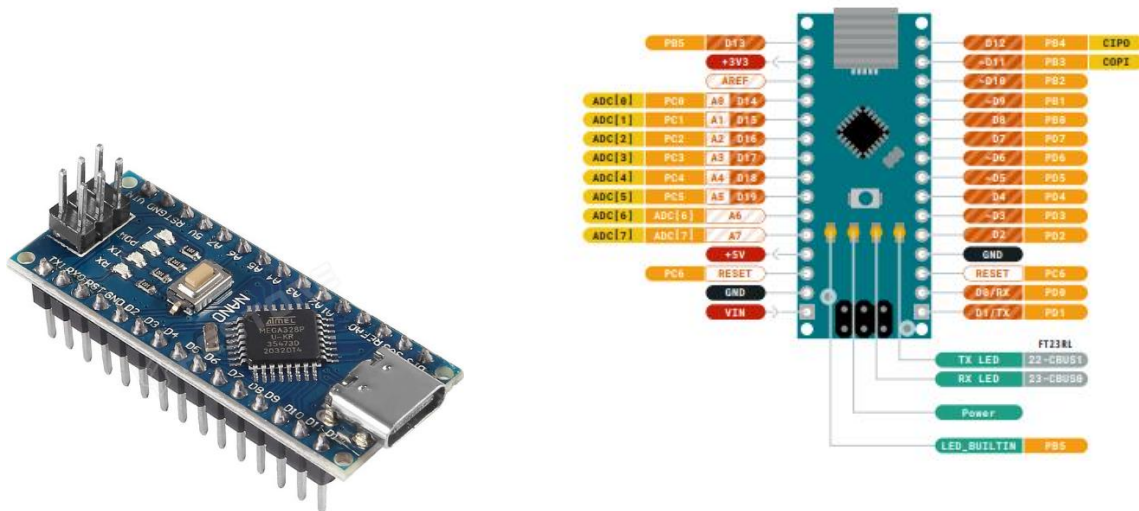
Az Arduino IDE (Integrated Development Environment) segítségével a felhasználók könnyedén írhatnak és tölthetnek fel kódot az Arduino mikrovezérlőkre. Az IDE C++ nyelvet használ alapul, amelyet egyszerűsítenek és bővítenek az Arduino-specifikus könyvtárak és

funkciók. Az Arduino IDE főbb részei a sketch és a soros monitor, amely lehetővé teszi az eszköz által generált adatok megtekintését és elemzését.

Arduino Nano bemutatása

Az Arduino Nano egy kisméretű, energiatakarékos mikrokontroller, amely 8-bites ATmega328P-processzorra épül. Ez a típusú Arduino kifejezetten helytakarékos alkalmazásokhoz készült, így ideális választás IoT-projektekhez és hordozható megoldásokhoz. A Nano 5V feszültség szinten működik, és számos digitális és analóg bemenettel rendelkezik, ami lehetővé teszi többféle szenzor és modul csatlakoztatását.

- **Processzor:** ATmega328P (8 bit, 16 MHz)
- **Flash memória:** 32 kB, amelyből 2 kB a bootloader számára fenntartva
- **RAM:** 2 kB
- **EEPROM:** 1 kB
- **Csatlakozási lehetőségek:** 14 digitális I/O (ebből 6 PWM-képes), 8 analóg bemenet
- **Áramellátás:** 5V USB-n keresztül, vagy 7–12V külső áramforrással



1. ábra. Az Arduino Nano és lábkiosztása [1]

Az Arduino Nano számos szenzort és modult képes kezelni egyszerre, így ideális választás olyan rendszerekhez, ahol számos szenzor működik egy időben. Kompakt mérete és alacsony energiafogyasztása lehetővé teszi, hogy kisebb, hordozható műszerdobozban is könnyen elférjen. A mikrokontroller könnyen programozható, mivel a jól dokumentált Arduino IDE és a széles körű közösségi támogatás egyszerűvé teszi a fejlesztést még kezdők számára is. Emellett az Arduino Nano költséghatékony megoldás, így a rendszer megvalósítása nem jelent nagy anyagi terhet.[1]

Hátrányok és összehasonlítás az Arduino Mega-val és Raspberry Pi-vel:

Arduino Mega

- **Előnyök:** Az Arduino Mega nagyobb RAM-mal (8 kB) és flash memóriával (256 KB) rendelkezik, mint az Arduino Nano, így alkalmasabb összetettebb programok futtatására. Több I/O-lábat kínál (54 digitális és 16 analóg bemenet), amelyek lehetővé teszik nagyobb rendszerek, például több szenzort igénylő projektek kezelését.
- **Hátrányok:** Az Arduino Mega nagyobb méretű, így kevésbé praktikus olyan alkalmazásokhoz, ahol a helytakarékoság kulcsfontosságú. Az ára is borsosabb, mint a Nanoé.
- **Alkalmazási terület:** Az Arduino Mega nagyobb méretének és kapacitásának köszönhetően alkalmasabb komplex vezérlőrendszerekhez.[2]

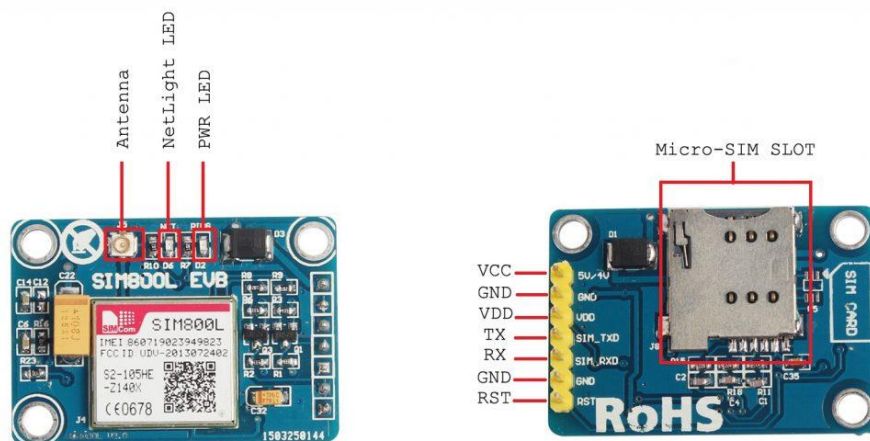
Raspberry Pi

- **Előnyök:** A Raspberry Pi nemcsak mikrokontrollerként, hanem teljes értékű „miniszámítógépként” is funkcionál. Nagyobb feldolgozási teljesítményt nyújt, rendelkezik operációs rendszerrel (Raspberry Pi OS) és könnyen csatlakozik hálózathoz Wi-Fi vagy Ethernet segítségével. Képes videók, képek feldolgozására és fejlettebb alkalmazások futtatására is.
- **Hátrányok:** A Raspberry Pi magasabb energiaigényű, ami nem ideális akkumulátoros rendszerekhez. A nagyobb számítási teljesítmény és rugalmasság miatt drágább és komplexebb a programozása is, különösen kezdők számára.
- **Alkalmazási terület:** A Raspberry Pi tökéletes választás olyan projektekhez, amelyek fejlett adatfeldolgozást, grafikus interfészeket vagy internetkapcsolatot igényelnek, de kevésbé praktikus az egyszerű szenzorvezérlési feladatokra. [3]
- Az Arduino Nano kis méretének és alacsony költségének köszönhetően kiváló választás alapvető szenzorvezérléshez és egyszerű vezérlési feladatokhoz. Az Arduino Mega nagyobb kapacitása és több I/O-lába komplexebb rendszereket tesz lehetővé, míg a Raspberry Pi fejlett számítási és hálózati képességeivel olyan projektekben használható, ahol teljes körű adatfeldolgozásra és széles körű funkcionalitásra van szükség. Az intelligens kaptármérő rendszerhez az Arduino Nano ideális, mivel az egyszerűbb vezérlési feladatokhoz bőven elegendő képességgel rendelkezik és gazdaságos megoldást kínál.

SIM800L EVB-modul ismertetése

A SIM800L egy alacsony költségű GSM/GPRS-modul, amely lehetővé teszi a mikrokontroller számára az internetes adatkapcsolatot. A SIM800L-modult gyakran használják IoT-projektekben, mivel kis mérete ellenére számos funkcióval rendelkezik, mint például az SMS-küldés, hívásfogadás, valamint a mobil adatkapcsolat (GPRS) biztosítása.

- **Működési feszültség:** 3.4–5V
- **Adatátvitel:** GSM/GPRS Class 10 (85.6 kbps)
- **Frekvenciasávok:** 850/900/1800/1900 MHz
- **Csatlakozás:** TX és RX UART pinek, amelyekhez egyszerűen csatlakoztatható az Arduino



2. ábra. A SIM800L EVB-modul [4]

Antenna-csatlakoztatás: külső antenna használata a jobb jelerősség érdekében

A modul használatához szükséges a megfelelő SIM-kártya behelyezése. A SIM-kártyáról el kell távolítani a PIN-kódot, hogy az eszköz akadálytalanul hozzáférhessen. Ezen kívül a modul működéséhez be kell állítani az APN-t (Access Point Name), amely a szolgáltatótól kapott internetkapcsolathoz szükséges. Ezen beállítások elvégzése után a SIM800L-modullal képes az Arduino adatokat küldeni és fogadni a GSM-hálózaton keresztül. [5]

A SIM800L-modul működésének ellenőrzésére és vezérlésére AT-parancsokat használunk, amelyek a modul és a mikrokontroller közötti kommunikáció alapját képezik. Az **AT-parancsok** (Attention Commands) egy szabványos kommunikációs protokoll részei, amelyeket modemek vezérlésére fejlesztettek ki. Segítségükkel beállítható a GSM-modul működése, lekérdezhetők az állapotinformációk, és végrehajthatók műveletek, például adatküldés.

Fontos AT-parancsok:

- **AT:** Ellenőrzi, hogy a modul válaszol-e. Ha a válasz "OK", akkor a modul működőképese.
- **AT+CSQ:** Lekérdezi a hálózat jelerősségét. Az érték 0–31 közötti lehet, ahol a 10 alatti értékek már gyenge jelet jelentenek, és adatküldésre nem biztos, hogy megfelelőek.
- **AT+CGMI:** Lekérdezi a gyártó nevét.
- **AT+GMR:** Lekérdezi a modul szoftververzióját.
- **AT+COPS?:** Megmutatja, hogy a modul melyik hálózathoz csatlakozott.

Hálózat jelerősségének fontossága:

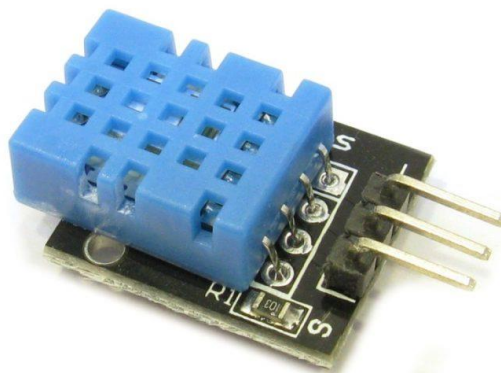
A GSM-modul stabil működéséhez megfelelő jelerősség szükséges. Ha a AT+CSQ parancs gyenge jelet mutat (például 5 vagy annál alacsonyabb), a modul nem tud megbízhatóan csatlakozni a hálózathoz, így adatküldés és fogadás nem lehetséges. Ilyen esetekben érdemes:

- Ellenőrizni az antenna csatlakozását.
- A modult egy olyan helyre áthelyezni, ahol jobb a térerő.
- Meggyőződni arról, hogy a szolgáltató lefedettsége megfelelő a helyszínen.

Az AT-parancsok nemcsak a modul alapvető működésének tesztelésére szolgálnak, hanem elengedhetetlenek a hálózati kapcsolat minőségének ellenőrzéséhez is. A megfelelő beállítások és körülmények nélkül a SIM800L-modul nem tud hatékonyan kommunikálni, így a projekt stabil működésének biztosítása érdekében fontos a rendszeres tesztelés és finomhangolás.

A SIM800L-modullal való adatküldés lehetőséget nyújt arra, hogy a méhész távolról figyelhesse a kaptárban zajló folyamatokat, mivel a mért adatokat közvetlenül az internetre továbbítja, amelyeket a felhasználó bárholnan elérhet. Azért választottam a 2G-képes SIM-modult, mert az országban szinte mindenhol biztosított a 2G hálózati lefedettség, amely különösen fontos a mezőgazdasági területeken vagy távoli helyszíneken, ahol más hálózati technológiák, például a 4G vagy 5G esetleg nem érhetők el. A SIM800L-modul előnye, hogy szabad ég alatt szinte azonnal képes csatlakozni a hálózathoz, minimális konfigurációs igénnyel, így gyors és megbízható adatkapcsolatot biztosít a projekt számára.[6]

Hőmérséklet- és páratartalom-érzékelő DHT11



3. ábra. A DHT11-szenzor [7]

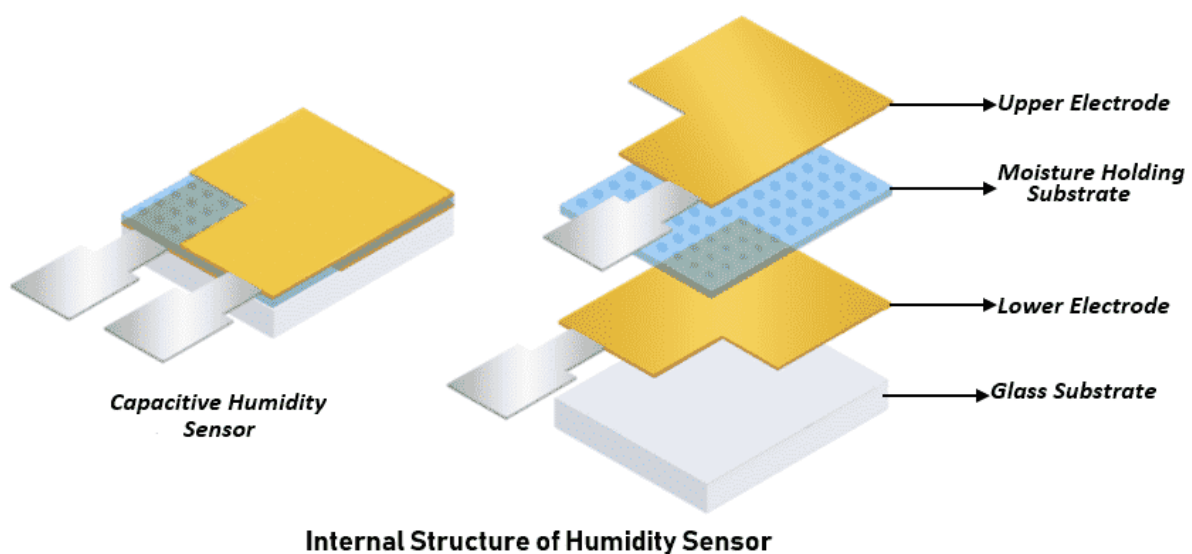
A DHT11 egy népszerű és költséghatékony hőmérséklet- és páratartalom-érzékelő, amely egyszerűen integrálható különböző projektekbe. Az érzékelő digitális jelet küld az Arduinonak, így nincs szükség analóg bemenetre vagy bonyolult adatfeldolgozásra.

- **Működési feszültség:** 3–5V, így kompatibilis a legtöbb mikrokontrollerrel.
- **Hőmérsékletmérési tartomány:** 0–50 °C, ± 2 °C pontossággal. Ideális beltéri és mérsékelt környezeti alkalmazásokhoz.
- **Páratartalommérési tartomány:** 20–80%, $\pm 5\%$ pontossággal. Bár nem alkalmas szélsőséges körülmények közötti mérésekre, alapvető páratartalomellenőrzési célokra megfelelő.
- **Frissítési idő:** 1 másodperc, ami azt jelenti, hogy percenként akár 60 mérés is végezhető.
- **Kimeneti formátum:** Egyedi digitális jel, amely csökkenti a külső zaj és interferencia hatását az analóg rendszerekkel szemben.

A DHT11-ben egy NTC hőérzékelő és egy kapacitív páratartalom-mérő található, amelyeket egy 8 bites mikrovezérlő dolgoz fel. A páratartalom-mérő kapacitív érzékelő két elektródával rendelkezik, amelyek között egy nedvességtartó szubsztrátum található mint dielektromos réteg. A kapacitás változása arányos a környezeti páratartalom szintjének változásával.

Az érzékelő egyszerű kommunikációs protokollt használ, amelynek működtetéséhez minimális programozási ismeretek szükségesek.

A DHT11 gyakran használt olyan rendszerekben, ahol a beltéri környezeti feltételek monitorozása a cél, például okosotthonokban, mezőgazdasági alkalmazásokban. [8]

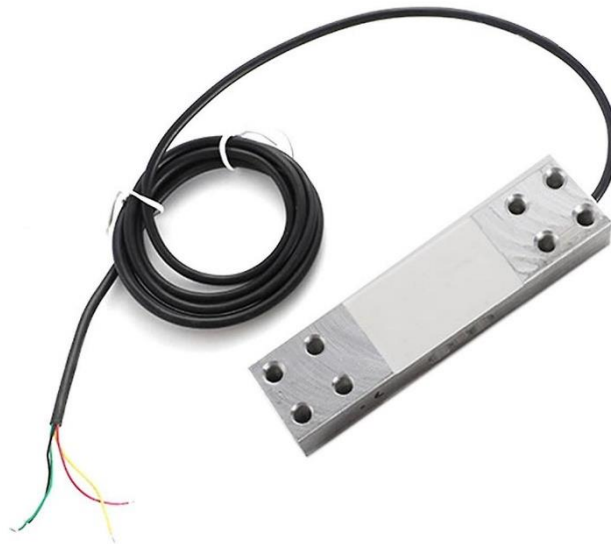


4. ábra. a DHT11 felépítése [8]

Súlymérő szenzor, HX711-átalakító

A tehermérés alapját a mérőcella adja, amelyet a köznyelvben tehercellának neveznek. Ez az eszköz egy biléta (strain gauge) segítségével érzékeli a mechanikai deformációt. A biléta egy vékony vezetőréteg, amelyet a mérőcella felületére rögzítenek. Amikor a cellára terhelés hat, a biléta megnyúlik vagy összenyomódik, ezzel megváltoztatva az elektromos ellenállását. Ez a változás a deformáció mértékével arányos, és egy elektromos jelként kerül továbbításra.

A mérőcellában egy Wheatstone-híd alakítja a biléta által keltett ellenállásváltozást mérhető feszültségkülönbséggé. Ez a rendszer rendkívül érzékeny, és a híd használata minimalizálja a külső zavarok, például hőmérsékleti ingadozások hatását, mivel a híd két ága kompenzálja egymást.



5. ábra. Mérőcella[9]

Mérőcella-specifikációk

- **Kimeneti érzékenység:** 1.0 ± 0.1 mV/V.
- **Hőmérséklet-kompenzációs tartomány:** -10 °C és +40 °C között, amely biztosítja, hogy a mérési pontosság megmaradjon változó hőmérsékleti körülmények között is.
- **Mérési pontosság:** $\pm 0.02\%$ F.S (Full Scale, teljes tartomány).
- **Üzemi hőmérséklet:** -20 °C és +60 °C.
- **Védelmi osztály:** IP65 (védett por és fröccsenő víz ellen).

Telepítés és mechanikai stabilitás

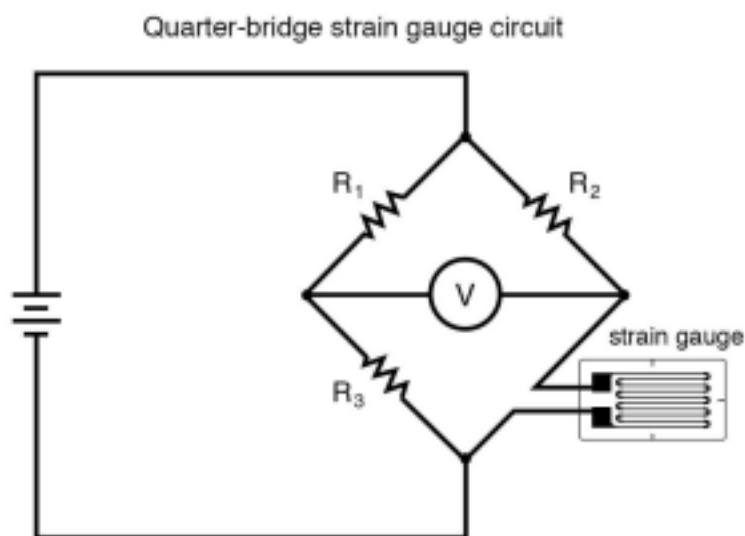
A mérőcellát 4 csavarral kell rögzíteni, hogy stabilitást biztosítson, és elkerülje az esetleges elmozdulást. Az erős és pontos rögzítés kritikus a mérési pontosság érdekében, különösen olyan alkalmazásokban, ahol a dinamikus terhelés okozhat zavarokat.

A biléta működése és szerepe

A biléta a feszültség-ellenállás konverzió elvén működik. Amikor a mérőcella deformálódik:

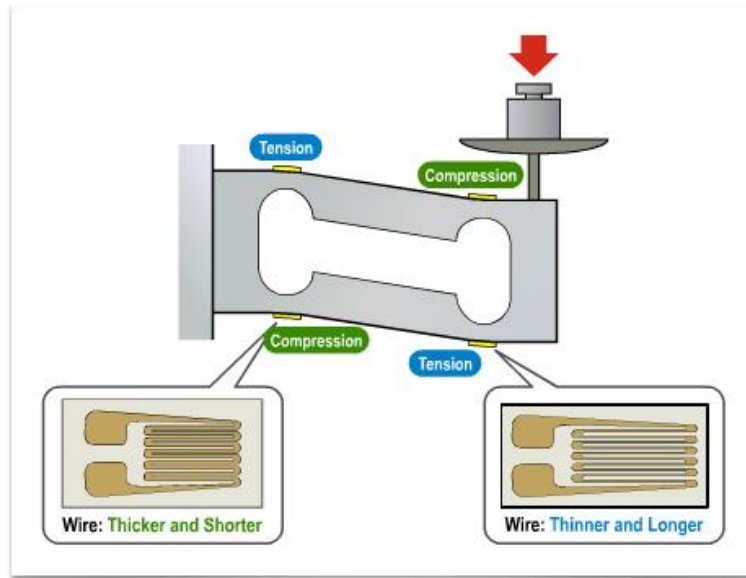
- Nyúláskor nő az ellenállás.
- Összenyomódáskor csökken az ellenállás.

Ez az ellenállásváltozás a Wheatstone-hídon keresztül feszültségváltozássá alakul, amelyet a HX711-es A/D-átalakító továbbít digitális formában az Arduino számára.[10]



6. ábra. Nyúlásmérő biléta[10]

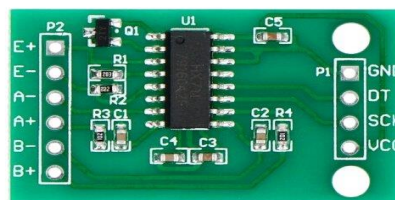
A HX711 egy nagy pontosságú, 24 bites A/D (analóg-digitális)-átalakító, amelyet kifejezetten terhercellák által keltett feszültségek mérésére terveztek. Alapvető feladata az, hogy a mérőcellából érkező nagyon gyenge analóg jelet erősítse, majd azt digitális formává alakítsa át. Ezáltal könnyen feldolgozhatóvá válik a mikrokontroller számára.



7. ábra. A mérőcella működésének ábrázolása[10]

HX711 főbb jellemzői és paraméterei

- **Tápfeszültség:** 2.6–5.5V DC.
- **Felbontás:** 24 bites, amely rendkívül precíz mérést tesz lehetővé.
- **Bemeneti feszültségtartomány:** ± 20 mV (ideális tehercellák alacsony jelszintjéhez).
- **Erősítési tényező (gain):** 32x, 64x, 128x (választható), így a legkisebb változásokat is érzékeli a bemeneti jelekben.
- **Fogyasztás:** Alacsony, működés közben mindössze 1.5 mA, készenléti állapotban kevesebb mint 1 μ A.
- **Működési hőmérséklet:** -40 °C-tól +85 °C-ig.
- **Csatlakoztatás:** 2 differenciális bemeneti csatorna (Channel A és Channel B).[11]



8. ábra. Hx711-átalakító[11]

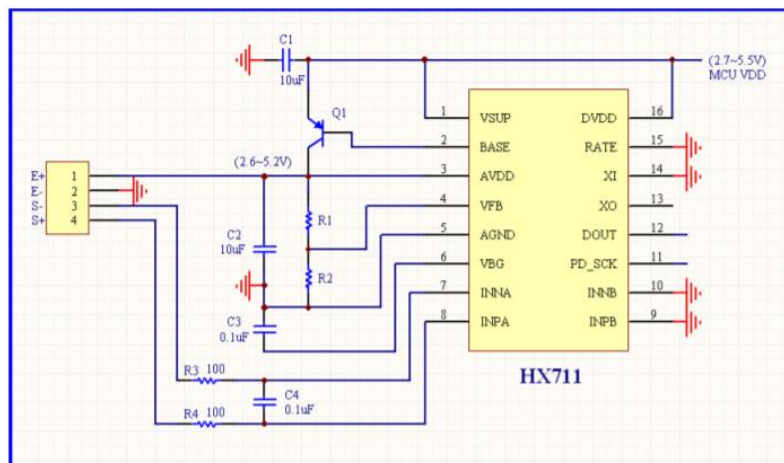
HX711 működési elve

A HX711 képes nagyon alacsony feszültségű jeleket érzékelni és azokat nagy pontossággal felerősíteni, köszönhetően az integrált jelerősítőjének. A chip két csatornával rendelkezik:

- **A-csatorna:** Nagyobb erősítést (128x vagy 64x) biztosít, amely ideális a tehercellák által generált jelekhez.

- **B-csatorna:** Kisebb erősítésű (32x) és általában nem használják tehercelláknál.

A beérkező analóg jeleket a chip digitális formába alakítja, majd soros interfészen keresztül továbbítja a mikrokontrollernek. Ez az interfész egyszerűsíti az integrációt és minimalizálja a külső zaj hatását.[12]



9. ábra. A HX711-átalakító kapcsolási rajza [12]

Használati tapasztalatok

Kezdetben egy kisebb, 5 kg-os mérőcellával kísérleteztem, amelynél a vezetékek kiosztása eltérő volt a szokásostól. Ez okozott némi bonyodalmat, mivel a bekötés nem volt szabványos. Ezért kiemelten fontos, hogy mindig gondosan átvizsgáljuk a dokumentációt, és pontosan ellenőrizzük a kábelezést.

Amennyiben nem vagyunk biztosak abban, hogy melyek a kimeneti (jelvezetékek) és a bemeneti (tápvezetékek) kábelek, ezt könnyen meghatározhatjuk egy egyszerű ellenállásméréssel:

- **Bemeneti vezetékek (tápvezetékek)** közötti ellenállás: **400 ohm $\pm$ 5 ohm.**
- **Kimeneti vezetékek (jelvezetékek)** közötti ellenállás: **350 ohm $\pm$ 5 ohm.**

A tipikus mérőcella vezetékbeosztása:

- **Fekete:** Tápfeszültség (+).
- **Fehér:** Tápfeszültség (-).

- **Piros:** Jel (+).
- **Zöld:** Jel (-).

Ez a rendszer a biléta és a Wheatstone-híd kombinációjával pontos és stabil mérést biztosít, amely rendkívül hasznos a méhészek számára a kaptár terhelésének és méztermelésének figyelemmel kísérésében.[10]

Gázérzékelő (MQ-2)

Az MQ-2-gázérzékelő rendkívül népszerű a beágyazott rendszerek világában, mivel olcsó, pontos és többféle gyúlékony gáz, valamint füst érzékelésére alkalmas. Ez különösen fontos a kaptárak tűzvédelmében, mivel az érzékelő képes gyorsan jelezni a füst jelenlétét, így időben képesek vagyunk azt kezelni.



10.ábra. MQ-2-gázszenzor[13]

Működési elv

Az MQ-2 egy félvezető-alapú érzékelő, amely az érzékelő elem felületén megkötött gázmolekulák által okozott vezetőképesség-változás elvén működik. Az érzékelő fűtőrendszerében nikkel-króm tekercs és alumínium-oxid alapú kerámia található, amely folyamatosan biztosítja az érzékelő elem megfelelő hőmérsékleten tartását. Az érzékelő fő rétege ón-dioxid (SnO_2), amely gyúlékony gázokra különösen érzékeny.

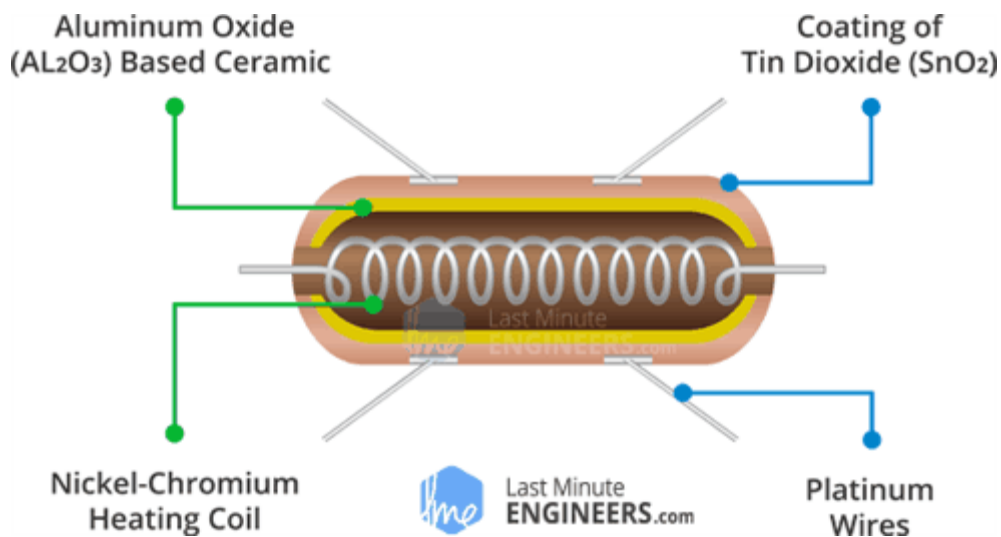
Tiszta levegőben az érzékelő felületén az oxigénmolekulák megkötik az ón-dioxid vezetési elektronjait, ami egy magas ellenállású réteget képez. Ez az állapot gátolja az áram áramlását. Amikor gyúlékony gázok (például metán, propán vagy hidrogén) vannak jelen, ezek reakcióba lépnek az oxigénmolekulákkal, csökkentve az oxigén sűrűségét a felszínen. Ennek hatására felszabadulnak az elektronok, az ellenállás csökken, és megnő az áramvezetés.[14]

Az érzékelő ellenállásának változását a rendszer elektromos jelekké alakítja, amelyek arányosak az érzékelt gáz koncentrációjával. Ezáltal az MQ-2-érzékelő pontosan jelzi a levegőben található gyúlékony gázok jelenlétét és koncentrációját.

Például:

- LPG
- Propán
- Metán
- Hidrogén
- Füst

Az érzékelő kimenete egy analóg feszültség, amely arányos a gázkoncentrációval, így könnyen beolvasható mikrokontrollerek, például Arduino segítségével.



11. ábra. MQ-2-szenzor működési ábrája[14]

Fő paraméterek

- **Működési feszültség:** 5V DC
- **Érzékenységi tartomány:** 200–10000 ppm (parts per million) gyúlékony gázok esetén

- **Fűtőelem fogyasztása:** $\leq 800 \text{ mW}$
- **Működési hőmérséklet:** $-20 \text{ }^{\circ}\text{C} - 50 \text{ }^{\circ}\text{C}$
- **Érzékenység:** többféle gyúlékony gázra és füstre
- **Érzékelési idő:** <10 másodperc
- **Kalibrálási idő:** kb. 24 óra az első használat előtt a pontosság érdekében

Miért választottam az MQ-2-t?

Az MQ-2 népszerűsége az olcsóságából és megbízhatóságából ered. Az eszköz pontosan képes érzékelni a füst és egyéb gázok jelenlétét a kaptár közelében, amely létfontosságú információ a kaptár védelmében. Ezenkívül az érzékelő analóg jelet ad, ami egyszerű feldolgozást tesz lehetővé mikrokontroller segítségével.

Kalibrálás

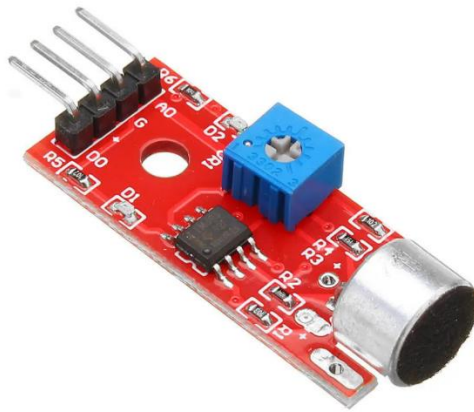
Mivel az MQ-2 által mért adatok mértékegység nélküliek, kalibrálást végeztem az érzékelőn. Ehhez a következő módszert alkalmaztam:

1. **Ellenőrzött körülmények:** Egy gyertya füstjét használtam referenciaértékként.
2. **Füst generálása:** A gyertyát az érzékelő közelébe tartottam, majd eloltottam, hogy füstöt hozzak létre.
3. **Értékek rögzítése:** Megmértem az érzékelő által adott analóg jeleket a füst jelenlétében.
4. **Értékhatár meghatározása:** A mért adatok alapján meghatároztam azt a küszöbértéket, amely fölött biztosan füstöt jelez az érzékelő.

Ez a módszer lehetővé tette, hogy az MQ-2-érzékelőt a kaptár körülményeihez igazítsam, és biztosítsam, hogy időben jelezze a füst jelenlétét. A kapott értékeket később az Arduino-rendszerben rögzítettem, és ennek alapján építettem ki a riasztási rendszert.

Hangérzékelő KY-038

A KY-038 típusú hangérzékelő modul egy mikrofon és egy LM393-komparátor segítségével méri a zajszintet. Ez a modul alkalmas arra, hogy nyomon kövesse a méhek zümmögését, mivel ezek aktivitása fontos jelzője a hordás állapotának. A modul különösen hasznos a rajzás előtti állapotok detektálására, amelyeket a kaptárban lévő zaj mintázataiból lehet következtetni.

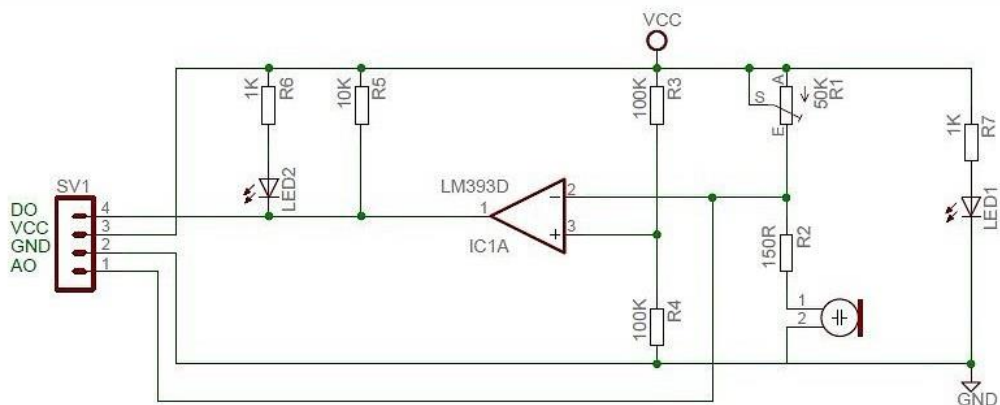


12.ábra. KY-038 hangérzékelő modul[15]

Működési elv

A KY-038 működése az alábbi fő komponenseken alapul:

1. **Elektromos kondenzátor-mikrofon:** Ez a mikrofon érzékeli a környezeti zajt, és elektromos jelet generál annak intenzitásával arányosan.
2. **LM393-komparátor:** A mikrofon jele az LM393 operációs erősítőbe kerül, amely összehasonlítja a jelet egy előre beállított küszöbértékkel. Ez a küszöbérték az állítható potenciométerrel szabályozható.
3. **Digitális és analóg kimenetek:**
 - A **DO (Digital Output)** kimenet alacsony szintű jelet ad, ha a zajszint meghaladja a küszöbértéket.
 - Az **AO (Analog Output)** kimenet analóg jelet biztosít, amely a hang intenzitásának mértékegység nélküli numerikus megjelenítésére használható.[15]



13. ábra. A hangérzékelő modul elektronikájának kapcsolási rajza[16]

Paraméterek

- **Működési feszültség:** 3.3V–5V DC
- **Kimenet:**
 - Digitális: magas/alacsony szint
 - Analóg: arányos feszültségjel
- **Érzékelési tartomány:** Mikrofontól függően változó
- **LED-visszajelzés:** Ha a zajszint meghaladja a küszöbértéket, a modulon lévő LED világít.
- **Kimeneti csatlakozók:**
 - **AO:** Analóg kimenet
 - **DO:** Digitális kimenet (komparátor) [17]

Miért választottam a KY-038-at?

Ez a modul közkedvelt az alacsony költsége és egyszerű kezelhetősége miatt. A zajszint érzékenysége potenciométerrel könnyen beállítható, így különböző alkalmazásokhoz jól illeszthető. Bár a modul által mért érték mértékegység nélküli, kalibrálással az értékek releváns következtetések levonására használhatók.

Kalibrálás

A kalibrálási folyamat során az alábbi lépéseket végeztem:

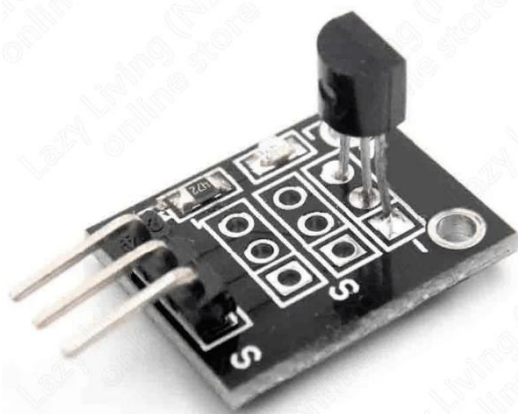
1. **Hangforrások használata:** Telefon segítségével különböző méhhangokat játszottam le, például:
 - Aktív méhek zümmögése
 - Rajzás hangjai
2. **Értékek mérése:** A modul által generált analóg jeleket rögzítettem.
3. **Küszöbérték beállítása:** Az érzékenységi potenciométerrel meghatároztam azt az értéket, amelynél biztosan kimutatható a rajzás állapota.

Értékelés

A kalibráció nem ad 100%-os pontosságot, a kísérleti eredmények megbízható alapot nyújtanak a kaptár aktivitásának nyomon követésére. Az érzékelő által nyújtott adatokat az Arduino-rendszer dolgozza fel, és egyértelmű jelzést ad a rajzáskor fellépő hangminták alapján.

DS18B20-hőmérséklet-érzékelő

A DS18B20 egy digitális, nagy pontosságú hőmérséklet-érzékelő, amely széles körben használatos különböző ipari és hobbi alkalmazásokban, beleértve a kaptármérő rendszereket is. Ez az eszköz megbízható és egyszerű integrációt kínál a legtöbb mikrokontrollerrel, köszönhetően az 1-Wire protokollnak, amely egyetlen adatvezetékkel használja a kommunikációhoz.



14. ábra. A DS18B20-hőmérséklet-érzékelő-modul[18]

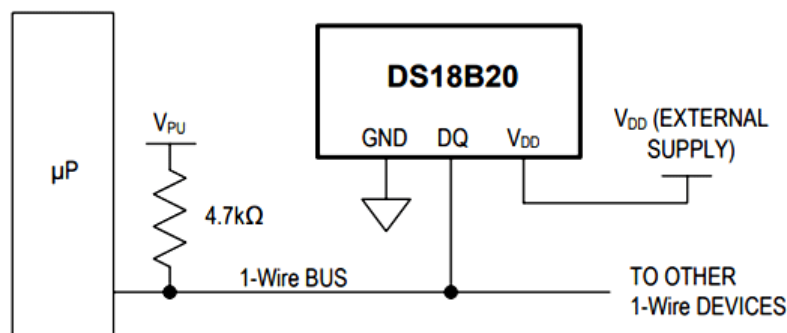
Főbb jellemzők

- **Mérési tartomány:** -55 °C és +125 °C között.
- **Pontosság:** $\pm 0,5$ °C -10 °C és +85 °C között.
- **Felbontás:** Programozható 9, 10, 11 vagy 12 bites felbontás (alapértelmezett 12 bit).
- **Működési feszültség:** 3,0–5,5V.
- **Egyedi 64 bites azonosító:** Minden DS18B20-érzékelő rendelkezik egyedi ROM-kóddal, amely lehetővé teszi több érzékelő egyidejű használatát ugyanazon az adatvonalon.
- **Energiaellátás:** Támogatja az úgynevezett "parazita tápellátást", amely az adatvezetéken keresztül történik, valamint hagyományos külső tápegységgel is működtethető.[19]

Működési elv

A DS18B20 belsejében egy félvezető-alapú hőmérséklet-érzékelő található, amely közvetlenül digitális jelet állít elő. Az érzékelő egy belső memóriaregiszterben tárolja az adatokat, amelyeket az 1-Wire protokollon keresztül lehet kiolvasni. Az eszköz két hőmérséklet-értéket is tárolhat.

Az érzékelő által mért adatokat két biteban tárolja. Az 1-Wire protokoll előnye, hogy több eszköz is csatlakoztatható egy adatvezetékre, amit az eszköz egyedi azonosítója biztosít.



15. ábra. A hőmérő modul elektronikájának kapcsolási rajza[20]

Alkalmazás a kaptármérő rendszerben

Azért választottam a DS18B20-érzékelőt, mert pontos és megbízható hőmérsékleti adatokat szolgáltat, amelyek kulcsfontosságúak a kaptár belső környezetének monitorozásában. A kaptár belső hőmérsékletének stabilitása jelzi a méhcsalád aktivitását és egészségét. A DS18B20 egyszerűen integrálható az Arduino-mikrokontrollerrel, egyetlen digitális pin használatával.

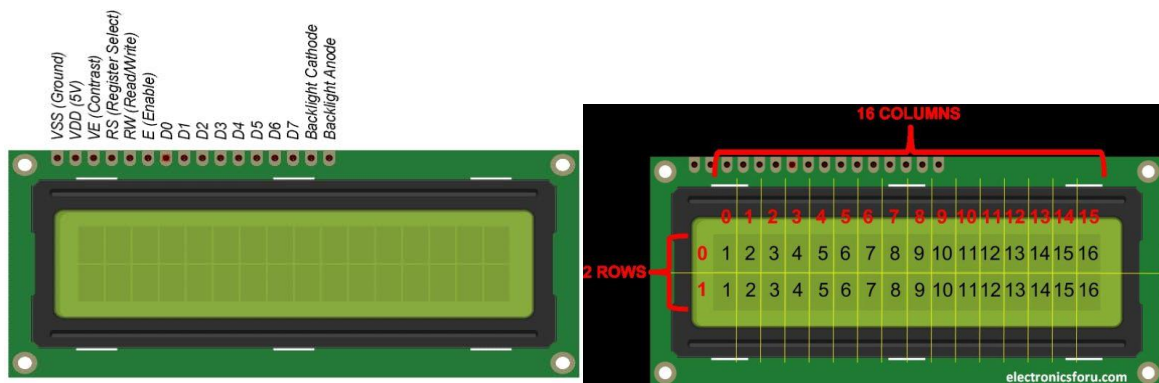
Az eszköz egyedi ROM-kódjának köszönhetően több DS18B20-érzékelő is egyszerűen kezelhető egy rendszerben.

A DS18B20 kis mérete, vízálló kivitele és széles hőmérsékleti tartománya ideális választássá teszi a méhészeti alkalmazásokhoz, különösen a kaptáron belüli méréshez.

LCD-kijelző szerepe és használata

16x2 LCD-kijelző jellemzői

A 16x2 karakteres LCD-kijelző az Arduino-rendszerekhez történő adatmegjelenítés egyik leggyakrabban használt komponense. A kijelző képes egyszerre két sorban, soronként 16 karaktert megjeleníteni, ami lehetővé teszi szöveges információk, mért adatok vagy státuszjelzések helyi megjelenítését.



16. ábra. Az LCD-kijelző lábkiosztása és a kijelző felépítése[21][22]

- **Tápfeszültség:** 3,3–5V DC.
- **Interfész:** Standard 4 vagy 8 bites párhuzamos adatátvitel.
- **Karakterek:** 16x2 elrendezés (16 karakter két sorban).
- **Méret:** 80 mm × 36 mm.
- **Háttérvilágítás:** A beépített LED-háttérvilágítás sötétben is jól olvashatóvá teszi a kijelzőt.
- **Állítható kontraszt:** Egy trimmer potenciométer segítségével a kijelző kontrasztja is könnyen szabályozható.

A kijelző kiváló olyan alkalmazásokhoz, ahol egyszerű adatmegjelenítésre van szükség. Alternatív megoldásként használhattam volna 7 szegmenses kijelzőt is, azonban az csak számokat tud megjeleníteni, így a szöveges üzenetekre nem alkalmas.[22][23]

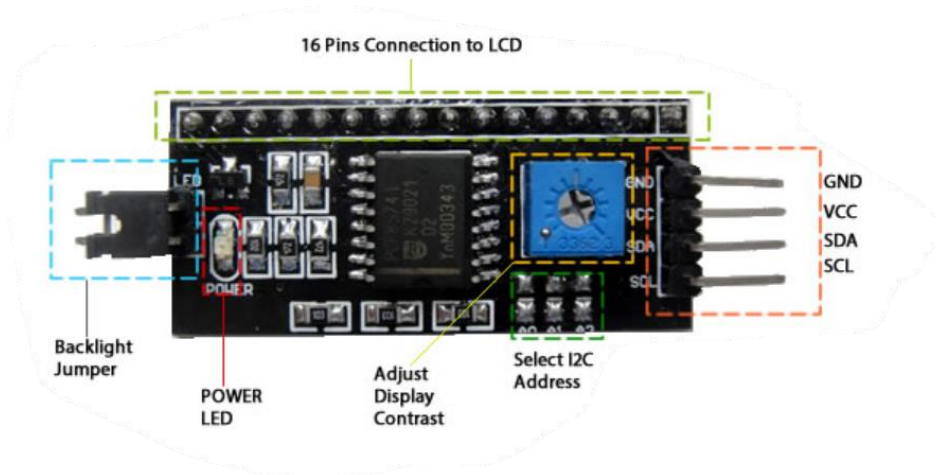
Az I2C interfész adapter szerepe

A hagyományos LCD-kijelzők számos adatpin használatát igénylik (4 vagy 8 adatvezeték, illetve vezérlővezetékek), ami megnehezíti az Arduino rendszerrel való integrációt, különösen, ha több periféria csatlakoztatása is szükséges. Az I2C interfész adapter ezt a problémát oldja meg:

- **Modul feladata:** Az **I2C protokoll** segítségével egyetlen adatvezetéken (SDA) és egy órajelező vezetéken (SCL) keresztül kommunikál az Arduinóval, jelentősen csökkentve a szükséges csatlakozók számát.
- **Chip típusa:** Az adapter jellemzően egy **PCF8574**-chipet tartalmaz, amely az I2C-jeleket párhuzamos jelekké alakítja.
- **Tápfeszültség:** 2,5–6V.

- **Jellemzők:**

- 8 bites portexpander I2C-buszhoz.
- Alacsony áramfogyasztás (max. 10 μ A standby módban).
- Több modul párhuzamos használata: Az I2C-cím hardveresen konfigurálható három címvezetékkel, így akár 8 modul is használható egy buszon.[25]



17. ábra. Az I2C interfész adapter[24]

I2C működési elve

Az I2C (Inter-Integrated Circuit) protokoll egy kétvezetékes, soros kommunikációs interfész:

- **SDA (Serial Data Line):** Adatvonal, amely az eszközök közötti adatátvitelt végzi.
- **SCL (Serial Clock Line):** Órajelező vonal, amely az adatátvitel időzítését szabályozza.

Az I2C-busz master-slave (mester-szolga) felépítésű. Az Arduino a mester szerepét tölti be, az LCD-modul pedig a szolga.

Az I2C-adapter és az LCD-kijelző együttes használata jelentősen egyszerűsíti a telepítést és a vezérlést, mivel az adapter minden szükséges jelet előállít az LCD számára.

Telepítés és használat

A 16x2 LCD-kijelző és az I2C-interfész integrálása során a következő lépéseket végeztem el:

1. **Csatlakoztatás:** Az adapter VCC és GND kivezetéseit a tápellátáshoz (5V és föld), az SDA és SCL kivezetéseit pedig az Arduino megfelelő I2C kivezetéseibe csatlakoztattam.[25]

Felhasználási tapasztalatok

Az I2C-adapter és az LCD-kijelző használata nemcsak helytakarékos, de a megjelenítés minőségét is javítja. A szenzoradatok egyszerűen olvasható formában jelennek meg, és a háttérvilágításnak köszönhetően gyenge fényviszonyok között is jól láthatók. A felhasználóbarát szöveges kijelzés nagyban megkönnyíti az eszköz használatát a terepen.

LM2596 DC-DC konverter modul

Az **LM2596** egy hatékony **step-down (buck)** DC-DC feszültségszabályzó modul, amelyet kifejezetten olyan helyzetekre terveztek, amikor egy magasabb bemeneti feszültséget kell alacsonyabb kimeneti feszültségre csökkenteni. Az alacsony energiafogyasztás és a magas hatásfok miatt ideális választás különböző elektronikai projektekhez.



18. ábra. A LM2596 DC-DC konverter modul[26]

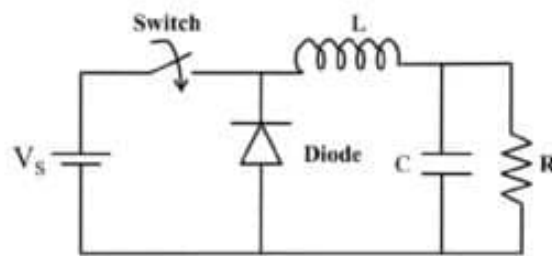
Főbb jellemzők és paraméterek

- **Bemeneti feszültség:** 4–40 V DC.
- **Kimeneti feszültség:** 1,25–37 V DC (állítható potenciométer segítségével).
- **Kimeneti áramerősség:** Maximum 3 A (folyamatos terhelés mellett, hűtés szükséges lehet).
- **Hatásfok:** Akár 92%,
- **Frekvencia:** 150 kHz kapcsolási frekvencia.
- **Működési hőmérséklet:** -40 °C és +85 °C között.
- **Méret:** Kicsi és kompakt, jellemzően 45 × 20 × 14 mm.[27]

Működési elv

Az LM2596-modul a kapcsolóüzemű tápellátás elvén működik, amely a bemeneti feszültséget egy nagy frekvenciájú kapcsolás segítségével csökkenti a kívánt kimeneti értékre. Ez az eljárás sokkal hatékonyabb, mint a lineáris szabályozók, mivel minimális hő keletkezik a feszültségcsökkentés során.

A kimeneti feszültség egy beépített potenciométerrel állítható be, amely lehetővé teszi a pontos szabályozást. A modul integrált hővédelemmel, túláram-védelemmel és rövidzár elleni védelemmel is rendelkezik.[28]



19. ábra. A DC-DC feszültségcsökkentő kapcsolási rajza[28]

Felhasználás a projektben

A **SIM800L-modul** közvetlenül az Arduino Nano **5V-os kimenetéről** nem tudta biztosítani a szükséges áramot. Emiatt az LM2596-modult használtam az energiaellátás stabilizálására:

1. A tápegységből érkező magasabb feszültséget (9V) az LM2596-tal **5V-ra csökkentettem.**
2. Az LM2596-modul kimenetét közvetlenül a SIM800L-modul táplálására használtam.

Ez a megoldás biztosította, hogy a SIM800L elegendő áramot kapjon a működéshez, miközben az áramkör stabil és biztonságos maradt. Az LM2596-modul könnyen integrálható, kompakt mérete és állítható kimenete miatt pedig ideális választásnak bizonyult a projekt számára.

NEO-6M GPS-modul bemutatása

A NEO-6M GPS-modul egy nagy teljesítményű, költséghatékony, önálló GPS-vevő, amelyet a u-blox 6 pozicionáló motor hajt. Ez a modul kis méretének, energiahatékony kialakításának és rugalmas csatlakoztathatóságának köszönhetően ideális választás olyan projektekhez, amelyekben szűk hely és költségvetés áll rendelkezésre.



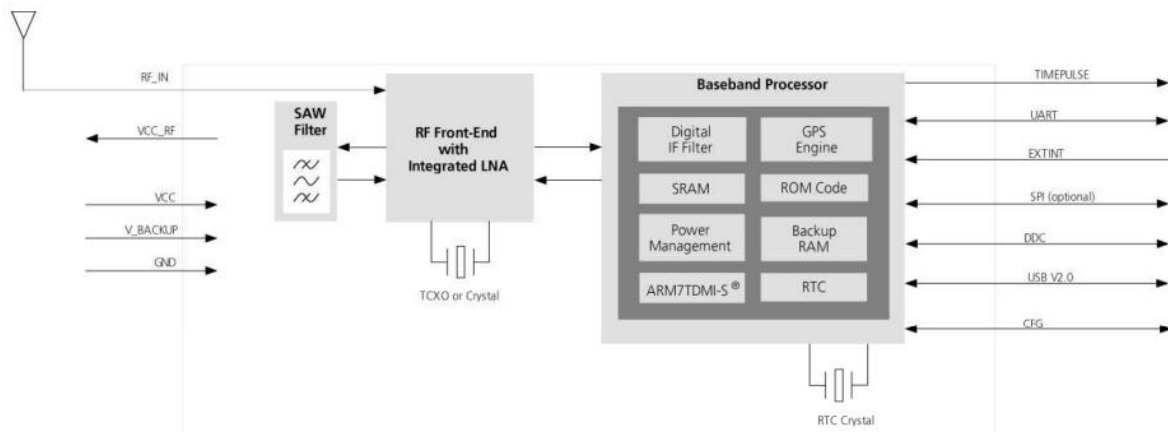
20. ábra. A NEO-6M GPS-modul [29]

Főbb paraméterei és specifikációi

- **Méret:** 16 x 12.2 x 2.4 mm.
- **Csatornák száma:** 50
- **Frekvenciasáv:** GPS L1 (1575.42 MHz), C/A-kód.
- **Pontosság:**
 - Vízzintes helymeghatározás: 2.5 m (standard GPS esetén).
 - Sebességpontosság: 0.1 m/s.
 - Tájéolási pontosság: 0.5°.
- **Érzékenység:**
 - Hidegindítás: -148 dBm.
 - Nyomkövetés: -161 dBm.
- **Indítási idők:**
 - Hidegindítás: 27 másodperc.
 - Melegindítás: 1 másodperc.
- **Adatfrissítési ráta:** Maximum 5 Hz.
- **Energiafogyasztás:**
 - Nyomkövetés: 37 mA (Eco üzemmódban).
 - Hidegindítás: 47 mA.
- **Interfészek:**
 - UART (9600 baud alapértelmezett beállítás).
 - USB (2.0 kompatibilis).
 - SPI- és I2C-kommunikáció támogatása.
- **Tápfeszültség:** 2.7–3.6V.
- **Támogatott protokollok:** NMEA, UBX, RTCM.[30]

Működési elve

A NEO-6M GPS-modul a műholdakról érkező rádiójelek alapján határozza meg a földrajzi pozíciót. A modul egy 50 csatornás pozicionáló motort használ, amely képes párhuzamosan több műhold jeleit feldolgozni, biztosítva a pontos helymeghatározást és gyors műholdcsatlakozást. A jelek feldolgozása során figyelembe veszi a késleltetéseket, multipath hatásokat és a zajokat, ezáltal stabil helymeghatározást nyújt nehéz környezetekben is.



21. ábra. A NEO-6M-modul blockdiagramja[30]

Kiemelkedő funkciók

1. Assisted GPS (A-GPS): Az almanach és efemerisz adatok előzetes letöltése gyorsítja az indítási időt.
2. Energiatakarékos üzemmódok:
 - Eco Mode: Energiahatékonyságra optimalizált nyomkövetés.
 - Power Save Mode: Az alkatrészek időszakos lekapcsolásával csökkenti az energiafogyasztást.
3. Multipath hatás csökkentése: Az innovatív kialakítás minimalizálja az épületekről visszaverődő jelek hatásait.
4. Pontosság:
 - SBAS-támogatás: A WAAS-, EGNOS- és MSAS-rendszerek segítségével tovább növeli a helymeghatározási pontosságot.

Alkalmazási területek

A NEO-6M-modul széles körben használható az alábbi területeken:

- Mobilrobotika,
- Drónok és UAV-rendszerek,

- Időzítésalapú rendszerek,
- Térképezési projektek.

A modul támogatja mind a passzív, mind az aktív antennákat. A telepítésnél figyelni kell az antenna megfelelő árnyékolására és elhelyezésére a pontos működés érdekében.

A NEO-6M GPS-modul egy sokoldalú, kis energiaigényű helymeghatározó eszköz, amely precíz és megbízható adatokat szolgáltat. Kompakt mérete és rugalmas interfészei ideális választássá teszik számos alkalmazásban, különösen olyan projekteknél, ahol a helymeghatározás pontossága kulcsfontosságú.

1.3. Firebase integrációja a projektben

Firestore és annak jellemzői

A Firestore egy Google által biztosított fejlesztési platform, amely számos szolgáltatást kínál a modern alkalmazások létrehozásához és kezeléséhez. Egyik legnagyobb előnye a valós idejű adatkezelés, amelyet különösen hasznosnak találtam a méhészeti kaptármérő rendszer távoli monitorozásában. A Firestore Real-Time Database segítségével az adatokat gyorsan és hatékonyan tudom tárolni és megjeleníteni.

Főbb jellemzők

1. Valós idejű adatbázis (Real-Time Database):

- NoSQL-alapú, hierarchikus szerkezetű adatbázis.
- Az adatok valós időben szinkronizálhatók a kliensek között.
- A változásokról azonnali értesítést küld, így nincs szükség folyamatos lekérdezésekre.
- Websocket-alapú adatkapcsolatot használ, amely gyorsabb, mint a hagyományos HTTP-kérések.

2. Felhőalapú tárolás:

- Az adatok tárolása biztonságos felhőben történik, amelyhez egyszerű hozzáférést biztosít a világ bármely pontjáról.
- Támogatja az adatok valós idejű elemzését és megjelenítését.

3. Autentikáció:

- Az autentikáció lehetővé teszi az email+jelszó alapú belépést, valamint a social media platformok (Google, Facebook, GitHub stb.) integrációját.

4. Többplatformos támogatás:

- Egyszerű integráció webes, iOS és Android alkalmazásokkal.
- Az SDK-k (Software Development Kit) révén különböző programozási nyelvekkel is használható, beleértve a C++-t és a Python-t.

5. Ingyenesen elérhető szolgáltatások:

- A legtöbb funkció kezdetben ingyenes (korlátozott használati keret mellett), ami különösen hasznos kisebb projektekhez.

6. Valós idejű pushértesítések:

- Az adatbázis változásairól azonnali értesítést lehet kapni.

A Firebase Real-Time Database beállítása

A Real-Time Database a projekt központi adattároló rendszere, amely az Arduino-eszközből érkező adatokat tárolja és továbbítja az alkalmazás számára. Az előkészületek és beállítások a következő lépéseken alapulnak:

1. Firebase-projekt létrehozása:

- A Firebase Console-on új projektet kell létrehozni.
- Az adatbázist be kell állítani „valós idejű adatbázis” módba.
- A hozzáférési szabályokat megfelelően kell konfigurálni (pl. nyilvános hozzáférés fejlesztési környezetben, biztonságos autentikációval a végleges verzióban).

2. API-kulcs beszerzése:

- Az Arduino vagy más eszköz csatlakoztatásához szükség van egy API-kulcsra, amely a Firebase Console-on érhető el.

3. Adatok szerkezete:

- A hierarchikus NoSQL-adatbázisban az adatok kulcs-érték párként tárolódnak. Például: json

4. Valós idejű adatküldés:

- Az eszközökről érkező adatokat valós időben küldi el a rendszer a Firebase-nek, amely azonnal frissíti a kliensoldali megjelenítést.[31]

Felhasználási tapasztalatok

A Firebase lehetővé teszi, hogy a kaptármérő rendszer valós időben gyűjtse és jelenítse meg az adatokat egy webszerveren. Az SDK egyszerűsége miatt gyorsan és hatékonyan tudtam integrálni a platformot az Arduino Nano-mikrokontrollerrel és a SIM800L-modullal. A gyors

szinkronizálás és a pushértesítések révén azonnal értesülök a kaptár környezetében bekövetkező változásokról.

1.4. MIT App Inventor: történeti háttér és működés

Az MIT App Inventor egy olyan fejlesztési platform, amely lehetővé teszi Android- és iOS-alkalmazások gyors és egyszerű készítését, különösen azok számára, akik nem rendelkeznek programozási tapasztalattal. Az eszköz története és működése egyaránt izgalmas példája annak, hogyan teszik elérhetővé a technológiai innovációkat szélesebb közönség számára.

Történeti háttér

Az MIT App Inventor fejlesztését a Google kezdte meg 2010-ben, azzal a céllal, hogy egy könnyen használható szoftvert biztosítson a mobilalkalmazások fejlesztésére. 2011-ben az MIT átvette a projektet, és a Massachusetts Institute of Technology egyik programja, az MIT Media Lab részeként folytatta a fejlesztést. Az App Inventor azóta is folyamatos támogatást és frissítéseket kap, hogy lépést tartson a modern alkalmazásfejlesztési igényekkel.

Az App Inventor célja az volt, hogy „demokratizálja” a szoftverfejlesztést, vagyis bárkinek lehetőséget adjon az alkalmazásfejlesztés kipróbálására, beleértve a fiatalokat, tanárokat és hobbifejlesztőket is. Az egyszerű blokkalapú programozási nyelv elősegíti a kreativitást és az innovációt, így számos iskolában is használják oktatási céllal. [32]

Működése és jellemzői

1. Felhasználói felület:

- Az App Inventor két fő részből áll:
 - Designer-felület: Itt tervezhető meg az alkalmazás kinézete és elrendezése. A komponenseket, például gombokat vagy szövegdobozokat egyszerűen, húzással helyezhetjük el.
 - Blocks Editor: A logikai működés programozása itt történik, vizuális blokkok segítségével, amelyek egymáshoz kapcsolhatók.

2. Valós idejű tesztelés:

- Az „MIT AI2 Companion”-mobilalkalmazással a fejlesztés alatt álló app valós időben tesztelhető a telefonon. Ehhez csak QR-kód beolvasása szükséges, ami egyszerűsíti a hibakeresést és a tesztelést.

3. Publikálás:

- Az alkalmazások APK-formátumba exportálhatók, így telepíthetők Android-eszközökre. [33]

Miért választottam az MIT App Inventort?

Az MIT App Inventor:

- Könnyen használható, még kezdők számára is.
- Gyorsan összekapcsolható a Firebase-szel.
- Vizualitása és átlátható működése miatt ideális volt a projekt számára.

Egy kedves véletlen egybeesés, hogy az MIT App Inventor logójában egy méhecske található, ami szimbolikusan kapcsolódik a méhészeti témájú projektemhez.



22. ábra. Az MIT App Inventor logója[32]

2. Hardveres megvalósítás

Szenzorok és modulok kapcsolása

A rendszerben használt szenzorokat és modulokat úgy kell csatlakoztatni az Arduino Nanohoz, hogy azok stabilan működjenek és megbízható adatokat szolgáltatassanak.

Az alábbiakban felsorolom az egyes szenzorokat és modulokat, valamint a csatlakoztatásukhoz szükséges pineket.

1. DHT11 (Hőmérséklet- és páratartalom-érzékelő):

- **DATA pin:** Az Arduino Nano 4-es digitális pinjéhez csatlakozik.
- **VCC és GND:** Az Arduino 5V és GND-pinjéhez csatlakozik.

2. HX711 (Tehercella-érzékelő):

- **DT és SCK pin:** Az Arduino Nano digitális 2-es és 3-as pinjeire csatlakozik.

- **VCC és GND:** 5V és GND csatlakoztatása.
- **Tehercella:** A HX711-hez csatlakoztatva mér tömegadatokat.

3. SIM800L (GSM/GPRS-modul):

- **TX- és RX-pin:** Az Arduino Nano 7-es (RX) és 6-os (TX) pinjeire csatlakozik.
- **VCC és GND:** A modul 5V-os tápellátást igényel, ezért feszültségstabilizáló szükséges (LM2596 DC-DC-konverter).

4. LCD-kijelző (I2C-kommunikáció):

- **SDA és SCL:** Az Arduino Nano A4 (SDA) és A5 (SCL) pinjeire csatlakozik.
- **VCC és GND:** 5V és GND csatlakoztatása.

5. Gázérzékelő (MQ-2):

- **Analog Output (AO):** A0 analóg pinjére csatlakozik.
- **VCC és GND:** 5V és GND csatlakoztatása.

6. KY-038 Hangérzékelő:

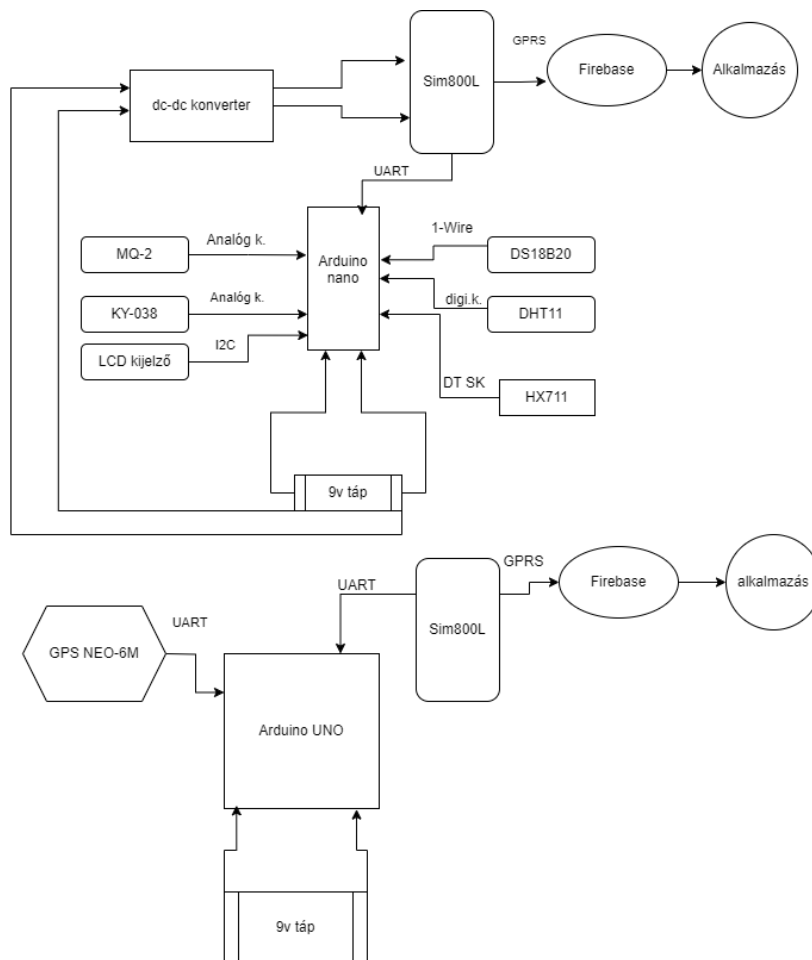
- **AO:** Az Arduino A1 analóg pinjére csatlakozik.
- **VCC és GND:** 5V és GND csatlakoztatása.

7. DS18B20 (Digitális hőmérséklet-érzékelő):

- **DATA pin:** Az Arduino Nano 5-ös digitális pinjéhez csatlakozik.
- **VCC és GND:** 5V és GND csatlakoztatása.

8. Gomb:

- **Csatlakozás:** Az Arduino Nano 9-es digitális pinjéhez csatlakozik.



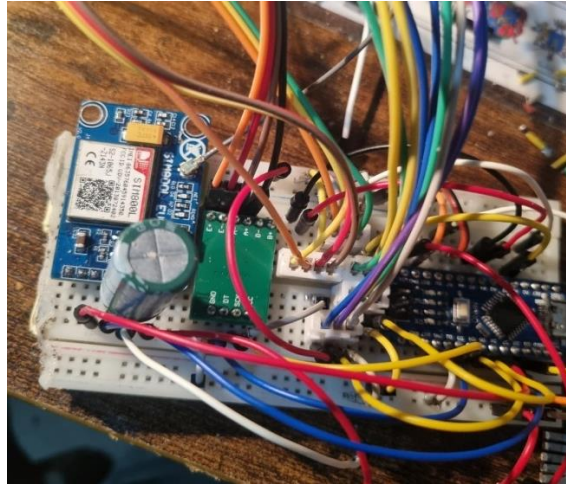
23. ábra. Blokkvázlat a rendszer felépítéséről

Az áramkörök felépítése és tervezési szempontok

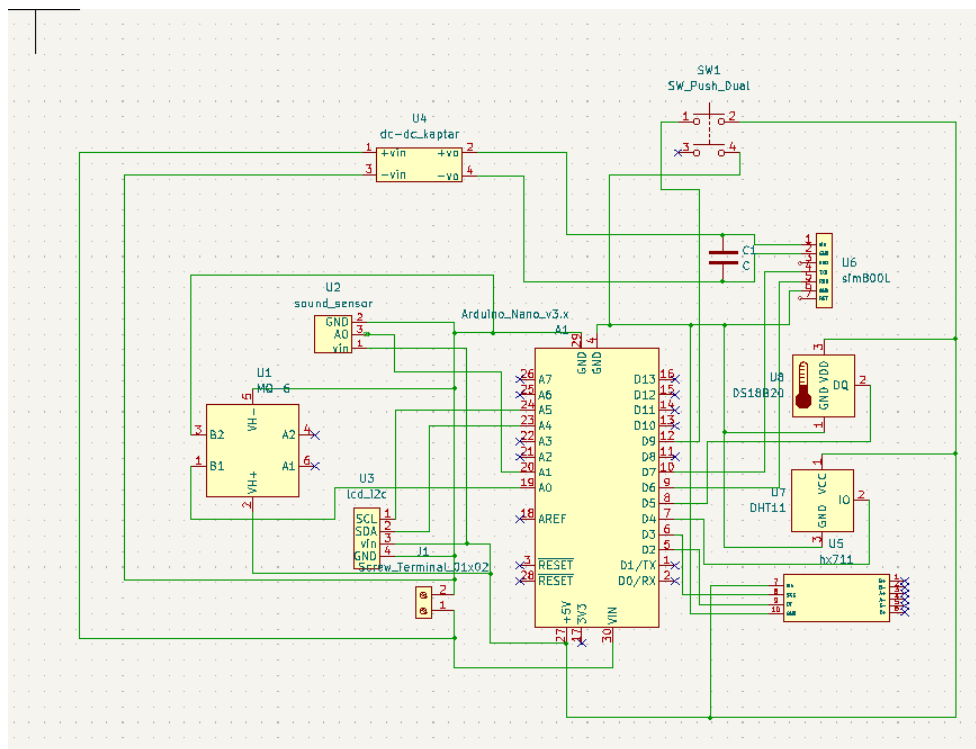
A kaptármérő rendszer áramköreinek tervezése során figyelmet fordítottam a szenzorok és modulok feszültségigényére, valamint a megfelelő huzalozás kialakítására, hogy a rendszer stabilan és megbízhatóan működjön. A fejlesztési folyamat több szakaszban zajlott, a kezdeti kísérletezéstől a végleges NYÁK kialakításáig.

2.1. Kezdeti kísérletek breadboardon

Az áramkör prototípusát breadboardon építettem meg, hogy tesztelhessem az összes szenzort és modult. A komponenseket jumperkábelekkel kötöttem össze az Arduino Nano megfelelő pinjeivel. A tesztelés során ügyeltem arra, hogy a huzalozás később is könnyen alkalmazható legyen a végleges áramkörben.



24. ábra. A rendszer összeállítása breadboardon



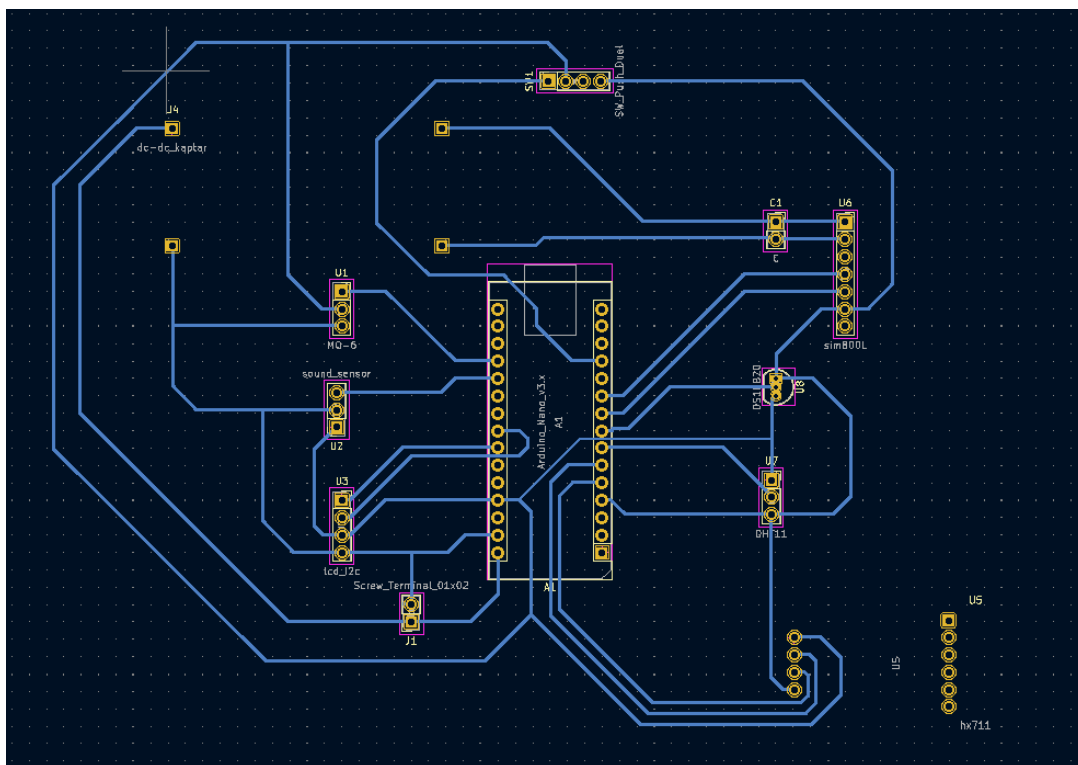
25. ábra. KiCadban készült schematik a rendszer kapcsolásáról[34]

Prefboard-kísérletek

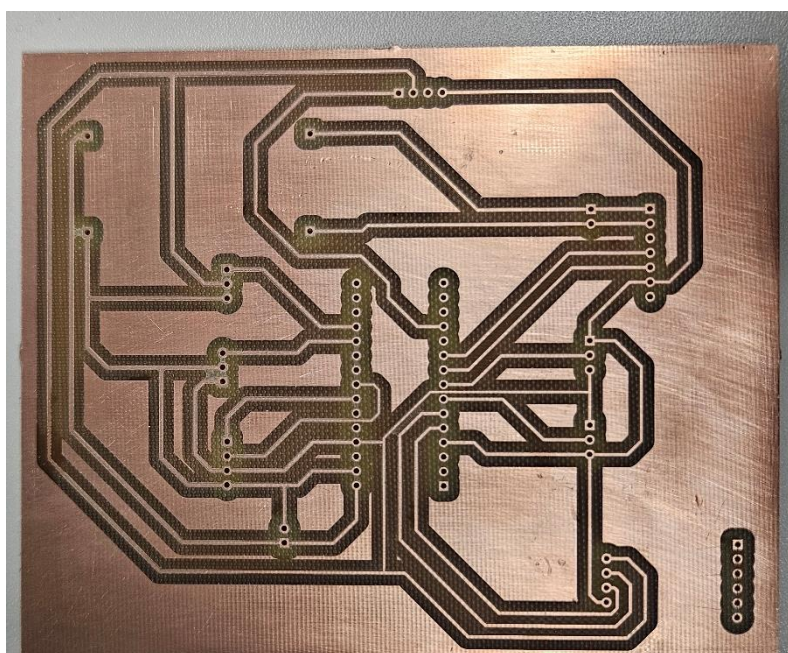
Miután a breadboardon stabilan működött a rendszer, egy prefboardon próbáltam meg kialakítani az áramkört. Ez azonban nem bizonyult megfelelőnek, mivel a prefboard mérete nem volt megfelelő, és a kapcsolás túl bonyolulttá vált. Az eszközök és forrasztási technikák korlátozottsága miatt a prefboardos megoldás nem felelt meg a projekt követelményeinek.

2.2. NYÁK-tervezés és -kivitelezés

A végleges áramkör megtervezéséhez KiCad-szoftvert használtam, amellyel elkészítettem a kapcsolási rajzt és a nyomtatott áramkör (NYÁK) terveit. A terveket az egyetem marógépe készítette el, amely precízen kimarta az áramkör vezetősávjait egy rézlemezre. Ez a módszer jelentősen megkönnyítette az áramkör pontos kialakítását és a későbbi összeszerelést.



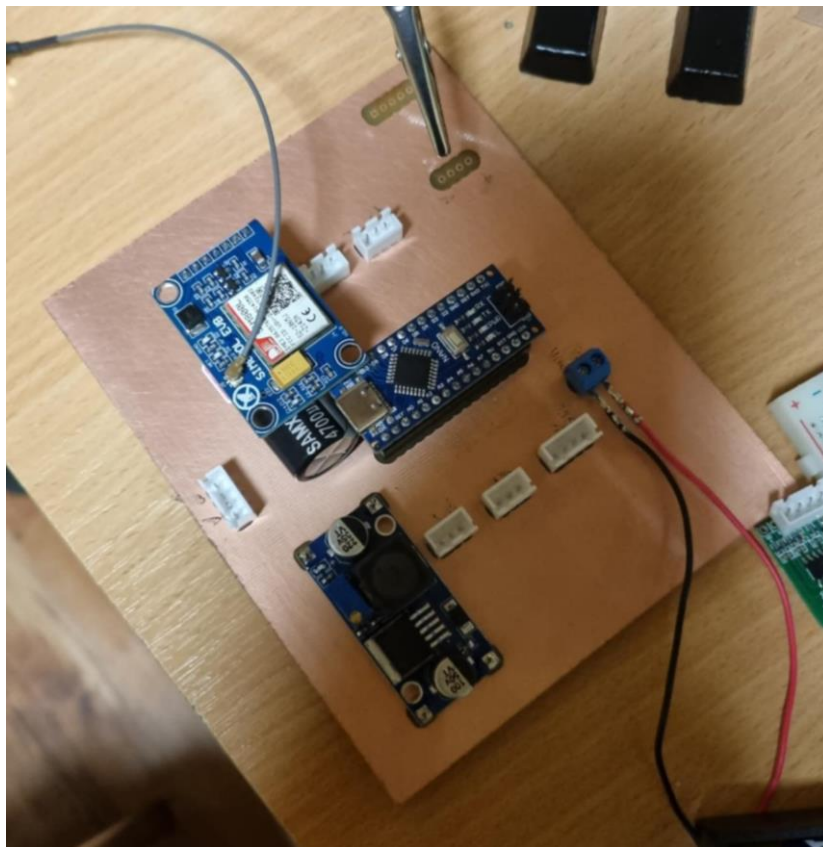
26. ábra. NYÁK-terv PCB-editorban[35]



27. ábra. A kimart NYÁK

Összeszerelés a NYÁK-on

1. **Kis alkatrészek forrasztása:** A legkisebb SMD-alkatrészek kerültek először a NYÁK-ra, hogy megkönnyítsék a forrasztási folyamatot.
2. **Tüskesorok használata:** Az Arduino Nano socketjeire tüskesorokat forrasztottam, így az Arduino könnyen eltávolítható maradt a későbbi programfrissítésekhez vagy meghibásodás esetére.
3. **Fixen forrasztott komponensek:** Az olyan komponensek, mint az **LM2596 DC-DC-konverter** és a **4700 μ F-os kondenzátor** közvetlenül a NYÁK-ra kerültek, mivel ezek állandó elemei a rendszernek.
4. **Csatlakozók alkalmazása:** JST-csatlakozókat forrasztottam a szenzorok számára, hogy azokat könnyen lehessen csatlakoztatni vagy eltávolítani. Hasonlóképpen, a gombhoz is egy JST-csatlakozót használtam, hogy szükség esetén könnyen cserélhető legyen.

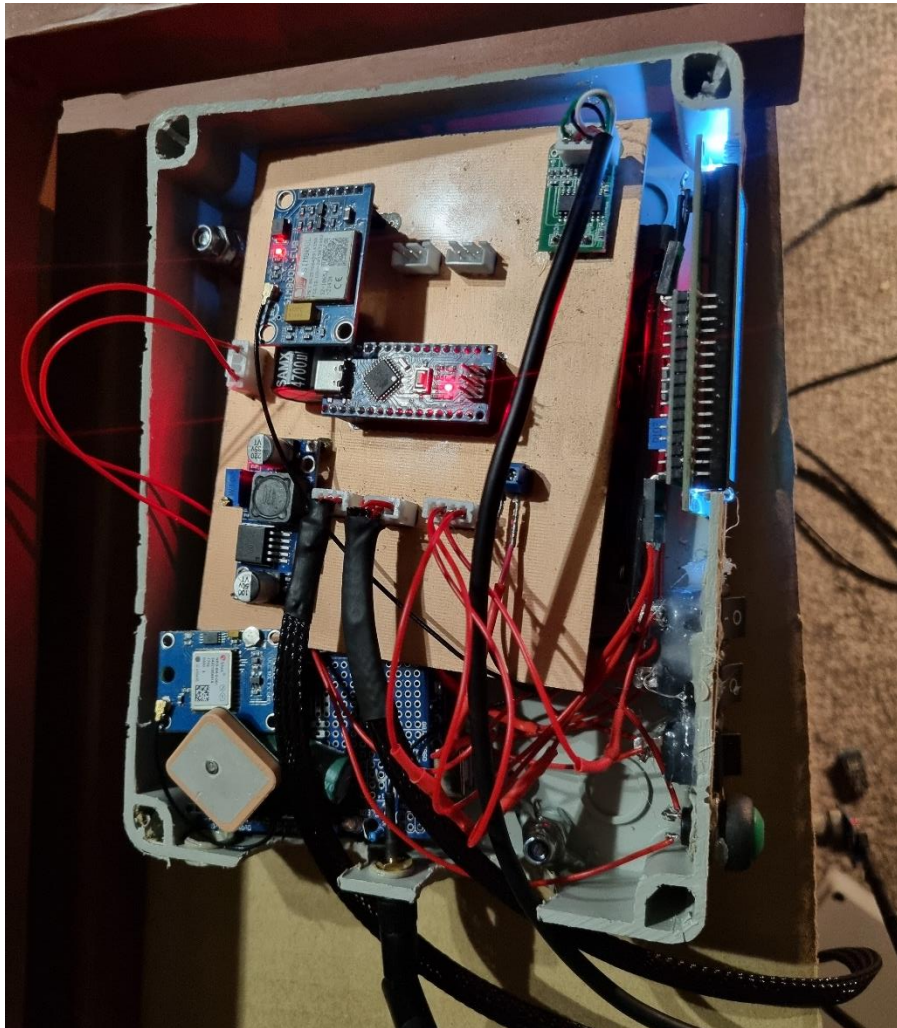


28. ábra. Az összeforrasztott NYÁK

2.3. Tápegység és burkolat

- Az áramellátást egy 6 db 1,5V-os elemet tartalmazó elemdoboz biztosítja, amelyet egy screw terminal segítségével kapcsoltam a NYÁK-hoz.

- Az egész áramkört és a tápegységet egy 20x15x6 cm-es műszerdobozba szereltem, amely elég nagy volt ahhoz, hogy kényelmesen elférjen benne az összes komponens. A dobozon lyukakat vágtam a szenzorok kábeleinek, az antennának, az LCD-kijelzőnek és a kapcsolóknak. A kapcsolók közül az egyik az egész rendszer áramellátását szabályozza, a másik az LCD-kijelző működését vezérli. A gomb és a kapcsolók stabilan rögzítésre kerültek a doboz előlapján.



29. ábra. A műszerdobozba integrált hardverrendszer

2.4. Váz tervezése és kivitelezése

A rendszer váztervét FreeCAD-szoftver segítségével alkottam meg. A fémváz két részből áll: egy felső tartórészből, amely közvetlenül a kaptár alatt van, és egy alsó alátámasztórészből, amely a talajon nyugszik. A két részt egy 200 kg teherbírású mérőcella köti össze, amely a kaptár tömegét méri.

A vázat fémlemezről hegesztettem össze, a mérőcella stabilitását pedig négy csavarral biztosítja. A műszerdobozt az alsó vázra rögzítettem, így a rendszer szállításkor és telepítéskor is stabil marad.

Ez a gondosan megtervezett és kivitelezett hardveres rendszer megbízható alapot biztosított a szenzorok és modulok működéséhez, valamint a terepen való használathoz.



30. ábra. A meghegesztett váz tesztelése

2.5. GPS-rendszer

A kiegészítő GPS-rendszert külön mikrokontroller, egy Arduino Uno vezérli, és egyszerűbb kialakítású, mint a kaptármérő rendszer. Ez a moduláris rendszer kizárólag a helymeghatározás funkciójára koncentrál, és nem foglal el nagy helyet, mivel kevesebb komponenst alkalmaz.

Rendszerösszetevők

A helymeghatározó rendszer három fő elemből áll:

1. GPS Neo-6-modul:

- A helymeghatározást végzi, és a hosszúsági (longitude) és szélességi (latitude) adatokat biztosítja.
- A pontos jel vételéhez egy külön antenna szükséges, amely biztosítja a stabil műholdkapcsolatot.

2. SIM800L-modul:

- Az adatokat GPRS-kapcsolaton keresztül továbbítja a Firebase-szerverre, ugyan úgy, mint ahogy a kaptármérő rendszer a szenzorok adatait.
- A stabil működés érdekében itt is szükség volt egy nagy kapacitású kondenzátor beépítésére, hogy támogassa a hálózati csatlakozást, amely rövid idő alatt jelentős energiát igényel.

Összeszerelés a Proto Shielden

A GPS-rendszerhez nem terveztem külön NYÁK-ot, mivel egyszerűbb megoldásként a Proto Shieldet választottam, amely közvetlenül az Arduino Unora helyezhető, és lehetővé teszi az alkatrészek rögzítését.

• **Forrasztási folyamat:**

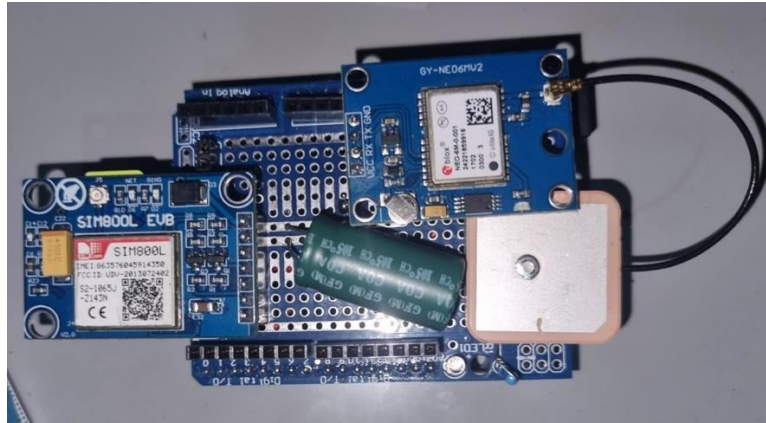
1. A Proto Shieldre először tűskesorokat forrasztottam, hogy a modulokat ne kelljen fixen rögzíteni.
2. A szükséges vezetékeket a tűskesorok megfelelő pontjaihoz forrasztottam, összekötve a GPS-modul, a SIM800L-modul, és az Arduino pinjeit.
3. Egy 2200 µF-os kondenzátort is a Proto Shieldre forrasztottam, amely biztosította a SIM-modul stabil energiaellátását.

Kivitelezés és tesztelés

Az összeszerelt rendszert teszteltem:

- Ellenőriztem, hogy a GPS-modul pontos helymeghatározást végezzen.
- Meggyőződtem arról, hogy a SIM800L-modul stabilan továbbítja a GPS-koordinátákat az adatbázisba.

Miután megbizonyosodtam a rendszer stabil működéséről, a kész egységet egy a műszerdobozba helyeztem. Egy kapcsolót is beépítettem, amely lehetővé teszi a rendszer ki- és bekapcsolását.



31. ábra. A GPS-rendszer

Integráció a kaptármérő rendszerhez

A GPS-rendszer nem része közvetlenül a kaptármérő áramkörnek, hanem önálló egységként működik. Az egyszerűbb felépítésnek és a Proto Shield használatának köszönhetően kompakt és költséghatékony megoldást nyújt a rendszer pontos helyzetének követésére, amely különösen fontos vándoroltatás vagy esetleges lopás esetén. Ez a rendszer tehát jól kiegészíti a kaptármérő alapfunktcióit, növelve annak biztonsági és nyomkövetési képességeit.



32. ábra. Az összeszerelt intelligens kaptármérő rendszer.



33. ábra. Tesztelés egy méhkaptáron

3. Szoftveres felépítés

3.1. A kód struktúrája és működése

A rendszer működéséhez szükséges kód több részre osztható, amelyek közül mindegyik szorosan kapcsolódik a szenzorokhoz és az adatküldési folyamathoz:

1. **Inicializálás:** A `setup()`-függvényben inicializálja az összes szenzort és a SIM800L-modult. Beállítja az LCD-kijelzőt, az I2C-kommunikációt és a soros kapcsolatot a debug céljából. Itt történik a hálózathoz való csatlakozás és az APN beállítása is.
2. **Adatgyűjtés:** A `loop()`-függvény ciklikusan olvassa a szenzorok adatait. Az összes adatot egy-egy változóban tárolja, amelyeket majd a Firebase adatbázisába küldi.
3. **Adatfeldolgozás és megjelenítés:** Az adatok kijelzőn való megjelenítéséhez egy függvény felel, amely a szenzorok aktuális adatait jeleníti meg. A kijelzett adatok egy gombnyomással válthatók.

- 4. Adatküldés:** A Firebase adatbázisba történő adatküldés a TinyGSM-könyvtár segítségével valósul meg, amely a SIM800L-modult használja az adatátvitelhez.

Gombfunkciók implementálása

A gomb funkciói az alábbiak szerint működnek:

- **Adatmegjelenítés váltása:** A gomb megnyomásával a kijelzőn megjelenített adatokat lehet váltani (tehercella tömegértéke, külső és belső hőmérséklet, páratartalom, gáz- és hangérzékelő adatok).

3.2. Az adatküldés folyamata a Firebase-re

Az adatküldés a TinyGSM- és az ArduinoHttpClient-könyvtárak segítségével zajlik.

- 1. APN beállítása és csatlakozás:**

A SIM800L-modul inicializálása és az APN beállítása biztosítja az adatkapcsolatot. A TinyGSM-könyvtár lehetőséget ad a hálózati csatlakozásra, így az adatküldés előtt mindig ellenőrizhető a kapcsolat stabilitása.

- 2. Adatstruktúra előkészítése:**

A szenzorok által mért adatok változókbán kerülnek tárolásra. Ezeket az adatokat JSON-formátumra alakítva lehet továbbítani a Firebase-adatbázisba, így azok kompatibilisek a rendszerrel.

- 3. Adatküldés:**

Az ArduinoHttpClient-könyvtár segítségével a JSON-formátumú adatok HTTP PUT kéréssel kerülnek a Firebase-adatbázisba. Az adatok kulcs-érték párként tárolódnak, például a ds18b20Temperature-kulcs a DS18B20 hőmérséklet-érzékelő által mért adatot tartalmazza. A SIM800L-modul GSM-hálózaton keresztül csatlakozik a Firebase URL-hez, ami lehetővé teszi a valós idejű frissítést.

- 4. Hibaellenőrzés és újrapróbálkozás:**

Az adatküldés közbeni hibák, például gyenge hálózati kapcsolat esetén, a rendszer automatikusan megkísérli újraküldeni az adatokat, mielőtt hibát jeleznék.

3.3. Kódrészletek és funkciók

```
#define TINY_GSM_MODEM_SIM800
#include <TinyGsmClient.h>
#include <ArduinoHttpClient.h>
#include <DHT.h>
#include <SoftwareSerial.h>
#include <HX711.h>
#include <LiquidCrystal_I2C.h>
#include <DallasTemperature.h>
```

```
#include <OneWire.h>
```

A kód elején importált könyvtárak mindegyike egy adott modul vagy szenzor működtetéséhez szükséges. A **TinyGsmClient** a SIM800L GSM-modul vezérlésére szolgál, beleértve a hálózati kapcsolatok kezelését és az adatküldést. Az **ArduinoHttpClient** a HTTP-protokoll használatát biztosítja az adatok továbbítására a Firebase-adatbázisba. A **DHT**-könyvtár a DHT11 hőmérséklet- és páratartalom-érzékelő működését támogatja, míg a **HX711** a tehercella által szolgáltatott analóg jelek digitalizálásáért felel. A **LiquidCrystal_I2C**-könyvtár az I2C interfészű LCD-kijelző vezérlését végzi, amelyen a szenzoradatok jelennek meg. A **DallasTemperature**- és a **OneWire**-könyvtárak a DS18B20-hőmérséklet-érzékelő működéséhez szükségesek. A **SoftwareSerial** további soros kommunikációs csatornát biztosít a SIM800L-modulhoz használatához. Ezek a könyvtárak biztosítják a modulok és szenzorok működését és az adatok hatékony feldolgozását.

Pindefiníciók és alapvető beállítások

```
#define HX_DOUT 3
#define HX_SCK 2
#define DHT_PIN 4
#define DHT_TYP DHT11
#define DS18B20_PIN 5
#define GAS_PIN A0
#define SND_PIN A1
#define BTN_PIN 9

SoftwareSerial sim800(7, 6); // SIM800L TX és RX
HX711 scale(HX_DOUT, HX_SCK);
DHT dht(DHT_PIN, DHT_TYP);
LiquidCrystal_I2C lcd(0x27, 16, 2);
OneWire oneWire(DS18B20_PIN);
DallasTemperature sensors(&oneWire);
```

Ebben a részben történik a szenzorok és modulok csatlakoztatásához szükséges lábak inicializálása.

A **SoftwareSerial**-könyvtár segítségével a 7-es és 6-os lábakon keresztül valósul meg a kommunikáció a SIM800L GSM-modullal, lehetővé téve a hálózati adatátvitelt. A **HX711**-, **DHT**- és **DallasTemperature**-objektumok az adott szenzorok vezérlését végzik, míg a **LiquidCrystal_I2C** az I2C interfészű LCD-kijelző működtetéséhez szükséges. Ez a

konfiguráció biztosítja, hogy az eszközök helyesen csatlakozzanak az Arduino Nano-mikrokontrollerhez.

Globális változók és beállítások

```
int dispState = 0;
bool btnPressed = false;
unsigned long lastBtnPressTime = 0;
const unsigned long debounceDelay = 500;
unsigned long lastSensorUpdate = 0;
const unsigned long sensorDelay = 1000;
const unsigned long dataSendDelay = 10000;
unsigned long lastDataSendTime = 0;
unsigned long buttonPressStartTime = 0;
const unsigned long resetDelay = 10000;

const char* FIREBASE_HOST = "mitkaptar-default-rtdb.firebaseio.com";
const char* FIREBASE_AUTH = "iaRvYnxWTJ7PmaUbYUixHp6lM4CiysyhiONTvGI4";
const char* FIREBASE_PATH = "kaptarsensors";

char apn[] = "online";
TinyGsm modem(sim800);
TinyGsmClientSecure gsm_client_secure_modem(modem, 0);
HttpClient http_client = HttpClient(gsm_client_secure_modem, FIREBASE_HOST,
443);
```

Az itt definiált változók a gomb nyomásának idejét és a szenzorfrissítések, valamint adatküldések időzítését kezelik. A FIREBASE_HOST, FIREBASE_AUTH és FIREBASE_PATH változók a Firebase-adatbázis eléréséhez szükséges információkat tárolják.

setup()-függvény

A setup() inicializálja az összes eszközt és a kapcsolatot a Firebase-hez:

```
void setup() {
    Serial.begin(115200);
    dht.begin();
    sim800.begin(9600);
    modem.restart();
    lcd.begin(16, 2);
    lcd.backlight();
    sensors.begin();
}
```

```

    scale.set_scale(-19000.0);
    pinMode(BTN_PIN, INPUT_PULLUP);
    lcd.print("Inditas...");
    delay(2000);
    lcd.clear();
}

```

A **setup()**-függvény felelős az eszközök inicializálásáért és az alapvető beállítások elvégzéséért. Először beállítja a soros kommunikációt, majd inicializálja a DHT11-érzékelőt, a SIM800L-modemet és az LCD-kijelzőt. A DS18B20-hőmérséklet-érzékelőt és a HX711-tehercella-érzékelőt is előkészíti a méréshez. A gombot belső felhúzó ellenállással készíti elő. Végül egy kezdeti üzenetet jelenít meg az LCD-kijelzőn, jelezve, hogy a rendszer elindult.

loop()-függvény

A loop-függvény, amely a gombkezelést, a kijelző frissítését, a GPRS-kapcsolat ellenőrzését és az adatküldést tartalmazza:

```

void loop() {
    // Gomb nyomásának kezelése a kijelző váltására és tehercella
    nullázására
    if (digitalRead(BTN_PIN) == LOW) {
        if (!btnPressed) {
            btnPressed = true;
            buttonPressStartTime = millis();
        }
    } else {
        if (btnPressed) {
            if (millis() - buttonPressStartTime >= resetDelay) {
                scale.tare();
                lcd.clear();
                lcd.print("Tehercella nullazva");
                delay(2000);
                lcd.clear();
            } else {
                dispState = (dispState + 1) % 5;
            }
            btnPressed = false;
        }
    }

    updateDisplay();
}

```

```

// GPRS csatlakozás ellenőrzése
if (!modem.isGprsConnected()) {
    if (modem.gprsConnect(apn)) {
        Serial.println("GPRS csatlakozás sikeres.");
    } else {
        Serial.println("GPRS csatlakozás sikertelen.");
    }
}

// Adatküldés a Firebase-be
if (millis() - lastDataSendTime > dataSendDelay) {
    lastDataSendTime = millis();
    sensor_loop();
}
}

```

A loop()-függvény figyeli a gomb állapotát, váltja a kijelző tartalmát.

updateDisplay()-függvény

A kijelzőn megjelenített információk frissítése:

```

void updateDisplay() {
    float load = scale.get_units();
    float hum = dht.readHumidity();
    float temp = dht.readTemperature();
    sensors.requestTemperatures();
    float tempExt = sensors.getTempCByIndex(0);
    lcd.clear();
    switch (dispState) {
        case 0: lcd.print("Teher: "); lcd.print(load, 2); lcd.print(" kg");
        break;
        case 1: lcd.print("Kint: "); lcd.print(tempExt, 1); lcd.print("
C"); break;
        case 2: lcd.print("Bent: "); lcd.print(temp, 1); lcd.print(" C");
        break;
        case 3: lcd.print("Par: "); lcd.print(hum, 1); lcd.print(" %");
        break;
    }
}
}

```

A dispState változó alapján az LCD-kijelzőn megjeleníti a tehercella, a külső és belső hőmérséklet, illetve a páratartalom értékeit.

sensor_loop()-függvény

Szenzoradatok begyűjtése és formázása Firebase-be történő küldésre:

```
void sensor_loop() {
    float dhtHumidity = dht.readHumidity();
    float dhtTemperature = dht.readTemperature();
    int gasLevel = analogRead(GAS_PIN);
    int soundLevel = analogRead(SND_PIN);
    float load = scale.get_units();
    sensors.requestTemperatures();
    float ds18b20Temp = sensors.getTempCByIndex(0);

    String data = "{";
    data += "\"dhtTemperature\":\"" + String(dhtTemperature) + ",";
    data += "\"dhtHumidity\":\"" + String(dhtHumidity) + ",";
    data += "\"load\":\"" + String(load) + ",";
    data += "\"gasLevel\":\"" + String(gasLevel) + ",";
    data += "\"soundLevel\":\"" + String(soundLevel) + ",";
    data += "\"ds18b20Temperature\":\"" + String(ds18b20Temp) + "}";

    PutToFirebase(FIREBASE_PATH, data, &http_client);
}
```

PutToFirebase()-függvény

Az adatokat JSON-formátumban továbbítja a Firebase-be.

```
void PutToFirebase(const char* path, const String& data, HttpClient* http)
{
    String url = "/" + String(path) + ".json?auth=" + FIREBASE_AUTH;
    int responseCode = http->put(url, "application/json", data);
    Serial.print("Válasz kód: ");
    Serial.println(responseCode);
}
```

Ez a függvény egy HTTP PUT-kérést küld a Firebase-adatbázisba.

3.4. GPS-helymeghatározás kódja

A GPS-rendszer célja, hogy pontos koordinátákat (szélesség és hosszúság) szolgáltatson a kaptármérő rendszer helyéről, és azokat valós időben továbbítsa a Firebase adatbázisába. Ez a funkció lehetővé teszi a rendszer helyzetének követését. A kód TinyGSM- és TinyGPS++-könyvtárakat használ, amelyek a GSM- és GPS-modulok kezeléséért felelősek.

A kód bemutatása

A kód négy fő szakaszra bontható: könyvtárak importálása, eszközök inicializálása, helyadatok olvasása és adatok küldése a Firebase-be.

Könyvtárak és definíciók

```
#define TINY_GSM_MODEM_SIM800
#include <TinyGsmClient.h>
#include <ArduinoHttpClient.h>
#include <SoftwareSerial.h>
#include <AltSoftSerial.h>
#include <TinyGPS++.h>
```

- **TinyGsmClient:** Ez a könyvtár kezeli a GSM/GPRS-modult, lehetővé téve a hálózati kommunikációt, például a Firebase-re való adatküldést.
- **ArduinoHttpClient:** A HTTP-protokollt használó kérések kezelésére szolgál, például a Firebase adatbázisába való írásra.
- **SoftwareSerial:** Alternatív soros kommunikációt biztosít a GSM-modul számára.
- **AltSoftSerial:** A Neo 6M GPS-modulhoz használatos, amely pontosabb időzítést biztosít a soros kommunikációhoz.
- **TinyGPS++:** Ez a könyvtár a GPS-modul által küldött NMEA-adatokat dekódolja, és ember által olvasható formátumba alakítja (koordináták).

3.5. Eszközök inicializálása

```
SoftwareSerial Sim800L(2, 3);
AltSoftSerial neogps;
TinyGsm modem(Sim800L);
TinyGsmClientSecure gsm_client_secure_modem(modem, 0);
HttpClient http_client = HttpClient(gsm_client_secure_modem, FIREBASE_HOST,
443);
TinyGPSPlus gps;

const char FIREBASE_HOST[] = "intelligenskaptarmerol-default-
rtddb.firebaseio.com";
const String FIREBASE_AUTH = "Hxobgk9OzzyaOh3Tgl76Nf3oQFLYNdx5KegDv";
const String FIREBASE_PATH = "/GPS";
char apn[] = "online";
```

A setup()-függvény

```
void setup() {  
    Serial.begin(115200);  
    Sim800L.begin(9600);  
    neogps.begin(9600);  
    modem.restart();  
    http_client.setHttpResponseTimeout(10 * 1000);  
}
```

- Itt a soros kommunikáció inicializálása történik mind az USBport, mind a GSM- és GPS-modulok számára.
- A modem.restart()-parancs újraindítja a GSM-modult, hogy biztosítsa az alapról induló működést.
- A HTTP-válaszidőt 10 másodpercre állítottam be a stabil adatátvitel érdekében.

A loop()-függvény

```
void loop() {  
    if (!modem.gprsConnect(apn, "", "")) {  
        delay(1000);  
        return;  
    }  
    Serial.println(F("GPRS OK"));  
    http_client.connect(FIREBASE_HOST, 443);  
    while (true) {  
        if (!http_client.connected()) {  
            http_client.stop();  
            Serial.println(F("HTTP not connect"));  
            break;  
        } else {  
            while (neogps.available() > 0) {  
                if (gps.encode(neogps.read())) {  
                    Lat = String(gps.location.lat(), 6);  
                    Lag = String(gps.location.lng(), 6);  
                    String Data = "{\"Lat\":\"" + Lat + "\",\"Lag\":\"" + Lag + "\"}";  
                    PostToFirebase("PATCH", FIREBASE_PATH, Data, &http_client);  
                    delay(1000);  
                }  
            }  
        }  
    }  
}
```

- **GPRS-kapcsolat létrehozása:** A `modem.gprsConnect()` ellenőrzi és beállítja a mobilhálózati kapcsolatot. Ha a kapcsolat sikertelen, újrapróbálkozás történik.
- **GPS-adatok olvasása:** A GPS-modul által küldött adatok feldolgozása a `gps.encode()` segítségével történik. A koordinátákat a `gps.location.lat()` és `gps.location.lng()` függvények adják vissza.
- **Adatok küldése:** A Firebase-adatbázisba történő adatküldés a `PostToFirebase()`-függvénnyel történik.

Adatok küldése a Firebase-re

```
void PostToFirebase(const char* method, const String& path, const String&
data, HttpClient* http) {
    String url = path + ".json?auth=" + FIREBASE_AUTH;
    http->put(url, "application/json", data);
    Serial.println("Response: " + String(http->responseBody()));
    if (!http->connected()) {
        http->stop();
        Serial.println(F("HTTP Not Connected"));
    }
}
```

- A `PostToFirebase()`-függvény összeállítja a JSON-formátumú adatokat és elküldi azokat a Firebase-adatbázisba. A `put()`-módszer frissíti a meglévő adatokat.

Ez a GPS helymeghatározó kód egy kompakt és hatékony megoldás arra, hogy valós időben küldjük el a rendszer pozícióját. A kód jól optimalizált, és a TinyGSM-, valamint TinyGPS++-könyvtárak használatával stabil és megbízható működést biztosít. Az ilyen típusú kód kiválóan illeszkedik a méhészeti alkalmazásokhoz, különösen a kaptár helyzetének nyomon követésében.

4. Firebase Realtime Database működése és használata

A **Firebase Realtime Database** egy felhőalapú NoSQL-adatbázis, amely JSON-formátumban tárolja az adatokat és biztosítja azok valós idejű frissítését. A Firebase egy teljeskörű platformot kínál mobil- és webalkalmazások fejlesztéséhez, és az egyik legfontosabb funkciója a valós idejű adatkezelés.

4.1. Firebase-adatbázis működése

1. **Adatok tárolása JSON-formátumban:** Az adatbázis szerkezetét úgy kell elképzelni, hogy a különböző szenzorok adatainak nevei kulcsokként, míg azok értékei adatként vannak tárolva.

"GPS":

```
"Latitude": 47.109066,  
"Longitude": 21.213774},
```

"kaptarsensors":

```
"dhtHumidity": 65,  
"dhtTemperature": 11,  
"ds18b20Temperature": 15.56,  
"gasLevel": 65,  
"load": 51.09,  
"soundLevel": 192
```

2. **Adatok frissítése:** Mivel a Firebase Realtime Database valós időben szinkronizálja az adatokat, minden egyes módosítás azonnal megjelenik az összes csatlakoztatott kliens számára. Ha például egy új adat érkezik, az nemcsak a szerveren kerül mentésre, hanem minden olyan eszköz képernyőjén is, amely figyeli ezt az adatot.
3. **API használata:** A Firebase biztosít egy REST API-t, valamint SDK-kat különböző platformokra, például Androidra és iOS-re, amelyek lehetővé teszik a mobilalkalmazások számára, hogy adatokat küldjenek és fogadjanak az adatbázisból.

4.2. Az alkalmazás célja és felépítése

Az alkalmazás két oldalon működik:

- **Kaptármérő rendszer oldala:** Az első oldal a kaptármérő rendszer szenzorai által gyűjtött adatokat jeleníti meg, amelyeket a Firebase Realtime Database-ből olvas.
- **Kaptár helymeghatározó oldala:** A második oldal a kaptár aktuális GPS-koordinátáit (hosszúsági és szélességi értékeket) jeleníti meg egy térképen, ahol a marker automatikusan frissül a Firebase-ből érkező új adatok alapján.

Mindkét oldal között navigációs gombok biztosítják a váltást.

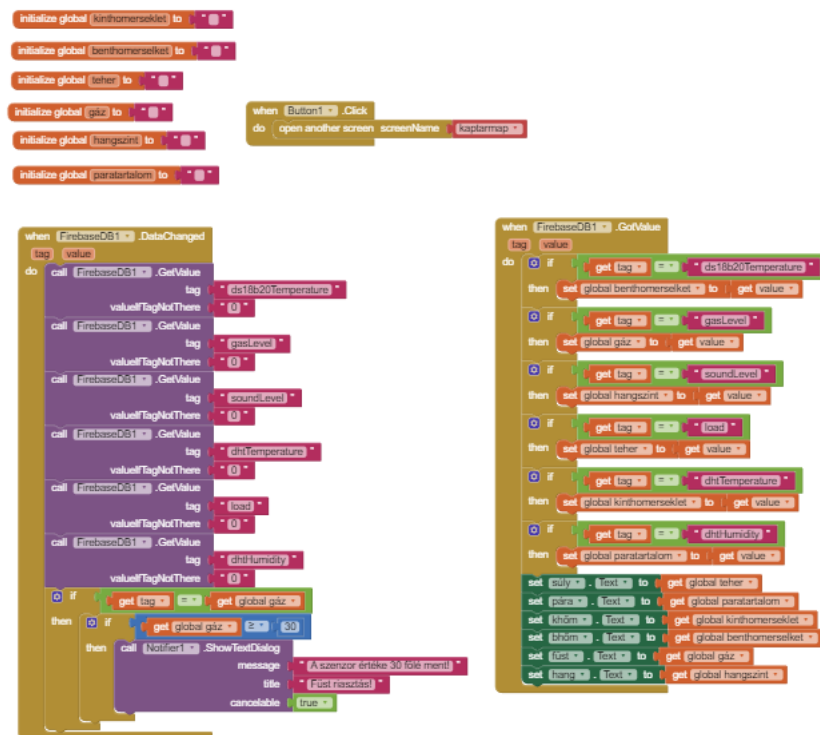
Első oldal – Kaptármérő rendszer adatainak megjelenítése

Felépítés és funkciók

- Label(címke)-komponensek:

- Minden szenzorhoz létrehoztam két címkét: az egyik a szenzor nevét, a másik az aktuális mérési értékét jeleníti meg.
- **Firestore-komponens:**
 - Ez a láthatatlan (nem megjelenített) komponens felelős az adatok valós idejű lekérdezéséért és frissítéséért a Firestore Realtime Database-ből.
 - A Firestore-komponens kapcsolódik a projekthez tartozó adatbázishoz, ahol az érzékelők adatai tárolódnak.
- **Clock-komponens:**
 - Ez a szintén láthatatlan komponens időzített események futtatására alkalmas. Az alkalmazásban például az adatok rendszeres lekérdezésére vagy automatikus frissítésére használható.
- **Notifier-komponens:**
 - Ez az értesítések megjelenítésére szolgál. Beállítottam, hogy amikor a füstszint meghaladja a 30-as értéket, az alkalmazás egy figyelmeztetést küldjön "Füstriasztás" üzenettel.

Blocks-program



34. ábra. Block-program a szenzor adatainak megjelenítéséhez[32]

- **Globális változók:**

- Minden szenzor adatához egy globális változót hoztam létre az Initialize Global-blokkal. Ez biztosítja az adatok tárolását és a felület különböző részein való elérhetőségét.

- **Adatok beolvasása a Firebase-ből:**

- A When FirebaseDB.GotValue-blokk segítségével beállítottam, hogy a Firebase-ből érkező adatok automatikusan frissítsék a megfelelő Label-komponensek szövegét.

- **Riasztás aktiválása:**

- Feltételként megadtam, hogy ha a füstszint 30 fölé emelkedik, a Notifier.ShowDialog-blokkon keresztül figyelmeztetést küldjön a felhasználó telefonjára.

- **Navigáció:**

- Az első oldalon egy gomb található, amely a második oldalra irányítja a felhasználót.

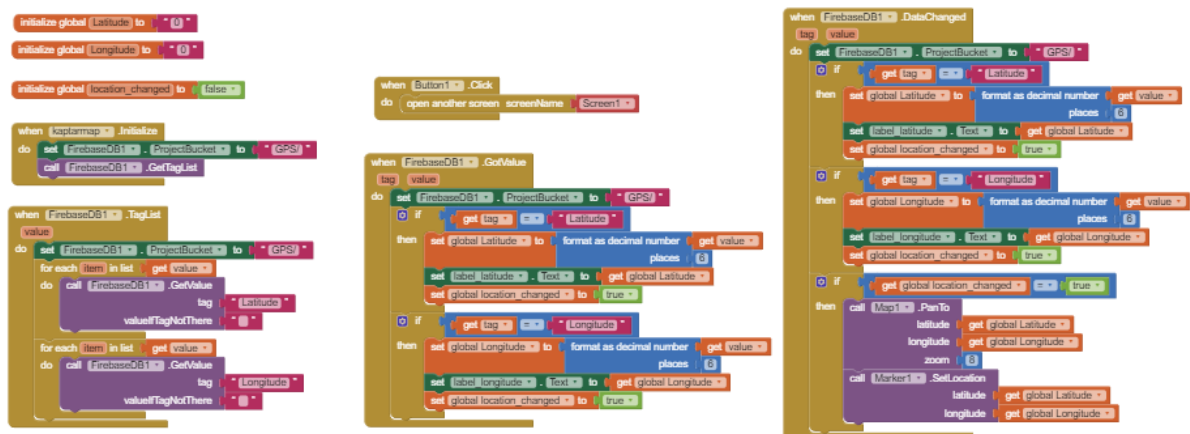
4.3. Második oldal – GPS-helymeghatározó

Felépítés és funkciók

- **Térkép és markerkomponensek:**

- Behúztam a Map-komponenst, amely egy 2D-s térkép.
- A marker az aktuális hosszúsági (latitude) és szélességi (longitude) értékeket mutatja a térképen.

Blocks-program



35. ábra. Blockprogram a helymeghatározáshoz[32]

- **Firestore-adatok frissítése:**
 - A When FirebaseDB.GotValue-blokk itt is felelős az adatok frissítéséért. Amikor a GPS-koordináták változnak az adatbázisban, a megfelelő címkék és a marker pozíciója is frissül.
- **Marker pozíciójának beállítása:**
 - A Set Marker.Latitude és Set Marker.Longitude-blokkok beállítja a marker új helyzetét a térképen. Ez biztosítja, hogy a térképen mindig a kaptár aktuális helyzete legyen látható.
- **Navigáció:**
 - Az oldalon található gomb lehetővé teszi a visszatérést az első oldalra.

4.4. Az alkalmazás működése

- Indításkor az alkalmazás a Firestore-ből lekéri a szenzorok aktuális értékeit és frissíti az első oldalon található címkéket.
- Ha a füstszint meghaladja a 30-as értéket, a telefon értesítést küld a felhasználónak.
- Az első oldalról egy gomb segítségével a második oldalra navigálhat a felhasználó, ahol a kaptár GPS-koordinátái jelennek meg. Az aktuális helyzetet egy marker mutatja a térképen, amely automatikusan frissül a Firestore-adatok alapján.



36. ábra. Telefonos alkalmazás az adatok megjelenítésére[32]

Az alkalmazás logikus és átlátható struktúrával rendelkezik. A Firebase biztosítja az adatok valós idejű szinkronizálását; a Clock az időzítéseket, a Notifier pedig az értesítéseket kezeli. Az első oldal a kaptármérő adatok bemutatására, a második pedig a helymeghatározásra fókuszál. Az alkalmazás intuitív navigációval köti össze a két oldalt, így felhasználóbarát módon szolgálja a kaptármérő rendszer működését.

5. Méréseredmények

5.1. A szenzorok által mért adatok elemzése

A rendszerben használt szenzorok mindegyike különböző típusú adatokat mér, és minden szenzor saját, specifikus jellemzőkkel rendelkezik. Az alábbiakban részletesen ismertetem, hogyan történik a különböző szenzorok adatainak mérése, azok kalibrálása és ellenőrzése:

1. **Hangérzékelő (Méhzúgás-mérés):** A hangérzékelő egy olyan analóg érzékelő, amely a környezeti zajokat érzékeli, és elektromos jelet generál. A méh zúgásának mérésére az érzékelőt különböző helyzetekben teszteltem. Különböző méh zúgásokat, mint például rajzási és nyugodt állapotú hangokat játszottam le, hogy lássam, milyen értéket mutat a szenzor a különböző zajforrások esetén. Az érzékelő nem rendelkezik mérési egységekkel (pl. dB), de a szenzor kimeneti jele arányos az érzékelt zaj intenzitásával.
 - **Kalibrálás:** A kalibrálás során megtudtam, hogy a mért értékek mit jelentenek, és azokat szakaszokra bontottam. A folyamat során a telefonom segítségével különböző méh zúgás-hangokat játszottam le a szenzor mellett, amelyek a méhek különböző aktivitásait imitálták, például a nektárhordás és a rajzás állapotát. Ennek eredményeként meghatároztam, hogy 540-es érték a teljes csendet jelzi, 550 fölött a méhek mozogni és hordani kezdenek, míg 560-as vagy annál magasabb érték rajzásra utalhat. Ez a módszer lehetővé tette, hogy pontosan kalibráljam az érzékelőt, és az adatokból megbízható következtetéseket vonjak le a méhek aktuális állapotáról.
2. **Füstjelző (Füstmérés):** A füstjelző szenzorokat olyan gázok és részecskék érzékelésére tervezték, mint a füst, amit a levegőben lévő apró részecskék jelenléte vált ki. A füstjelzők nem rendelkeznek közvetlen mérési egységgel, hanem egy viszonylag magas értéket adnak akkor, ha a levegőben a füst koncentrációja meghalad egy bizonyos szintet.

- **Kalibrálás:** A füstjelző kalibrálásához egy egyszerű módszert alkalmaztam: egy elfűjt gyertyát helyeztem a szenzor közelébe, hogy megnézzem, hogyan reagál a rendszer a füst jelenlétére. Az érzékelő értéke általában nem lineárisan változik, de az értékek közötti változások elegendőek voltak ahhoz, hogy meghatározzam, mikor aktiválódik a füstérzékelés. A kísérletből kiderült, hogy 30-as értéknél jelez be a füstre a szenzor.
- 3. **Tehercella:** A tehercella egy érzékelő, amely a tömeget méri. A tehercellák leggyakrabban analóg kimeneti jelet generálnak, amit digitális értékké alakítanak. A súly mérésekor az értékek megjelenítése nem mindig pontos, és ezért szükséges a kalibrálás.
 - **Kalibrálás:** A kalibrálás során egy házi mérleget használtam, hogy ellenőrizzem a szenzorok pontosságát. A kalibrálási értéket a programban állítottam be, és fokozatosan finomítottam a kalibrációt addig, amíg a tehercella által mért értékek deka pontossággal nem egyeztek meg a házi mérleg adataival.

4. **Külső hőmérséklet- és páratartalom-érzékelő:**

A DHT11-szenzor méri a kaptár környezetének hőmérsékletét és páratartalmát.

- **Kalibrálás:**

A szenzort hagyományos hőmérővel és páratartalom-mérővel összehasonlítva ellenőriztem.

A külső környezeti paraméterek ismerete segíti a méhészt abban, hogy megértse, miért csökkenhet az aktivitás. Magas páratartalom mellett a méhek nehezebben párologtatják el a nektár nedvességtartalmát, ami híg mézet eredményez, és ez csökkentheti a méz minőségét és értékét.

5. **Belső hőmérséklet-érzékelő:**

A DS18B20-szenzor pontosan méri a kaptár belső hőmérsékletét, amely közvetlen információt nyújt a kaptár állapotáról.

- **Kalibrálás:**

A hőmérőt egy hagyományos eszközzel összehasonlítva teszteltem, hogy megbizonyosodjak a mért értékek pontosságáról.

- Ha a belső hőmérséklet 30 °C alá esik, az arra utalhat, hogy a méhcsalád létszámiánnyal küzd, vagy legyengült, ami lehet betegség vagy

parazitafertőzés következménye. Ilyen esetben azonnali beavatkozás szükséges a méhész részéről.

5.2. Eredmények kiértékelése

Miután elvégeztem a kalibrálást és az ellenőrzéseket, az egyes szenzorok által mért adatokat és azok pontosságát többféleképpen értékeltem. Az alábbiakban összegzem a kiértékelés során szerzett tapasztalataimat:

1. **Hangérzékelő kiértékelése:** A hangérzékelő jól reagált a különböző zajintenzitásokra, és az értékek arányosan változtak a zaj szintjével. A szenzor érzékenysége alapján pontosan mérte a különböző méhzúgásokat, és a kalibrált értékek a rendszerben való alkalmazásra megfelelőnek bizonyultak.
2. **Füstjelző kiértékelése:** A füstjelző megfelelően reagált a gyertya füstjére, és az értékek megerősítették, hogy a szenzor jól érzékeli a füst koncentrációját. Mivel a szenzor nem használ mértékegységeket, az értékek viszonylagosak, de jól mutatták a füst koncentrációjának növekedését.
3. **Tehercella kiértékelése:** A tehercella kalibrálásával sikerült elérni, hogy a mérési értékek megfeleljenek a házi mérleg adatainak, és így pontos súlyméréseket végeztem. A kalibráció folyamán végzett finomhangolás biztosította, hogy a szenzor pontosan tükrözze a súlyok változásait. A vázba való telepítés után le kellett vonnom a váz tömegét, hogy csak a kaptár értéke látszódjon a kijelzőn.

6. Összegzés és fejlesztési lehetőségek

6.1. A rendszer eredményeinek értékelése

A projekt során sikerült egy működő mérőrendszert kialakítani, amely a különböző szenzorok adatainak mérésére és azok valós idejű továbbítására képes. Az adatok a Firebase Realtime Database-en keresztül tárolódnak, és az MIT App Inventor-alkalmazásban valós időben megjeleníthetők, lehetővé téve az adatok egyszerű monitorozását.

A szenzorok kalibrálása és ellenőrzése is sikeres volt. Mindezek a lépések biztosították a mérési pontosságot és a rendszer megbízhatóságát. Az adatok megjelenítése és a valós idejű monitorozás megvalósulása lehetővé teszi a rendszer folyamatos ellenőrzését és kiértékelését.

6.2. További fejlesztési irányok

Bár a rendszer működése alapvetően sikeres volt, számos területen van még lehetőség a fejlesztésre. Az alábbiakban bemutatom a lehetséges fejlesztési irányokat és a projekt során felmerült problémákat.

Fejlesztési irányok:

- **Kamera integrálása:** A rendszerhez eredetileg egy kamera integrálása volt tervben, hogy képeket küldjön a méhkijáratról. Az Arduino Nano tárolókapacitása azonban nem volt elegendő a képek kezelésére, így ezt nem sikerült megvalósítani. A Raspberry Pi használata ebben az esetben jobb megoldás lehetett volna, de az nem lett volna költséghatékony.
- **Több szenzor:** Egy giroszkópot lehetne integrálni a rendszerbe, amely akkor lenne különösen hasznos, ha a kaptár felborulna, például egy megvadult állat következtében. A giroszkóp segítségével érzékelhető lenne a kaptár dőlése, így időben észlelhetővé válna a helyzet, és megfelelő intézkedések hozhatók a stabilitás visszaállítása érdekében.
- **Biztosabb tápellátás:** Egy nagyobb áramforrás, például egy akkumulátor stabilabb megoldást kínálna, mint a ceruzaelemek, és hosszabb élettartamot biztosítana. Továbbá, a rendszer energiáját egy napkollektor segítségével is lehetne biztosítani, amelyet a kaptár tetejére lehetne felszerelni. Mivel a kaptár teteje alapvetően fémről készült, és nyáron akár 70 °C-ra is felmelegedhet a napsütés hatására, a napenergia hatékonyan kihasználható lenne a rendszer működtetésére, biztosítva ezzel a folyamatos energiaellátást.

- A rendszer alkalmazása különösen hasznos lehet Új-Zélandon, ahol a méhészek a világhírű manukaméztermelők. A manukaméz kiemelkedő immunerősítő tulajdonságai miatt rendkívül értékes, és gyakran alkalmazzák rákos betegek kezelésében. A manukaméz alapját adó manukacserjék az új-zélandi őserdőkben és hegyvidéki területeken nőnek, amelyek nehezen megközelíthetők. Ezekre a helyekre gyakran csak helikopterrel lehet eljutni, ami a méhészek számára jelentős kihívást jelent.
- Az intelligens kaptármérő rendszer ebben az alkalmazási környezetben nagyban megkönnyítené a méhészek munkáját. A rendszer valós idejű adatokat szolgáltat a kaptár állapotáról, beleértve a hordás intenzitását, a kaptáron belüli hőmérsékletet és páratartalmat, valamint a méhek aktivitását. Ez lehetővé teszi a méhészek számára, hogy pontosan nyomon kövessék, mikor zajlik intenzív nektárgyűjtés, és mikor szükséges beavatkozni a kaptárak kezelésébe. A nehezen megközelíthető helyszínek miatt különösen előnyös, hogy a rendszer távoli monitorozást biztosít, így csökkenti a helyszíni ellenőrzések gyakoriságát és költségét, miközben növeli a termelés hatékonyságát.

Zsákutcák és megghiúsult megvalósítások:

- **Kamera integrálása:** Az Arduino Nano nem tudott képeket kezelni, így ezt a funkciót nem sikerült megvalósítani. A Raspberry Pi-ra való váltás lehetőségét elvettem a magas költségek miatt.
- **A7670E-modul használata:** Az A7670E-modult a tervek szerint a képek küldésére és az adatok továbbítására szerettem volna használni, de a memória- és tárolókapacitás hiányosságai miatt nem tudtam megfelelően megvalósítani. Ez a modul csak adatcsomagot volt képes küldeni a Firebase-be. A JSON-formátumban küldött adatcsomagok nem voltak megfelelőek a valós idejű adatok megjelenítéséhez az MIT App Inventor-alkalmazásban.

Köszönetnyilvánítás

Ezúton szeretném kifejezni hálámat és köszönetemet mindazoknak, akik segítettek és támogattak engem a projekt során.

Külön köszönetet mondok édesapámnak, aki nemcsak a méhészet elméleti hátterének megértésében segített, hanem biztos alapot is adott a projekt sikerességéhez.

Köszönetemet szeretném kifejezni Dr. Csarnovics István Tanár Úrnak, aki önként vállalta el a konzultációt, mivel látta a lehetőséget az ötletemben. Hozzájárult ahhoz, hogy biztos legyek a szakdolgozatom témaválasztásában, és folyamatosan bátorított és támogatott a projekt megvalósítása során.

Hálás vagyok Kiss Rebeka tanárnőnek is, aki a NYÁK tervezésében és elkészítésében nyújtott fáradhatatlan segítséget. Kedvessége, szakértelme és hasznos tanácsai nélkül nem sikerült volna ilyen szintre hozni a tervezési folyamatot.

Irodalomjegyzék

- [1]. <https://docs.arduino.cc/hardware/nano/>
- [2]. <https://docs.arduino.cc/hardware/mega-2560/>
- [3]. https://hu.wikipedia.org/wiki/Raspberry_Pi
- [4]. <https://www.faranux.com/product/sim800l-v2-0-5v-wirelessgsm-gprs-module-quad-band/>
- [5]. SIM800 Series_Embedded AT_Application Note_V1.04.pdf datasheet
- [6]. <https://en.simcom.com/product/SIM800F.html>
- [7]. <https://hobbielektronikabolt.hu/spd/YXB051/DHT11-Homerseklet-es-pa-ratartalom-szenzor-modul-T>

- [8]. <https://nerdyelectronics.com/working-of-dht-sensor-dht11-and-dht22/>
- [9]. https://www.fruugo.hu/200-kg-elektronikus-merlegerzekelo-digitalis-merocella-sulyerzekelo-platform-merleg-merocella-tartos-bun/p-271827840-602752376?language=hu&ac=KelkooCSS&asc=pmax&gad_source=1&gclid=Cj0KCQjwvpy5BhDTARIsAHSilynV_Nbmtd9udD_i3
- [10]. <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all>
- [11]. <https://botland.store/pressure-sensors/12479-hx711-amplifier-for-tensometers-5904422319366.html>
- [12]. <https://electronics.stackexchange.com/questions/504553/how-does-the-hx711-circuit-work>
- [13]. <https://store.fut-electronics.com/products/smoke-sensor-module-mq2-digital-analog>
- [14]. https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/?utm_content=cmp-true
- [15]. <https://aaravpatel.com/2024/01/31/tutorial-23-how-do-you-use-a-ky-038-sound-sensor/>
- [16]. https://www.hestore.hu/prod_10035574.html
- [17]. <https://www.alldatasheet.com/datasheet-pdf/pdf/1138845/ETC2/KY-038.html>
- [18]. <https://components101.com/sensors/ds18b20-temperature-sensor>
- [19]. DS18B20-3468014.pdf datasheet
- [20]. <https://components101.com/sensors/ds18b20-temperature-sensor>
- [21]. <https://www.malnasuli.hu/alapok/16x2-lcd-kijelzo/>
- [22]. <https://www.electronicsforu.com/technology-trends/learn-electronics/16x2-lcd-pinout-diagram>
- [23]. lcd kijelzo lcd016n002bcfh.pdf datasheet
- [24]. <https://www.14core.com/wiring-i2c-module-on-16x2-lcd-with-sclsd/>
- [25]. LCD-I2C-IF_EN_10035516.pdf datasheet
- [26]. <https://makerselectronics.com/product/lm2596s-dc-dc-converter-3a-step-down-module>
- [27]. LM2596S-PSUM_EN_10034287.pdf datasheet
- [28]. <https://soldered.com/learn/hum-lm2596-buck-converter/?srsltid=AfmBOop0R12o12BVUJMBZ2YF8YB262ZOOQ90cyfvzdWb-bIfBxogOw15>
- [29]. <https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/>

[30]. GY-NEO6MV2_EN_10035556.pfd datasheet

[31]. <https://firebase.google.com/docs/guides>

[32]. <https://appinventor.mit.edu/>

[33]. <https://appinventor.mit.edu/explore/ai2/tutorials>

[34]. A NYÁK-terv a KiCad-szoftver segítségével készült