2024-2-HU01-KA210-VET-000271388 számú, Ipar 4.0 technológiák alkalmazása okosházak gyártásában és üzemeltetésében című pályázat

Tartalom

1. A projekt bemutatása

1.1. Bevezetés – A projekt és az oktatási környezet

A jelen kézikönyv egy nemzetközi együttműködésben megvalósított, Ipar 4.0 és IoT technológiákra épülő oktatási projekt szakmai és módszertani dokumentációja. A projekt középpontjában egy okosház / okosfarm modell áll, amelynek fejlesztése során a résztvevő tanulók a digitális tervezéstől és additív gyártástól kezdve az elektronikai integráción és szoftverfejlesztésen át egy komplex, webalapú épületfelügyeleti rendszer létrehozásáig jutnak el.

A projekt kiindulópontját egy kereskedelmi forgalomban elérhető oktatókészlet (Smart Farm Kit) jelentette, amely megfelelő alapot biztosított az érzékelők, beavatkozók és az IoT-alapú vezérlés megismeréséhez. A fejlesztési folyamat azonban nem állt meg a gyári megoldások alkalmazásánál: a résztvevők kritikai elemzést végeztek, feltárták a rendszer korlátait, majd ezekre reagálva egy saját, ipari mintát követő, bővíthető rendszerarchitektúrát alakítottak ki.

A kézikönyv célja nem csupán egy konkrét projektmegvalósítás dokumentálása, hanem egy olyan nyílt, adaptálható oktatási modell bemutatása, amely más intézmények, tanulócsoportok számára is követhető és továbbfejleszthető, akár gyári elemek használata nélkül is.

1.2. A kézikönyv célja és szerepe

A jelen kézikönyv célja, hogy átfogó szakmai és módszertani útmutatót nyújtson egy olyan oktatási projekthez, amely az Ipar 4.0 és az IoT technológiák gyakorlati alkalmazásán keresztül támogatja a tanulók digitális, műszaki és mérnöki kompetenciáinak fejlesztését. A dokumentum nem kizárólag kész megoldásokat közöl, hanem egy olyan tanulási és fejlesztési folyamatot mutat be, amelyben a problémamegoldás, az elemzés és az iteráció kiemelt szerepet kap.

A kézikönyv szemlélete szerint az oktatás nem passzív ismeretátadásra, hanem aktív alkotó tevékenységre épül. A projekt során a tanulók nem pusztán felhasználói, hanem tervezői, kivitelezői és fejlesztői egy komplex technikai rendszernek. A dokumentum ezt a megközelítést támogatja végig, a tervezéstől a gyártáson és programozáson át a rendszerintegrációig.

A kézikönyv elsődleges funkciója oktatási segédanyagként való felhasználás. Tartalma alkalmas arra, hogy:

- tanórai és tanórán kívüli projektmunka alapjául szolgáljon,
- szakköri, tehetséggondozó vagy projektalapú képzési programokhoz illeszkedjen,
- pedagógusok számára módszertani támogatást nyújtson komplex műszaki és informatikai projektek lebonyolításához.

A módszertani megközelítés középpontjában a projektszemléletű és felfedezéses tanulás áll. A tanulók fokozatosan, egymásra épülő lépésekben jutnak el az egyszerűbb részfeladatoktól a komplex rendszermegértésig. A kézikönyv ennek megfelelően nem egyetlen „helyes”

megoldást rögzít, hanem alternatívákat, döntési pontokat és továbbfejlesztési lehetőségeket is bemutat.

A dokumentum több célcsoport számára is releváns:

- Tanulók számára: a kézikönyv tanulóbarát logikával mutatja be a fejlesztési folyamatokat, lehetőséget adva az önálló kísérletezésre, hibázásra és újratervezésre. A projekt során a tanulók megtapasztalják a mérnöki gondolkodás alapjait: problémát definiálnak, megoldást terveznek, tesztelnek, értékelnek és finomítanak.
- Oktatók számára: a kézikönyv támogatja a tanórai szervezést, a differenciált feladatkiosztást és a heterogén csoportok kezelését. Segítséget nyújt abban, hogyan lehet eltérő előképzettségű tanulókat közös projektbe bevonni, és hogyan lehet a technikai tartalmakat pedagógiailag strukturált, mégis rugalmas módon feldolgozni.
- Intézmények számára: a projekt olyan adaptálható mintát mutat be, amely beilleszthető a helyi tantervbe, projekthetekbe vagy nemzetközi együttműködésekbe. A nyílt jelleg lehetővé teszi, hogy az intézmények saját eszközparkjukhoz, infrastruktúrájukhoz és képzési profiljukhoz igazítsák a megvalósítást.

1.3. A projekt rövid bemutatása és célkitűzései

Kapcsolódás a technikumi és középfokú informatikai képzésekhez: a projekt eredetileg technikumi környezetben indult, ahol az ipari informatika, mechatronika és kapcsolódó műszaki képzések tanulói már rendelkeznek alapvető technológiai ismeretekkel. A nemzetközi együttműködés során azonban egy elméleti líceum is bekapcsolódott a programba, amelynek tanulói nem rendelkeztek előzetes műszaki tapasztalatokkal. Ez a helyzet egyszerre jelentett kihívást és pedagógiai lehetőséget.

Előnyök:

- Az eltérő háttérrel rendelkező tanulók együttműködése erősítette a magyarázó, rendszerező gondolkodást.
- A technikumi tanulók mentor szerepbe kerültek, ami elmélyítette saját tudásukat.
- Az elméleti líceum tanulói számára a projekt hidat képezett az absztrakt informatikai ismeretek és a fizikai rendszerek között.

Lehetséges nehézségek:

- A műszaki alapismeretek hiánya lassíthatta a kezdeti haladást.
- Fokozott hangsúlyt kellett helyezni az alapfogalmak (elektronika, szenzorika, vezérlés) lépésről lépésre történő bevezetésére.
- Az oktatói szerep erősebben facilitáló, támogató jellegűvé vált.

A projekt tapasztalatai alapján megállapítható, hogy a bemutatott modell nem kizárólag technikumi környezetben működőképes. Megfelelő módszertani felépítéssel és differenciálással középfokú, nem műszaki profilú intézményekben is sikeresen alkalmazható.

1.4. Az „okosház / okosfarm” projekt alapgondolata

A projekt alapgondolata egy olyan fizikai és digitális modell létrehozása, amely kézzelfogható módon szemlélteti az okos rendszerek működését, miközben a tanulók számára értelmezhető és valós problémákon keresztül mutatja be a modern technológiák alkalmazását. Az okosház / okosfarm modell lehetőséget teremt arra, hogy a műszaki, informatikai és környezeti szempontok egységes rendszerként jelenjenek meg az oktatásban.

A modell alkalmas arra, hogy egyszerre mutassa be:

- az érzékelők és beavatkozók szerepét a környezeti állapotok figyelésében és szabályozásában,
- az adatgyűjtés, adatfeldolgozás és visszacsatolás folyamatát,
- a vezérlés és felügyelet logikáját helyi és távoli környezetben,
- a webalapú, távoli elérés lehetőségeit, mint az energia- és erőforrás-hatékony működés eszközét.

A projektben az okosház és az okosfarm nem csupán technológiai demonstrációként jelenik meg, hanem a környezettudatos szemlélet hordozójaként is. A szenzorokkal támogatott mérés és automatizált beavatkozás révén a tanulók megismerik, hogyan járulhat hozzá az intelligens vezérlés az energiafelhasználás optimalizálásához, a környezeti erőforrások tudatos használatához, valamint a fenntartható működéshez (pl. világítás, szellőztetés, öntözés szabályozása).

A modell nem végtermék, hanem tudatosan nyitott, folyamatosan fejleszthető tanulási eszköz, amely a tervezés, a kísérletezés, az értékelés és az újratervezés ciklusain keresztül támogatja a rendszerben való gondolkodás kialakulását. Ez a megközelítés elősegíti, hogy a tanulók ne csupán technikai megoldásokat sajátítsanak el, hanem megértsék azok környezeti, gazdasági és társadalmi összefüggéseit is.

1.5. Integrált Ipar 4.0 oktatási architektúra és nemzetközi fejlesztési környezet

A projekt az Ipar 4.0 szemlélet alapelveit követi:

- elosztott rendszerarchitektúra,
- adatvezérelt működés,
- hálózatba kapcsolt eszközök alkalmazása,
- szabványos kommunikációs protokollok használata.

Az IoT technológiák alkalmazása lehetővé teszi, hogy a tanulók ne elszigetelt eszközökkel dolgozzanak, hanem egy összefüggő rendszer részeként értelmezzék a hardver- és szoftverkomponenseket.

A Smart Farm Kit tudatosan csak kiindulópontként szolgált. Az egyik kulcseleme a gyári rendszer elemzése, korlátainak felismerése, majd ezekre reagálva egy saját, ipari mintát követő, bővíthető rendszerarchitektúra kialakítása volt. Ez a folyamat rávilágított arra, hogy a technológiai fejlődés alapja a kritikus gondolkodás és az iteratív fejlesztés.

A megközelítése teljes egészében projektszemléletű. A tanulók nem előre definiált lépéssorokat hajtanak végre, hanem problémák mentén haladnak, saját döntéseket hoznak, és azok következményeit elemzik. A tanulási folyamat során a hibák nem kudarcként, hanem a fejlődés természetes részeként jelennek meg.

A fizikai modellek, az azonnali visszajelzések és a látható eredmények természetes módon növelik a tanulói motivációt. A projekt során megjelennek a gamifikáció elemei, mint a kihívások, mérföldkövek és közös célok, amelyek strukturálják és fenntartják az érdeklődést.

Komplex módon fejleszti a digitális kompetenciákat: programozás, adatbáziskezelés, hálózati ismeretek, digitális tervezés és dokumentáció egyaránt megjelenik. Emellett erősíti a mérnöki gondolkodást, a rendszerben való gondolkodást, valamint a funkcionális egységek közötti kapcsolatok felismerését.

A projekt nemzetközi együttműködésben valósult meg, amely a szakmai tartalmak mellett jelentős kompetenciafejlesztési és pedagógiai hozzáadott értéket biztosított. A részt vevő intézmények eltérő országban, oktatási rendszerben és intézményi profilban működnek, így a közös munka valós, a későbbi munkaerőpiaci helyzeteket jól modellező együttműködési környezetet teremtett.

A közös tevékenységek során hangsúlyosan megjelentek a nyelvi különbségek. A szakmai kommunikáció több esetben angol nyelven zajlott, ami fejlesztette a tanulók idegen nyelvi kompetenciáit, és tudatosította a pontos, egyértelmű műszaki kommunikáció jelentőségét.

A multikulturális együttműködés a munkamenet természetes részévé vált. A különböző oktatási háttérrel és problémamegoldási stratégiákkal rendelkező tanulók közös munkája erősítette a rugalmasságot, az alkalmazkodóképességet és a szakmai párbeszédet.

A működése több szempontból is leképezte a multinacionális vállalatok elvárásait: idegen nyelvű kommunikáció, közös dokumentációkezelés, csapatmunka és felelősségvállalás komplex rendszerekben. A megszerzett tapasztalatok hozzájárultak ahhoz, hogy a tanulók műszaki ismereteiket szélesebb, nemzetközi kontextusban tudják értelmezni.

2. A projekt felépítése és megvalósítási logikája

2.1. A projekt szakaszai és egymásra épülése

A projekt megvalósítása tudatosan egymásra épülő szakaszok mentén történt. Ez a felépítés nem csupán technikai szükségszerűség volt, hanem pedagógiai döntés is: a cél az volt, hogy a tanulók fokozatosan, saját tapasztalataikra építve jussanak el az önálló fejlesztési feladatokig. A szakaszolás lehetővé tette az eltérő előképzettségű tanulók bevonását, valamint az egyes lépésekhez kapcsolódó reflexiót és értékelést.

Az első szakaszában a tanulók egy már működő, kész rendszerrel találkoztak. Ennek célja az alapfogalmak, működési összefüggések és rendszerlogika megismerése volt. A hangsúly nem a megoldás „lemásolásán”, hanem a megértésen volt: mit miért csinál a rendszer, milyen elemek alkotják, és hogyan kapcsolódnak egymáshoz.

Ez a szakasz különösen fontosnak bizonyult a kevésbé műszaki háttérrel rendelkező tanulók esetében, mivel közös kiindulópontot teremtett, és csökkentette a belépési küszöböt a későbbi, összetettebb feladatokhoz.

A megismerést a második, a tudatos elemzési szakasz követte. A diákok nemcsak használták a rendszert, hanem vizsgálták annak működési határait, erősségeit és hiányosságait. Ez a lépés kulcsszerepet játszott abban, hogy megértsék: egy technikai megoldás mindig kompromisszumok eredménye, és adott célokra optimalizált.

A kritikai értékelés során megtanulták megfogalmazni saját fejlesztési igényeiket, és felismerni, mikor indokolt egy meglévő rendszer továbbfejlesztése vagy újragondolása. Ez a gondolkodásmód közvetlenül kapcsolódik a mérnöki és informatikai problémamegoldás alapelveihez.

A projekt harmadik szakaszában a tanulók az elemzési tapasztalatokra építve saját fejlesztési munkába kezdtek. Itt már nem előre meghatározott lépések vezették a folyamatot, hanem a közösen meghozott döntések, az iteratív tervezés és a folyamatos tesztelés.

Ebben a szakaszban vált igazán láthatóvá a projektszemlélet előnye: a tanulók megtapasztalták, hogy egy komplex rendszer nem egyszerre „készül el”, hanem fokozatosan formálódik, és minden döntés hatással van a teljes működésre.

2.2. Résztvevők és szerepkörök

A projekt egyik meghatározó jellemzője a résztvevők tudatos szerepmegosztása volt. A tanulók, oktatók és intézmények eltérő, de egymást erősítő szerepkörökben működtek együtt, ami hozzájárult a projekt rugalmasságához és fenntarthatóságához.

A tanulók a részvétel során nem passzív befogadók, hanem aktív fejlesztők voltak. Együttműködtek a tervezésben, a problémák értelmezésében, a megoldások kipróbálásában és értékelésében is. A részfeladatok elosztása során megtapasztalták a felelősségvállalás jelentőségét, valamint azt, hogy munkájuk hogyan illeszkedik egy összetettebb rendszer egészébe. Ez a szerep erősítette az önállóságot, az együttműködést és a hosszabb távú gondolkodást.

Az oktatók szerepe elsősorban támogató és irányító jellegű volt. A hangsúly az útmutatáson, a kérdésfeltevésen és a tanulási folyamat strukturálásán volt, nem pedig a kész megoldások átadásán.

Ez a megközelítés lehetővé tette, hogy a diákok saját tapasztalataikon keresztül jussanak el a megoldásokhoz, miközben biztos szakmai és pedagógiai háttér állt rendelkezésükre.

A nemzetközi együttműködés vegyes összetételű tanulócsoportokat eredményezett, amelyekben eltérő oktatási háttérrel és tapasztalattal rendelkező diákok dolgoztak együtt. Ez a helyzet erősítette a kommunikációs készségeket, az alkalmazkodóképességet és a közös problémamegoldást.

A vegyes csapatmunka a projekt egészét végigkísérte, és valós együttműködési helyzeteket teremtett, amelyek túlmutattak a hagyományos iskolai kereteken.

2.3. Tanulási eredmények és fejlesztendő kompetenciák

A tervezés során tudatos cél volt, hogy a tanulási eredmények ne szűküljenek le egyetlen kompetenciaterületre. A fejlesztési folyamat egyszerre támogatta a technikai, kognitív és szociális készségek fejlődését.

A projekt hozzájárult a digitális eszközök tudatos használatához, a strukturált munkavégzéshez és a digitális környezetben történő alkotáshoz. A diákok megtapasztalták, hogyan válik egy összetett rendszer értelmezhetővé és kezelhetővé.

Olyan komplex problémákkal találkoztak, amelyek megoldása több lépést, tervezést és visszacsatolást igényelt. Ez erősítette a rendszerszintű gondolkodást, az ok-okozati összefüggések felismerését és a hosszabb távú tervezési képességet.

A projekt szerves része volt az elvégzett munka dokumentálása és bemutatása. A tanulók megtanulták, hogyan lehet egy fejlesztési folyamatot átláthatóan rögzíteni és értelmezhető formában bemutatni. A létrejövő portfóliók a tanulási eredmények hosszabb távú hasznosítását is lehetővé teszik.

3. A Smart Farm Kit részletes bemutatása és feldolgozása

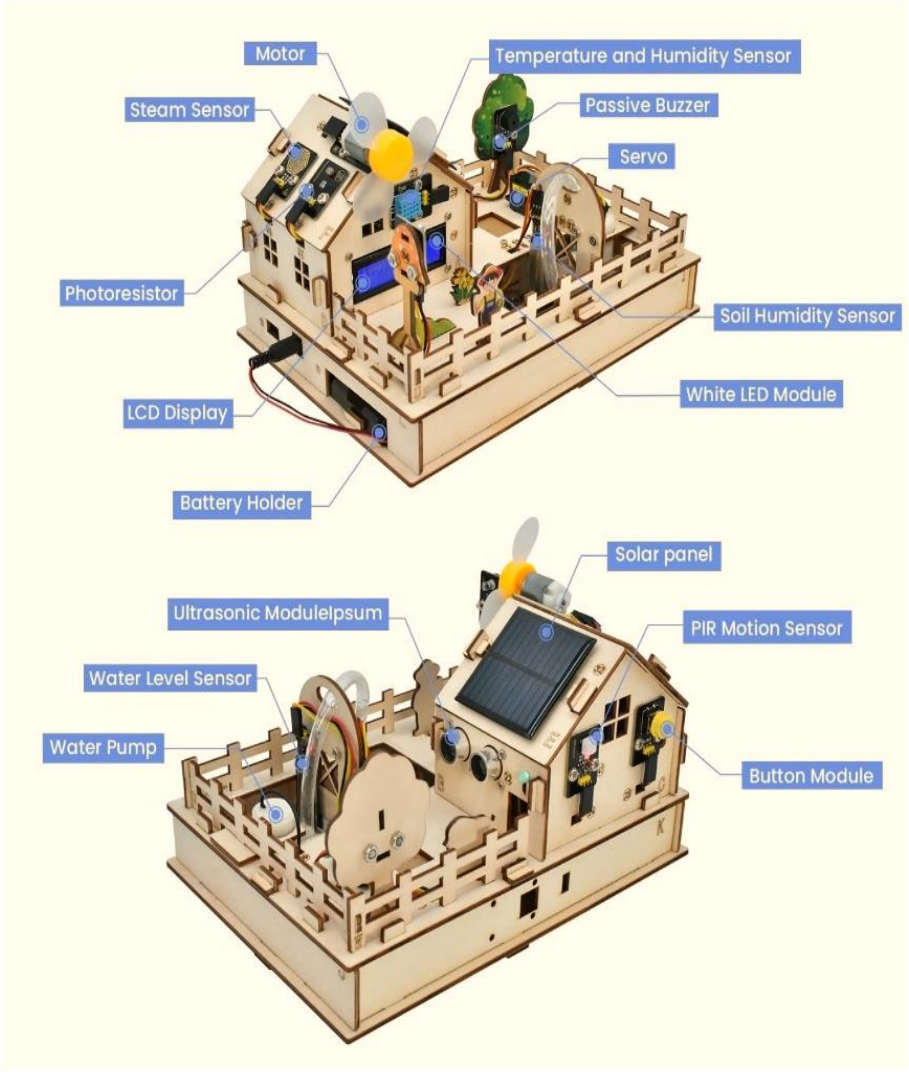
3.1. A Smart Farm Kit szerepe a tanulási folyamatban

A Smart Farm Kit egy oktatási célra tervezett, moduláris IoT-készlet, amely az okos rendszerek alapvető működési elveit egy kézzelfogható modell segítségével mutatja be. A készlet célja, hogy a tanulók egy valós, működő rendszerrel találkozzanak, mielőtt saját tervezésű megoldások fejlesztésébe kezdenének.

A projektben a készlet nem végtermékként, hanem tanulási eszközként jelenik meg. A feldolgozás során a hangsúly az elemek működésének megértésén, a rendszer logikájának feltérképezésén és a későbbi továbbfejlesztés megalapozásán van.

3.2. A Smart Farm Kit fő komponensei

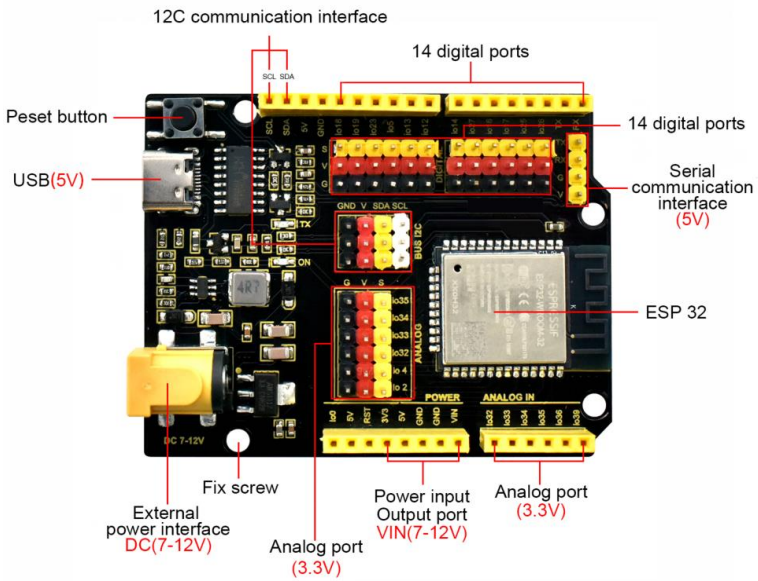
A Smart Farm Kit elektronikai elemei egy komplex IoT-rendszer bemutatására alkalmasak, amelyben többféle bemeneti szenzor, vezérlőmodul és kimeneti aktor szerepel. A lista a gyári forrás és megbízható termékleírások alapján készült, és lefedi a készlet tanulási programja során ténylegesen használt eszközöket.



1. ábra: A Smart Farm Kif fő részei (Forrás: https://www.keyestudio.com)

3.3. Vezérlő és alapmodulok

- ESP32 mikrovezérlő:
  - Felépítés: Wi-Fi és Bluetooth modullal integrált mikrokontroller
  - Funkció: A készlet „agya”, amely kezeli a szenzoradatokat, a logikát és a kommunikációt.
  - Oktatási jelentőség: Hálózati kommunikáció, párhuzamos eseménykezelés és I/O-logika tanulmányozása.



2. ábra: ESP32 mikrovezérlő (Forrás: https://docs.keyestudio.com)

- Relé modul:
  - Felépítés: Opto-izolált relé 5 V-os vezérléssel
  - Funkció: Nagyobb fogyasztású eszközök kapcsolása (pl. szivattyú).
  - Oktatási cél: Beavatkozók függetlenítése és biztonságos vezérlése.

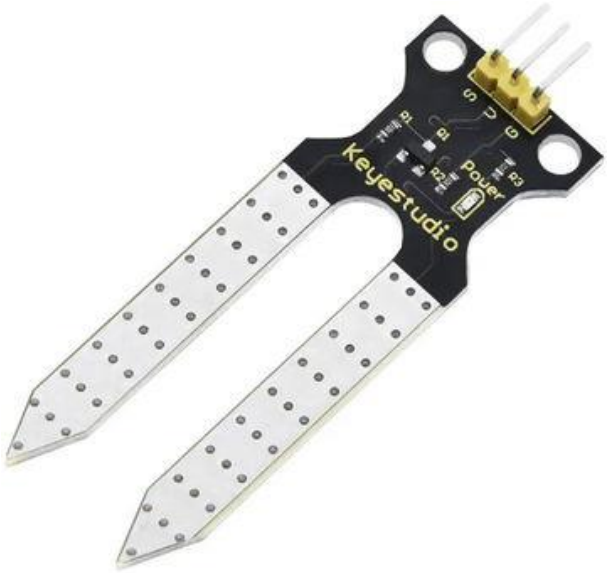
3. ábra: Relé modul (Forrás: https://docs.keyestudio.com)

3.4. Bemeneti elemek – Szenzorok

- Fotoellenállás (photoresistor):
  - Felépítés: fényérzékeny félvezető ellenállás
  - Működés: az ellenállás fényintenzitás függvényében változik (analóg jel).
  - Oktatási cél: fényviszonyok mérése, analóg bemenetek kezelése.

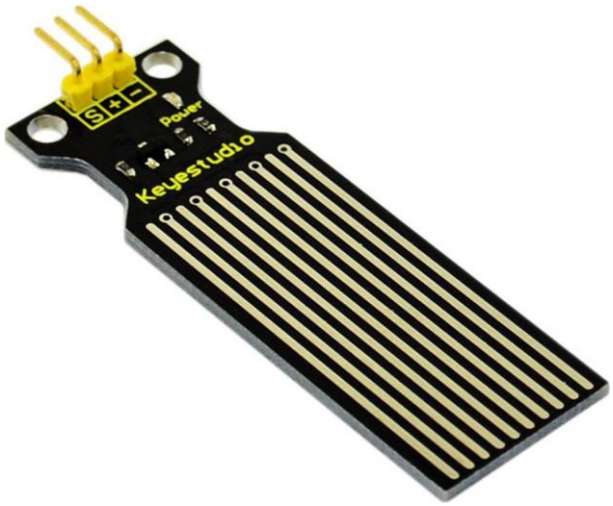
4. ábra: Fotoellenállás (Forrás: https://docs.keyestudio.com)

- Talajnedvesség-érzékelő:
  - Felépítés: két elektróda és jelátalakító
  - Működés: talajban lévő víz hatására csökken az ellenállás → magasabb jelérték.
  - Oktatási cél: analóg mérési technikák és környezeti adatok értelmezése.



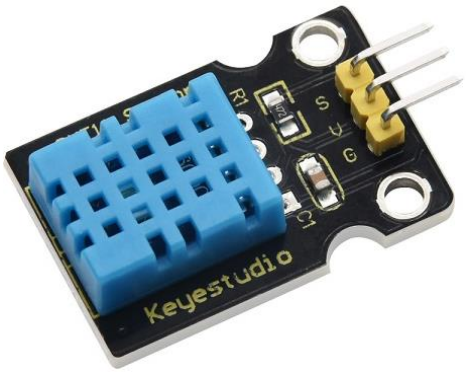
5. ábra: Talajnedvesség-érzékelő (Forrás: https://docs.keyestudio.com)

- Vízszint-érzékelő:
  - Felépítés: több elektródás érintkező
  - Működés: folyadékszint alapján záródó/nyitódó jel.
  - Oktatási cél: digitális eseményvezérelt mérés.



6. ábra: Vízszint érzékelő (Forrás: https://docs.keyestudio.com)

- DHT11 hőmérséklet- és páratartalom szenzor:
  - Felépítés: egységbe integrált digitális szenzor
  - Működés: hőmérséklet és páratartalom kiszámítása integrált algoritmussal.
  - Oktatási cél: digitális adatkommunikáció és fizikai környezet mérése.



7. ábra: DHT11 hőmérséklet- és páratartalom szenzor (Forrás: https://docs.keyestudio.com)

- SR01 V3 ultrahangos távolságérzékelő:
  - Felépítés: adó és vevő piezoelemek.
  - Működés: ultrahang impulzus → visszaverődés → időmérésből távolság számítása
  - Oktatási cél: fizikai hullámok és időalapú mérés megértése.

8. ábra: SR01 V3 ultrahangos távolságérzékelő (Forrás: https://docs.keyestudio.com)

- PIR mozgásérzékelő:

- Felépítés: passzív infravörös szenzor
- Működés: hirtelen hőmérséklet-változás érzékelése a látómezőben
- Oktatási cél: eseményvezérelt logika és valós idejű érzékelés.

9. ábra: PIR mozgásérzékelő (Forrás: https://docs.keyestudio.com)

- Passzív hangjelző (buzzer):
  - Felépítés: piezo hangszóró
  - Működés: feszültség impulzusokra válaszol hangkibocsátással
  - Oktatási cél: kimeneti modulok és gyakorlati jelzésrendszer kapcsolata.

3.5. Kimeneti elemek – Aktorok és kijelzők

- Fehér LED modul:
  - Felépítés: előre szerelt LED
  - Működés: digitális kimenet vezérli a világítást
  - Oktatási cél: vizuális visszajelzés egyszerű logikához.

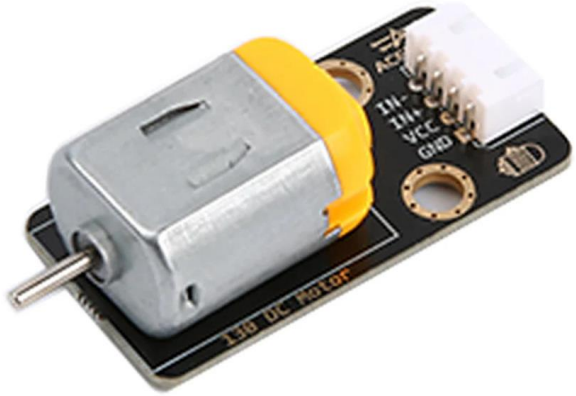
10. ábra: Fehér LED modul (Forrás: https://docs.keyestudio.com)

- SG90 9G szervomotor:
  - Felépítés: mikro szervo motor pozíció visszacsatolással
  - Működés: PWM jel alapján pozícióvezérlés

- Oktatási cél: beavatkozók precíz vezérlése és mechanikai hatások.

11. ábra: SG90 9G szervomotor (Forrás: https://docs.keyestudio.com)

- DC motor / ventilátor:
  - Felépítés: egyenáramú motor kis ventilátorral
  - Működés: kimeneti eszköz, amely a környezeti levegő mozgatását szolgálja
  - Oktatási cél: motorvezérlés és teljesítményfelügyelet.

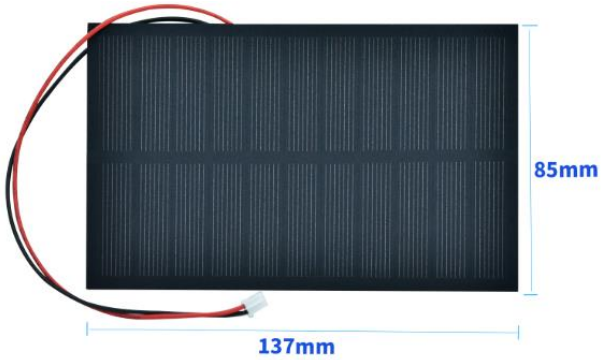


12. ábra: DC motor/ventilátor (Forrás: https://docs.keyestudio.com)

- I2C 1602 LCD kijelző:
  - Felépítés: 2 sor × 16 karakter, I2C kommunikáció
  - Működés: karakteres adat megjelenítés a mikrokontroller üzeneteiből
  - Oktatási cél: adatmegjelenítés és kommunikációs protokollok tanulása.

13. ábra: I2C 1602 LCD kijelző (Forrás: https://docs.keyestudio.com)

- Szolár panel:
  - Felépítés: kis méretű fotovoltaikus panel
  - Működés: fotovoltaikus hatás → villamos energia
  - Oktatási cél: megújuló energiaforrások IoT rendszerekben.



14. ábra: Szolár panel (Forrás: https://docs.keyestudio.com)

- Vízpumpa:
  - Felépítés: kis DC pumpa
  - Működés: folyadékszállítás relévezérléssel
  - Oktatási cél: aktorvezérlés és mechanikai munkavégzés megértése.

15. ábra: Vízpumpa (Forrás: https://docs.keyestudio.com)

3.6. Kiegészítők és szerelési eszközök

- átlátszó akril elemek és faforgács burkolat
- tápkábelek, DuPont vezetékek, csavarok, rögzítők
- elemhordozó vagy akkumulátor-tartó
- USB-kábel kommunikációhoz és tápellátáshoz

3.7. A komponensek szerepe az oktatásban

A készletben szereplő 17 elektronikai elem jól lefedi a bevezető IoT és automatizálási tananyagot:

- a szenzorok fizikai környezetből származó jelátvitelt biztosítanak,
- a vezérlőegység interpretál és dönt,
- az aktorok megjelenítik vagy beavatkoznak a környezetbe.

Ez a struktúra szisztematikus áttekintést ad arról, hogy egy valós IoT rendszerben miként működnek az elemek együtt, és hogyan kapcsolhatók össze programozással, mérési logikával és vezérléssel.

A Smart Farm Kit összeszerelése a projekt első olyan lépése, ahol a tanulók egy komplex, több komponensből álló fizikai rendszerrel dolgoznak. A gyári dokumentáció lépésről lépésre végigvezeti a felhasználót a mechanikai és elektronikai elemek összeépítésén, amely egyben tudatos tanulási folyamatként is értelmezhető.

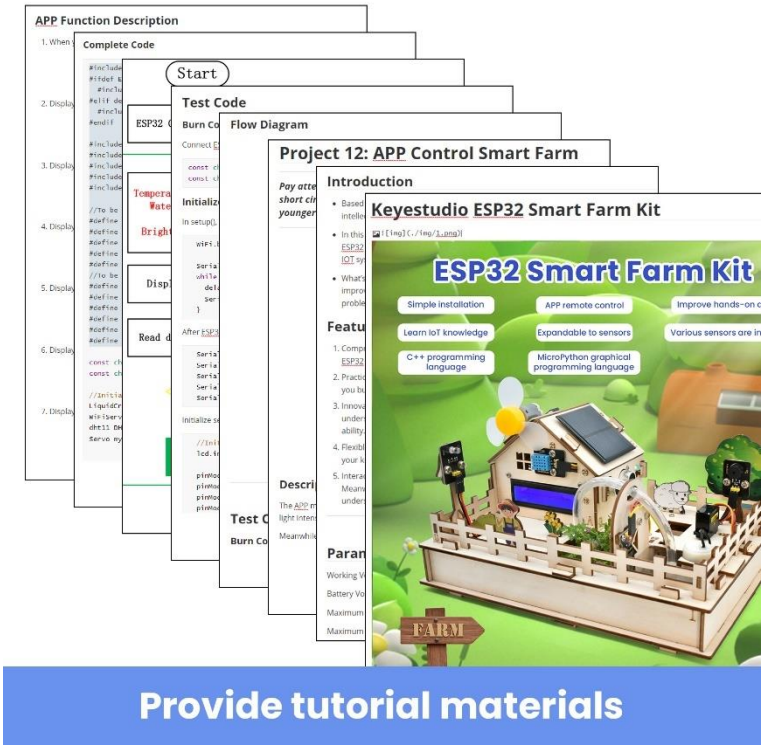
Az összeszerelés nem pusztán technikai feladat, hanem olyan tapasztalatszerzés, amely megalapozza a későbbi önálló tervezési és gyártási munkát.

3.8. A gyári összeszerelési útmutató szerepe

A Smart Farm Kit gyári útmutatója az összeszerelés során elsődleges referenciaként szolgál. Az útmutató:

- meghatározza az egyes szerelési lépések sorrendjét,
- bemutatja az alkatrészek helyes pozícióját és rögzítését,
- segíti az elektronikai elemek biztonságos bekötését,
- vizuális ábrákkal támogatja a megértést.

A tanulók az útmutató használata során megtapasztalják, hogy egy mérnöki rendszer összeállítása nem improvizáció, hanem strukturált, dokumentációra épülő folyamat.

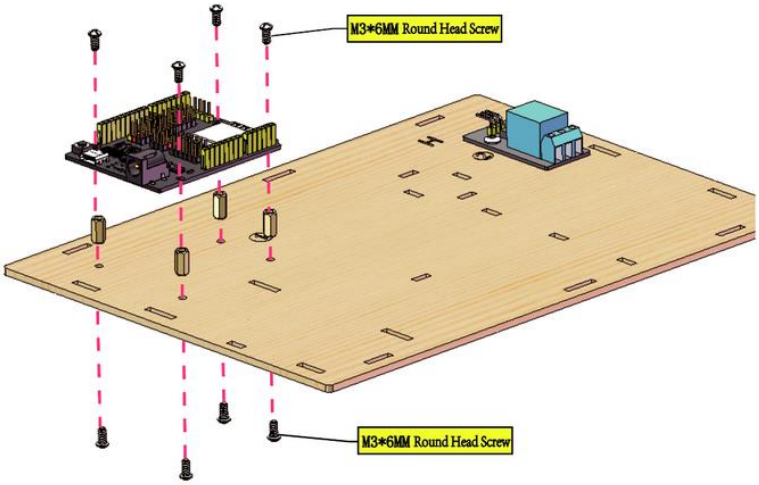


16. ábra: Gyári összeszerelési útmutató (Forrás: https://www.keyestudio.com)

3.9. Mechanikai és elektronikai szerelés és moduláris felépítés

A szerelés során a tanulók megismerkednek a készlet moduláris mechanikai kialakításával. Az egyes elemek előre kialakított rögzítési pontokkal rendelkeznek, melyek szabványos csatlakozási távolságokra épülnek, ezáltal lehetővé teszik az alkatrészek cseréjét és áthelyezését.

Ez a moduláris szemlélet fontos mérnöki alapelvet közvetít: a rendszer bővíthetősége és karbantarthatósága már a fizikai tervezés szintjén eldől.

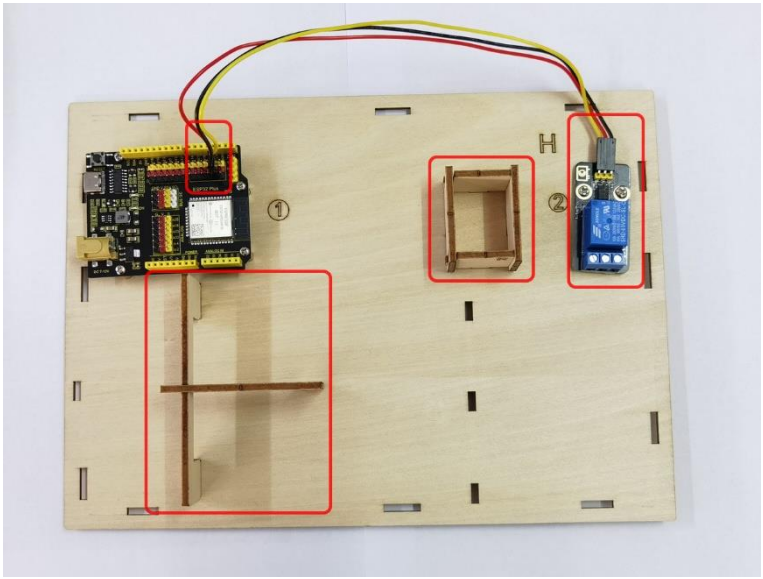


17. ábra: Okosház alaplemez (Forrás: https://docs.keyestudio.com)

Az elektronikai szerelés során a tanulók azonosítják a szenzorok és aktorok csatlakozási pontjait, megismerkednek a tápellátás és a jelvezetékek szerepével, valamint követik a gyári bekötési diagramokat.

A bekötések során kiemelt figyelmet kap a polaritás helyes kezelése, a digitális és analóg bemenetek megkülönböztetése, valamint a vezetékek mechanikai védelme és áttekinthetősége.

Ez a lépés alapozza meg a későbbi hibakeresési és rendszerdiagnosztikai gondolkodást.



18. ábra: Okosház alaplemez csatlakozási pontok (Forrás: https://docs.keyestudio.com)

Az összeszerelés nem egyszeri, lezárt folyamat, hanem iteratív tevékenység. A tanulók a szerelés egyes szakaszai után:

- ellenőrzik a mechanikai stabilitást,
- tesztelik az elektromos csatlakozásokat,
- rövid próbaprogramokkal validálják az egyes elemek működését.

Ez a megközelítés megelőzi a későbbi összetett hibákat, és ráirányítja a figyelmet a fokozatos ellenőrzés fontosságára.

3.10. Mérnöki szempontok az összeszerelés tapasztalatai alapján

Az összeszerelési folyamat során számos olyan mérnöki szempont válik megfigyelhetővé, amelyek közvetlenül hasznosíthatók a későbbi, saját tervezésű elemek létrehozásánál:

- Geometriai illeszkedés, az alkatrészek méreteinek és rögzítési pontjainak pontossága alapvetően befolyásolja a szerelhetőséget.
- Kábelezési útvonalak tervezése, a vezetékek elrendezése hatással van az átláthatóságra, karbantarthatóságra és esztétikára.
- Hozzáférhetőség és szervizelhetőség, a szenzorok és vezérlőelemek elhelyezése befolyásolja a későbbi módosítások és bővítések lehetőségét.

- Modularitás és cserélhetőség tekintetében a gyári megoldás rávilágít arra, hogyan lehet egy rendszert úgy felépíteni, hogy az egyes elemek önállóan is kezelhetők legyenek.

Ezek a tapasztalatok tudatosítják a diákokban, hogy a fizikai kialakítás és a funkcionális működés elválaszthatatlan egységet alkot.

Az összeszerelési szakasz lezárásaként a tanulók már nem csupán használói, hanem elemzői is a rendszernek. Az itt szerzett tapasztalatok közvetlenül előkészítik:

- a későbbi geometriai méréseket,
- a saját 3D tervezési feladatokat,
- az egyedi rögzítések és házelemek kialakítását,
- a gyári megoldások továbbfejlesztését vagy kiváltását.

A Smart Farm Kit összeszerelése így nem végállomás, hanem kiindulópont egy magasabb szintű, önálló mérnöki alkotófolyamathoz.

3.11. A Smart Farm Kit programozása

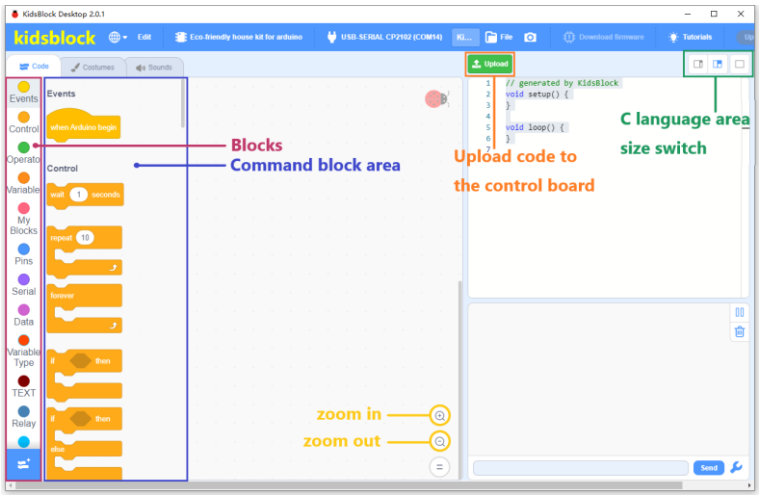
3.12. Blokk-alapú programozási környezet

A blokk-alapú programozás szerepe fontos a rendszermegértésben és az átmenetben más programozási modellek felé. A Smart Farm Kit grafikus programozási környezete (KidsBlock) Scratch-alapú logikát követ. A programozás vizuális blokkok egymáshoz illesztésével történik, amelyek mögött valós, futtatható vezérlési utasítások állnak. Ennek köszönhetően a tanulóknak nem kell a programnyelvek szintaktikai részleteivel foglalkozniuk, miközben a rendszer tényleges működését irányítják.

Oktatási szempontból ez a környezet több, egymást erősítő funkciót tölt be:

- Leválasztja a logikai gondolkodást a nyelvi nehézségekről, így a tanulók a vezérlési folyamatokra, az ok–okozati kapcsolatokra és az állapotváltozásokra tudnak koncentrálni.
- Azonnali visszacsatolást biztosít, mivel a program módosítása után a fizikai rendszer működése közvetlenül megfigyelhető.
- Támogatja az iteratív kísérletezést, a próbálkozás–hibázás–javítás természetes tanulási ciklusát.
- Előkészíti a későbbi szöveges programozást, mivel a blokkok mögött olyan alapstruktúrák jelennek meg, mint az eseménykezelés, feltételvizsgálat, ciklusok és állapotkezelés.

A blokk-alapú programozás ebben a projektben nem végállomás, hanem tudatosan megválasztott belépési szint egy több lépcsős programozási tanulási úton.



19. ábra: Blokk-alapú programozási felület, fő blokkkategóriák áttekintése (Forrás: https://docs.keyestudio.com)

3.13. Kapcsolódás szöveges programozási környezetekhez

A Smart Farm Kit mögöttes vezérlőegysége, az ESP32 mikrokontroller, nem kizárólag grafikus környezetben programozható. A projekt lehetőséget biztosít arra is, hogy ugyanazokat a funkciókat szöveges programozási nyelven, például Arduino-alapú környezetben valósítsák meg.

Az Arduino környezet:

- C/C++ alapú szintaxist használ,
- lehetővé teszi az alacsony szintű hardverkezelést,
- közvetlen rálátást ad a program futására, változóira és memóriahasználatára.

Pedagógiai értelemben ez a váltás:

- segít megérteni, hogy a blokk-alapú elemek mögött konkrét kód fut,
- fejleszti az absztrakciós képességet (blokk ↔ kódsor megfeleltetés),
- felkészít a professzionálisabb fejlesztői környezetek használatára.

A megvalósítás során a blokk-alapú és a szöveges programozás nem egymást kizáró, hanem egymást kiegészítő eszközként jelenik meg.

A Smart Farm projekt oktatási értékét tovább növeli, hogy a tanulók által elsajátított vezérlési logika szoros kapcsolatban áll az ipari automatizálásban alkalmazott programozási elvekkel.

A szakmai lehetőségeket tovább bővíti a nyílt forráskódú OpenPLC szoftverplatform, amely lehetővé teszi az IEC 61131-3 szabvány szerinti programozást nemcsak klasszikus PLC-ken, hanem Arduino- és ESP32-alapú vezérlőkön is. Az OpenPLC környezet teljes IEC 61131-3 nyelvi támogatást biztosít ezáltal lehetővé teszi az ipari vezérlési logika gyakorlását alacsony költségű hardveren, valamint támogatja az ipari kommunikációs protokollokat, különösen a Modbus és Modbus TCP szabványokat.

Ez a megközelítés hidat képez az oktatási célú mikrokontrolleres fejlesztések és a valós ipari automatizálási rendszerek között. A tanulók így olyan környezetben ismerkedhetnek meg az

ipari programozással, amely technológiailag egyszerűbb, de szemléletében teljes mértékben kompatibilis a professzionális rendszerekkel.

3.14. Kapcsolódás a projekt későbbi szakaszaihoz

A Smart Farm projekt további fázisaiban az ipari kommunikáció kiemelt szerepet kap. Ennek megfelelően:

- a Modbus / Modbus TCP protokoll használata a rendszerintegráció alapját képezi,
- a mikrokontrolleres vezérlésben továbbra is Arduino-alapú, C nyelvű programozás kerül alkalmazásra,
- a blokk-alapú és IEC 61131-3 szemlélet elsősorban gondolkodási és tervezési keretként szolgál.

Ez a döntés biztosítja, hogy a tanulók megismerjék az ipari szabványokat és elvárásokat, ugyanakkor egy rugalmas, oktatásbarát fejlesztési környezetben dolgozzanak, valamint felkészüljenek a későbbi, PLC-alapú vagy ipari IoT-rendszerekkel való munkára.

A Smart Farm projekt a blokk-alapú programozást nem önmagában, hanem egy tágabb ipari kontextusba ágyazva alkalmazza. Az IEC 61131-3 szabvány, az OpenPLC platform/projekt és a Modbus-alapú kommunikáció bemutatása lehetővé teszi, hogy a diákok már középfokon olyan rendszerszemléletet sajátítsanak el, amely közvetlenül illeszkedik a modern, multinacionális ipari környezet elvárásaihoz, miközben megmarad az oktatás biztonságos, fokozatos és motiváló jellege.

3.15. Projekt 1 – Világításvezérlés (Lighting System)

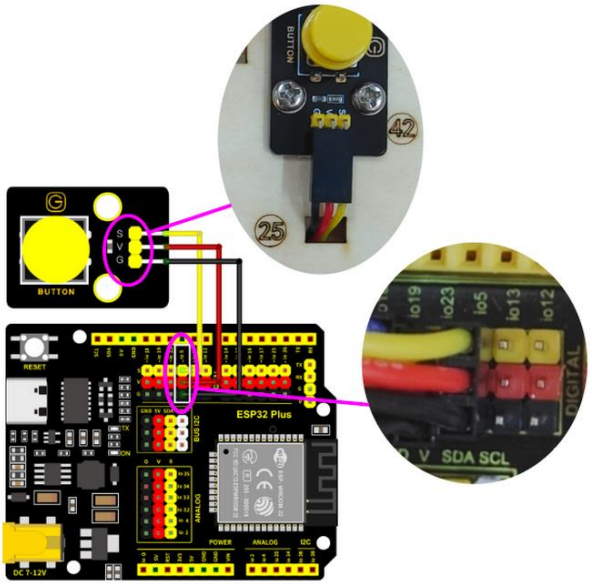
A világításvezérlés a Smart Farm Kit egyik legegyszerűbb, ugyanakkor oktatási szempontból kiemelten fontos mintaprojektje. Ebben a feladatban jelenik meg először teljes, zárt egységként a digitális bemenet → döntési logika → digitális kimenet vezérlési lánc. A projekt célja nem a világítás, mint funkció, hanem az automatizálás alaplogikájának megértése.

A nyomógomb, mint digitális bemenet

A nyomógomb a legegyszerűbb digitális bemeneti eszköz, amely két jól elkülöníthető állapotot képes közvetíteni a vezérlő felé:

- nem lenyomott állapot → logikai LOW (0),
- lenyomott állapot → logikai HIGH (1).

A vezérlő (ESP32) számára a nyomógomb nem „fizikai tárgy”, hanem egy digitális jel, amely egy adott bemeneti lábon jelenik meg. A bemenet állapota jellemzően belső vagy külső felhúzó- vagy lehúzóellenállással (pull-up/pull-down) stabilizált, így a vezérlő minden pillanatban egyértelmű HIGH vagy LOW értéket érzékel.



20. ábra: Nyomógomb bekötési rajza digitális bemenetként (ESP32 GPIO) (Forrás: https://docs.keyestudio.com)

A nyomógomb kiválóan alkalmas a digitális jel fogalmának bevezetésére, mivel nincs „köztes” érték, az állapot mindig egyértelműen értelmezhető, valamint az információ nem az érték nagyságából, hanem az állapotváltozásból származik.

Ez a megközelítés éles kontrasztot mutat az analóg szenzorokkal szemben, ahol a jel folytonos, és numerikus értelmezést igényel. A két jelleg közötti különbség megértése alapvető az automatizálási rendszerek tanulásában.

A nyomógomb csak egy a számos digitális bemenet közül. A projekt során – vagy annak továbbfejlesztéseként – az alábbi eszközök is hasonló elven működnek:

- Reed relé (mágneskapcsoló): mágneses tér hatására zár vagy nyit áramkört; tipikus alkalmazása ajtó- és ablakérzékelés.
- Végálláskapcsoló: mechanikai mozgás határhelyzetének érzékelésére szolgál.
- PIR mozgásérzékelő digitális kimenete: mozgás észlelése esetén HIGH jelet ad, egyébként LOW állapotban marad.
- Digitális reléérintkező visszajelzés: ipari rendszerekben gyakori visszacsatolási forma.

Ezek az eszközök közös tulajdonsága, hogy állapotinformációt szolgáltatnak, nem pedig mért mennyiséget.

Oktatási jelentősége, hogy a digitális bemenetek alkalmazása – különösen a nyomógombé – több alapvető gondolkodási mintát fejleszt:

- az eseményvezérelt gondolkodást (valami történik → reakció következik),
- az ember–gép interfész alapfogalmát,
- az absztrakciót, amely során a fizikai cselekvés digitális információvá alakul,
- a későbbi állapotgép- és vezérlési logikák megértését.

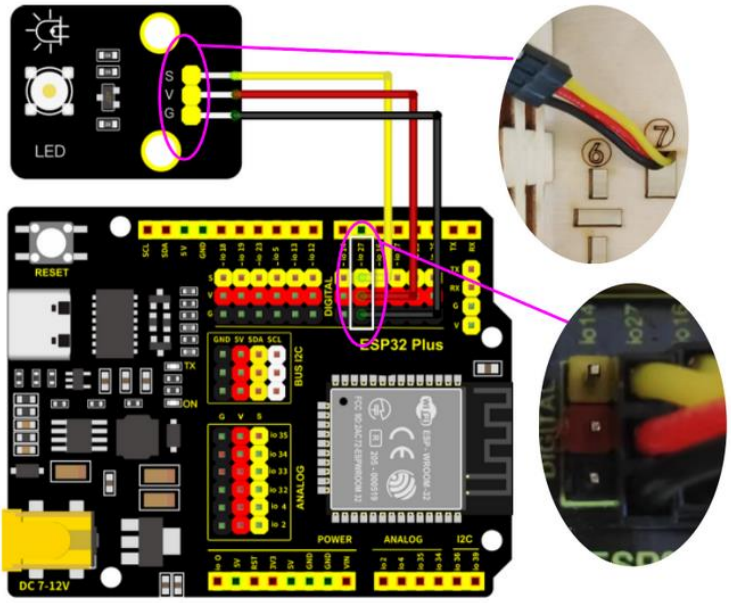
A tanulók ezen a ponton értik meg először, hogy egy automatizált rendszer nem „folyamatosan figyel”, hanem eseményekre reagál, ami az ipari vezérlések egyik alapelve.

Digitális kimenet – LED mint vezérelt beavatkozó

A LED (Light Emitting Diode) a Smart Farm Kit egyik legegyszerűbb digitális kimeneti eszköze, amely vizuális módon jeleníti meg a vezérlő döntéseit. Működése kétállapotú:

- kimenet LOW → LED kikapcsolt állapot,
- kimenet HIGH → LED bekapcsolt állapot.

A vezérlő (ESP32) digitális kimeneti lába a LED-et jellemzően áramkorlátozó ellenálláson keresztül vezérli, amely megvédi az alkatrészt a túláramtól. A tanulók itt találkoznak először azzal a gyakorlati ténnyel, hogy a vezérlő nem képes közvetlenül bármilyen terhelést meghajtani, hanem az elektronikai környezet figyelembevételével kell tervezni.



21. ábra: LED bekötési diagram digitális kimenetre (ESP32 GPIO + áramkorlátozó ellenállás) (Forrás: https://docs.keyestudio.com)

A digitális kimenet fogalma alapvetően eltér a bemenetektől. Míg a bemenetek információt szolgáltatnak a rendszerről, addig a kimenetek aktív beavatkozást jelentenek, mivel a rendszer nemcsak érzékel, hanem hatással van a környezetére.

A LED ebben a szakaszban nem világítási célokat szolgál, hanem állapotjelzőként működik. A tanulók számára világossá válik, hogy a LED állapota a program belső logikájának leképezése, mivel a kimenet mindig egy döntési folyamat eredménye. A fizikai visszajelzés kulcsszerepet játszik a megértésben és hibakeresésben.

A LED vezérlése során a tanulók először tapasztalják meg a teljes vezérlési láncot:

1. bemeneti esemény (pl. gomb megnyomása),
2. feldolgozás / döntés (feltételvizsgálat, állapotellenőrzés),
3. kimeneti beavatkozás (LED be- vagy kikapcsolása).

Ez a lánc a későbbi, összetettebb rendszerek (motorvezérlés, relézés, automatizmusok) absztrakt modelljeként szolgál.

A LED mint digitális kimenet oktatási szempontból előkészíti a következő eszközök megértését:

- relémodulok (nagyobb teljesítményű eszközök kapcsolása),
- buzzer / hangjelzők,
- jelzőlámpák (piros–sárga–zöld),
- digitális kijelzők vezérlési vonalai.

A diákok így felismerik, hogy a LED csupán egy egyszerűsített modell, amely mögött ugyanaz a vezérlési elv húzódik meg, mint az ipari beavatkozók esetében. Alkalmazása fejleszti:

- az ok–okozati gondolkodást,
- a rendszerszemléletet (bemenet → feldolgozás → kimenet),
- a vizuális visszacsatolás értelmezését,
- az absztrakciós képességet, amely később lehetővé teszi a fizikai eszközök elhagyását és szimulált rendszerek megértését.

A tanulók ezen a ponton értik meg, hogy egy automatizált rendszer „nem gondolkodik”, hanem előre definiált szabályok szerint reagál, és hogy a kimenet minden esetben a program logikai állapotának következménye.

Világításvezérlés – bemenet és kimenet összekapcsolása

A projekt következő lépése a nyomógomb és a LED összekapcsolása egy egyszerű, de teljes értékű vezérlési logikával. Ez a feladat elsőként jeleníti meg a bemenet – feldolgozás – kimenet gondolkodási modellt egységes rendszerként.

A világításvezérlés logikai modellje az alábbi elemekből épül fel:

- esemény: a nyomógomb állapotváltozása,
- döntés: a LED aktuális állapotának vizsgálata,
- beavatkozás: a LED be- vagy kikapcsolása.

Ez a struktúra már egy teljes automatizálási ciklust ír le, még akkor is, ha a megvalósítás technikailag egyszerű.

A tanulók számára tudatosítani kell, hogy a feladat önmagában nem igényel mikrovezérlőt. Egy nyomógomb és egy LED egyszerű huzalozott áramkörként is összekapcsolható, ahol a gomb közvetlenül zárja az áramkört és a LED azonnal reagál a fizikai kapcsolásra. Ez azonban nem automatizálás, hanem közvetlen, hardveres működés.

A mikrovezérlő bevonásának célja nem a feladat „megoldása”, hanem a teljes hatáslánc modellezése, vagyis fizikai esemény digitalizálása, logikai döntéshozatal szoftverben, valamint programozott beavatkozás végrehajtása. Ez a megkülönböztetés kulcsfontosságú a tanulók rendszerértésének kialakításában.

Programozási elemek (blokk-alapú környezetben)

A blokk-alapú programozás során az alábbi elemek jelennek meg:

- eseményindító blokkok (gomb lenyomása vagy állapotváltozása),
- digitális kimenetet vezérlő blokkok (LED be-/kikapcsolása),
- opcionális állapotváltozó, amely a LED aktuális állapotát tárolja.

A változó bevezetése nem technikai kényszer, hanem tudatos tervezési döntés, amely elkülöníti az eseményt (mi történt) és az állapotot (mi a rendszer aktuális helyzete).

22. ábra: Világításvezérlő blokk-program teljes szerkezete, változóhasználattal (Forrás: https://docs.keyestudio.com)

A feladat során a tanulók hamar felismerik, hogy a vezérlő nem „emlékszik” automatikusan a korábbi állapotokra, hanem az állapot megőrzése programozási felelősség, vagyis az esemény és az állapot külön kezelendő fogalmak.

Ez az élmény közvetlen előkészítése a többállapotú automatizmusoknak, az időzített vezérléseknek, valamint a hálózatba kapcsolt, aszinkron rendszereknek.

Oktatási reflexió

A világításvezérlési projekt egyszerűsége ellenére kiemelkedő oktatási értékkel bír. A tanulók megtapasztalják, hogy a digitális rendszerek nem gondolkodnak, hanem végrehajtanak, vagyis a rendszer működése teljes mértékben a tervezett logikától függ, így egy látszólag triviális feladat mögött is tudatos mérnöki döntések állnak.

A huzalozott megoldás és a vezérlőalapú megvalósítás összehasonlítása segít abban, hogy a diákok megértsék: az automatizálás nem az eszközökről, hanem a gondolkodásmódról szól.

Ez a projektrész stabil alapot teremt a későbbi szenzoros, feltételalapú és hálózatba kapcsolt vezérlési rendszerek megértéséhez, és természetes átmenetet biztosít az összetettebb IoT-megoldások felé.

3.16. Projekt 2 – Fényvezérelt rendszer (Light Control System)

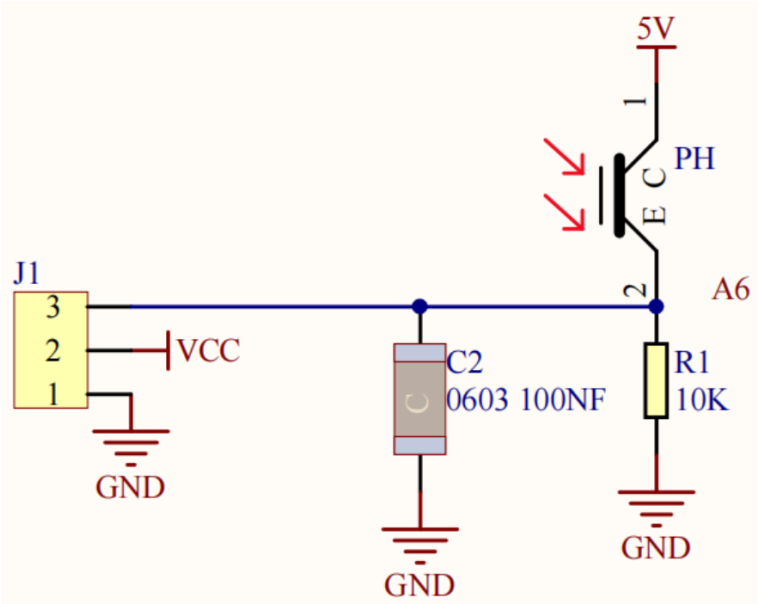
Ebben a projektben jelenik meg először az analóg mennyiségek mérése és feldolgozása, amely alapvető különbséget jelent a korábbi, kizárólag digitális logikára épülő feladatokhoz képest. A tanulók itt már nem kétállapotú jelekkel dolgoznak, hanem folyamatosan változó fizikai mennyiségeket értelmeznek és alakítanak át vezérlési döntésekké.

Fényérzékelő (LDR) működése – analóg bemenet

Az LDR (Light Dependent Resistor) olyan félvezető alapú ellenállás, amelynek ellenállásértéke a ráeső fény intenzitásának függvényében változik:

- erős megvilágítás → kisebb ellenállás,
- gyenge megvilágítás → nagyobb ellenállás.

Ez a változás folytonos, tehát analóg jellegű. Az ESP32 a fényerőt nem közvetlenül „látja”, hanem az LDR-rel kialakított feszültségosztó kapcsoláson megjelenő feszültséget méri az analóg bemenetén.



23. ábra: LDR karakterisztika és feszültségosztó bekötési diagram (ESP32 analóg bemenet) (Forrás: https://docs.keyestudio.com)

Az analóg bemenet:

- nem HIGH vagy LOW értéket ad,
- hanem egy számszerű tartományba eső mérési eredményt (pl. 0–4095),
- amely a fény intenzitásával arányos.

A projektben résztvevő tanulók itt szembesülnek azzal, hogy a mérési érték nem abszolút, hanem környezetfüggő, a szenzor és az elektronika pontossága korlátozott, valamint a mért adat értelmezése mindig kontextust igényel.

Döntési folyamat és beavatkozás – analóg feldolgozás

A fényvezérelt rendszer legegyszerűbb logikai felépítése:

1. fényintenzitás mérése (analóg bemenet),
2. összehasonlítás egy határértékkel,
3. beavatkozás végrehajtása (világítás ki- vagy bekapcsolása).

Ez a modell még digitális döntést eredményez egy analóg bemenet alapján.

24. ábra: Feltételes vezérlés blokk-szerkezete analóg bemenethez (Forrás: https://docs.keyestudio.com)

Analóg és digitális átmenet oktatási jelentősége

Ebben a lépésben hangsúlyosan jelenik meg az analóg jel digitalizálása, a határérték megválasztásának problémája, valamint a zajos, ingadozó mérési adatok kezelése.

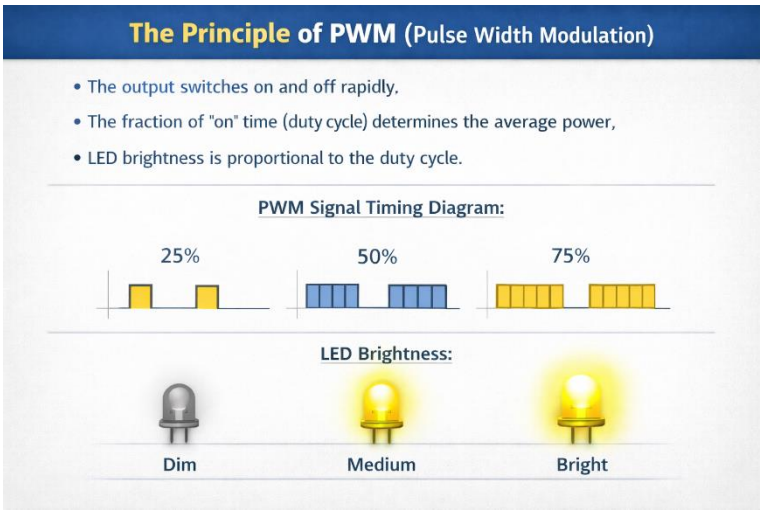
A tanulók megtapasztalják, hogy a túl alacsony vagy túl magas határérték instabil működést okozhat, ezért szükség lehet hiszterézisre vagy szűrésre, így megállapítható, hogy a mérnöki döntések hatással vannak a rendszer viselkedésére.

Analóg kimenet – PWM-alapú fényerőszabályzás

A projekt következő szintjén a beavatkozás már nem bináris, hanem folyamatos jellegű. A LED fényerejét nem csak be- és kikapcsoljuk, hanem arányosan szabályozzuk.

PWM (Pulse Width Modulation) egy olyan technika, amely lehetővé teszi, hogy digitális kimenettel analóg hatást érjünk el:

- a kimenet gyorsan ki- és bekapcsol,
- a bekapcsolt idő aránya (kitöltési tényező) határozza meg az átlagos teljesítményt,
- a LED fényereje a kitöltési tényezővel arányos.



25. ábra: PWM jel idődiagramja és LED fényerő-változás (Forrás: AI generált)

Ebben a projektben a tanulók egy arányos kapcsolatot hoznak létre:

- bemenet: mért fényintenzitás (analóg érték),
- feldolgozás: skálázás, átalakítás,
- kimenet: PWM jellel vezérelt LED fényerő.

Ez a megközelítés már folyamatvezérlési szemléletet tükröz, és közvetlen előképe az ipari szabályozásoknak.

Oktatási reflexió

A fényvezérelt rendszer projekt kulcsszerepet játszik a tanulók gondolkodásának fejlesztésében:

- különbséget tesznek digitális és analóg mennyiségek között,
- megértik, hogy a valós világ folyamatos, a vezérlők pedig diszkrétek,
- felismerik az arányos szabályozás alapelvét,
- megtapasztalják, hogy a „szebb” működés több tervezést igényel.

Ez a rész természetes hidat képez a későbbi hőmérséklet-szabályozási, motorvezérlési, energiafelhasználást optimalizáló rendszerek felé, és jelentős lépés a mérnöki gondolkodás elmélyítésében.

3.17. Projekt 3 – Távolságérzékelés és automatikus beavatkozás (Smart Feeding System)

A Smart Feeding System olyan integrált projekt, amelyben a tanulók egy időalapú mérésen alapuló szenzort és egy mechanikus beavatkozót kapcsolnak össze működő automatizmussá. Egy egyszerű, mégis valós élethelyzetet modellez: egy tárgy közelségének érzékelése hatására egy mechanikus művelet végrehajtása történik.

Ez teljes egészében lefedi az automatizált rendszerek alapvető működési láncát:

A Smart Feeding System funkcionális célja, hogy a rendszer egy automatikus adagoló működését valósítja meg, amely érzékeli egy tárgy (pl. kéz) közeledését, majd a mért távolság alapján döntést hoz és szervómotor segítségével mechanikus mozgást hajt végre (pl. fedél nyitása/zárása).

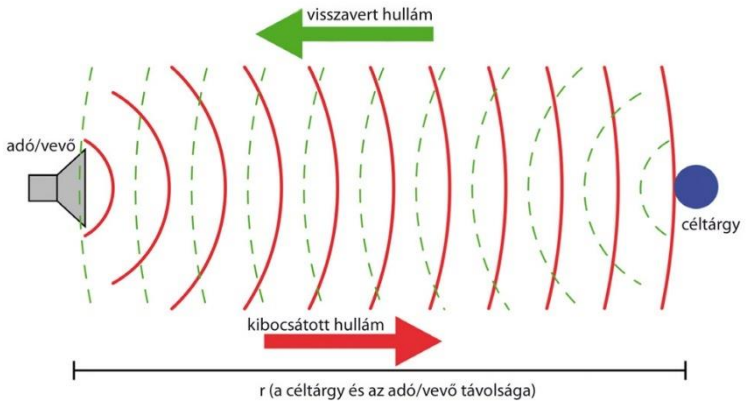
A funkció egyszerűen értelmezhető, ugyanakkor összetett technikai és mérnöki gondolkodást igényel.

Ultrahangos távolságérzékelő

Feladata a tárgytávolság meghatározása időmérés segítségével. A szenzor működése digitális vezérlésen alapul (trigger/echo), a távolságérték számítással áll elő.

Oktatási jelentőség:

- új mérési elv megismerése,
- a nyers mérési adat és az értelmezett információ elkülönítése.



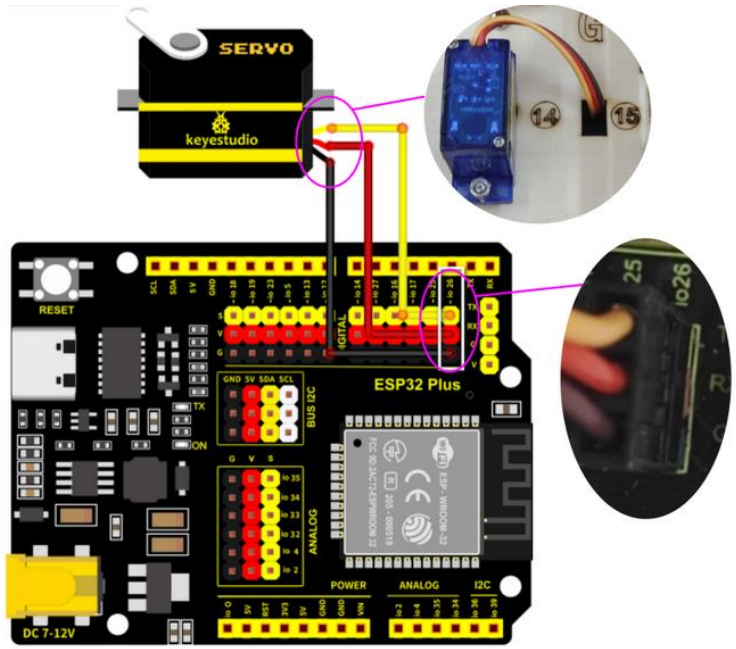
26. ábra: Ultrahangos távolságmérés elvi ábrája (Forrás: https://docs.keyestudio.com)

Szervómotor, mint mechanikus beavatkozó

A szervómotor a vezérlési döntés fizikai megvalósításáért felel. PWM jellel vezérelhető, meghatározott szöghelyzetekbe állítható, pontos és ismételhető mozgást biztosít.

Oktatási jelentőség:

- elektronika és mechanika összekapcsolása,
- a vezérlés közvetlen fizikai hatásának megtapasztalása.



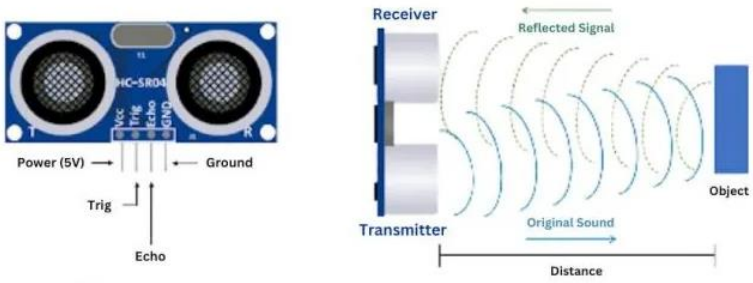
27. ábra: Szervómotor felépítése és vezérlési elve (Forrás: https://docs.keyestudio.com)

Ultrahangos távolságmérés alapelve

Az ultrahangos mérés folyamata:

1. • a szenzor hangimpulzust bocsát ki
2. • a hang visszaverődik a tárgyról
3. • a vezérlő méri az eltelt időt
4. • a távolság számítással meghatározható

Ez nem analóg feszültségmérés, hanem digitális jeleken alapuló időalapú mérés, amely számítási lépést igényel.



28. ábra: Ultrahangos mérés folyamata (Forrás: https://docs.keyestudio.com)

Szenzoradat feldolgozása és döntési logika folyamatának segítéséhez a mért távolság a vezérlési döntés alapját képezi. A logikai folyamat:

1. • a távolság mérése
2. • összehasonlítás határértékkel
3. • döntés meghozatala
4. • beavatkozás indítása

Ez a struktúra már egy teljes automatizált folyamatot ír le.

29. ábra: Távolságadat feldolgozása blokk-alapon (Forrás: https://docs.keyestudio.com)

Szervómotor vezérlése

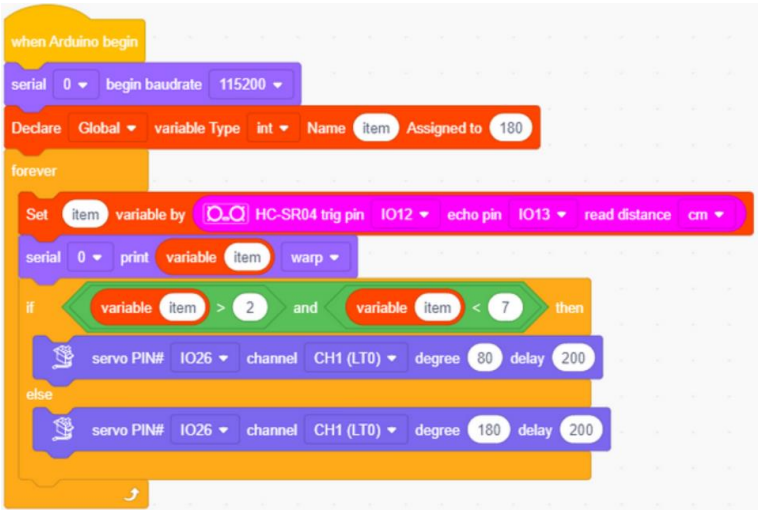
A szervómotor vezérlése PWM jellel történik, ahol az impulzusszélesség határozza meg a szöghelyzetet és a vezérlés nem teljesítményt, hanem pozíciót szabályoz.

Ez lényeges különbség a LED-ek PWM-es fényerőszabályozásához képest.

30. ábra: PWM jel és szervópozíció kapcsolata (Forrás: https://docs.keyestudio.com)

Smart Feeding System – integrált működés

A Smart Feeding System rendszer működése egységes logikai egészet alkot, ha a mért távolság egy küszöbértéken belül van, a rendszer döntést hoz, a szervómotor elmozdul és mechanikus művelet történik.



31. ábra: Smart Feeding System teljes blokk-program (Forrás: https://docs.keyestudio.com)

Oktatási reflexió

A Smart Feeding System projekt különösen alkalmas arra, hogy a tanulók:

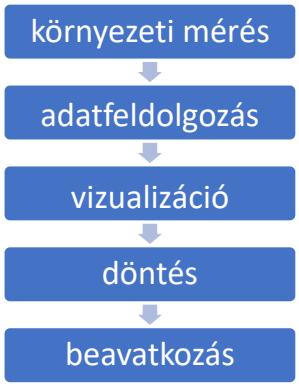
- rendszerszinten lássák a vezérlés működését,
- megértsék az időalapú mérés sajátosságait,
- felismerjék a döntési logika és a mechanikus hatás kapcsolatát,
- tudatos mérnöki gondolkodást alakítsanak ki.

A projekt világosan megmutatja, hogy az automatizált rendszerek működése nem az egyes elemekben, hanem azok összekapcsolásában és a tervezett logikában rejlik.

3.18. Projekt 4 – Hőmérséklet-szabályozás (Temperature Control System)

A Temperature Control System olyan klasszikus szabályozási feladatot modellez, amely az ipari automatizálásban és az épületfelügyeleti rendszerekben is alapvető jelentőségű. A projekt középpontjában egy környezeti mennyiség folyamatos mérése, annak megjelenítése, majd egy beavatkozó eszköz automatikus vezérlése áll.

A projekt teljes működési lánca:



Ez a felépítés a tanulók számára jól értelmezhető, ugyanakkor valós ipari szabályozási logikát tükröz.

Hőmérséklet- és páratartalom-érzékelő (DHT szenzor)

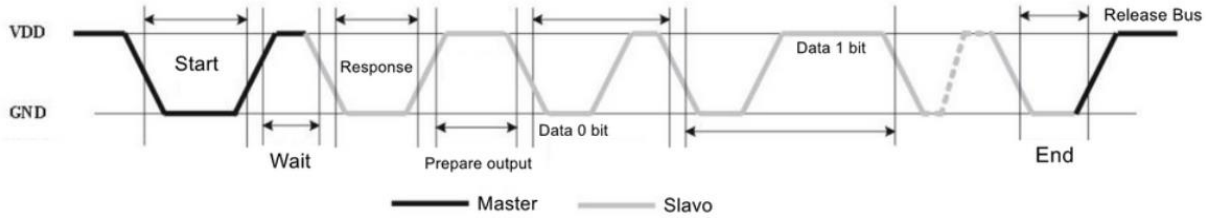
A DHT szenzor digitális környezeti érzékelő, amely hőmérsékletet, relatív páratartalmat mér, és az adatokat digitális adatcsomag formájában továbbítja a vezérlő felé.

Elektronikai sajátosságok:

- egyetlen adatvezetéken kommunikál,
- a mérési eredmény nem analóg feszültség, hanem feldolgozott digitális érték,
- a szenzor belső időzítést és ellenőrző mechanizmust használ.

Oktatási jelentőség:

- elkülöníti az analóg mérést a digitális adatkommunikációtól,
- rámutat arra, hogy a szenzorok egy része már „feldolgozott” adatot szolgáltat,
- bevezeti a mintavételezés és frissítési idő fogalmát.



32. ábra: DHT szenzor működési elve és adatátviteli folyamata (Forrás: https://docs.keyestudio.com)

LCD 1602 kijelző – mérési adatok megjelenítése

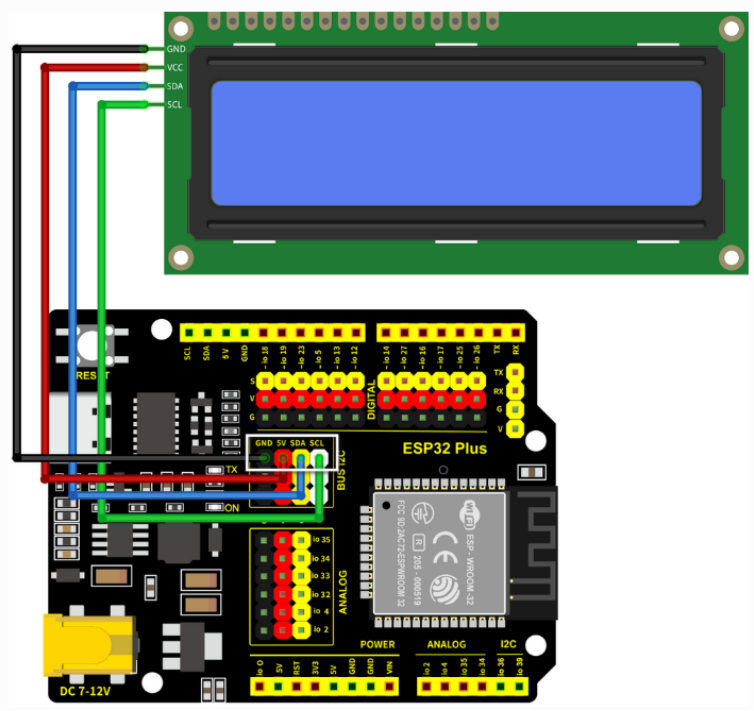
A hőmérséklet-szabályozó rendszer fontos eleme a felhasználói visszajelzés. Ezt a szerepet az LCD 1602 kijelző tölti be, amely numerikus formában jeleníti meg a mért adatokat és lehetővé teszi a rendszer aktuális állapotának folyamatos követését.

Funkcionális szerep:

- hőmérséklet és páratartalom kijelzése,
- üzemállapot visszajelzése (pl. ventilátor aktív/inaktív),
- diagnosztikai információk megjelenítése.

Oktatási jelentőség:

- hangsúlyozza a mérés és megjelenítés szétválasztását,
- bemutatja, hogy a rendszer működése nem „láthatatlan”,
- támogatja a hibakeresést és a rendszermegértést.



33. ábra: LCD 1602 bekötési rajz (Forrás: https://docs.keyestudio.com)

Ventilátor, mint beavatkozó elem

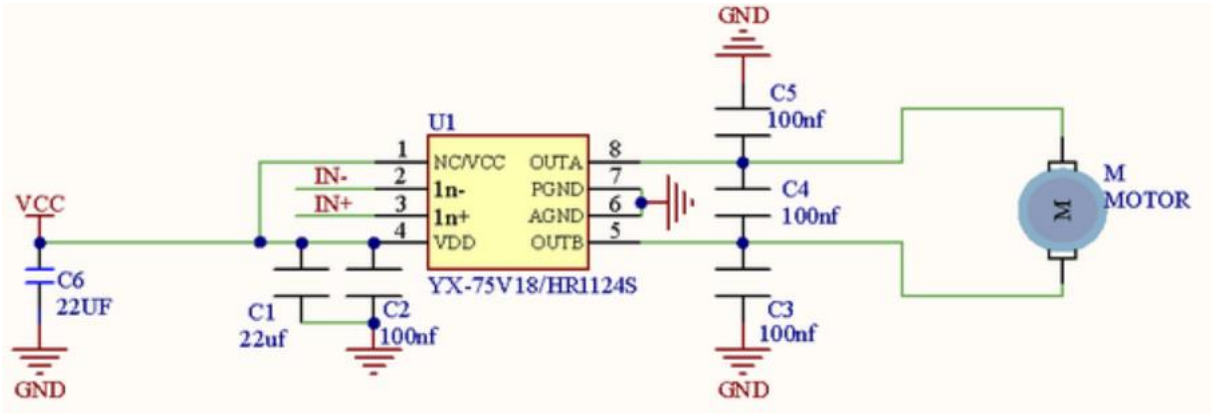
A ventilátor a rendszer aktív beavatkozója, amely a környezeti feltételek megváltoztatására szolgál. A vezérlő digitális kimeneten keresztül kapcsolja a ventilátort közvetlenül vagy meghajtóelemen keresztül.

Szabályozási logika:

- ha a mért hőmérséklet meghalad egy beállított határértéket → ventilátor bekapcsol,
- ha a hőmérséklet a határérték alá csökken → ventilátor kikapcsol.

Oktatási jelentőség:

- bemutatja az egyszerű, kétállapotú szabályozást,
- megérteti a határérték-alapú vezérlés előnyeit és korlátait,
- előkészíti a későbbi hiszterézis- és PID-szabályozás megértését.



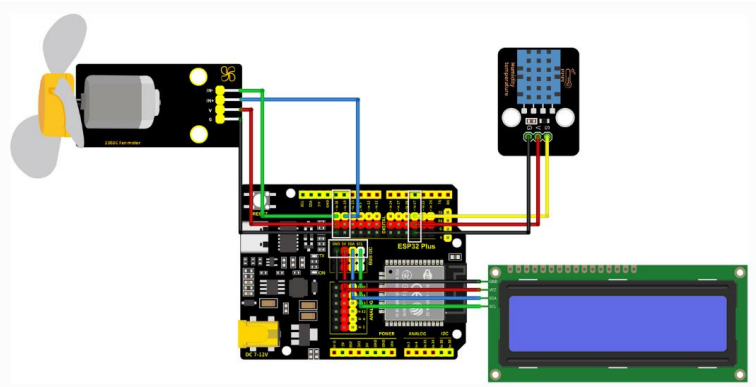
34. ábra: Ventilátorvezérlés kapcsolási rajza (Forrás: https://docs.keyestudio.com)

35. ábra: Ventilátorvezérlés blokk-logikai diagramja (Forrás: https://docs.keyestudio.com)

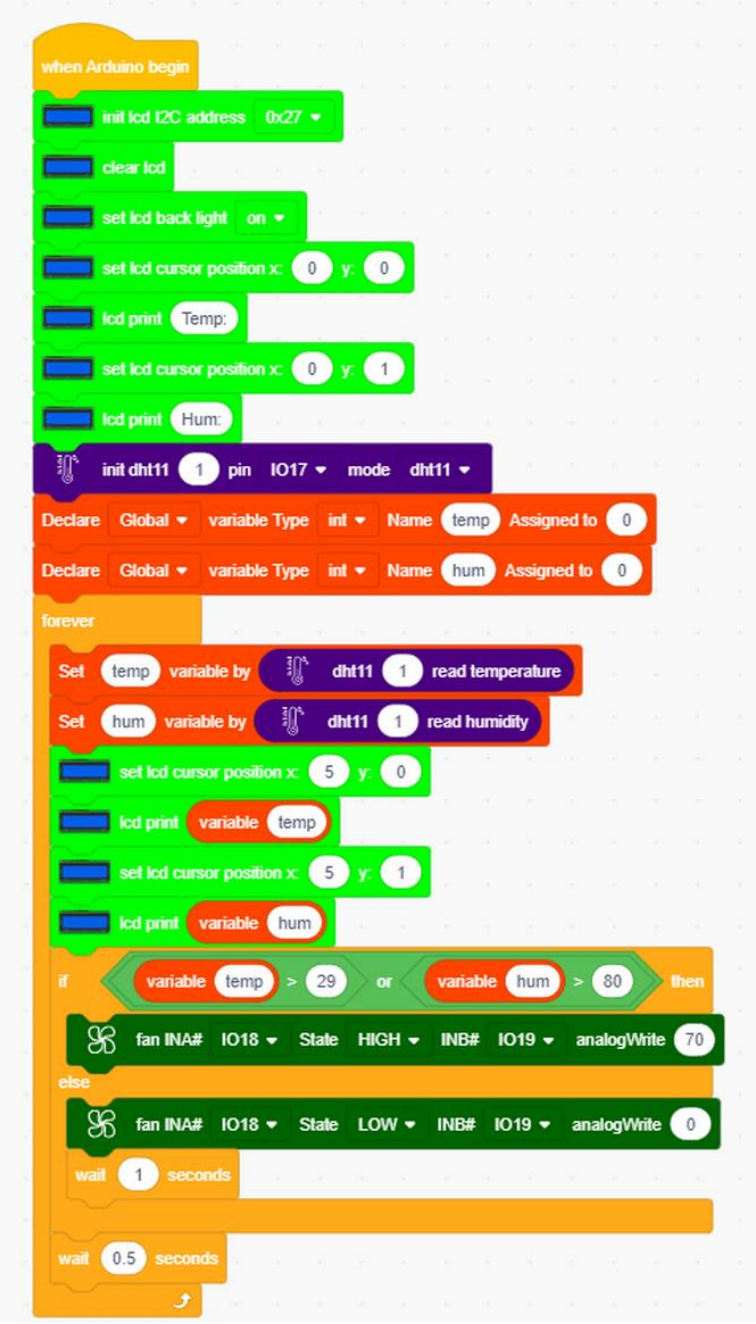
A Temperature Control System integrált működése

A Temperature Control System projekt teljes rendszere az alábbi logika szerint működik:

1. • a DHT szenzor mérést végez
2. • az adat feldolgozásra kerül
3. • a mért érték megjelenik az LCD kijelzőn
4. • a vezérlő összehasonlítja az értéket a határértékkel
5. • a ventilátor állapota ennek megfelelően változik



36. ábra: Temperature Control System bekötési diagramm (Forrás: https://docs.keyestudio.com)



37. ábra: Hőmérséklet-szabályozó rendszer teljes blokk-programja (Forrás: https://docs.keyestudio.com)

Ez a struktúra már egy valódi szabályozási rendszer leegyszerűsített modellje.

Oktatási reflexió

A Temperature Control System projekt lehetőséget ad arra, hogy a tanulók:

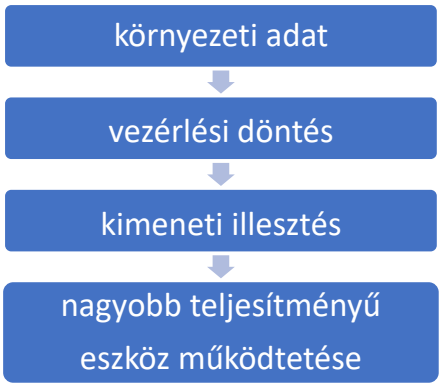
- megértsék a környezeti adatok szerepét az automatizálásban,
- különbséget tegyenek mérés, megjelenítés és beavatkozás között,
- felismerjék a határérték-alapú szabályozás működését és korlátait,
- rendszerszinten gondolkodjanak egy látszólag egyszerű feladat kapcsán.

Ez jól illeszkedik az okosház- és okosfarm-koncepcióhoz, és stabil alapot teremt a későbbi, összetettebb szabályozási és hálózatba kapcsolt megoldások megértéséhez.

3.19. Projekt 5 – Automatikus öntözés (Auto-Irrigation System)

Az Auto-Irrigation System projektben jelenik meg először hangsúlyosan az a mérnöki probléma, hogy a vezérlőegység elektromos képességei nem elegendőek a beavatkozó közvetlen meghajtására. A feladat célja egy olyan automatizált öntözőrendszer modelljének létrehozása, amelyben a döntéshozatal és a fizikai végrehajtás között egy biztonságos, ipari mintát követő illesztés valósul meg.

A projekt működési lánca:



Ez a felépítés alapvető minden valós ipari automatizálási rendszerben.

A mikrovezérlő kimeneteinek korlátai

A Smart Farm Kit vezérlője egy mikrovezérlő-alapú rendszer, amelynek digitális kimenetei alacsony feszültségszinten működnek (jellemzően 3,3 V vagy 5 V DC) és csak kis áram leadására képesek (néhány tíz mA nagyságrendben).

Ez elegendő LED-ek, kisebb elektronikai áramkörök, vagy logikai jelek vezérlésére.

Ugyanakkor nem elegendő motorok, szivattyúk, szelepek, vagy hálózati feszültségű eszközök (pl. 230 V AC) közvetlen működtetésére.

Oktatási szempontból ez kulcspont: a tanulók felismerik, hogy a vezérlés és a teljesítmény nem azonos fogalom.

Feszültségszintek és teljesítménykülönbségek

Az automatikus öntözés jó példája annak, hogy egy rendszerben több feszültségszint él egymás mellett:

- vezérlési oldal: alacsony feszültség (5 V DC),
- beavatkozói oldal: magasabb feszültség vagy teljesítmény (pl. 12 V DC, 24 V DC, akár 230 V AC).

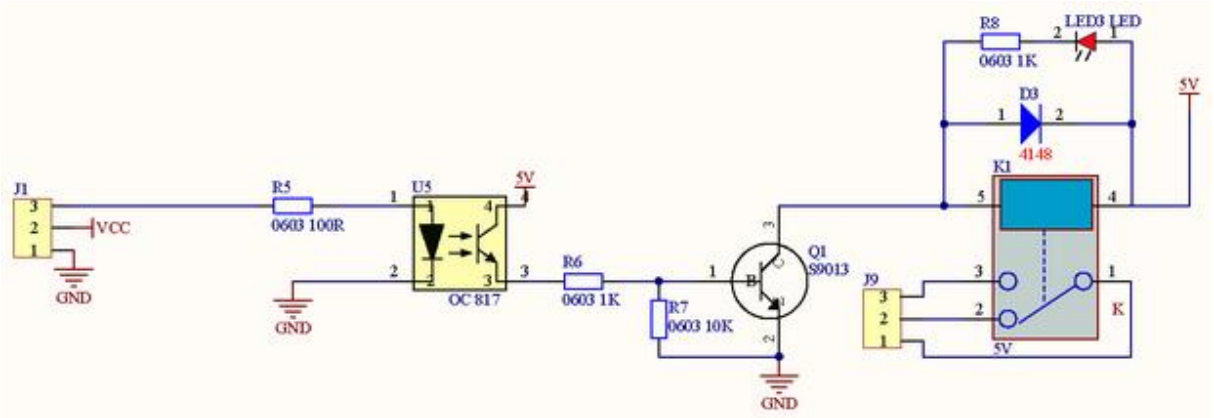
A projekt során hangsúlyossá válik miért nem köthető egy motor közvetlenül a mikrovezérlő kimenetére, miért szükséges teljesítményillesztő elem alkalmazása, valamint hogyan lehet biztonságosan elválasztani a két világot.

Relémodul mint teljesítményillesztő elem

Az Auto-Irrigation System központi eleme a relémodul, amely lehetővé teszi, hogy a mikrovezérlő kis teljesítményű jele egy nagyobb teljesítményű áramkört vezéreljen.

A relé működési elve alapján a vezérlő egy tekercset kapcsol, a tekercs mágneses tere mechanikusan zár vagy bont érintkezőket, az érintkezők pedig egy külön áramkörben működnek.

Ez megvalósítja a galvanikus leválasztást, amely védi a vezérlőelektronikát, növeli a rendszer üzembiztonságát és megfelel az ipari automatizálás alapelveinek.



38. ábra: Relémodul érintkezői és működési logikája (Forrás: https://docs.keyestudio.com)

Ipari párhuzam – PLC-k kimeneti megoldásai

A relé alkalmazása nem „hobbi megoldás”, hanem ipari alapelv. A programozható logikai vezérlők (PLC-k) kimenetei is jellemzően:

- relés kimenetek,
- tranzisztoros (PNP/NPN) kimenetek,
- ritkábban triakos kimenetek váltakozó áramhoz.

A relés PLC-kimenetek előnyei:

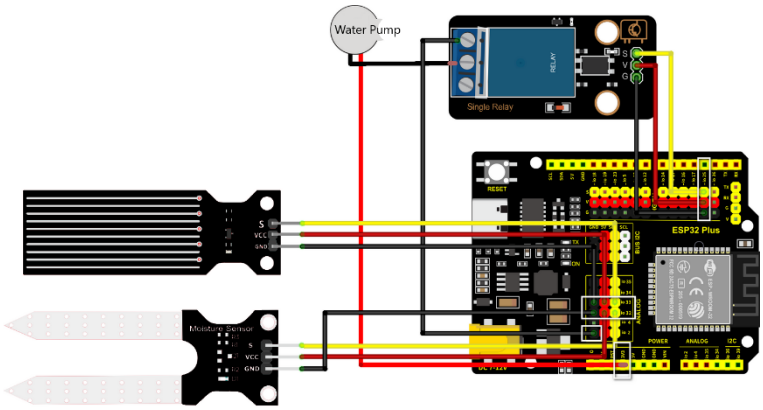
- széles feszültségtartomány,
- AC és DC terhelések kapcsolása,
- elektromos leválasztás.

A Smart Farm projekt ezen a ponton ipari gondolkodásmódot modellez, még akkor is, ha az alkalmazott eszköz oktatási célú.

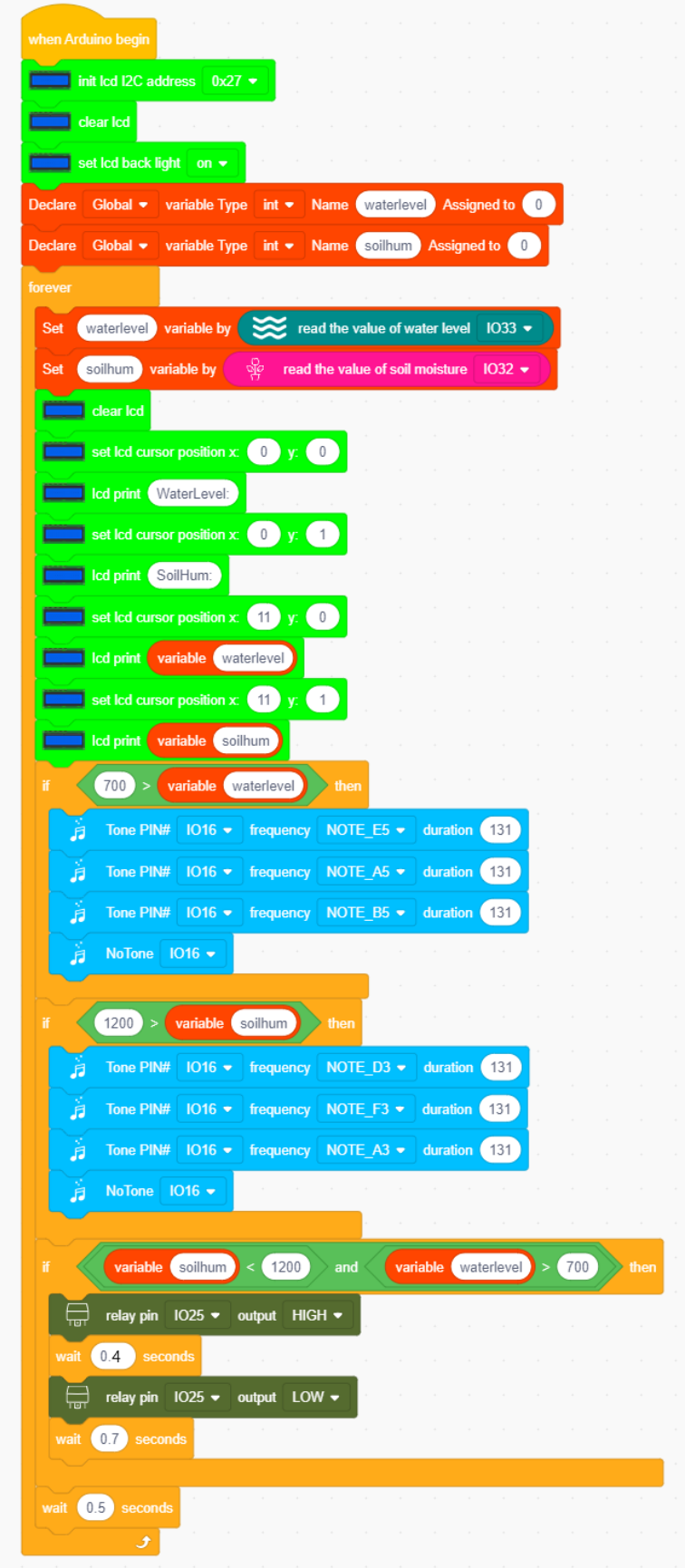
Az Auto-Irrigation System vezérlési logikája

A rendszer működése egyszerű, de teljes automatizmust valósít meg:

1. a vezérlő kiértékeli a bemeneti feltételeket
2. összehasonlítja azokat egy határértékkel,
3. dönt az öntözés szükségességéről
4. digitális kimeneten keresztül aktiválja a relét
5. a relé kapcsolja az öntözőegységet



39. ábra: Auto-Irrigation System bekötése (Forrás: https://docs.keyestudio.com)



40. ábra: Auto-Irrigation System blokk-alapú vezérlés (Forrás: https://docs.keyestudio.com)

Ez a struktúra közvetlenül megfeleltethető ipari folyamatvezérlési ciklusoknak.

Oktatási és mérnöki tanulságok

Az automatikus öntözés projekt során a tanulók:

- megértik a mikrovezérlők fizikai korlátait,
- felismerik a teljesítményillesztés szükségességét,
- találkoznak a galvanikus leválasztás fogalmával,
- ipari automatizálási elvekkel analóg megoldásokat alkalmaznak.

A projekt különösen erős abban, hogy tudatosítja a vezérlési döntések fizikai következményeit, előkészíti a PLC-alapú gondolkodást, valamint megalapozza a későbbi, hálózatba kapcsolt és ipari kommunikációt alkalmazó rendszerek megértését.

Ez a fejezet természetes átmenetet képez az egyszerű IoT-automatizmusok és a professzionális ipari vezérlőrendszerek világa között.

3.20. Projekt 6 – WiFi-vezérelt Smart Farm rendszer

A WiFi-vezérelt Smart Farm projekt jelenti az átmenetet a lokálisan működő automatizmusok és a hálózatba kapcsolt, távolról felügyelhető rendszerek között. Ebben a szakaszban a tanulók már nem csupán szenzorokat és beavatkozókat vezérelnek, hanem egy teljes IoT-alapú felügyeleti modellt valósítanak meg.

A projekt központi gondolata, hogy a rendszer:

- hálózatra csatlakozik,
- adatokat szolgáltat távoli kliens számára,
- HMI webes felületen keresztül vezérelhető.

Ez a működés már túlmutat az egyszerű beágyazott rendszereken, és közelebb visz a valós ipari és okoseszköz-megoldásokhoz.

WiFi kapcsolat szerepe az IoT-rendszerben

A Smart Farm Kit ESP32 vezérlője beépített WiFi modullal rendelkezik, amely lehetővé teszi a vezeték nélküli hálózathoz történő közvetlen csatlakozást. A WiFi kapcsolat ebben a projektben nem csupán technikai kiegészítő, hanem rendszerszintű szemléletváltást jelent: a vezérlés és a felügyelet kilép a lokális, vezetékekkel összekötött környezetből, és hálózati kontextusba kerül.

A hálózatra csatlakozás révén a rendszer nem egy elszigetelt vezérlőként működik, hanem a szenzoradatok távolról, böngészőn keresztül is elérhetők, mivel a beavatkozások nem közvetlen fizikai bemenethez, hanem hálózati kérésekhez kapcsolódnak.

Oktatási szempontból itt válik kézzelfoghatóvá az Internet of Things (IoT) fogalma: az eszköz nem önmagában létezik, hanem egy hálózatba kapcsolt rendszer aktív csomópontja.

A WiFi-kapcsolat kialakítása során a tanulók alapvető hálózati fogalmakkal találkoznak, mint például Hozzáférési pont (Access Point). Az ESP32 egy meglévő WiFi hálózathoz csatlakozik (pl. iskolai vagy otthoni router), amely hozzáférési pontként működik, így megértik, hogy a vezérlő kliensként csatlakozik és a hálózat biztosítja a kommunikáció közegét.

A hálózatra csatlakozott ESP32 egy egyedi IP-címet kap, amely azonosítja az eszközt a hálózaton, lehetővé teszi a böngészőből történő elérést, valamint alapja a későbbi kliens–felügyeleti rendszer kommunikációnak.

Ez az első alkalom, amikor a tanulók megtapasztalják, hogy egy fizikai eszköz „címmel rendelkezik” a hálózaton, hasonlóan más számítógépekhez vagy felügyeleti rendszerekhez.

A projekt során világossá válik, hogy az ESP32, a felhasználó számítógépe vagy mobil eszköze ugyanazon a hálózaton helyezkedik el, vagyis a kommunikáció csak akkor lehetséges, ha ezek az eszközök hálózati szinten „látják egymást”. A vezérlés és adatlekérés hálózati üzenetek formájában történik.

Oktatási jelentőség

A WiFi kapcsolat bevezetése során a tanulók:

- elsajátítják az alapvető hálózati gondolkodást,
- megértik az IP-alapú elérés szerepét a modern rendszerekben,
- felismerik a különbséget a lokális és hálózati vezérlés között,
- megtapasztalják az IoT-rendszerek egyik alapvető működési feltételét.

Ez a tudás közvetlenül előkészíti:

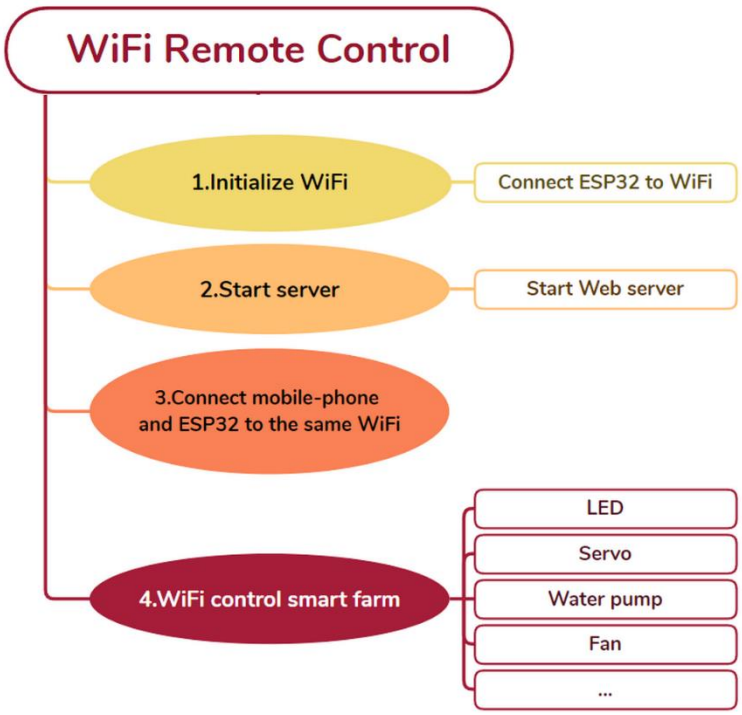
- a webes felügyeleti felület megértését,
- a későbbi kliens–felügyeleti rendszer architektúrák feldolgozását,
- valamint az ipari és elosztott rendszerek hálózati szemléletének befogadását.

Beágyazott webfelügyeleti rendszer működése

A gyári Smart Farm megoldásban a webes felügyeleti felület közvetlenül az ESP32-n futó webfelügyeleti rendszer segítségével valósul meg. Ez azt jelenti, hogy maga a vezérlő szolgáltatja a weboldalt, így nincs külső felügyeleti rendszer vagy számítógép a rendszerben, hanem a kliens (böngésző) közvetlenül az ESP32-höz csatlakozik.

Ez a megoldás oktatási szempontból rendkívül szemléletes, mert egyetlen eszközön belül jelenik meg:

- az adatgyűjtés,
- a feldolgozás,
- a megjelenítés,
- a vezérlés.

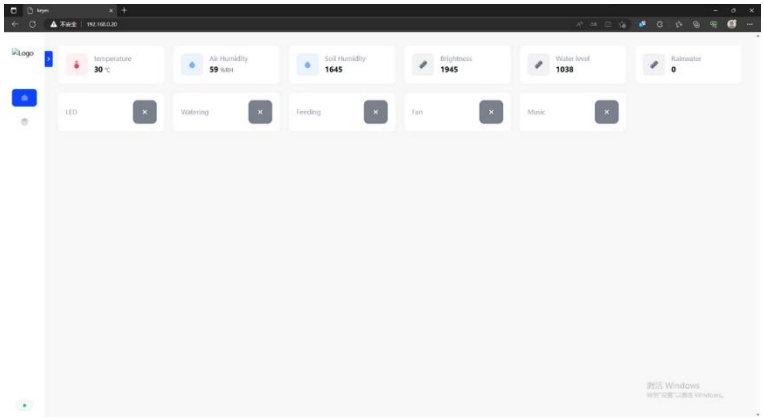


41. ábra: ESP32-n futó webfelügyeleti rendszer kommunikációs modellje (Forrás: https://docs.keyestudio.com)

A webes felügyeleti oldal a rendszer „ablaka” a felhasználó felé. A gyári tutorial alapján a felület jellemzően az alábbi funkciókat valósítja meg:

- aktuális szenzoradatok megjelenítése,
- kimenetek állapotának visszajelzése,
- egyszerű vezérlőelemek (gombok, kapcsolók) biztosítása.

A felület nem grafikai komplexitásában erős, hanem abban, hogy valós időben tükrözi a fizikai rendszer állapotát.



42. ábra: Smart Farm webes felügyeleti oldal megjelenése (Forrás: https://docs.keyestudio.com)

Fontos oktatási elem annak felismerése, hogy a webfelügyeleti rendszer nem korlátlan erőforrásokon fut. Az ESP32 esetében a tanulók szembesülnek az alábbi tényezőkkel:

- korlátozott memória,
- korlátozott tárterület,
- korlátozott számú egyidejű kapcsolat kezelése,

- egyszerű HTML-struktúrák használata.

Ez azt eredményezi, hogy a weboldal funkcionalitása tudatosan egyszerű, így a megjelenített adatok száma korlátozott, vagyis az ESP32 nem alkalmas komplex webalkalmazások futtatására.

Ez a felismerés kulcsfontosságú mérnöki tanulság: minden rendszer kompromisszumok mentén épül fel.

A HMI webes felületen megjelenő vezérlőelemek lehetővé teszik, hogy a felhasználó parancsot küldjön a rendszernek, a vezérlő ezt digitális utasításként értelmezze és a beavatkozók reagáljanak a hálózatról érkező jelre.

Ez a működés bemutatja a modern IoT-rendszerek alapelvét, ahol a felhasználó fizikailag nincs jelen, a vezérlés mégis azonnali és visszacsatolt.

A WiFi-vezérelt Smart Farm projekt során a tanulók:

- megértik a hálózati alapú vezérlés lényegét,
- felismerik a beágyazott rendszerek erőforrás-korlátait,
- megtapasztalják a kliens–felügyeleti rendszer modell alapjait,
- elkülönítik a vezérlési és a megjelenítési feladatokat.

A projekt erőssége, hogy:

- valós IoT-élményt nyújt minimális eszközigénnyel,
- világosan rámutat a gyári megoldások határaira,
- természetes előkészítést ad a későbbi, külső felügyeleti rendszeres, ipari architektúrák megértéséhez.

Ez a fejezet lezárja a gyári Smart Farm rendszer lehetőségeinek feltérképezését, és egyben megteremti az alapot a saját, erőforrás-szétválasztott és ipari mintát követő rendszerarchitektúra kialakításához.

3.21. Összegző pedagógiai értelmezés

A blokk-alapú programozásra épülő Smart Farm projektrész nem önálló tanulási célként, hanem tudatosan megtervezett pedagógiai átmenetként jelenik meg a projekt egészében. Ebben a szakaszban a tanulók olyan alapfogalmakat, összefüggéseket és működési elveket sajátítanak el, amelyek később elengedhetetlenek a komplexebb, ipari jellegű rendszerek megértéséhez és fejlesztéséhez.

Rendszerszintű gondolkodás kialakításának elősegítése miatt a projekt során a tanulók következetesen ugyanazzal az alapmodellel dolgoznak:

Ez a hatáslánc minden projektrészben megjelenik:

- digitális bemenetek (pl. nyomógomb, kapcsoló),
- analóg bemenetek (pl. LDR, potenciométer),
- időalapú mérések (ultrahangos szenzor),
- digitális és PWM-alapú kimenetek (LED, ventilátor, szervómotor),
- relézett teljesítménykapcsolás (öntözőrendszer),
- hálózati kommunikáció és webes vezérlés (WiFi).

A tanulók fokozatosan felismerik, hogy az egyes elemek nem önmagukban értelmezhetők, hanem egy nagyobb rendszer funkcionális részei.

Analóg és digitális világ kapcsolata

Kiemelt pedagógiai eredmény, hogy a tanulók világos különbséget tudnak tenni:

- digitális jelek (kétállapotú, eseményvezérelt),
- analóg mennyiségek (folyamatos értékek),
- az analóg–digitális átalakítás szerepe,
- határértékek, arányos szabályozás és PWM-moduláció között.

A fényvezérlés, a hőmérséklet-szabályozás és a ventilátorvezérlés során megjelenik a mérési zaj fogalma, a döntési küszöb megválasztásának jelentősége, valamint a „mérés ≠ információ” szemlélet.

Beavatkozás és teljesítménykezelés megértése miatt a relémodul és a motoros beavatkozók alkalmazása során a tanulók szembesülnek azzal, hogy a mikrovezérlők kimenetei korlátozott áram- és teljesítményleadásra képesek, ezért a vezérlés és a teljesítményátvitel elválik egymástól, így az alacsony feszültségű logikai jelek (3,3–5 V DC) irányíthatnak nagyobb teljesítményű köröket (pl. 12 V DC, 230 V AC).

Ez közvetlen párhuzamot teremt az ipari vezérlések működésével, ahol a PLC-k relés, tranzisztoros, vagy félvezetős kimeneteken keresztül vezérelnek külső aktuátorokat.

Hálózati gondolkodás és IoT-szemlélet

A WiFi-vezérelt projektben a tanulók megtapasztalják:

- a kliens–felügyeleti rendszer modell alapjait,
- a beágyazott webfelügyeleti rendszer működését,
- a fizikai erőforrás-korlátok (memória, tárhely) hatását a rendszertervezésre,
- a távoli felügyelet és vezérlés lehetőségeit és korlátait.

Ez a tapasztalat előkészíti annak felismerését, hogy ipari környezetben a feladatok gyakran szétválasztásra kerülnek (terepi csomópont (field node) vezérlő – központi felügyelet – backend szolgáltatások).

Programozási szemléletváltás előkészítését segíti a blokk-alapú programozás, mely ebben a kontextusban:

- leválasztja a vezérlési logikát a szintaktikai nehézségekről,
- vizualizálja az állapotkezelést, eseményeket és feltételeket,
- jól megfeleltethető az ipari programozási paradigmáknak (pl. IEC 61131-3, funkcióblokkok, állapotgépek).

A tanulók így nem „Scratch-et tanulnak”, hanem vezérlési gondolkodást sajátítanak el, amely később a C-alapú mikrovezérlő-programozásban, az ipari PLC-kben, a Modbus-alapú kommunikációban, valamint az elosztott rendszerarchitektúrákban is alkalmazható.

Összegzés

Ez a projektfázis stabil alapot teremt a saját tervezésű hardverelemek fejlesztéséhez, a szöveges programozásra való áttéréshez, az ipari szemléletű, elosztott architektúrák megértéséhez és a későbbi felügyeleti kliens–terepi csomópont (field node), backend–frontend felépítésű rendszerek kialakításához.

A blokk-alapú környezet tehát nem végállomás, hanem egy tudatosan felépített tanulási lépcső, amely a tanulókat fokozatosan vezeti el a valós mérnöki és ipari problémák kezelésének irányába.

3.22. A fejezet gyakorlati megvalósulása

A Smart Farm Kit összeszerelése és beüzemelése volt az első igazán „kézzel fogható” élmény a tanulók számára. A gyári készlet működés közbeni megfigyelése gyors sikerélményt adott, ugyanakkor hamar kiderült, hogy a látszólag egyszerű automatizmusok mögött összetett logikai és elektronikai összefüggések állnak.

Komoly kihívást jelentett a gyári dokumentáció feldolgozása, különösen az idegen nyelvű leírások értelmezése. A részleges magyar nyelvű fordítás és oktatási adaptáció során a tanulók aktívan bekapcsolódtak a tartalomértelmezésbe, ami jelentősen mélyítette a megértést. A blokk-alapú programozás kezdetben könnyűnek tűnt, de az állapotkezelés és feltételalapú vezérlés már komolyabb gondolkodást igényelt.

4. Saját okosház / okosfarm modell digitális tervezése (Fusion 360)

Ez a fejezet a projekt egyik meghatározó mérföldköve: itt válik a korábban megismert szenzoros, vezérlési és IoT-logika konkrét, megfogható fizikai struktúrává. A tanulók ebben a szakaszban lépnek át az elektronikai és szoftveres gondolkodás világából a mérnöki tervezés területére, ahol minden döntés geometriai, szerelési és gyártástechnológiai következményekkel jár.

A fejezet célja nem pusztán egy 3D modell létrehozása, hanem annak megértése, hogy a 3D nyomtatásra szánt fizikai modell hordozóstruktúra: helyet kell biztosítania a vezérlőnek, szenzoroknak, aktoroknak és kábelezésnek, miközben támogatnia kell a szerelhetőséget, a mérési tevékenységeket és a későbbi módosításokat. A modell ebben az értelemben tanulási eszköz, nem végtermék.

A tanulási útvonala tudatosan több egymásra épülő lépésből áll. A digitális tervezést megelőzi egy fizikai koncepcióalkotási szakasz, amely a Smart Farm Kit gyári kialakításának elemzésére,

a funkcionális elemek azonosítására és az additív gyártás korlátainak felismerésére épül. Ezt követi a Fusion 360 tervezési környezet megismerése, majd a parametrikus, komponensalapú digitális modell kialakítása, egészen a gyártásra kész állapotig.

A tanulók a folyamat során megtapasztalják, hogy:

- a jó tervezés nem a szoftverrel kezdődik,
- a forma mindig a funkció következménye,
- a parametrikus gondolkodás lehetővé teszi az iterációt és az újratervezést,
- a digitális modell „virtuális prototípusként” segít megelőzni a gyártási és szerelési problémákat.

Ez a rész így hidat képez az elektronikai–informatikai rendszerértés és a fizikai megvalósítás között, és stabil alapot ad a projekt későbbi, 3D nyomtatással és rendszerintegrációval foglalkozó szakaszaihoz.

4.1. Kiindulási alap: a Smart Farm Kit fizikai kialakításának elemzése

A digitális tervezési folyamat első lépése nem a CAD-szoftver használata, hanem a meglévő fizikai megoldások tudatos elemzése. A Smart Farm Kit ebben a projektben referenciarendszerként szolgál: olyan kiindulási alap, amelynek megértése nélkül nem hozható létre jól működő, oktatási célokra optimalizált saját modell.

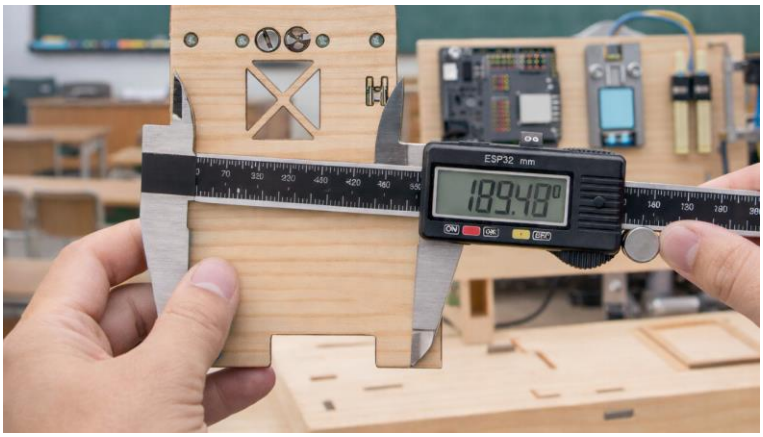
A Smart Farm Kit fizikai kialakítása nem másolandó mintaként, hanem elemzendő példaként jelenik meg. A tanulók célja annak feltárása, hogy:

- mely elemek hordoznak tényleges funkciót,
- hogyan kapcsolódnak az elektronikai komponensek a fizikai térhez,
- mely megoldások szolgálnak oktatási célt, és melyek gyártástechnológiai vagy esztétikai megfontolásból kerültek kialakításra.

Ez a megközelítés már ebben a korai szakaszban rögzíti a funkcióvezérelt tervezés alapelvét: a forma a funkció következménye, nem annak kiindulópontja. A tanulók így megtanulják kritikusan szemlélni a kész megoldásokat, és megkülönböztetni a szükséges elemeket a kontextusfüggő vagy elhagyható részletektől.

A koncepcióalkotás alapját a valós méretek megismerése jelenti. A tanulók a gyári elemeket egyszerű, mindenki számára hozzáférhető eszközökkel vizsgálják. Tolómérő segítségével meghatározzák a befoglaló méreteket, majd milliméterpapírra arányos vázlatokat készítenek és rögzítik a nyílások, rögzítési pontok és szenzorpozíciók helyét.

Pedagógiai szempontból fontos, hogy ebben a fázisban nem készülnek szabványos műszaki rajzok mivel nem a formai pontosság az elsődleges cél, hanem a térbeli viszonyok és arányok megértése.



43. ábra: Smart Farm fizikai elemeinek mérése (Forrás: AI generált)

Ez a lépés közvetlenül előkészíti a későbbi parametrikus gondolkodást, miközben megerősíti a kapcsolatot a fizikai valóság és az absztrakt tervezés között.

4.2. 3D nyomtatási korlátok és tervezési következmények

A gyári elemek elemzése során a tanulók szembesülnek az additív gyártás gyakorlati korlátaival. Felismerik, hogy egyes geometriai részletek nem reprodukálhatók asztali 3D nyomtatással, mivel a túl vékony falak, zárt üregek és alámetszések problémát jelentenek, így a teljes geometriai hűség helyett tudatos egyszerűsítésre van szükség.

Ez a felismerés bevezeti a design for manufacturing (gyártásra tervezés) szemléletét, és rávilágít arra, hogy a tervezési döntések mindig összefüggnek a választott gyártási technológiával.

A fizikai elemzés következő lépése a tudatos döntéshozatal: mit kell megtartani, és mit lehet elhagyni a saját modell kialakítása során.

A tanulók közösen határozzák meg, hogy mely elemek szükségesek a szenzorok, aktorok és vezérlők elhelyezéséhez, azokhoz hol kell rögzítési pontokat biztosítani és mely részek hagyhatók el a tanulási célok sérülése nélkül.

A döntések alapja minden esetben a funkció és az oktathatóság, nem pedig a formai hasonlóság a gyári megoldáshoz. A koncepcióalkotás célja egy olyan fizikai struktúra kialakítása, amely:

- nem tartalmaz felesleges üres tereket,
- anyagtakarékos és gyorsan nyomtatható,
- könnyen szerelhető és módosítható,
- jól átlátható oktatási környezetet biztosít.

Ez a lépés közvetlen hatással van a nyomtatási időre, az anyagfelhasználásra és az iterációs ciklusok gyorsaságára.

4.3. Fizikai koncepció, mint rendszerleírás

Ebben a szakaszban a tanulók a modellt már nem „házként” vagy „dobozként” értelmezik, hanem a szenzorok elhelyezésére szolgáló pontok rendszerének, aktorok mozgástartományát biztosító struktúrának, valamint vezérlő és kábelezés számára kialakított hordozónak.

Ez a szemlélet segít abban, hogy a tervezés ne fulladjon túl korán formai részletekbe, hanem rendszerszinten maradjon értelmezhető.

Parametrikus tervezés fogalmi előkészítése

Bár ebben a fejezetben még nem indul el a Fusion 360 használata, megjelenik a parametrikus tervezés alapgondolata. A méretek nem elszigetelt számok, hanem egymással összefüggő paraméterek, ahol egy módosítás több elemet is érinthet és amelyben a tervezés célja az alkalmazkodóképesség és az újratervezhetőség.

Ez a gondolkodásmód közvetlen átmenetet képez a következő részhez, ahol a fizikai koncepció digitális, parametrikus modellé alakul.

Pedagógiai összegzés

Ez az előkészítő és koncepcióalkotó szakasz biztosítja, hogy a későbbi Fusion 360-alapú tervezés ne puszta szoftverhasználati gyakorlat legyen, hanem tudatos mérnöki döntések digitális leképezése.

A tanulók megtapasztalják, hogy a jó tervezés a számítógép bekapcsolása előtt kezdődik, az egyszerűsítés és az absztrakció mérnöki erény, valamint a fizikai és digitális világ szoros egységet alkot.

4.4. Fusion 360 tervezési környezet – alapok és szemlélet

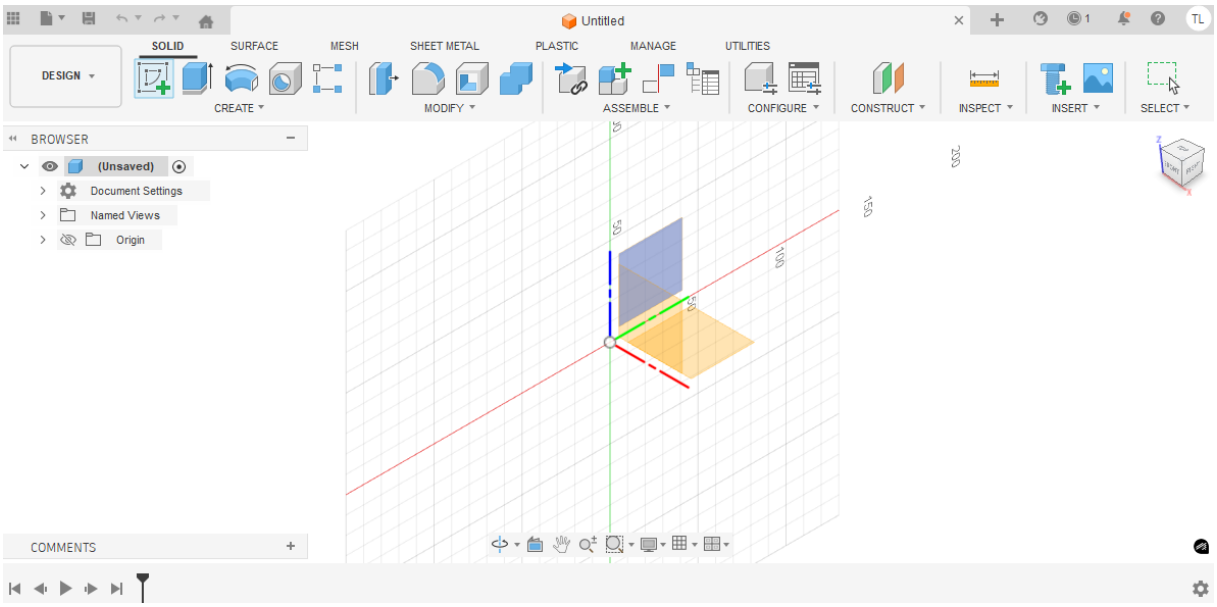
Ebben a szakaszban a tanulók már nem elméleti szinten beszélnek a digitális tervezésről, hanem konkrét műveleteken keresztül tapasztalják meg, hogyan válik egy fizikai koncepció digitális, módosítható modellel leképezhetővé.

A tanulók az Autodesk hivatalos kiadványának a „Autodesk Fusion 360 Training: The Future of Making Things Attendee Guide” segítségével sajátítják el a Fusion 360 szemléletét és használatának alapjait.

A feldolgozott oktatási anyag alapján a tanulási folyamat nem funkciólisták megtanulására, hanem alapvető tervezési rutinok kialakítására épül, például:

- hogyan indítunk el egy új modellt,
- hogyan tájékozódunk a 3D térben,
- hogyan hozunk létre egyszerű, de stabil geometriát.

Pedagógiai cél az, hogy a tanulók felismerjék, hogy a CAD-tervezés tevékenység, nem nézeti kép létrehozása.



44. ábra: Fusion 360 tervezési környezet – alapok és szemlélet (Forrás: Saját szerkesztés)

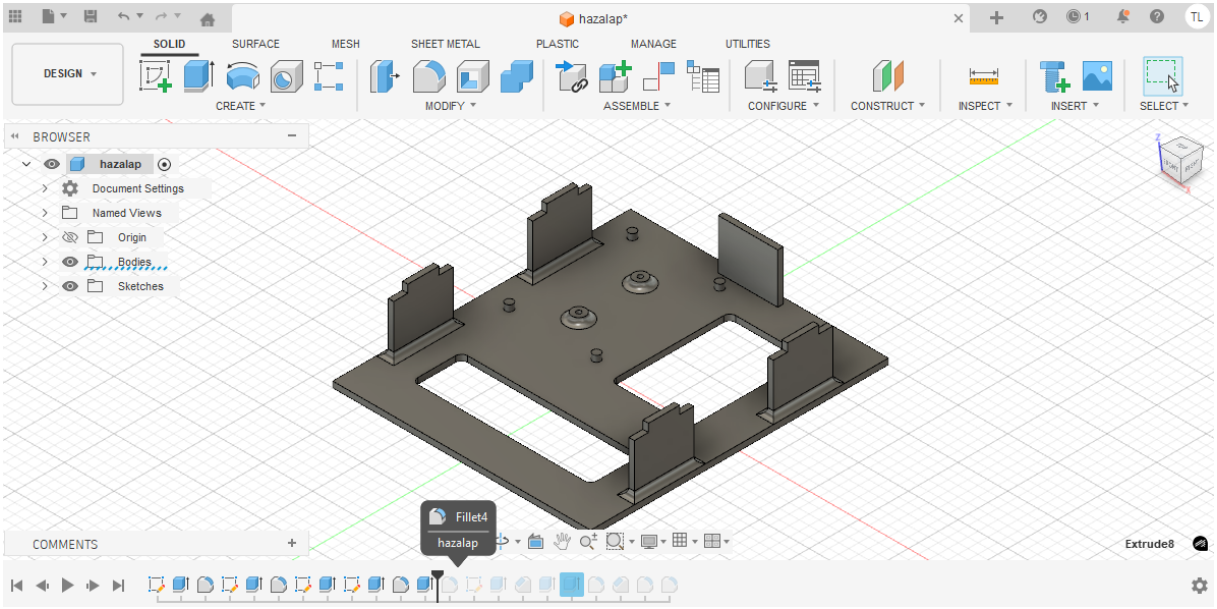
A Fusion 360 az oktatásban „gondolkodó felületként" jelenik meg. A dokumentum hangsúlyozza, hogy már kezdő szinten is teljes tervezési ciklus dolgozható fel, létrehozzák a modellt, módosítják, visszalépnek korábbi állapotokhoz, és megfigyelik a változtatások hatását. Ez a megközelítés bevezeti őket a parametrikus modellezés és a történetalapú tervezés világába, ahol a modell nem statikus objektum, hanem egy folyamat eredménye.

Konkrét tanulói tevékenységek:

- Új Design létrehozása
- Mentés projektbe (felhőalapú működés megértése)
- Visszalépés a Timeline-ban egy korábbi állapothoz

Kulcsfogalmak a dokumentumból:

- Parametric Modeling
- History-Based Design
- Non-destructive Editing



45. ábra: Új Design létrehozása – okosház alap (Forrás: Saját szerkesztés)

Pedagógiai hangsúly: a tanulók megtapasztalják, hogy a modell története ugyanolyan fontos, mint az aktuális állapota.

Az Autodesk Fusion 360 Training anyag kifejezetten kezdőknek mutatja be a felületet, funkciók helyett használati logika szerint.

4.5. Canvas – térbeli mozgás elsajátítása

A térbeli gondolkodás fejlesztése kiemelt szerepet kap. A tanulók aktívan használják a nézetkezelési műveleteket – forgatás, nagyítás, mozgatás –, valamint az alapnézeteket, hogy magabiztosan tudjanak tájékozódni a térben. Ez a készség alapfeltétele annak, hogy később pontos és átgondolt modelleket tudjanak létrehozni.

Tanulói gyakorlatok:

- modell forgatása (Orbit),
- nagyítás/kicsinyítés (Zoom),
- mozgatás (Pan),
- alapnézetek használata (Top, Front, Right).

Pedagógiai cél: a térbeli gondolkodás fejlesztése még a rajzolás előtt.

Browser – a modell „belső térképe”

A modell szerkezetének megértésében a rendszer „belső térképe”, azaz a Browser játszik fontos szerepet. A tanulók megtanulják kezelni a vázlatokat, testeket és komponenseket, valamint felismerik, hogy a jól strukturált modell nemcsak átláthatóbb, hanem később könnyebben módosítható is.

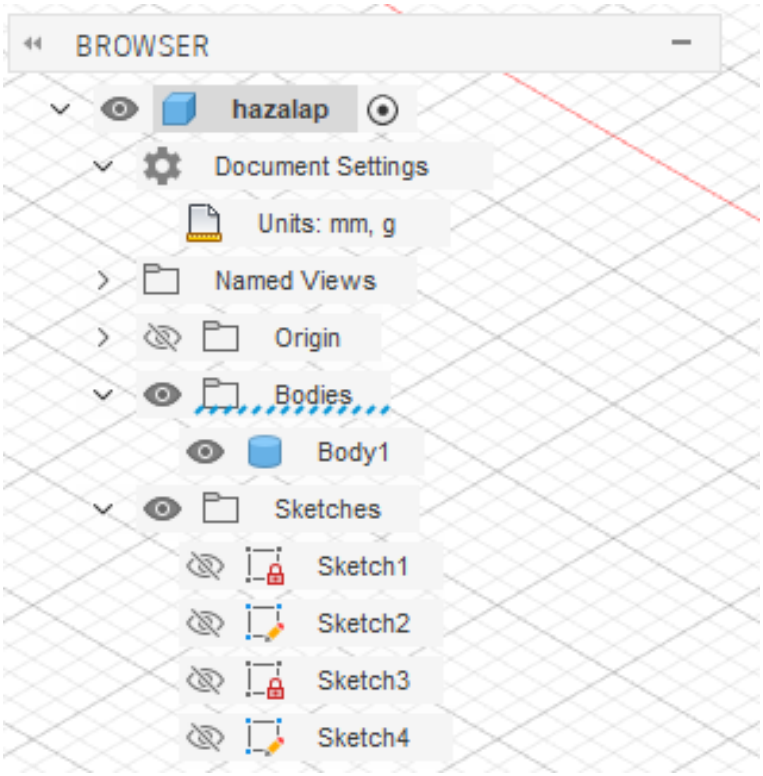
Tanulói tevékenységek:

- Sketch elrejtése/megjelenítése,
- Body-k elkülönítése,

- elemek elnevezése.

Kulcsfogalmak:

- Sketch Folder
- Bodies Folder
- Components



46. ábra: A modell "belső térképe" (Forrás: Saját szerkesztés)

Ezzel párhuzamosan a Timeline használata révén láthatóvá válik a tervezési folyamat okokozati rendszere: egy korábbi lépés módosítása hatással van az egész modellre, ami segíti a rendszerszintű gondolkodás kialakulását.

A dokumentum hangsúlyosan használja a Timeline-t mint tanulási eszközt.

Tanulói gyakorlat:

- egy Extrude lépés módosítása,
- következmények megfigyelése a teljes modellen.

Kulcsfogalom: Design History

47. ábra: Timeline használata (Forrás: Saját szerkesztés)

4.6. Alapvető geometriai és tervezési fogalmak – kézzelfogható módon

Vázlat (Sketch) – nem rajz, hanem szabályrendszer

Kiemelt jelentőségű a vázlatkészítés (Sketch) szemléletének megértése, miszerint nem rajz, hanem szabályrendszer. A tanulók megtapasztalják, hogy a vázlat nem egyszerű rajz, hanem egy matematikailag meghatározott rendszer. A méretek és geometriai kényszerek alkalmazása biztosítja, hogy a modell stabil és egyértelmű legyen. A „nem teljesen definiált” vázlatok problémái gyorsan láthatóvá válnak, így a tanulók saját tapasztalatból értik meg a pontos tervezés jelentőségét.

Az Autodesk Fusion 360 Training szerint a kezdő tervezők leggyakoribb hibája a nem meghatározott vázlat.

Tanulói gyakorlat:

- téglalap rajzolása,
- méretezés hozzáadása,
- kényszerek alkalmazása.

Kulcsfogalmak:

- Fully Constrained Sketch
- Degrees of Freedom
- Geometric Constraints

Pedagógiai felismerés: ami nincs definiálva, az később problémát okoz.

Test (Body) és komponens (Component) – mikor melyiket?

A testek és komponensek közötti különbség szintén fontos mérnöki döntési helyzetként jelenik meg. A tanulók felismerik, hogy míg egy test gyors formaalkotásra alkalmas, addig a komponensek használata már szerkezeti gondolkodást igényel, különösen összetettebb modellek esetén. Ez a különbségtétel előkészíti őket a későbbi összeállítási feladatokra is.

A dokumentum gyakorlati példákon keresztül mutatja meg:

- Body = gyors formaalkotás,
- Component = szerkezeti gondolkodás.

Tanulói döntési helyzet:

- mikor elég egy test,
- mikor indokolt külön komponens.

Kulcsfogalom: Create Component from Body

Paraméterek, mint tervezési eszközök

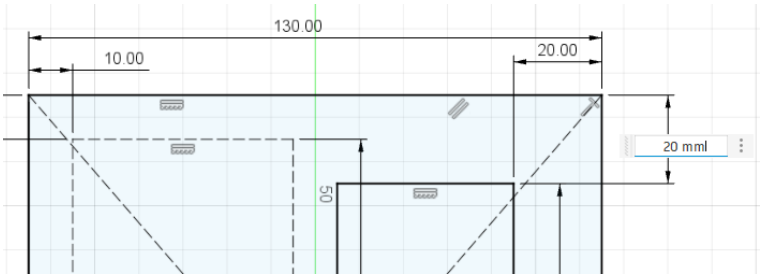
A parametrikus tervezés bevezetése során a diákok megtapasztalják, hogy egyetlen méret módosítása az egész modellre hatással lehet. Ez nemcsak technikai, hanem szemléleti váltást is jelent: a modell már nem egy rögzített forma, hanem egy változtatható rendszer, amely alkalmazkodik a tervezési igényekhez.

Az Autodesk Fusion 360 Training alapján a tanulók kipróbálják:

- egy méret megváltoztatását,
- a teljes modell átalakulását.

Kulcsfogalmak:

- Parameters
- Change Parameters Dialog



48. ábra: Paraméterek használata (Forrás: Saját szerkesztés)

Testmodellezés – alapműveletek biztos kézzel

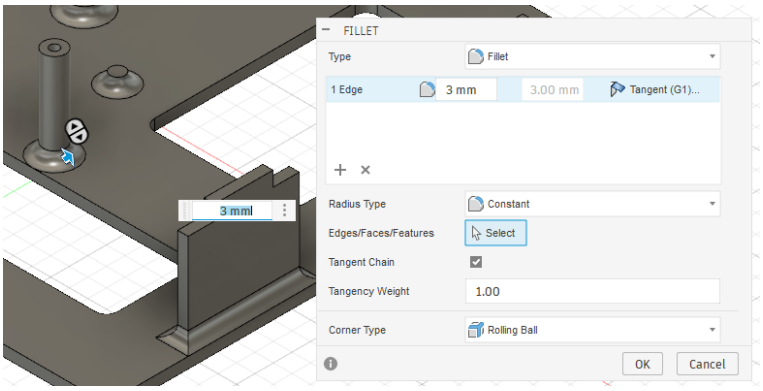
Az alapvető testmodellezési műveletek – például kihúzás, lekerekítés vagy ismétlés – nem önmagukban jelennek meg, hanem mindig konkrét tervezési helyzetekhez kapcsolódva. A tanulók így nem parancsokat tanulnak meg, hanem megértik azok szerepét és következményeit a modell egészére nézve.

Konkrét, tanítható műveletek a Autodesk Fusion 360 Training alapján:

- Extrude (Join / Cut),
- Fillet és Chamfer,
- Pattern (Lineáris ismétlés).

Kulcsfogalmak:

- Join vs. Cut
- Edge Fillet
- Rectangular Pattern



49. ábra: Testmodellezés (Forrás: Saját szerkesztés)

4.7. Előkészítés a 3D nyomtatáshoz – záró gyakorlati fókusz

Az Autodesk Fusion 360 Training 62–65. oldalai kifejezetten a „kész modell” állapotára koncentrálnak.

A tervezési folyamat záró szakaszában a hangsúly a digitális modell fizikai megvalósíthatóságára kerül. A tanulók megtanulják, hogy egy jól megtervezett modell nem feltétlenül alkalmas automatikusan gyártásra, ezért külön figyelmet kell fordítani a 3D nyomtatás követelményeire.

A modell ellenőrzése során elsődleges szempont a zárt geometria biztosítása, mivel a nyomtatás csak egyértelműen definiált térfogatok esetén lehetséges. Emellett vizsgálni kell a falvastagságot, a részletek méretét és a geometria stabilitását is. A tanulók megtapasztalják, hogy a tervezési döntések közvetlen hatással vannak a gyártás minőségére és sikerességére.

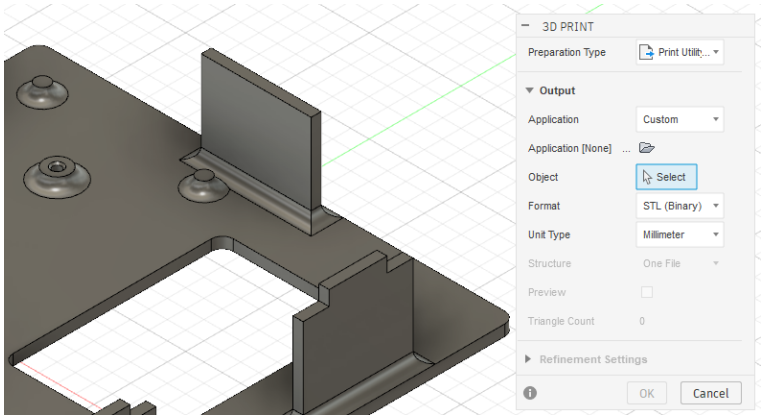
A modell exportálása során megismerkednek a gyártáshoz szükséges formátumokkal (például STL vagy 3MF), valamint azzal a folyamattal, amely során a CAD-modellből nyomtatható adatállomány készül. Ez a lépés lezárja a tervezési ciklust, és egyben visszacsatolást is ad: a fizikai modell elkészítése során azonnal láthatóvá válnak a tervezési hibák és hiányosságok.

Tanulói ellenőrzőlista:

- zárt test (watertight),
- megfelelő falvastagság,
- export STL / 3MF formátumban.

Kulcsfogalmak:

- Mesh Preview
- Export for Manufacturing



50. ábra: Előkészítés 3D nyomtatáshoz (Forrás: Saját szerkesztés)

Didaktikai összegzés

Ez a szakasz különösen fontos didaktikai szempontból, mivel itt válik egyértelművé a tanulók számára, hogy a digitális tervezés nem öncélú tevékenység, hanem a valós fizikai világban megvalósuló rendszerek előkészítése.

A teljes folyamat végére a tanulók nemcsak egy modellt hoznak létre, hanem megértik a tervezés teljes ciklusát: az ötlettől a parametrikus modellen keresztül a gyártható formáig. Ez a tapasztalat alapozza meg azt a mérnöki szemléletet, amelyben a tervezés és a megvalósítás elválaszthatatlan egységet alkot.

A projekt megvalósításának ezen részében a tanulók nem egy szoftvert tanulnak meg kezelni, hanem tervezni kezdenek, lépésről lépésre építik fel első digitális modelljüket, felismerik, hogy a CAD a gondolkodás külső leképezése.

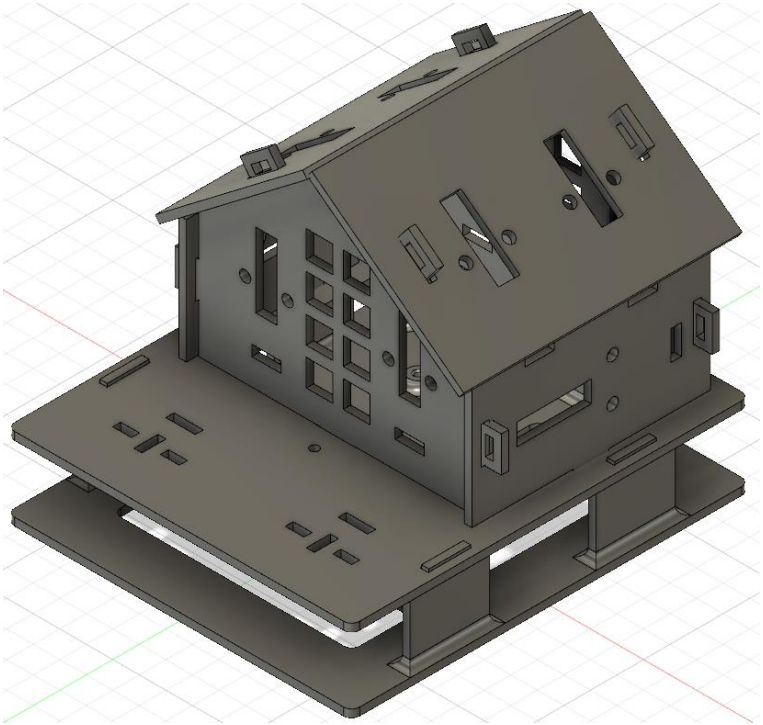
Ez stabil alapot teremt a következő fázishoz, ahol a digitális modell valódi fizikai tárggyá válik a 3D nyomtatás során.

4.8. A digitális tervezési folyamat eredményei – tanulói modellek bemutatása és értelmezése

Ebben az alfejezetben a projekt során elkészült digitális modellek kerülnek bemutatásra. Ezek az elemek nem előre gyártott minták, hanem olyan 3D modellek, amelyeket a tanulók a korábban megismert tervezési elvek alapján, tanári irányítással hoztak létre.

A modellek elkészítése során a tanulók tudatosan alkalmazták a funkcióvezérelt tervezés elvét, a parametrikus gondolkodást, a modularitás és szerelhetőség szempontjait, valamint az additív gyártás (3D nyomtatás) geometriai korlátait.

Pedagógiai szempontból ezek a modellek a tanulási folyamat kézzelfogható lenyomatai: megmutatják, hogy az absztrakt tervezési elvek miként jelennek meg konkrét fizikai formákban.



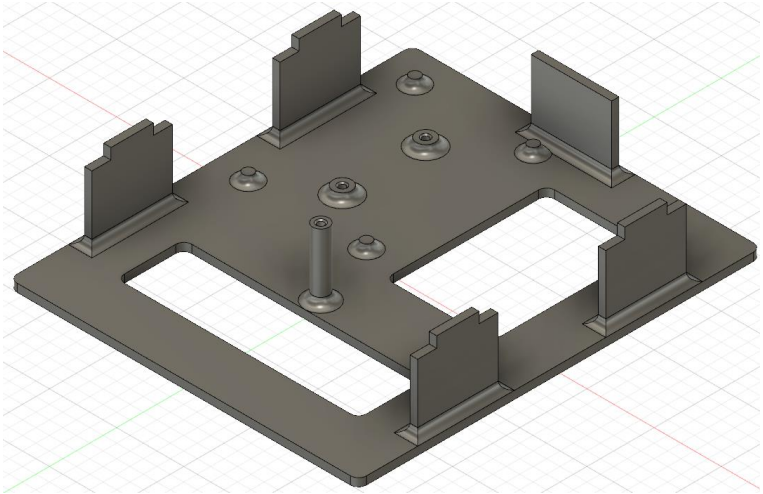
51. ábra: Okosház összeállítási modell (Forrás: Saját szerkesztés)

A tanulók által létrehozott STL-fájlok együtt egy moduláris okosház / okosfarm modell fizikai alapját alkotják. A tervezés célja nem egy esztétikailag végleges tárgy létrehozása volt, hanem egy olyan hordozóstruktúra kialakítása, amely:

- alkalmas elektronikai elemek befogadására,
- támogatja a szenzorok és aktorok rögzítését,
- lehetővé teszi a szerelést, mérést és módosítást,
- megfelel az asztali 3D nyomtatás követelményeinek.

A tanulók számára így világossá vált, hogy a digitális modell nem önálló cél, hanem az elektronikai és vezérlési projektek fizikai kerete.

Alaplap és alsó szerkezeti elemek – tanulói tervezési döntések



52. ábra: Alaplap és alsó szerkezeti elemek (Forrás: Saját szerkesztés)

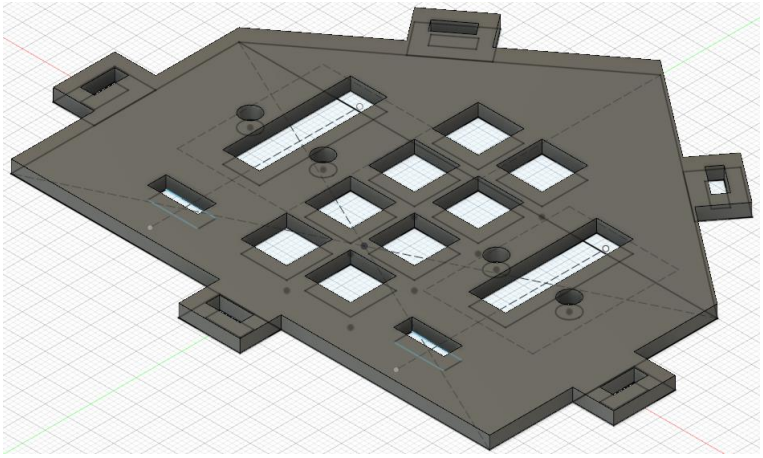
Az alaplap és az alsó fedőelem tervezése során a tanulók a korábban megismert mérnöki szempontokat alkalmazták.

A tervezési folyamatban megjelent döntések:

- stabil, sík hordozófelület kialakítása,
- elektronikai modulok rögzítésének előkészítése,
- kábelek rendezett elvezetésének biztosítása.

Didaktikai jelentőség: a tanulók megtapasztalták, hogy az alapstruktúra minősége meghatározza a teljes rendszer használhatóságát, és hogy a látszólag „egyszerű” elemek mögött is tudatos mérnöki gondolkodás áll.

Oldalpanelek – funkció és hozzáférhetőség



53. ábra: Oldalpanelek (Forrás: Saját szerkesztés)

Az oldalpanelek kialakításakor a tanulók már tudatosan törekedtek arra, hogy a modell ne zárt burkolat legyen, hanem oktatási célokat támogató, „olvasható” szerkezet.

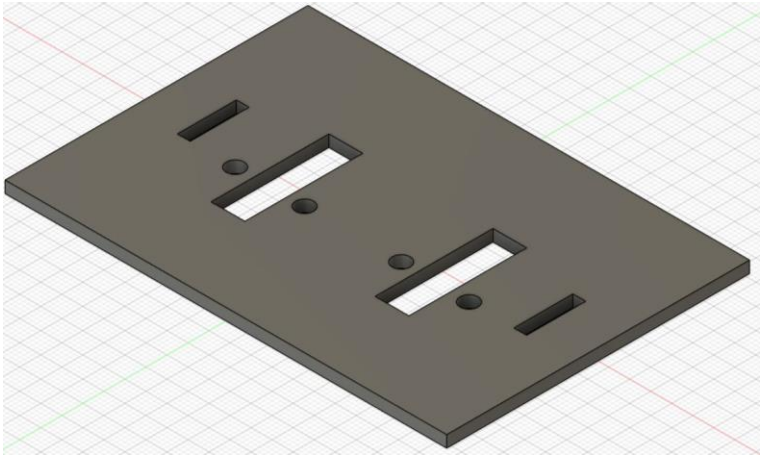
A tervezés során figyelembe vett szempontok a szenzorok és kijelzők helyének biztosítása, a szereléshez és méréshez szükséges hozzáférés, valamint a belső tér vizuális átláthatósága.

Pedagógiai tanulság: a tanulók felismerték, hogy az oktatási modell burkolata nem elrejti, hanem értelmezhetővé teszi a működést.

Tetőelemek – védelem és szétszerelhetőség egyensúlya

A tetőelemek tervezésekor a tanulók olyan megoldásokat alkalmaztak, amelyek védik az elektronikát, ugyanakkor nem akadályozzák a szétszerelést, és támogatják a későbbi módosításokat.

Ez a tervezési feladat jól rávilágított arra, hogy a „lezárás” nem jelent véglegességet, hanem tudatos kompromisszum a védelem és a hozzáférhetőség között.



54. ábra: Tetőelem (Forrás: Saját szerkesztés)

4.9. A tanulási eredmények összegzése a kész modellek alapján

A projekt során elkészült modellek lehetőséget adnak arra, hogy a tanulók visszatekintsenek saját tervezési döntéseikre, és felismerjék azok következményeit.

A modellek révén rögzült kulcsfogalmak:

- parametrikus tervezés,
- modularitás,
- szerelhetőség,
- design for manufacturing,
- iteratív fejlesztés.

Ez a szakasz egyértelművé teszi, hogy a tanulók nem csupán szoftverhasználatot, hanem mérnöki szemléletet sajátítottak el, amely a következő fejezetekben a 3D nyomtatás és a fizikai összeszerelés során válik teljessé.

4.10. A fejezet gyakorlati megvalósulása

Ebben a szakaszban vált igazán láthatóvá, mennyire nehéz a tanulóknak elfogadni, hogy a digitális tervezés alapja a fizikai valóság pontos megértése. A mérések, arányos vázlatok és térbeli viszonyok elemzése eleinte lassúnak és „kevésbé látványosnak” tűnt számukra, később azonban ők maguk ismerték fel, hogy a pontatlanságok a modellben és a nyomtatásnál azonnal visszaütnek.

Az Autodesk Fusion bevezetésekor gyakran kellett tudatosan lassítani a folyamatot. A tanulók rajzolni szerettek volna, miközben meg kellett tanulniuk a parametrikus gondolkodást. A fordulópontot az jelentette, amikor saját tapasztalatból értették meg, hogy egy rosszul felépített vázlat később komoly problémákat okoz, míg az átgondolt struktúra stabil, jól módosítható modellt eredményez.

A saját modell tervezésekor már valódi mérnöki döntések születtek: mit egyszerűsítünk, hol biztosítunk hozzáférést, mi számít valódi funkciónak. Gyakori volt a túlzsúfoltság, amit közösen kellett visszafogni. Tanári szemmel az egyik legnagyobb eredmény az volt, hogy a tanulók gondolkodása átalakult: a modellre nem tárgyként, hanem működő rendszerként kezdtek tekinteni.

5. 3D nyomtatás és gyártási tapasztalatok

A projekt megvalósításának egyik kulcsfontosságú eleme a digitális tervezési folyamatok fizikai kivitelezése volt. A 3D nyomtatás alkalmazása lehetőséget biztosított arra, hogy a tanulók közvetlen tapasztalatot szerezzenek a számítógépes modellezés és a valós gyártás közötti összefüggésekről, valamint megértsék a gyártásra tervezés alapelveinek jelentőségét. A napjainkban népszerű és folyamatosan növekvő felhasználási területekkel rendelkező 3D nyomtatás több tekintetben is ideális választás volt a projekt ezen szakaszának fizikai megvalósítására.

A gyártási fázis során világossá vált, hogy a digitális térben helyesnek tűnő megoldások a fizikai megvalósítás során gyakran módosítást igényelnek. Ez a tapasztalat különösen értékes volt oktatási szempontból, mivel a tanulók nem elszigetelt feladatként, hanem egy összefüggő fejlesztési folyamat részeként találkoztak a problémákkal, azok elemzésével és megoldásával.

5.1. A 3D nyomtató és a Bambu Studio bemutatása

5.2. Az alkalmazott eljárás rövid bemutatása

A projekt során alkalmazott additív gyártási eljárás az FDM (Fused Deposition Modeling) technológia volt, amely napjaink egyik legelterjedtebb és legkönnyebben hozzáférhető 3D nyomtatási módszere. Az eljárás lényege, hogy a nyomtató egy folyamatos műanyag szálat (filamentet) olvaszt meg, majd azt rétegenként, előre meghatározott pályák mentén helyezi el.

Az FDM technológia oktatási környezetben különösen jól alkalmazható, mivel a gyártási folyamat lépései jól nyomon követhetők, és a nyomtatási paraméterek módosításának hatásai azonnal érzékelhetők. A tanulók számára ez lehetőséget teremtett arra, hogy ne csupán a végeredményt szemléljék, hanem a teljes gyártási folyamatot megértsék és értelmezzék.

5.3. Hardveres felépítés és működés

A nyomtatási feladatokat egy **Bambu Lab X1 Carbon** típusú, zárt kialakítású asztali 3D nyomtatóval valósítottuk meg. A berendezés fejlett szenzorrendszerrel, automatikus asztalszintezéssel és nagy pontosságú mozgatórendszerrel rendelkezik, amelyek együttesen stabil és megbízható működést biztosítanak. Azért esett a projekt megvalósítása során erre a nyomtatóra a választásunk, mert a piacon elérhető alternatívák közül olyan eszközt kerestünk, amely nemcsak magas minőségben, számos kalibrációs és testreszabhatósági lehetőséggel képes legyártani a tárgyakat, de mindezt időhatékonyan teszi. A nyomtató használata jelentősen kevesebb háttér kalibrációs folyamatot igényelt, mint más készülékek, amelynek következtében lehetőségünk volt a projekt lényegesebb területeire többletidőt fordítani. Alternatívaként egy Prusa i3 MK3S+ nyomtatót is kiválasztottunk, azonban a könnyebb kezelhetőség és a professzionális kivitel miatt esett végül döntésünk az X1 Carbonra.

55. ábra: Bambu Lab X1 Carbon 3D nyomtató (Forrás: https://www.3djake.hu/bambu-lab/x1c)

A zárt nyomtatótér különösen fontos szerepet játszott a projekt során, mivel a környezeti hatások (hőmérséklet, légmozgás) minimalizálásával csökkentette a nyomtatási hibák előfordulását. Ez lehetővé tette, hogy a tanulók a technológiai összefüggésekre koncentráljanak, és ne a környezeti instabilitásból fakadó problémák dominálják a tapasztalataikat.

5.4. Az AMS filament adagoló rendszer

56. ábra: AMS filament adagoló (Forrás: https://www.3djake.hu/bambu-lab/x1c)

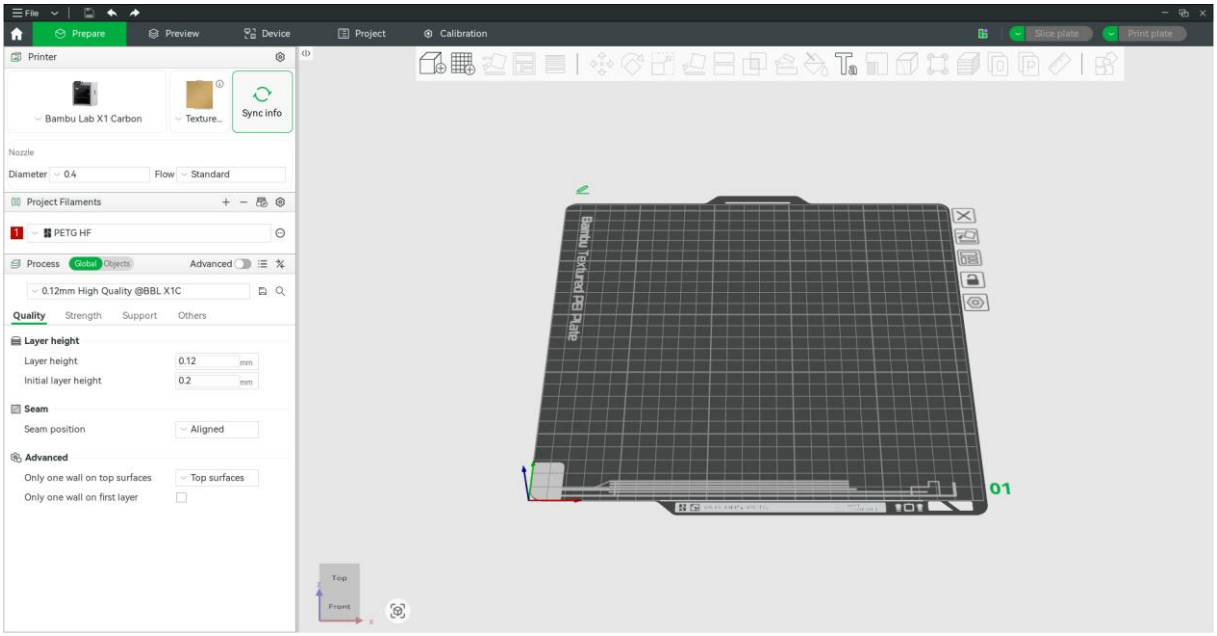
A nyomtatóhoz csatlakozó AMS (Automatic Material System) automatikus filament adagoló rendszer több tekercs alapanyag egyidejű kezelésére képes. Bár a projekt során alapvetően egyszínű és egyféle anyagból készült alkatrészeket gyártottunk, az AMS rendszer használata fontos technológiai ismereteket adott át. Emellett a gyártási folyamatot is elősegítette, mivel nem kellett a kifogyott tekercsek cseréjénél, vagy a színek váltásánál ki- és befűzni a filamentet a nyomtatóba.

A tanulók megismerték az anyagkezelés és az alapanyagok állapotának jelentőségét, különös tekintettel a nedvesség hatására és a tárolás fontosságára, kiemelve az AMS passzív és aktív

filament szárító eszközeit. Emellett az AMS működése jó kiindulási alapot jelentett a többanyagú és több színű gyártási lehetőségek későbbi bemutatásához.

5.5. Szeletelési alapelvek

A digitális modellek gyártható formára alakítását a **Bambu Studio** szeletelő szoftver segítségével végeztük. A Bambu Studio a nyomtató gyártójának (Bambu Lab) saját, nyílt forráskódú szoftvere, amely a PrusaSlicer szeletelő programra alapul és elérhető valamennyi főbb asztali operációs rendszerre.



57. ábra: Bambu Studio szeletelő szoftver (Forrás: Saját szerkesztés)

A szeletelés során a tanulók szembesültek azzal, hogy a geometria önmagában nem elegendő a sikeres nyomtatáshoz; a megfelelő paraméterezés legalább ilyen fontos.

A szeletelési beállítások módosításával érzékelhetővé vált a nyomtatási idő, az anyagfelhasználás és a szerkezeti stabilitás közötti összefüggés. Ez a tapasztalat hozzájárult ahhoz, hogy a tanulók a gyártási folyamatot komplex rendszerként értelmezzék, nem pedig elszigetelt lépések sorozataként. A szeletelési folyamatok során jó kiindulási alapot jelentettek a szoftverbe integrált, alapértelmezett nyomtatási profilok, melyek nemcsak a nyomtatás minőségére, de a felhasznált anyagból eredő optimális konfigurációra is kitértek. Ezek segítségével még szemléletesebbé vált az egyes beállítások közötti eltérés.

A nyomtatási paraméterek megadása mellett hasonlóan fontos szerepet kapott a nyomtatásba foglalt modellek orientációja, színbeállítása, a tálcák előkészítése. Külön figyelmet kellett fordítani a modellek optimális tapadására, valamint az alátámasztások szükségességének vizsgálatára.

5.6. Anyagok (PLA, PETG, ABS)

A projekt során több gyakran használt 3D nyomtatási alapanyag tulajdonságait is megvizsgáltuk, azonban a tényleges gyártáshoz **PETG** anyagot választottunk. A döntés mögött szakmai és pedagógiai megfontolások egyaránt álltak.

A PETG megfelelő egyensúlyt biztosít a könnyű nyomtathatóság és a mechanikai ellenállás között, miközben kevésbé hajlamos a vetemedésre, mint az ABS. Ez különösen előnyös volt a hosszabb nyomtatási idejű, funkcionális alkatrészek esetében. A tanulók így valós mérnöki döntési helyzetben tapasztalhatták meg az anyagválasztás jelentőségét.

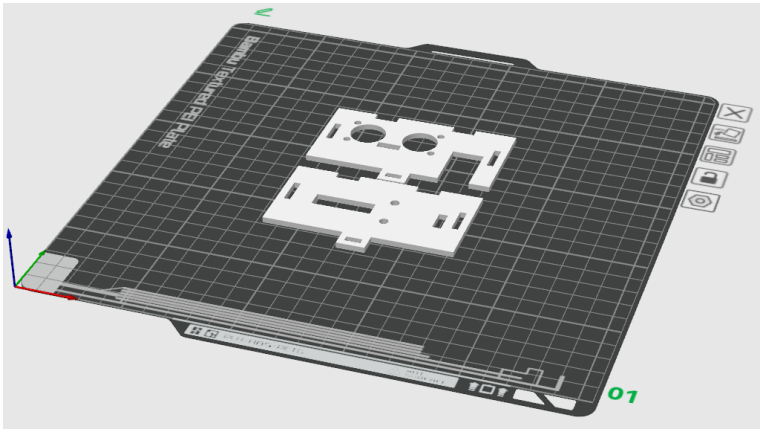
Megjegyeznénk, hogy a projekt megvalósítása során rendelkezésünkre állt PLA filament is, amely alapértelmezett anyagválasztásnak tekinthető a legtöbb 3D nyomtatási projekt esetén, azonban annak tartóssága és fizikai ráhatásokkal szemben tanúsított alacsony ellenállása miatt elvetettük és csak prototipizálási, valamint tesztelési célokkal alkalmaztuk. Az említett három típusú anyagon túlmenően más típusú filament (például TPU) alkalmazását nem vizsgáltuk.

5.7. A nyomtatási folyamat lépései

A 3D nyomtatás gyakorlati megvalósítása strukturált, egymásra épülő lépések mentén történt. A folyamat tudatos felépítése lehetővé tette, hogy a tanulók ne elszigetelt műveletekként, hanem egy összefüggő gyártási rendszer részeként értelmezzék az egyes fázisokat. A nyomtatási lépések során különös hangsúlyt kapott az előkészítés, a folyamatkövetés és az elkészült alkatrészek értékelése.

5.8. Modell-előkészítés és ellenőrzés

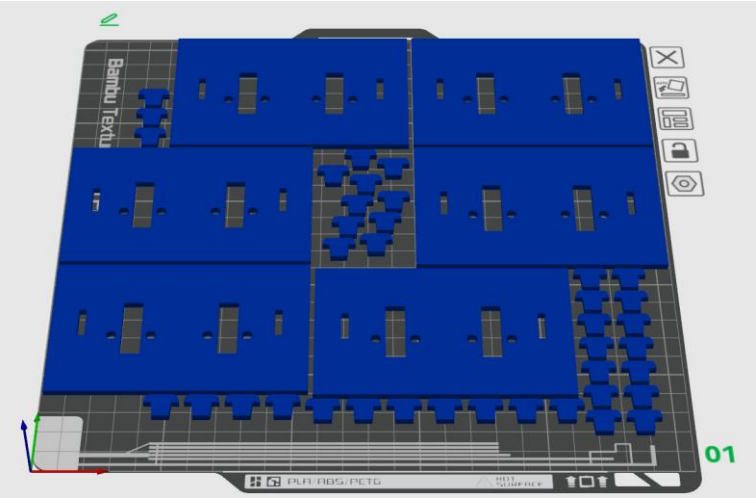
A nyomtatást minden esetben részletes modell-előkészítés előzte meg, amelynek célja a gyártási hibák megelőzése és az erőforrások hatékony felhasználása volt. A tanulók ellenőrizték a digitális modellek geometriai helyességét, különös tekintettel a zárt testekre, a falvastagságokra és a kritikus illesztési felületekre.



58. ábra: Modell-előkészítés és ellenőrzés
(Forrás: Saját szerkesztés)

Az előkészítés során kiemelt szerepet kapott a nyomtatási orientáció megválasztása is. A tanulók megtapasztalták, hogy az alkatrész elhelyezése a nyomtatótérben közvetlen hatással

van a felületi minőségre, a mechanikai szilárdságra és a szükséges támaszanyag, alátámasztás mennyiségére. Ez a döntési helyzet elősegítette a térbeli szemlélet fejlődését, valamint a gyártástechnológiai szempontok tudatos alkalmazását. Magától értetődik, hogy az elemek optimális elrendezésével lehetősége volt a diákoknak optimalizáltabb nyomtatási folyamatokat indítani, mivel egy tálcára több modellt is el tudtak helyezni.



59. ábra: Nyomtatás optimalizálása
(Forrás: Saját szerkesztés)

Az ellenőrzési folyamat végén a modellek szeletelés előtti validálása történt meg, amely megerősítette azt a szemléletet, hogy a hibák korai felismerése lényegesen hatékonyabb, mint azok utólagos javítása.

5.9. Nyomtatás és utómunkák

A nyomtatási folyamat megkezdésekor a tanulók fokozott figyelmet fordítottak az első réteg megfelelő tapadására, amely az egész gyártás sikerességének egyik legkritikusabb tényezője. Az X1 Carbon nyomtató rendelkezik egy beépített AI segéddel, amely az első réteg lehelyezése után rövid időre megállítja a nyomtatást, majd megvizsgálja azt a beépített kamerán keresztül. Amennyiben problémát érzékel, azt jelzi a szeletelő programon (vagy a külön telepíthető Bambu Handy mobilalkalmazáson) keresztül a felhasználónak. A nyomtatás beindítását követően szemléletesebbnek tartottuk, ha a résztvevő tanulók saját szemükkel elemzik a nyomtatás minőségét. Az első réteg megfigyelése lehetőséget adott arra, hogy a tanulók azonnali visszajelzést kapjanak az előkészítés és a beállítások helyességéről.

60. ábra A nyomtató munka közben

A hosszabb nyomtatási feladatok során a folyamat folyamatos felügyelete is fontos tanulási elem volt. A tanulók megtapasztalták, hogy a nyomtatás nem teljesen autonóm művelet, hanem olyan gyártási folyamat, amelynél szükség lehet beavatkozásra vagy újratervezésre. Ilyen eset például, ha egy tárgy elmozdul a helyéről, vagy a filament kifogy az AMS adagolóból. A tanulóknak lehetőségük volt megtekinteni ezeknek a helyzeteknek a kezelését is.

A nyomtatást követően az elkészült alkatrészek utómunkálatai következtek. Ezek közé tartozott a felületi egyenetlenségek kezelése, valamint az illesztési pontok finomítása. Az utómunkák során világossá vált, hogy a 3D nyomtatás olykor félkész állapotú terméket eredményez, amely további manuális munkát igényel a funkcionális használathoz. Ez a modellek összetettségével, valamint az alátámasztások mennyiségének növekedésével fokozódik.

61. ábra: Hibás nyomtatás eredményei (Forrás: Saját szerkesztés)

5.10. Minőségellenőrzés és hibajavítás

Az elkészült alkatrészeket minden esetben részletes minőségellenőrzésnek vetettük alá. A vizsgálat nem csupán esztétikai szempontokra terjedt ki, hanem elsődlegesen a funkcionális megfelelőséget és a szerelhetőséget értékelte.

A tanulók összevetették a nyomtatott alkatrészek méreteit a digitális modell adataival, és elemezték az esetleges eltérések okait. A hibák feltárása során a szeletelési beállítások, az anyagjellemzők és a nyomtatási környezet hatásait is figyelembe vették. A hibajavítás így nem izolált beavatkozásként, hanem a teljes gyártási folyamat visszacsatolásaként jelent meg.

Ez a megközelítés elősegítette az analitikus gondolkodás fejlődését, és megalapozta a következő tervezési iterációk szakmailag indokolt végrehajtását.

5.11. Méretezési és illesztési problémák

A projekt során szerzett tapasztalatok egyik legfontosabb tanulsága a méretezési és illesztési kérdések gyakorlati jelentősége volt. A tanulók közvetlenül megtapasztalták, hogy a digitális tervezés során alkalmazott névleges méretek a gyártás során nem mindig eredményeznek megfelelő illeszkedést, ezért a tűrések tudatos kezelése elengedhetetlen.

5.12. Gyártási pontatlanságok kezelése

A 3D nyomtatás során fellépő pontatlanságok több tényező együttes hatásából adódtak, mint például az anyag zsugorodása, a rétegfelépítés sajátosságai vagy a nyomtatási orientáció. Ezek a jelenségek a tanulók számára kézzelfogható módon tették érzékelhetővé a gyártástechnológia korlátait, valamint tették szükségessé a modellek méretezésének minimális igazítását.

A gyártási eltérések kezelése érdekében a tanulók megtanulták az illesztési hézagok alkalmazását, valamint azt, hogy a mozgó vagy összeillesztendő alkatrészek esetében eltérő tervezési megközelítés szükséges, mint az egy darabból álló elemeknél. Ez a szemlélet hozzájárult a műszaki gondolkodás mélyüléséhez.

5.13. Tervezési iterációk

A méretezési problémák megoldása gyakran több egymást követő tervezési és gyártási ciklust igényelt. Olyan esetek kezelése is szükséges volt, amikor nem a nyomtatási folyamat, hanem tervezése hiba eredményeképp nem valósult meg az elemek egymáshoz illeszthetősége. A tanulók a tapasztalatok alapján módosították a modelleket, majd újranyomtatták az érintett alkatrészeket, így rövid időn belül visszajelzést kaptak döntéseik helyességéről.

Az iteratív folyamat során különösen jól érvényesült a parametrikus tervezés előnye, mivel a méretek gyors módosítása és újragyártása lehetővé tette a hatékony kísérletezést. Ez a módszer segített abban, hogy a tanulók ne statikus megoldásokban gondolkodjanak, hanem rugalmas fejlesztési folyamatban.

5.14. A folyamatok oktatásbeli jelentősége, tanulói tapasztalatok és tanulságok

A méretezési és illesztési problémák kezelése kiemelkedő oktatási értékkel bírt, mivel a tanulók valós mérnöki helyzetekkel szembesültek. A problémák elemzése, a hibák felismerése és a megoldások kidolgozása során fejlődött az önálló gondolkodásuk és a felelősségteljes döntéshozataluk.

A tanulók számára világossá vált, hogy a tervezés és a gyártás szoros egységet alkot, és egyik sem értelmezhető a másik nélkül. A tapasztalatok hozzájárultak ahhoz, hogy a projekt nem csupán technológiai ismereteket adott át, hanem hosszú távon is hasznosítható szemléletet alakított ki.

Emellett a 3D nyomtatásban szerzett tapasztalat további tudástranszfer lehetőségeket rejt, melyet kamatoztathatnak más tantárgyak tanulásánál (matematika, fizika, műszaki szakmai tantárgyak), valamint akár otthoni környezetben is, saját 3D nyomtató üzemeltetése során.

62. ábra Nyomtatott, összerakott ház

6. Integrált vezérlőrendszer – koncepcionális alapok

6.1. A rendszer kialakításának célja

A projekt elektronikai–szoftveres szakaszában tudatos tervezési döntés született: a rendszer nem egyetlen mikrokontrollerre épülő, monolitikus eszköz, hanem elosztott vezérlőarchitektúra formájában valósul meg.

Fontos hangsúlyozni, hogy ez nem az ESP32 „gyengesége” miatt történt. Az ESP32 kiváló terepi vezérlő és adatgyűjtő egység: stabil digitális I/O kezelést biztosít, több kommunikációs interfésszel rendelkezik (I^2C, SPI, UART, WiFi), és önálló vezérlési logika futtatására is alkalmas. Amennyiben a GPIO-k száma nem elegendő, ipari gyakorlatnak megfelelően I/O bővítők (pl. I^2C port expanderek), vagy további csomópontok (node-ok) felfűzése alkalmazható — ez már önmagában az elosztott rendszerek irányába mutat.

A valódi korlát a fejlesztés során nem az I/O vagy a kommunikáció területén jelentkezett, hanem a felügyeleti és megjelenítési (HMI) funkciók megvalósításakor. Az ESP32:

- korlátozott RAM és flash tárterülettel rendelkezik,
- nem webes felhasználói felületek, adatbáziskezelés és tartós naplózás futtatására optimalizált,
- komplex grafikus megjelenítés és több kliens kiszolgálása esetén gyorsan erőforrás-határhoz ér.

Ez a gyakorlatban azt eredményezte, hogy bár az ESP32 képes volt a terepi feladatokat stabilan ellátni (szenzorok, aktorok, lokális vezérlés), a rendszer felügyeleti rétege már indokolttá tette egy nagyobb számítási kapacitású eszköz bevonását.

A kialakított architektúra így világosan elkülönülő rétegekre bontja a rendszert:

Szint	Funkció	Eszköz
Terepi szint	Szenzoradat-gyűjtés, aktorvezérlés, lokális logika	ESP32
Vezérlési / adatkezelési szint	Ciklikus feldolgozás, logikai döntések, Modbus master szerep	Raspberry Pi / PC (Modbus)
Felügyeleti szint (HMI)	Webes megjelenítés, felhasználói vezérlés, adatnaplózás	Raspberry Pi / PC (Flask alapú szerver)

Ebben a modellben az ESP 32 terepi csomópont (field node) szerepkörben működik, és ipari mintát követve szabványos protokollon (Modbus TCP) keresztül szolgáltat adatokat a magasabb szintű rendszernek. Fontos, hogy egy tisztán monolitikus, egyeszközös rendszerben erre a kommunikációs szabványra valóban nem lenne szükség — az elosztott architektúra azonban indokolttá teszi.

A projekt célja tehát nem pusztán egy működő okosfarm-modell létrehozása volt, hanem annak demonstrálása, hogy:

- hogyan válik egy mikrokontroller terepi csomóponttá,
- hogyan szerveződik fölé egy felügyeleti rendszeroldali logikai és HMI réteg,
- és hogyan modellezhető oktatási környezetben az ipari automatizálás rétegzett, hálózati felépítése.

Ez a szemlélet közelíti a tanulók számára a valós ipari rendszerek működését, miközben a rendszer továbbra is könnyen bővíthető új szenzorokkal, csomópontokkal és szolgáltatásokkal.

Az elosztott rendszer koncepciója

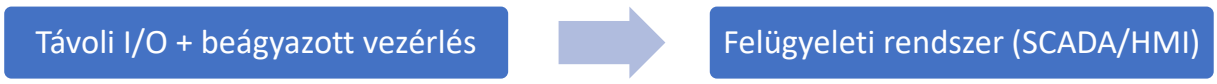
A projektben kialakított vezérlési modell tudatosan egy két szintű, rétegzett architektúrát követ. Ez a felépítés nemcsak technikai döntés, hanem szemléletbeli lépés is volt a monolitikus, „mindent egy mikrokontrolleren” megközelítéstől az ipari rendszerekre jellemző struktúra felé.

Az alsó szintet a terepi/mezőszint (field level) alkotja, ahol az érzékelők és beavatkozók közvetlen kezelése történik. Ezt a szerepet a rendszerben az ESP32 tölti be. Feladatai közé tartozik a digitális és analóg jelek kezelése, a szenzoradatok begyűjtése, az alapvető előfeldolgozás, valamint a fizikai kimenetek (pl. LED, relé, szervó) működtetése. Ezen a szinten zajlik a valós fizikai folyamattal való közvetlen kapcsolat.

A felső réteget a felügyeleti szint (supervisory level) jelenti, amelyet a projektben egy Raspberry Pi (vagy PC) alapú rendszer valósít meg. Fontos különbség, hogy ezen a szinten nem a központi vezérlési logika fut, hanem a rendszer digitális állapotképe jelenik meg. A Pi a mezőszintű eszközök (ESP32) adatait gyűjti, naplózza és HMI webes felületen megjeleníti, valamint lehetőséget ad távoli beavatkozásra.

A tényleges vezérlés a mezőszinten, az ESP32-n történik. Az I/O kezelés, a szenzoradatok feldolgozása és az alap működési logika helyben fut, ezért a rendszer offline módban is működőképes marad. A felügyeleti réteg kiesése nem állítja le a fizikai folyamatot, csak a megjelenítést és a távoli elérést érinti.

Ez a struktúra így pontosabban a következő ipari modellhez hasonlítható:



A projektben ennek megfelelői:

- ESP32 mint intelligens terepi egység, amely
  - I/O kezelést végez
  - szenzoradatokat gyűjt
  - lokális vezérlési logikát futtat
  - Modbus felügyeleti rendszerként kommunikál
- Raspberry Pi + Python + Flask mint felügyeleti réteg, amely
  - a rendszer állapotát tükrözi
  - adatokat gyűjt és megjelenít

- operátori beavatkozást tesz lehetővé
- de nem kritikus a folyamat működéséhez

A jövőben a Raspberry Pi szinten megvalósíthatók olyan kiegészítő funkciók, mint például:

- adatnaplózás és hosszú távú trendanalízis
- grafikonos megjelenítés
- felhasználói jogosultságkezelés
- értesítések, riasztási naplók
- távoli konfigurációs felület

Ezek a funkciók csak felügyeleti és információs jellegűek lehetnek. Tervezési alapelv, hogy a Raspberry Pi-n futó kiegészítések nem befolyásolhatják a rendszer alapműködését vagy biztonságát. A kritikus vezérlési logika és az alapállapotok kezelése továbbra is a terepi szinten marad.

6.2. Az elosztott architektúra műszaki előnyei

A rendszer egyik kulcsa a funkcionális szétválasztás, amely közvetlenül növeli a megbízhatóságot és a bővíthetőséget. Az ESP32 a terepi szinten marad: szenzorok, bemenetek, kimenetek és az alap vezérlési logika helyben fut, így a fizikai folyamat működése nem függ folyamatos hálózati kapcsolattól. Ez determinisztikusabb I/O viselkedést és stabilabb működést eredményez.

A felső szint feladata a felügyelet, adatkezelés és megjelenítés, nem pedig a kritikus vezérlés. Ez a rétegzett felépítés ipari mintát követ, ahol a terepi eszköz, a vezérlési szint és a felügyeleti rendszer külön szerepkörben működik.

Az architektúra jelentős előnye a skálázhatóság: új ESP-alapú node-ok illeszthetők a hálózathoz, további szenzorok integrálhatók, és a rendszer funkcionálisan bővíthető anélkül, hogy egyetlen eszköz erőforrásaira terhelnénk mindent. Ez a horizontális bővítés az elosztott rendszerek alapvető sajátossága.

A megbízhatóság szempontjából alapelv, hogy a mezőszint autonóm marad: a felügyeleti rendszer vagy a hálózat kiesése nem állítja le a helyi működést, csak a monitorozást és a távoli beavatkozást érinti. Ez közvetlenül megfelel az ipari automatizálás gyakorlatának.

6.3. Kommunikációs modell és szemlélet

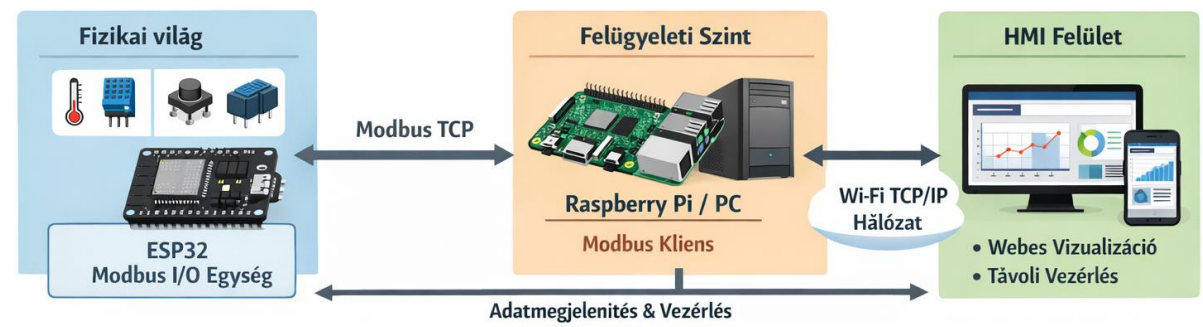
A rétegek közötti kapcsolat Modbus TCP alapú. Ez állapotalapú adatcserét valósít meg, szabványos regiszterstruktúrával. A szenzoradatok és vezérlőjelek nem konkrét „kérések–válaszok” formájában mozognak, hanem jól definiált címeken elérhető állapotokként. Ez eltér a tipikus HTTP-alapú, tranzakciós szemlélettől, és közelebb áll az ipari kommunikációs modellekhez, ahol a vezérlő ciklikusan olvassa a mezőszint állapotait, majd ennek megfelelően írja a kimeneti regisztereket.

A rendszer váza – logikai felépítés

Magas szinten a rendszer adatútja a következőképpen írható le:

A szenzorok (gomb, DHT stb.) és az aktorok a fizikai világban az ESP32-höz csatlakoznak. Az ESP32 Modbus felügyeleti rendszerként működik, és regisztereken keresztül teszi elérhetővé a bemenetek és kimenetek állapotát. A WiFi-alapú TCP hálózaton keresztül a felügyeleti szinten futó rendszer – Raspberry Pi vagy PC – Modbus kliensként ciklikusan olvassa és írja ezeket az adatokat. A feldolgozott állapotok és mért értékek ezután a HMI webes felületen jelennek meg, ahol vizualizáció és távoli vezérlés is megvalósul.

Ebben a modellben az ESP nem központi vezérlő, hanem intelligens terepi I/O egység, míg a Raspberry Pi tölti be a logikai és felügyeleti központ szerepét.



63. ábra: A rendszer váza (Forrás: AI generált)

Oktatási jelentőség

Az alkalmazott architektúra nemcsak műszaki megoldás, hanem szemléletformáló eszköz. A tanulók megértik a terepi és a felügyeleti szint szétválasztásának okát, az erőforrás-elosztás szerepét, valamint azt, hogyan épül fel egy ipari automatizálási rendszer rétegzett struktúrája. Ez közvetlen alapot ad a PLC-programozási, HMI-tervezési és SCADA-rendszerek megértéséhez, mert a projektben alkalmazott modell ezek leegyszerűsített, de valósághű leképezése.

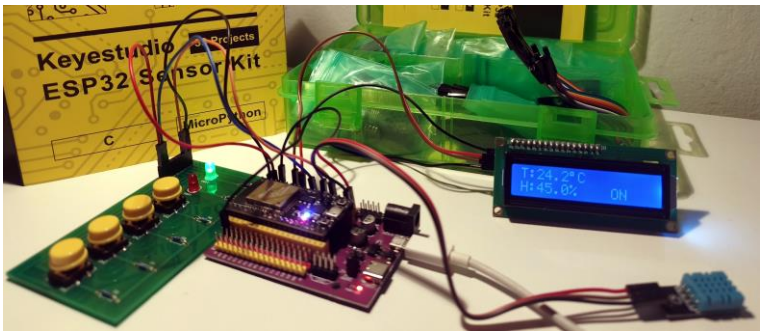
A következő részben az architektúra konkrét szoftveres megvalósítását mutatjuk be: az ESP terepi programját, a felügyeleti oldali ciklikus feldolgozást, valamint a webes kezelőfelület működését.

6.4. Terepi vezérlőcsomópont – önálló működésű mintaegység

A projekt elektronikai rendszerének alsó szintjét egy önállóan működő terepi csomópont (field node) képviseli, amely egy ESP32 mikrokontrollerre épül. Ez az egység közvetlenül kapcsolódik a fizikai világhoz: érzékelőket olvas, beavatkozókat vezérel, és helyi visszajelzést biztosít. A bemutatott konfiguráció tudatosan egyszerű, oktatási célú minta, amely a rendszer alaplogikáját szemlélteti, nem pedig végleges kiépítést jelent.

A csomópont a következő elemeket kezeli:

- egy digitális bemenetként működő nyomógomb,
- egy digitális kimeneti LED,
- egy DHT11 hőmérséklet- és páratartalom-érzékelő,
- egy I²C buszon csatlakozó, 16×2 karakteres LCD kijelző.



64. ábra: Tesztkörnyezet (Forrás: Saját szerkesztés)

A működés lényege, hogy az eszköz teljesen önállóan, hálózati kapcsolat nélkül is ellátja alapfeladatait. A felhasználó a fizikai gombbal képes beavatkozni a rendszer állapotába (LED ki-/bekapcsolás), miközben a környezeti adatok folyamatosan mérésre kerülnek és megjelennek a kijelzőn. Ez a megközelítés azt az ipari alapelvet modellezi, hogy a terepi szintnek képesnek kell lennie autonóm működésre, függetlenül a felügyeleti rendszertől.

A bemutatott program tehát egy intelligens I/O egység viselkedését demonstrálja. A struktúra moduláris: új szenzorok, kimeneti eszközök vagy kijelzők egyszerűen hozzáadhatók további GPIO-k vagy kommunikációs buszok felhasználásával. A későbbiekben ez a csomópont magasabb szintű rendszerekhez (PLC, HMI, SCADA) is illeszthető, de jelen fejezetben kizárólag a helyi erőforrások kezelésére koncentrálunk.

6.5. A program felépítése funkcionális egységekre bontva

Könyvtárak és hardverdefiníciók

```
// ================= KÖNYVTÁRAK BETÖLTÉSE =================
// Ebben a szakaszban történik a felhasznált perifériákhoz szükséges könyvtárak betöltése.
// DHT könyvtár a szenzor kommunikációját valósítja meg.
// A Wire és hd44780_I2Cexp az I²C LCD vezérléséhez szükséges.
#include <DHT.h>
#include <Wire.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h>


// ================= HARDVER KIOSZTÁS =================
//A #define direktívák rögzítik a fizikai bekötés és a program közötti kapcsolatot, így a hardverkonfiguráció egyértelműen dokumentált és könnyen módosítható.

#define BTN 17      // Fizikai gomb bemenet
#define LED 2       // LED kimenet
#define DHTPIN 4    // DHT adatvonal
#define DHTTYPE DHT11 //DHT típus megadása
```

Ebben a szakaszban történik a felhasznált perifériákhoz szükséges könyvtárak betöltése.

- A DHT könyvtár a szenzor kommunikációját valósítja meg.
- A Wire és hd44780_I2Cexp az I²C LCD vezérléséhez szükséges.

A #define direktívák rögzítik a fizikai bekötés és a program közötti kapcsolatot, így a hardverkonfiguráció egyértelműen dokumentált és könnyen módosítható.

Objektumok és globális változók

```
//Itt jönnek létre a szenzor- és kijelzőobjektumok. Ezek a program teljes futása alatt elérhetők, így bármelyik ciklusban használhatók.
// Szenzor objektum
DHT dht(DHTPIN, DHTTYPE);
// I2C LCD objektum
hd44780_I2Cexp lcd;
```

Itt jönnek létre a szenzor- és kijelzőobjektumok. Ezek a program teljes futása alatt elérhetők, így bármelyik ciklusban használhatók.

Gomb prellegés (debounce) kezelése

```
// ================= GOMB PRELLEGÉS SZŰRÉS =================
// A mechanikus gombok érintkezői záráskor rövid ideig „rezegnek” (prellegnek), ami több hamis jelváltást okozhat.
// Ez a logika időalapú szűréssel biztosítja, hogy csak a stabil állapotváltás legyen érvényes.

bool lastReading = false;      // Legutóbbi nyers bemenet
bool stableState = false;      // Szűrt, stabil állapot
unsigned long lastChangeTime = 0;
const unsigned long debounceTime = 50;  // 50 ms stabilitási küszöb
```

A mechanikus gombok érintkezői záráskor rövid ideig „rezegnek” (prellegnek), ami több hamis jelváltást okozhat. Ez a logika időalapú szűréssel biztosítja, hogy csak a stabil állapotváltás legyen érvényes.

LED állapot és mérési változók

```
// A LED állapota külön változóban tárolódik. A DHT mérések időzítéséhez és az aktuális értékek tárolásához is globális változók szükségesek.
bool ledState = false;
unsigned long lastDHTread = 0; // Utolsó olvasás időpontja
float temp = 0;                // Hőmérséklet cache
float hum = 0;                 // Páratartalom cache
```

A LED állapota külön változóban tárolódik. A DHT mérések időzítéséhez és az aktuális értékek tárolásához is globális változók szükségesek.

Inicializálás – setup()

```
void setup() {

  // Hardver inicializálás
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);

  // DHT inicializálás
  dht.begin();

  // I2C inicializálás ESP32 alapértelmezett lábakkal
  Wire.begin(21,22);

  // LCD inicializálás
  lcd.begin(16,2);
  lcd.backlight();

  lcd.setCursor(0,0);
  lcd.print("Smart Farm Node");
  delay(2000);
  lcd.clear();
```

A setup() egyszer fut le induláskor. Itt történik:

- a GPIO irányok beállítása,
- a szenzor és az I²C busz inicializálása,
- az LCD indítása.

Gombkezelés és LED váltás

Ez a blokk valósítja meg a lenyomási élre történő LED-váltást. A LED csak akkor változik, amikor a gomb stabilan lenyomott állapotba kerül.

```
74    //Ez a blokk valósítja meg a lenyomási élre történő LED-váltást.
75    //A LED csak akkor változik, amikor a gomb stabilan lenyomott állapotba kerül.
76    // Gomb olvasása
77    bool reading = digitalRead(BTN);
78
79    // Prellegés detektálása
80    if (reading != lastReading) {
81      lastChangeTime = millis();
82    }
83
84    // Stabil állapot vizsgálata
85    if ((millis() - lastChangeTime) > debounceTime) {
86
87      // Állapotváltozás kezelése
88      if (reading != stableState) {
89        stableState = reading;
90
91        // Lenyomás detektálás
92        if (stableState == true) {
93          ledState = !ledState;         // LED kapcsolás
94        }
95      }
96    }
97
98    lastReading = reading;
```

DHT11 periodikus mérés

```
100   // DHT szenzor olvasás időzítetten
101   //A szenzor csak 2 másodpercenként kerül kiolvasásra, ami megfelel a DHT11 időzítési követelményeinek.
102   if(millis() - lastDHTread > 2000){
103     lastDHTread = millis();
104
105     float h = dht.readHumidity();
106     float t = dht.readTemperature();
107
108     // Hibás olvasás kizárása
109     if(!isnan(h) && !isnan(t)){
110       temp = t;
111       hum = h;
112     }
113   }
```

A szenzor csak 2 másodpercenként kerül kiolvasásra, ami megfelel a DHT11 időzítési követelményeinek.

LCD kijelzés

```
115   // LCD kijelzés
116   //A kijelző a mért adatokat és a LED állapotát jeleníti meg, így a rendszer állapota hálózat nélkül is
      ellenőrizhető.
117   lcd.setCursor(0,0);
118   lcd.print("T:");
119   lcd.print(temp,1);
120   lcd.print((char)223);
121   lcd.print("C ");
122
123   lcd.setCursor(0,1);
124   lcd.print("H:");
125   lcd.print(hum,1);
126   lcd.print("% ");
127
128   lcd.setCursor(11,1);
129   lcd.print(ledState ? "ON " : "OFF");
```

A kijelző a mért adatokat és a LED állapotát jeleníti meg, így a rendszer állapota hálózat nélkül is ellenőrizhető.

Összegzés

Ez a program egy autonóm terepi vezérlőegység működését demonstrálja:

- fizikai bemenet feldolgozás
- aktorvezérlés
- szenzoradat-gyűjtés
- helyi vizualizáció

A kód szerkezete jól elkülöníti az egyes funkciókat, ami megkönnyíti a későbbi bővítést és magasabb szintű rendszerekhez való csatlakoztatást.

6.6. Terepi vezérlőcsomópont hálózati bővítéssel – WiFi + Modbus TCP

Ebben a verzióban a korábban bemutatott önálló terepi vezérlő kiegészül hálózati képességekkel.

A helyi működés (gomb → LED, DHT mérés, LCD kijelzés) változatlanul megmarad, viszont az eszköz már képes:

- WiFi hálózatra csatlakozni
- Modbus TCP felügyeleti rendszerként működni
- a fizikai bemenetek és kimenetek állapotát regisztereken keresztül elérhetővé tenni
- külső rendszer (PLC / felügyeleti szoftver) által történő távoli olvasásra és beavatkozásra

Fontos, hogy a logika nem költözik ki a terepi eszközről: a helyi vezérlés továbbra is autonóm, a hálózat csak állapotmegosztásra és távoli beavatkozásra szolgál.

A program felépítése funkcionális egységek szerint

Könyvtárak és hardverdefiníciók

```
#include <WiFi.h>          // ESP32 WiFi stack
#include <ModbusTCP.h>     // Modbus TCP szerver kezelése
#include <DHT.h>           // DHT szenzor magas szintű kezelése (protokollt elrejti)
#include <Wire.h>          // I2C kommunikáció (LCD miatt)
#include <hd44780.h>       // LCD vezérlő
#include <hd44780ioClass/hd44780_I2Cexp.h> // I2C LCD expander kezelés

ModbusTCP mb;  // Az ESP Modbus TCP szerverének létrehozása.


// ================= HARDVER KIOSZTÁS =================
// GPIO-k definiálása hardver absztrakcióként

#define BTN 17         // Fizikai gomb bemenet
#define LED 2          // Fő LED kimenet (vezérelt fogyasztó)
#define WIFI_LED 5     // WiFi státusz visszajelző LED (terepi hibajelzés)
#define DHTPIN 4       // DHT adatvonal
#define DHTTYPE DHT11 // Szenzor típusa

// Szenzor objektum létrehozása
DHT dht(DHTPIN, DHTTYPE);

// I2C LCD objektum létrehozása (automatikus I2C címfelismerés)
hd44780_I2Cexp lcd;
```

Hálózati paraméterezés

```
// ================= WIFI HÁLÓZATI PARAMÉTEREK =================
// Ezek konfigurálják a hálózati kapcsolatot, valamint biztosítják,
// hogy az ESP mindig ugyanazon IP címen legyen elérhető.

const char* ssid = "plc";
const char* pass = "12345678";

IPAddress local_IP(192, 168, 137, 200);
IPAddress gateway(192, 168, 137, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8,8,8,8);
IPAddress secondaryDNS(8,8,4,4);


// ================= WIFI ÚJRACSATLAKOZÁSI LOGIKA =================
// A rendszer nem blokkoló reconnect stratégiát használ.
// Ha a kapcsolat megszakad, a program nem áll meg,
// hanem meghatározott időközönként újrapróbál csatlakozni.

unsigned long lastReconnectAttempt = 0;         // Utolsó reconnect próbálkozás időpontja
const unsigned long reconnectInterval = 300000; // 5 perc újrapróbálkozási ciklus
bool wifiConnected = false;                     // Hálózati állapot logikai jelző

// WiFi csatlakozási rutin
// Ez a függvény bontja az előző kapcsolatot,
// majd új kapcsolatot kezdeményez a megadott SSID-re.
void connectToWiFi() {
  WiFi.disconnect(true);
  delay(100);                // Rövid stabilizációs várakozás
  WiFi.begin(ssid, pass);
}


// ================= WIFI HIBA LED IDŐZÍTÉS =================
// Amennyiben nincs WiFi kapcsolat,
// a WIFI_LED másodperces periódussal villog.
// A megvalósítás millis() alapú, tehát nem blokkolja a fő ciklust.

unsigned long lastBlinkTime = 0;     // Utolsó LED állapotváltás
const unsigned long blinkInterval = 1000; // 1 másodperces villogási periódus
bool wifiLedState = false;           // WiFi LED aktuális állapota
```

Gomb prellegés (debounce) kezelése, LED állapot és mérési változók

```
// ================= BELSŐ ÁLLAPOT RÉTEG =================
// A rendszer egyetlen "igazságforrást" használ a LED állapotára.
// A fizikai gomb, a Modbus és a fizikai LED is ezt az állapotot követi.

bool ledState = false;   // A LED központi logikai állapota


// ================= GOMB PRELL SZŰRÉS =================
// Mechanikus gomb érintkezése záráskor pattog (prell).
// A stabil állapot detektálás millis alapú időszűréssel történik.

bool lastReading = false;       // Utolsó nyers bemeneti érték
bool stableState = false;       // Szűrt, stabil állapot
unsigned long lastChangeTime = 0;
const unsigned long debounceTime = 50;  // 50 ms stabilitási küszöb


// ================= DHT MINTAVÉTELI IDŐZÍTÉS =================
// A DHT szenzor olvasása nem történhet túl gyakran,
// ezért 2 másodperces mintavételezési ciklust alkalmazunk.

unsigned long lastDHTread = 0;
float temp = 0;
float hum = 0;
```

Inicializálás

```
// ===================== SETUP =====================
// =================================================
void setup() {

  Serial.begin(115200);

  // Hardver inicializálás
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(WIFI_LED, OUTPUT);

  // ===== Stabil statikus IP konfiguráció =====
  // Az ESP32 WiFi stack sajátossága miatt:
  // - Station mód explicit beállítása szükséges
  // - DHCP cache törlése
  // - Statikus IP konfigurálása csatlakozás előtt

  WiFi.mode(WIFI_STA);
  WiFi.persistent(false);
  WiFi.disconnect(true);
  delay(100);

  WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS);
  WiFi.begin(ssid, pass);

  // Induláskori maximum 10 másodperces várakozás.
  // Ha ezalatt nem jön létre kapcsolat,
  // a program offline módban folytatódik.
  unsigned long startAttemptTime = millis();
  while (WiFi.status() != WL_CONNECTED &&
         millis() - startAttemptTime < 10000) {
    delay(500);
  }

  wifiConnected = (WiFi.status() == WL_CONNECTED);

  // Szenzor és perifériák inicializálása
  dht.begin();
  Wire.begin(21,22);

  lcd.begin(16,2);
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Smart Farm Node");
  delay(2000);
  lcd.clear();
```

Modbus regiszterkiosztás

```
  // ================ MODBUS REGISZTERKIOSZTÁS ================
  // Coil 0  → LED állapot
  // Ists 0  → Gomb állapot
  // Hreg 0  → Hőmérséklet (x10 skálázva)
  // Hreg 1  → Páratartalom (x10 skálázva)

  mb.server();
  mb.addCoil(0);
  mb.addIsts(0);
  mb.addHreg(0);
  mb.addHreg(1);

  // Coil inicializálása a belső logikai állapotból
  mb.Coil(0, ledState);
}
```

WIFI felügyelet, Modbus kiszolgálás aktív kapcsolat mellett

```
void loop() {

  // ===== WIFI FELÜGYELET =====
  // Nem blokkoló hálózati állapotellenőrzés.
  // Ha nincs kapcsolat, 5 percenként újrapróbál.

  if (WiFi.status() != WL_CONNECTED) {

    wifiConnected = false;

    if (millis() - lastReconnectAttempt >= reconnectInterval) {
      lastReconnectAttempt = millis();
      connectToWiFi();
    }
  }
  else {
    wifiConnected = true;
  }

  // ===== WIFI HIBA LED KEZELÉS =====
  // Offline állapotban 1 Hz villogás.
  // Online állapotban LED kikapcsolva.

  if (!wifiConnected) {

    if (millis() - lastBlinkTime >= blinkInterval) {
      lastBlinkTime = millis();
      wifiLedState = !wifiLedState;
      digitalWrite(WIFI_LED, wifiLedState);
    }
  }
  else {
    digitalWrite(WIFI_LED, LOW);
    wifiLedState = false;
  }

  // ===== MODBUS KISZOLGÁLÁS =====
  // Csak aktív hálózat esetén kezel TCP kéréseket.
  if (wifiConnected) {
    mb.task();
  }

  // ================================================
  // ===== MODBUS → BELSŐ ÁLLAPOT SZINKRON =====
  // ================================================
  // Ha külső rendszer írja a Coil 0-t,
  // a belső állapot frissül.

  bool coilState = mb.Coil(0);

  if (coilState != ledState) {
    ledState = coilState;
  }
```

Gomb olvasása, LED állapot vezérlése és távoli monitorozása

```
218   // ================================================
219   // ===== GOMB → BELSŐ ÁLLAPOT =====
220   // ================================================
221   // Fizikai gombnyomás esetén a belső állapot toggolódik,
222   // majd a Coil regiszter tükrözi azt.
223
224   bool reading = digitalRead(BTN);
225
226   if (reading != lastReading) {
227     lastChangeTime = millis();
228   }
229
230   if ((millis() - lastChangeTime) > debounceTime) {
231
232     if (reading != stableState) {
233       stableState = reading;
234
235       if (stableState == true) {
236         ledState = !ledState;
237         mb.Coil(0, ledState);
238       }
239     }
240   }
241
242   lastReading = reading;
243
244   // ================================================
245   // ===== FIZIKAI LED VEZÉRLÉS =====
246   // ================================================
247   // A fizikai kimenet mindig a belső állapotot követi.
248
249   digitalWrite(LED, ledState);
```

DHT11 periodikus mérése, távoli monitorozása, gombállapot jelentése a Modbuson

```
251   // ================================================
252   // ===== DHT IDŐZÍTETT OLVASÁS =====
253   // ================================================
254
255   if (millis() - lastDHTread > 2000) {
256
257     lastDHTread = millis();
258
259     float h = dht.readHumidity();
260     float t = dht.readTemperature();
261
262     if (!isnan(h) && !isnan(t)) {
263       mb.Hreg(0, (int)(t * 10));
264       mb.Hreg(1, (int)(h * 10));
265       temp = t;
266       hum = h;
267     }
268   }
269
270   // Gomb státusz jelentése Modbuson
271   mb.Ists(0, stableState);
```

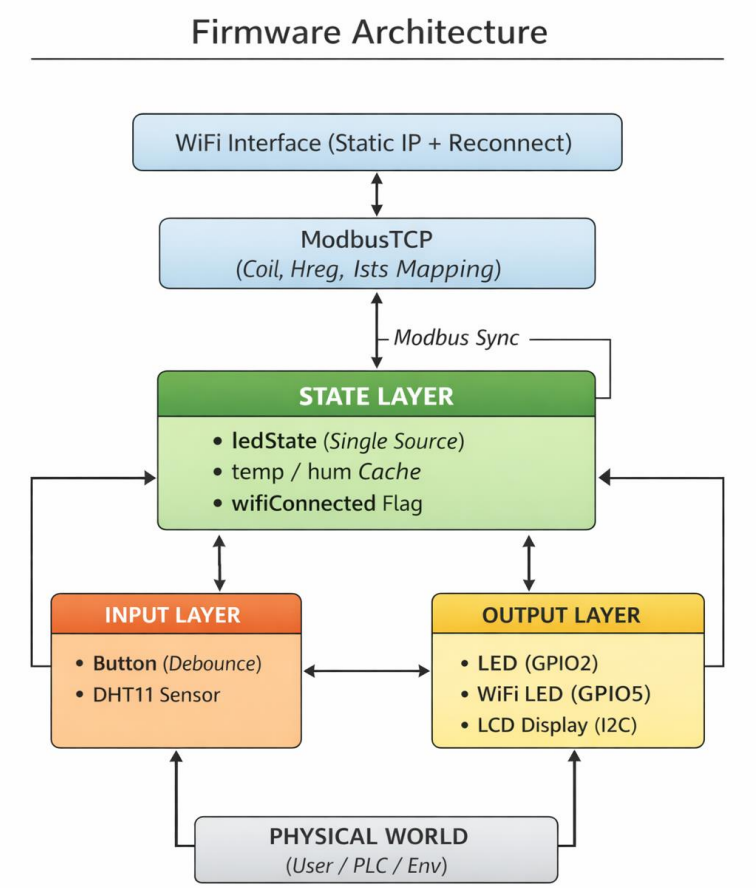
LCD kijelzés

```
273     // ================================================
274     // ===== LCD KIJELZÉS =====
275     // ================================================
276     // Első sor: Hőmérséklet + Hálózati állapot
277     // Második sor: Páratartalom + LED állapot
278
279     lcd.setCursor(0,0);
280     lcd.print("T:");
281     lcd.print(temp,1);
282     lcd.print((char)223);
283     lcd.print("C ");
284     lcd.print(wifiConnected ? "N-ON " : "N-OFF");
285
286     lcd.setCursor(0,1);
287     lcd.print("H:");
288     lcd.print(hum,1);
289     lcd.print("% ");
290     lcd.print(ledState ? "L-ON " : "L-OFF");
291
292     delay(10);  // CPU terhelés csökkentése
293 }
```

A rendszer blokkdiagramja

A blokkdiagram az ESP32-alapú firmware réteges, PLC-szerű architektúráját szemlélteti. A felső szinten a WiFi interfész és a ModbusTCP kommunikáció található, amelyek kizárólag a hálózati adatcserét és a regisztertér kezelését végzik. A rendszer központja a State Layer, amely az összes belső állapotváltozót – például a LED állapotát, a mért hőmérsékletet, páratartalmat és a WiFi státuszt – egyetlen igazságforrásként tárolja.

Az Input Layer a fizikai bemeneteket, míg az Output Layer a kimeneteket kezeli, és mindkettő közvetlenül a belső állapotrétegre támaszkodik. A vezérlés teljes egészében az ESP32-n, azaz terepi szinten marad, így a rendszer hálózattól függetlenül autonóm módon működik. A Modbus és a felügyeleti réteg csupán digitális árnyékként tükrözi a belső állapotot, de nem hordozza a vezérlési logikát. Ez a felépítés biztosítja a determinisztikus, offline működést és az ipari rendszerekre jellemző robusztus architektúrát.



65. ábra: A rendszer blokkdiagramja (Forrás: AI generált)

6.7. Felügyeleti szint – Python alapú HMI / felügyeleti ciklus

A terepi ESP32 csomópont a szenzorok és aktorok állapotát Modbus regisztereken keresztül elérhetővé teszi. A felső szint feladata nem a fizikai vezérlés átvétele, hanem a rendszer digitális leképezése és felügyelete.

Ez a réteg a projektben egy Python alapú, ciklikusan működő alkalmazás, amely funkcióját tekintve egy HMI/SCADA-jellegű felügyeleti rendszer.

Feladatai:

- a terepi adatok ciklikus lekérdezése
- állapotok tárolása és feldolgozása
- távoli beavatkozási lehetőség biztosítása
- webes megjelenítés kiszolgálása

66. ábra: Python alapú HMI (Forrás: Saját szerkesztés)

6.8. A Raspberry Pi alapú felügyeleti rendszer előkészítése és üzembe helyezése

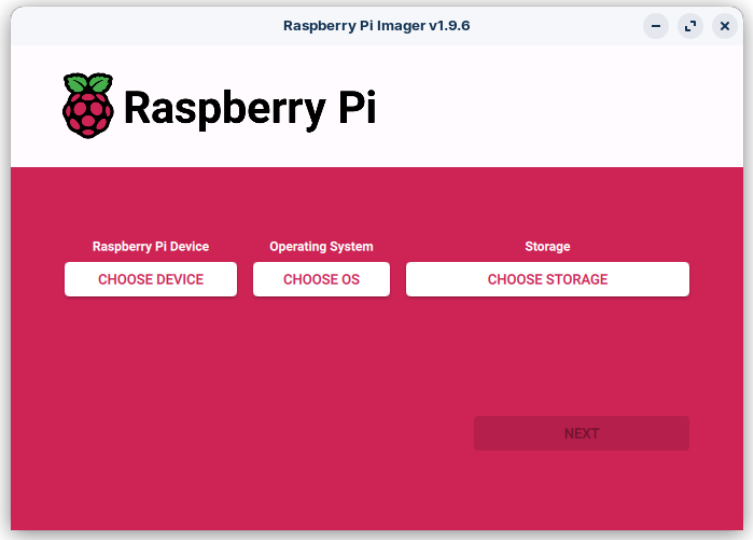
A következő lépcsőfokot a szoftveres felügyeleti logika futtatása jelentette egy önálló, alacsony fogyasztású, ipari környezetben is alkalmazható hardveren. Erre a célra **Raspberry Pi** alapú számítógépet használtunk, amely ideális platformot biztosít a Python alapú, hálózati kommunikációt és webes felületet egyaránt tartalmazó alkalmazások számára. A Raspberry Pi egy jó ár-érték aránnyal rendelkező, viszonylag nagy teljesítményű számítógépet takar, mely rugalmasan illeszthető számos felhasználási területhez. Alapértelmezett esetben egy Debian alapú disztribúciót, a Raspberry OS-t futtat, mely teljesen megfelelt a projekt követelményeinek.

67. ábra: Raspberry Pi (Forrás: https://malnapc.hu/raspberry-pi-5-4gb)

A Raspberry Pi ebben az architektúrában **felügyeleti és vizualizációs rétegként** funkcionál, amely Modbus TCP protokollon keresztül kommunikál az ESP32 alapú alrendszerrel, miközben webes HMI felületet biztosít a felhasználók számára.

Operációs rendszer és alaprendszer telepítése

Az eszköz előkészítése egy stabil, hosszú távon is támogatott Linux alapú operációs rendszer telepítésével kezdődött. Bár lehetőség van egyéni operációs rendszerek telepítésére is, a Raspberry Pi OS biztosította a megfelelő hardvertámogatást, a rendszerstabilitást, valamint a Python fejlesztői környezet egyszerű elérhetőségét. A rendszer telepítését a hivatalos Raspberry Pi Imager programmal végeztük, amely alapvető konfigurációs beállítások megadását követően már egy kész asztali környezetet biztosított a tanulóknak a további munkára.



68. ábra: Raspberry Pi OS telepítése (Forrás: Saját szerkesztés)

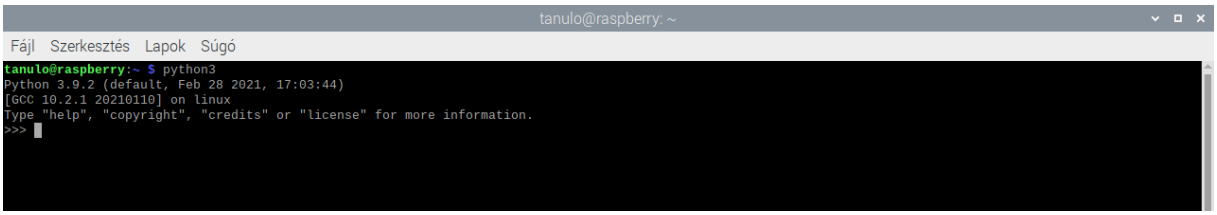
A rendszertelepítés során megtörtént:

- az alapértelmezett felhasználói fiók konfigurálása,
- a rendszer frissítése,
- valamint a hálózati kapcsolat ellenőrzése.

A megbízható hálózati működés kiemelt fontosságú volt, mivel a teljes felügyeleti logika IP-alapú kommunikációra épül.

Python futtatókörnyezet és függőségek előkészítése

Mivel a felügyeleti alkalmazás Python nyelven készült, ezért a Raspberry Pi-n egységes és jól karbantartható Python környezet kialakítása történt meg. A rendszerhez tartozó Python 3



verzió biztosította a Flask keretrendszer és a Modbus kommunikációhoz szükséges könyvtárak kompatibilis futtatását.

A környezet előkészítése során hangsúlyt kapott:

- a Python csomagkezelő (PIP) használata,
- a szükséges külső könyvtárak (webszerver, Modbus kliens) telepítése,
- valamint a verzióütközések elkerülése.

```
tanulo@raspberry:~ $ pip install flask
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: flask in /usr/lib/python3/dist-packages (1.1.2)
tanulo@raspberry:~ $ pip install pymodbus
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pymodbus
  Downloading https://www.piwheels.org/simple/pymodbus/pymodbus-3.8.6-py3-none-any.whl (164 kB)
     |████████████████████████████████| 164 kB 938 kB/s
Installing collected packages: pymodbus
  WARNING: The script pymodbus.simulator is installed in '/home/tanulo/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pymodbus-3.8.6
tanulo@raspberry:~ $
```

Ez a lépés felhívta a tanulók figyelmét a szoftveres rendszerek reprodukálhatóságára és karbantarthatóságára.

Hálózati konfiguráció és Modbus TCP kapcsolat

A rendszer működésének alapfeltétele a Raspberry Pi és az ESP32 közötti stabil Modbus TCP kapcsolat. Ennek érdekében a Raspberry Pi fix IP-címmel, kiszámítható hálózati környezetben működött, így a felügyeleti alkalmazás mindig elérte az alrendszert.

A tanulók megértették, hogy a hálózati kommunikáció megbízhatósága kulcsfontosságú egy felügyeleti rendszer esetében.

A Modbus kapcsolat ebben a kontextusban nem alacsony szintű vezérlésre, hanem állapotlekérdezésre és parancsközvetítésre szolgált, ami jól illeszkedik a szoftveres PLC-ciklus koncepciójához.

Alkalmazás indítása és párhuzamos működés kezelése

A felügyeleti program indításakor külön szálon fut a ciklikus állapotfrissítést végző logika, míg a Flask alapú webkiszolgáló a felhasználói kéréseket kezeli. Ez a megoldás biztosítja, hogy a webes felület kiszolgálása ne blokkolja a rendszer felügyeleti funkcióit.

A Raspberry Pi erőforrásai elegendőnek bizonyultak:

- a folyamatos Modbus kommunikációhoz,
- a ciklikus állapotfeldolgozáshoz,
- valamint a webes HMI kiszolgálásához.

Ez a működési modell jól demonstrálta a tanulók számára a párhuzamos feldolgozás és a nem valós idejű, de determinisztikus működés közötti különbséget.

Oktatási és rendszerintegrációs tapasztalatok

A Raspberry Pi alapú futtatókörnyezet kialakítása szervesen kapcsolódott a projekt korábbi szakaszaihoz. A tanulók egy olyan komplett rendszert láttak működés közben, amelyben az érzékelők és beavatkozók fizikai szinten jelennek meg, a kommunikáció ipari protokollon keresztül történik, valamint a felügyelet és vezérlés pedig szoftveres eszközökkel valósul meg.

Továbbá a Debian alapú operációs rendszer lehetőséget adott a diákok Linux ismereteinek elmélyítésére. Mivel ipari környezetben nem szükségszerűen grafikus felületen történik ezeknek az eszközöknek a konfigurációja, ezért törekedtünk arra, hogy a diákok minél többet dolgozzanak parancssoros felületen.

A gyakorlati megvalósítás során azonban a diákokkal elértük a Raspberry Pi hardveres korlátait. Amikor a teljes tanulói csapat egyszerre kapcsolódott a webszerverekre, az adatok feldolgozásában lassulás volt tapasztalható. Emellett a többletterhelés szükségessé tette volna aktív hűtés felszerelését, mivel a lapka PC több alkalommal is újraindult túlmelegedés miatt. Bár a projekt szükségleteit kielégítette, fontos tanulságként szolgált a megvalósítás során, hogy egy ipari szintű rendszernél kiemelt szempont kell, hogy legyen a kiszolgáló eszköz erőforrásainak előre tervezése és a várható terhelés felmérése.

6.9. A Python kliens szerepe

Az ESP Modbus felügyeleti rendszer, a Python program célja a felügyeleti logika és digitális „árnyék” megvalósítása.

Kapcsolódás az ESP-hez

```python
# ================= FLASK + MODBUS TCP PLC RÉTEG =================
# Ez a program egy "szoftveres PLC ciklust" valósít meg.
# Feladata:
#   • ESP32 Modbus eszköz lekérdezése
#   • LED vezérlési logika kezelése
#   • Szenzoradatok továbbítása a webes HMI felé


# ================= MODULOK IMPORTÁLÁSA =================
from flask import Flask, render_template, jsonify
from pymodbus.client import ModbusTcpClient
import threading, time


# ================= WEB SZERVER OBJEKTUM =================
# Flask kezeli a HTTP kéréseket (HMI felület)
app = Flask(__name__)


# ================= MODBUS KLIENS (ESP32 FELÉ) =================
# Ez a sor létrehozza a TCP kapcsolatot az ESP Modbus felügyeleti rendszerével.
esp = ModbusTcpClient("192.168.0.200", port=502)


# ================= ÁLLAPOTVÁLTOZÓK LÉTREHOZÁSA =================
# Ezek a változók a rendszer állapotát tükrözik a szerver oldalon

led_state = False        # LED logikai állapot (kimenet)
btn_state = False        # Fizikai gomb állapota
prev_btn_state = False   # Előző ciklus gomb állapota (élérzékeléshez)

temperature = 0          # Hőmérséklet cache
humidity = 0             # Páratartalom cache
```

A felügyeleti ciklus szerkezete

```
# ================= PLC CIKLUS (IPARI LOGIKA MINTÁJÁRA) =================
# Ez a függvény a felügyeleti alkalmazás ciklikusan futó központi része.
# Bár a neve „felügyeleti ciklus”, funkcionálisan nem valós idejű vezérlési ciklust, hanem egy állapotfrissítő, felügyeleti lekérdezési ciklust valósít meg.
# Ez egy végtelen ciklus, amely fix periódusban (200 ms) végzi az I/O beolvasást és a vezérlési logikát.

def plc_cycle():
    global led_state, btn_state, prev_btn_state, temperature, humidity

    while True:

        # ===== DIGITÁLIS BEMENET OLVASÁSA (FIZIKAI GOMB) =====
        # Modbus Input Status regiszter 0
        # Az ESP az aktuális gombállapotot a Discrete Input 0 címre írja.
        rr_btn = esp.read_discrete_inputs(address=0, count=1)

        if not rr_btn.isError():
            btn_state = rr_btn.bits[0]


        # ===== ÉLÉRZÉKELÉS (EDGE TRIGGER) =====
        # Ez a PLC logika:
            # ha a gomb éppen most lett lenyomva → LED állapot vált,
            # a parancs visszakerül az ESP-re
        # Ipari PLC logikában ez "RISING EDGE DETECT".
        # Csak akkor reagálunk, amikor a gomb 0→1 állapotba vált

        if btn_state and not prev_btn_state:
            led_state = not led_state          # LED állapot váltás
            esp.write_coil(0, led_state)       # Fizikai LED frissítés Modbuson

        prev_btn_state = btn_state


        # ===== ANALÓG ADATOK OLVASÁSA =====
        # Holding Register 0 → hőmérséklet
        # Holding Register 1 → páratartalom
        # Az ESP tizedes pontossággal küldi az adatot.

        rr_temp = esp.read_holding_registers(address=0, count=2)

        if not rr_temp.isError():
            temperature = rr_temp.registers[0] / 10.0
            humidity = rr_temp.registers[1] / 10.0


        # ===== PLC CIKLUS IDŐZÍTÉS =====
        # 200 ms ciklusidő (ipari PLC mintára)
        time.sleep(0.2)
```

Webes (HMI) megjelenítés

```
# ================= WEB OLDAL =================
# A HMI felület betöltése

@app.route("/")
def index():
    return render_template("index.html")


# ================= ÁLLAPOT API =================
# A webes felület innen kapja az aktuális rendszerállapotot

@app.route("/status")
def status():
    return jsonify({
        "led": led_state,       # LED állapot
        "btn": btn_state,       # Gomb állapot
        "temp": temperature,    # Hőmérséklet
        "hum": humidity         # Páratartalom
    })


# ================= VIRTUÁLIS GOMB (WEB HMI) =================
# Ez a fizikai gomb funkcióját utánozza a weboldalon keresztül

@app.route("/toggle_led", methods=["POST"])
def toggle_led():
    global led_state

    led_state = not led_state          # LED állapot váltás
    esp.write_coil(0, led_state)       # Parancs küldése az ESP-nek

    return "OK"


# ================= PROGRAM INDÍTÁS =================
if __name__ == "__main__":

    # PLC ciklus külön szálon fut, hogy a web szerver ne blokkolódjon
    threading.Thread(target=plc_cycle, daemon=True).start()

    # Flask webszerver indítása
    app.run(host="0.0.0.0", port=5000)
```

6.10. Szenzoradatok naplózása SQL adatbázisba

A felügyeleti rendszer továbbfejlesztésének következő lépéseként célul tűztük ki, hogy a szenzoradatok ne csupán valós időben jelenjenek meg a webes HMI felületen, hanem strukturált módon, visszakereshető formában is rögzítésre kerüljenek. Ennek megvalósítására egy SQL alapú adatbázis-integráció készült a meglévő Python alkalmazás kiegészítésével.

Ez a fejlesztés új dimenziót adott a projektnek, mivel a rendszer így már nemcsak felügyeleti, hanem adatgyűjtési funkciókat is ellát, lehetővé téve azok későbbi elemzését.

Az adatnaplózás szerepe a felügyeleti rendszerben

A szenzoradatok adatbázisba történő mentése lehetővé teszi az időbeli változások nyomon követését, a működés hosszabb távú elemzését, valamint az adatok későbbi feldolgozását. A tanulók számára ez jól szemléltette, hogy a valós rendszerekben a pillanatnyi állapot megjelenítése önmagában nem elegendő; az adatok historikus tárolása alapvető igény.

A fejlesztés során hangsúlyt kapott annak megértése, hogy az adatgyűjtés nem folyamatos vezérlési célokat szolgál, hanem döntéstámogató és elemzési alapot teremt.

Adatbázis-választás és logikai felépítés

Az oktatási környezethez illeszkedve egy egyszerű, SQL alapú adatbázis-struktúra került kialakításra, amely könnyen kezelhető, mégis jól szemlélteti a relációs adatkezelés alapelveit. Az adatbázis egyetlen, jól definiált táblában rögzíti a mért adatokat. Az adatbázishoz MariaDB (MySQL alapú) adatbázismotort használtunk.

A tárolt adatok tipikusan (**sensordata** tábla):

- rögzítés azonosítója (**dataid**, int, auto-incerement, primary key)
- időbélyeg (**time_of_measurement**, datetime, default current_timestamp)
- hőmérséklet-értékek (**temp**, double),
- páratartalom-adatok (**hum**, double),

Ez a struktúra lehetővé teszi az adatok időrend szerinti visszakeresését és egyszerű statisztikai feldolgozását.

A Python alkalmazás kiegészítése adatbázis-kezeléssel

A meglévő Flask alapú alkalmazás bővítése során az adatbázis-kezelés egy elkülönített logikai egységként került beépítésre. A szenzoradatok kiolvasása továbbra is a Modbus TCP kommunikáción keresztül, ciklikusan történik, azonban a frissített értékek bekerülnek az adatbázisba is.

Bár a jelenlegi implementációban az adatok beküldése folyamatos az adatbázisba, a tanulók figyelmét felhívtuk, hogy egy komplex rendszerben különböző funkciók eltérő időkritikussággal működnek. Ebből kiindulva az alkalmazás fejlesztésének következő lépcsőfoka lehet az adatok naplózásának ritkítása, hogy az csak néhány másodpercenként történjen meg.

A Python program bővítéséhez először is szükség volt a mysql.connector modul importálására.

```
9   # ================= MODULOK IMPORTÁLÁSA =================
10  from flask import Flask, render_template, jsonify
11  from pymodbus.client import ModbusTcpClient
12  import threading, time
13  import mysql.connector #Az SQL kapcsolatot kezelő modul importálása.
```

A következő lépésben létrehoztuk az adatbázis kapcsolat kezeléséhez és az adatok beküldéséhez szükséges objektumokat.

```
# ================= ADATBÁZIS KAPCSOLAT KONFIGURÁLÁSA =================
# Az adatbázis kapcsolatot kezelő objektum létrehozása.
db = mysql.connector.connect(
    host="localhost",
    port="3306",
    user="root",
    password="",
    database="sensordatabase"
)

#Az adstbázist kezelő cursor objektum létrehozása.
dbcursor = db.cursor()
```

Végül pedig az analóg szenzoradatok lekérdezésénél bővítettük a kódot az adatok adatbázis szerverbe küldésével:

```
        # ===== ANALÓG ADATOK OLVASÁSA =====
        # Holding Register 0 → hőmérséklet
        # Holding Register 1 → páratartalom
        # Az ESP tizedes pontossággal küldi az adatot.

        rr_temp = esp.read_holding_registers(address=0, count=2)

        if not rr_temp.isError():
            temperature = rr_temp.registers[0] / 10.0
            humidity = rr_temp.registers[1] / 10.0
            dbcursor.execute("INSERT INTO sensordata (temp,hum) VALUES (" + temperature + ","+humidity+")")


        # ===== PLC CIKLUS IDŐZÍTÉS =====
        # 200 ms ciklusidő (ipari PLC mintára)
        time.sleep(0.2)
```

Oktatási jelentőség és tanulói tapasztalatok

Az SQL adatbázis-integráció bevezetése jelentősen bővítette a projekt tanulási értékét. A tanulók nemcsak a valós idejű adatkezeléssel, hanem az adatok strukturált tárolásával és későbbi felhasználhatóságával is megismerkedtek.

A fejlesztés során fejlődtek:

- az adatkezelési és adatmodellezési alapismeretek,
- a rendszerszintű gondolkodás,
- valamint az adatbiztonság és adatkonzisztencia iránti érzékenység.

A szenzoradatok adatbázisba történő rögzítése így szervesen illeszkedett a projekt egészéhez, és tovább erősítette azt a szemléletet, hogy a modern informatikai és ipari rendszerek alapja az adatok tudatos és strukturált kezelése.

6.11. Webes felügyeleti réteg – szerepe a rendszerben

A Python alkalmazás biztosítja az adatok elérését HTTP végpontokon keresztül, míg a böngészőben futó HTML + JavaScript felület gondoskodik a megjelenítésről és a felhasználói beavatkozásról.

Ez a réteg:

- nem vezérli közvetlenül a fizikai I/O-t
- az ESP állapotának digitális árnyékát jeleníti meg
- felhasználói parancsokat továbbít a Python alkalmazás felé

Az adatút:

LED állapot blokk

Ez a gomb nem közvetlenül az ESP-t vezérli, hanem a Python felügyeleti rendszert.

```
<!-- ====================== LED KÁRTYA ====================== -->
<!-- Ez a blokk a LED állapotát mutatja és innen lehet vezérelni -->
<div class="card">
    <h2>LED állapot</h2>

    <!-- A LED vizuális visszajelzője
         A színét a JavaScript módosítja:
         zöld = világít
         piros = nem világít -->
    <div id="led" class="led"></div>

    <!-- Virtuális kapcsoló gomb
         onclick esemény → toggleLED() JS függvény fut le
         Ez HTTP POST kérést küld a Flask szervernek (/toggle_led)
         Ez a gomb nem közvetlenül az ESP-t vezérli, hanem a Python felügyeleti rendszert.-->
    <button onclick="toggleLED()">LED kapcsolás</button>
</div>
```

Gomb állapot blokk

```
<!-- ====================== FIZIKAI GOMB ÁLLAPOT ====================== -->
<!-- Ez CSAK kijelzés, nem vezérlő szerv -->
<div class="card">
    <h2>Gomb állapot</h2>

    <!-- A fizikai gomb állapotának szöveges kijelzése.
         A Flask szerver PLC ciklusa frissíti.
         "Nyomva" / "Nincs nyomva" szöveg jelenik meg -->
    <div id="btn">Nincs nyomva</div>
</div>
```

A fizikai gomb állapotának szöveges kijelzése.

Szenzor blokk

```
<!-- ====================== KÖRNYEZETI ADATOK ====================== -->
<!-- DHT11 szenzor által mért adatok -->
<div class="card">
    <h2>Környezeti adatok</h2>

    <!-- Hőmérséklet kijelzés
         Ezek a DHT szenzor értékei számára fenntartott mezők.
         A program ide írja be a /status API-ból érkező értéket. -->
    <div>🌡 Hőmérséklet: <span id="temp">--</span> °C</div>

    <!-- Páratartalom kijelzés -->
    <div>💧 Páratartalom: <span id="hum">--</span> %</div>
</div>
```

Ezek a DHT szenzor értékei számára fenntartott mezők.

JavaScript – élő frissítés

```
// ======================== ÁLLAPOTFRISSÍTŐ FÜGGVÉNY ========================
// Ez a függvény 200 ms-onként lefut.
// Feladata: lekérdezni a Flask szervertől az aktuális rendszerállapotot és frissíteni a weboldalon látható adatokat.

function update(){

  // HTTP GET kérés a szerver /status végpontjára
  // A szerver JSON formátumban küldi vissza az adatokat:
  // { led: bool, btn: bool, temp: float, hum: float }
  fetch("/status")

    // A válasz átalakítása JSON objektummá
    .then(r => r.json())

    // A kapott adatok feldolgozása
    .then(data => {

      // ===== LED VIZUÁLIS ÁLLAPOT =====
      // A LED kör színének beállítása:
      // zöld ha világít (true), sötétpiros ha nem (false)
      document.getElementById("led").style.background =
        data.led ? "limegreen" : "darkred";


      // ===== FIZIKAI GOMB ÁLLAPOT KIÍRÁS =====
      // A Flask PLC ciklus által olvasott bemenet jelenik meg
      document.getElementById("btn").innerText =
        data.btn ? "Nyomva" : "Nincs nyomva";


      // ===== HŐMÉRSÉKLET KIÍRÁS =====
      // DHT11 szenzor adata az ESP → Modbus → Flask útvonalon
      document.getElementById("temp").innerText = data.temp;


      // ===== PÁRATARTALOM KIÍRÁS =====
      document.getElementById("hum").innerText = data.hum;
    });
}
```

A JavaScript ezeket a változókat kapja vissza, frissíti.

LED kapcsoló gomb működése

```
// ======================== VIRTUÁLIS LED KAPCSOLÓ ========================
// Ez a függvény akkor fut le, amikor a felhasználó megnyomja a weboldalon a "LED kapcsolás" gombot.

function toggleLED(){

  // HTTP POST kérés küldése a szervernek
  // A Flask oldalon ez a /toggle_led útvonalat hívja meg, ami megfordítja a LED állapotát és kiírja Modbus-on az ESP-re.
  fetch("/toggle_led", { method: "POST" });

  // Fontos: itt nincs válaszfeldolgozás, a következő update() ciklus már lekéri az új állapotot.
}
```

Folyamat:

Frissítési ciklus

```
// ======================== IDŐZÍTETT FRISSÍTÉS ========================
// A böngésző 200 ms-onként automatikusan meghívja az update() függvényt.
// Ez kvázi PLC ciklus a weboldalon (HMI polling).
// Ez megfelel a Python programban lévő ciklus ütemének.

setInterval(update, 200);
```

Ez megfelel a Python ciklus ütemének, így a rendszer szinkron marad.

A stílus szerepe a rendszerben

A CSS biztosítja, hogy a HMI webes felület:

- jól olvasható legyen gyenge fényviszonyok között is
- egyértelmű vizuális visszajelzést adjon az állapotokról
- elkülönítse az információs blokkokat (HMI panelek logikája)

Ez a struktúra SCADA / ipari HMI (Human–Machine Interface)ek minimalista szemléletét követi.

Teljes CSS – részletesen kommentelve

```css
/* ========================= OLDAL ALAPSTÍLUS ========================= */
body {
    font-family: Arial;
    background: linear-gradient(135deg,#0f2027,#203a43,#2c5364);
    color: white;
    text-align: center;
    margin: 0;
}

/* ========================= KÁRTYA (PANEL) STÍLUS =========================
   Minden információs blokk ezt használja:
   LED, Gomb, Szenzor adatok*/
.card {
    background: #222;
    margin: 20px auto;
    padding: 20px;
    width: 320px;
    border-radius: 12px;
    box-shadow: 0 0 10px #000;
}

/* ========================= LED VISSZAJELZŐ KÖR =========================
   Ez a vizuális LED indikátor*/
.led {
    width: 40px;
    height: 40px;
    margin: 15px auto;
    border-radius: 50%;
    background: darkred; /* Alapértelmezett állapot: LED kikapcsolva. A program dinamikusan felülírja:
    limegreen = bekapcsolva */
}


/* ========================= GOMB STÍLUS =========================
   Virtuális LED kapcsoló*/
button {
    padding: 10px 15px;
    border-radius: 8px;
    border: none;
    background: #00aaff;
    color: white;
    cursor: pointer;
}
```

Ez megfelel a valódi PLC HMI, SCADA terminál és ipari kezelőpanel megjelenítési elveinek.

Miért fontos ez oktatásban?

A tanulók nem csak adatot jelenítenek meg hanem megtanulják, hogy egy HMI (Human–Machine Interface):

- funkcionális vizuális rendszer
- ahol a szín, méret, elrendezés jelentéssel bír

A webes réteg ezzel teljes:

6.12. A webes HMI felület tanulói továbbfejlesztése csapatmunkában

A felügyeleti rendszer alapfunkcióinak megvalósítását követően a projekt következő eleme a webes HMI (Human–Machine Interface) felület továbbfejlesztése volt. Ebben a szakaszban a tanulók már nem előre meghatározott megoldásokat valósítottak meg, hanem saját elképzeléseik mentén, csapatmunkában alakították tovább a meglévő HTML és CSS alapú felületet.

A feladat célja az volt, hogy a tanulók megtapasztalják, miként lehet egy működő, de alapvető megjelenésű felületet felhasználóbarátabbá, áttekinthetőbbé és vizuálisan is informatívabbá tenni, miközben megmarad a rendszer funkcionális stabilitása.

Csapatmunka és feladatmegosztás

A tanulók kisebb munkacsoportokban dolgoztak, amelyekben a feladatok megosztása tudatosan történt. Egyes tanulók elsősorban a struktúra átalakítására (HTML elemek rendezése, logikai csoportosítás), míg mások a megjelenés finomítására (színek, elrendezés, vizuális visszajelzések) koncentráltak.

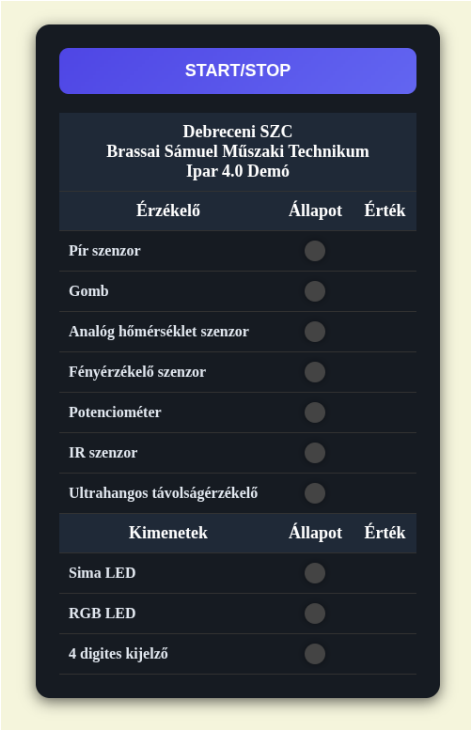
Ez a munkaszervezési forma lehetővé tette, hogy a tanulók megtapasztalják a közös fejlesztés előnyeit és kihívásait, valamint gyakorolják az egyeztetést és az egymás munkájára épülő fejlesztést.

A felület funkcionális és vizuális továbbfejlesztése

A kiindulási alapként szolgáló webes felület már tartalmazta a rendszer alapállapotainak megjelenítését és az alapvető vezérlési lehetőségeket. A tanulók ezt a felületet továbbfejlesztve:

- átrendezték az információk megjelenítését,
- bővítették a megjelenített szenzoradatok tárházát, előkészítve egy későbbi továbbfejlesztést,
- egyértelműbb vizuális visszajelzéseket alakítottak ki,
- valamint a megjelenési stílust csapatuk igényei szerint módosították.

A módosítások során kiemelt figyelmet kapott az áttekinthetőség és a használhatóság, különösen annak érdekében, hogy a felület valós felügyeleti környezetben is könnyen értelmezhető legyen.



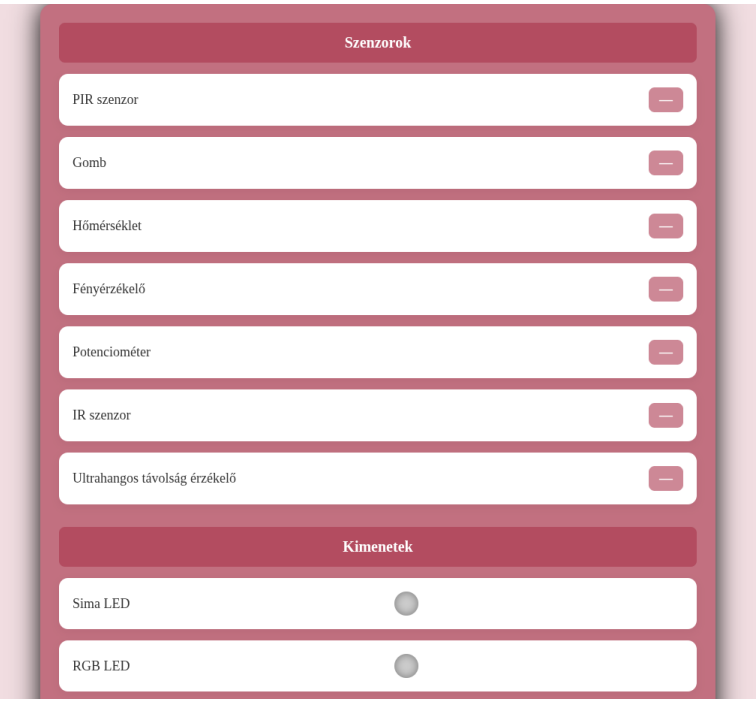
69. ábra: Tanulók továbbfejlesztett munkája 1. (Forrás: Saját szerkesztés)

Egyéni igények és kreatív megoldások megjelenése

A fejlesztési folyamat során a tanulók saját elképzeléseik szerint alakíthatták a felület megjelenését és működését. Ennek eredményeként különböző megoldások születtek, például:

- eltérő színkódolás az állapotjelzésekhez,
- alternatív elrendezések a szenzoradatok megjelenítésére,
- hangsúlyosabb visszajelzések a beavatkozások során.

Ez a szabadság lehetőséget adott a kreativitás kibontakoztatására, miközben a tanulók megtanulták, hogy az egyéni ötleteket össze kell hangolni a rendszer műszaki és funkcionális korlátaival.



70. ábra: Tanulók továbbfejlesztett munkája 2. (Forrás: Saját szerkesztés)

Oktatási jelentőség és fejlesztési tapasztalatok

A webes HMI felület továbbfejlesztése során a tanulók komplex módon alkalmazták korábban megszerzett ismereteiket. A feladat egyszerre fejlesztette:

- a digitális és informatikai kompetenciákat,
- az együttműködési és kommunikációs készségeket,
- valamint a felhasználóközpontú gondolkodást.

A folyamat jól illeszkedett a projekt célkitűzéseihez, mivel a tanulók nem csupán technikai megoldásokat hoztak létre, hanem egy működő rendszer gondolkodtak a felületről, figyelembe véve annak szerepét a teljes felügyeleti architektúrában.

6.13. Összegzés – rendszerintegrációs és tanulási eredmények

Ebben a fejezetben bemutatott tevékenységek a projekt egyik legkomplexebb és pedagógiailag legértékesebb szakaszát alkották. Ebben a fázisban a tanulók egy teljes, működő felügyeleti rendszert valósítottak meg, amelyben a hardveres elemek, a hálózati kommunikáció, a szoftveres vezérlési logika és a webes megjelenítés egységes rendszerként működött.

A Raspberry Pi alapú futtatókörnyezet előkészítése és a Flask alapú alkalmazás üzembe helyezése lehetőséget adott arra, hogy a tanulók valós informatikai és ipari jellegű környezetben alkalmazzák programozási és rendszerüzemeltetési ismereteiket. A Modbus TCP kommunikáció használata különösen fontos tapasztalatot jelentett, mivel a tanulók egy

iparban elterjedt protokollal dolgozhattak, és megértették annak szerepét egy felügyeleti architektúrában.

A webes HMI felület továbbfejlesztése során a tanulók nem csupán technikai megoldásokat készítettek, hanem felhasználóközpontú szemléletet is elsajátítottak. A csapatmunkában végzett fejlesztések során megjelent az egyéni kreativitás, az önálló döntéshozatal, valamint a közös szakmai egyeztetés gyakorlata. A felületek testreszabása hozzájárult ahhoz, hogy a tanulók sajátjuknak érezzék a rendszert, és felelősséget vállaljanak annak működéséért.

Pedagógiai szempontból kiemelendő, hogy a tanulók a teljes fejlesztési folyamat során folyamatos visszacsatolást kaptak döntéseik következményeiről. A rendszer működésének tesztelése, a hibák feltárása és javítása, valamint a megoldások közös értékelése erősítette az önreflexiót és a problémamegoldó gondolkodást. A hibák kezelése nem kudarcként, hanem a tanulási folyamat természetes részeként jelent meg.

A 6. fejezetben megvalósított tevékenységek hozzájárultak több kulcskompetencia fejlődéséhez, különösen:

- a digitális kompetencia,
- az együttműködési és kommunikációs készségek,
- az algoritmikus és rendszerszintű gondolkodás,
- valamint az önálló tanulás és felelősségvállalás területén.

Összességében ez a fejezet jól példázza, hogy a projektalapú, gyakorlatorientált megközelítés miként képes összekapcsolni az elméleti ismereteket a valós alkalmazási környezettel. A tanulók nem csupán egy működő technikai rendszert hoztak létre, hanem olyan tapasztalatokat szereztek, amelyek hosszú távon is megalapozzák szakmai fejlődésüket és pályaorientációjukat.

7. Összefoglalás

A dokumentum egy komplex, Ipar 4.0 és IoT-alapú oktatási projektet mutat be, amelynek középpontjában egy okosház/okosfarm modell fejlesztése áll. A projekt filozófiája szerint a tanulás nem passzív befogadás, hanem aktív alkotási folyamat, ahol a diákok valós problémákon keresztül jutnak el a megoldásokig. A hangsúly nem a kész válaszokon, hanem a gondolkodási folyamaton, a döntéseken és az iteráción van. A tanulók nem egyszerűen használják a technológiát, hanem tervezik, elemzik és fejlesztik azt, miközben megtapasztalják, hogy egy rendszer működése mindig kompromisszumok és tudatos választások eredménye.

Az oktatási célok ennek megfelelően túlmutatnak a klasszikus műszaki ismereteken. A projekt egyszerre fejleszti a programozási és elektronikai tudást, a rendszerszintű gondolkodást és az együttműködési készségeket. A tanulók megtanulják értelmezni a fizikai világ jelenségeit digitális rendszerekben, felismerik az ok-okozati kapcsolatokat, és képesek lesznek komplex problémákat több lépésben kezelni. Különösen fontos, hogy a projekt lehetőséget ad eltérő előképzettségű tanulók bevonására is, ami erősíti a magyarázó gondolkodást és a közös problémamegoldást.

Műszaki szempontból a projekt célja egy olyan IoT-alapú rendszer létrehozása, amelyben szenzorok, vezérlőegységek és beavatkozók egységes logikai rendszerként működnek együtt. A kiindulópont egy gyári Smart Farm Kit, amely azonban csak alapként szolgál: a tanulók feladata nem annak reprodukálása, hanem a rendszer megértése, majd továbbfejlesztése egy saját, ipari szemléletű architektúra irányába. Ez a folyamat jól modellezi a valós mérnöki munkát, ahol a meglévő megoldások elemzése után új, optimalizált rendszerek jönnek létre.

A megvalósítás egy világosan strukturált, mégis rugalmas folyamat mentén történik, amelynek során jól azonosíthatók bizonyos kulcsfontosságú mérföldkövek. A tanulási út nem lineáris, de tipikusan az alábbi lépések mentén halad:

- a gyári rendszer (Smart Farm Kit) megismerése és összeszerelése,
- az egyes komponensek működésének elemzése (szenzorok, aktuátorok, vezérlés),
- a rendszer korlátainak és hiányosságainak feltárása,
- saját fejlesztési célok és funkciók meghatározása,
- egyedi mechanikai elemek és házrészek tervezése (pl. 3D modellezés),
- fizikai elemek gyártása (pl. 3D nyomtatás),
- elektronikai rendszer újratervezése és bővítése,
- programozás (blokkalapú → szöveges),
- alrendszerek integrációja egy egységes rendszerbe,
- tesztelés, hibakeresés és iteratív fejlesztés.

Ez a folyamat nemcsak technikai fejlődést eredményez, hanem megtanítja a tanulókat arra, hogy egy komplex rendszer soha nem „kész”, hanem folyamatosan alakul és finomodik.

71. ábra: Okosház (Forrás: Saját kép, AI generált háttér)

A projekt során jelentkező nehézségek – az alapfogalmak hiánya, az analóg és digitális jelek közötti különbségek megértése, illetve a hibakeresés – a folyamat végére értékes tanulási tapasztalatokká váltak. A műszaki problémák, mint a zajos mérési adatok vagy a nem megfelelő határértékekből adódó instabil működés, segítették a tanulókat abban, hogy felismerjék: a rendszer viselkedése közvetlenül a tervezési döntések következménye.

A projekt végére a tanulók képessé váltak rendszerekben gondolkodni, értelmezni a bemenet–feldolgozás–kimenet logikát, valamint megérteni a fizikai és digitális világ kapcsolatát. A létrehozott megoldások már tudatosan felépített, működő rendszerek voltak.

A legfontosabb tanulság, hogy a projekt nem csupán technikai ismereteket adott, hanem olyan szemléletet alakított ki, amely a modern ipari és technológiai környezetben közvetlenül alkalmazható.

Az Európai Unió finanszírozásával. Az itt szereplő információk és állítások a szerző(k) álláspontját képviselik, és nem feltétlenül tükrözik az Európai Unió vagy a Tempus Közalapítvány hivatalos véleményét. Sem az Európai Unió, sem a támogatást nyújtó hatóság nem vonható felelősségre miattuk.