Liceul Teoretic “Salamon Ernő” Gheorgheni

Lucrare de atestat disciplina INFORMATICĂ

Profesor coordonator:
Csíki Zoltán

Elev:
Kedves Attila-János

Gheorgheni
2025

Okos melegház

Vezérlő

-IoT project/Automatizált rendszer-

Tartalom

Bevezetés

A vizsgamunkámon gondolkodva több egymástól eltérő projekt ötlet is megfogalmazódott bennem. Abban viszont biztos voltam, hogy amit véghez szeretnék vinni az legyen hasznos és képviseljen valamilyen értéket. Olyas valamit szerettem volna készíteni, ami nem csak számomra érdekes, hanem bárki más számára is alkalmazható.

Az automatizáció már mondhatni régóta létező technológia, mégis úgy gondolom, hogy nem elérhető mindenki számára, és kevesen alkalmazzák kiváltképpen a mezőgazdasági szektort nézve észlelhető, hogy egy egyszerű automatizációt alkalmazva nagy produktivitási eredmények érhetőek el. De ha ilyen hasznos lehet az automatizáció, akkor feltevődik a kérdés, hogy miért nincs elterjedve. Szerintem ez a kérdés abból adódik, hogy nem ezen technológiák nem kifejezetten elérhetőek magánszemélyek számára, és csak a nagy tőkével rendelkező mezőgazdasági cégek engedhetik meg maguknak az efféle fejlesztéseket. Nézeteim szerint erre megoldást a nyílt forráskódú (opensource) projekt nyújthatnak. Ily módon könnyebben eljuthat a technológia olyan felhasználókhoz, akiknek azelőtt nem volt rá lehetőségük.

Az IoT, rövidítés az angol „**Internet of things**” szókapcsolatból jön, és magyarul annyit tesz, hogy **dolgok internete.** Ez röviden a különböző elektronikai eszközöket jelenti, amelyek képesek felismerni, esetleg feldolgozni valamilyen lényegi információt, és egy másik hálózaton lévő eszközzel kommunikálni. Ebből jól következik, hogy miért fontos, hogy egy automatizált vezérlő ilyen, IoT csoportba tartozó legyen. Egy melegházvezérlő egyik legfőbb tulajdonsága az, hogy a felhasználó távollétében is ellenőrizni tudja az aktuális paramétereket, ezt egy IoT eszköz tudja.

Ezen rendszer legfőbb értéke az a szenzoros értékek feldolgozásában rejlik. Úgy terveztem, hogy képes legyen minden, egy melegházat jellemző paraméter vizsgálatára. Ez a páratartalom, hőmérséklet, föld nedvessége, és a fény erősségé, kevésbé fontosnak tartottam, de meghagytam a széndioxid szint mérését, mint bővíthető funkció. Fontosnak tartottam a külső hőmérséklet és páratartalom vizsgálatát, hogy követhető legyen, hogy a környezeti tényezők hogyan befolyásolják a melegház belső állapotát.

Az internetes kezelés mellett fontos, hogy a funkciók helyszínen láthatóak és módosíthatóak legyenek. A nyomon követést egy 0.96 colos oled kijelző teszi lehetővé, ami menüpontokra leosztva jeleníti meg az adatokat. A módosításokat egy joystick modul teszi lehetővé.

Persze az adatok feldolgozása és kezelése mellett alapvető a belső környezet aktív, mesterséges befolyásolása. Erre egy 6 csatornás relé modul nyújt lehetőséget, amely be és ki kapcsolja a fény ellátást, párásítást, öntözést, és ventilátorok állapotát.

Végső soron úgy gondolom, hogy fontos olyan automatizációs rendszerek létrehozása, amely elérhető széleskörű felhasználásra magán személyek, illetve kisvállalkozások számára egyaránt. Illetve szintén fontos, hogy egy ilyen automatizációs rendszer opensource legyen, hogy mindenki számára érthető és a jövőben továbbfejleszthető legyen. Innen fogva fel is eleveníteném *Linus Torvalds* egyik gondolatát: „Az intelligencia abból fakad, hogy az emberek képesek mások ötleteit felhasználni és továbbfejleszteni.”

Fejlesztési lehetöségek

A jelenlegi melegházvezérlő rendszer számos tekintetben sikeresen valósítja meg az alapvető funkciókat – például a hőmérséklet- és páratartalom-mérést, illetve azok OLED kijelzőn való megjelenítését, valamint a szenzoradatok továbbítását az ESP8266 modul segítségével egy belső hálózati szerverre. Ugyanakkor több irányban is lehetőség nyílik a rendszer továbbfejlesztésére és optimalizálására, amelyek nemcsak a teljesítményt növelik, hanem a rendszer megbízhatóságát és funkcionalitását is jelentősen javítják.

1. Memóriaoptimalizálás

A rendszer jelenlegi állapotában több helyen is redundáns, ismétlődő szövegkiírásokkal operál, például az OLED kijelzőre történő kiírásoknál. Ezeket a szövegrészeket célszerű statikus karaktertömbökbe, illetve #define vagy const char* változókba szervezni, így csökkenthető a SRAM terhelése. Emellett a String típus használata helyett ajánlott a C-szerű char[] tömböket alkalmazni, mivel a String típus fragmentálhatja a memóriát hosszú futás során.

2. AT parancsok elhagyása

Az ESP8266 vezérlése jelenleg valószínűleg AT parancsokon keresztül

történik, ami nem a leghatékonyabb megoldás. Az AT firmware helyett ajánlott a NodeMCU (Lua) vagy az ESP8266 Arduino core használata, és a mikrokontroller közvetlen programozása. Így nem kell AT parancsokat küldeni soros kapcsolaton keresztül, hanem maga az ESP8266 veszi át a vezérlést, és az Arduino Mega csak szenzoradatokat továbbít. Ez növeli a válaszidőt, csökkenti a hibalehetőségeket, és bővíti a lehetőségeket (pl. HTTPS kapcsolat, JSON kezelés).

3. Külső hálózatról való elérhetőség

A rendszer jelenleg csak a belső hálózaton elérhető. Egy következő fejlesztési lépés lehet a webes interfész interneten keresztüli elérhetősége. Ez megvalósítható dinamikus DNS (DDNS) használatával, port forward segítségével a routerben, vagy – biztonságosabb és skálázhatóbb megoldásként – egy felhőalapú szerverre történő adatküldéssel (pl. MQTT, HTTPS POST REST API használatával). Az adatokat akár egy valós idejű webes dashboard is megjelenítheti, pl. Node-RED.

4. Adatbiztonság és titkosítás

A továbbított adatok jelenleg valószínűleg titkosítatlanul utaznak a hálózaton, így egy fontos fejlesztési lehetőség a titkosított adatkommunikáció bevezetése. Ez történhet HTTPS használatával, vagy alacsonyabb szinten AES titkosítással (akár Arduino oldalon titkosítás, szerver oldalon dekódolás). Ez különösen fontos, ha az eszköz internetre van kötve, vagy szenzitív adatokat is kezelne a jövőben.

5. Modularizált és újrahasznosítható kód

A projekt további strukturálásával egy általánosabb, könnyen újrahasználható kódbázis is kialakítható. Célszerű az egyes hardverelemek kezelését külön osztályokba vagy függvénycsoportokba szervezni (pl. SHT szenzor kezelő, kijelző vezérlő, hálózati kommunikációs modul). Ez nemcsak olvashatóbbá és karbantarthatóbbá teszi a kódot, hanem előkészíti a projektet egy későbbi, több modulból álló rendszerré való bővítésre (pl. több zónás melegház).

6. Hibatűrés és diagnosztika

Hasznos lenne egy alapvető hibatűrő rendszer kialakítása, például watchdog timer használatával, amely újraindítja a rendszert, ha lefagy, vagy ha nem érkezik adat meghatározott ideig. Emellett egy logolási lehetőség is hasznos, amely menti a hibákat vagy a főbb eseményeket egy SD kártyára vagy a felhőbe.

Fejlesztési környezet

Arduino IDE 2.0

Egy arduino programozásában, mint a legtöbb programozási nyelv esetében egy úgynevezett **Integrált fejlesztői környezet,** röviden IDE használatos. Az **integrált fejlesztői környezet** (*integrated development environment*, **IDE**) egy olyan szoftveralkalmazás, amely átfogó létesítményeket biztosít a számítógépes programozók számára a szoftverfejlesztéshez. Az IDE általában legalább egy forráskódszerkesztőből, automatizálási eszközökből és hibakeresőből áll.

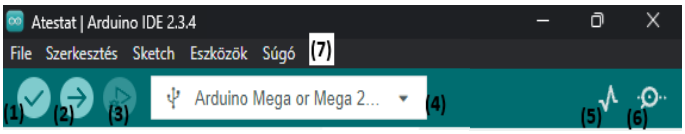
Az arduino programozására több IDE közül lehet választani. Az Arduino hivatalosan két különböző Arduino IDE-t kínál a fejlesztők számára:

1. **Arduino IDE 1.x** – Ez a klasszikus verzió, amely hosszú ideig az alapértelmezett fejlesztői környezet volt. Egyszerű, gyors, és széles körben támogatott.
2. **Arduino IDE 2.x** – Az új generációs Arduino IDE, amely modern felhasználói felületet, fejlettebb szerkesztési funkciókat (pl. automatikus kódkiegészítés, sötét mód) és jobb hibakeresési lehetőségeket kínál.

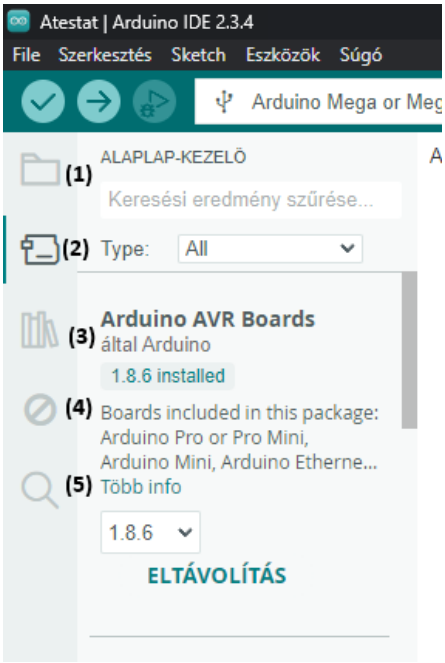
Én a fejlesztés során az Arduino IDE 2.0 választottam, mivel ez a verzió fejlesztőbarátabb környezetet nyújt.

Ez az IDE könnyű hibakeresési lehetőségeket biztosit. Elérhető egy úgynevezett **soros monitor,** ami segítségével nyomon követhetjük a változókat, lefutási időt, és függvények helyes meghívását. Ezt a kódba behelyezett egyszerű Serial.print(Val, Format) sorral tehetjük meg. **Val** – Az az adat, amit ki szeretnénk írni. Lehet szám, szöveg (string), karakter vagy változó. **Format**– Ez egy opcionális paraméter, amely megadja, hogy milyen formátumban jelenjen meg az érték.

Az arduino IDE 2 áttekintése



(1) Hibaellenörzés
(2) Feltöltés, a program feltöltése a mikrokontrollerre
(3) Ellenörzés/debugg



(1.) A használatban lévő fájlok kezelése
(2.) Alaplap-kezelő, különböző alaplapok telepítése
(3.) Könyvtárkezelő, könyvtárak telepítése
(4.) Hibakeresés, debugger
(5.) Keresés kulcsszó alapján a projektben

Verziokezelő

Verziókezelés alatt több verzióval rendelkező adatok kezelését értjük. Leggyakrabban a mérnöki tudományokban és a szoftverfejlesztésben használnak verziókezelő rendszereket fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg. Az egyes változtatásokat verziószámokkal vagy verzióbetűkkel követik nyomon.

A verziókezelő rendszerek (angolul: Version Control Systems, röviden VCS) olyan szoftvereszközök, amelyek lehetővé teszik a programkód vagy más digitális fájlok változásainak nyomon követését az időben. A szoftverfejlesztés során a forráskód folyamatosan változik: új funkciók kerülnek beépítésre, hibák kerülnek javításra, vagy éppen kísérleti módosításokat végeznek a fejlesztők. Egy jól működő verziókezelő rendszer lehetővé teszi ezen változások dokumentálását, visszakereshetőségét, valamint azt, hogy több fejlesztő egyszerre dolgozzon ugyanazon a projekten anélkül, hogy egymás munkáját véletlenül felülírnák.

A verziókezelés segítségével minden változtatás naplózásra kerül: ki, mikor és mit módosított. Ez nemcsak átláthatóbbá és biztonságosabbá teszi a fejlesztést, hanem visszaállíthatóvá is teszi egy korábbi, stabil állapotot, ha például egy új módosítás hibás működést okoz. A verziókezelés nemcsak a csapatmunkát segíti, hanem az egyéni fejlesztő munkáját is rendszerezi és archiválja.

A legismertebb és legelterjedtebb verziókezelő rendszer ma a Git, amelyet Linus Torvalds – a Linux operációs rendszer megalkotója – fejlesztett ki. A Git elosztott verziókezelő, tehát minden fejlesztő saját gépén is rendelkezik a teljes projekt történetével. Ez különösen hasznos lehetőség, ha ideiglenesen nincs internetkapcsolat, vagy ha biztonsági mentésre van szükség. A Git rendszer legnépszerűbb online tárhelye a GitHub, amely nemcsak a verziókezelést teszi lehetővé, hanem egy teljes közösségi felületet is kínál, ahol más fejlesztők projektjei is elérhetők, és akár együtt is lehet dolgozni.

Könyvtárak

A fejlesztés során a hatékony és gyors programozás érdekében különféle **könyvtárakat** (angolul **libraries**) használunk.

A "programkönyvtár" kifejezés egyszerűen egy olyan fájlt jelöl, ami lefordított tárgykódot (és adatot) tartalmaz, amit később egy programmal össze lehet szerkeszteni (link). A programkönyvtárak lehetővé teszik, hogy az alkalmazás modulárisabb, gyorsabban újrafordítható és könnyebben frissíthető legyen. A programkönyvtárakat három típusba sorolhatjuk: statikus programkönyvtárak, megosztott programkönyvtárak és dinamikusan betölthető (DL) programkönyvtárak.

Az Arduino ökoszisztémában a könyvtárakat **beépített kezelőfelületen** keresztül lehet telepíteni, és egyetlen **#include** utasítással beilleszthetők a kódba. Minden könyvtár egy adott hardver vagy funkció támogatására szolgál, például: #include <SPI.h> – A Serial Peripheral Interface (SPI) kommunikáció kezelésére szolgál.

- **#include <Wire.h>** – Az **I2C (Inter-Integrated Circuit)** kommunikációt kezeli.
- #include <Adafruit_GFX.h> – Az Adafruit Graphics Library, amely grafikus objektumokat (szöveg, vonalak, formák) rajzol a kijelzőre.
- #include <dht.h> – Egy egyszerű könyvtár a DHT11/DHT22 hőmérséklet- és páratartalom-érzékelők kezelésére.

Eszközök

A projekthez különböző hardveres eszközöket alkalmaztam, melyek biztosítják a vezérlés, az adatgyűjtés és a kijelzés működését. Ezek az eszközök egymással együttműködve valósítják meg a rendszer funkcionalitását.

Mikrovezérlő: Arduino és ESP8266

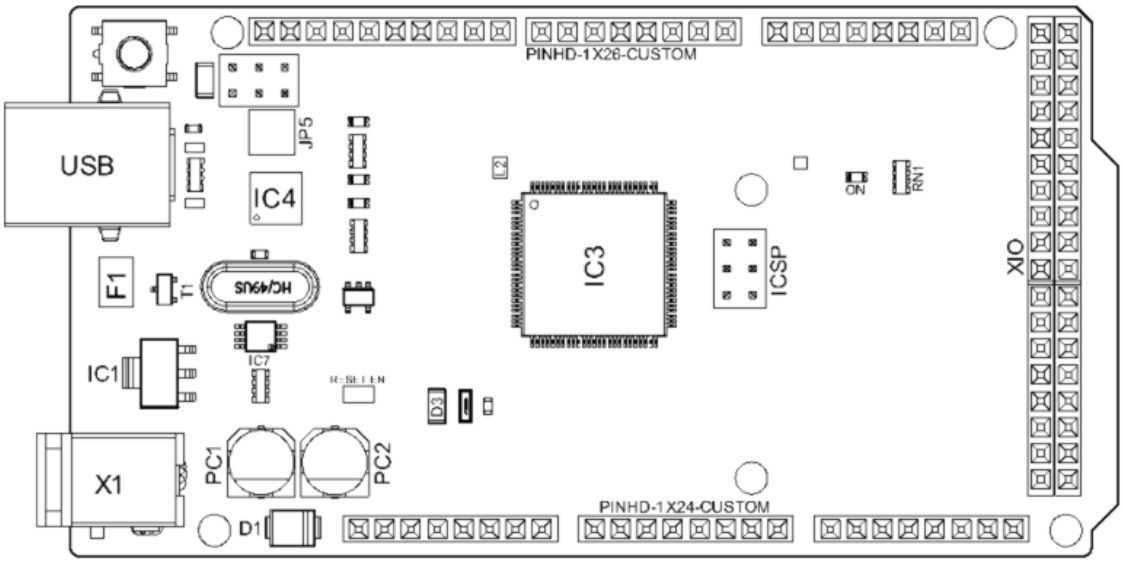
A projekt alapját egy Arduino Mega adja, amely a szenzorok és kijelzők kezeléséért felelős. Emellett az ESP8266 Wi-Fi modul lehetővé teszi a rendszer számára, hogy hálózatra kapcsolódjon és adatokat küldjön egy szerverre.

Arduino Mega

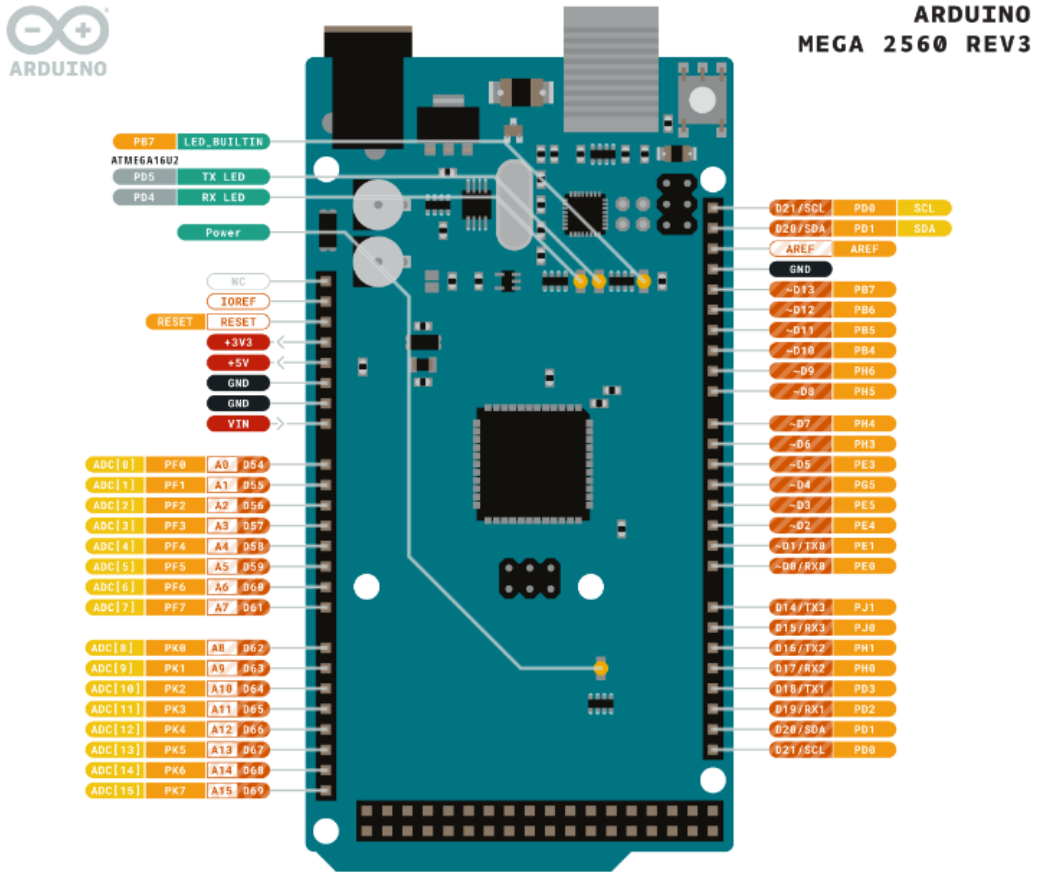
Az Arduino mikrokontroller végzi az érzékelők adatainak feldolgozását és a kimeneti eszközök vezérlését.

Az Arduino Mega 2560 egy ATmega2560-as mikrokontroller lapka. Rendelkezik 54 digitális be- és kimeneti tűvel (ebből 15 PWM-kimenetként használható), 16 analóg bemenettel, 4 UART (hardveres soros port), 16 MHz-es kristályoszcillátorral, USB-csatlakozóval, tápcsatlakozóval, ICSP-fejléccel és reset-gombbal.

Az arduino mega felépítése



Az arduino mega lábkiosztása



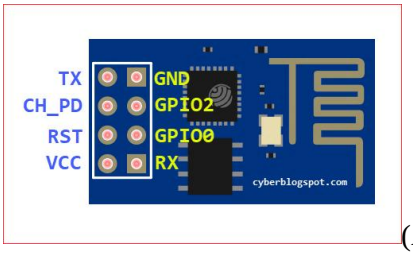
ESP8266:

Egy önálló Wi-Fi modulként is működő mikrokontroller, amely AT parancsok segítségével kommunikál soros kapcsolaton keresztül az Arduinóval, és lehetővé teszi az adatok internetre való továbbítását.

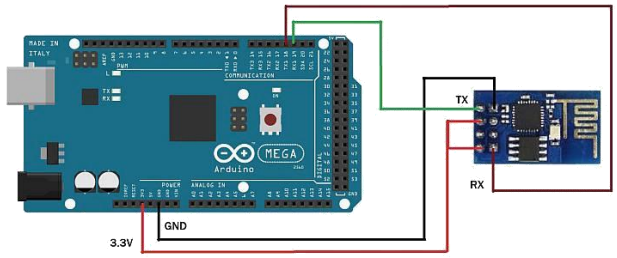
Az ESP8266 egy rendkívül népszerű, alacsony költségű Wi-Fi képes mikrokontroller modul, amelyet az Espressif Systems fejlesztett ki. A modul integrált TCP/IP protokoll stackkel és 32 bites RISC CPU-val rendelkezik, amely akár 80–160 MHz órajellel is működhet. Beépített Wi-Fi funkciójának köszönhetően ideális választás IoT (Internet of Things) projektekhez, mivel képes önállóan csatlakozni vezeték nélküli hálózatokra, vagy akár egy másik mikrokontroller kiegészítőjeként működni Wi-Fi modemként.

Az ESP-01 mindössze 8 lábbal rendelkezik, ebből kettő GPIO portként (GPIO0, GPIO2) használható, a többi a tápellátást, a soros kommunikációt (TX/RX), illetve a vezérlést szolgálja.

A programozása történhet AT parancsokkal egy másik mikrokontrolleren keresztül (például Arduino UNO-val), vagy közvetlenül is, ha rendelkezésre áll megfelelő USB–soros átalakító és az Arduino IDE-ben a megfelelő firmware.
Az alábbi ábrán látható a mikrokontroller tüskéinek funkciója (Ábra 1), illetve egy példa a csatlakoztatására egy arduino mega-val (Ábra 2.).



(Ábra 1.)



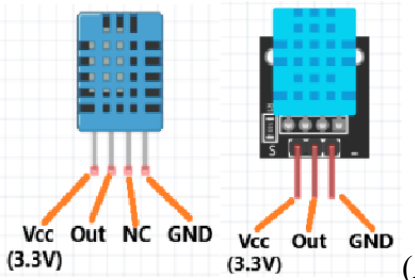
(Ábra 2,)

Érzékelők (szenzorok)

A rendszer működéséhez különféle érzékelőket használtam, amelyek lehetővé teszik a környezeti adatok mérését és feldolgozását.

DHT11 – Hőmérséklet- és páratartalom-érzékelő

- Lehetővé teszi a levegő hőmérsékletének és páratartalmának mérését.
- Digitális kimenetet biztosít, amelyet könnyen feldolgozhat az Arduino.



(A szenzor tüskéinek kimenete)

Talajnedvesség-érzékelő

- A talaj nedvességtartalmát méri, amely segíthet az automatikus öntözés szabályozásában.
- Analog kimenetet ad, amely az arduino egyik analog tüskéjével fogadható.

(Az érzékelő tüskéinek funkciója)

Fényszint érzékelő LDR

- Fotoellenállás, az ellenállás mértéke a rásugárzó fény intenzitásától függ.
- Az elleállás az arduino egyik analog bemenetével olvasható.

Kijelző: OLED SSD1306

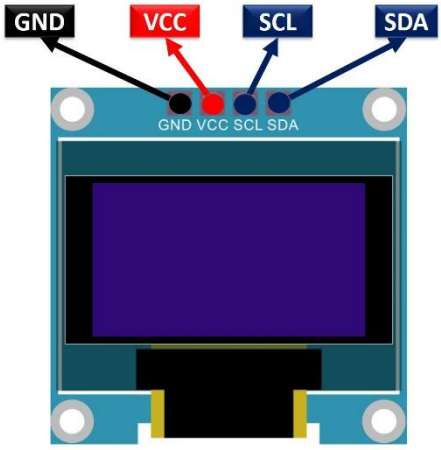
A **0.96 colos OLED kijelző (SSD1306)** segítségével a mért adatok vizuálisan is megjeleníthetők, menüpontokra bontva . Az Adafruit SSD1306 és GFX könyvtárak használatával egyszerűen lehet szöveget és grafikus elemeket (például ikonokat vagy diagramokat) kirajzolni.

Az **OLED (Organic Light Emitting Diode)** technológiát használó **SSD1306** típusú kijelző egy alacsony fogyasztású, nagy kontrasztú megjelenítő, amelyet gyakran használnak beágyazott rendszerekben, mint például Arduino vagy ESP8266 mikrokontrollerekkel. A kijelző leggyakoribb változata **0,96 hüvelyk** átmérőjű, **128x64 pixeles** felbontással rendelkezik, és egy **SSD1306** jelű vezérlőchipet tartalmaz, amely az OLED mátrix működtetéséért felel. Ez a vezérlő támogatja az **I2C** és **SPI** kommunikációs protokollokat is, ami egyszerű csatlakoztatást és gyors adatátvitelt tesz lehetővé.

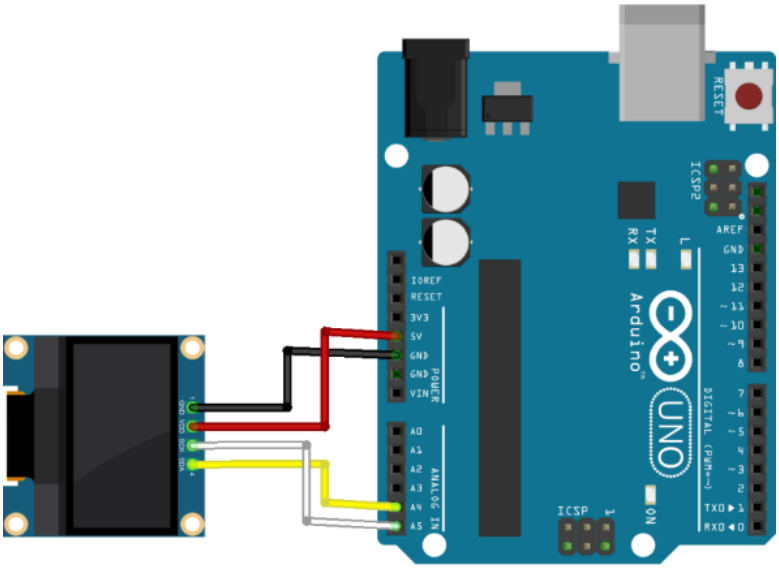
A kijelzők előnyei közé tartozik a nagy fényerejű, kiváló kontrasztú kép, mivel minden pixel önálló fényt bocsát ki, így nincs szükség háttérvilágításra. Kis méretük és alacsony energiaigényük miatt ideálisak mobil és IoT alkalmazásokhoz, például szenzoradatok vagy egyszerű menük megjelenítésére.

A következő ábrákon látható az arduinoval való kommunikációhoz szükséges csatlakozások. Az első ábrán (Ábra 1) látható egy arduino uno-val létrehozott kapcsolat, I2C kommunikációs protokolt használva. A második ábrán (Ábra 2) a kijelző tüskéinek a funkciója látható (szintén I2C kommunikációs protokol hassználatával). A harmadik ábrán (Ábra 3) szintén a tüskék funkciója látható, viszont egy SCA kommunikációt használó kijelzővel.

Ábra 3.



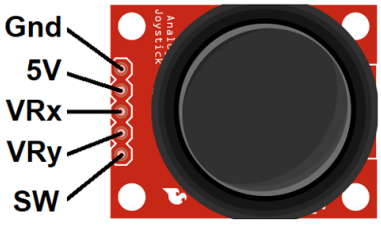
Ábra 2.

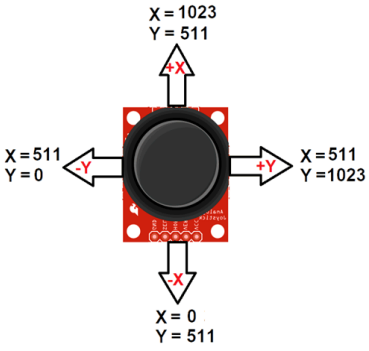


Ábra 1.

Joystick modul

- A projektben egy **két tengelyes** analóg joystick modult használok, amely lehetővé teszi a felhasználó számára, hogy fizikai bemenetet adjon a rendszernek.
- Ez a modul két potenciométerből és egy nyomógombból áll, így X-Y irányú mozgás érzékelésére és kattintásra is képes.
  Az alábbi képeken láthatóak a modul csatlakozói, illetve, az elmozdulás viszonyában visszatérített értékek .



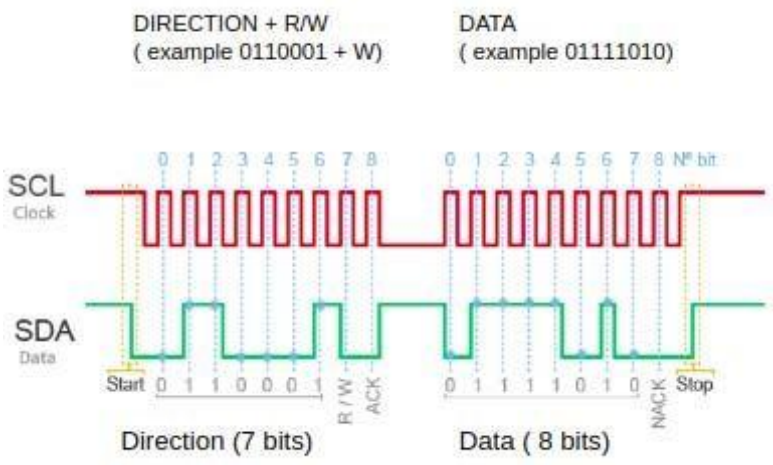


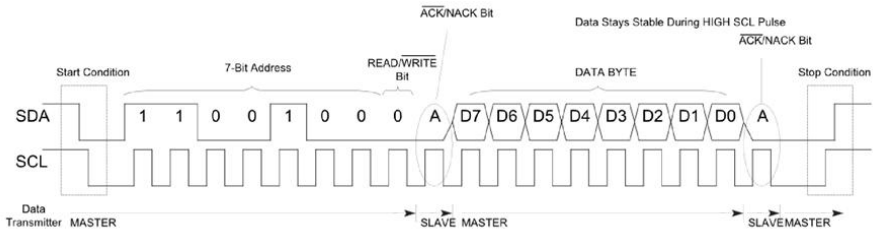
Protokolok

A projektben különböző kommunikációs protokollokat használnak az eszközök közötti adatátvitel megvalósítására. Ezek a protokollok lehetővé teszik a szenzorok, kijelzők és az ESP8266 Wi-Fi modul kommunikációját az Arduino-val.

1. I²C (Inter-Integrated Circuit) protokoll
    - Az **I²C egy soros kommunikációs protokoll**, amelyet főként érzékelők, kijelzők és más perifériák csatlakoztatására használnak. Két vezetékes rendszer, amely egy **órajelvonalból (SCL)** és egy **adatvonalból (SDA)** áll.
    - Az I²C jellemzően kis távolságú, viszonylag alacsony sebességű IC és fedélzeti rendszerek közötti kommunikációra szokás alkalmazni
    - Az I²C (Inter-Integrated Circuit) egy multi-master, multi-slave, csomagkapcsolt soros busz, melyet a Philips Semiconductor (ma NXP Semiconductors) fejlesztett ki.
    - Ebben a projektben az I2C protokoll a kijelzővel való kommunikációra van használva, de mint fejlesztési lehetőség bővíthető egy RTC modullal, az idő valós idejű nyomon követesére.

Az I2C protokoll ábrákon való bemutatása

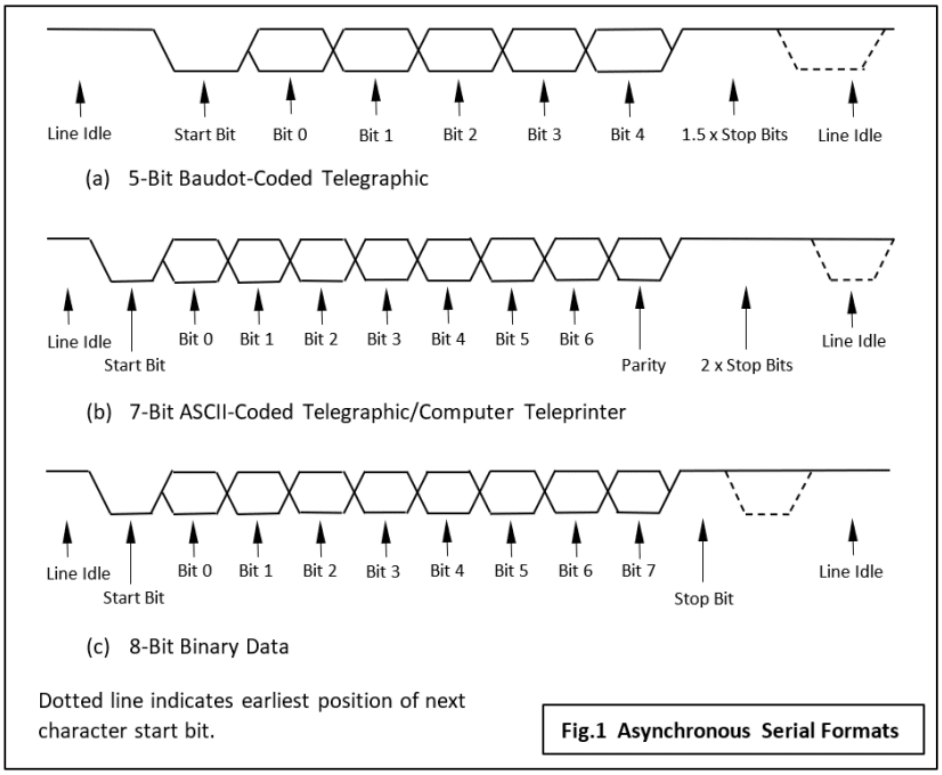




2. UART (Universal Asynchronous Receiver-Transmitter) – Soros kommunikáció
    - Az univerzális aszinkron adóvevő (röviden UART) egy olyan hardver, amely fordítást végez a soros és párhuzamos interfészek között.

- Az UART protokoll az egyik leggyakrabban használt kommunikációs módszer mikrokontrollerek és külső modulok között. Két vezetékes rendszeren alapul: TX-Transmitter – Adó, RX-Receiver – Vevő
- A projektemben az arduino és az esp8266 van sorosan összekapcsolva

Az UART protokoll ábrán



3. AT protokol
    - A protokoll segítségével egy mikrokontroller vagy számítógép **soros (UART) kommunikáción keresztül** adhat utasításokat egy eszköznek, például egy **ESP8266 Wi-Fi modulnak**.
    - Az **AT-parancsok** (vagy **Hayes AT-parancsok**) egy szabványos **modemvezérlő parancskészlet**, amelyet eredetileg a **Hayes Microcomputer Products** fejlesztett ki az 1980-as években. Az **"AT"** rövidítés az "attention" (figyelem) szóból származik, mivel minden parancs az "AT" előtaggal kezdődik, jelezve a modem számára, hogy egy új utasítás következik.
    - Példák az at parancsok használatára esp-vel:
        - AT+CWJAP="SSID","PASSWORD" – Wi-Fi hálózatra csatlakozás (ESP8266).
        - AT+CIFSR-Ip cim lekérdezése.
        - AT+CIPSTART="TCP","192.168.1.50",8080-szerverhez való csatlakozás.

Használt programozási környezet

A projekt megvalósításához az **Arduino programozási nyelvet** alkalmaztam, amely a C és C++ nyelveken alapul, de leegyszerűsített szintaxissal rendelkezik.

A programozás során az **Arduino natív függvényeire** támaszkodtam, például:

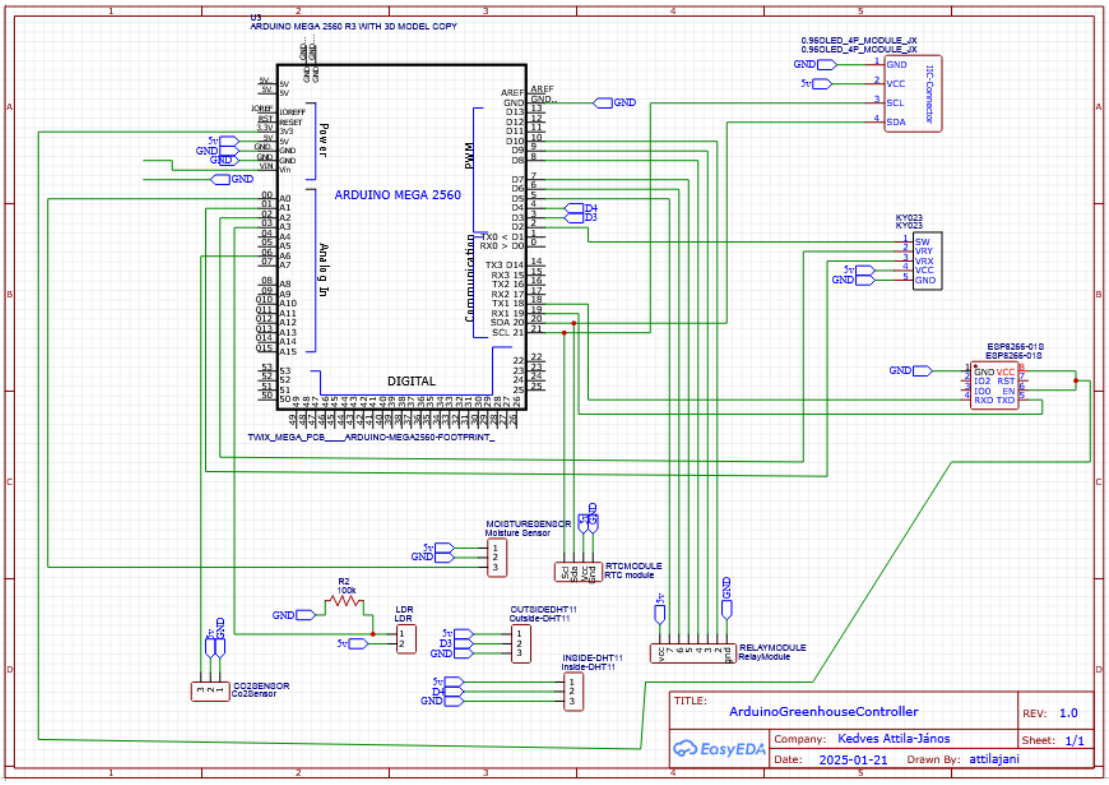
1. **pinMode(pin, mode)**
   Egy adott digitális láb bemenetként (INPUT), kimenetként (OUTPUT), vagy bemenetként belső felhúzó ellenállással (INPUT_PULLUP) történő beállítása.
   Konkrét példa:
   - pinMode(13, OUTPUT); // D13 láb kimenetként beállítása (pl. LED vezérléséhez)
   - pinMode(2, INPUT); // D2 láb bemenetként beállítása (pl. gombhoz)
   - pinMode(3, INPUT_PULLUP); // D3 bemenet, belső felhúzó ellenállással
2. digitalWrite(pin, value)
   Egy **digitális kimenetre** (HIGH vagy LOW) jelet küld.
   Konkrét példa:
   - digitalWrite(13, HIGH); // Bekapcsolja a D13-ra kötött LED-et
   - digitalWrite(13, LOW); // Kikapcsolja a D13-ra kötött LED-et
3. digitalRead(pin)
   Egy digitális bemenet állapotát (HIGH vagy LOW) olvassa be.
4. **analogWrite(pin, value)**
   Egy **PWM (Pulse Width Modulation) jelet** generál a megadott kimeneten. Az érték **0 és 255** között lehet. Csak PWM-képes lábakon működik.
   Konkrét példa: analogWrite(9, 128); // A D9 lábon 50%-os kitöltési tényezőjű PWM jel
5. analogRead(pin)
   Egy **analóg bemenet** értékét olvassa be **0 és 1023** között.
   Konkrét példa: int Hömérséklet = analogRead(A0); // Az A0 láb feszültségét olvassa be (0-1023)
6. delay(ms)
   Egy megadott **ezredmásodpercig (ms) megállítja a program futását**.
   Konkrét példa: delay(1000); // 1 másodperces késleltetés
7. millis()
   A program indítása óta eltelt időt adja vissza **ezredmásodpercben**.
   Konkrét példa: unsigned long inditasotaElteltIdo = millis();
8. map(value, fromLow, fromHigh, toLow, toHigh)
   Egy számot egyik tartományból a másikba skáláz át.
   Konkrét példa: int szenzor = analogRead(A0);
   int skálázva = map(szenzor, 0, 1023, 0, 255);

Rendszerterv és kapcsolási rajzok

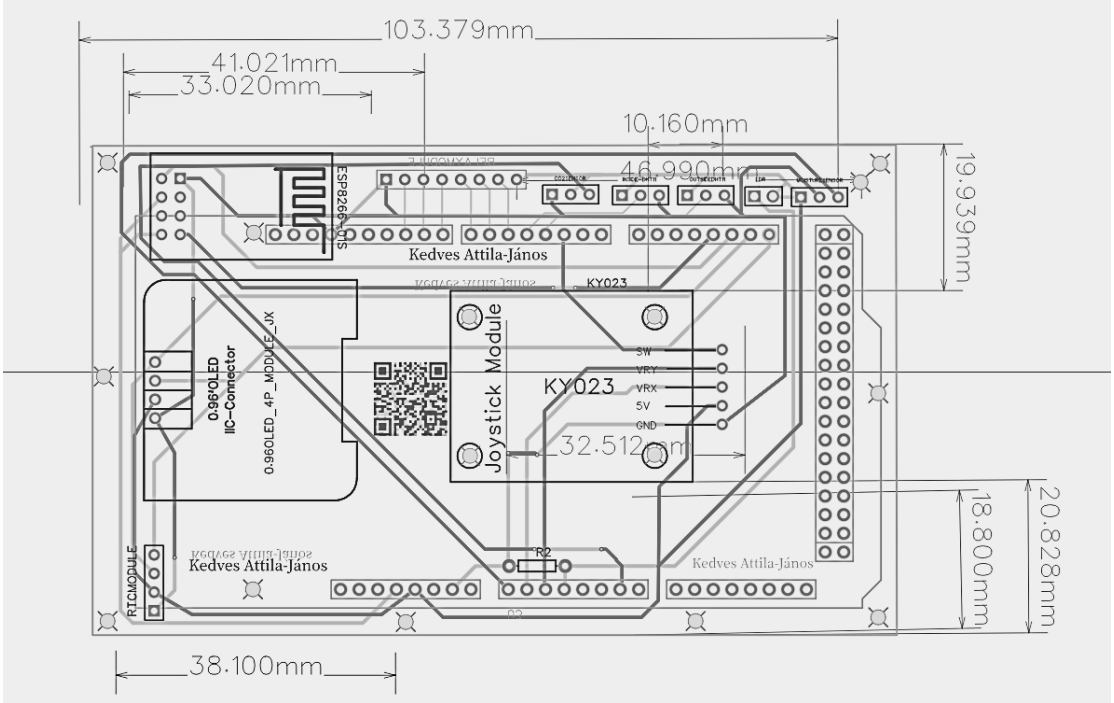
Áttekintés

A rendszer célja a melegház környezeti paramétereinek monitorozása és vezérlése.Az arduino mega fogadja az adatokat a szenzoroktól, a megadott paraméterek alapján vezérli a rellé-modult. Az arduino az esp8266 wifi modulal kommunikálva elküldi az adatokat. Az arduino I2C kommunikációval vezérli az oled kijelzöt. A joystick modul megadja felhasználó által bevitt vezérlést, két analóg bemeneten keresztül és egy digitális, gombon keresztül.

Kapcsolási Rajz



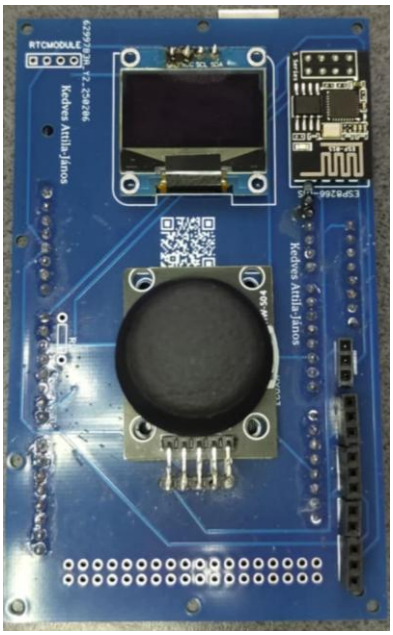
Áramkör

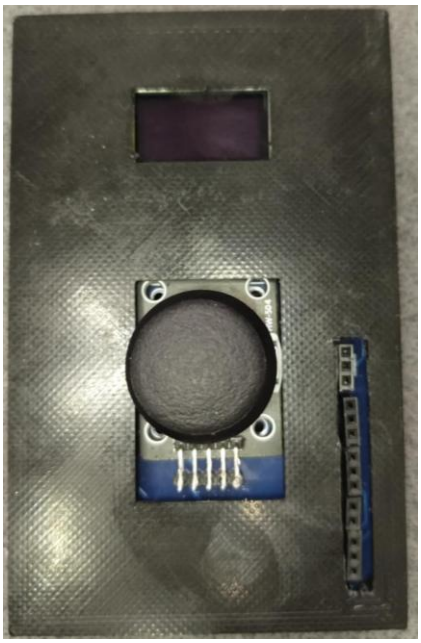


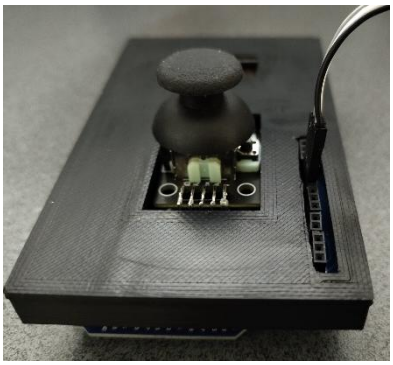
Kötözési táblázat

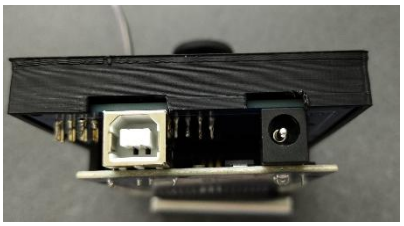
Arduino Mega 2560 Pin	Csatlakoztatott Eszköz	Kapcsolat típusa	Megjegyzés
3V	ESP8266	Tápellátás	Feszültség
5V	Több modul (pl. OLED, DHT)	Tápellátás	Feszültség
GND	Több modul (az összes)	Földelés	Közös GND
D2	Nyomógomb	Digitális bemenet	Gomb
D3	DHT11	Digitális bemenet	Hőmérséklet/páratartalom
D4	DHT11	Digitális bemenet	Hőmérséklet/páratartalom
D5-D10	Relé modul	Digitális kimenet	Relé vezérlés
Serial1 RX	ESP8266 TX	Soros kommunikáció	ESP TX – Mega RX
Serial1 TX	ESP8266 RX	Soros kommunikáció	ESP RX – Mega TX
A1	Joystick modul X-tengely	Analóg bemenet	X mozgás
A2	Joystick modul Y-tengely	Analóg bemenet	Y mozgás
A3	Fény szenzor	Analóg bemenet	Fényérékeny ellenállás
A6	C02 szenzor	Analóg bemenet	Széndioxid szint
SDA (A4)	OLED kijelző, I2C modul	I2C kommunikáció	Adatvonal
SCL (A5)	OLED kijelző, I2C modul	I2C kommunikáció	Óravonal

A kész vezérlő









Forráskód

```
1. //A használt könyvtárak beillesztése
```

```
2. #include <SPI.h>
3. #include <Wire.h>
4. #include <Adafruit_GFX.h>
5. #include <Adafruit_SSD1306.h>
6. #include <dht.h>
7. #include <SoftwareSerial.h>
8. #include <Vector.h>
9. #include <avr/wdt.h>
10.
```

```
11. //Az eszközök csatlakoztatott tüskéi és az állandó értékek megadása, illetve a használandó objektumok deklarálása
```

```
12. //Temp and humidity
13. #define dht1 3
14. #define dht2 4
15. // A két DHT szenzor objektumok deklarálása
16. dht DHT1;
17. dht DHT2;
18. //Moisture sensor
19. #define MOISTURESENSOR A0
20. #define MINMOISTURE 500
21. #define MAXMOISTURE 180
22. //Output relays
23. #define FANPIN 7
24. #define IRRIGATIONPIN 8
25. #define SPRAYPIN 9
26. #define HEATERPIN 10
27. #define LIGHTPIN 6
28. // Az oled kijelző deklarálása i2c kommunikációval
29. #define SCREEN_WIDTH 128
30. #define SCREEN_HEIGHT 64
31. #define OLED_RESET     -1
32. #define SSD1306_I2C_ADDRESS 0x3C
33. Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
34.
35. //Joystick
36. #define JOYSTICK_X A1
37. #define JOYSTICK_Y A2
38. #define JOYSTICK_BUTTON 2
39. #define JOYSTICK_MAXTRESHOLD 900
40. #define JOYSTICK_MINTRESHOLD 100
41.
42. #define MAX_MENU_COUNT 7 // a menupontok 0-tol vannak indexelve, igy ez a szam 1-el kissebb
43. #define MIN_CHANGE_SPEED 0
44. #define MAX_CHANGE_SPEED 100
45. #define SERIALPRINTINTERVALL 5000
46. #define DEBUG 0
47. #define NETWORKTYPE 0 //0-STA, 1 AP
48. #define SERIALNUMBER 123456
49. #define WIFINAME "DIGI_57c120"
50. #define PASSWORD "195da83e"
51. #define AITHINKER "AI-THINKER_394EDC"
```

```
52. // Bemeneti és kimeneti változók deklarálása:
```

```
53.   // Bemeneti változók (érzékelők adatai):
54.     byte insideTemperature, outsideTemperature, insideHumidity, outsideHumidity, soilMoistureLevel, lightIntensity;
55.   // Kimeneti változók (vezérelt eszközök):
56.     enum DeviceState
57.     {
58.       OFF,   // 0
59.       ON,    // 1
60.       AUTO   // 2
```

```
    };
    DeviceState fanState, irrigationState, sprayState, heaterState, lightState;

//Belső változók deklarálása:
  //wifi
  String ssid = String(WIFINAME);     // A wifi neve AI-THINKER_394EDC
  String password = String(PASSWORD);     // A wifi hálózat jelszava
  String ipAddress = "";
  int joystickAxisX, joystickAxisY; //A beviteli eszkoz (joystick) tengelyekre vonatkozo valtozoi
  bool joystickButtonState;
  byte currentMenuNumber;
  byte currentSubmenuNumber;
  byte maxMenuCount;
  bool isSubMenu;
  long long lastTimeMenuRefresh;
  // Button lenyomas kozotti eltelt ido vizsgalata
  const byte menuChangeIntervall = 500; // menu/submenu valtozas
  long long lastTimeChange = 0;// Left/Right valtozas kozott eltelt ido vizsgalata
  long long lastTimeSerial = 0;
  const char deviceStateStrings[3][4] = {"OFF", "ON","AUTO"}; // kiirashoz a konnyebben erthetosegert kiirando szoveg
  //Min és max célváltozók deklarálása:
  byte maxTemp, minTemp, maxHumidity, minHumidity, maxMoisture, minMoisture, minLightLevel;
  const char* languages[3] = {"Eng","Hun","Ro"};
  byte languageSet;
  bool enableSerial = DEBUG;
  bool networkType = NETWORKTYPE;
  byte changeSpeed = 20;
  byte connectionId;
//A fömenü értékeinek flash memóriában való tárolása
  const char menu1[] PROGMEM = "Device Owerview";
  const char menu2[] PROGMEM = "Sensor Overview";
  const char menu3[] PROGMEM = "Temperature Controll";
  const char menu4[] PROGMEM = "Humidity Controll";
  const char menu5[] PROGMEM = "Irrigation Controll";
  const char menu6[] PROGMEM = "Light Controll";
  const char menu7[] PROGMEM = "System Settings";
  const char menu8[] PROGMEM = "Network Settings";
  const char* const messages[] PROGMEM = {menu1,menu2,menu3,menu4,menu5,menu6,menu7,menu8};
```

```
//Fuggvények
```

```
void readSensor()//A szenzorok által mért értékek olvasása
{
  int insideSensor = DHT1.read11(dht1);
  insideTemperature = DHT1.temperature;
  insideHumidity = DHT1.humidity;
  int outsideSensor = DHT2.read11(dht2);
  outsideTemperature = DHT2.temperature;
  outsideHumidity = DHT2.humidity;
  soilMoistureLevel = map(analogRead(MOISTURESENSOR), MINMOISTURE, MAXMOISTURE, 0, 100);
}

void restartArduino() // az arduino ujraindíndítása
{
  wdt_enable(WDTO_15MS); // Watchdog aktiválása 15ms időre
  while (1);             // Várakozás az újraindításra/Lefagyasztjuk a programot
}
void espSetup() // az esp inicializálása
{
  sendData("AT+RST\r\n",2000,enableSerial); // reset module
  if(networkType)
  {
    ssid = String(AITHINKER);
    password = "-";
    sendData("AT+CWMODE=2\r\n",1000,enableSerial); // configure as access point
```

```
  }
  else
  {
    ssid = String(WIFINAME);
    password = String(PASSWORD);
  delay(1000); // kis extra várakozás
  //configure as station mode, connect to wifi
  sendData("AT+CWMODE=1\r\n", 1000, enableSerial);
  // Connect to wifi (ssid+password)
  String connectCommand = "AT+CWJAP=\"" + ssid + "\",\"" + password + "\"\r\n";
  sendData(connectCommand, 8000, enableSerial);
  }
  sendData("AT+CIPMUX=1\r\n",1000,enableSerial); // configure for multiple
connections
  sendData("AT+CIPSERVER=1,80\r\n",1000,enableSerial); // turn on server on port
80
}
void getIp() // Az ip cím lekérdezése
{
  delay(1000);
  // IP cím lekérése
  ipAddress = sendData("AT+CIFSR\r\n", 1000, enableSerial);
  byte first = ipAddress.indexOf('"');
  byte second = ipAddress.indexOf('"', first + 1);
  if (first != -1 && second != -1)
  {
  ipAddress = ipAddress.substring(first + 1, second);
  } else
  {
    Serial.println("ip error");
  }
}
String booleanStateOnOff(bool state) // bool stringként való visszatérítése
kiiratás céljából
{
  if(state)
    return "On";
  else
    return "Off";
}
String sendData(String command, const int timeout, boolean debug) //Parancsok
küldése az esp-nek, és a válaszának fogadása
{
    String response = "";
    Serial1.print(command);
    long int time = millis();
    while( (time+timeout) > millis())
    {
      while(Serial1.available())
        {
            char c = Serial1.read(); // read the next character.
            response+=c;
        }
    }
    if(debug)
          {
          Serial.print(response); //displays the esp response messages in arduino
Serial monitor
          }
    return response;
}
void espsend(String date) //Szoveg parancsának felépitése és elkuldése
{
  String cipSend = " AT+CIPSEND=";
  cipSend += connectionId;
  cipSend += ",";
  cipSend +=date.length();
  cipSend +="\r\n";
  sendData(cipSend,1000,enableSerial);
  sendData(date,1000,enableSerial);
```

```
189. }
190. void showMessageOnScreen(String message) // Szöveg megjelenítése a kijelző közepén
191. {
192.   display.clearDisplay();
193.   display.setTextSize(1);      // Szöveg méretének meghatározása
194.   display.setTextColor(WHITE);
195.   display.setCursor(0, 10);    // Szöveg pozíciójának megadása
196.   display.println(message);
197.   display.display();
198. }
199. void transmitDateOnEsp(String data)// Az esp-vel való kommunikáció kezelése
200. {
201.   if(Serial1.available())
202.   {
203.     if(Serial1.find("+IPD,")) //kérés érkezésének figyelése
204.     {
205.      showMessageOnScreen("Network communication\nin progress...");
206.      delay(300);
207.      connectionId = Serial1.read()-48;
208.       String webpage = data;
209.       webpage+=connectionId;
210.       espsend(webpage);
211.      }
212.      String closeCommand = "AT+CIPCLOSE=";  ////////////////close the socket
connection////esp command
213.      closeCommand+=connectionId; // append connection id
214.      closeCommand+="\r\n";
215.      sendData(closeCommand,3000,enableSerial);
216.   }
217. }
218. String constructDateString() //A hálózatra kiküldött string felépítése a vezérlő
adatai alapján
219. {
220.   String dataString = "InTemp=" + String(insideTemperature) + "&OutTemp=" +
String(outsideTemperature) + "&InHum=" + String(insideHumidity) + "&OutHum=" +
String(outsideHumidity) + "&Moi=" + String(soilMoistureLevel) + "&Light=" +
String(lightIntensity) + "&maxTemp=" + String(maxTemp) + "&minTemp=" + String(minTemp) +
"&maxHum=" + String(maxHumidity) + "&minHum=" + String(minHumidity) + "&maxMoi=" +
String(maxMoisture) ;
221.   dataString +="&minMoi=" + String(minMoisture) + "&minLightLevel=" +
String(minLightLevel)+"&fanState=" + String(fanState) + "&irrigationState=" +
String(irrigationState) + "&sprayState=" + String(sprayState) + "&heaterState=" +
String(heaterState) + "&lightState=" + String(lightState);
222.   String html = "<!DOCTYPE html><html><head><meta http-equiv=\\\"refresh\\\"
content=\\\"5\\\"><title>Greenhouse" + String(SERIALNUMBER) + "</title></head><body>" +
dataString + "</body></html>";
223.
224.   return html;
225. }
226. void outRelays() // A rellé modulok be- és kikapcsolása az állapotuk
alapjánminLightLevel
227. {
228.   if(heaterState == 2)
229.   {
230.     if(insideTemperature < minTemp)
231.       digitalWrite(HEATERPIN, HIGH);
232.     if(insideTemperature >(minTemp+maxTemp)/2)
233.       digitalWrite(HEATERPIN, LOW);
234.   }
235.     if (heaterState == 1)
236.       digitalWrite(HEATERPIN, HIGH);
237.     if (heaterState == 0)
238.       digitalWrite(HEATERPIN, LOW);
239.   if(fanState == 2)
240.   {
241.     if(insideTemperature > maxTemp)
242.       digitalWrite(FANPIN, HIGH);
243.     if(insideTemperature < (minTemp+maxTemp)/2)
244.       digitalWrite(FANPIN, LOW);
245.   }
```

```
246.   if(fanState == 1)
247.       digitalWrite(FANPIN, HIGH);
248.   if(fanState == 0)
249.       digitalWrite(FANPIN, LOW);
250.   if(irrigationState == 2)
251.   {
252.     if(soilMoistureLevel < minMoisture)
253.       digitalWrite(IRRIGATIONPIN, HIGH);
254.     if(soilMoistureLevel > maxMoisture)
255.       digitalWrite(IRRIGATIONPIN, LOW);
256.   }
257.   if(irrigationState == 1)
258.       digitalWrite(IRRIGATIONPIN, HIGH);
259.   if(irrigationState == 0)
260.       digitalWrite(IRRIGATIONPIN, LOW);
261.   if(lightState == 2)
262.   {
263.     if(lightIntensity < minLightLevel)
264.       digitalWrite(LIGHTPIN, HIGH);
265.     if(lightIntensity>=minLightLevel)
266.       digitalWrite(LIGHTPIN,LOW);
267.   }
268.   if(lightState == 1)
269.       digitalWrite(LIGHTPIN, HIGH);
270.   if(lightState == 0)
271.       digitalWrite(LIGHTPIN, LOW);
272.   switch (sprayState)
273.   {
274.     case 0:digitalWrite(SPRAYPIN, LOW);break;
275.     case 1:digitalWrite(SPRAYPIN, HIGH);break;
276.     case 2:
277.     if(insideHumidity < minHumidity)
278.       digitalWrite(SPRAYPIN, HIGH);
279.     if(insideHumidity > (minHumidity+maxHumidity)/2)
280.       digitalWrite(SPRAYPIN, LOW);
281.       break;
282.   }
283.
284. }
285. short languageChooser (short currentLanguage)// A nyelvválasztó vátltozó
változtatása
286. {
287.     if (joystickAxisY > JOYSTICK_MAXTRESHOLD)
288.   {
289.     currentLanguage ++;
290.     if(currentLanguage > 2)
291.       currentLanguage = 0;
292.   }
293.   else
294.   {
295.     if (joystickAxisY < JOYSTICK_MINTRESHOLD)
296.     {
297.       currentLanguage --;
298.       if(currentLanguage < 0)
299.         currentLanguage = 2;
300.
301.     }
302.
303.   }
304.   delay(changeSpeed+100);
305.   return currentLanguage;
306. }
307. void changeSubmenuBoolean(bool &booleanToChange)// Bool menüváltozó
megváltoztatása, joystickAxisY alapján
308. {
309.   if (joystickAxisY > JOYSTICK_MAXTRESHOLD || joystickAxisY <
JOYSTICK_MINTRESHOLD)
310.   {
311.     booleanToChange = !booleanToChange;
312.   }
```

```
313.   delay(changeSpeed+30);
314. }
315. short changeSubmenuVariable(short numberToChange) // A számértékek változtatása az almenükben
316. {
317.
318.   if (joystickAxisY > JOYSTICK_MAXTRESHOLD)
319.   {
320.     numberToChange++;
321.   }
322.   else
323.   {
324.     if (joystickAxisY < JOYSTICK_MINTRESHOLD)
325.     {
326.       numberToChange--;
327.     }
328.   }
329.   delay(changeSpeed+30);
330.   return numberToChange;
331.
332. }
333. DeviceState changeDeviceState(DeviceState deviceState)// Az eszközök állapotának változtatása
334. {
335.   short counter = deviceState;
336.
337.   if (joystickAxisY > JOYSTICK_MAXTRESHOLD)
338.   {
339.     counter ++;
340.     if(counter > 2)
341.       counter = 0;
342.   }
343.   else
344.   {
345.     if (joystickAxisY < JOYSTICK_MINTRESHOLD)
346.     {
347.       counter --;
348.       if(counter < 0)
349.         counter = 2;
350.     }
351.     else return static_cast<DeviceState>(counter);
352.   }
353.   delay(changeSpeed+100);
354.   return static_cast<DeviceState>(counter);
355. }
356. void changeMenuNumber(bool isSubMenu, byte maxNumber)// A menü és az almenü értékének növelése/csökkentése
357. {
358.   if(millis()-lastTimeMenuRefresh > changeSpeed+50)//Módosítások közötti idő vizsgálata, a megfelő működésért
359.   {
360.       lastTimeMenuRefresh = millis();
361.     if(isSubMenu) // dontés, hogy a főmenü, vagy az almenü értékét kell-e változtatni
362.     {
363.       if (joystickAxisX > JOYSTICK_MAXTRESHOLD)
364.       {
365.         if(maxNumber > currentSubmenuNumber)
366.           currentSubmenuNumber++;
367.         else
368.           currentSubmenuNumber = 0;
369.
370.       }
371.     else
372.     {
373.       if (joystickAxisX < JOYSTICK_MINTRESHOLD)
374.       {
375.         if(currentSubmenuNumber > 0)
376.           currentSubmenuNumber--;
377.         else
```

```
        currentSubmenuNumber = maxNumber;

      }
    }
    }
    else
    {
        if (joystickAxisX > JOYSTICK_MAXTRESHOLD)
      {
        if(maxNumber > currentMenuNumber)
          currentMenuNumber++;
        else
          currentMenuNumber = 0;

      }
    else
    {
      if (joystickAxisX < JOYSTICK_MINTRESHOLD)
      {
        if(currentMenuNumber > 0)
          currentMenuNumber--;
        else
        currentMenuNumber = maxNumber;

      }
    }
    }
  }
}
void readJoystickValues()// A joystick modul állapotának olvasása
{
  joystickAxisX = 1024-analogRead(JOYSTICK_X);
  joystickAxisY = analogRead(JOYSTICK_Y);
  joystickButtonState = !digitalRead(JOYSTICK_BUTTON);
}
void serialMonitorPrint() // Adatok kikuldese sorosan tesytelés céljából
  {
    if(millis()-lastTimeSerial > SERIALPRINTINTERVALL)
    {
      lastTimeSerial = millis();
      Serial.println();
      Serial.println(millis());
      Serial.print(F("Kimeneti változók:")); // Az F() szükséges, ugyanis az SRAM
korlatozott, es igy a Flash memoriat terheli
      Serial.print(F(" fanState: "));
      Serial.print(fanState);
      Serial.print(F(" irrigationState: "));
      Serial.print(irrigationState);
      Serial.print(F(" sprayState: "));
      Serial.print(sprayState);
      Serial.print(F(" fanState: "));
      Serial.print(fanState);
      Serial.print(F(" lightState: "));
      Serial.println(lightState);
      Serial.println();
      Serial.print(F("Szenzor értékek: "));
      Serial.print(F(" insideTemperature: "));
      Serial.print(insideTemperature);
      Serial.print(F(" outsideTemperature: "));
      Serial.print(outsideTemperature);
      Serial.print(F(" insideHumidity: "));
      Serial.print(insideHumidity);
      Serial.print(F(" outsideHumidity: "));
      Serial.print(outsideHumidity);
      Serial.print(F(" soilMoistureLevel: "));
      Serial.print(soilMoistureLevel);
      Serial.print(F(" lightIntensity: "));
      Serial.println(lightIntensity);
      Serial.println();
      Serial.print(F("Joystick értékek: "));
```

```
447.       Serial.print(F(" joystickAxisX: "));
448.       Serial.print(joystickAxisX);
449.       Serial.print(F(" joystickAxisY: "));
450.       Serial.print(joystickAxisY);
451.       Serial.print(F(" joystickButtonState: "));
452.       Serial.println(joystickButtonState);
453.       Serial.println();
454.       Serial.print(F("Menu value: "));
455.       Serial.print(currentMenuNumber);
456.       Serial.print(F("Submenu value: "));
457.       Serial.print(currentSubmenuNumber);
458.       Serial.print(F("isSubmenu: "));
459.       Serial.print(isSubMenu);
460.       Serial.println("\n-----------------------------------");
461.
462.     }
463.
464.   }
465. void killAllDevices()//Minden eszköz OFF-ra állítása, és minden rellélekapcsolása
466. {
467.   //az eszkök állapotát off-ra állítani
468.   fanState = OFF;
469.   irrigationState = OFF;
470.   sprayState = OFF;
471.   heaterState = OFF;
472.   lightState = OFF;
473.   //megadni a parancsot a relléknek
474.   digitalWrite(FANPIN, LOW);
475.   digitalWrite(IRRIGATIONPIN, LOW);
476.   digitalWrite(SPRAYPIN, LOW);
477.   digitalWrite(HEATERPIN, LOW);
478.   digitalWrite(LIGHTPIN, LOW);
479. }
480. void displayInitialize()//Kezdőképernyő mutatása
481. {
482.   if(!display.begin(SSD1306_SWITCHCAPVCC, SSD1306_I2C_ADDRESS)) {
483.     Serial.println(F("SSD1306 allocation failed"));
484.     for(;;); // Nem sikerült létrehozni a kapcsolatot, végtelen ciklus
485.   }
486.   display.clearDisplay();// Clear the buffer
487.
488.   display.setTextSize(1);      // Szöveg méretének meghatározása
489.   display.setTextColor(WHITE);
490.   display.setCursor(0, 0);   // Szöveg pozíciójának megadása
491.   display.println(F("Kedves Attila Janos"));
492.   display.println("Github:");
493.   display.println(F("https://github.com/KedvesAttilaJanos"));
494.   display.display();
495.   delay(5000);
496.   showLoadingScreen();
497. }
498. void loaderSwitch(byte loadNum) // A betöltő képernyőnek megfelelően, a setup
feladatok végrehalytási sorrendjének meghatározása
499. {
500.   switch(loadNum)
501.   {
502.     case 30: espSetup(); Serial.println(F("Esp setuped")); break;
503.     case 50: getIp(); Serial.println(F("Ip get")); break;
504.     case 60: killAllDevices(); Serial.println(F("Devices off")); break;
505.     case 70: zeroAllSensor(); Serial.println(F("Sensors Zero")); break;
506.     case 80: zeroAllInternalVariable(); Serial.println(F("Internal variables
zero")); break;
507.     case 90: readSensor(); Serial.println(F("First read")); break;
508.   }
509. }
510. bool joystickAxisYMoved(String direction) //overload, paraméterként megadható,
hogy merre nézze a joystick elmozdulását
511. {
512.   if(direction == "right")
513.     return joystickAxisY > JOYSTICK_MAXTRESHOLD;
```

```
  if(direction == "left")
    return joystickAxisY < JOYSTICK_MINTRESHOLD;
}
bool joystickAxisYMoved() // A jobbra/balra történő elmozdulás figyelése
{
  return joystickAxisY < JOYSTICK_MINTRESHOLD || joystickAxisY >
JOYSTICK_MAXTRESHOLD;
}
void showLoadingScreen()//Betöltőképernyő megjelenítése
{
  display.clearDisplay();
  display.setTextSize(1);      // Szöveg méretének meghatározása
  display.setTextColor(WHITE);
  display.setCursor(10, 10);   // Szöveg pozíciójának megadása
  display.println(F("Loading..."));
  display.display();

  // Animációs betöltési sáv
  for (int i = 0; i <= 100; i += 2) {
    display.fillRect(10, 30, i, 10, WHITE); // Sáv rajzolása
    loaderSwitch(i);
    display.display();
    delay(2); // Sáv frissítési sebessége
  }

  // Törlés vagy következő képernyő
  delay(500);
  display.clearDisplay();
  display.display();
}
void zeroAllSensor()//Minden szenzorérték lenullázása
{
  insideTemperature = 0;
  outsideTemperature = 0;
  insideHumidity = 0;
  outsideHumidity = 0;
  soilMoistureLevel = 0;
  lightIntensity = 0;
}
void zeroAllInternalVariable()// A belső vátozók alaphelyzetbe állítása
{
  joystickAxisX = 0;
  joystickAxisY = 0;
  joystickButtonState = 0;
  isSubMenu = false;
  currentMenuNumber = 0;
  currentSubmenuNumber = -1;
  lastTimeMenuRefresh = 0;
  maxMenuCount = MAX_MENU_COUNT;
  maxTemp = 0;
  minTemp = 0;
  maxHumidity = 0;
  minHumidity = 0;
  maxMoisture = 0;
  minMoisture = 0;
  minLightLevel = 0;
  languageSet = 0;
  enableSerial = DEBUG;
}
void printCentered(const String &text, int y)// A címek középreigazított kiírása
{
  int textWidth = text.length() * 6; // 6 pixel széles karakterek
  int x = (SCREEN_WIDTH - textWidth) / 2;
  display.setCursor(x, y);
  display.println(text);
}
void changeSelectedMenuVariable()// A kiválasztott értékek megváltoztatása
{
  if(isSubMenu)
  {
```

```
 583.     switch (currentMenuNumber)
 584.     {
 585.       case 0:
 586.         if (currentSubmenuNumber == 0) fanState = changeDeviceState(fanState);
 587.         if (currentSubmenuNumber == 1) irrigationState =
changeDeviceState(irrigationState);
 588.         if (currentSubmenuNumber == 2) sprayState = changeDeviceState(sprayState);
 589.         if (currentSubmenuNumber == 3) heaterState =
changeDeviceState(heaterState);
 590.         if (currentSubmenuNumber == 4) lightState = changeDeviceState(lightState);
 591.       break;
 592.     }
 593.   }
 594. }
 595. void mainMenuSystem()// A menük kiírása, döntésekkel, hogy az adott menüpontot
háttérrel jelenítse meg
 596. {
 597.   display.setTextSize(1);
 598.   display.clearDisplay();
 599.   short devices[5] {fanState, irrigationState, sprayState, heaterState,
lightState};
 600.   if(!isSubMenu) //ellenörizzük, hogy almenüben, vagy főmnüben vagyunk-e
 601.   {
 602.     display.setTextColor(WHITE);
 603.     display.setCursor(0,0);
 604.     char buffer[20]; // Ide töltjük a PROGMEM-ből
 605.     for (size_t i = 0; i < 8; i++)
 606.     {
 607.       strcpy_P(buffer, (char*)pgm_read_word(&(messages[i])));
 608.       if (currentMenuNumber == i)
 609.         display.setTextColor(BLACK,WHITE);
 610.       else
 611.         display.setTextColor(WHITE);
 612.       display.println(buffer);
 613.     }
 614.       display.display();
 615.
 616.   }
 617.   else
 618.   {
 619.     switch(currentMenuNumber)
 620.     {
 621.       case 0:
 622.         maxMenuCount = 4;
 623.         switch(currentSubmenuNumber)
 624.         {
 625.         case 0:
 626.         printCentered(F("Device Overview"),0);
 627.         display.setTextColor(BLACK,WHITE);
 628.         display.print(F("Fan State        "));
 629.         display.println(deviceStateStrings[fanState]);
 630.         display.setTextColor(WHITE);
 631.         display.print(F("Irrigation State "));
 632.         display.println(deviceStateStrings[irrigationState]);
 633.         display.print(F("Spray State      "));
 634.         display.println(deviceStateStrings[sprayState]);
 635.         display.print(F("Heater State     "));
 636.         display.println(deviceStateStrings[heaterState]);
 637.         display.print(F("Light State      "));
 638.         display.println(deviceStateStrings[lightState]);
 639.         display.display();
 640.         fanState = changeDeviceState(fanState);
 641.         break;
 642.         case 1:
 643.         printCentered(F("Device Overview"),0);
 644.         display.print(F("Fan State        "));
 645.         display.println(deviceStateStrings[fanState]);
 646.         display.setTextColor(BLACK,WHITE);
 647.         display.print(F("Irrigation State "));
 648.         display.println(deviceStateStrings[irrigationState]);
```

```
649.          display.setTextColor(WHITE);
650.          display.print(F("Spray State      "));
651.          display.println(deviceStateStrings[sprayState]);
652.          display.print(F("Heater State     "));
653.          display.println(deviceStateStrings[heaterState]);
654.          display.print(F("Light State      "));
655.          display.println(deviceStateStrings[lightState]);
656.          display.display();
657.          irrigationState = changeDeviceState(irrigationState);
658.          break;
659.          case 2:
660.          printCentered(F("Device Overview"),0);
661.          display.print(F("Fan State        "));
662.          display.println(deviceStateStrings[fanState]);
663.          display.print(F("Irrigation State "));
664.          display.println(deviceStateStrings[irrigationState]);
665.          display.setTextColor(BLACK,WHITE);
666.          display.print(F("Spray State      "));
667.          display.println(deviceStateStrings[sprayState]);
668.          display.setTextColor(WHITE);
669.          display.print(F("Heater State     "));
670.          display.println(deviceStateStrings[heaterState]);
671.          display.print(F("Light State      "));
672.          display.println(deviceStateStrings[lightState]);
673.          display.display();
674.          sprayState = changeDeviceState(sprayState);
675.          break;
676.          case 3:
677.          printCentered(F("Device Overview"),0);
678.          display.print(F("Fan State        "));
679.          display.println(deviceStateStrings[fanState]);
680.          display.print(F("Irrigation State "));
681.          display.println(deviceStateStrings[irrigationState]);
682.          display.print(F("Spray State      "));
683.          display.println(deviceStateStrings[sprayState]);
684.          display.setTextColor(BLACK,WHITE);
685.          display.print(F("Heater State     "));
686.          display.println(deviceStateStrings[heaterState]);
687.          display.setTextColor(WHITE);
688.          display.print(F("Light State      "));
689.          display.println(deviceStateStrings[lightState]);
690.          display.display();
691.          heaterState = changeDeviceState(heaterState);
692.          break;
693.          case 4:
694.          printCentered(F("Device Overview"),0);
695.          display.print(F("Fan State        "));
696.          display.println(deviceStateStrings[fanState]);
697.          display.print(F("Irrigation State "));
698.          display.println(deviceStateStrings[irrigationState]);
699.          display.print(F("Spray State      "));
700.          display.println(deviceStateStrings[sprayState]);
701.          display.print(F("Heater State     "));
702.          display.println(deviceStateStrings[heaterState]);
703.          display.setTextColor(BLACK,WHITE);
704.          display.print(F("Light State      "));
705.          display.println(deviceStateStrings[lightState]);
706.          display.setTextColor(WHITE);
707.          display.display();
708.          lightState = changeDeviceState(lightState);
709.          break;
710.
711.          }
712.        break;
713.        case 1:
714.          printCentered(F("Sensor Overview "),0);
715.          display.print(F("Inside Temp.    "));
716.          display.println(insideTemperature);
717.          display.print(F("Outside Temp.   "));
718.          display.println(outsideTemperature);
```

```
719.          display.print(F("Inside Humidity "));
720.          display.println(insideHumidity);
721.          display.print(F("Outside Humidity"));
722.          display.println(outsideHumidity);
723.          display.print(F("Soil Moisture   "));
724.          display.println(soilMoistureLevel);
725.          display.print(F("Light Level     "));
726.          display.println(lightIntensity);
727.          display.display();
728.        break;
729.        case 2:
730.          maxMenuCount = 3;
731.          switch(currentSubmenuNumber)
732.          {
733.            case 0:
734.              printCentered(F("Temp Controll"),0);
735.              display.setTextColor(BLACK,WHITE);
736.              display.print(F("Max Temp         "));
737.              display.println(maxTemp);
738.              display.setTextColor(WHITE);
739.              display.print(F("Min Temp         "));
740.              display.println(minTemp);
741.              display.print(F("Fan State        "));
742.              display.println(deviceStateStrings[fanState]);
743.              display.print(F("Heater State     "));
744.              display.println(deviceStateStrings[heaterState]);
745.              display.print(F("Inside Temp      "));
746.              display.println(insideTemperature);
747.              display.print(F("Outside Temp     "));
748.              display.println(outsideTemperature);
749.              display.display();
750.              maxTemp = changeSubmenuVariable(maxTemp);
751.            break;
752.            case 1:
753.              printCentered(F("Temp Controll"),0);
754.              display.print(F("Max Temp         "));
755.              display.println(maxTemp);
756.              display.setTextColor(BLACK,WHITE);
757.              display.print(F("Min Temp         "));
758.              display.println(minTemp);
759.              display.setTextColor(WHITE);
760.              display.print(F("Fan State        "));
761.              display.println(deviceStateStrings[fanState]);
762.              display.print(F("Heater State     "));
763.              display.println(deviceStateStrings[heaterState]);
764.              display.print(F("Inside Temp      "));
765.              display.println(insideTemperature);
766.              display.print(F("Outside Temp     "));
767.              display.println(outsideTemperature);
768.              display.display();
769.              minTemp = changeSubmenuVariable(minTemp);
770.            break;
771.            case 2:
772.              printCentered(F("Temp Controll"),0);
773.              display.print(F("Max Temp         "));
774.              display.println(maxTemp);
775.              display.print(F("Min Temp         "));
776.              display.println(minTemp);
777.              display.setTextColor(BLACK,WHITE);
778.              display.print(F("Fan State        "));
779.              display.println(deviceStateStrings[fanState]);
780.              display.setTextColor(WHITE);
781.              display.print(F("Heater State     "));
782.              display.println(deviceStateStrings[heaterState]);
783.              display.print(F("Inside Temp      "));
784.              display.println(insideTemperature);
785.              display.print(F("Outside Temp     "));
786.              display.println(outsideTemperature);
787.              display.display();
788.              fanState = changeDeviceState(fanState);
```

```
789.           break;
790.           case 3:
791.             printCentered(F("Temp Controll"),0);
792.             display.print(F("Max Temp         "));
793.             display.println(maxTemp);
794.             display.print(F("Min Temp         "));
795.             display.println(minTemp);
796.             display.print(F("Fan State        "));
797.             display.println(deviceStateStrings[fanState]);
798.             display.setTextColor(BLACK,WHITE);
799.             display.print(F("Heater State     "));
800.             display.println(deviceStateStrings[heaterState]);
801.             display.setTextColor(WHITE);
802.             display.print(F("Inside Temp      "));
803.             display.println(insideTemperature);
804.             display.print(F("Outside Temp     "));
805.             display.println(outsideTemperature);
806.             display.display();
807.             heaterState = changeDeviceState(heaterState);
808.           break;
809.         }
810.
811.       break;
812.       case 3:
813.         maxMenuCount = 2;
814.         switch(currentSubmenuNumber)
815.         {
816.           case 0:
817.             printCentered(F("Humidity Controll"),0);
818.             display.setTextColor(BLACK,WHITE);
819.             display.print(F("Max Humidity     "));
820.             display.println(maxHumidity);
821.             display.setTextColor(WHITE);
822.             display.print(F("Min Humidity     "));
823.             display.println(minHumidity);
824.             display.print(F("Spray State      "));
825.             display.println(deviceStateStrings[sprayState]);
826.             display.print(F("Inside Humidity  "));
827.             display.println(insideHumidity);
828.             display.print(F("Outside Humidity "));
829.             display.println(outsideHumidity);
830.             display.display();
831.             maxHumidity = changeSubmenuVariable(maxHumidity);
832.           break;
833.           case 1:
834.             printCentered(F("Humidity Controll"),0);
835.             display.print(F("Max Humidity     "));
836.             display.println(maxHumidity);
837.             display.setTextColor(BLACK,WHITE);
838.             display.print(F("Min Humidity     "));
839.             display.println(minHumidity);
840.             display.setTextColor(WHITE);
841.             display.print(F("Spray State      "));
842.             display.println(deviceStateStrings[sprayState]);
843.             display.print(F("Inside Humidity  "));
844.             display.println(insideHumidity);
845.             display.print(F("Outside Humidity "));
846.             display.println(outsideHumidity);
847.             display.display();
848.             minHumidity = changeSubmenuVariable(minHumidity);
849.           break;
850.           case 2:
851.             printCentered(F("Humidity Controll"),0);
852.             display.print(F("Max Humidity     "));
853.             display.println(maxHumidity);
854.             display.print(F("Min Humidity     "));
855.             display.println(minHumidity);
856.             display.setTextColor(BLACK,WHITE);
857.             display.print(F("Spray State      "));
858.             display.println(deviceStateStrings[sprayState]);
```

```
            display.setTextColor(WHITE);
            display.print(F("Inside Humidity  "));
            display.println(insideHumidity);
            display.print(F("Outside Humidity "));
            display.println(outsideHumidity);
            display.display();
            sprayState = changeDeviceState(sprayState);
          break;
        }
      break;
      case 4:
        maxMenuCount = 2;
        switch(currentSubmenuNumber)
        {
          case 0:
            printCentered(F("Irrigation Controll"),0);
            display.setTextColor(BLACK,WHITE);
            display.print(F("Max Moisture     "));
            display.println(maxMoisture);
            display.setTextColor(WHITE);
            display.print(F("Min Moisture     "));
            display.println(minMoisture);
            display.print(F("Irrigation State "));
            display.println(deviceStateStrings[irrigationState]);
            display.print(F("Moisture Level   "));
            display.println(soilMoistureLevel);
            display.display();
            maxMoisture = changeSubmenuVariable(maxMoisture);
          break;
          case 1:
            printCentered(F("Irrigation Controll"),0);
            display.print(F("Max Moisture     "));
            display.println(maxMoisture);
            display.setTextColor(BLACK,WHITE);
            display.print(F("Min Moisture     "));
            display.println(minMoisture);
            display.setTextColor(WHITE);
            display.print(F("Irrigation State "));
            display.println(deviceStateStrings[irrigationState]);
            display.print(F("Moisture Level   "));
            display.println(soilMoistureLevel);
            display.display();
            minMoisture = changeSubmenuVariable(minMoisture);
          break;
          case 2:
            printCentered(F("Irrigation Controll"),0);
            display.print(F("Max Moisture     "));
            display.println(maxMoisture);
            display.print(F("Min Moisture     "));
            display.println(minMoisture);
            display.setTextColor(BLACK,WHITE);
            display.print(F("Irrigation State "));
            display.println(deviceStateStrings[irrigationState]);
            display.setTextColor(WHITE);
            display.print(F("Moisture Level   "));
            display.println(soilMoistureLevel);
            display.display();
            irrigationState = changeDeviceState(irrigationState);
          break;
        }
      break;
      case 5:
        maxMenuCount = 1;
        switch(currentSubmenuNumber)
        {
          case 0:
            printCentered(F("Light Controll"),0);
            display.setTextColor(BLACK,WHITE);
            display.print(F("Min Light Level: "));
            display.println(minLightLevel);
```

```
929.              display.setTextColor(WHITE);
930.              display.print(F("Ligth State      "));
931.              display.println(deviceStateStrings[lightState]);
932.              display.print(F("Ligth Intensity  "));
933.              display.println(lightIntensity);
934.              display.display();
935.              minLightLevel = changeSubmenuVariable(minLightLevel);
936.            break;
937.            case 1:
938.              printCentered(F("Light Controll"),0);
939.              display.print(F("Min Light Level: "));
940.              display.println(minLightLevel);
941.              display.setTextColor(BLACK,WHITE);
942.              display.print(F("Ligth State      "));
943.              display.println(deviceStateStrings[lightState]);
944.              display.setTextColor(WHITE);
945.              display.print(F("Ligth Intensity  "));
946.              display.println(lightIntensity);
947.              display.display();
948.              lightState = changeDeviceState(lightState);
949.            break;
950.
951.          }
952.        break;
953.        case 6:
954.          maxMenuCount = 5;
955.          switch(currentSubmenuNumber)
956.          {
957.            case 0:
958.              printCentered(F("System Setting"),0);
959.              display.setTextColor(BLACK,WHITE);
960.              display.println(F("Restart Sensors "));
961.              display.setTextColor(WHITE);
962.              display.println(F("Restart Devices "));
963.              display.print(F("Language          "));
964.              display.println(languages[languageSet]);
965.              display.print(F("Enable Serial     "));
966.              display.println(booleanStateOnOff(enableSerial));
967.              display.print(F("Change Slowenes   "));
968.              display.println(changeSpeed);
969.              display.println(F("Restart Arduino   "));
970.              display.print(F("Name: "));
971.              display.print(SERIALNUMBER);
972.              display.display();
973.              if(joystickAxisYMoved())
974.              {
975.                zeroAllSensor();
976.                display.clearDisplay();
977.                printCentered(F("Sensors Restarted"),16);
978.                display.display();
979.                delay(500);
980.              }
981.            break;
982.            case 1:
983.              printCentered(F("System Setting"),0);
984.              display.println(F("Restart Sensors "));
985.              display.setTextColor(BLACK,WHITE);
986.              display.println(F("Restart Devices "));
987.              display.setTextColor(WHITE);
988.              display.print(F("Language          "));
989.              display.println(languages[languageSet]);
990.              display.print(F("Enable Serial     "));
991.              display.println(booleanStateOnOff(enableSerial));
992.              display.print(F("Change Slowenes   "));
993.              display.println(changeSpeed);
994.              display.println(F("Restart Arduino   "));
995.              display.print(F("Name: "));
996.              display.print(SERIALNUMBER);
997.              display.display();
998.              if(joystickAxisYMoved("left"))
```

```
 999.             {
1000.               killAllDevices();
1001.               display.clearDisplay();
1002.               printCentered(F("Devices Restarted"),16);
1003.               display.display();
1004.               delay(500);
1005.             }
1006.           break;
1007.           case 2:
1008.             printCentered(F("System Setting"),0);
1009.             display.println(F("Restart Sensors "));
1010.             display.println(F("Restart Devices "));
1011.             display.setTextColor(BLACK,WHITE);
1012.             display.print(F("Language          "));
1013.             display.println(languages[languageSet]);
1014.             display.setTextColor(WHITE);
1015.             display.print(F("Enable Serial     "));
1016.             display.println(booleanStateOnOff(enableSerial));
1017.             display.print(F("Change Slowenes   "));
1018.             display.println(changeSpeed);
1019.             display.println(F("Restart Arduino   "));
1020.             display.print(F("Name: "));
1021.             display.print(SERIALNUMBER);
1022.             display.display();
1023.             languageSet = languageChooser(languageSet);
1024.           break;
1025.           case 3:
1026.             printCentered(F("System Setting"),0);
1027.             display.println(F("Restart Sensors "));
1028.             display.println(F("Restart Devices "));
1029.             display.print(F("Language          "));
1030.             display.println(languages[languageSet]);
1031.             display.setTextColor(BLACK,WHITE);
1032.             display.print(F("Enable Serial     "));
1033.             display.println(booleanStateOnOff(enableSerial));
1034.             display.setTextColor(WHITE);
1035.             display.print(F("Change Slowenes   "));
1036.             display.println(changeSpeed);
1037.             display.println(F("Restart Arduino   "));
1038.             display.print(F("Name: "));
1039.             display.print(SERIALNUMBER);
1040.             display.display();
1041.             changeSubmenuBoolean(enableSerial);
1042.           break;
1043.           case 4:
1044.             printCentered(F("System Setting"),0);
1045.             display.println(F("Restart Sensors "));
1046.             display.println(F("Restart Devices "));
1047.             display.print(F("Language          "));
1048.             display.println(languages[languageSet]);
1049.             display.print(F("Enable Serial     "));
1050.             display.println(booleanStateOnOff(enableSerial));
1051.             display.setTextColor(BLACK,WHITE);
1052.             display.print(F("Change Slowenes   "));
1053.             display.println(changeSpeed);
1054.             display.setTextColor(WHITE);
1055.             display.println(F("Restart Arduino   "));
1056.             display.print(F("Name: "));
1057.             display.print(SERIALNUMBER);
1058.             display.display();
1059.             changeSpeed = changeSubmenuVariable(changeSpeed);
1060.             if(changeSpeed <= MIN_CHANGE_SPEED)
1061.             {
1062.               changeSpeed = MAX_CHANGE_SPEED;
1063.             }
1064.             else if(changeSpeed > MAX_CHANGE_SPEED)
1065.               {
1066.               changeSpeed = MIN_CHANGE_SPEED;
1067.               }
1068.           break;
```

```
1069.           case 5:
1070.             printCentered(F("System Setting"),0);
1071.             display.println(F("Restart Sensors "));
1072.             display.println(F("Restart Devices "));
1073.             display.print(F("Language          "));
1074.             display.println(languages[languageSet]);
1075.             display.print(F("Enable Serial     "));
1076.             display.println(booleanStateOnOff(enableSerial));
1077.             display.print(F("Change Slowenes   "));
1078.             display.println(changeSpeed);
1079.             display.setTextColor(BLACK,WHITE);
1080.             display.println(F("Restart Arduino   "));
1081.             display.setTextColor(WHITE);
1082.             display.print(F("Name: "));
1083.             display.print(SERIALNUMBER);
1084.             display.display();
1085.             if(joystickAxisYMoved())
1086.             {
1087.               showMessageOnScreen("Arduino restarting");
1088.               restartArduino();
1089.             }
1090.
1091.           break;
1092.         }
1093.       break;
1094.       case 7:
1095.         maxMenuCount = 1;
1096.         printCentered(F("Network Settings"),0);
1097.         switch (currentSubmenuNumber)
1098.         {
1099.           case 0:
1100.             display.setTextColor(BLACK,WHITE);
1101.             display.println(F("Left for reset"));
1102.             if(joystickAxisY > JOYSTICK_MAXTRESHOLD)
1103.             {
1104.               display.clearDisplay();
1105.               printCentered(F("Network restarting"),16);
1106.               display.display();
1107.               espSetup();
1108.               getIp();
1109.               display.clearDisplay();
1110.               printCentered(F("Network restarted"),16);
1111.               display.display();
1112.               Serial.println(F("Esp restarted"));
1113.               delay(500);
1114.             }
1115.             display.setTextColor(WHITE);
1116.             display.print("Operation mode: ");
1117.             if(networkType)
1118.               display.println("AP");
1119.             else
1120.               display.println("STA");
1121.           break;
1122.
1123.           case 1:
1124.             display.setTextColor(WHITE);
1125.             display.println(F("Left for reset"));
1126.             display.setTextColor(BLACK,WHITE);
1127.             display.print("Operation mode: ");
1128.             changeSubmenuBoolean(networkType);
1129.             if(networkType)
1130.               display.println("AP");
1131.             else
1132.               display.println("STA");
1133.               display.setTextColor(WHITE);
1134.           break;
1135.
1136.         }
1137.         display.print(F("Network Name: "));
1138.           display.println(ssid);
```

```
1139.           display.print(F("Password: "));
1140.           display.println(password);
1141.           display.print(F("Ip address: "));
1142.           display.println(ipAddress);
1143.           display.display();
1144.       break;
1145.     }
1146.
1147.   }
1148.
1149.
1150. }
1151. void setup() // az indításkor fut le
1152. {
1153.   // Setup resz, az inditaskor egyszer fut le:
1154.   Serial.begin(9600); //Soros Kommunikacio a teszteleshez
1155.   Serial1.begin(115200);
1156.   //meghívjuk a létrehozott függvényeket, amik az első lefutásnál kell
végrehajtódjanak
1157.   displayInitialize();
1158.   //megadjuk a, hogy a tüskék funkcióját
1159.   pinMode(JOYSTICK_BUTTON, INPUT_PULLUP);
1160.   pinMode(FANPIN, OUTPUT);
1161.   pinMode(IRRIGATIONPIN, OUTPUT);
1162.   pinMode(SPRAYPIN, OUTPUT);
1163.   pinMode(HEATERPIN, OUTPUT);
1164.   pinMode(LIGHTPIN, OUTPUT);
1165.
1166. }
1167.
1168. void loop() {
1169.
1170.   // put your main code here, to run repeatedly:
1171.   readJoystickValues();
1172.   if(enableSerial) //csak akkor küldi ki sorosan az adatokat, ha kap bemenetet
sorosan, igy nem vész el foloslegesen erőforrás
1173.     serialMonitorPrint();
1174.   changeMenuNumber(isSubMenu,maxMenuCount);
1175.   mainMenuSystem();
1176.   if(joystickButtonState == true)
1177.   {
1178.     if(millis()-lastTimeChange > menuChangeIntervall)
1179.     {
1180.       isSubMenu = !isSubMenu;
1181.       maxMenuCount = MAX_MENU_COUNT;
1182.       currentSubmenuNumber = 0;
1183.       lastTimeChange = millis();
1184.     }
1185.
1186.   }
1187.   transmitDateOnEsp(constructDateString());
1188.   readSensor();
1189.   outRelays();
1190.   display.clearDisplay();
1191. }
1192.
```

Bibliográfia

https://hu.wikipedia.org/wiki/Arduino#Szoftver

https://en.wikipedia.org/wiki/Integrated_development_environment

https://web.archive.org/web/20120227114343/http://tldp.fsf.hu/HOWTO/Program-Library-HOWTO-hu/index.html

https://docs.arduino.cc/hardware/mega-2560/

https://hu.wikipedia.org/wiki/I%C2%B2C

https://web.archive.org/web/20130511150526/http://www.nxp.com/documents/user_manual/UM10204.pdf

https://en.wikipedia.org/wiki/Hayes_AT_command_set

https://web.archive.org/web/20151028101531/http://www.zoomtel.com/documentation/dial_up/100498D.pdf

https://docs.arduino.cc/language-reference/

https://docs.arduino.cc/hardware/mega-2560/

https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf

https://docs.arduino.cc/software/ide/#ide-v1

https://docs.arduino.cc/programming/

https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1741392150506&ref_url=https%253A%252F%252Fwww.google.com%252F

https://docs.arduino.cc/micropython/micropython-course/course/serial/

https://docs.arduino.cc/libraries/ssd1306/

https://github.com/lexus2k/ssd1306

https://docs.arduino.cc/libraries/adafruit-gfx-library/

https://github.com/adafruit/Adafruit-GFX-Library

https://github.com/bonezegei/Bonezegei_DHT11

https://docs.arduino.cc/libraries/dht11/

https://github.com/dhrubasaha08/DHT11

https://blog.embeddedexpert.io/?p=613

https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/

https://components101.com/modules/joystick-module

https://www.researchgate.net/figure/The-pin-diagram-of-DHT11-temperature-sensors_fig2_359068957

https://www.google.com/url?sa=i&url=https%3A%2F%2Flastminuteengineers.com%2Fcapacitive-soil-moisture-sensor-arduino%2F&psig=AOvVaw3d6lzLw9fkb919Blm9CME5&ust=1744631596137000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCIjpmrD51IwDFQAAAAAdAAAAABAE
https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
https://dcc-ex.com/reference/hardware/wifi-boards/esp-01.html#gsc.tab=0
https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.analog.com%2Fen%2Fresources%2Ftechnical-articles%2Fi2c-primer-what-is-i2c-part-1.html&psig=AOvVaw0BWqK9NhBzePunKXn-zUoF&ust=1744908285940000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCPjLr4yA3YwDFQAAAAAdAAAAABA9
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.industrialshields.com%2Fblog%2Farduino-industrial-1%2Fi2c-bus-on-the-arduino-based-plc-for-industrial-automation-192&psig=AOvVaw0BWqK9NhBzePunKXn-zUoF&ust=1744908285940000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCPjLr4yA3YwDFQAAAAAdAAAAABBU
https://hu.wikipedia.org/wiki/Verzi%C3%B3kezel%C3%A9s