# TÓTH MÁTÉ

**NYÍREGYHÁZI EGYETEM**

JRAY0I

IoT [Programtervező Informatikus]

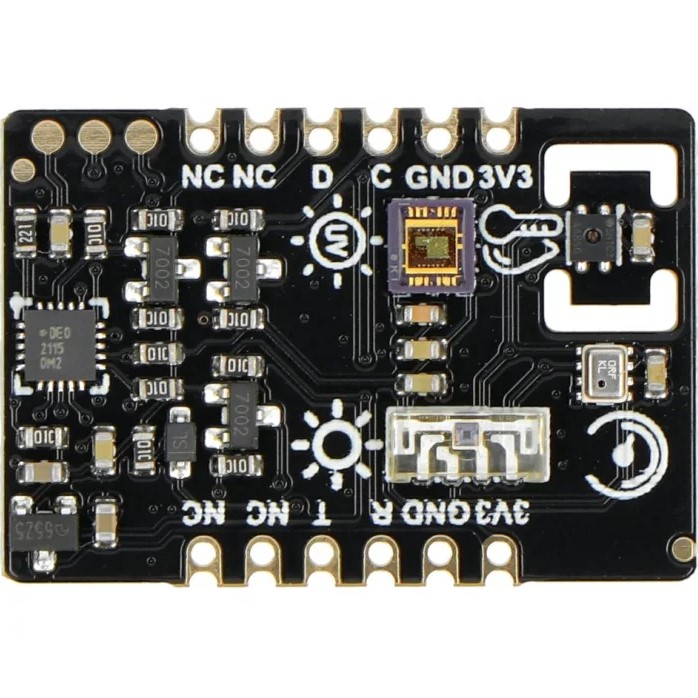
# A feladat ismertetése

Feladatomul egy Arduino-ban végzett programozást választottam, mely egy ESP32 S3 típusú mikrokontrollert működtet amihez egy, a DFRobot által gyártott SEN0500 típusú szenzort kötöttem. Ezt a programot valamint a hozzá kötött szenzorokat belső terek illetve külső környezet monitorozására alkottam.

Érdekesnek találtam ezt a programot, hogy sok mindent meg lehet benne csinálni akár kevés tapasztalattal is, és mivel szabad fejlesztésű szoftverről van szó ezért rengeteg segítséget lehet találni széles körben az interneten.

A program írásához a fejlesztő által elkészített alapszoftver is igen hasznosnak bizonyul majd a későbbiekben.

 A feladat ismertetése  
1. ábra



2. ábra

* 1. **Néhány alapfogalom**

Az érzékelő szerv fogalma: Az érzékelő szerv az információ szerzés legfontosabb eszköze, amely a folyamat állapotáról egyértelműen meghatározható jelet szolgáltat. (hőmérséklet, nyomás, páratartalom, fény intenzitás, UV intenzitás) Egy technológiai folyamat állapotjellemzői a legtöbb esetben nem villamos mennyiségek. A korszerű vezérlők viszont villamos működésűek, ezért ezeknek az eszközöknek a feladata nem csak az érzékelés, hanem az érzékelt jelek átalakítása villamos jellé is. Éppen ezért sok esetben érzékelő és átalakító szerv a neve.

# A hardver ismertetése

* 1. **A hardver részei**

A hardver 4 fő részből áll:

1. Egy ESP32 S3 TFT típusú mikrokontroller
2. Egy DFRobot által gyártott SEN0500 típusú multiszenzor (T,p,ρ,
3. Egy 18650 Li-Ion akkumulátor (opcionális)
4. Egy kapcsoló
   1. **Általános leírása**

A mikrokontroller elsősorban futtatja a saját tárhelyén tárolt programot, amit az Arduino kezelőfelületén tudunk megírni, majd a beolvasást követően a mikrokontroller processzora elkezd kommunikálni a másik panelon lévő szenzorokkal az I²C protokoll alapján és beolvassa a szenzorok által megadott értékeket. Ezt követően kiírja a kapott adatokat a mikrovezérlőn megtalálható 240x135 felbontású, színes kijelzőre.

# A program és ismertetése

Az Arduino programfelület sajátossága, hogy 2 alap funkcióval kezdődik minden. Az első a „Setup” ami a mikrovezérlő elindulásával egyetlen egyszer fut le. Ez a funkció van minden esetben elől. A második funkció a „Loop” pedig akkor kezdődik el, ha a setup sorokba írt programrész már befejeződött, ezt követően a loop-ba írt utasítások egészen addig futnak folyamatosan, ameddig meg nem állítjuk manuálisan azt.

A program megírását a feltételezhetően szükséges könyvtárak beillesztésével kezdtem meg. Ezek az úgynevezett könyvtárak segítik a fejlesztőket, ugyanis nem kell például egy kijelző meghajtásához minden karaktert program nyelven megrajzoltatni hanem a már meglévő könyvtárakból elég egyszerűen csak kihozni belőle.

A könyvtára importálása után az elsődleges lépésünk lesz az, hogy meghatározzuk a program felhasználási hardverét. Ez biztosítja azt, hogy nem csak ezen a specifikus platformon használhatjuk a szoftvert, hanem Pl. Arduino vagy ESP8266 mikrovezérlőkön is.

Továbbá meghatározzuk a kommunikációs hidat. Kettő közül választhatunk. Jelen esetben én az I2C protokollt használom az egyszerűsége miatt.

A program következő részében megadjuk a lehetőséget az akkumulátor figyelő chip típusának, két fajta fordulhat elő alapvetően beépítve ezekbe a mikrovezérlőkbe, ezzel biztosítjuk a lefedettséget több panel között. Utána megadjuk a beépített kijelző vezérlőjének az adatokat a könyvtár segítségével. A következő sorokban elkezdjük az automatikus fényerő állítás alapjait.

A program egyik fő része most kezdődik. A Setup-ot követően Inicializáljuk a beépített kijelzőt, megadjuk a felbontását pixelekben kifejezve, beállítjuk a szöveg betűtípusát és ezzel együtt a betűméretet és a kijelzőt megtöltjük teljes mértékben feketével, majd a kijelző vezérlő memóriájából előhívjuk a benne lévő információt. Erre azért van szükség, mert ha ebben a sorrendben hívjuk a sorokat, akkor nem lesz véletlenszerű zaj látható a kijelzőn. A következő sorban megadjuk a szöveg kezdetét pixelekben kifejezve és átállítjuk a kiírandó szöveg színét fehérre.

A következő sorokat rögtön egy feltétellel kezdjük amelyben automatikusan felismeri a mikrovezérlőn megtalálható akkumulátor vezérlő ic-t és kiírja annak pontos típusát és azt, hogy melyik I2C címen kommunikál a processzorral. Ennek a végén előhívjuk a kijelző memóriájába eddig bevitt adatokat majd csak utána adunk háttérvilágítást, így elkerüljük az üres kijelző közben a fölösleges fogyasztást, és megspórolhatunk ezzel egy kevés energiát az akkumulátorban. Ezután várunk másfél másodpercet, hogy a kiírt adatokat el tudjuk olvasni, ha ez letelt akkor újra üressé tesszük a kijelzőt és visszaállunk a kezdeti állásba.

Ha eljutott a program eddig, akkor megkezdi a SEN0500 szenzor inicializálását és szöveges formában tájékoztat a sikertelenségéről vagy a sikerességéről, ha sikeres, akkor kiírja az I2C buszon lévő összes megtalált eszköz azonosítóját HEX formában (Pl.: 0x22). A végén megint előhívjuk a memóriába került képet, várunk másfél másodpercet és a háttérvilágítást kikapcsoljuk.

Most kezdődik a program fő része, ez fog futni folyamatosan, újra és újra egészen addig, amíg a folyamatot le nem állítjuk. Az automatikus fényerő további beállításait megadjuk, ezek között: a minimum, maximum értéket, a PWM (Pulse Width Modulation) értékét, ami jelen esetben 8 bites. Egy kis matematika, amivel átalakítjuk a jelen esetben túl magas, 0 és 2500 közötti értéket 0 és 255 közé. Továbbá két egyszerű logika követi ezt, ami megadja, hogy 15% alá nem mehet illetve a másik lekorlátozza 100%-ra.

Itt megkezdjük a szenzorból beolvasott adatok kiírását a kijelzőre. A hőmérséklethez hozzáadtam egy extra funkciót, amivel behatároltam bizonyos értékeket, amik között más-más színnel írja ki a hőmérsékletet, ez jelzi majd a szélsőségesebb időjárást. Az értékek behívását a könyvtárból úgynevezett hívóparancsokkal valósítjuk meg. A vessző után írt számok pedig a kiírt értékek utáni pontosságot adják meg, lehet egészre kerekített éréktől kezdve egészen ezredes pontosság tartó tartomány is. AZ UV intenzitás utáni hosszú szám az egy általam kimért korrekciós szám.

A kód utolsó részében pedig meghatározzuk, hogy melyik akkumulátor figyelő ic lesz használatban és az alapján írja majd az akkumulátor adatait a kijelzőre. Itt is hozzáadtam egy figyelmeztetést szín alapján, ha bizonyos százalék alá esik a töltöttség, akkor először sárgával, majd ha tovább esik, akkor pirossal jelzi nekünk ezt. A kód legvégén pedig az eddigi lépésekben memóriába juttatott információt kiküldjük a kijelzőre és a Loop funkció kezdődik előröl.