Szegedi Tudományegyetem

Informatikai Intézet

A képen szöveg, embléma, címerpajzs, jelvény látható

Automatikusan generált leírás

Szakdolgozat

Akvárium automatizálás és távfelügyelet megvalósítása

Implementation of aquarium automation and remote supervision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Készítette: |  | Témavezető: |  |
|  | Takács Ábel |  | Dr. Mingesz Róbert Zoltán |  |
|  | mérnökinformatika szakos hallgató |  | egyetemi adjunktus |  |

Szeged

2024

**A feladat kiírása**

Ahhoz, hogy egy akvárium mikroklímája megfelelő legyen, számos berendezés kell összehangoltan működjön. Ilyenek pl. a vízforgató és szűrő berendezések, fényforrások, fűtőszálak, ventilátorok és CO2 adagoló rendszerek. Ezek egy jó részét egyszerű időzítőkkel szokták vezérelni, működésüknek a monitorozása pedig általában nincs megoldva. Számos műveletet (pl. halak etetése, vegyszerek adagolása) pedig csak manuálisan lehet elvégezni. A hallgató feladata egy olyan rendszer megvalósítása, mely egyrészt vezérelni tudja a különböző eszközöket, másrészről pedig monitorozza az akvárium állapotát, a mért paramétereket pedig távolról is le lehet kérdezni. A megvalósítás során figyelni kell az üzembiztonságra, vagyis a rendszer semmilyen körülmény között nem fagyhat le.

A következő funkciók jöhetnek szóba:

* Hőmérséklet és fénymennyiség monitorozása
* Világítás ki/be kapcsolása, több világítótest
* Ventillátorok ki/be kapcsolása
* Folyadékszínt monitorozása
* Folyadékszínt utántöltése
* Vegyszerek adagolása
* Haleledel adagolása
* Zajszintmérés, hibajelzés
* Szivárgásdetektálás
* Kémiai tulajdonságok monitorozása.

**Tartalmi összefoglaló**

**A téma megnevezése:**

Akvárium automatizálás- és távfelügyeletre használható rendszer tervezése és megvalósítása

**A feladat megfogalmazása:** Egy olyan rendszer megtervezése és letrehozása, mely képes egy adott akvárium jellemzőit (Hőmérséklet, Ph, Vízszint, Fénymennyiség) monitorozni, valamint a fenntartó berendezések (Levegőztető(k), Lámpatest(ek), Etetés) időzítését ellátni. Emellett megbízható, üzembiztos és sokáig képes működni. A felhasználó telefonjáról WiFi-n keresztül bárhonnan személyre szabhatja az időzítéseket, a monitorozott adatokat megtekintheti, valamint a preferált jellemzőket beállíthatja, értesítéseken keresztül tájékozódhat a beállított takarítási/karbantartási periódusokról. Figyelmeztetést kap, ha esetleg valamely jellemző elhagyja a preferált tartományt vagy az akvárium szivárog.

**A megoldás módszere:** A rendszer funkcionalitási elvárásainak felmérése, kutatás a felhasználandó eszközök után. A rendszer megtervezése, tesztelése, majd sajátcélú otthoni használat, így is feltárva az esetleges tesztek által fel nem fedett hibákat.

**A megoldáshoz használt módszerek, eszközök:**

* ESP8266 12E NodeMCU mikrovezérlő-WiFi modul
* Visual Studio Code, PlatformIO
* KiCad áramkörtervező szoftver
* React Native keretrendszer, Expo platform
* Hőmérséklet, Ph, Vízszint szenzorok, fotorezisztor, analóg kétirányú multiplexer
* Relé modul, LCD kijelző

**Elért eredmények:** A kész rendszer üzembe helyezése és sikeres működtetése, sikeres távirányítás és monitorozás az applikáción keresztül. A rendszer lefagyás nélküli üzemeltetése több héten át. A tervezett nyomtatott áramkör és az ESP-re tervezett szoftver is bővíthető további egy szenzorral és egy időzíthető konnektorral.

**Kulcsszavak:** *Akvárium, IoT, Mobil applikáció, Mikrovezérlő*

**Tartalomjegyzék**

Tartalmi összefoglaló 3

Bevezetés 5

Irodalmi áttekintés 6

1. Hardver 6
   1. Mikrovezérlő 6
   2. Tápegység és tápfeszültség 7
   3. Szenzorok 7
   4. Aktuátorok 10
   5. Multiplexer 11
   6. Kijelző 11
2. Szoftver 12
   1. Visual Studio Code és PlatformIO 12
   2. Az ESP programozásához használt függvénykönyvtárak 13
   3. Az adatbázihoz és API-hoz használt technológiák 15
   4. React Native, Typescipt és Expo 15
   5. React Native *thrid-party* komponensek 17

**Bevezetés**

Az intelligens eszközök egyre nagyobb részben vannak jelen a mindennapi életben, hiszen nemcsak kényelmi funkciókat látnak el, de rengeteg idő is megspórolnak nekünk. Ilyen például a napjainkban már szinte mindenhol előforduló termosztátos fűtési renszer, mely lehetővé teszi, hogy a fűtés automatikusan tartsa a beállított hőfokot a lakásban. De már sok helyen előfordulnak okos hűtők is, melyeken bevásárló listát lehet írni, vagy akár még rendelni is tud nekünk. Ezek a rendszerek sokban megkönnyíthetik a mindennapi életünket. Ez utóbbi beletartozik az ’IoT’ világába, melynek lényege, hogy a berendezéseinket interneten keresztül elérhetjük, irányíthatjuk.

Az akváriumok nagyon szép díszei lehetnek bármely lakásnak, feltéve hogy rendszeresen karban vannak tartva és az érzékeny mikroklímájuk megfelelően van beállítva és szabályozva. Napi szinten személyesen tapasztalom, mennyire szép és megnyugtató látványt tud nyújtani egy jól karbantartott akvárium, hiszen az íróasztalom, ahol napjaim nagy részét töltöm, helyet ad egy 54 literes akváriumnak. Annak érdekében, hogy egy ilyen rendszer megfelelően szolgálja lakóit (és persze minket), a berendezéseknek folyamatosan összhangban kell működniük, a vílágítást érdemes időzíteni, hogy mindig ugyanakkor kapcsoljon ki és be, valamint a halakat etetni kell, az akváriumot pedig tisztán tartani. Emellett pedig a halaknak megfelelő környezet megteremtése is kulcsfontosságú, hiszen ezek az állatok érzékenyek a hőmérséklet- és Ph-változásokra.

Ennek a dolgozatnak a célja, hogy egy olyan rendszert valósítak meg, mellyel ezek a feladatok leegyszerűsödnek, néhány pedig teljesen automatává válik. Önmagában az időzítési feldatok ellátása nem egy bonyolult dolog, viszont ezeknek a személyre szabhatósága és a beállítás valós idejű frissítése már okozhat fejtörést. A monitorozás a szenzorok kezelésén és a megfelelő mérési technika kiválasztásán alapul. A leginkább elgondolkodtató rész pedig az automata etetési funkció, melynek mechanikai háttere, valamint a különféle eledelekkel való kompatibilitás is gondos tervezést igényel.

Az elején ismertetni fogom a felhasznált eszközöket, ezek működését és használatát, majd pedig bemutatom hogyan is lehet ezen eszközök felhasználásával egy az imént említett rendszert létrehozni. A fejezetek tartalmazzák a hardveres és szoftveres megoldásokat, ideértve a használt áramkörök tervezetét, a szoftverek (Mikrovezérlő és telefonos applikáció) tervét és megvalósítási módját. Valamint a felmerült problémákat és az ezekre nyújtott megoldásokat.

**Irodalmi áttekintés**

1. **Hardver**

**1.1 Mikrovezérlő**

A mikrovezérlő kiválasztásánál fontos szempont volt, hogy beépített WiFi antennával rendelkezzen, valamint az Arduino családba tartozzon, így lehetővé téve a rengeteg előre elkészített függvénykönyvtár használatát. A választásom egy **ESP8266 12E NodeMCU** [1] mikrovezérlő modulra esett. Ez a modul kifejezetten IoT területen lévő felhasználásra van kifejlesztve, valamint a lap amire integrálva van lehetővé teszi a breadboardon való kísérletezést és az egyszerű USB-n keresztül történő programozást, minimális felhasználói felülettel (reset, flash gombok).

A circuit board with many wires

Description automatically generated

* 1. ábra ESP8266 12E NodeMCU és portjai [1]

Ez egy 32 bites mikrovezérlő modul, melyben 64KB utasítás memória és 96KB adatmemória áll rendelkezésre, emellett pedig 4MB flash memória használható adatok tárolására, ez esetemben bőven elegendő a WiFi hálózat és a beállítások adatainak tárolására. Mivel ez a modul 3.3V-on üzemel viszont pár szenzor 5V-os tápfeszültésget igényel, így szükségem volt egy erre a célra megfelelő tápegységre is melynek 5V-os kimenetét lehet ennek a mikrovezérlőnek a meghajtására is használni Vin-en keresztül.

**1.2 Tápegység és tápfeszültség**

A tápegység kiválasztásánál az egyik legfontosabb tényező a méret volt, hiszen egy viszonylag kis dobozban kell majd helyet foglalnia, ugyanakkor képesnek kell lennie elegendő áramot biztosítani a mikrovezérlő, a szenzorok, a kijelző és a relék számára. Így a választásom a Mean Well   
RS-25-5 típusú tápegységre [1] esett. Ennek fontosabb adatai az alábbi táblázatban láthatóak [2].

A close-up of a power supply

Description automatically generatedA hand holding a power outlet

Description automatically generated

1.2.1 ábra Mean Well RS-25-5 tápegység [1] és az SPST csatlakozó [3]

Az RS-25-5 tápegység fontosabb jellemzői [2]

|  |  |
| --- | --- |
| DC feszültésg | 5V |
| Max. áram | 5A |
| Kimeneti áram tartomány | 0A ~ 5A |
| Bemeneti feszültésg (VAC) | 88 ~ 264V |
| Hatékonyság | 78.5 % |
| Túlfeszültség védelem | 5.75 ~ 6.75V |
| Méretek | 78 \* 51 \* 28mm |

Ezt a tápegységet felhasználva sikeresen meg tudom hajtani az összes 5V-ot igénylő eszközt, valamint a mikrovezérlőt is. A tápegység megfelelően hatékony, valamint rendelkezik túlfeszültség elleni védelemmel is, ami hálózati feszültségforrás esetén hasznos. A tápegység bementi feszültéségéhez egy SPST típusú csatlakozóval ellátott aljzatot [3] választottam, mely beépített kapcsolóval és biztosítékfoglalattal rendelkezik, így extra védelmet nyújt a rendszernek akár esetleges rövidzár esetére is, ez különösen fontos, hiszen vízközelben lesz működtetve.

**1.3 Szenzorok**

A hőmérséklet figyeléséhez egy **DS18B20** típusú vízálló digitális hőmérőt [1]. Ennek 3 kivezetése van, VCC és GND, melyeket 3.3V-ra és földre kell kötni, valamint az adat átvitelére szolgáló DAT, ezt egy tetszőleges digitális bemenetre kössük. Utóbbi vezetéket egy 4,7kΩ-os felhúzó ellenállással együtt kell bekötni, ezt a kivitelező egy a szenzorhoz mellékelt adapterrel meg is oldotta, így csak az adapter kell megfelelően bekötni.

A close-up of a circuit board

Description automatically generated

1.3.1 ábra DS18B20 digitális hőmérő adapterrel [1]

A Ph mérésére egy **Peukerty** gyártótól származó **analóg** **Ph szenzort** [2] választottam, mivel ezt volt a legegyszerűbb beszerezni itthon amazonon keresztül. Ez egy szenzor csövet és egy adapter modult tartalmaz, mellyel viszonylag egyszerűen lehet a mikrovezérlőhöz kapcsolni. Az adapter [3] 7 portjából csak 3-at fogok használni, mivel ennyi elegendő a Ph megállapításához. Fontos megjegyezni, hogy ezt a szenzort kalibrálni kell a pontos használathoz.

A soldering iron and a circuit board

Description automatically generatedA diagram of a device

Description automatically generated

* + 1. ábra Ph szenzor adapterrel [2] és az adapter adatai [3]

A vízszint mérésére egy **analóg vízszintmérő szenzort** [4] választottam. Ennek 3 kivezetése van, VCC, GND és egy S névre hallgató, ami az adatot közvetíti. A szenzor elve, hogy vízbe merítve változik az ellenállása, így az S lábon olvasott feszültségértékből megállapítható a szenzor mekkora része van vízben.

Maga a szenzor érzékelő része mindössze ~4cm hosszú, így maximum ekkora vízszintkülönbséget lehet vele mérni, de ez esetemben teljesen elegendő, hiszen egy akvárium esetén már 2cm hiányzó víz is baj lehet. Vegyük például a célakváriumot (60cm \* 30cm alapterület), ebből könnyedén kiszámítható, hogy 60 \* 30 \* 2 = 3600 cm3 = 3,6l. Belátható, hogy ez már egy ilyen kis akvárium esetében is jelentős vízveszteség. Ez a szivárgásdetektáláshoz is megfelelő mérési alapot biztosít.

A white rectangular object with red and orange stripes

Description automatically generated

1.3.3 ábra, Analóg vízszintmérő szenzor [4]

Végezetül pedig a fényméréshez választott szenzorom egy egyszerű **fotorezisztor** [5]. Ennek a szenzornak a működési elve, hogy fény hatására csökken az ellenállása, így kiszámítható milyen erősségű fény éri. Azért ezt a szenzort választottam mivel ez a legkézenfekvőbb és költséghatékonyabb megoldás, elegendő pontosságot nyújt ahoz, hogy 5 nagyobb kategóriára osszuk a mért fénymennyiséget, és ezt közvetítsük a felhasználó felé. Valamint tökéletes arra a célra, hogy esetleges kiégett lápmatest, vagy más hiba közben fellépő fényhiányt érzékelje.

A close-up of a resistor

Description automatically generated

1.3.4 ábra, Fotorezisztor [5]

**1.4 Aktuátorok**

Az automatikus etetés funkciójához egy **MG90S** [1] típusú fém mechanikájú szervómotort választottam mint beavatkozó szerv. Ez egy kis motor, melyet egy adatvezetéken keresztül PWM alapon lehet üzemeltetni, 3 vezetékkel rendelkezik [2] és 5V-on üzemel. A választott motor 180 fokot képes fordulni, így elegendő mechanikai behatást tud nyújtani egy rotációs alapon működő etetőhöz. Valamint elegendően nagy erőt tud kifejteni (1,8kgf \* cm) ahoz, hogy az etetőt elforgassa.

A small electronic device with wires

Description automatically generatedA close up of text

Description automatically generated

1.4.1 ábra, MG90S szervómotor [1] és a vezetékek értelmezése [2]

Az időzítésekhez egy **4 csatornás relé modult** [3] választottam. A relék működési elve, hogy amikor az irányításra szolgáló bemenetén logikai magas értéket kap akkor a relében található elektromágnes bekapcsol, kapcsolva a benne lévő mágneses kapcsolót. Így lehetővé téve hálózati feszültség (~220V) kapcsolását a mikrovezérlő feszültségével (3.3V). Ehez a modulhoz egy 3 kivezetésű konnektorállást választottam, így az akvárium 3 berendezését rá lehet csatlakoztatni (3 csatorna elegendő a különféle berendezéseknek).

A close-up of a circuit board

Description automatically generated

1.4.2 ábra, 4 csatornás relé modul [3]

**1.5 Multiplexer**

A választott mikrovezérlőnek csupán 1 ADC-vel ellátott analóg portja van, viszont nekem 3 analóg szenzorom, így szükségem volt egy analóg multiplexerre. Erre a célra a CD4052 2 \* 4 csatornás analóg multiplexer/demultiplexer-t [1] választottam. A multiplexer lényege, hogy egy bemenetét vezérlővonalak segítségével képes több kimenete egyikére kapcsolni. Esetemben 2 vezérvonal  
(A, B) és 2 \* 1 bemenet (x, y) valamint 2 \* 4 kimenet (0-3x, 0-3y) van [2]. Ezen keresztül tudom a szenzorok jeleit az analóg portra kapcsolni, valamint a relék vezérlését is elvégezni.

A black and silver electronic chip

Description automatically generatedA diagram of a circuit board

Description automatically generated

1.5.1 ábra, CD4052 multiplexer [1] és lábkiosztása [2]

**1.6 Kijelző**

A rendszernek szüksége van valamiféle adatmegjelenítő felületre is, már csak önmagában a WiFi hálózat konfigurálásához szükséges adatok megjelenítésére, hiszen a mikrovezérlő önmagában nem képes adatot megjeleníteni. Erre a célra egy 20 \* 4 karakteres LCD kijelzőt [1] választottam.

A close-up of a circuit board

Description automatically generated

1.6.1 ábra, LCD kijelző I2C interfésszel [1]

Az LCD kijelzőt a gyártó már felszereli egy I2C interfésszel, így a kommunikáció és bekötés is sokkal egyszerűbb. A kijelzőnk ezzel az interfésszel összesen 4 portot kap, természetesen itt is van VCC és GND emellett pedig egy SCA (adat) és egy SCL (órajel) pineket találunk. Utóbbi két vezetéken keresztül van megvalósítva a szinkron soros adatátvitel.

**2 Szoftver**

A dolgozat megvalósításához több különféle fejlesztői eszközt és kész függvénykönyvtárat használtam, ebben a fejezetben ezeket mutatom be, többek között az ESP mikrovezérlő programozásához használt környezetet és könyvtárakat, valamint a mobil alkalmazáshoz felhasznált technológiákat.

**2.1 Visual Studio Code és PlatformIO**

Az ESP mikrovezérlő programozása többféle környezet felhasználásával is történhet, egyik legelterjedtebb talán az Arduino IDE, azonban ez a környezet számomra alkalmatlannak bizonyult, mivel viszonylag nagyobb forráskódot kell létrehoznom, ennek könnyű átláthatósága és karbantarthatósága érdekében inkább a VSCode IDE mellett döntöttem, mely a Microsoft által fejlesztett környezet, a Visual Studio kicsit lebutított, több platformon is elérhető változata. Az IDE PlatformIO (továbbiakban PIO) bővítménye [1] kifejezetten mikrovezérlők programozásához van kifejlesztve. Itt kiválasztható a programozni kívánt mikrovezérlő (ez esetemben egy ESP8266 12E NodeMCU), majd a kiválasztás után a bővítmény automatikusan telepíti a szükséges függőségeket, mely esetemben egy *esptool* névre hallgató python program, ami ESP alapú mikrokontrollerek felprogramozására lett létrehozva. Ezt a PIO automatikusan futtaja amikor az IDE-ben a feltöltés opciót választjuk, valamint a feltöltés státuszáról az *esptool* részletes információkat biztosít a terminál ablakban. A PIO-hoz tartozik egy *platformio.ini* fájl, melyben a programozni kívánt mikrovezérlő mellett a használt függvénykönyvtárakat és fordítási preferenciákat is be lehet állítani. A programunkat fordítani az ablak alján bal-középen található pipa ikonnal tudjuk, a feltöltést pedig a mellette lévő nyílra kattintva tehetjük meg. Emellett a bővítmény biztosítja számunkra a soros monitorozáshoz szükséges feltételeket, melyet a Serial Monitor fül alatt érhetünk el.

A screenshot of a computer

Description automatically generated

2.1.1 ábra, VSCode PlatformIO bővítménnyel (PIO kezdőlap és .ini fájl) [1]

A bővítmény által kezelt mappaszerkezetben két fő mappa van, az *src* és a *lib*, az src-ben a main.cpp fájlunk található, benne a szükséges setup és loop függvényekkel (ezt a bővítmény automatikusan legenerálja). Valamint utóbbi mappa a saját készítésű könyvtárak tárolására szolgál, ezeket a bővítmény fordításkor automatikusan összelinkeli a main fájllal. Ugyanakkor a bővítmény lehetőséget biztosít külső könyvtárak használatára is, erre használhatjuk a .ini fájlt, ahol a *lib\_deps* változóhoz kell hozzárendelni a használni kívánt könyvtárat, vagy egy egyszerűbb és felhasználóbarátabb megoldás, ha a PIO Home fülön található Libraries menüpont alatt választjuk ki a használni kívánt könyvtárat, itt lehetőségünk nyílik új könyvtár keresésére és hozzáadására, valamint az eddig telepített könyvtárak menedzselésére is. Ezeken felül természetesen még rengeteg funkcióval fel van szerelve ez a bővítmény, de itt most csak az én esetemben fontosabb eszközöket soroltam fel.

**2.2 Az ESP programozásához használt függvénykönyvtárak**

Az ESP alapú mikrovezérlők előnye, hogy kompatibilisek az összes Arduino alapú függvénykönyvtárral, így ezen könyvtárak széles választéka közül szabadon tudtam válogatni. Ebben a fejezetben felsorolásszerűen minimálisan részletezve mutatom be a használt könyvtárakat.

* **Arduino –** Az Arduinon használható függvényeket tartalmazza, első sorban a Serial Monitor működtetéséhez szükséges függvények miatt van rá szükség.
* **ESP8266WiFi** – WiFi kapcsolat létesítésében játszik fontos szerepet, egyszerűen csatlakoztatja a mikrovezérlőt a WiFi hálózathoz, valamint a WiFi kapcsolat irányításában vesz részt pl.: kapcsolat státusza, kapcsolat bontása.
* **ESP8266 HTTPClient –** HTTP kérések küldésében játszik szerepet, ennek a segítségével küldhetőek kérések, állíthatóak a header-ek valamint a válasz is ezen keresztül érhető el.
* **ArduinoJson –** JSON fájlok kezelésére használható (szérializáció, deszérializáció).
* **NTPClient –** A pontos idő hálózaton keresztül történő szinkronizálására szolgál, segítségével csatlakozhatunk egy közeli NTP szerverhez, és letülthetjük a pontos időt, állítható offszettel.
* **WiFiUdp –** Az NTPClient használatához szükséges UDP protokollt kezeli.
* **TimeLib –** A mikrovezérlő belső RTC-jéhez való hozzáférést és ennek könnyű használatát biztosítja, használatával lekérdezhető pl. az aktuális órák, percek, másodpercek, vagy akár az epoch idő is.
* **EEPROM** – A mikrovezérlő memóriájához (EEPROM/flash) biztosít egyszerű hozzáférést író és olvasó függvények megvalósításával.
* **LiquidCrystal\_I2C** – I2C interfésszel felszerelt LCD kijelzők egyszerű használatához biztosít függvényeket, pl.: adott sor írása, kurzor pozíciójának beállítása, a kijelzőn lévő adatok törlése.
* **OneWire** – Egyvezetékes szenzorok jeleinek olvasásához tartalmaz függvényeket, pl. a szenzor kiválasztása, az adat olvasása.
* **DallasTemperature** – A DS18B20-as egyvezetékes hőmérséklet szenzor egyszerű kezeléséhez biztosít függvényeket (a OneWire ennek a függősége).

**2.3 Az adatbázihoz és API-hoz használt technológiák**

A használt adatbázis egy MariaDB alapú adatbázis, melynek tárhelye egy saját szerveren található. Az adatbázis kezelését MySQL-en keresztül valósítom meg, mely egy adatbázis kezelő nyelv. Ennek segítségével tudunk az adatbázisba adatot beírni, törölni, módosítani azt valamint kinyerni az adatokat. Az adazbázis az adatokat táblákban tárolja, melyekben rekordoknak hívjuk a tábla által meghatározott adatszerkezetből előálló adatokat, a tábla egy sora egy rekord.

Az adatbázist természetesen valahogyan kezelni kell, tehát kell egy a szerveren futtatható program amivel a különböző adatbázisműveleteket végre lehet hajtani. Erre a célra a Php (Hypertext Preprocessor) szerveroldali szkriptnyelvet választottam, ez utóbbi azt jelenti, hogy a kód a szerveren fut, így a felhasználó számára teljes mértékben rejtve marad, csupán a generált válaszok jutnak el a klienshez. Ez a nyelv széleskörűen elterjedt első sorban webes alkalmazások területén, hiszen beágyazható HTML kód, valamint dinamikusan is képes HTML kódot generálni. Emelett pedig rengeteg beépített nyelvi funkcióval és függvénykönyvtárral rendelkezik különféle problémákra. Emelett jól moduralizálható, támogatja az OOP (Object Oriented Programming) megközelítést is.

Az adatbázis hozzáféréshez és kezeléshez a *mysqli* [1] könyvtárat választottam, ennek segítségével viszonylag egyszerűen hajthatóak végre adatbázis műveletek. Valamint a könyvtár lehetővé teszi úgynevezett paraméterezett lekérdezések végrehajtását amely egy fontos biztonsági vonalat ad a programozóknak az SQLI (SQL Injection) alapú támadások ellen.

**2.4 React Native, Typescipt és Expo**

A mobilalkalmazás fejlesztéséhez olyan nyelvet és rendszert próbáltam találni aminek használatában van tapasztalatom és léteznek felhasználható könyvtárak/csomagok amiknek segítségével egyszerűsödhet az app létrehozása, így a választásom a React Native-ra esett melyet az Expo keretrendszerrel használva egyszerűen, az újrafelhasználhatóságot és karbantarthatóságot szem előtt tartva lehet multiplatform alkalmazásokat fejleszteni akár web, IOS vagy Android rendszerre.

A React Native a Facebook által fejlesztett nyílt forráskódú keretrendszer, mely lehetővé teszi a multiplatform alkalmazásfejlesztést egy forráskód felhasználásával. Ez azt jelenti, hogy a forráskódot elég egyszer megírni, majd ezt lehet IOS, Android, web de még akár MacOS rendszerekre is buildelni. Másik előnye, hogy magába foglalja a React komponensorientált rendszerét, mellyel az újrafelhasználhatóság

Magát a keretrendszert lehet Javascript vagy Typescript nyelven is használni, itt a Typescript-re esett a választásom, hiszen ez magába foglalja a Javascript előnyeit kiegészítve a típusossággal, melynek segítségével könnyebben karbantartható kódot lehet írni, valamint sok a Javascriptben csak futás időben előlépő hiba itt már fejlesztési időben előjön. Maga a Typescript egy fordító segítségével Javascript kóddá fordul, mely azért is hasznos, mert ez az eredményül kapott Javascript kód ember számára szinte olvashatatlan, így senki nem tudja megfejteni mit is csinál a kódunk, ha esetleg sikerülne hozzáférnie.

Magát a fejlesztést pedig az Expo (Exponent) keretrendszeren belül folytatom, ez a keretrendszer sok hasznos funkciót bizotsít, a különféle csomagok menedzselése mellett fejlesztési idejű fordításra (a program adott komponensét újrafordítja a fájl mentése esetén), buildelésre és emellett pedig push-notification kezelésre is alkalmas, melyet egy Expo szerveren keresztül tudunk az alkalmazáshoz eljuttatni azonosító alapján. Az Expo-cli [1] segítségével terminálon keresztül vezérelhető a fejlesztési folyamat.

A screenshot of a computer

Description automatically generated

2.4.1 ábra, Expo-cli [1]

Az ábrán látható a program futás közben, az Expo projekt gyökérmappájában az *npx expo start* parancs segítségével indítható a fejlesztési folyamat. Ilyenkor a keretrendszer ellenőrzi a használt csomagokat, majd generál egy QR kódot, amit a mobil eszközünkkel be tudunk olvasni, és ha telepítve van az Expo Go alkalmazás akkor először az adott platformra lefordul a forráskódunk, majd elindul az alkalmazás az eszközünkön.

Látható, hogy a keretrendszer számos fejlesztést támogató funkcióval van felszerelve többek közt ilyen a debugger, mellyel debuggolási funkciókat biztosít, valamint a progrmunkban lévő logolások is itt jelennek meg. De látható az is, hogy lehetőségünk van Android, Web-es vagy emulátoron kersztül történő fejlesztéshez is (emelett még támogatott rendszer az IOS, Windows és MacOS), valamint bármikor újrafordíthatjuk a teljes alkalmazást.

Összességében ezek a technológiák bizonyultak a legjobbnak számomra, hiszen így olyan technológiával tudok dolgozni amiben van tapasztalatom, másfelől pedig az alkalmazás amit készítek elérhető lehet mind Android mind IOS eszközökre, fontos ugyanakkor megemlíteni, hogy a dolgozat tartalmában csak Android eszközökre fejlesztek, ezt a forráskódot a későbbiekben lehet IOS eszközökre is kiterjeszteni.

**2.5 React Native *thrid-party* komponensek**

Mint korábban említettem azért választottam a React Native keretrendszert, mivel számos már előre elkészített nyílt forráskódú komponens található hozzá, amivel a fejlesztési időt jelentősen le lehet csökkenteni. Ebben a fejezetben ezeket fogom felsorolásszerűen röviden ismertetni.

* **React-native-async-storage** – Az eszköz tárhelyéhez való hozzáférést és azzal való műveletek (írás, olvasás) elvégzését biztosítja aszinkron függvényekkel.
* **React-native-community/datetimepicker –** Dátumés idő kiválasztásához alkalmas komponenst biztosít, mely az adott eszköz operációs rendszerének megfelelő választófelületet jelenít meg.
* **React-navigation/bottom-tabs –** A navigációhoz nyújt segítséget, egy az ablak alján elhelyezhető navigációs vezérlő segítségével.
* **React-navigation/stack –** Navigációs komponens, alapvető navigációs funkciókat lát. Ennek segítségével lehet navigálni pl. gomb megnyomására.
* **Expo-notifications –** Expo keretrendszer segítségével küldhető értesítések használatára kifejlesztett függvénykönyvtár, ezzel elkerülhető a Firebase Cloud Messaging vagy az Apple értesítésszolgáltató használata, helyette egy Expo szerverről kerül kiküldésre az értesítés.
* **Expo-secure-store –** Egy titkosítással ellátott tárhelyet biztosít az eszközön, érzékeny adatok tárolására.
* **Expo-status-bar –** Egy a töltés folyamatát jelző animációs komponenst biztosít.
* **React-native-animatable –** Komponensek animálását teszi lehetővé.
* **React-native-bouncy-checkbox –** Egy animált checkbox komponenst biztosít amely személyre szabható.
* **React-native-chart-kit –** Különféle diagramok egyszerű létrehozásához szükséges komponenseketr tartalmaz.
* **React-native-dropdown-select-list –** Egy legördülő lista komponenst tartalmaz, mely személyre szabható.
* **React-native-keyboard-aware-scroll-view –** Olyan görgethető megjelenítőfelületet tartalmaz, melynek megjelenítését a billentyűzet jelenléte nem befolyásolja.
* **React-native-vector-icons –** Különféle ikonokat tartalmaz különféle családokból.

**3 Szoftveres megoldások**

// GITHUB TOKEN:

**ghp\_tbGUSjXY1OcpXwEFzpmG2drFl0kMJl43RJhv**

4 ESP8266 (Szoftveres megoldások)

**5 Adatbázis és API**

Az adatbázis egy fontos eleme a projektnek, hiszen itt kerülnek eltárolásra az adatok és a beállítások is, valamint innen tudjuk a telefonos applikációban megjeleníteni a szenzorok adatit is. Az itt eltárolásra kerülő adatok a felhasználó adatai (email cím, név, jelszó), az akvárium (egy felhasználónak lehet több is) adatai (azonosító, név, méretek), a konfigurációs adatok az adott akváriumhoz valamint az adott akvárium szenzorai által mért adatok. Utóbbiakat a mikrovezérlő adott időközzel mintavételezi (ez külön konfigurálható, max 15 perc), és feltölti azokat az adatbázisba. Ezen felül az értesítések a mikrovezérlőről érkeznek a PhP alapú API felé, ahonnan tovább lesznek küldve a telefonos applikációba úgynevezett ’push notification’ formájában.

**5.1 Az adatbázis**

Az adatbázist belső kapcsolatait leíró egyed-kapcsolat diagramot a // TODO melléklet tartalmazza. Ennek a digramnak a relációsémákká való leképzése itt látható (az Outlet helyett az OL rövidítést használjuk):

User(email, firstName, lastName, deviceToken, authToken);

Aquarium(Id, name, length, depth, height, fishCount);

HaveAquarium(*email*, *Id*);

Config(*Id*, minTemp, maxTemp, minPh, maxPh, OL1On, OL1Off, OL2On, OL2Off, OL3On, OL3Off, waterLvlAlert, prefLight, feedingTime, foodPortions, filterClean, waterChange, sampleTime, lastModifiedDate);

SensorSample(*Id*, When, temperature, Ph, waterLvl, lightAmount);

Ezek alapján már egyértelműen felvehetjük az adatbázisunk tábláit, melyeknek SQL leírása megtalálható a // TODO számú mellékletben. Értelemszerűn összekapcsoljuk a táblákat, hogy esetleges felhasználó törlés esetén az ehez tartozó adatok is törlődjenek, valamint ha mondjuk a felhasználó kitörli egyik akváriumát akkor az ahoz tartozó konfigurációk és minták is törlődjenek.

Az akvárium azonosítója automatikusan generálódik majd, amikor a rendszer először csatlakozik az adatbázishoz, ez a WiFi hálózat megadása után meg is történik (a rendszer létrehoz egy akváriumot az adatbázisban alapértelmezett paraméterekkel, melyek később módosíthatók), ekkor a felhasználó a rendszer kijelzőjén láthatja majd a kiosztott azonsítót, ezt a telefonos applikációba való regisztrációkor meg kell adni, vagy ha már regisztrált (ez implikálja, hogy már van egy rendszere), ilyenkor az új akvárium felvételéhez kell megadni az azonosítót. Ennek megadása nélkül nem köthető össze a felhasználó és az akvárium így ez kötelező. Ezután a felhasználó már tetszőlegesen állíthatja az akvárium további adatait.

Minden konfigurációnak van egy azonosítója, amely az akvárium azonosítójával egyezik meg, külső kulcsot alkotva, így minden akváriumhoz csak egy konfiguráció tartozhat és fordítva, melyet tetszőlegesen lehet állítani az applikációban bejelentkezés és az adott akvárium kiválasztása után.

**5.2 Php API**

A screen shot of a computer

Description automatically generatedAz API (Access Point Interface) minden alkalmazás esetén létfontosságú, hiszen ezen keresztül történnek az adatok kezelése és a kérések kiszolgálása is, ez adja a kapcsolatot az alkalmazás és az adatbázis közt. Esetünkben ez az API Php alapú, és kezeli mind a mikrovezérlő, mind a telefonos applikáció által küldött és fogadott adatokat. Implementálásra kerülnek benne a használt adatszerkezetek modelljei, az ezek alapján az adatbázissal interaktáló DAO (Data Access Object) valamint a különféle kérések kiszolgálására létrehozott vezérlések (minden kérés egy-egy különböző fájlba fut be, mely kezeli azt). Az API mappaszerkezetének felépítése a 5.2.1-es ábrán látható. Természetesen a CONTROLS mappa tartalma jelentősen több, hiszen ide kerül minden funkció megvalósítását ellátó fájl is, de szemléletesség céljából itt ennyi is elegendő. 5.2.1 ábra

**5.3 Funkcionális követelmények**

* Az API-nak minden lehetséges kérést ki kell tudnia szolgálni, hibás kérés esetén pedig jelezni a hibát egy hibaüzenettel.
* A jelszavak titkosítva kerüljenek átadásra, valamint az adatbázisban is titkosítva legyen tárolva.
* Lehetőség szerint minimalizáljuk a lekérdezéseket, a már lekérdezett és használni kívánt adatokat tároljuk el a szerveroldalon.
* Lehetőség szerint minimalizáljuk a mikrovezérlő felől érkező kéréseket, beüzemelés után csak a minták feltöltése, értesítések és esetleges hibák okozzanak kéréseket.
* A frissített beállítások, melyek befolyásolhatják a mikrovezérlő működését lehetőleg azonnal kerüljenek frissítésre az eszközön is.

**Mellékletek**

A diagram of a computer program

Description automatically generated

1. Melléklet - A koncepció blokkvázlata