Automatizált melegház monitorozás és beavatkozó szellőztető rendszer

A Sapientia Erdélyi Magyar Tudomány Egyetem Marosvásárhelyi Karának a 2023/2024 - es tanév első félévében a *Mikrovezérlős rendszerek* tantárgy keretében létrehozott projekt dokumentációja

Diák:

Vasi András, Sapientia EMTE Marosvásárhelyi kar, Számítástechnika szak IV. Év

vasi.andras@student.ms.sapientia.ro

Vezető:

Dr. Losonczi Lajos, Sapientia EMTE Marosvásárhelyi kar, Villamosmérnöki tanszék

lajos@lamda.ro

Tartalomjegyzék

Bevezető

Ez a dokumentáció az Automatizált melegház monitorozás és beavatkozó szellőztető rendszer és beavatkozó rendszer című projekt részeként készült, melynek fókusza az ESP32 mikrovezérlő architektúrájának megismerése és alkalmazása. A projekt célja, hogy átfogó betekintést nyújtson az ESP32 platformba, és bemutassa annak alkalmazási lehetőségeit a gyakorlatban és egy konkrét felhasználási eset bemutatása.

Projekt és project dokumentáció áttekintés

Az ESP32, mint rendkívül sokoldalú mikrovezérlő egység, egyre növekvő népszerűsége miatt nyújt izgalmas lehetőségeket az elektronikai fejlesztők és a beágyazott rendszerek területén.

1. Felhasználói platform (ESP32) bemutatása

- Az ESP32 általános bemutatása, főbb jellemzői és alkalmazási területei.

2. Tervdokumentáció

- Alkatrészlista: Az összes használt alkatrész és eszköz listaformában, beleértve a szállítók nevét, cikkszámát és egyéb releváns információkat.
- Kapcsolási rajzok: ábrák és leírások, melyek bemutatják az ESP32 típusát, lábkiosztását, a projekt analóg is digitális elektronikai összetevőit, az alkatrész listát, valamit a kommunikációt ezek között

3. Megvalósítás/működés leírása, következtetések

- Részletes leírás a projekt megvalósításáról és működéséről, beleértve az elért eredményeket és a felmerült kihívásokat.
- Következtetések: Összegzés a projekt során szerzett tapasztalatokról, tanulságokról és esetleges továbbfejlesztési lehetőségekről.

4. Beágyazott vezérlő programok rövid leírása

- Rövid összefoglaló a beágyazott vezérlő programokról, azok működési elvéről és funkcióiról.

5. Felhasználói interfész programok rövid leírása

- Rövid áttekintés a felhasználói interfész programokról, azok felépítéséről és működéséről.

6. Programok forráskódja, dokumentációja

- A projektben felhasznált programok forráskódjának dokumentációval együtt történő bemutatása

Célkitűzés

Ez a dokumentáció nem csak a konkrét ESP32 projekt bemutatására összpontosít, hanem arra is, hogy segítse az olvasót az ESP32 platform általánosabb megértésében és annak alkalmazási lehetőségeinek felismerésében egy konkrét példa projekten keresztül.

Probléma bemutatása

A továbbiakban bemutatom egy melegházba (üvegház) tervezett automatizált szellőztető (hőmérséklet és páratartalom szabályzó) berendezés elméleti hátterét. A jelen dokumentációban a paradicsomra specifikus adatokkal dolgozom.

A paradicsom optimális termeléséhez melegházban ideális hőmérsékleti tartomány szükséges. Általában a paradicsom számára optimális hőmérsékleti tartomány melegházban a következőképpen néz ki:

1. Éjszakai hőmérséklet: Ideálisan 15-18 Celsius-fok közötti éjszakai hőmérsékletre van szükség a paradicsomnak a virágzás és a termésképzés támogatásához. Az alacsonyabb éjszakai hőmérséklet túl hideg lehet, és gátolhatja a virágzást és a gyümölcsképződést.
2. Nappali hőmérséklet: A paradicsomnak napközbeni hőmérsékleten általában 24-29 Celsius-fok között van a legjobb termelőképessége. A melegebb hőmérséklet támogathatja a növekedést, de túl meleg környezetben a paradicsom növény stressz alá kerülhet.
3. Hőmérsékleti különbség: A nappali és éjszakai hőmérséklet közötti különbség, amit *napi hőingadozásnak* neveznek, fontos a paradicsom termelése szempontjából. Ideálisan a napi hőingadozás 10-15 Celsius-fok között legyen.

A paradicsom optimális páratartalma melegházban a termelés szempontjából a következő tartományban lehet:

1. Virágzás és beporzás: A virágzás és a beporzás során a magas páratartalom előnyös lehet, mivel segíti a virágok megfelelő megporzását. Ideálisan a páratartalom 70-80% között lehet ebben az időszakban.
2. Gyümölcstermékenyítés: A gyümölcstermékenyítés során a páratartalom csökkenthető, mivel a magas páratartalom a beporzást követően a túlzott nedvességet okozhatja a növények között. Ebben az időszakban a páratartalom általában 60-70% között lehet optimális.
3. Betakarítás előtti időszak: A paradicsom betakarítása előtt, amikor a gyümölcsök érnek, a páratartalom csökkentése segíthet a gombás megbetegedések és a gyümölcsök megrepedésének elkerülésében. Ilyenkor a páratartalom általában 50-60% között lehet ideális.

Fontos megjegyezni, hogy a paradicsomfajták és a termesztési körülmények (például a melegházban használt fűtési és hűtési rendszerek) befolyásolhatják az optimális hőmérsékleti és páratartalombéli tartományt. A fajtától függően vannak kisebb eltérések, és az időszakos gondozás is befolyásolhatja a hőmérsékletigényt. Az optimális hőmérsékleti körülmények megtartása segít a paradicsom hatékony termelésében a melegházban.

A huzat vagy szellőzés egy meglehetősen fontos tényező lehet a növények, például a paradicsom növekedése szempontjából, különösen, ha melegágyban vagy melegteremben termesztik. Megfelelő szellőzés segít a levegő áramlásában, páratartalom szabályozásában és hőeloszlásban, és ezáltal hozzájárul a növények egészségéhez és jó terméshozamhoz.

A melegágyban vagy melegházban a páratartalom és a hőmérséklet általában változhat a magasság függvényében. A melegágyakban a hőmérséklet gyakran magasabban van a magasság növekedésével. Az alsó részek közelebb vannak a talajhoz, ami hőmérsékleti különbségeket okozhat a magasabb részekhez képest. A melegágyakban a páratartalom is változhat a magasság növekedésével. Az alsó részeken a páratartalom magasabb lehet, mivel a talajnedvesség és a növényekből párolgó víz hozzájárulhat hozzá. A magasabb részeken a páratartalom általában alacsonyabb lehet, mivel a levegő szárazabb.

A szellőzés és a légmozgás is befolyásolhatja a hőmérséklet és a páratartalom változásait a melegágyban. Például a jó szellőzés segíthet a hőmérséklet szabályozásában és a páratartalom egyenletesebb eloszlásában.

Látható, hogy az ideális termelési mutatató létrehozásához nagyon sok paramétert megfelelő értékek között kell tartani. A kiegyensúlyozott hőmérséklet és páratartalom segíthet a növények egészségének megőrzésében és a jó terméshozam elérésében. Az automatikus vezérlőrendszerek és a szenzorok segíthetnek a környezeti paraméterek pontos monitorozásában és szabályozásában.

Felhasználói platform (ESP32) bemutatása

Általános tudnivalók

Az ESP32 egy rendkívül sokoldalú, integrált rendszerchip (SoC - System on Chip), amely az Espressif Systems által fejlesztett és gyártott. Ez a chip a 2,4 GHz-es Wi-Fi és Bluetooth LE (Low Energy) kapcsolódást támogatja, és két maggal rendelkezik, melyeket a Tensilica LX6 processzorok alkotnak. Ezek a processzorok gyorsaságot és hatékonyságot biztosítanak a számítási műveletek során.

Az ESP32 a rendkívül alacsony energiafogyasztás mellett rendkívüli teljesítményt és sokoldalúságot kínál. Ez a kombináció ideálissá teszi az ESP32-t olyan területeken, ahol a hosszú akkumulátor-élettartam és a megbízható vezeték nélküli kapcsolat elengedhetetlen, például az IoT eszközök, okos otthon rendszerek és szenzorhálózatok terén.

Az ESP32 az Espressif Systems által biztosított átfogó SDK-val (Software Development Kit - szoftverfejlesztői eszközkészlet) és támogatott fejlesztői eszközökkel rendelkezik, amelyek lehetővé teszik a fejlesztők számára, hogy könnyen és hatékonyan programozhassák és alkalmazhassák a chipet az egyedi projektekben.

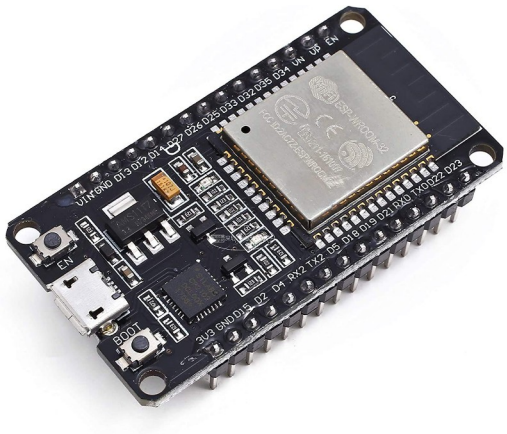
Technikailag az ESP32 számos perifériát és funkciót kínál, mint például analóg és digitális be- és kimenetek, PWM (Pulse Width Modulation - impulzusszélesség moduláció), I2C, SPI, UART kommunikációs interfészek, analóg digitális átalakítókat (ADC), és más hardveres funkciókat, amelyeket a fejlesztők programozhatnak és használhatnak a projektek során.

Az ESP32 rendelkezik beépített biztonsági funkciókkal is, mint például különböző titkosítási és hitelesítési mechanizmusok, ami fontos az adatbiztonság szempontjából.

Összességében az ESP32 széles körű alkalmazásokban használható, mivel kiváló teljesítményt, alacsony energiafogyasztást és sokoldalúságot kínál, mindezt támogatott fejlesztői környezet mellett.

Projekt specifikus tudnivalók

A projektem magvalósításában egy ESP32 WROOM típusú általános célú fejlesztői lapot használtam.



ESP32 WROOM, forrás: amazon.com

Az ESP32 WROOM modul egy olyan rendkívül népszerű, integrált Wi-Fi és Bluetooth SoC (System on a Chip) modul, amelyet az Espressif Systems fejlesztett ki. Ennek a modulnak a központi eleme a már említésre került ESP32 mikrovezérlő egység, amely teljes kétmagos processzorral rendelkezik.

Néhány specifikus tudnivaló az ESP32 WROOM modulról:

Mikrovezérlő és Teljesítmény:

Kétmagos Processzor: Az ESP32 WROOM rendelkezik két erős processzormaggal, amelyek gyors és hatékony adatfeldolgozást tesznek lehetővé.

Frekvencia: Általában a processzor 80 vagy 160 MHz-es órajelen működik, de konfigurálható egészen 240 MHz –ig is.

Alacsony Energiafogyasztás: Különleges alacsony energiafogyasztási módokkal rendelkezik, lehetővé téve hosszú akkumulátor-üzemidőt az eszközök számára, akár akkumulátorról betáplálva.

Kommunikáció:

Wi-Fi: Támogatja a 802.11 b/g/n Wi-Fi szabványokat, lehetővé téve a vezeték nélküli kapcsolódást, esetünkben akár 150 Mbps sávszélességen is.

Bluetooth: Az ESP32 WROOM modul támogatja a Bluetooth 4.2 és Bluetooth Low Energy (BLE) technológiákat, amelyek lehetővé teszik a különböző eszközök közötti adatátvitelt és kommunikációt.

Memória és Tárolás:

Flash memória: Különböző verziókban kapható, általában 4 MB flash memóriával rendelkezik, esetünkben azonban egy 16MB –os módolt használok.

RAM: Általában 520 KB vagy több RAM áll rendelkezésre a programok és adatok tárolására, jelen esetben 520KB –os verzióval dolgozom.

Perifériák és Támogatás:

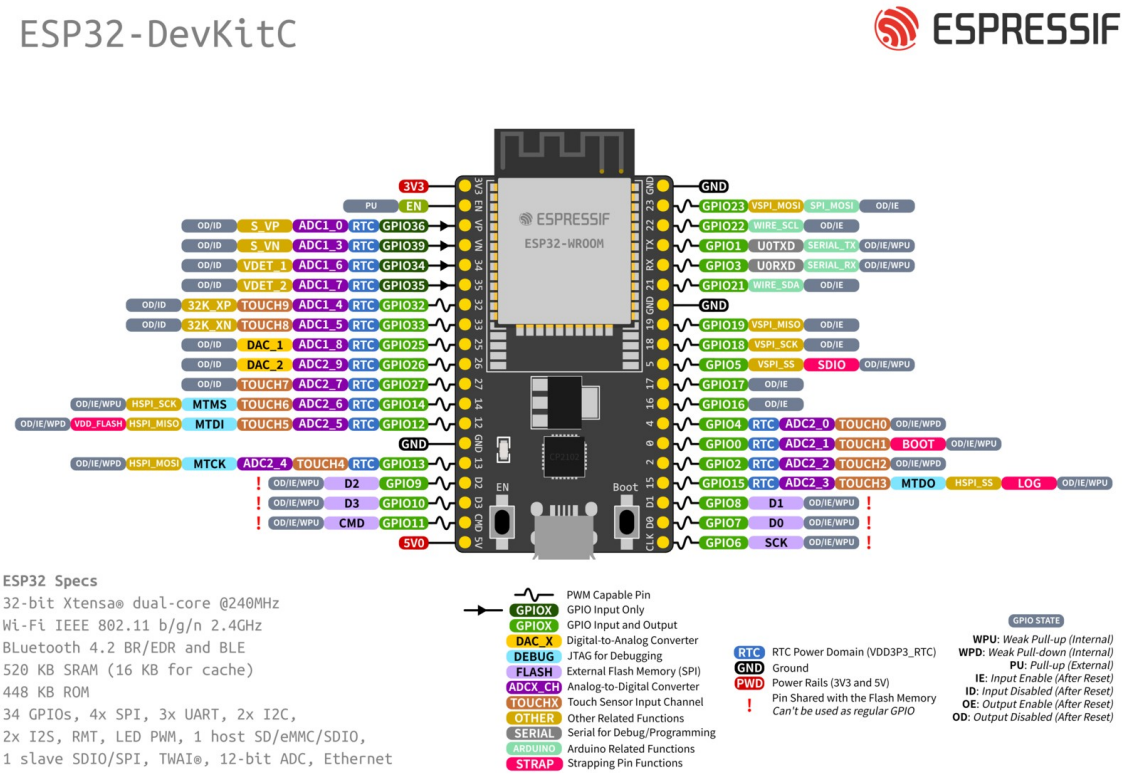
A GPIO (General Purpose Input/Output) a mikrovezérlők általános bemeneti és kimeneti tűje, amely lehetővé teszi az eszköz számára, hogy kommunikáljon a külső világgal. Az ESP32 WROOM modul számos GPIO tűvel rendelkezik, amelyek nagyfokú sokoldalúságot és rugalmasságot biztosítanak a külső eszközök csatlakoztatásában és azokkal való kommunikációban.

GPIO-k tulajdonságai az ESP32 WROOM modulon:

Tüskék száma és funkciók:

Az ESP32 WROOM általában 34, ahogyan esetünkben is vagy több GPIO tűvel rendelkezik, amelyek közül sok különböző funkcióval rendelkezik, mint például digitális bemenet/kimenet, PWM (Pulse Width Modulation) kimenet, analóg bemenet, és egyebek.

Az ESP32 WROOM lábkiosztása a következő:



Az ESP32 WROOM lábkiosztása, forrás: docs.espressif.com

Digitális Bemenet/Kimenet (GPIO):

A GPIO-k digitális bemenetként vagy kimenetként működhetnek. Digitális bemenetként érzékelhetik a külső eseményeket vagy jeleket, míg digitális kimenetként tudnak jeleket küldeni más eszközök felé.

Pulzusszélesség-moduláció (PWM) kimenetek:

Ezen tüskék lehetőséget adnak a PWM jelek generálására, amelyek alkalmazhatók például motorok sebességének szabályozására, LED-ek fényerejének szabályozására vagy más analóg jel szimulációjára.

Analóg Bemenet:

Néhány GPIO képes analóg jeleket fogadni, ami lehetővé teszi analóg érzékelők csatlakoztatását és azok által mért értékek kiolvasását.

Konfigurálhatóság:

Az ESP32 lehetővé teszi a GPIO tüskék konfigurálását, így a fejlesztők széles körű irányítást kaphatnak azok működésének beállításában és funkcióiknak meghatározásában.

Az ESP32 WROOM GPIO tüskéi számos projektben használhatók, legyen szó robotikáról, IoT eszközökről, szenzorokról vagy egyéb elektronikai alkalmazásokról. A GPIO-k nagy szerepet játszanak az eszközök kommunikációjában a külső eszközökkel, és a megfelelő konfigurálással rendkívül sokféle feladatot elláthatnak az elektronikában.

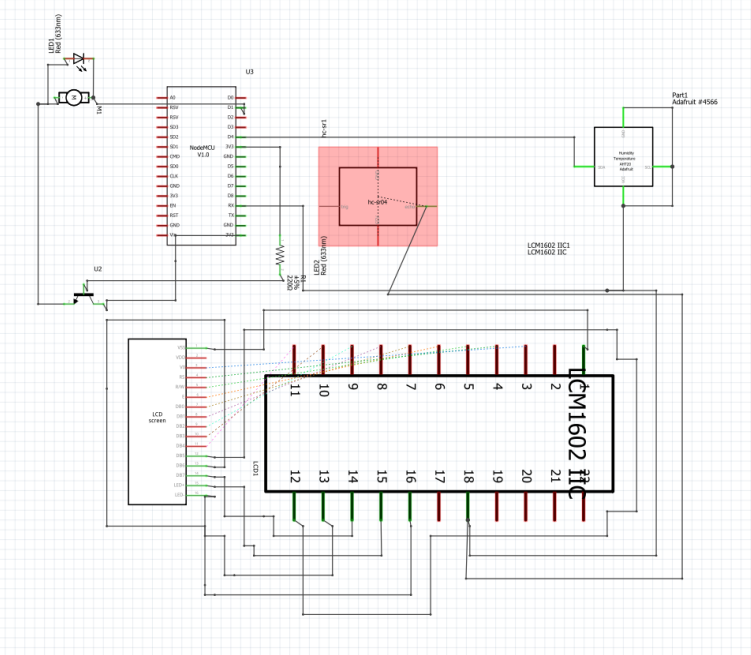
Fejlesztői támogatás
Széles körű fejlesztői támogatást nyújt, köztük az ESP-IDF (Espressif IoT Development Framework) és az Arduino IDE, amelyekkel könnyen programozható és fejleszthető. A projekt fejlesztése azonban egy modernebb környezetben történt, a Visual Studio Code PlatformIO bővítménye segítségével. A Visual Studio Code (VS Code) egy népszerű fejlesztőkörnyezet, melyet széles körben használnak különféle programozási projektekhez. A PlatformIO egy kiterjesztés (bővítmény) a VS Code-hoz, mely lehetővé teszi a beágyazott rendszerek fejlesztését és programozását, ideértve az ESP32 mikrovezérlőt is.

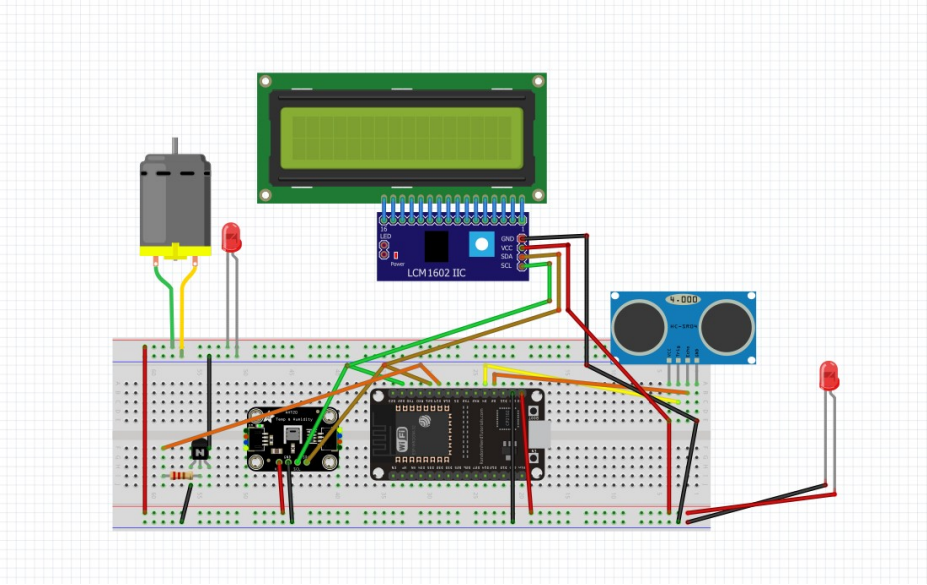
A PlatformIO VS Code bővítménye ideális választás az ESP32 és más beágyazott rendszerek fejlesztéséhez, mivel biztosítja a könnyű kezelhetőséget és a teljes funkcionalitást egyetlen környezeten belül. A fejlesztők széles körű eszközöket és támogatást kapnak az ESP32 programozásához és fejlesztéséhez ezen a platformon.

Fő előnyök:

- Egyszerűség: Könnyen kezelhető fejlesztői környezetet nyújt az ESP32 és más mikrovezérlők számára.
- Teljes funkcionalitás: Teljes körű fejlesztési funkciókat biztosít, mint például fordítás, feltöltés és debuggolás.
- Sokoldalúság: A PlatformIO támogatja többféle mikrovezérlőt, nemcsak az ESP32-t.

Tervdokumentáció





A rendszer prototípus bekötési rajza

Alkarész lista

A rendszer több komponensből épül fel. Funkcionalitás szempontjából 4 csoport szerint osztályozhatjuk:

- Feldolgozó egység
- Érzékelő egység

- Beavatkozó egység
- Monitorizáló egység

Feldolgozó egység
A rendszer központi eleme a feldolgozó egység, a már korábban tárgyalt és bemutatott ESP32 WROOM (30 db GPIO kimenetes verzió) típusú mikrovezérlő. Azért esett erre a típusú egységre a választásom, mivel olcsó és a célnak megfelelő számítási kapacitással és kellő számú interface –szel (GPIO kimenettel) rendelkezik.

Érzékelő egység
A rendszer érzékelői biztosítanak értékeket, melyek alapján a központi egység képes kiszámolni, hogy milyen jelre is van szüksége a beavatkozónak a helyes működés érdekében. A rendszer 3 paraméter érzékelésére képes:

- Hőmérséklet
- Páratartalom
- Távolság

Mivel a hőmérséklet igencsak befolyásolja a növények bioritmusát és fejlődését egy zárt légtérben is, fontos, hogy pontosan mérni tudjuk ezt. A méréshez egyetlen szenzort használtam, egy **AHT10** típusú érzékelőt, mely képes mindkettő mérésére és továbbítására a feldolgozó egység fele I^2C protokollon kérészül. Az AHT10 egy digitális hőmérséklet- és páratartalom-szenzor, melyet az Aosong (Guangzhou) Electronics Co., Ltd. fejlesztett ki. Az AHT10 szenzor kiváló tulajdonságokkal rendelkezik a mérési pontosság és a működési paraméterek tekintetében. Az alábbiakban látható néhány főbb technikai jellemzője:

- Hőmérséklet-tartomány: -40°C és +85°C közötti mérési tartományt képes lefedni.
- Hőmérséklet-pontosság, felbontás: Körülbelül ±0,3°C-os (maximum) hőmérsékleti pontosságot biztosít a mérési tartományon belül, 16 bites felbontással
- Páratartalom-tartomány: 0% -tól 100% -ig terjedő relatív páratartalom mérésére képes.
- Páratartalom-pontosság, felbontás: Körülbelül ±2%-os relatív páratartalom pontosságot nyújt, 16 bites felbontással

A hőmérséklet és a páratartalom változik a földszinthez képest mért magasság függvényében, ezért fontos tudni azt is, hogy az adott szenzor mekkora földszinthez viszonyított magasságon van elhelyezve. Ennek okán a rendszerbe beépítésre került egy ultrahangos távolságmérő szenzor is. A szenzor egy **HC-SR04** típusú érzékelő. Az HC-SR04 ultrahangos távolságmérő szenzor egy népszerű és könnyen használható eszköz, mely az ultrahangot használja távolságok mérésére. Ezek a szenzorok gyakran alkalmazhatók robotikai projektekben, távolságméréshez vagy akár akadályok érzékeléséhez is. Itt vannak néhány főbb technikai jellemzői:

- Mérési Távolság: Az HC-SR04 szenzor körülbelül 2 cm-től 400 cm-ig terjedő távolságok mérésére alkalmas.
- Pontosság: Általában körülbelül ±3 mm-es pontosságot biztosít a mérési távolságtól függően.
- Működési Elv: Az eszköz ultrahangos impulzusokat bocsát ki, majd méri azok visszaverődési idejét, hogy meghatározza a távolságot.
- Mérési Felbontás: A szenzor 1 cm-es felbontással képes mérni a távolságot.
- Jellemzők: A HC-SR04 alacsony költségű, egyszerűen kezelhető és könnyen integrálható különböző elektronikai projektekbe.

Beavatkozó egység

A rendszer beavatkozó egységei segítenek biztosítani minél ideálisabb körülményeket ahhoz, hogy a növények életkörülményei a lehető legideálisabbak legyenek. Esetünkben egyetlen egységet használunk, ami egy tranzisztor által vezérelt motor, melyre ventilátort erősítünk, ezzel segítve a levegő keringgetését a melegágyban.

A vezérlő tranzisztor egy **2n2222** típusú általánosan használt NPN (negatív-pozitív-negatív) bipoláris tranzisztor. Ezt a tranzisztort széles körben alkalmazzák különféle elektronikai áramkörökben, például erősítőkben, kapcsolókban és más alkalmazásokban. Néhány fontos jellemzője:

- Funkció: Az 2N2222 NPN tranzisztor egy közös emitterű tranzisztor, amely vezérlésre, kapcsolásra és erősítésre alkalmas.
- Maximális Feszültség és Áramerősség: Általában a maximális kollektor-emitter feszültség körülbelül 40V, a kollektor-áram 800mA körül van.
- Átviteli tényező (béta): Az 2N2222 béta értéke általában 100-300 között változik, ami azt mutatja meg, hogy milyen mértékben erősíti a bemenő áramot a kimeneten.
- Felépítés: Ez egy tokozott tranzisztor, ami azt jelenti, hogy könnyen integrálható a nyomtatott áramkörbe.
- Alkalmazások: Az 2N2222-t sokféle alapvető elektronikai áramkörben használják, beleértve az erősítőket, a jelek kapcsolását és vezérlését különböző elektronikai rendszerekben.

A prototípus esetben a vezérelt motor egy **6 voltos DC**, azaz egyenáramú motor, ami csak szemléltetési céllal van jelen a projekt prototípus fázisában, ezért nem is részletezném a specifikációit.

Monitorizáló egység

A rendszer paraméterei élőben követhetőek. A paraméterek leolvashatóak fizikailag egy LCD kijelzőről, továbbá Digitális felületről is, melyről a későbbiekben beszélek. Szellőztető ventilátor állapotát (tehát be vagy kikapcsolt) egy további LED jelzi mely, amikor ég, akkor a rendszer hűt, mikor nem akkor ellenkezőleg.

A használt **LCD 1602 I²C** (Inter-Integrated Circuit) típusú kijelző egy olyan alfanumerikus LCD kijelző, melyet az I2C kommunikációs protokollon keresztül lehet vezérelni. Ezek a kijelzők széles körben alkalmazottak különféle elektronikai projektekben, mivel könnyen integrálhatók és sok információt képesek megjeleníteni. Néhány fontos jellemzője:

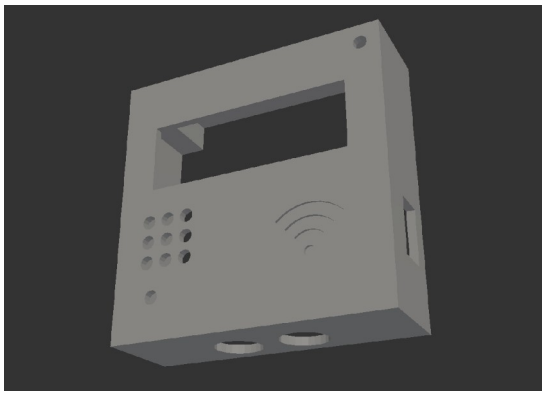
- 1602 Felbontás: Az 1602 azt jelenti, hogy a kijelző 16 karaktert képes soronként megjeleníteni, összesen pedig 2 sorban.
- I2C Kommunikáció: Az I2C interfész használatával történik a kommunikáció, ezáltal kevesebb GPIO tüskét igényel a vezérléshez.
- Háttérvilágítás: Általában háttérvilágítással rendelkezik, ami lehetővé teszi az olvashatóságot különböző fényviszonyok között.
- Könnyű Kezelhetőség: Az I2C interfésznek köszönhetően könnyen és kevesebb vezetékkel lehet vezérelni, ezáltal egyszerűbb az integráció.

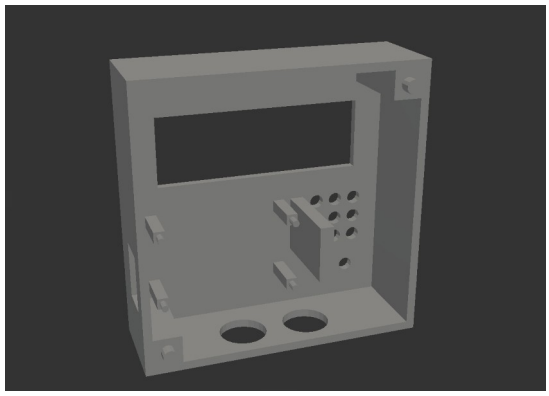
Az LCD kijelzőn látható az adott hőmérséklet és páratartalom érték, valamit a földtől mért magasság és a ventilátor állapota.

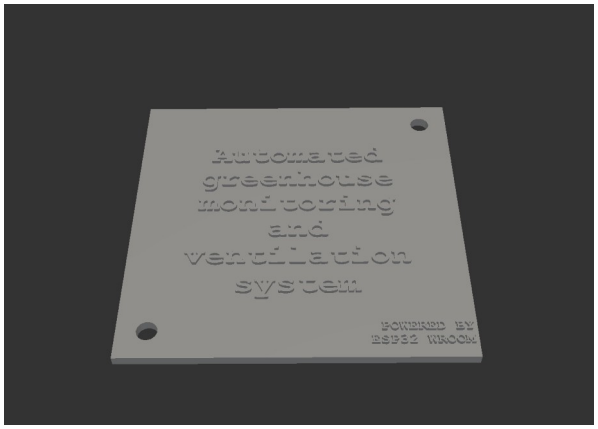
Megvalósítás és működés leírása, következtetések

Tokozás tervezése

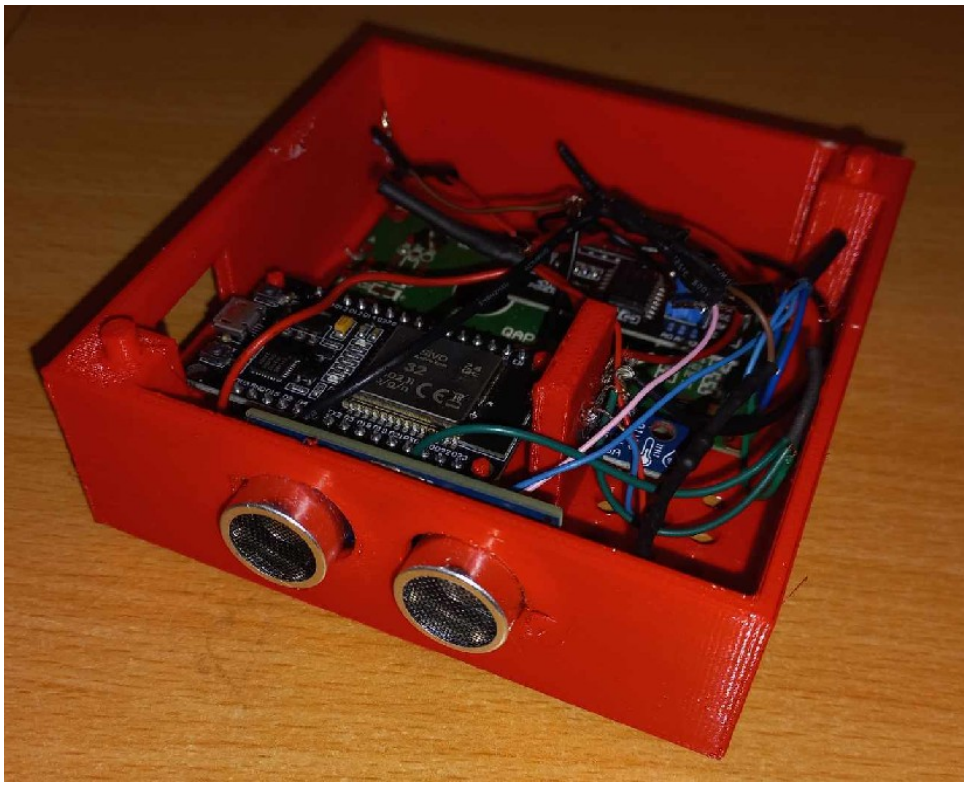
A teljes hardver egy 3D –s nyomtatóval készült tartó dobozban kapott helyet. A 3 dimenziós modell Autodesk Inventor Professional 2024 programban készült. A doboz anyaga egy speciális műanyag, a PET-G, ami ellenálló az időjárási viszonyoknak. A modellt egy Ultimaker 3 típusú 3D nyomtató valósította meg. Alább látható a 3D –s modell és az elkészült fizikai tárgy.







3 dimenziós modellek



Fizikailag elkészült hardver tokozás és a már összeszerelt szerkezet

Megvalósított rendszer leírása

A tervezett rendszer az elképzelés és a megvalósítás között nem mutat sok különbséget. A megvalósítás során több problémába ütköztem, de többségében megoldódott és hasznos tanulságokkal gazdagodtam.

A végső rendszer működése a következők szerint alakult. A tervezetthez képest közel minden funkcionalitás megvalósításra került. Az apró módosítások csupán azért történtek, mert a fejlesztés során rájöttem, hogy jobb megoldások is léteznek az egyes feladatokra. Ilyen volt például a relé lecserélése NPN tranzisztorra.

A rendszer azonnal bekapcsol, amikor tápforrást kapcsolunk rá. Ez jelen pillanatban az ESP32 USB interfészén keresztül van megvalósítva. Miután bekapcsolt azonnal elkezdi az ESP32 a felprogramozott inicializálási folyamatot. Ennek során inicializáljuk az ultrahangos érzékelőt, az LCD kijelzőt, beállítjuk a tranzisztor által vezérelt feszültség forrás kimenetét, a soros monitort. Inicializáljuk továbbá a hőmérséklet és páratartalom érzékelőt, a beépített speciális flash alapú fájlrendszert, illetve a WiFi modul is. A továbbiakban beállítjuk a web szervert és az ehhez tartozó HTTP végpontokat.

Ha az inicializálás sikeres volt minden rétegben, akkor elindul az ismétlő ciklus. Ennek megadott időpillanatokban az a feladata, hogy lekérje a szenzor által olvasott hőmérsékelt és páratartalom adatot, ezután pedig küldje el a webszervernek, hogy az ki tudja jelezni a webes interfészen.

Az ismétlő ciklusban kiírjuk sorra a szenzorok által olvasott értékeket, a ventilátor állapotát és a földtől mért távolságot is időben váltogatva. Itt történik minden ciklusban a ventilátor állapotának beállítása is.

Következtetések és lehetőségek a tovább fejlesztésre

A rendszer tervezésekor több következtetésre jöttem rá. Első sorban elmondhatom, hogy nem tartom napjainkban bonyolultnak egy személyre szabott, egy az enyémhez hasonló rendszer megtervezését és megvalósítását, mivel a gyártók lényegesen leegyszerűsítették napjainkra a programozást és a prototípus gyártást egyaránt.

A rendszer tovább fejleszthető akár addig, hogy fizikai termékként eladható legyen. A fizikai megvalósítás terén annyiban fejleszthető, hogy az összes alkatrészt egy nyomtatott áramkörre forrasztjuk. A nagyobb fejlesztési potenciál a szoftver terén jöhetne létre. Első sorban, jelenleg a rendszer csak egy darab felprogramozott modell szerint tud viselkedni. Ezt tovább fejleszthetnénk mindaddig, ameddig akár a felhasználó saját profilokat állíthat be fajokra specifikálva akár éjszakai, nappali időszakokra, vagy figyelembe véve az egyes fajok fejlődési fázisaira ideális paramétereket. Ehhez szükség van a web szerver és interfész fejlesztésére is.

A rendszer bővíthető más szenzorokkal, például egy termőföld nedvesség érzékelővel, és így automatizálni tudnánk az öntözést is.

Beágyazott vezérlő programok rövid leírása

Szenzor inicializálás

```
void initATHX0()
{
  if (!aht.begin())
  {
    Serial.println("Could not find AHT? Check wiring");
    while (1)
      delay(10);
  }
  Serial.println("AHT10 or AHT20 found");
}
```

Ez a kód inicializálja az AHT10 vagy AHT20 szenzort. Ha a szenzor inicializációja sikertelen, hibaüzenetet küld a soros monitorra, majd végtelen ciklusba lép, ami megállítja a programot. Ha sikeres az inicializáció, akkor egyszerűen üzenetet küld a soros monitorra, hogy az AHT10 vagy AHT20 szenzor megtalálható.

Szenzor érték olvasás

```
String getSensorReadings()
{
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);

  readings["temperature"] = String(temp.temperature);
  readings["humidity"] = String(humidity.relative_humidity);

  String jsonString = JSON.stringify(readings);
  return jsonString;
}
```

Ez a kód érzékelő méréseket végrehajt, majd visszaad egy JSON objektumot. Az AHT érzékelőből lekérdezi a hőmérsékletet és a páratartalmat, majd ezeket hozzáadja egy readings nevű JSON objektumhoz. Végül a JSON objektumot karakterlánc formájában visszaadja.

Az eredményül kapott JSON objektum például így nézhet ki:

```
{
  "temperature": "25.5",
  "humidity": "50.2"
}
```

SPIFFS inicializálása

```
void initSPIFFS()
{
  if (!SPIFFS.begin())
  {
    Serial.println("An error occurred while mounting SPIFFS");
  }
```

```
  Serial.println("SPIFFS mounted successfully");
}
```

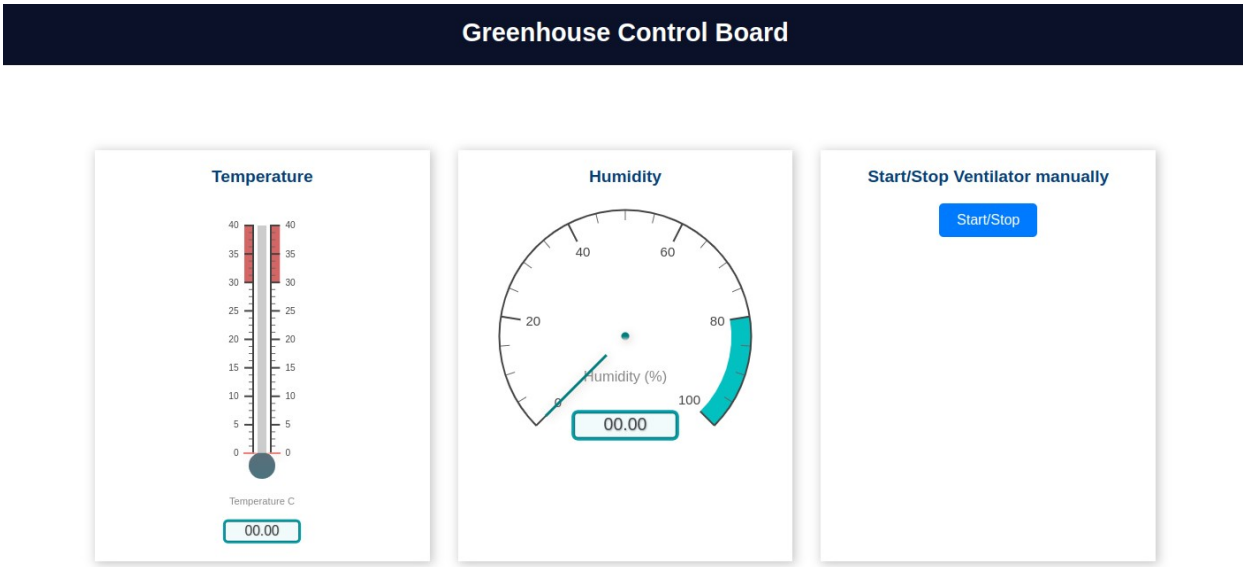
Ez a kód inicializálja a SPIFFS fájlrendszert. Ha az inicializáció sikertelen, hibaüzenetet küld a soros monitorra. Ha sikeres, akkor jelzi a soros monitorban, hogy a SPIFFS sikeresen csatolva lett.

WiFi modulinicializálása

```
void initWiFi()
{
  for (int i = 1; i <= 3; i++) {
    delay(1000);
    Serial.println(i);
  }
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}
```

Felhasználói interfész programok rövid leírása

A felhasználó felület a következően néz ki:



Az első kártyán látható a mért hőmérséklet érték, a másodikon a páratartalom érték, az utolsón pedig egy gomb található, amivel a ventilátor manuálisan vezérelhető (azaz be és ki kapcsolható).

A felület reszponzív, mobilon is megfelelően követhetőek az adatok.

A felület nem teljes mértékben saját fejlesztés, az alap és a design is a https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/raw/master/Projects/ESP32/ESP32_Gauges/ESP32_Gauges.zip webhelyen érhető el és a Random Nerd Tutorials tulajdona. A felhasználó felület csak saját felhasználásra lett felhasználva és kijelentem, hogy semmiféle bevételt termelő szolgáltatáshoz nem fogom felhasználni.

Az eredeti kódban más típusú szenzorokat és mikrovezérlő volt használva, ezért szükség volt az átalakításra. A felhasználói felület három komponensből épül fel:

- index.html
- styles.css
- script.js

A .html fájl a weboldal vázát, a .css az oldal stílusát, a .js pedig a viselkedését leíró nyelv. A weboldal használ továbbá .css és .js fájlokat, melyeket az internetről érhetünk el, a következő címeken:

```
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
<script
src="http://cdn.rawgit.com/Mikhus/canvas-gauges/gh-pages/download/
2.1.7/all/gauge.min.js"></script>
```

A weboldal működése a https://randomnerdtutorials.com/esp32-web-server-gauges/ webcímen olvasható. Ezen dokumentációnak nem célja ennek az ismertetése, ezért csak a hozzátett részekre térnék ki. Pár mondatban a működése a következő:

```
source.addEventListener('new_readings', function (e) {
    console.log("new_readings", e.data);
    var myObj = JSON.parse(e.data);
    console.log(myObj);
    gaugeTemp.value = myObj.temperature;
    gaugeHum.value = myObj.humidity;
  }, false);
```

A fenti kódrészlet segítségével az ESP32 által beolvasott és feldolgozott adatok továbbítva lesznek a webszerver irányába a new_readings végponton keresztül és az új értékek beállításra kerülnek a grafikonokon.

A weboldalon szereplő 2 kártyához hozzáadtam további 1 kártyát, melyen egy be és kikapcsoló gomb kapott helyet. Ezt lenyomva megváltoztathatjuk a ventilátor aktuális állapotát. Az ezt vezérlő .js kódrészlet a következő:

```
function toggleLED() {
  fetch('/toggle')
    .then(response => response.text())
    .then(data =>
          console.log('Válasz: ' + data);
    })
    .catch(error => {
          console.error('Hiba történt: ' + error);
    });
}
```

A fenti függvény küld egy HTTP GET kérést az ESP32 felé a "/toggle" végpontra. A fetch() függvénnyel könnyedén megtehetjük ezt. A válaszban kapott értéket opcionálisan feldolgozhatjuk.

A weboldalon található grafikonok dinamikusan változnak meghatározott időpillanatokban.

A weboldalt az ESP32 ossza meg lokális hálózaton azon a címen, amit olvashatunk a soros monitoron, miután sikeresen csatlakozott az előre definiált WiFi hálózathoz. A SPIFFS flash alapú fájlrendszert a Visual Studio Code Platform IO bővítménye segítségével fordítottam és töltöttem fel a mikrovezérlőre.

Forráskódok

Projekt tervezésekor és megvalósításakor használt összes forráskódegtalálható a GitHub felületen. Az elérése a következő:

https://github.com/VasiAndras/Design-with-esp32-type-microcontrollers/tree/main/

/ServerTemp/src/main.cpp

Ebben a fájlban találhatóak a beágyazott rendszerhez tartozó viselkedést leíró c++ nyelvben írt kódok.

/ServerTemp/data/

Ezen könyvtárban található a weboldalt kódoló összes fájl (favicon.png; index.html; script.js; style.css)

/case3Dmodel/

Ezen könyvtárban találhatóak a fizikai tokozás 3D modellek, melyek 3D nyomtatóval kinyomtathatóak.

/platformio.ini

A platformio.ini fájl a PlatformIO projekt konfigurációs fájlja. Ebben a fájlban konfigurálhatod a hardvert, a firmware-t és más beállításokat a fejlesztési projekt számára. Az alábbiakban a különböző beállítások jelentése található:

```
[env:denky32]: Meghatározza az ESP32 mikrovezérlő specifikus
környezetet a konfigurációhoz, amelyet "denky32"-nek nevez el.
platform = espressif32: Kiválasztja az ESP32 mikrovezérlő platformját,
amely a fejlesztéshez szükséges eszközöket és könyvtárakat biztosítja.
board = denky32: Kiválasztja a "denky32" nevű fejlesztői lapkát
(board) az ESP32 mikrovezérlőhöz. Ez az elnevezés lehetővé teszi a
PlatformIO számára, hogy a megfelelő konfigurációkat és beállításokat
használja a kiválasztott laphoz.
framework = arduino: Kiválasztja az Arduino keretrendszert a
fejlesztéshez, ami lehetővé teszi az Arduino stílusú programozást az
ESP32 mikrovezérlőn.
build_flags = -DMBEDTLS_CONFIG_FILE=\"mbedtls/esp_config.h\": A
fordító kap flags-ként ezt az opciót, ami azt mondja meg neki, hogy
használja az "esp_config.h" nevű fájlt a mbedTLS (TLS könyvtár)
konfigurációjához.
lib_deps: A projekt függőségeinek listája. Ezek a könyvtárak és
kiegészítő modulok, amelyeket a projekt használ. A felsorolt
könyvtárak a lib_deps sorban vannak megadva, például:
Arduino_JSON
Adafruit AHTX0
https://github.com/me-no-dev/ESPAsyncWebServer.git: Ez a könyvtár egy
aszinkron webszervert biztosít az ESP32 mikrovezérlőhöz.
AsyncTCP
SPIFFS
```

```
Wire.h
LiquidCrystal_I2C
teckel12/NewPing@^1.9.7: Ez egy ultrahangos távolságérzékelő modulhoz
használt könyvtár, amely lehetővé teszi a távolság mérését.
```

Ezen konfigurációk alapján a PlatformIO képes lesz a megfelelő környezetet és könyvtárakat használni a "denky32" fejlesztői lapkához, és a projekt így megfelelően konfigurálva lesz az ESP32 fejlesztéséhez.