ÓBUDAI EGYETEM

Kandó Kálmán Villamosmérnöki Kar
Elektronikai és Kommunikációs Rendszerek Intézet

Mikroelektronikai és Technológia Tanszék

SZAKDOLGOZAT

OE-KVK
2022.

Hallgató neve: **Kreuz Fanni**

Hallgató törzskönyvi száma: T007314/FI12904/K

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Mikroeletronikai és Technológia Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: Kreuz Fanni
Szakdolgozat száma: SZD22032912444178
Törzskönyvi száma: T007314/FI12904/K **Neptun kódja:** [redacted]

Szak: villamosmérnöki (magyar nyelven)
Specializáció: Mikroelektronika és technológia specializáció

A dolgozat címe: Növénynevelő kamra
A dolgozat címe angolul: Phytotron
A feladat részletezése: Készítsen el egy elektronikus érzékelőkkel és beavatkozókkal eláttot kísérleti növénynevelő kamrát. Legyen benne többek között világítás és megvilágításmérés, hőmérséklet- és páratartalommérés és számítógépes kapcsolat.

Intézményi konzulens neve: Horváth Márk

A kiadott téma elévülési határideje: 2024. 06. 30.
Beadási határidő: 2022. 05. 15.

A szakdolgozat: Nem titkos.

Óbudai Egyetem Kandó Kálmán Villamosmérnöki Kar Elektronikai és Kommunikációs Rendszerek Intézet Igazgató

Kiadva: Budapest, 2022. 03. 31. P.H.

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM

Kandó Kálmán Villamosmérnöki Kar
Elektronikai és Kommunikációs Rendszerek Intézet
Mikroelektronikai és Technológia Tanszék

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2022. 05. 14

Kreisz Gábor
hallgató aláírása

Tartalomjegyzék

Bevezetés

Szakdolgozatnak a növénynevelő kamrát választottam, illetve az abban használatos érzékelés többmódú felhasználását. Ez véleményem szerint egy igen hasznos témakör, amely több módú felhasználást tesz lehetővé. Akár a kamrában szabályozott környezetben, akár attól függetlenül otthoni, esetleg akváriumi felhasználásra.

A növények nevelése, és az ahhoz szükséges ideális környezet kialakítása, szabályozása nem csupán labori, illetve nagyipari körülmények között elengedhetetlen, de az otthonokban is. Az ember már azzal szabályozza egy növény környezetét, mikor a napfényes ablakhoz közelebb viszi szeretett virágát, hogy több fényhez jusson, esetleg pont hogy el az erős napfény közeléből, ami a zárt ablak üvegén keresztül megégetheti a leveleket.

Azonban nem csak a növények számára elengedhetetlen a környezet ellenőrzése, és a hőmérséklet és páratartalom mérése. A mindennapokban is lényeges ezt a két paraméterét ismerni a levegőnek. Akár otthoni növényekről beszélünk, akár pusztán csak magáról a szobáról, a levegő páratartama az emberi test számára is fontos szempont. A túl alacsony páratartalom kiszáríthatja a szemet, és a légutakat, a túl magas páratartalom pedig penészedést indíthat el, vagy segíthet elő a falakon. A penészgomba, ha még nincs is látható jele a kialakulásának, mégis a levegőbe jutva, azt belélegezve rendkívül ártalmas az emberi szervezetre nézve.

Ilyen szempontból a mindennapokban is előnyös az ellenőrzött környezet, de növényeket nem csak szimulált környezetben, vagy szobában lehet tartani. Sok kialakítású és fajta akvárium is van, ahol a betelepített növények ki tudják idővel alakítani a saját mikroklímájuk, amibe szintén be lehet helyezni felülre szellőztetést, fénytesteket, fűtőtesteket vagy párásítást is.

Projektemben leginkább a pára, illetve hőmérsékletre fókuszáltam, illetve növénynek az orchideákat választottam.

Az orchideák manapság egyre közkedveltebb, és elterjedt virágai a családi házaknak. A legelterjedtebb lepkeorchideának összesen kettő színárnyalatú vadonélő változata

létezik: fehér, és lila. Elterjedésük viszont a különböző nemesítési eljárással azt eredményezték, hogy lassan már többféle növény, és virágméret, alak, és színárnyalat található meg. Népszerűségüket pedig az is rendkívül megemeli, hogy az orchideák virágzási ideje rendkívül magas. Egy jól tartott, erős, tápoldatozott lepkeorchidea fél éven keresztül is tarthatja a virágait, mielőtt lehullajtaná azokat, és a virágzásból a növekedési szakaszába lépne ismét. Ilyenkor a növény nem arra fókuszál, hogy virágot hozzon, sokszor teljesen vissza is szárítja az előző virágszárát. Növekedési szakaszban a gyökérzet, és új levelek, hajtások megjelenése jellemzi. Ez a szakasz fajtától függően szintén eltarthat fél évig, de van amelyik nagyon hamar újabb szárat kezd kihozni. Ezt is nagyban befolyásolják a tartási körülmények, a fénymennyiség, és hőmérsékletváltozás.

Például vannak fajtái, egy példát említve a Dendrobium Nobile, melynek virágzása teleltetéshez köthető, azonban ehhez nincs szükség kontrollált környezetre, legfeljebb olyan mértékben, hogy a fűtéssel ne engedjük a növényt 10°C alá süllyedni, azonban ebben az időszakban nincs szüksége fénytestekre, párára vagy nedvességre.

Azonban a többi fajnál is közrejátszhat a hőmérsékletkülönbségek, sokkal könnyebb virágzást elérni, ha legalább 5-8°C különbség van az éjszakai és nappali órák között.

Ha fénytesteket használunk sem mindegy, milyen LED-ekkel dolgozunk. A projekthez hidegfehér katód fénycsövet, mivel a vörös és kék ledek spektruma valójában megfelelő, viszont pusztán a virágzási fázishoz, mikor a növény már ki is hozta a virágait. Azonban nem segít ez előidézni a növény számára ezt a fázist.

Ezek a fajta virágok trópusi környezetben epifita életmódot folytatnak, így számukra nincs szükség öntözőrendszer kialakítására sem. Sajnos a nemesítés hátránya, hogy érzékeny növények a gombás, vírusos és bakteriális megbetegedésekre, így sokan nem tudják, de merítéses módszerrel ajánlott csak az öntözés, ahol magát az átlátszó cserepet merítik vízbe. Így nem juthat a tőhöz, vagy a levelek tövéhez.

Az orchideák közege is teljesen eltér a normális talaj menti növényekétől. A lepkeorchideák közege fenyőkéreg, néha kókuszrosttal, agyag, vagy perlit darabokkal. Így nedvességmegtartásuk rövididejű, öntözéskor nem a közeg állapotát kell figyelembe venni, hanem a gyökérzet színét. Így az öntözésen túl az is feleslegessé válik számukra, hogy szenzorokkal a közeg mélyebbi nedvességtartamát ellenőrizzük.

Így tehát ehhez a növényhez három elengedhetetlen környezeti tényező kialakítására van szükség:

- Fénymennyiség
- Páratartalom
- Hőmérséklet

Kamra, és környezete:

Maga a tervezett kamra egy megközelítőleg 20x20x40cm-es nagyságú, műanyag falú kamrának készül, amelynek lapjai ragasztásos illesztéssel valósulnak meg.

A belső megvilágítást két darab 30cm hosszúságú 5W-os hideg fehér, katód fénycső oldja meg, míg a szellőzésért két ventilátor (egy felül, egy alul) felel a levegő mozgatása érdekében.

Maga a plexi, vagy üveg alapú kamrafal kivitelezése abban előnyösebb, hogy az átlátszóság miatt lehetőséget biztosít az odabenti környezet külső megfigyelésére anélkül, hogy a kamrát felnyitva kellene ugyanezt tennünk, ezzel megzavarva a bent optimalizált környezetet.

Ezen túl maga a plexi mint anyag költséghatékony, könnyen megmunkálható és könnyű mozgathatóságot biztosít.

Hátránya ugyan, hogy viszonylag alacsony olvadásponttal rendelkezik (130-140°C), így figyelni kell arra, hogy a fűtőtest, és a kamra teteje közti érintkezés ne hevülhessen fel.

Magával a kamrával pedig kontrollált és ellenőrzött környezettel pedig optimalizálni tudjuk a különböző növények igényeinek megfelelő környezetet, így azok növekedését lehetünk képesek vele serkenteni, virágzó növények esetében, mint az orchideák,

megfelelő tápoldatozás mellett a szárak, és azon kihozott virágok számát serkenthetjük, a szirmok, így a teljes virág méretét, és a színek intenzitását, mivel maga a szín is gyengébb fénymennyiség mellett fakóbb színekben fog pompázni.

Gyümölcsök, zöldségek esetében pedig szintén a terméssz̈amot képes gyarapítani, szintén megfelelő tápoldatozás mellett, illetve az ízletesebb, élénkebb színű és nagyobb méretű terméseket lehet majd leszüretelni.

A növénynevelő kamrák mérete az ehhez hasonló, hordozható, és egy növény férőhelyű kis tesztkamráktól kezdve az egészen több száz literes kamrákig terjedhetnek, hisz az ilyen egységekben különféle éghajlati viszonyokat lehetséges előidézni.

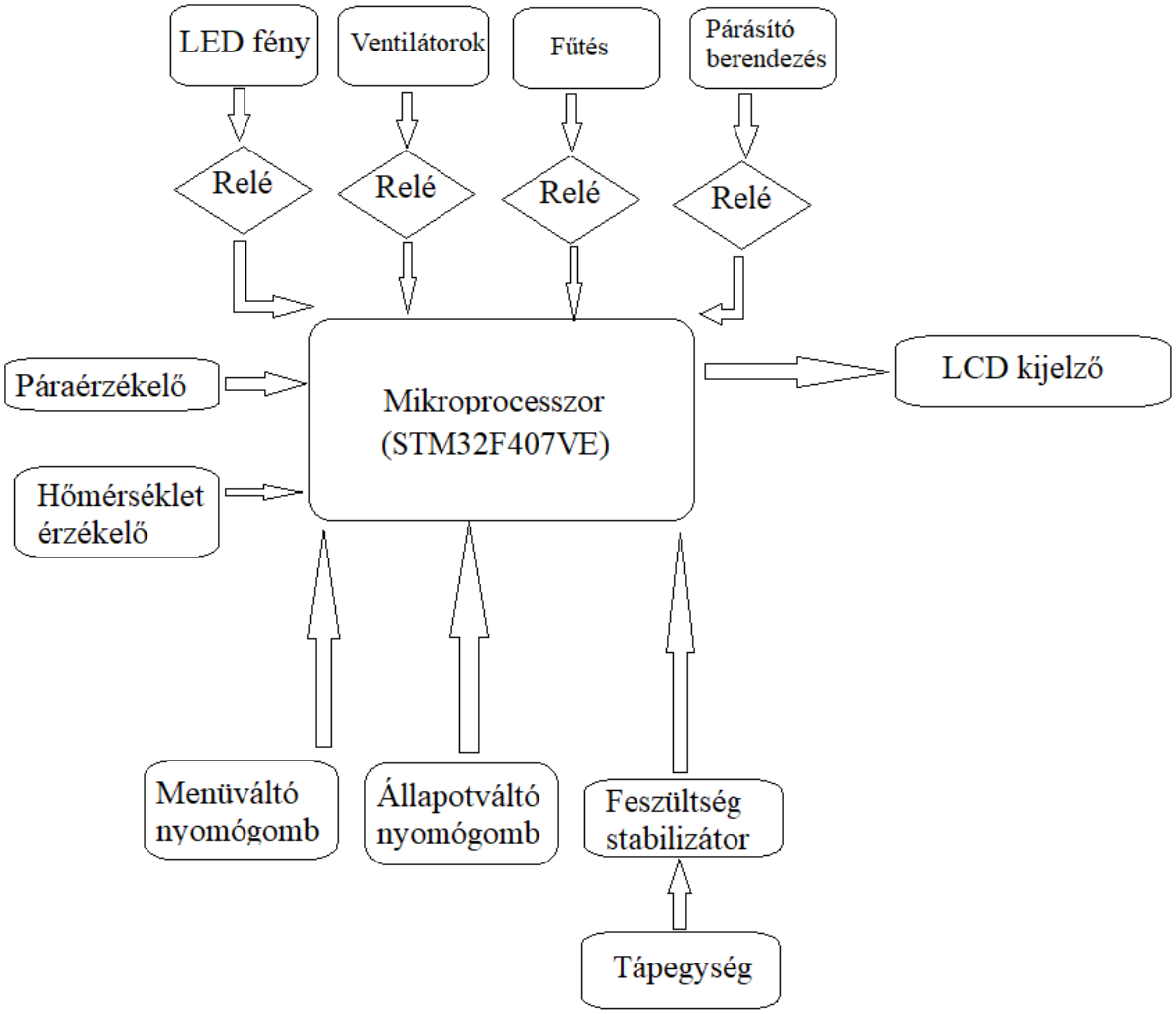
Felhasználási lehetőségek közül néhány példa:

- tavasz-nyár klímakamra
- ősz-tél klímakamra
- csíráztató kamra

Kamra költségvetése:

Felhasznált eszközök	Darabszám	Ráfordítás
STM32F407VE	1db	3 000 Ft
DHT11 páraérzékelő szenzor	1db	800 Ft
Plexiüveg lap	5db	~ 5 600 - 12 000 Ft
LM35 hőmérsékletérzékelő	1db	900 Ft
30cm 5W-os fehér led fénytest	2db	~ 3 200 - 5 000 Ft
Ventilátor - 12cm	2db	2 800 Ft
Összesen:		~ 16 300 - 24 500 Ft

A projekt blokkvázlata:



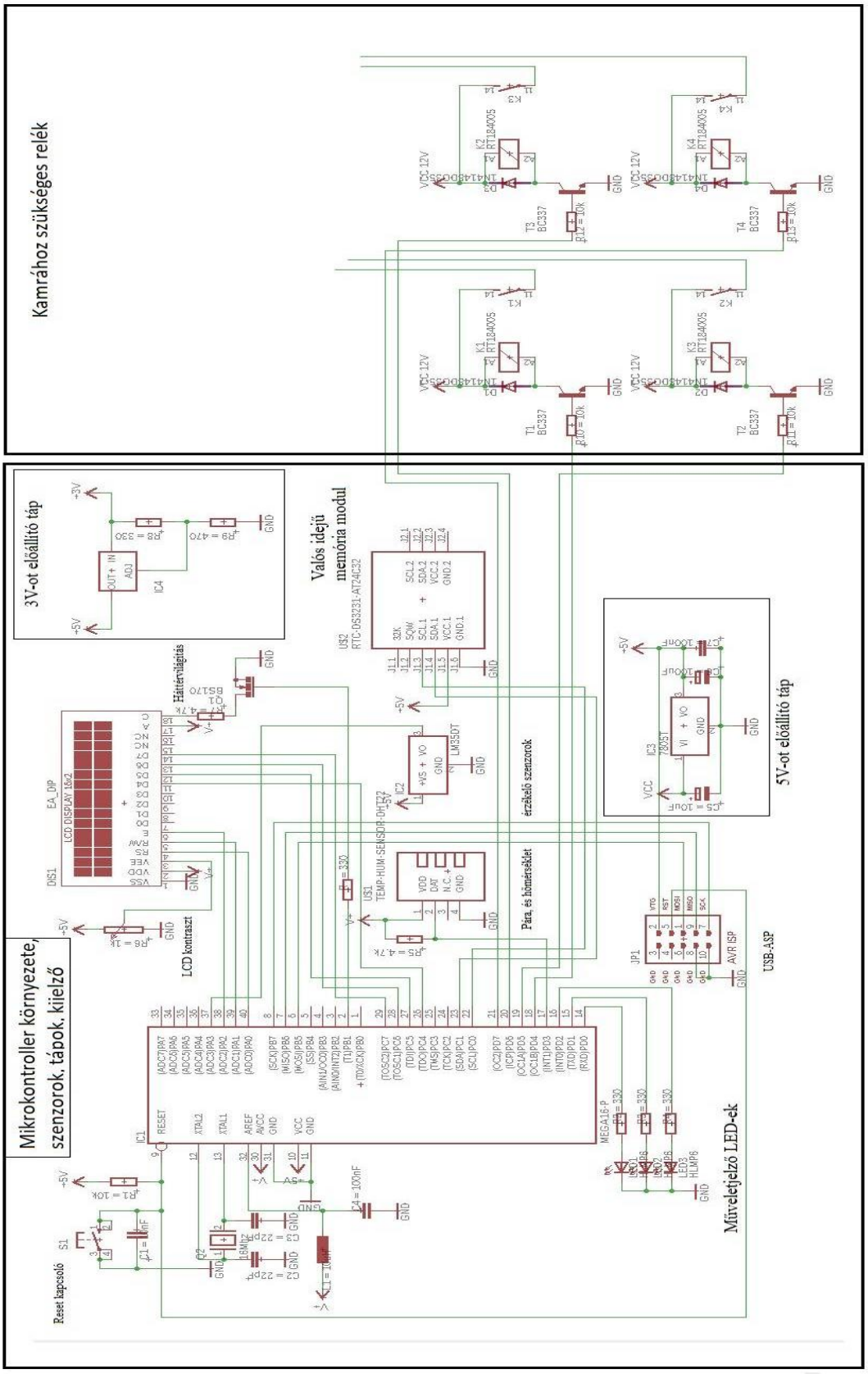
1.ábra A szakdolgozat projektjének blokkvázlata

I. A projekt első szakasza:

A szakdolgozat projektjének első felében létrehoztam a mikrokontrollert és környezetét, illetve az azzal való összekötését a fénytesteknek, fűtésnek, párásításnak és légmozgásnak.

Megismerkedtem, és mélyebben belemerültem az érzékelők működési elvébe, felépítésébe, kommunikációs módjukba a mikrokontrollerrel. A kapcsolási rajz megvalósításához pedig az Eagle nevű programot használtam, melyet a tanulmányai során már megismerhettem, így csak az akkor megszerzett tudásomat kellett csak felfrissíteni az elkészítéshez.

I.1 A projekt első felében létrehozott kapcsolási rajz:



2. ábra A projekt első felében elkészített kapcsolási rajz

I.2 ATmega16 port kiosztás:

Lábkiosztás	Megnevezés	Bekötés
1	PB0	-
2	PB1	LCD kijelző C lába (18) - Háttérvilágítás
3	PB2	-
4	PB3	-
5	PB4	-
6	PB5	USB-ASP port 1
7	PB6	USB-ASP port 9
8	PB7	USB-ASP port 7
9	RESET	RESET gomb
10	VCC	+5V tápellátás
11	GND	földelés
12	XTAL2	kristályoszcillátor
13	XTAL1	kristályoszcillátor
14	PD0	LED1
15	PD1	LED2
16	PD2	LED3
17	PD3	DHT22 (páraérzékelő) port 2 - kimenet
18	PD4	RELÉ1
19	PD5	RELÉ2
20	PD6	RELÉ3
21	PD7	RELÉ4
22	PC0	RTC-IC port J1 3
23	PC1	RTC-IC port J1 4
24	PC2	-
25	PC3	-
26	PC4	LCD port 11 (D4 - Kijelző adat)
27	PC5	LCD port 12 (D5 - Kijelző adat)
28	PC6	LCD port 13 (D6 - Kijelző adat)
29	PC7	LCD port 14 (D7 - Kijelző adat)
30	AVCC	aluláteresztő szűrő
31	GND	földelés
32	AREF	
33-36	PA7-PA4	-
37	PA3	LM35DT (Hőmérsékletérzékelő) port 3 - analóg kimenet
38	PA2	LCD kijelző E lába (Engedélyező érintkező)
39	PA1	LCD kijelző R/W lába (írás/olvasás)
40	PA0	LCD kijelző RS lába (adat/utasítás)

Az áramkör központjában először egy ATMEL ATmega16-os mikrokontroller chipet választottam, és dolgoztam az áramkör kialakításához. A meghajtáshoz szükséges 5V-os egyenfeszültséget egy 7805-ös feszültség stabilizátorral lehet kivitelezni, illetve egy 3V-ot előállító tápot is létrehoztam a kisebb feszültség előállítása érdekében. Ehhez az egyenfeszültség előállításhoz terveztem egy tápforrás választó jumpert is esetleg a későbbiekben hozzáadni az áramkörhöz.

I.3 Felhasznált érzékelők:

I.3.1 Hőmérséklet:

3.ábra LM35DZ hőmérsékletérzékelő szenzor

Jelen kapcsolásomban egy LM35DZ típusú hő szenzor is helyet kapott. Ez analóg kimenetelű, ezért az A/D bemenetre lett kötve.

Ezek az integrált áramköri hőmérsékleti eszközöknek kimeneti feszültsége lineárisan arányos a hőmérséklettel Celsius fokban mérve. Ezért ez sokkal előnyösebb a Kelvinben kalibrált lineáris hőmérséklet érzékelőkkel szemben, így nem igényel semmilyen külső kalibrálást vagy beállítást ahhoz, hogy szobahőmérsékleten +/- negyed fok lehetséges eltérés mellett, -55°C és 150°C közötti hőmérsékleti tartományban pedig +/- háromnegyed fok pontosságot biztosítson.

Ráadásul a termék előállítása is költséghatékony az ostyaszintű vágásnak köszönhetően (Wafer-Level Trimming).

Az LM35 hőmérsékletérzékelő chip meglehetősen pontos, ráadásul nem igényel külső alkatrészeket a működéshez. Pusztán csak a mikrovezérlőhöz kell kötni. Mindemellett lassan használódik és igen széles a hőmérsékleti tartománya.

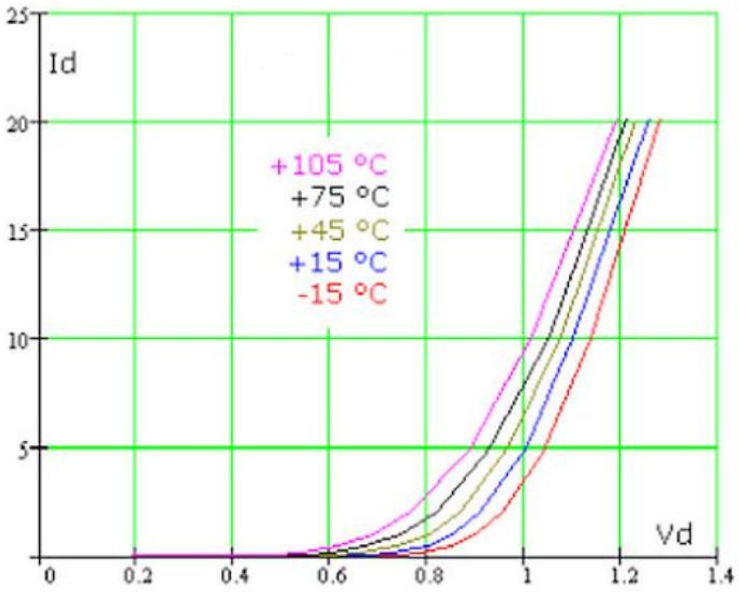
Az érzékelő 4V-tól egészen 30V-ig terjedő tápfeszültségről is használható, és kevesebb, mint 60 μA-t fogyaszt az aktív hőmérsékletmérések során. Ezért is alacsony az önmelegedése is, ami mindössze kevesebb, mint 0,08°C. Így tehát a hőmérsékletérzékelő saját hőmérséklete nem, vagy csak elhanyagolhatóan kis mértékben képes csak befolyásolni a mért külső hőmérsékletet.

Egyetlen hátránya, hogy negatív hőmérséklet mérésénél negatív előfeszítő feszültséget igényel, vagyis ha ebben a tartományban 0°C alatt szeretnénk mérni, akkor a talpponti feszültséget meg kell emelni.

Működési elve:

A hőmérséklet mérésének elve azon alapul, hogy a félvezető pn átmenetének feszültségjellemzője hőmérsékletfüggő, és a hőmérséklet emelkedésével az átmeneti feszültség csökken.

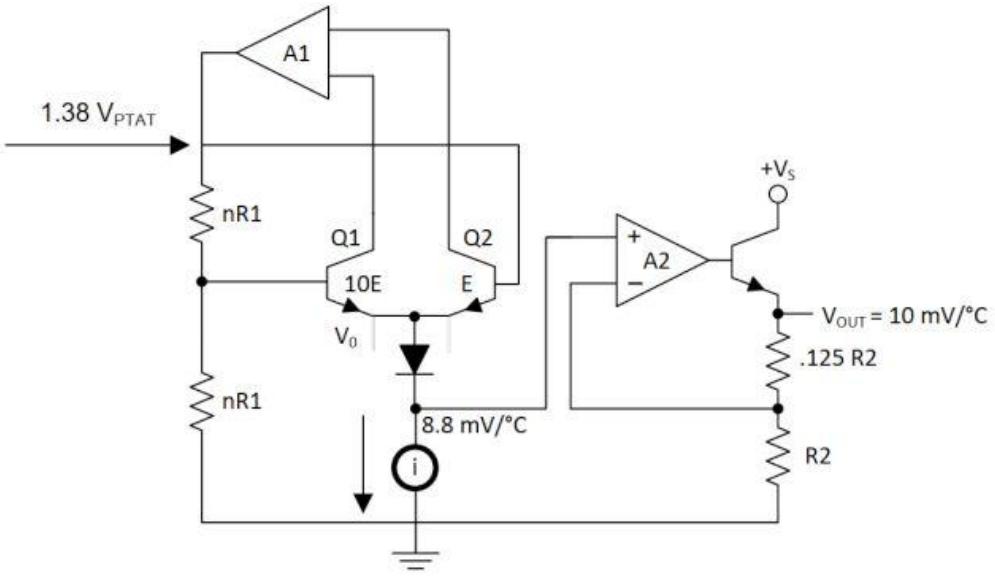
A feszültségváltozásnak pontos felerősítésével előállítható a hőmérséklettel egyenesen arányos analóg jel.



4.ábra nyitófeszültség-változás hőmérséklet függvényében

Félvezetők esetén a feszültségesés és a hőmérséklet között lineáris a kapcsolat. Ez az oka, hogy a diódákat, vagy bármilyen pn átmenetet hőmérsékletmérésre alkalmas eszközökként használják fel.

A pn átmenet feszülségváltozása szinteltolásra, belső kalibrálásra és linearizálásra is kerül. Ezeket az összetett számításokat (linearizálás, kalibráció) az LM35 belsejében az elektronika fogja majd elvégezni, így ad ki magából egy a hőmérséklettel lineárisan arányos feszültséget.



5.ábra Az LM35 hőmérsékletmérő chip belső felépítésének vázlata

I.3.2 Páratartalom:

Páratartalom mérésére én egy DHT22-es típusú szenzort szerettem volna alkalmazni a saját összeállításomnál. Bár a páratartalom számításhoz elengedhetetlen számára egy hőmérsékleti paraméter, így rendelkezik termisztorral, aminek hőmérséklet-mérési tartománya -40 és +125 fok között mozog + -0,5 fokos pontossággal, mégsem biztos megoldás a környezeti hőmérsékelt meghatározásához, ezért kapott helyet az LM35DZ típusú hőmérséklet érzékelő szenzor. Arra viszont alkalmas lehet, hogy egy gyors mérés keretein belül össze lehessen hasonlítani a két szenzor által visszaadott értéket.

Mivel az abszolút páratartalom a levegő vízgőz tartalma kg/m3 vagy mol/dm3 koncentráció-egységben. A képlettel adott definíciója: abszolút páratartalom

$$AH = \frac{n \cdot M_v}{V}$$

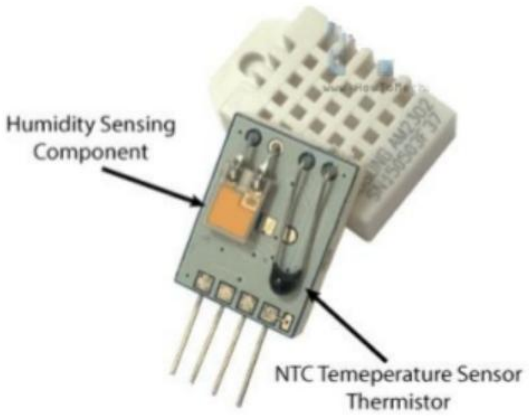
ahol n a vízmolekulák száma, Mv a víz molekuláris tömege, V pedig a térfogat. A levegő nem tud tetszőleges mennyiségben felvenni vizet – ha telítődik, megindul a kicsapódás. Azt a nyomást, amelyen ez a jelenség megtörténik, telített gőznyomásnak nevezzük.

A tapasztalatok szerint az abszolút páratartalom nincsen összhangban a szubjektív nyirkosság-érzetünkkel, ezért vezették be a relatív páratartalmat, amely a levegőben oldott vízgőz mennyisége a maximálisan oldható vízmennyiség százalékában kifejezve. A pontos definíciója a következő: relatív páratartalom

$$RH = 100 \cdot \frac{p_v}{p_s}$$

ahol pv a részleges vízgőznyomás, ps pedig az adott hőmérséklethez tartozó telítési nyomás. Ez utóbbi – és így a teljes mennyiség is – hőmérsékletfüggő. Ennek következményeként egy páratartalom-érzékelőnek tartalmaznia kell egy hőmérséklet-érzékelőt is a mért értékek értelmezéséhez.

A DHT22 érzékelő jobb szélesebb páratartalom tartománnyal rendelkezik a 11-es testvéréhez képest. 0 és 100% közötti, 2-5% -os pontossággal mér.



6.ábra DHT22 páraérzékelő szenzor

I.3.3 RTC modul:

A valós idő megfelelő méréséhez majd az időzítők beállításához egy DS3231+AT24C3 valós idejű memória modult választottam. Az RTC, vagyis Real Time Clock, ahogy a nevéből is látszik, egy valós idejű óra, amely képes kezelni órát, percet és másodpercet rendkívül nagy pontossággal. Ugyancsak képes kezelni a napokat, hónapokat, éveket. A pontosságért a DS3231 típusú chip felel, míg az adattárolásért AT24C32, amely 32kBit nagyságú.

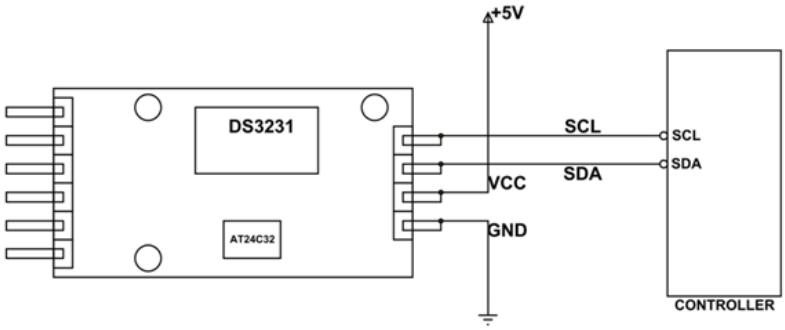
Az óra pontossága +/- 2ppm (0-50°C). Ez annyit jelent, hogy szélsőséges esetekben 1.2 másodperc lehet a késés havonta. *(Normál kvarckristályos órák esetén ez 30 másodperc is lehet.)*

A modul 3,3 vagy 5 V-os feszültséggel is képes működni. A hátulján lévő akkumulátor bemenete 3V-os (CR2032 típusú akkumulátor), amely elősegíti, hogy a modul több mint egy évig megtarthatja az információkat.

7.ábra DS3231RTC RTC modul

Kommunikáció létrehozása:

Az RTC modullal való kommunikáció csak abban az esetben jöhet létre, ha egy I2C interfészhez kapcsolódik. Az adatok ezen keresztül kerülnek elküldésre a modulnak, vagy onnan érkeznek.



8.ábra RTC modul kapcsolása az I2C interfésszel

A modul +5V-os szabályozott teljesítményen működhet legfeljebb, ennél magasabb feszültség már károsíthatja az eszközt. Az interfész pedig az ábrán látható módon jöhet létre: Az RTC modul SDA pontját a vezérlő azonos kimenetéhez kell csatlakoztatni, az SCL pontokkal pedig ugyanezt megismételni.

Mivel a vezérlő és a modul közötti kommunikáció nagyon összetett, általában bájtról bájtra küldik és fogadják az információkat, így előnyösebb a DS3231 modulhoz írt könyvtárak használata a programozáshoz, amelyek így nagyban megkönnyítik majd a kommunikációt.

Az idő mindig aktuális lesz, mivel ha megszűnik a tápforrás, az RTC modul chip automatikusan a hozzá csatlakoztatott akkumulátorforrásból veszi az áramot.

II. A projekt második szakasza:

A szakdolgozatom második felében, főképp a fejlesztői környezettel, és a programmal, illetve annak működési elvével foglalkoztam. Ennek megvalósítása miatt, a könnyebb felhasználhatóság érdekében és egyéb más okokból végül megváltoztattam az általam választott mikrokontrollert.

A cserét követően már nem volt szükség külön valós idejű memória modulra, mivel a választott mikrokontroller rendelkezik saját maga is beépítettel.

Ezen kívül a felhasznált páraérzékelő is cserére került.

II.1 STM32F407VE mikrokontroller:

Az új mikrokontrollerre a választásom egy STM32F407VE-re esett, mivel az eszköz programozásához az ST honlapjáról letölthető ingyenesen elérhető STM32CubeIDE fejlesztői környezetet is tudtam hozzá használni.

A CubeIDE egy Eclipse alapú fejlesztői környezet, amelyet az ST fejleszt. Eclipse alapú kódszerkesztőket az iparban is használnak beágyazott rendszerek fejlesztésénél. A chip emellé tartalmaz egy beépített RTC modult, így célszerűnek láttam, hogy erre cseréljem az előzőt, mivel így helytakarékosabb, és kezelhetőbb megoldás volt még az én számomra.

A CubeIDE fejlesztői környezet, pedig nagyban megkönnyíti a használó dolgát, hogy a portlábak programozása, azok beállításai és a hozzárendelések sokkal könnyebb módon lehessenek kivitelezhetőek.

Ez a fejlesztői panel sok lehetőséggel rendelkezik, amik közül csak néhány példa:

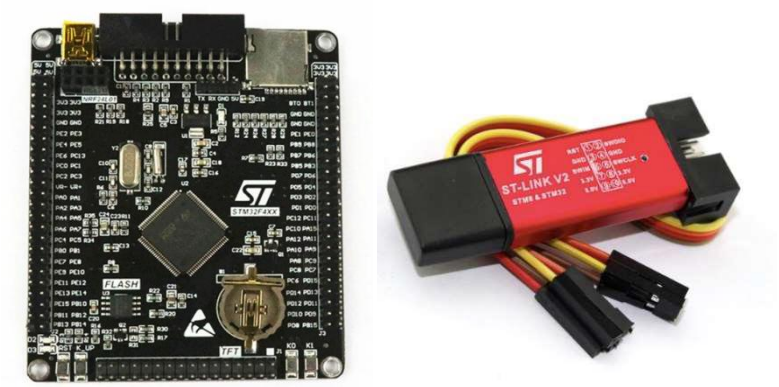
- MicroSD kártyaolvasó
- NRF2401 modul csatlakozás RF kommunikációhoz
- Beépített RTC modul
- SPI Flash chip

Néhány példa, amire a STM32CubeIDE fejlesztői környezet képes:

- Az órajel, perifériák, lábkiosztások konfigurációja

- Projektek létrehozása, és kódok legenerálása a beállított perifériák és lábkiosztások alapján, ami módosítások után szintén frissül folyamatosan
- Hibakeresési funkciók
- Windows, Linux és macOS operációs rendszerekkel is kompatibilis

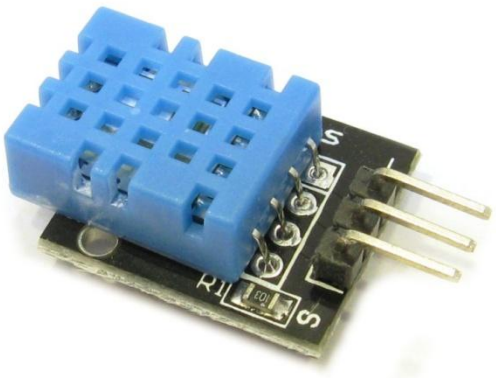
Ahhoz, hogy a mikrokontrollert össze tudjam kötni a számítógéppel, és a programozás ellenőrizhető lehessen, egy ST-LINK V2 debuggert is rá kellett kötnöm. Ez azonban csak addig szükséges, míg a program megírása, vagy módosítása megtörténik



9.ábra STM32F407VE mikrokontroller, és debugger (programozó modul)

II.2 Felhasznált páraérzékelő szenzor, és működési elve:

DHT11



10.ábra DHT11 páratartalom szenzor modul

A DHT22 szenzorom sérülése miatt alternatívaként a DHT11-es szenzorra cseréltem le.

A két szenzor közötti alapvető különbség, hogy a DHT11 csak egész értékeket ad vissza hőmérséklet és páratartalom tekintetében. A szenzor használatához szükséges kódot az adatlapban leírt működés alapján írtam meg.

Amikor a mikrokontroller egy start jelet küld, a DHT11 alacsony fogyasztású módból running módba vált, és vár a mikrokontrollerre, hogy befejezze a start jelet. Amint ez megtörtént, a DHT11 küld egy válaszjelet, ami 40bit adatból áll. Az adatok decimális és integrális részből állnak. A kommunikáció során a szenzor az MSB-ket küldi először. Az adatforma: 8 bit integrál adat relatív páratartalomról, 8bit decimális adat a relatív páratartalomról, 8bit integrál adat hőmérsékletről, 8 bit decimális adat a hőmérsékletről, majd 8 bit check sum. Ha az adatcsere sikeres volt, akkor a check sum értéke az előbb felsorolt részek összegének utolsó 8 bitjével kell, hogy megegyezzen.

Amikor a DHT11 detektálja a start jelet, egy alacsony feszültségű jelet fog válaszként küldeni, ami 80 us-ig tart. Ezután az adat láb feszültsége alacsonyról magasra vált, és megtartja ezt a szintet 80 us-ig, hogy felkészüljön az adatok küldésére.

Amikor az adat láb alacsony feszültségszintre vált, ez azt jelenti, hogy a DHT11 elkezdi küldeni a válaszjelet. Amikor a válaszjelet teljesen elküldte, az adatláb feszültsége ismét magas szintre kerül újabb 80 us-ig, hogy felkészüljön a kommunikációra.

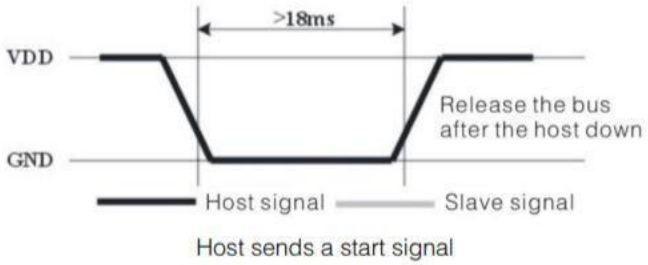
Amikor a szenzor adatot küld a mikrokontrollernek, minden egyes adatbit 50 us-os alacsony jelszintel kezdődik, és az ezt követi magas jelszintű pulzus hossza határozza meg, hogy az adott bit 0 vagy 1.

Ha a válaszjel mindig magas jelszinten van, az azt jelenti, hogy a szenzor nem megfelelően válaszol, ezért ellenőrizni kell a csatlakozását. Amikor az utolsó adat bit is el lett küldve, a DHT11 lehúzza a feszültségszintet, és megtartja ezen a szinten 50us-ig. Ezután az adat láb feszültsége pedig a felhúzó ellenállás miatt újra magas szintre kerül, amivel a szenzor azt jelzi, hogy elérhető.

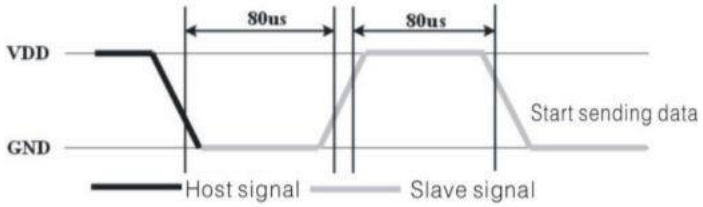
Inicializálás lépései:

1. Az adat lábat kimenetként beállítani
2. Az adatlábat alacsony jelszintre kell állítani 18 ms-ig

3. Az adatlábat el kell engedni, azáltal, hogy bemenetnek állítjuk



11. ábra Inicializálás jelalak



12.ábra Inicializálási jelalak

Válasz üzenet:

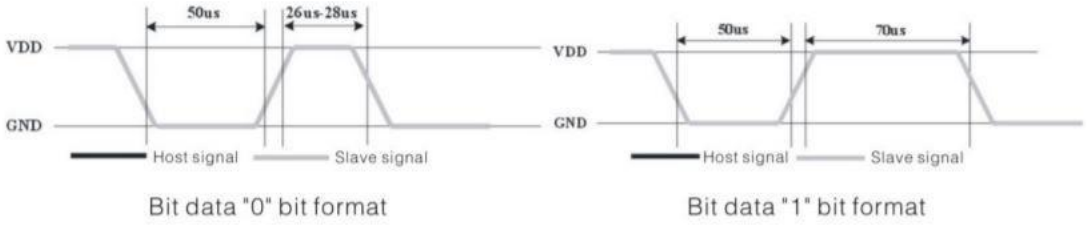
Ahhoz hogy ellenőrizhessük a szenzor jelenlétét, miután megkapta a start jelet, a DHT11 szenzor egy válasz jelet fog küldeni. Ez 80 us –ig alacsony szintre húzza az adat lábat, majd magas szintre újabb 80 us-ig. Ezt a választ az alábbi módon tudjuk detektálni:

1. Várunk 40 us-ot
2. Beolvassuk az adat láb állapotát, aminek alacsony szinten kell lennie ekkor
3. Újból várunk 80 us-ot
4. Ismét beolvassuk az említett lábat, és most már magas szinten kell lennie.

Ha az előbb felsorolt feltételek teljesülnek, ez azt jelenti, hogy a szenzor jelen van, elkezdhetjük az adatok beolvasását.
Az adatok beolvasása:

1. Várakozás, amíg az adat láb magas szintre kerül

2. Újabb árakozás 40 us-ig, mert a 0 bit hossza kb 26-28 us, és a láb magasba kapcsol 40us után, ami azt jelzi, hogy a bit 1
3. az így megkapott bitet letárolni



13. ábra Jelhosszak értelmezése bitként

Különbség a DHT11 és a DHT 22 szenzorok között:

DHT11:

- Rezisztív mérési elv
- Hőmérsékletmérés tartománya: 0-50 °C
- Hőmérsékletmérés felbontása: 1°C
- Mérés pontossága: +/- 2 %
- Páratartalom mérés határértékek: 20-90% RH
- Mérés felbontása: 1%RH
- Mérés pontossága: +/- 5%

DHT22:

- Kapacitív mérési elv
- Hőmérsékletmérés tartománya: -40 – +80 °C
- Mérés felbontása: 0.1°C
- Mérés pontossága: +/- 0.5 %
- Páratartalom mérés határértékek: 0-100% RH
- Mérés felbontása: 0.1%RH
- Mérés pontossága: +/- 5%

Ha esetleg szükség lenne a jövőben olyan környezetre, amely meghaladja majd a 90%-os páratartalmat, abban az esetben mindenképpen ajánlatosabb lesz lecserélni a DHT11-es szenzort. Erre azonban rugalmasan lehetőség van bármikor.

A hőmérséklet függvényében nem igazán relevánsak a legszélsőbb paraméterek, sem az emberi szervezet számára, sem pedig a növények igényei nem érik el, és végképp nem haladják meg a DHT11 szenzor legfelső határértékét.

Egy összehasonlító mérést sikerült végezni a DHT22 és DHT11-el.

Szenzor típus	Páratartalom	Hőmérséklet
DHT11	17%	26°C
DHT22	21.3%	26.2°C

Mivel a DHT11 a fent leírtak szerint csak teljes értéket ad vissza, így nem került vele tizedes szám kiíratva, mivel az folyamatosan nullás értéket adna vissza.

Bár a páraérzékelő szenzorok hőmérsékletérzékelése nem teljesen pontos, hisz előzetesen nem az a funkciója, attól még szobahőmérsékleten a visszakapott értékek függvényében még pontos adatot ad vissza, és a két szenzor értéke megegyezik.

A páratartalom %-os értéke viszont 4.3%-os különbséget mutat. A DHT22 magasabb értéket mért, amit viszont külső hatások nem befolyásolhattak.

II.3 Kijelző:

Kijelzőnek egy egyszerű kis 0,96 hüvelykes, 4 tűs OLED kijelző SPI/I2C modult választottam. A 4 tűs kialakításnak hála egyszerű az összekötése, mivel a több tűs kijelzők kihasználatlanok maradnak, illetve a 0,96 hüvelyk mérete elég kicsi és helytakarékos, miközben még tökéletesen ki lehet rajta láthatóan írattatni minden szükséges adatot.

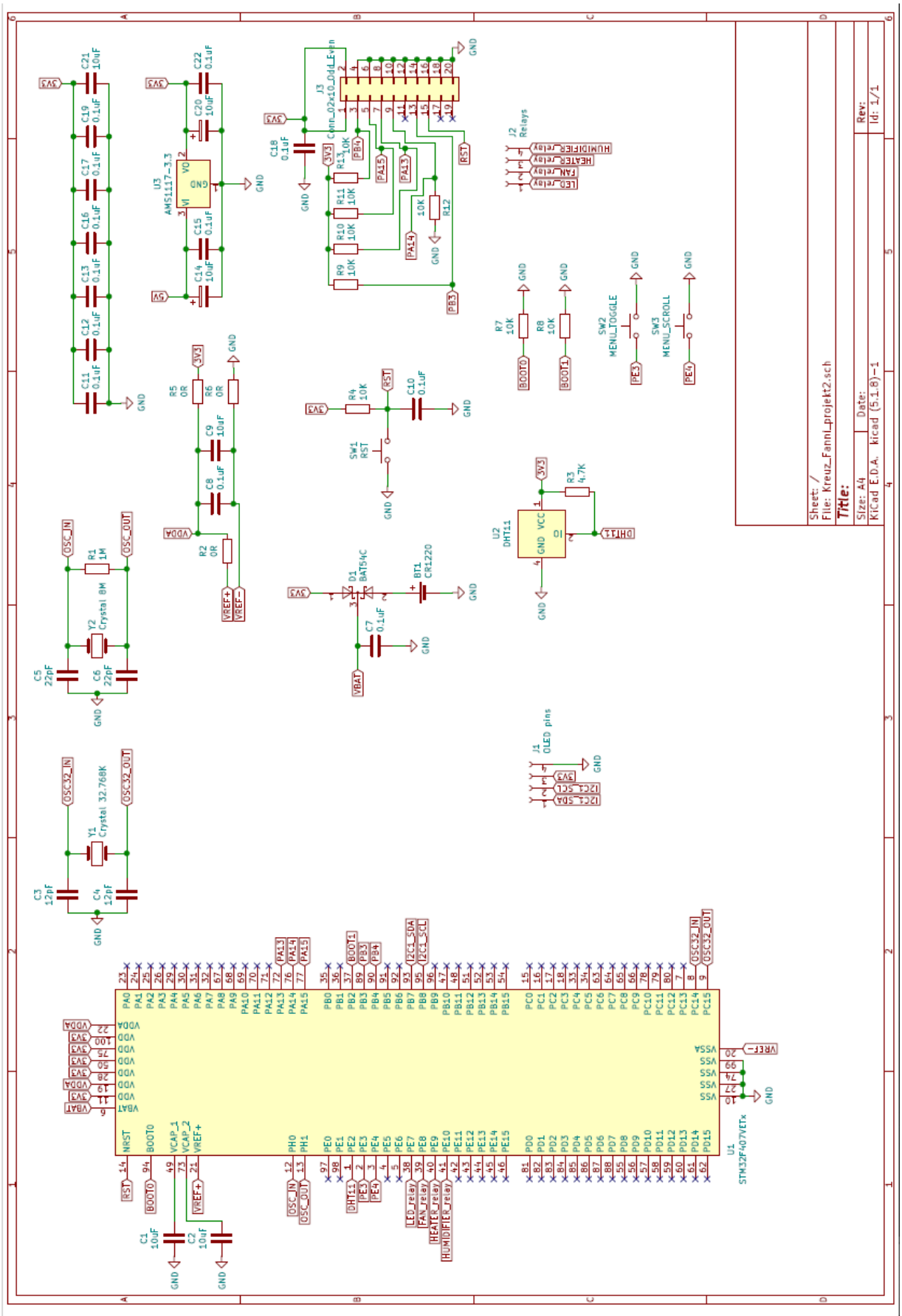
A 128x64 felbontású kijelzőmodul SPI/IIC protokollokat használva bármilyen mikrokontrollerrel kompatibilisen összekapcsolható.

A pontmátrix kijelzőmodul jellemzői:

- nagy, állítható kontrasztarány
- nagy, állítható fényerő
- széles látószög
- vékony körvonalak
- széles hőmérséklettartomány
- alacsony energiafogyasztás

A kijelzőt 3,3V és 5V között lehet meghajtani. Négy tűje közül a VCC kerül a tápforrásra, a GND (ground) a földre, míg az SCL és SDA pedig a serial clock és serial data pinek.

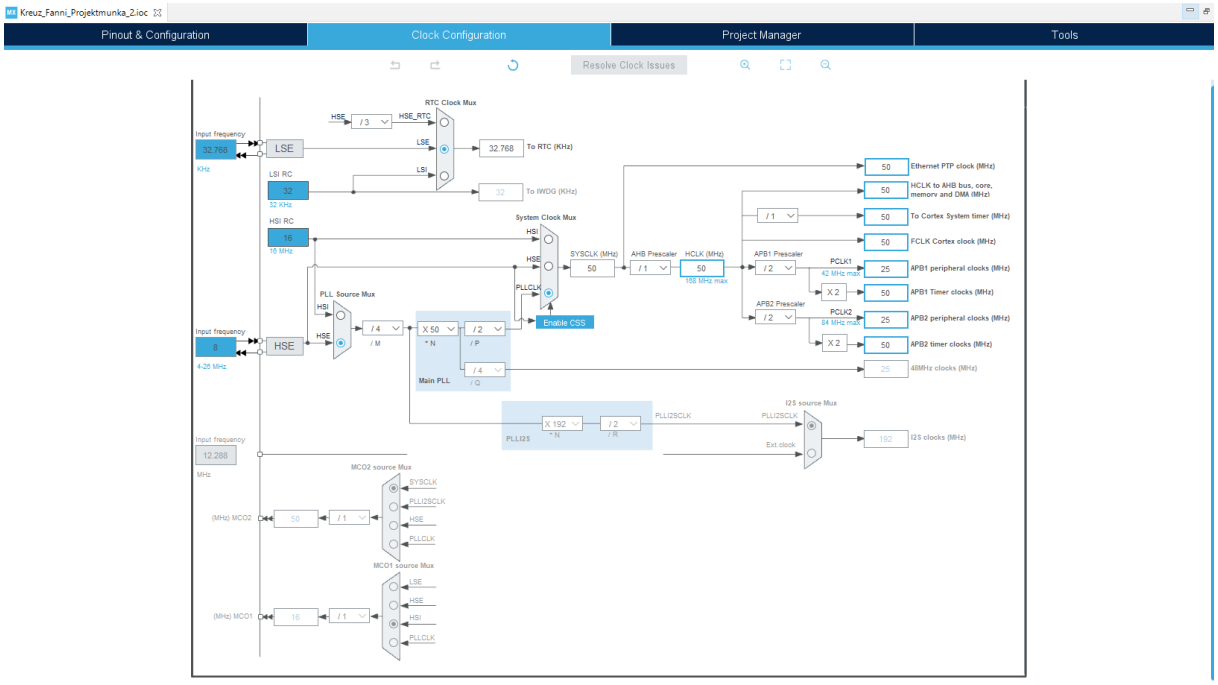
II.4 A projekt második felében létrehozott kapcsolási rajz:



14.ábra Projekt második szakaszának kapcsolási rajza

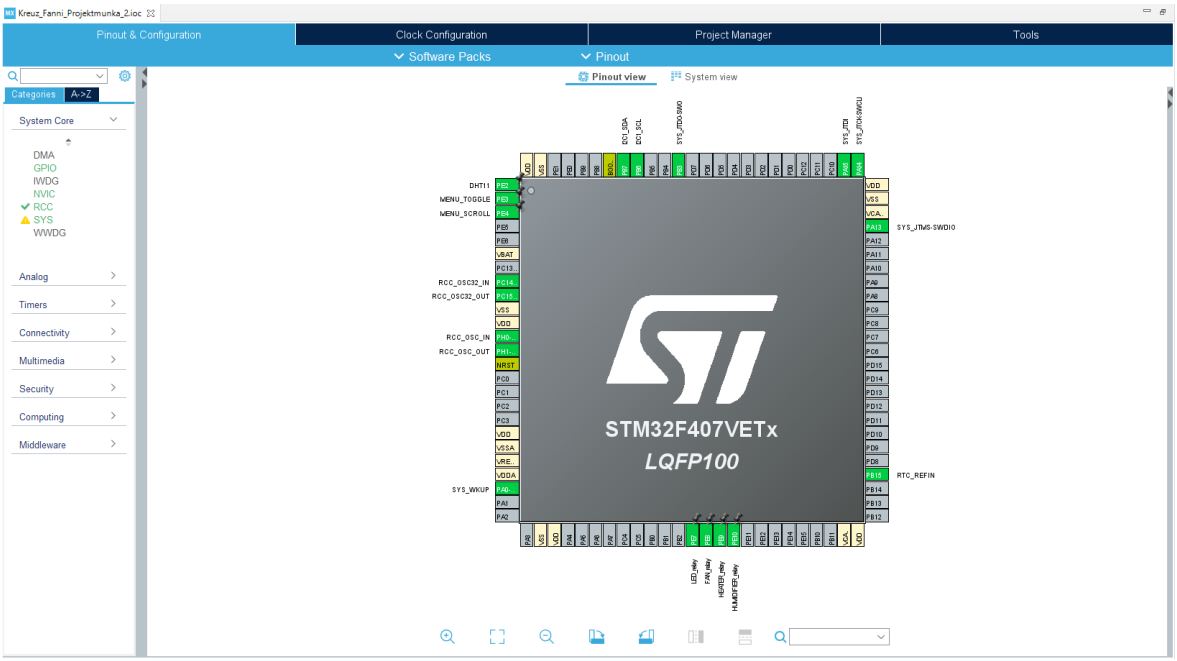
II.5 A fejlesztői környezet működési elve:

Első lépésként a programba belépve az új projekt készítésénél ki kell tallóznunk az általunk használt chip típusát. Ezután a projektet létrehozva egy <projekt neve>.ioc kiterjesztésű lap kerül elénk, ahol egy grafikus felületen konfigurálhatjuk az imént kiválasztott chips lábkiosztását, az órajeleket, és a chipben található belső modulokat. A konfigurálás mentésekor a program rákérdez, hogy akarunk-e kódot generálni a beállítások alapján. Természetesen ez egy nagyon hasznos funkció, amely elősegíti a tanulást, és visszafele gondolkodva segít megérteni bizonyos részek működését.



15.ábra Clock configuration

Teszteléshez egy olcsó, ebay-ről is rendelhető stm32f407vet6 panelt használtam, amelyen egy 8 Mhz-es frekvenciájú oszcillátor kapott helyet, így az órajel beállításoknál a HSE doboz előtti Input frequency-t erre az értékre állítottam. Ezután a fázis zárt hurok (PLL) segítségével 50MHz frekvenciára felszoroztam a jelet, amely a HCLK dobozba került. Ez fogja megadni azt az órajelet, amivel az utasítás feldolgozás történik. Az APB1 Timer-hez szintén 50 Mhz-et állítottam be, ennek majd később lesz jelentősége.



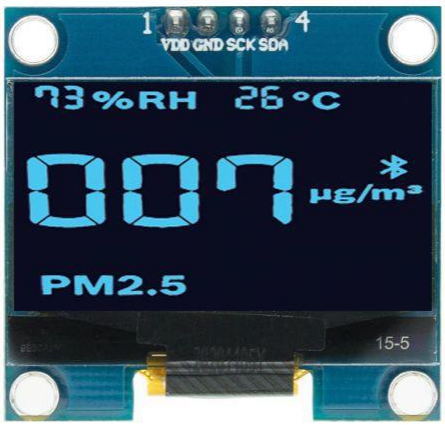
16. ábra Pin configuration

A lábakat a Pin configuration tabon állítottam be. Ezen kívül természetesen szükséges megadni az RCC modulon belül, hogy mind a HSE (high speed clock) illetve LSE (low speed clock) órajel források is használva vannak. Ezen kívül még elengedhetetlen a SYS modulon belül a Debug interfész típusának kiválasztása, (JTAG 4 pins) hiszen ezen keresztül csatlakozunk majd az eszközhöz.

II.6 Módosult lábkiosztás:

Lábkiosztás	Megnevezés	Bekötés
1	PE2	DHT11 signal
2	PE3	Menüpont funkció gomb
3	PE4	Menü görgető gomb
6	VBAT	RTC táppont
8	PC14-OSC32_IN	LSE input
9	PC15-OSC32_OUT	LSE output
10	VSS	Tápellátás
11	VDD	Tápellátás
12	PH0-OSC_IN	HSE input
13	PH1-OSC_OUT	HSE output
14	NRST	Reset
19	VDD	Tápellátás
20	VSSA	Tápellátás
21	VREF+	Tápellátás
22	VDDA	Tápellátás
23	PA0-WKUP	System wake-up
27	VSS	Tápellátás
28	VDD	Tápellátás
38	PE7	LED relé
39	PE8	Ventilátor relé
40	PE9	Fűtés relé
41	PE10	Párásító relé
49	VCAP_1	Tápellátás
50	VDD	Tápellátás
54	PB15	RTC ref. órajel bemenet
72	PA13	SWDIO (4pin debug, JTAG)
73	VCAP_2	Tápellátás
74	VSS	Tápellátás
75	VDD	Tápellátás
76	PA14	SWCLK (4pin debug, JTAG)
77	PA15	JTDI (4pin debug, JTAG)
89	PB3	SWO (4pin debug, JTAG)
92	PB6	Oled kijelző i2c SCL
93	PB7	Oled kijelző i2c SDA
94	BOOT0	Boot pin
99	VSS	Tápellátás
100	VDD	Tápellátás

A kijelző is lecseréltem egy kisebb méretű, kevesebb kábelt igénylő SSD1306 típusú 128*64 pixeels OLED kijelzőre. A kijelző i2c interfésszel csatlakozik a fejlesztői panelre, így csak 4 kábelre van szükség: VDD, GND, SCK, SDA



17.ábra SSD1306

A kijelző meghajtásához a Github-ról töltöttem le szabadon felhasználható könyvtárat. A könyvtár használata előtt a definiált SLAVE címet át kell írni a meglévő eszközön feltüntetett címre.

II.7 A program működési elve:

A main függvénybe belépve először a bekonfigurált perifériák inicializálása történik, mint például: GPIO, I2C, RTC, OLED kijelző.

Ezután egy végtelen while ciklusba lépbe kezdődik a folytonos, loop szerű működés. Első lépésként, a program megvizsgálja, hogy manuális vagy automata módban van-e a rendszer. Ennek függvényében lefuttatja az Automatic_Mode függvényt, amely az előre definiált limitek segítségével automatikusan vezérli a rendszerhez kapcsolt eszközöket (ventilátor, fűtés, világítás, párásító). A függvény előre definiált hiszterézis értékeket is figyelembe vesz, elkerülve ezzel a prellegés szerűen történő vezérlését a reléknek. A funkció kiértékeli a jelenleg rendelkezésre álló adatok alapján, hogy melyik relét kell ki illetve bekapcsolni, majd ezeket az értékeket, kiírja, a hozzájuk tarozó lábakra, megvalósítva ezzel a ki/bekapcsolást.

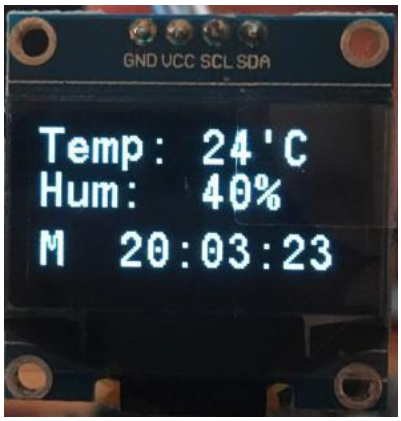
Természetesen az automata mód megfelelő működéséhez, szükséges a hőmérséklet és páratartalom, valamint az idő adatok folyamatos frissítése.

A működési mód felülvizsgálata után a végtelen while cikluson belül a második lépés a gombok kezelése, beolvasása. Ezt a Button_Handler funkció valósítja meg: a MENU_SCROLL gomb lenyomásával válthatunk az éppen kijelzett menüpontról egy másikra. A váltás egy irányú, ha pedig körbeért a menü, akkor újbóli gombnyomásra a kezdőképernyőre kerülünk.

A menü az alábbi részekből áll:

- MENU_RTC: egy általam alapképernyőnek nevezett megjelenés, ahol az aktuális idő, hőmérséklet, és relatív páratartalom kerül emgjelenésre, és folyamatosan frissül is. Ezen kívül, meg a képernyő bal sarkában az éppen aktív üzemmód is kijelzésre kerül (M -> manuális, A -> automata)
- MENU_LED: A világítás állapotát jelzi ki. A MENU_TOGGLE gomb megnyomásával megváltoztatható az állapota. Tehát épp bekapcsolható, vagy kikapcsolható. Automata üzemmódban a toggle gomb megnyomása értelmetlen, hiszen a logika amúgy is felül fogja írni, előbb célszerű manuális üzemmódba váltani.
- MENU_FAN: A ventilátor állapotát jelzi ki. A MENU_TOGGLE gomb megnyomásával megváltoztatható az állapota. Tehát épp bekapcsolható, vagy kikapcsolható. Automata üzemmódban a toggle gomb megnyomása értelmetlen, hiszen a logika amúgy is felül fogja írni, előbb célszerű manuális üzemmódba váltani.
- MENU_HEATER: A fűtés állapotát jelzi ki. A MENU_TOGGLE gomb megnyomásával megváltoztatható az állapota. Tehát épp bekapcsolható, vagy kikapcsolható. Automata üzemmódban a toggle gomb megnyomása értelmetlen, hiszen a logika amúgy is felül fogja írni, előbb célszerű manuális üzemmódba váltani.
- MENU_HUMIDIFIER: A párásító állapotát jelzi ki. A MENU_TOGGLE gomb megnyomásával megváltoztatható az állapota. Tehát épp bekapcsolható, vagy kikapcsolható. Automata üzemmódban a toggle gomb megnyomása értelmetlen, hiszen a logika amúgy is felül fogja írni, előbb célszerű manuális üzemmódba váltani.

- MENU_MODE: Az automata mód állapotát jelzi ki. A MENU_TOGGLE gomb megnyomásával megváltoztatható az állapota. Tehát épp bekapcsolható, vagy kikapcsolható. Automata üzemmódban a toggle gomb megnyomása értelmetlen, hiszen a logika amúgy is felül fogja írni, előbb célszerű manuális üzemmódba váltani.



18.ábra MENU_RTC képernyő

19.ábra A többi képernyőre alkalmazott séma

Még a Button_Handler függvényen belül természetesen a MENU_TOGGLE gomb érzékelése is megtörténik. A gomb lenyomása az éppen aktív kijelzésre van hatással, tehát ha éppen a 8. ábrán látható képnél nyomnánk meg, akkor a Kikapcsolva felirat Bekapcsolva feliratra vált, és be is kapcsolódna a FAN relé.

A gombok érzékelésénél a pattogás jelenségét megakadályoztam úgy, hogy a gomb felengedéséig nem fogad el új gomb lenyomást a program. Tehát ha nem engedjük el a gombot, mire a program futása egyszer körbeér, akkor sem történik probléma.

Szintén a függvényen belül maradva, ha egyik gomb sincs lenyomva, és a MENU_RTC kijelzés aktív, vagy csak simán automata módban vagyunk, akkor a friss hőmérséklet, páratartalom, és idő adatok lekérése szükséges, vagy abból a célból, hogy az automata üzemmód megfelelően működjön, vagy éppen azért, mert ezeket az adatokat látjuk a kijelzőn, ezért célszerű frissíteni őket. Tehát ha ezek a feltételek teljesülnek, akkor a DHT11_Step függvény kerül meghívásra, illetve az update_display értékével jelzem, hogy a kijelző tartalmát frissíteni kell.

A while ciklusban maradva, még a Button_Handler funkció után a Display_Handler funkció az utolsó lényegesebb lépés: itt történik meg a képernyő tartalmának frissítése, amennyiben azt az update_display változóban jeleztem. Az éppen aktív kijelzés alapján megtörténik a megfelelő adatok kiíratása, illetve az azokhoz tartozó állapotok kiírása.

A többször végrehajtandó utasítások egyszerűsítésére kisebb fügvényeket is ltérehoztam, mint például a Display_State.

```
807 void Display_State(uint8_t state)
808 {
809   ssd1306_SetCursor(0,18);
810   if (state)
811   {
812     ssd1306_WriteString("Bekapcsolva", Font_11x18, White);
813   }
814   else
815   {
816     ssd1306_WriteString("Kikapcsolva", Font_11x18, White);
817   }
818 
819 }
```

20.ábra Display_State

```
760 void Display_Handler(void)
761 {
762   if (update_display)
763   {
764     ssd1306_Fill(Black);
765     ssd1306_SetCursor(0,0);
766 
767     switch(menu_page)
768     {
769       case MENU_RTC:
770         Display_RTC_and_DHT11();
771         break;
772 
773       case MENU_LED:
774         ssd1306_WriteString("LED:", Font_11x18, White);
775         Display_State(led_state);
776         break;
777 
778       case MENU_FAN:
779         ssd1306_WriteString("FAN:", Font_11x18, White);
780         Display_State(fan_state);
781         break;
782 
783       case MENU_HEATER:
784         ssd1306_WriteString("Futes:", Font_11x18, White);
785         Display_State(heater_state);
786         break;
787 
788       case MENU_HUMIDIFIER:
789         ssd1306_WriteString("Parasito:", Font_11x18, White);
790         Display_State(humidifier_state);
791         break;
792 
793       case MENU_MODE:
794         ssd1306_WriteString("Auto mod:", Font_11x18, White);
795         Display_State(operating_mode);
796         break;
797 
798       default:
799         break;
800     }
801     update_display = FALSE;
802     ssd1306_UpdateScreen();
803   }
```

21.ábra Display_Handler

Az RTC működésénél fontos, hogy 3.3V-os CR1120-as elemet belehelyezzük a tartóba. Ezáltal a chip backup domain része áram alatt marad. A backup domain tartalék regisztereket, SRAM-ot tartalmaz. Ehhez az RTC-nek szüksége van, hogy a mikrokontroller kikapcsolt állapotában is el tudja tárolni az időt. Ehhez az adatlap alapján először engedélyezni kell a low power feszültség regulátort, meg kell várni a regulátor készenlétet, illetve át kell váltani az RTC órajelet a már korábban említett LSE forrásra. Ezután az órajel forrás változtatása csak reset után lehetséges. Ha ezekkel a lépésekkel megvagyunk, akkor a kontroller áramellátásának megszűnése esetén is képes lesz a low power domain tovább működni, hiszen órajelet, és tápforrást (elem) is

biztosítottunk a számára. A szükséges beállításokat a termék felhasználói kézikönyvében találtam meg. (5. fejezet: Power controller, 26. fejezet: Realtime clock)

Nagyon fontos lépés, hogy az inicializáló kódban a dátum és időbeállítást az első feltöltés után töröljük ki, vagy kommentként jelöljük, majd végezzünk el egy újbóli felprogramozást.

Ezekkel a lépésekkel azt érhetjük el, hogy első feltöltésre beíródik a regiszterekbe az kezdeti érték, de mivel minden inicializáláskor megtörténne, ez ezért el kell távolítanunk ezeket a kódrészleteket, hiszen így minden reset után ugyanattól az időponttól kezdve kezdenénk el.

II.7.1 Függvény leírások:

Display_RTC_and_DHT11:

Lekéri az RTC modultól az időt és a dátumot, majd ezeket az értékeket karakterekké alakítja, és eltárolja az így kinyert órát, percet, másodpercet. A kijelző első sorára kiírja a hőmérsékletet és a páratartalmat. Ezután beállítja a kurzort a kijelzőn a 3. sor elejére, és annak függvényében, hogy manuális vagy automata módban van, a rendszer kiírja a mód kezdőbetűjét nagybetűvel. Ezután pedig kiírja az előbb kinyert időt.

delay:

Milliszekundumos nagyságrendű késleltetést eredményez a 6-os időzítő segítségével. Bemeneti paramétere adja meg a késleltetés hosszát milliszekundumban.

DHT11_Step:

Meghívja a DHT11_Start_Measurement függvényt, jelezve ezzel a DHT11 szenzornak, hogy egy mérést szeretnénk elvégezni. Ezután ellenőrzi a szenzor válaszát a DHT11_Check_Response függvényhívás visszatérési értékével. Amennyiben a szenzor válaszolt, akkor bájtonként beolvassa a mérés adatait a megfelelő sorrendben: Először az relatív páratartalmat, majd a hőmérsékletet lehet lekérni bájtonként. Majd végül egy checksum-ot lehet kiolvasni, amit az eredmény ellenőrzésére használhatunk fel. Ezután a kapott eredményeket karakterekké alakítja, hogy azokat ki lehessen írni a kijelzőre. Ha a sznezor nem válaszolt, akkor pedig kötőjelek kerülnek az értékek helyére a kijelzőn, ezzel láthatóvá válik, hogy nem sikerült az adott mérés.

DHT11_Start_Measurement:

A hőmérséklet és páratartalom méréséhez szükséges korábban ismertetett kondíciókat végzi el.

DHT11_Check_Response:

A szenzor DHT11_Start_Measurement függvényre adott válaszát figyeli.

DHT11_Read_byte:

A szenzor méréséből származó adatokat olvassa be bájtonként.

Pheripheral_Handler:

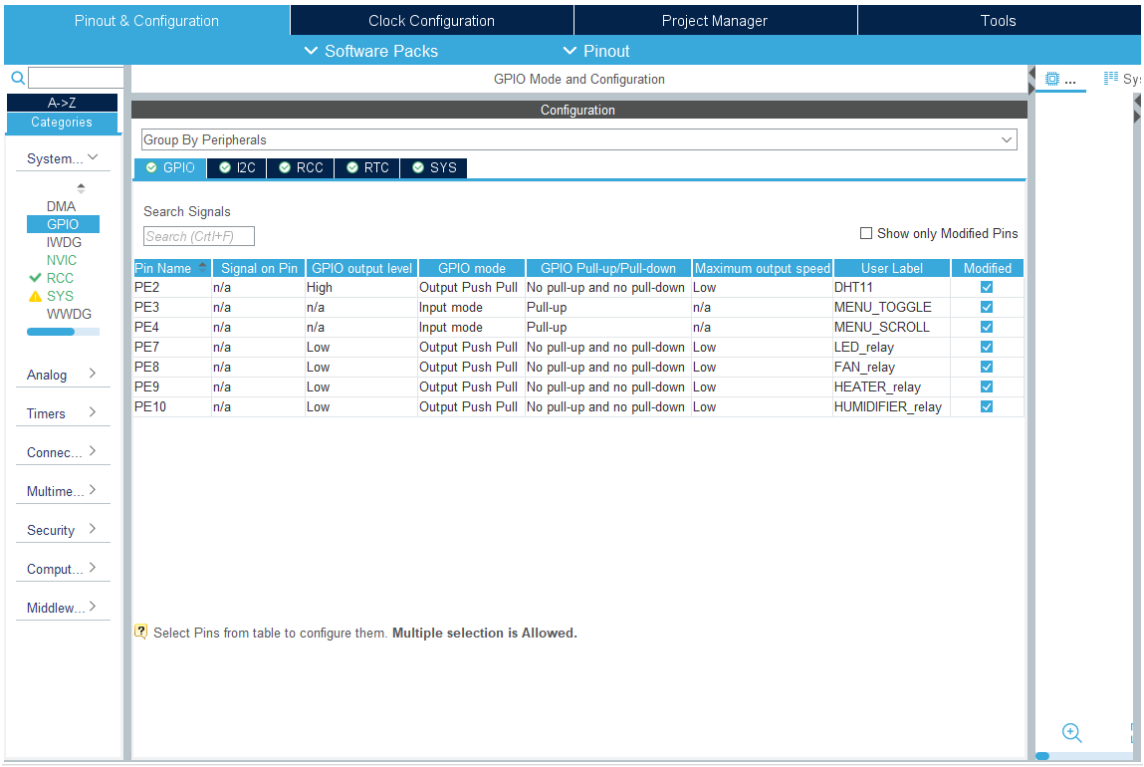
Az egyes menüpontokhoz tartozó állapotokat változtatja meg, amennyiben a MENU_TOGGLE gomb le volt nyomva. Ezek az állapotok lesznek felhasználva később a megfelelő feliratok kiírásához, illetve a relék kivezérléséhez.

Display_Handler:

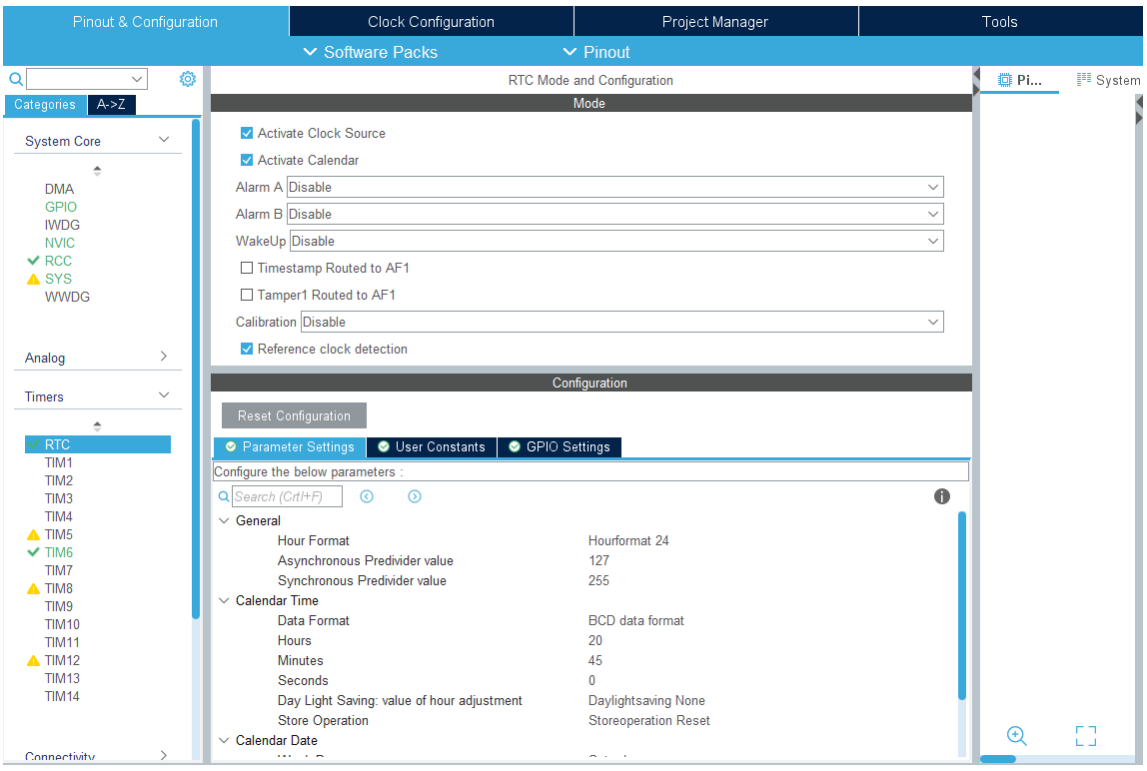
Amennyiben szükséges változtatni a kijelző tartalmán (update_display változó értékéből következtethetünk erre) akkor a kijelzőt kitölti fekete háttérrel, majd a kijelző bal felső sarkába helyezi a kurzort. Ezután pedig a menü állapota alapján kiírja az éppen kiválasztott menüpontot, és az ahhoz tartozó értéket.

Automatic_Mode:

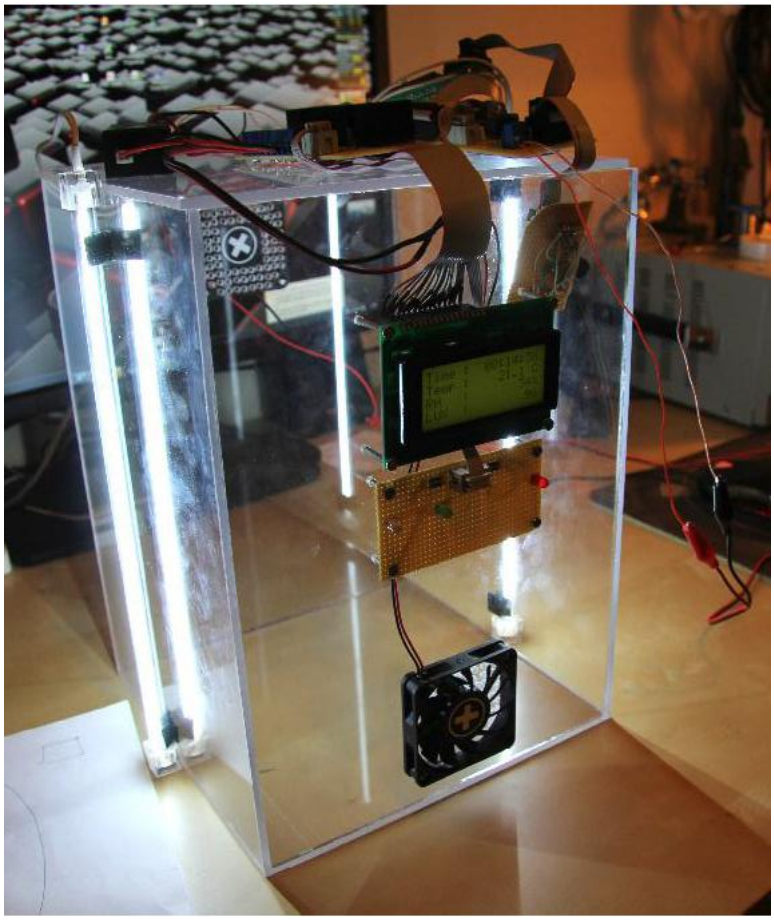
Ez a függvény ciklikusan kerül meghívásra, amennyiben automata módban van az eszköz. Az előre definiált érték és a pillanatnyi idő alapján eldönti, hogy szükséges-e be/kikapcsolni a világítást. Ezután a hőmérséklet érték alapján a ventilátor és a fűtés vezérlése egyszerre történik, hogy elkerüljük azt az esetet, amikor megy a ventilátor és megy a fűtés. Ezután pedig hasonló módon a párásító be/kikapcsolásának vizsgálata is megtörténik. Ezután pedig ezeket az állapotokat ki kell vezérelni a reléhez kapcsolt lábakra. Ez meg is történik a függvény végén.



22. ábra A fejlesztői környezetben felhasznált mikrokontroller lábak beállításai.



23. ábra Az RTC beállításai



24.ábra 40x20x20-as kamra

Összefoglaló

A szakdolgozat alatt megismerkedhettem, és először tervezhettem saját áramkört és hozzá kapcsolási rajzokat, illetve mélyebben beleláthattam az egyes érzékelők működési elvébe, felépítésébe, illetve a mikrokontrollerrel való kommunikációjukba.

Ezen kívül a már ismert ATmega mikrokontroller chip mellett jobban belemélyedhettem STM32F407VE mikrokontrollert, a felépítését, hozzá tartozó CubeIDE nevű fejlesztői környezetet, annak lehetőségeit, a portlábak beállításának, kiosztásának módjait és lehetőségeit, az ez által legenerált kód tartalmát, illetve az ilyen irányú tanulmányaim után ismét belemélyedhettem a mikrokontoller programozásba, annak felépítésébe.

Szakdolgozatomban bemutathattam:

- Pár szót az orchideákról, és a szimulált környezet előnyeiről
- A növénynevelő kamrát és környezetét
- DHT22 és DHT11 hőmérséklet, és páratartalom érzékelő szenzorok
    - Felépítését
    - Működési elvüket
    - Kommunikációs módjukat
- LM35 hőmérsékletérzékelő szenzor, és működési elve
- RTC modul bemutatása, kommunikációja
- STM32F407VE mikrokontrollert, illetve annak lehetőségeit
- STM32CubeIDE fejlesztői környezetet, és annak:
    - néhány lehetőségeit
    - a fejlesztői környezet működési elvét
    - a program működési elvét
- az OLED kijelzőmodult, és annak jellemzőit

Számomra a növénynevelő kamra megépítésében sok lehetőség rejlett. Nem csak magában a kamrában, hanem az ellenőrzött, és irányított környezetben, hisz ezt kamrák

esetében, terráriumoknál, akváriumoknál is alkalmazni lehet, akár növények, akár hüllőkkel benépesített terráriumok esetében is, illetve a mindennapi környezet megfigyelésére is alkalmasak ezek a szenzorok. Nem csak a növények, vagy az emberek, de velünk élő kis kedvenceink számára is lényeges a hőmérséklet és páratartalom. Az előbbiekben említett hidegvérű hüllők, és kétéltűek számára szintén nagyon fontos az ellenőrzött hőmérséklet, a pikkelyes bőrüknek pedig a levegő páratartalma, de csak egy másik példát említve, a papagájok számára sem egészséges a túl száraz levegő.

Maga a mikrovezérlő pedig egy számomra nagyon érdekes, és könnyen kezelhető választás volt. A grafikus felület segítségével könnyen beállíthatóak lettek így a mikrovezérlő lábai, könnyen átlátható felületet biztosított, a letisztult, és egyszerű környezetet pedig hamar képes átlátni és megérteni még olyan is, aki először találkozik csak vele.

Ezúton szeretném megköszönni a lehetőséget, hogy foglalkozhattam ezzel a témával. A jövőben mindenképp szeretném folytatni, akkor már teljesen egyéni felhasználás céljából ezeket a megoldásokat egy tervezett otthoni akvárium berendezését követően, amelynek ötletét ezen szakdolgozatomban folytatott munkám inspirálta.

Forrásjegyzék:

1. https://www.st.com/en/microcontrollers-microprocessors/stm32f407ve.html

2. https://www.alldatasheet.com/view.jsp?Searchword=STM32F407VET6

3. https://learn.adafruit.com/dht

4. https://www.st.com/resource/en/user_manual/dm00629856-stm32cubeide-user-guide-stmicroelectronics.pdf

5. Programozás 1-2 laboratóriumi anyagok

6. Programming and STM32CubeIDE basic tutorial videos

7. E-mailben rendelkezésemre bocsájtott, a kamrához tartozó szakdolgozat

8. https://starduino.hu/2020/08/15/dht11-dht22-paratartalom-homerseklet-arduino/

9. https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

10. https://components101.com/modules/ds3231-rtc-module-pinout-circuit-datasheet

11. https://www.elprocus.com/atmega16-next-generation-micro-controller/

12. http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf

13. http://mti.kvk.uni-obuda.hu/adat/tananyag/sensor/erzekelok_01.pdf

14. http://mti.kvk.uni-obuda.hu/adat/tananyag/elektronika4/E4_8_ADDA2_v3.pdf