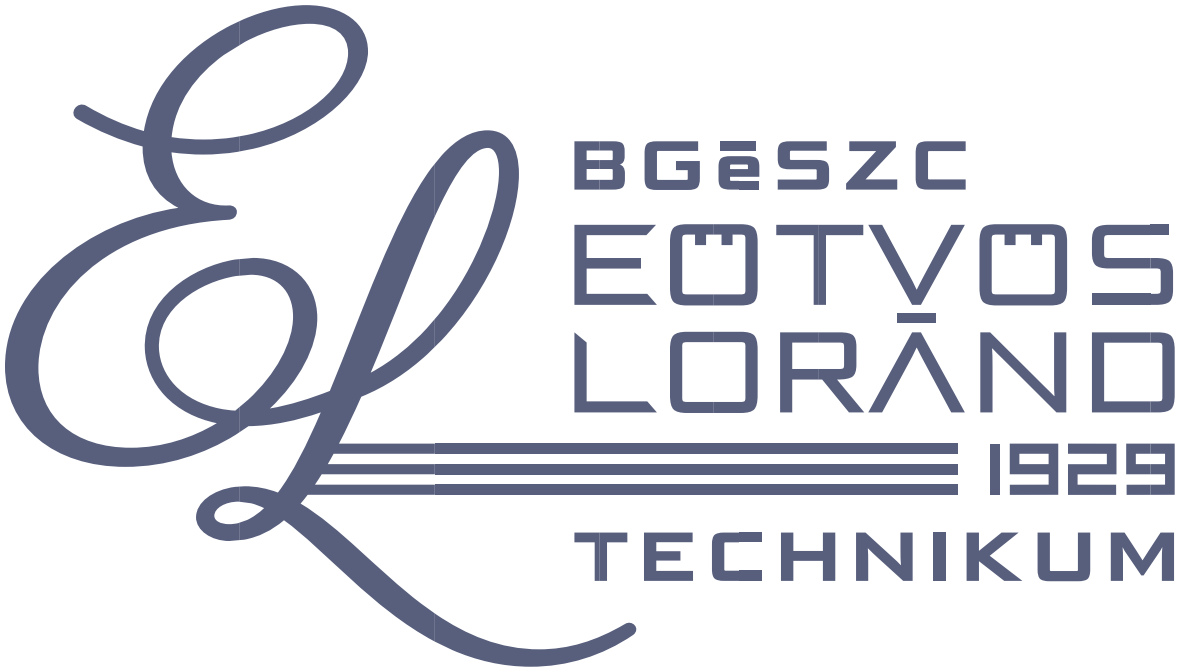


BUDAPEST GÉPÉSZETI SZAKKÉPZÉSI CENTRUM

Eötvös Loránd

Technikum

Műszaki informatikus

54 481 05

Dockery Robert

Patrick

2024

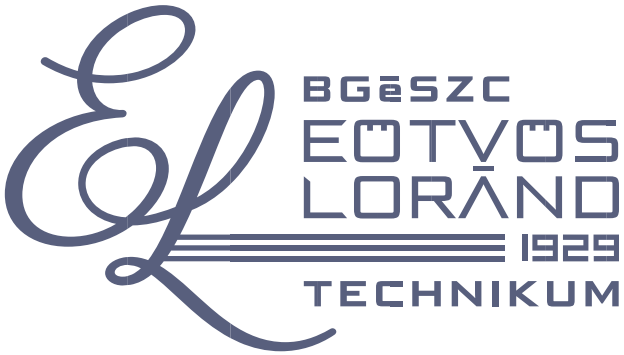
ZÁRÓDOLGOZAT

Okosüvegház

Dockery Robert Patrick

BUDAPEST

2024



BUDAPEST GÉPÉSZETI SZAKKÉPZÉSI CENTRUM

Eötvös Loránd

Technikum

Műszaki informatikus

54 481 05

Készítette

Dockery Robert Patrick

Konzulens

Molnár József

2024

TANULÓI NYILATKOZAT

Alulírott ..... műszaki informatikus tanuló kijelentem, hogy ezt a záró dolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző (k), cím, magyar nyelvű tartalmi kivonat, készítés éve, konzulens (ek) neve) a BGéSZC Eötvös Lóránd Technikum nyilvános hozzáférhető elektronikus formában, a munka teljes szövegét pedig az iskola belső hálózatán keresztül (vagy hitelesített felhasználó számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik.

Kelt: Budapest, 2022. április 17.

.....

Dockery Robert Patrick

OKOSÜVEGHÁZ

TARTALOMJEGYZÉK

BEVEZETÉS

Ebben a szakdolgozatban részletesen elmagyarázom a projektmunkám működését, milyen célt akarok elérni vele, továbbá, miért ezt a szakterületet választottam.

Az okosüvegház (elektronika) projekt célja, hogy egy automatizált és távolról vezérelhető rendszert hozzon létre, amely segíti a növények optimális környezetének fenntartását egy üvegházban. A rendszer központi eleme egy Arduino, amely különböző szenzorok és eszközök segítségével figyeli és szabályozza az üvegház körülményeit. A projekt hardveres komponensei között található egy DHT22 szenzor, amely a hőmérsékletet és a páratartalmat méri, valamint egy talajnedvességmérő, amely a talaj víztartalmát figyeli.

A rendszer tartalmaz továbbá egy 12V-os ventilátort és egy 12V-os vízpumpát (ezek csupán külső motorok), amelyek a levegő és a talaj optimális állapotának fenntartásáért felelnek. Az Arduino a szenzorok adatait egy webszerverre továbbítja, ahol azok egy adatbázisban kerülnek tárolásra. Az adatbázis minden rekordja tartalmaz: egy egyedi azonosítót, a hőmérsékletet, a páratartalmat, a talajnedvességet és a rögzítési időt.

A ventilátor és a vízpumpa vezérlése a webszerveren keresztül történik, amely Laravel keretrendszert használ. A felhasználók két-két gomb segítségével (bekapcsolás és kikapcsolás) irányíthatják ezeket az eszközöket, így biztosítva a növények számára szükséges optimális környezetet. Az adatgyűjtés és a távoli vezérlés kombinációja lehetővé teszi a hatékonyabb üvegház-kezelést, csökkentve az emberi beavatkozás szükségességét és növelve a termelékenységet.

Nevéből adódóan, egy műanyag dobozt felhasználva, szimulálok egy üvegházat, ebben helyezkednek el az Arduino által mérendő elemek (talaj, levegő…), melyeknek adatai egy adatbázisban vannak eltárolva.

1. TÉMA INDOKLÁSA

Több okból választottam ezt a témakört. Gyerekkorom óta játszottam, barátaimmal vagy a testvéreimmel és mindig is kíváncsi voltam, hogy az adott játék, hogy működött és, hogy volt felépítve. Kíváncsi voltam, hogy az eszközök amiken, játszottunk, hogyan voltak megépítve, ezért is választottam ezt az iskolát, ezzel a szakmával. Szeretném magam még jobban elmélyíteni a programozásba, ugyanis szeretnék a világ problémáira valamilyen megoldást keresni, legyen az egészségügyi(tudományi), akár közösségi vagy akármilyen területen elrendelt. Továbbá szeretnék a saját hobbijaimmal is foglalkozni, ezért előbb vagy utóbb valamilyen játék megtervezésével, kiépítésével, majd fenntartásával is kívánok foglalkozni. Végül a saját karrierem változatosságának érdekében egy személyes weboldalt, vagy webshopot is el akarok készíteni.

Jelen esetben ezzel a projekttel, egy korszerű megoldást szimulálok legfőképpen azok számára, akik valamilyen úton-módon foglalkoznak avagy rendelkeznek számos növényzettel és esetleg tartanak állatot. A projekt elkészítésekor számításba vettem édesanyámat és nagymamámat, mivel ők vidéken tartanak állatokat (tyúkok, kecskék), veteményeket, illetve zöldséget. Én magam is tapasztaltam, de legtöbbször láttam, hogy napi szinten mennyi macerát tud okozni rendre tenni a termést, és a baromfit (újonnan a kecskék is). Ez az „életmód” bár számtalan embernek hangzik jól, nem veszik ezeket a szempontokat figyelembe. Amiket igazán nem szoktak figyelembe venni ilyenkor az: megfelelő környezet megteremtése (veteménynek – **üvegház**, tyúkol, stb…); e környezet megteremtésére szükséges és majd a folytonos eltartáshoz igényelt költségek; végül időbeosztás. Nagyon sokan munka (állás, hobbik) mellett akarnak az imént elhangzottakkal foglalkozni.

Az én személyes életemnek ez a része adta az ötletet, hogy ha egy kis téren képes vagyok szimulálni, vezérelni és ezáltal automatizálni egy üvegházat, akkor fenn áll az esélye annak, hogy egy nagyobb méretű, napi szinten használt üvegházat is lehet az elektronika segítségével működtetni. Továbbá a mezőgazdaságban is alkalmazhatók lesznek ezek berendezések, így segíthetjük az aktív dolgozókat és akár globális szinten, a globális felmelegedés bizonyos problémáira is szóba jöhetnek a vezérelt/automatizált rendszer.

Összességében az automatizált és távoli vezérlésű rendszerek alkalmazása az üvegházakban és a mezőgazdaságban jelentős előnyökkel jár, növelve a termelékenységet, csökkentve a költségeket, és hozzájárulva a fenntartható fejlődéshez.

2. TÉMA KIFEJTÉSE

A dokumentáció alatt már számtalanszor előfordult az automatizálás és a vezérlés. Minden félreértés elkerüléséért szeretném bővebben kifejteni, hogy ezek egész pontosan mit is jelentenek, ezáltal egy kicsit magyarázni az elektronikát is.

„**Automatizáció** alatt azt a folyamatot értjük, amely során a vállalatok bizonyos munkafolyamatokat vagy azok egyes részfeladatait a humán munkaerő helyett automatizált munkavégzésre képes robotokra, algoritmusokra bízzák. Ez a fogalom magában foglalja a robotizált folyamatautomatizálási megoldások (RPA), valamint a mesterséges intelligencia (AI) alkalmazását is.” „A cégek digitális átalakulásának és a bizonytalanság kezelésének alapvető eszköze.” (dataxo group, 2024.)

Tehát az automatizálás olyan technológiai folyamat, amely során emberi beavatkozás nélkül végzik el a feladatokat gépek, szoftverek vagy más automatizált rendszerek. Az automatizálás célja az, hogy növelje a hatékonyságot, csökkentse a hibák számát, és lehetővé tegye az emberek számára, hogy összetettebb, kreatívabb feladatokra koncentráljanak. Az automatizálás számos területen alkalmazható, beleértve a gyártást, az informatikát, az ügyfélszolgálatot, a pénzügyi szolgáltatásokat és még sok mást.

Az automatizálásnak több típusa is van. Ezek közé tartozik:

- Ipari automatizálás: Gépek és robotok használata a gyártási folyamatokban. Például összeszerelő sorok, hegesztő robotok, és automatizált minőségellenőrző rendszerek.
- IT automatizálás: Számítógépes folyamatok automatizálása szoftverek és szkriptek segítségével. Például hálózati konfigurációk, adatmentés, és rendszerfelügyelet.
- Üzleti folyamat automatizálás (BPA): Adminisztratív és operatív feladatok automatizálása. Például számlázás, ügyfélszolgálat, és készletkezelés.
- Robotikus folyamatautomatizálás (RPA): Szoftverrobotok használata, amelyek utánozzák az emberi interakciókat a digitális rendszerekkel. Például adatbevitel, jelentéskészítés, és tranzakciófeldolgozás.
- Intelligens automatizálás: Gépi tanulás és mesterséges intelligencia alkalmazása az automatizált rendszerekben, hogy azok képesek legyenek

> tanulni és alkalmazkodni. Például ügyfélszolgálati chatbotok, prediktív karbantartás, és adaptív ajánlórendszerek. (Mérnökkapu, 2024)

Az automatizálás előnyei közé tartozik a hatékonyság növelése, bizonyos feladatokat, gyorsabban és pontosabban képesek elvégezni, mint az átlagos ember; hosszú távon kevesebb költségekkel jár és kevesebb hibákból való veszteségekkel járhat; továbbá az automatizált rendszerek egy stabil szintet alkalmazva mindig ugyanúgy hajtják végre a feladatokat, így megbízhatóak.

Sajnos az automatizálásnak van pár kevésbé előnyös szempontja is. Megépíteni egy automatizált rendszert, elég magas költségekkel jár. Ezen kívül megtervezni, fejleszteni, illetve karbantartani egy ilyen rendszert elég bonyolult lehet. Egy automatizált rendszer helyettesítheti az emberi munkaerőt is, ami társadalmi, lehetségesen gazdasági hatásokat eredményezhet. Bár igaz egyik előnye egy ilyen rendszernek a megbízhatósága, felügyelet alapján biztosítani kell, hogy semmiféle képpen ne végezzen hibát, főleg egy kritikus feladat elvégzésekor.

Mindennek ellenére, az automatizálás kulcsfontosságú szerepet játszik a modern társadalom és gazdaság fejlődésében, lehetőséget nyújtva a hatékonyság növelésére és az új technológiai innovációk kihasználására. (PLC GLOBAL, 2024)

A PLC Program szerint „további nyomás fog nehezedni a régimódi stílusú munkákra. A munkaerőpiacon további széttagoltságot tapasztalhatunk majd a „bennfentesek/magasan képzett” és a „kívülállók/alacsonyan képzett” munkaerő között.” Mindenesetre az automatizálás diktálhatja a jövőt, bár vannak akik, okkal, de még ha nem is óvakodnak, igyekeznek vigyázva alkalmazni az automatizálást rendszerben és ez az idézet erre fényt derít.

Összességében az automatizálás egy olyan folyamat folytonos elvégzésére szolgál, ahol az eredmények következetesek. Ezzel szemben a vezérlés egy olyan folyamat vagy mechanizmus befolyásolása, ahol egy kívánt eredményt vagy kimenetet akarunk elérni. A vezérlésnél alapvetően a rendelkező jel végigfut. A vezérlőberendezés utolsó szerve a beavatkozó jel. A vezérléssel egy manuális vagy akár automatizált rendszereket is lehet irányítani. Ennek eredménye képpen, lehet biztosítani az automatizált mechanizmusokban a hibák kiküszöbölését.

Kézi (manuális) vezérlésnél, a rendelkező jelet a kezelő személy tevékenysége határozza meg. Önműködő (automatizált) vezérlésnél, „a kezelőszemélyzet (az ember) akaratlagosan nem vesz részt a vezérlés működésében.” (Sulinet, 2024)

A vezérlést fel lehet osztani a vezetőjel alapján:

- Követő vezérlés: A vezetőjel határozza meg a rendelkező jelet(pl.: riasztórendszer, önműködő ajtó)
- Menetrendi vezérlés: A rendelkező jel előre meghatározott terv szerint jön létre. A rendelkezés függhet az időtől vagy a helytől.
- Időterv lefutó (menetrendi vezérlés): A rendelkező jelét az időterv tároló szolgáltatja. (pl.: időkapcsoló óra, amely adott időpontban kapcsol be és ki)
- Lefutó vezérlés: Vezető jelét a külső környezetből és a vezérelt folyamat állapotából származó feltételek határozzák meg.

A vezérlés elemei:

- Irányított jellemző: Amit vezérléskor irányítunk.
- Vezérlés: „Az irányításnak azt a módját, amelyiknél az irányított jellemző nem hat vissza az irányításra, vezérlésnek nevezzük.” (Sulinet, 2024)
- Érzékelők (szenzorok): Mérik a rendszer állapotát vagy környezeti változókat (pl. hőmérséklet, nyomás, sebesség).
- Vezérlő egységek (kontrollerek): Feldolgozzák az érzékelőktől kapott adatokat, és döntéseket hoznak a beavatkozásokra vonatkozóan.
- Beavatkozók (aktuátorok): A vezérlő egységek parancsait hajtják végre, és módosítják a rendszer működését (pl. motorok, szelepek).

Példák a leggyakrabban előforduló vezérlési rendszerekre: Ipari vezérlés - Gyártósorok, robotkarok, és gépek működésének szabályozása. Itt a vezérlés biztosítja a precíz és hatékony működést; Automatizált közlekedés - Járművek, mint például az önvezető autók, amelyek számos érzékelőt és vezérlő rendszert használnak a biztonságos és hatékony közlekedés érdekében; Épületirányítási rendszerek - HVAC (fűtés, szellőzés és légkondicionálás) rendszerek, világítási rendszerek, biztonsági rendszerek, amelyek automatikusan szabályozzák az épület környezetét.

Összefoglalva, a vezérlés olyan folyamat, amely irányítja és szabályozza egy rendszer működését, és kulcsfontosságú szerepet játszik a modern technológiai és ipari alkalmazásokban.

Ebben a projektben az Arduinot felhasználva, létesítettem egy automatizált rendszer, miközben folyamatosan vezérlem egyes elemeit (bizonyos feltételek alapján). Az Arduino egy nyílt forráskódú elektronikai platform, amelyet könnyen használható hardver és szoftver jellemez. Az Arduino platformot széles körben használják oktatási, hobbista és professzionális célokra, amivel lehetővé teszi az elektronikus eszközök egyszerű tervezését és fejlesztését.

Hardver: Az Arduino alaplapok (például Arduino Uno, Arduino Nano, Arduino Mega) különböző típusú mikrovezérlőket tartalmaznak, amelyek számos digitális és analóg bemeneti/kimeneti (I/O) porttal rendelkeznek. Ezek a portok különféle szenzorok, motorok, LED-ek és más elektronikus komponensek csatlakoztatását teszik lehetővé. Ezek eltérhetnek egymástól méret, mikrovezérlő, belső memória típus alapján. Továbbá az Arduino rendlelkezik olyan board-okkal (alaplapok) amelyekben beépített Wi-Fi, Ethernet, illetve Bluetooth csatlakozásra van lehetőség, mindeközben beteljesíti az Arduino board-ok feladatát. Ilyen például a Wemos-nak számos alaplapja. Pár forgalomban lévő modell: „***Arduino/Genuino UNO*** - Széles körben használt modell, 8 bites, 16 MHz-es ATmega328P processzorral. 14 digitális I/O lábbal és 32 kB flash memóriával rendelkezik.”; „***Arduino/Genuino Micro*** - Kisméretű, ATmega32U4 processzorral rendelkező változat, amely micro USB csatlakozóval rendelkezik és breadboardon való használatra tervezték. Beépített USB vezérlője van, aminek segítségével akár egérként vagy billentyűként is működhet. 32 kB flash memóriája van.” (Wikipedia, 2024). Az Arduino ezen kívül rendelkezik még shield-ekkel, amikkel valójában bővithetjük az Arduino (alaplap) és a különböző szenzorok közötti funkcionalitást. Ezek a shield-ek egyszerűen illeszthetőek az elektronikai áramkörökhöz, esetenként hasonló a kinézetük, mint egy másik alaplap vagy szenzor, mikrokontroller. (Pár példa hivatalos shield-re: Arduino GSM Shield, Servo Shield.) Az Arduinonak alapvetően három fő memória típusa van. Ezek a **flashmemória**, **SRAM**, **EEPROM**. A **flashmemória** felelős a letöltött programok, könyvtárak tárolására, melyeket kikapcsolás után is megőriz. Fontos megjegyezni, hogy a letöltőprogram (bootloader) innen is használ memóriát. Alap esetben 32 kB áll rendelkezésre. Az **SRAM,** vagyis *Static Random-Access Memory* „tárolja a programban definiált belső változókat. Az **SRAM** - szemben a flash-memóriával - árammentes állapotban nem őrzi meg a tartalmát, ezért minden bekapcsolást követően a program újradefiniálja a változókat és azok az ott meghatározott "alapértelmezett" értékükkel kerülnek az **SRAM**-ba.” (Wikipedia, 2024) Végül az **EEPROM**

(*Electrically Erasable Programmable Read-Only Memory*) Nem felejtő memória, amely megőrzi az adatokat a tápfeszültség kikapcsolása után is, de lassabban írható és olvasható, mint az SRAM. Képes konfigurációs beállítások és kalibrációs adatok tárolására. Mérete (alaplaponként más) és élettartama miatt, csak szükség esetén használják.

Összességében, mivel korlátozott mennyiségű memóriával rendelkeznek az Arduino alaplapok, fontos, hogy hatékonyan használjuk fel ezt a memóriát. Pár fontos szempont, amit érdemes figyelembe venni, a memória felhasználásakor: A megfelelő adatszerkezetek és típusok használata, a memória felhasználásának minimalizálásáért; a programkód optimalizálása, csak a szükséges könyvtárak és kódrészletek használata; végül előfordulhat, hogy egy ritkán változó adatot az EEPROM-ban tárolunk az SRAM helyett (pl.: felül említett konfigurációs beállítás).

Szoftver: Az Arduino Integrated Development Environment (IDE) egy egyszerű és ihletes, de közvetlen szoftverkörnyezet, amely C++ alapú programozási nyelvet használ. Az Arduino IDE lehetővé teszi a programok írását, tesztelését és feltöltését az Arduino alaplapokra. „Az Arduinonak van saját nyelve (az alapja a Wiring), és fejlesztő környezete (alapja a Processing).” (raptor13, 2017) Az Arduino IDE platform független, vagyis bármelyik platformra letölthető (Windows, macOS, Linux…), továbbá nyílt forráskódú, ezáltal könnyedén lehet a rendszerhez kiegészítéseket készíteni (bővítmények, könyvtárak). Az Arduino IDE képes c-alapú programozásra is, fel tudja dolgozni a direkt AVR-C kódot is. (raptor13, 2017) Az Arduino IDE tartalmaz egy fordítót is, ezzel képesek vagyunk az Arduino C/C++ nyelvről, gépi kódra, amit az Arduino alaplapja szintén megérthet. A felhasználó számára ez csupán egy hibaellenőrzési folyamat, ahol a kód kerül vizsgálatra. A felhasználó a Soros monitoron keresztül képes a szoftverrel kommunikálni, továbbá itt láthatja az érzékelők, mikrokontrollerek által meghatározott adatokat (bejövő adatok – pl.: hőmérséklet, távolság). A kész programot USB-n keresztül lehet feltölteni az Arduino alaplapra. Az Arduino kód fejlesztésére más eszközök is használhatók, mint például a Visual Studio Code az Arduino kiterjesztéssel, PlatformIO vagy az Eclipse. Az Arduinohoz ezen kívül lehet társítani egy parancssort, ahol szintén lehet a fejlesztési folyamatokat kezelni, fordítani és még akár feltölteni. Az Arduino szoftver(ek) célja az egyszerűség (használatában), míg széles választékot nyújt az elektronikai projektek fejlesztéséhez, megkönnyítve a hardveres és szoftveres komponensek integrációját.

Az Arduino-t rugalmassága miatt, gyakran párosítják más szoftverekkel, így bőséges választási elérhetőségek merülnek fel (kódban – ezáltal kivitelezés), továbbá magasabb szint(ek)re lehet emelni egy projektet, szerkezetet. Nagyon gyakori az Arduino bővítésénél a webes szolgáltatások kiépítése, mint például az adatok feldolgozása és tárolása, vagy az Arduino kezelése. Erre nagyon gyakran használt a Laravel keretrendszer. Ez a PHP-nak az egyik legismertebb keretrendszere, továbbá egy MVC-s keretrendszer, „azaz a kód strukurálására a Modell-Nézet-Vezérlő *(Model-View-Controller)* mintát ajánlja." (totadavid95, 2022). Ingyenes és nyílt forráskódú, mind ezek mellett modern és könnyű használni. Erős parancssori támogatottsággal rendelkezik, és szép szintaxissal. Az MVC mintában az adatok tárolásáért és feldolgozásáért a modell felelős. Szintén itt tudjuk ezeket az adatokat akár visszaolvasni vagy törölni. A Laravel rendelkezik még nézetekkel (view), ahol tetszőlegesen be lehet állítani a weboldal, illetve az adatok megjelenését, kinézetét. Egyszerre több nézetet is alkalmazhatunk, ahol egy nézet a teljes weblapnak egy részéért felelős és tartalmazza annak stílusait. A nézet a felhasználói felületért felelős. A vezérlő (controller) a modellel van összeköttetésben olyan értelemben, hogy a vezérlőben kezelhetjük az adatokat és azok megjelenését különböző függvényekkel (sql…). Be tudjuk olvasni a modell adatait, utána meghívhatjuk a modell függvényeit. A vezérlőben „fogadjuk a http kérést." (totadavid95, 2022). Végül a routing lehetővé teszi a felhasználó számára a különböző útvonalak egyszerű kezelését. Ezek a route-ok magukban tartalmaznak egy függvényt a felhasznált kontrollerből (vezérlő). Ezek a route-ok hasonló célt szolgálnak, mint egy szétbontott URL, ahol nem mindegyik lesz megjelenítve, de mindegyikhez hozzá van rendelve egy függvény, ami lefuttathat sql parancsokat vagy akár ellenőrizheti az adatok érvényességét (validálás). A Laravel egyik sajátossága a Blade sablonmotor. Ez egy beépített motor, amivel dinamikusan létre tudunk hozni HTML oldalakat. A Laravelben tudunk migrálni és adatbázisokat kezelni. A migrációs rendszerrel tudjuk kezelni az adatbázisokat, és a sémaváltoztatásokat könnyebben. (Kiszervezett Marketing, 2024). A Laravel saját objektum-relációs leképezési (ORM) rendszert kínál, az Eloquent-et, amely lehetővé teszi az adatbázis rekordokkal való egyszerű és intuitív munkát. (Kiszervezett Marketing, 2024). Egy másik funkciója a Laravelnek az Artisan, ami egy parancssori eszköz, számos beépített parancsot kínál, ezek segítenek az alkalmazás fejlesztésében és karbantartásában, például a kód generálásában és az adatbázis migrációk kezelésében. A Laravel továbbá alkalmaz egy Middleware-t (köztes szoftver), ami

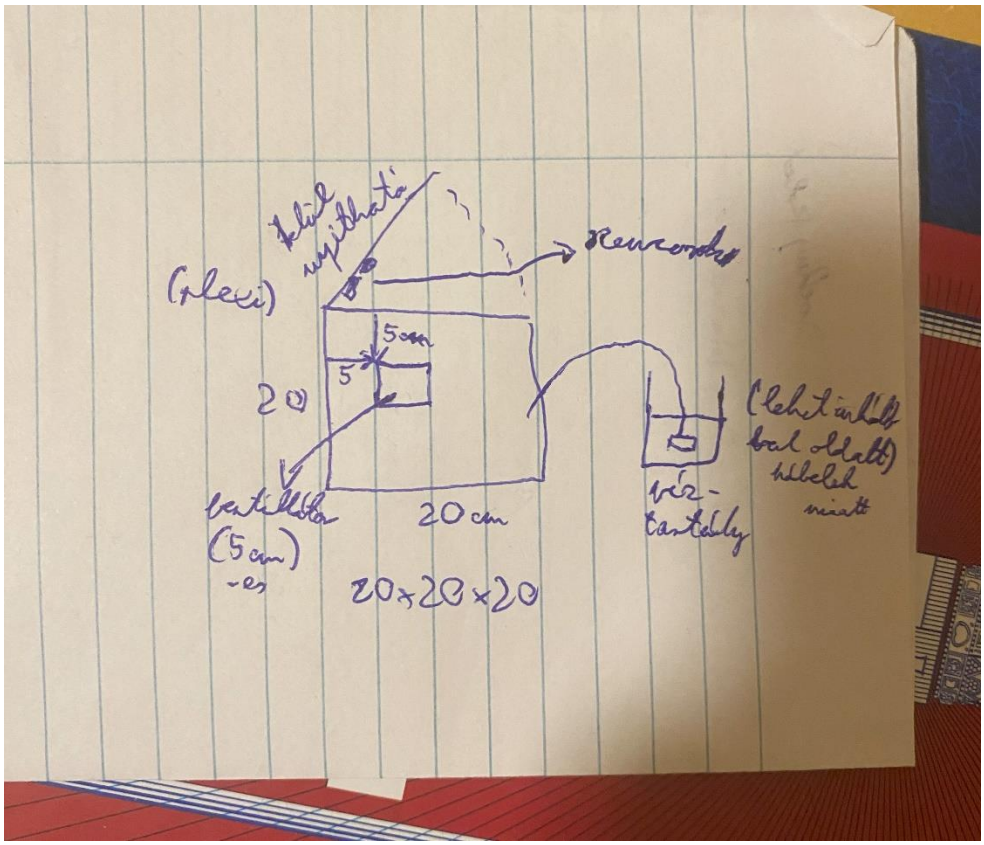
„kényelmes mechanizmust biztosítanak az alkalmazásba érkező HTTP-kérések ellenőrzéséhez és szűréséhez. Például a Laravel tartalmaz egy köztes szoftvert, amely ellenőrzi az alkalmazás felhasználójának hitelesítését. Ha a felhasználó nincs hitelesítve, a köztes szoftver átirányítja a felhasználót az alkalmazás bejelentkezési képernyőjére. Ha azonban a felhasználó hitelesített, a köztes szoftver lehetővé teszi, hogy a kérés tovább haladjon az alkalmazásban." (Laravel, 2024). Gyakran használt még a CSRF token (cross-site request forgery), ez védelmet nyújt, olyan kérés-hamisítási támadásokkal szemben, ami webhelyek között zajlik le. Ez az egyik olyan Laravel szolgáltatás, ami része a felhasználó hitelesség- és jogosultságkezelésének. A Laravel ugyanúgy rendelkezik ciklusokkal és elágazásokkal, mint a C# (másnéven C Sharp). Ezáltal a Laravel tartalmaz, és gyakran alkalmazzák az objektum orientált parancsokat. „A Laravel session-ök lehetőséget biztosítanak a felhasználó információinak tárolására több kérésben. Ezeket a felhasználói információkat általában egy állandó tárolóban/háttérrendszerben helyezik el, amely a következő kérésekből érhető el." (Laravel, 2024). A Laravel HTTP session lehetővé teszi a felhasználói adatállapotok (például bejelentkezési állapot, űrlapadatok) tárolását és kezelését az alkalmazásban. A session-ök biztosítják, hogy a felhasználók információi megmaradjanak az oldalak között történő navigáció során. Az applikáció session konfigurációs fájlja a config mappában található, azon belül a session.php fájlban tárolja el a session-t. A session.php rendelkezik a „driver" parancssal, ezzel tudjuk eltárolni a session vagy session-ök adatait a megadott kéréshez. Alap esetben a Laravelben a „database" (adatbázis) van konfigurálva a driver-hez. A Laravel több driver variációval rendelkezik, melyekkel megadhatjuk, hogy hol tároljuk el a session-öket. Tárolhatjuk biztonságos, titkosított cookie-ban, az alapértelmezett relációs adatbázisban (database), vagy akár a gyorsítótárak egyikében (memcache/redis). A Laravel session tartalmaz olyan háttérprogramokat, mint például a Memcached, Redis, amelyek egy egyesített API-n keresztül elérhetőek. (Laravel, 2024). Továbbá a Laravel köztes szoftvere biztosítja a cookie-k automatikus titkosítását és visszafejtését. (lehet-pullrequest).

Az Arduino és Laravel közötti kapcsolat lehetővé teszi, hogy a webes alkalmazások vezéreljé a fizikai eszközöket, és visszajelzést kapjon tőlük. Ez az összekapcsolás rendkívül hasznos az IoT (Internet of Things) projektek, okosotthon rendszerek és

egyéb automatizált megoldások esetén. Ezen funkciók sokaságát, fizikai szemléleteket felhasználva építettem ki a projektemet.

3. RENDSZER BEMUTATÁSA

Kezdetben az elektronika projektemhez szükségem volt meghatározni a célt, a rendszert, amit működtetek, milyen eszközökre lesz szükségem, illetve ezeket a szenzorokat miben fogom eltárolni. Elképzelésem szerint egy műanyag, plexi, vagy fadobozt készítettem volna el, ezek létesítették volna a házat, a projekt központját, amihez képest végeztem el a további vázlatokat, terveket. Miután már volt egy alapvető pont ahonnan el tudtam indulni, elkészítettem a projekt prototípusát kartonból, ami minden irányba 20 cm-es, így 8000 cm^3 térfogatú. Ennek köszönhetően lett volna hely elhelyezni a ventilátort akármelyik oldalán és elgondolás szerint 5 cm-el beljebb lett volna a mellette és a felette lévő élektől. A rajz alapján közelebb helyezkedik el a doboz tetejéhez, ahol be lehet nyitni és a nyitható lap sarkához, ahol összeérnek az élek. További elképzelések szerint a hőmérő ráilleszkedne a nyitó/fedőlapra, ezáltal az egész dobozban képes lemérni a hőmérsékletet és a páratartalmat. Eleinte a talajnedvesség mérő csupán be lett volna eresztve a doboz tetejéről kicsi szélessége miatt, ugyan is akkor még nem volt megtervezve egy pont a dobozban, hogy a talajnedvesség mérő onnan illeszkedjen be és nem nagyon szükséges, ha alapvetően kisebb, a kábelek pedig nem akadályozzák a hő megtartását. A vizet pumpáló motor is hasonló elképzelés alapján volt megvázlatolva, hiszen az csak egy kábellel kell hogy hozzáférjen a doboz belsejéhez. Ez viszonylag vastagabb kábel, elvégre vizet fog magán keresztül szállítani. Emiatt kifolyólag a dobozon már számításba vettem egy kis rés elkészítését, így a vízpumpa kábele belefér a dobozba. A kezdeti vázlatokban a vízpumpa a dobozon kívül helyezkedne el, de azt még akkoriban nem határoztam meg, hogy hogyan. Valójában két opción gondolkodtam: a doboz külső részén lenne egy kisebb tartály, viszont ez a méretekbe is beavatkozott volna, az eredeti méretekkel nem lenne számottevő mennyiségű víz és felboríthatná a doboz egyensúlyát, ha viszont megnagyobbítom az egész dobozt lehetségesen újra kell méreteznem. A másik lehetőség, egy külön tartály, palack, amiben lenne a vízpumpa és csak a kábelt kellene bedugni a dobozba. Bár ez a megoldás kényesebb, abból a szempontból, hogy lehet többet cipelnék, egyszerűbb megvalósításában és a doboz anyagától függetlenül tudom társítani. Ha mondjuk plexiből készíteném el a dobozt, azt alapvetően nehezebb hordani és megépíteni, mint mondjuk a műanyag.



1. Üvegház vázlata/tervrajza

Amint elkészült a doboz tervrajza papíron, a prototípus is meg volt építve, utána néztem, hogy milyen Arduino-s eszközök lennének szükségesek a projekthez. Mivel csak három beérkező adatot dolgozok fel, melyek a hőmérséklet, páratartalom és a talaj nedvessége, ebből kettőt egy szenzor érzékel, kiválasztottam a megfelelő szenzorokat. Ezek voltak az Arduino dobozban található talajnedvesség mérő és a DHT11-es hőmérséklet és páratartalom érzékelő. Miután meghatároztam a szükséges szenzorokat, elkezdtem Tinkercadban szimulálni a projektet, bár sajnos nem elérhető az összes Arduino által használt érzékelő. Mivel még a Wemos sem található meg a Tinkercadban, ezért egy Arduino Uno-hoz csatlakoztattam az eszközöket. Mindeközben utána néztem online blogokban és videókban, a külső motorok működtetéséhez és esetenként találtam egy projektet, ami szintén alkalmazta volna az általam is használt szenzorokat. Esetek többségében mások egy hőmérőt alkalmaztak, de nem azt amelyik számomra elérhető volt. Számomra ez elég feltűnő volt, ezért utána néztem, hogy mi a különbség a különböző hőmérők között és láttam, hogy a DHT szenzorok rendelkeznek egy újabb verzióval, ez pedig a DHT22 szenzor. Ugyanazokkal a funkciókkal és kapacitással rendelkezik, mint az előző verziós

érzékelők, csak pontosabban méri le az a adatokat pár tizedes számmal, ezáltal enyhén egyszerűbb is kezelni a számok kerekítésekor. A DHT szenzorok bár nem rendelkeznek sok verziókkal, az egyes típusú DHT szenzoroknak több fizikai variációjuk létezik. Van olyan hőmérő, aminek csak három lába van, míg egy másik hőmérőnek, ami ugyanolyan, három lába van. Ezek a szenzorok a belső áramkörüktől különböznek. Egy három lábú hőmérőnek a lábai felelősek az áram szolgáltatásáért, alapvetően szükséges a hőmérő működtetéséhez, vagyis innen kapja a megfelelő tápfeszültséget; az adat elküldéséért, amit majd az Arduinoban (szoftver) feldolgozunk, majd Laravelben kezelünk az adat eltárolásáért adatbázisba; a harmadik lába felelős a földelésért. Egy négy lábú DHT szenzor ugyanezeket tartalmazza annyi különbséggel, hogy nem rendelkezik beépített ellenállással, ezért azt külön kell hozzá csatlakoztatni az áramkörhöz. Ez az opció annyival kedvez, hogy a felhasználó eldöntheti magának az ellenállás mennyiségét (ohmokban), azonban gyakran nem szokták össze kötni ezt a lábát semmivel épp ezért elnevezése sincs. A talajnedvesség mérőnek a kapcsolatai közvetlenül össze vannak kapcsolva a kábelek, voltaképp nem a lábaival, hanem a vezetékek egymással vannak összekötve. Ezeknek az összekötése egyszerűbb volt , mivel hogy ezeket az évfolyamán gyakoroltuk. A két érzékelőn kívül még alkalmaztam egy kis LED-et, ami szintén az Arduino dobozában volt benne. Ennek a LED diódának két lába van. A dióda egy olyan félvezető, aminek két kivezetése van. Ezek az anód és a katód. Az anód az általában a hosszabb láb és a pozitív pólus, ide kell csatlakoztatni a tápfeszültséget. Általában ezt a lábat egy digitális pinbe kötjük az Arduino vagy Wemos board-okon, így tudjuk a programon belül is irányítani a LED-et és mivel a board alapvetően kap áramot bekötéskor ezért a LED ezt használja tápfeszültségnek. Ilyen esetben egy ellenállásból kell kivezetni egy digitális pinbe. Ezzel ellentétben a katód a rövidebb lába és a negatív pólus, melyet a földeléshez kötünk. Az áram a katódon keresztül lép ki a LED-ből. Ha esetleg a LED-nek mindkettő lába ugyanolyan hosszú, a LED házában is van egy jelölés a katód lábára a lapos oldalán. Miután a hőmérőt és a talajnedvességmérőt összekötöttem szimulációban, társítottam hozzájuk a LED-et, így a központi Arduino egységeim már teljesek voltak. Ahhoz, hogy teljesen meggyőződjek az összekötésről és a vezetékek kapcsolódásáról, elkészítettem egy áramköri diagramot a projektem jelenlegi állapotáról. Az aktuális fejleményeket rögzítettem a Tinkercad tervrajz, szimulációban is. Mindezeket az érzékelőket és a LED-et először külön működésre bírtam Arduino-ban a saját kódjukkal. A LED viszonylag egyszerű volt, ugyanis az Arduino IDE

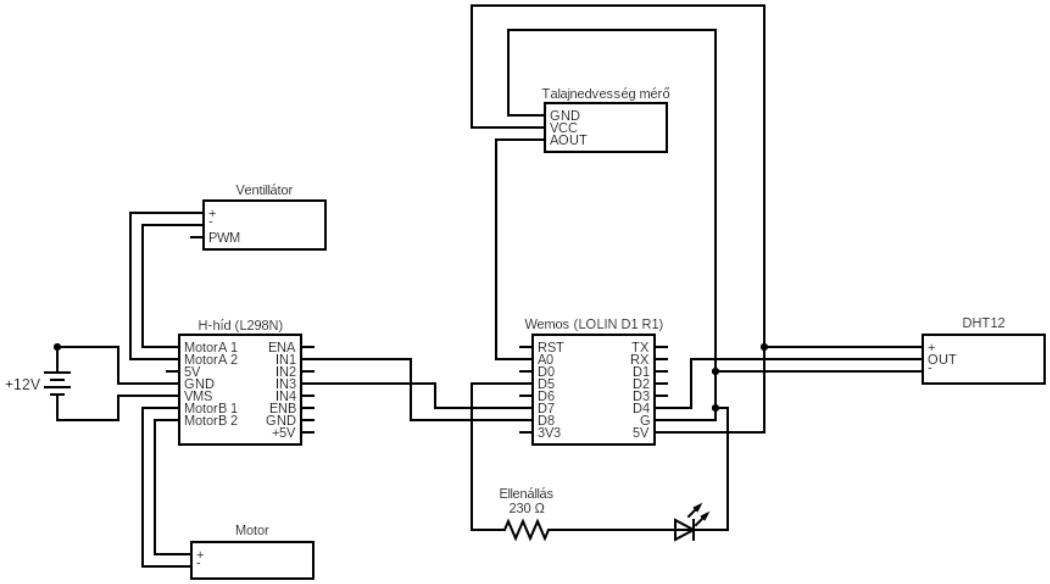
tartalmazz egy minta kódot a LED-ek (beépített vagy sem) felkapcsolására, ami „Blink” néven található a „File” azon belül pedig az „Examples” mappában. Mivel a projektet még az iskolai gépeken kezdtem el, a saját órai anyagaimból ki tudtam keresni a talajnedvesség mérőnek a kódját. A legtöbb problémát a hőmérő okozta, ugyanis eleinte nem működött a DHT11 által használt kóddal. Többféle képpen próbáltam ezt a problémát megoldani. Úgy gondoltam hogy lehet a kód nem jó, ezért több kódrészletet is kerestem és ha van egy olyan ami többször előfordul és bizonyítottan jó, esetleg azt összevethetem a sajátommal, ezáltal kiszűrném a hibát, de sajnos nem jártam sikerrel. A DHT22 továbbra sem működött az elvárt módon, sőt a DHT11 sem funkcionált rendesen a felújított kóddal. Ezek után visszatértem az eredeti kódomhoz, de az akadályok továbbra is fennálltak. A hőmérőt valamiért, még mindig nem tudom kalibrálni, nem ismeri fel az Arduino a DHT22-t definiálásakor. Próbáltam átírni a hőmérő típusát, ellenőriztem a kötéseimet. Hosszas próbálkozások után a DHT22 szenzor már végre működött, helyesek voltak az összekötések, a szoftver is felismeri a szenzort és a típusát. Ezek után frissítettem az áramköri diagramomat és a Tinkercad szimulációt. Amint az alapvető Arduino eszközöket már működésre tudtam bírni, elkészítettem egy HTML dokumentumot, ami voltaképp a kész weboldalnak a prototípusa. Itt csak a kinézettel kellett foglalkoznom, ezért igyekeztem a lehető legjobban kiépíteni, hogy ne keljen előröl tervezzem a nézetét a végleges weblapnak. A prototípust majd fel tudom használni alapnak, amin csak bővíteni kell. Ez már tartalmazta a megfelelő Bootstrap alakzatot is. Következő lépésnek meg kellett határoznom, hogyan kötöm össze a külső motorokat az Arduino-val. Sok megoldást találtam, ahol egyesek tranzisztorokat alkalmaztak, mások viszont valamilyen relét. Ez a két megoldás fordult elő a leggyakrabban, ezért én is relével próbálkoztam. Mivel csak egy relém volt, nem tudtam a ventilátort és a vízpumpát egyszerre bekötni. Sőt, mivel a két motorhoz szükséges tápfeszültséget is hozzá kell kötnöm a reléhez, ezért valójában egyik motort se voltam képes egyelőre összekötni. Ezen fejlemények következtében, újra át kellett gondolnom az összekötést. Ebben a pillanatban egy újabb relé modulra próbáltam beruházni, aminek már több reléje van, így hozzá tudom kötni a két motort és ezeket el tudom látni árammal. Miközben kerestem egy relét, láttam, hogy léteznek különféle hidak, amik hasonlóképpen működnek, mint a relé modulok vagy relé shield-ek; engedélyezi, hogy irányítsam a motorokat Arduino IDE-n keresztül, hozzá lehet kötni az Arduino-s áramkörhöz. Szokták még ezeket a hidakat „driver”-nek (vezető) hívni. Az egyik ilyen híd, ami szerepel a projektemben, az

L298N motorvezérlő IC (integrált áramkör), amelyet DC motorok és léptető motorok vezérlésére használnak. Két motorvezérlővel rendelkezik, melyeket fel és le lehet kapcsolni. Ezek részei az IC-nak, ezekkel a motorvezérlőkkel függetlenül tudjuk vezérelni a két egyenáramú (DC) motort vagy egy kétfázisú léptetőmotort. Ezen kívül még szoktak lenni védődiódák, különböző LED-ek és egy hűtőborda az L298N hídon. A híd fő bemeneti motorja három bemenettel rendelkezik, ezek: egy VCC bemenet, a tápfeszültség bemenete ami már 12 volton is tud operálni, maximum 46 voltot bír; egy GND bemenet, ez a közös föld, hasonlóan mint az Arduino board-okon; és a harmadik bemenet egy 5 voltos bemenet, ami a logikai tápfeszültség bemenete. Ezt a modult egy belső 5V-os regulátorral is lehet táplálni. Az L298N híd ezen kívül még rendelkezik kimeneti vagy vezérlő jelekkel, amiket már hozzá köthetünk az Arduino (vagy más típusú) board-hoz a motorok irányításához, ezért vezérlő jelek. Hat vezérlő jele van a hídnak, ebből kettő engedélyezi a motorok működését (Fel és le lehet kapcsolni, de még szabályozni is mivel ezek analóg jelek). Ezek az ENA és az ENB, nevük az „enable” és a hozzájuk tartozó motorvezérlő betűjele. A maradék négy vezérlő jel a két motorvezérlőnek két-két bemenetét, vagyis a motorok forgási irányát engedi irányításra. Ez a négy jel az input pinek amik el vannak számozva egytől négyig, ahol az első kettő az „a” motorvezérlőhöz, a maradék kettő pedig a „b” motorvezérlőhöz van társítva. Látván, hogy ez a híd tökéletes lenne a projektemhez, próbáltam beruházni egyre, de szerencsémre Molnár József Tanárúr szolgáltatott egyet, a hőmérővel együtt. Ezek mellé még kaptam egy Wemos D1 R1-et is. A Wemos D1 R1 rendelkezik egy ESP8266-os chippel, ez engedélyezi a vezeték nélküli kommunikációt. Az ESP8266 továbbá rendelkezik beépített WiFi szolgáltatásokkal. Alapvetően ugyanúgy vannak rajta digitális (11) és analóg (1) jelek (pinek), 5 és 3,3 volt tápfeszültséget is tud szolgáltatni a különböző érzékelőknek, ezért természetesen földelése is van. Két soros kommunikációs jele (RX, TX) és egy visszaállítás (RST - Reset) gomb is van rajta. Négy megabyte flashmemóriája van. Miután az összes eszköz elérhető volt számomra, elkezdtem együtt kipróbálni őket és elhárítani a lehetséges hibákat. Bár eleinte nem jártam sikerrel a motorok kalibrálásában, azért hogy biztosan haladjak, elkezdtem a Laravel projektet is. Itt létrehoztam az alap, illetve szükséges beállításokat. Létrehoztam egy nézetet, a „php artisan make:view” parancssal, ezzel megteremtettem a főoldalt, ezen belül még beillesztettem a már létező HTML prototípust, így már csak pár apróságokat kellett beállítanom, miközben kiépítem a webszervert. Ezután létrehoztam a parancssorból a kontrollert a „php artisan

make:controller” parancssal. Itt még nem állítottam be semmit eleinte, majd csak akkor amikor volt adat amit kezelni kellett és már megvolt a modell. Épp ezért következőleg létrehoztam a modellt szintén a parancssorban: „php artisan make:model”. Ahhoz hogy adatbázisba tudjam tárolni a majd megérkező adatokat létrehoztam a migrációt is, „php artisan make:migrate”, de még nem futattam az adatbázist mivel nem volt akkoriban adatom és még nem szerkesztettem meg a migrációt. Továbbá, ahhoz hogy tényleg legyen egy funkcionáló adatbázisom, létre kellett hoznom a PHPMyAdmin-ban ugyanazt az adatbázist és táblát. Azért, hogy a Laravel projektem össze legyen kötve az újonnan létrehozott adatbázissal, Laravel-ben a .env fájlomat, módosítottam annyival, hogy megadtam az adatbázisom nevét. Azért is szükséges külön létrehozni az adatbázist, hogy képes legyek Laravel-en keresztül csatlakozni és kommunikálni vele. Mivel az adatbázisom a projekt során nem változott az adattáblát is elkészítettem. Az első oszlop egy azonosítószám, utána jelennek meg a hőmérséklet, páratartalom és a talajnedvesség adatai, majd végül a rögzítési idő amikor ezek az adatok bekerültek az adatbázisba az Arduino-ból. Amikor megbizonyosodtam arról, hogy az alap Arduino szenzorok, amik mérnek le adatot működnek, megírtam az Arduino kódot, amivel el tudom küldeni az adatok a webszerverre. Az ESP8266 három könyvtárát, WiFi, HTTPClient, Webserver használom. Utána definiáltam a WiFi SSID-jét (hálózat neve) és a WiFi jelszavát. Fix IP beállításokat is alkalmazok, abban az esetben amikor statikus hálózatra vagyok rákapcsolódva, WiFi esetén nem. Végül elkezdtem a webszervert. Ugyan itt kalibráltam a hőmérő szenzort, definiáltam típusát, és a pint ahová bekötöttem a Wemos-ba. Ezek előtt megadtam a DHT könyvtárat, amit használok. Alatta egy sorral beolvastam a hőmérőt az Arduino-ba (jele, típusa). Ennél több könyvtárat már nem kellett rögzítenem, de a program legelejére még megadtam a „Wire.h” könyvtárat, ami egy alapértelmezett könyvtára az Arduino-nak, I2C (Inter-Integrated Circuit) protokollal kapcsolatos kommunikációt kezeli. Ez a protokol lehetővé teszi több eszköz összecsatlakoztatását egy kétvezetékes buszon. Azok után, hogy megadtam a könyvtárakat és elvégeztem a szükséges kalibrációs beállításokat, elkezdtem létrehozni a változókat az érzékelőkre és a motorokra. Eleinte több változót is létrehoztam külön-külön a motorokra, ugyanis nem voltam teljesen tisztában a konfigurálásukkal. Volt hogy egy motornak három változót definiáltam, hogy tudjam irányítani a motorvezérlőt, és az ahhoz tartozó motor irányát. Ilyen esetben nem tudtam az egész rendszert összekötni, mert a motorvezérlőt irányító jel analóg és az általam

használt Wemos-on csak egy analóg bemenet áll rendelkezésre, továbbá semmi szüksége nincs analóg jellel kezelni a vezérlőt, nem hat rá a projektemre. Ebből kifolyólag, próbáltam két-két változóval irányítani a motorokat, ahol a két változó valamelyik irányra felelős. Mivel a cél a motorok működtetése volt, elég csak az egyik változónak az értékét változtatni, az irány nem fontos. Igyekeztem a motorok tápfeszültségét vevő kábeleket párosítani, a motorvezérlő bemeneti jeleihez (IN1-4) ami a Wemos-al lett összekötve. Így remélve, hogy nem fordított logika alapján tudom őket felkapcsolni, de ez sajnos nem sikerült, úgy ahogy vártam, ezért míg Arduino kódban lehet úgy tűnik, hogy lekapcsolom a motort (alacsony – LOW állapot), gyakorlatban akkor kapcsol fel. Ezek a motorok a weblapon keresztül vannak kezelve, mindkét motorhoz társítottam két-két gombot, hogy fel és le tudjam kapcsolni őket. Az Arduino-ban maga a kapcsolás folyamata külön függvényekben történik, Laravel-ben gombnyomáskor ezek a függvények meghívásra kerülnek. Miután a változókat és a konstansokat létrehoztam, a setup függvényben, inicializáltam a hőmérő szenzort a „dht.begin()” parancssal, a soros kommunikációt is megkezdtem hasonlóan „Serial.begin()”; és a többi szenzornak is beállítottam a feldolgozott jel módját a „pinMode()” parancssal, amelynek két paramétere van, az egyik az valamely szenzornak a változó neve, a másik pedig lehet bemenet (INPUT) vagy kimenet (OUTPUT) attól függően, hogy olvasunk vagy írunk a megadott szenzorról. Ezen kívül még csatlakoztatom a Wemos-t a WiFi-hez és megkezdem a webszervert és elküldöm motorok által használt (több) URL-t szintén a webszerveren. A ventilátor és a vízpumpa külön függvényekben van kezelve és mivel két gomb van a weblapon a két motornak ezért Arduino-ban is két függvényben kapcsolom őket fel és le, utána elküldöm az állapotot, mint üzenetet a webszerveren, ezzel is tudtam ellenőrizni, hogy megkapja-e mindkét motor a várt jelet gombnyomáskor. Ezek az üzenetek nem jelennek meg a főoldalon, csak ha rákeresünk a hálózaton belül arra az URL-re (függvény neve). A „loop” függvény tartalmazza az ismétlődő parancsokat, itt az első parancs a webszerveren lévő HTTP kéréseket dolgozza fel és küldi a válaszokat. Ezután beolvasom a talajnedvesség mérő jelét az „analogRead” parancssal, és annak értékét kiiratom soros monitoron. Ezután deklaráltam három változót, amiből kettő a hőmérsékletet, a harmadik a páratartalmat olvassa be a DHT könyvtár beépített függvényeit használva, majd ezeknek adatait eltárolja. Lekéri a hőmérsékletet celsius-ban és fahrenheit-ben, a számításokat magától elvégzi. A változókat már egyből fel is használom egy elágazásban, ahol ellenőrzőm, hogy van-e értékük, ezzel tudom

ellenőrizni, hogy működik a hőmérő. Ha nincs értékük, válaszként a soros monitor kiírja, hogy nem tud a hőmérő mérni. Ha ezeknek a változóknak van értéke, azok egy sorban szétválasztva megjelennek a soros monitoron, de mivel ezek csak számok, kiiratom csoportosítva, hogy melyik a páratartalom, melyik a hőmérséklet. A hőmérsékletet celsius fokban iratom ki, a fahrenheit-et nem alkalmazom. Ez alatt van még egy elágazás, ami a hőmérsékletet vizsgálja és az alapján kapcsolja fel és le a LED diódát. 30°C felett kezd el világítani a LED. Végül elküldi az adatokat egy másik függvénynek. Ez az adatküldésre szolgáló függvény, itt van egy konstans karakter típusú változó, ami a cél URL-t tartalmazza, ahová rögzítésre kerülnek az adatok. Továbbá van egy szöveg típusú adat változó, ami szintén a továbbítandó adatokat tartalmazza. Végül lezajlik az adatküldés egy http kliensen keresztül és soros monitoron kiírja válaszüzenetet. Ennek a függvénynek három paramétere van, ezek az adatbázisban rögzített adatok, tehát a hőmérő és a talajnedvesség mérő adatai. Normál esetben a Laravel nem lenne képes egyszerre adatot küldeni és fogadni, ezért alkalmaztam cURL-t. „A cURL egy másik php http kérés eszköz a http kérések küldésére, legyen az GET, POST, PUT stb. A webszolgáltatásokkal való integrációhoz is használható.” (educative, 2024) API.php-n keresztül hivatkozok rá, itt adtam meg az URL-eket. Ezeket a beállításokat a kontrollerben végeztem el, miszerint egy gombcsoportnak az állapotát vizsgálom és attól függően kezdi meg a megfelelő URL-t, hogy vezérelje az ahhoz tartozó motort. Mivel két eszközre kell ezt beállítanom, kétszer szerepel ez a kódrészlet a megfelelő cél URL-el. Két gombcsoportom van a két motorra és ezeknek az állapotát egy ajax-al vizsgálom, melyet elküld a cURL-hez. Amikor az állapot ’1’ felkapcsolja a megfelelő motort, ’0’-nál lekapcsolja. Az adatbázis tábláját a modellben definiáltam, az attribútumait a migrációban, majd lefutattam a migrációt. „A Laravel migráció egy olyan utasításkészlet, amely meghatározza az adatbázissémán végrehajtani kívánt változtatásokat. Ezek a változtatások magukban foglalhatják új táblák létrehozását, meglévő táblák módosítását, oszlopok hozzáadását vagy módosítását, valamint az adatbázis kezdeti adatokkal való feltöltését.” (Saad, 2023.)



1. Elméleti összekötési sémája a rendszernek

4. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK/SAJÁT VÉLEMÉNY

Több irányba is lehet fejleszteni ezt a projektet, hogy hatékonyabb, fenntarthatóbb, vagy a felhasználónak kedvezőbb legyen a jövőben. Lehetne a rendszer komponensein fejleszteni azzal, hogy még több szenzorokat integrálunk, amik mondjuk ellenőrzik a víz és a levegő minőségét, vagy beépíthetünk fényérzékelőket és akár szén-dioxid szint mérőket, ezzel átfogóbb képet kapunk a növény környezetéről. Ezen kívül bővíthetjük a rendszer IoT technológiáját a szenzorok közötti adatkommunikációért. Továbbá lehetne fokozni az automatizálást a rendszerben, hogy esetleg emberi interferenciára már ne legyen szükség. Ezt egy folyton tanuló mesterséges intelligenciával lehet elérni. Továbbá az AI használatával elkészíthetünk adatelemzést a terméshozam optimalizálására, különféle betegségek megelőzésére és a növekedési előrejelzésre. Egy nagyobb skálán már megújuló energiaforrásokat is alkalmazhatunk (pl.: napelem), hogy csökkentsük az üvegház energiafogyasztását. Az energiafogyasztás érdekéért alkalmazhatunk energiahatékony ventilátorokat és világítást, ezzel optimalizálhatjuk a ventilátort egy teljes szellőztető rendszerre és a beépített világítás (LED) az éjszakára nyújtana segítséget, ha esetleg olyan növényt vagy növényeket tartunk az üvegházban. A távfelügyeletét ki lehetne fejleszteni egy mobil alkalmazásra, hogy a felhasználó számára bárhonnan képesek ellenőrizni és irányítani az üvegház különböző elemeit. Az adatok tárolására felhőszolgáltatásokat is alkalmazhatunk az elemzésre és szintén a távoli elérésre, ezzel biztosítva a skálázhatóságot és a megbízhatóságot. Kisebb méretekben akár okos otthoni eszközökkel is lehet integrálni, mint például okos lámpák, hőmérséklet-szabályzót vagy különböző biztonsági rendszerekkel. Olyan rendszereket is ki lehet építeni amik automatikusan képesek karbantartani az üvegházat, például ellenőrzik az eszközöket vagy szűrik a légkörnyezetet. Magas szintű skálákon ahhoz, hogy több gyártó és platform között legyen kompatibilis lehet szabványosítani az üvegházat, ettől kedvezőbb is lehet az együtt működésre a projekt.

Saját tapasztalataim és véleményem szerint, elég aggódottan álltam hozzá a projekthez. Meg voltak a kételyeim a kivitelezésében, nem voltam magabiztos a projekt teljesítésében. Eleinte tudatlannak és iránytalannak éreztem magam, nem tudtam elképzelni egy ilyen projekt elkészítését. Mindenesetre próbáltam összeszedett és türelmes lenni. Azok után, hogy Molnár József Tanárúr segített a projektben, éreztem, hogy minden egyes alkalommal, amikor foglalkozok a projekttel jobban kilátom az aktuális célt, ezáltal könnyebben tudok folyamatosan haladni.

5. Összefoglalás

Ez a projekt egy valóságos üvegházat szimulál az Arduino által készített szenzorok, egy ventilátor és egy vizet pumpáló motort felhasználva. Ez a rendszer webszerveren keresztül vezérelhető, ugyanis a ventilátor és a vízpumpa innen van kezelve. Szintén webszerveren keresztül kapja meg az Arduino által beolvasott adatokat, melyek el vannak tárolva egy adatbázisban és meg vannak jelenítve egy táblázatban a főoldalon. A projekt ihlete az édesanyámtól és a nagymamámtól jött, amikor eszembe jutott, hogy ők rendelkeznek terméssel és állatokkal, nagymamámnak számos termése és növénye egy üvegházban van. Ebből kifolyólag kíváncsi voltam, hogyan lehetséges - az iskolai eszközöket felhasználva – szimulálni és megtervezni egy fejlesztett rendszert, ami már rendelkezik automatizálással és alkalmaz IoT-t. Majd aztán remélhetőleg jobban belemerülni a világ egyik legnagyobb problémájában – a globális felmelegedés, és hogyan lehet kivitelezni egy ilyen projektet magasabb színtéren, majd azt fejleszteni vagy akár a mostaniból fejlesztési lehetőségeket levonni. Pontosabban, hogyan lehet egy ilyen projektet megvalósítani és fejleszteni, hogy az kedvezzen a környezetnek is a házon kívül és belül. Továbbá, hogyan lehet az ezzel foglalkozó dolgozók életét megkönnyíteni, így segíteni a mezőgazdaságban és kedvező munkakörülményeket teremteni. Ezzel a projekttel szándékoztam jobban elmerülni a webfejlesztés és az elektronika világában, ami előreláthatóan elősegíti a tanulmányaimat és a karrieremet a jövőben. Sikerült még többet tanulnom az elektronikáról, bizonyos érzékelők és mikrovezérlőkről, ezáltal az ilyen eszközök elméletét is sikerült jobban megértenem. A PHP nyelvet is minél jobban megértettem, különösebben az AJAX-ot és a cURL-t. Meg tudok bizonyosodni, hogy az adatbázisok létrehozása, kezelése és magának az Arduino-nak a használata sok gyakorlás után megy.

Összességében elégedett vagyok a teljes projekttel és a rendszer működésével, egy kellemes kihívásnak minősült, aminek az elején kételkedtem és gátlásosan álltam neki. Ahogy haladtam a projektben úgy egyre jobban alakult és kezdtek úgy működni a szenzorok, mint ahogy az meg volt tervezve. Számomra a projekt legnehezebb része az eleje volt, a kezdet után már lineárisan tudtam haladni előre.

IRODALOMJEGYZÉK

Kiszervezett Marketing. (2024). *Weboldal készítés*. Forrás: https://kiszervezettmarketing.hu/weboldal-keszites/laravel/

Arduino docs. (2022. július 19.). *EEPROM Library*. Forrás: https://docs.arduino.cc/learn/built-in-libraries/eeprom/

dataxo group. (2024.). *Az automatizáció jelentősége a folyamatosan változó világban*. Forrás: https://dataxogroup.com/az-automatizacio-jelentosege-a-folyamatosan-valtozo-vilagban/

educative. (2024). *What is cURL?* Forrás: https://www.educative.io/answers/what-is-curl

EVSINT. (2024.). *Disadvantages and advantages of robots in the workplace*. Forrás: https://www.evsint.com/hu/disadvantages-and-advantages-of-robots-in-the-workplace/

Laravel. (2024). *#middleware*. Forrás: https://laravel.com/docs/11.x/middleware#introduction

Laravel. (2024). *session*. Forrás: https://laravel.com/docs/11.x/session

Mérnökkapu. (2024). *Milyen típusai vannak az automatizálási rendszereknek?* Forrás: https://mernokkapu.hu/milyen-tipusai-vannak-az-automatizalasi-rendszereknek/

NYE - DUÁL. (2017). *Automatika I.* Forrás: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://zeus.nyf.hu/~elat/Automatika_I.pdf

PLC GLOBAL. (2024). *Az automatizálás előnyei és költségei*. Forrás: https://www.plc-program.hu/blog/az-automatizalas-elonyei-es-koltsegei

PullRequest. (2024.. március 15.). Forrás: https://www.pullrequest.com/blog/hardening-your-laravel-applications-tips-for-secure-session-and-cookie-management/

raptor13. (2017. december 24.). *MI az az Arduino*. Forrás: https://electro.blog.hu/2017/12/24/mi_az_az_arduino

Saad. (2023.. szeptember 26.). Forrás: https://www.cloudways.com/blog/laravel-migrations/

Sulinet. (2024). *A vezérlés működése*. Forrás: https://tudasbazis.sulinet.hu/hu/szakkepzes/gepeszet/gepeszeti-szakismeretek-3/a-vezerles-alapelemei/a-vezerles-alapelemei

Sulinet. (2024). *Kézi és önműködő vezérlés*. Forrás: https://tudasbazis.sulinet.hu/hu/szakkepzes/gepeszet/gepeszeti-szakismeretek-3/a-vezerlestechnika-alapjai/kezi-es-onmukodo-vezerles

totadavid95. (2022. február 7.). *LaravelPojektszerkezet.md*. Forrás: https://github.com/szerveroldali/leirasok/blob/main/LaravelProjektszerkezet.md

WEBiskola. (2024). *#Laravel*. Forrás: A legjobb php keretrendszerek: https://webiskola.hu/php-ismeretek/a-legjobb-php-keretrendszerek/#laravel

Wikipedia. (2024. január 9.). *Arduino*. Forrás: https://hu.wikipedia.org/wiki/Arduino