Önálló Laboratórium Dokumentáció

Okosotthon távoli felügyelete

Név: Kiss Bence

Neptun: G9XE68

Email: *kisb20000909@gmail.com*

Projekt link: *https://github.com/kisb9999/Onlab_2023*

Feladat ismertetése

Az elmúlt években az IoT (Internet of Things) világa hatalmas fejlődésen esett keresztül, melynek során a mindennapi élet egyik legnélkülözhetetlenebb technológiájává nőtte ki magát. Az élet szinte minden területén jelen van az egészségügytől kezdve a gyártáson át egészen a háztartásokig. Az IoT alapja az adatgyűjtés különböző hardveres eszközök (okos szenzorok) felhasználásával és ezen adatok biztonságos központosított felhő alapú eltárolása, valamint felhasználása. Ezekből az adatokból olyan értékes információkhoz juthatunk, amelyek segítségével betekintést nyerhetünk a közösségi terek, lakások okos eszközeinek és az ipari létesítmények működésébe (Ipar 4.0).

Az IoT technológiák széleskörű felhasználhatósága miatt úgy gondoltam, hogy érdekes lenne ilyen irányba elindulnom az önálló laboratóriumi feladatom során. Így esett a választásom az okosotthon működésének vizsgálatára.

Az okosotthonokon belül is számos különböző adatot lehet gyűjteni, mint például a hűtők belső hőmérséklete vagy a házon belüli hőmérséklet és fényviszonyok. Úgy döntöttem, hogy a labor keretein belül az utóbbival foglalkoznék mélyebben. A szobák belső hőmérsékletét lehet befolyásolni modern légkondicionáló berendezésekkel, amelyek képesek mind hűtési, mind fűtési funkciók ellátására, valamint a redőnyök lehúzásával vagy felhúzásával pedig a hőérzet alakítható. Az árnyékolók alkalmazásával csökkentjük a napfény által sugárzott infravörös tartományban érkező hullámokat, ezzel csökkentjük a fény hatására fellépő hőmérséklet emelkedést.

Ennek a két technológiának az összefüggését vizsgálom ebben a feladatban és egy valódi okosotthon működését szimulálom.

A hőmérsékletet és a fényerőt két különböző szenzorral mérem (**temperature sensor**, ambient light sensor (**ALS**)), ezeknek az adatait jelenítem meg az idő függvényében egy Androidos mobilapplikációban felhasználóbarát formában. Úgy ítéltem meg, hogy itt a késő tavaszi és kora nyári időjárásban a benti hőmérséklet 18 és 24 fok között optimális, ezen értékek között nincsen teendő, azonban, ha a 18 fok alá csökken a hőmérséklet, akkor valamilyen módon fel kell melegítenünk a szobát egy ideális hőfokra. Ugyanez a teendő akkor, ha 24 fok fölé emelkedik a hőmérséklet csak értelemszerűen ilyenkor hűtenünk kell.

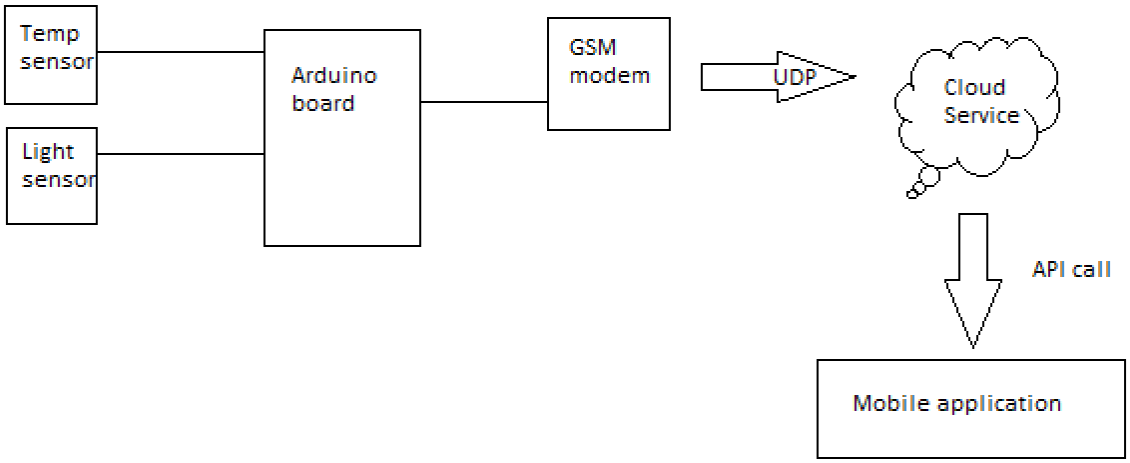
Tervezés

A tervezés első lépése az volt, hogy eldöntsem, hogy tulajdonképpen hogyan is akarom kivitelezni az adatok beolvasását, felhőbe juttatását, tárolását és későbbi felhasználását. Tudtam, hogy kelleni fognak nekem szenzorok a hőmérséklet és a fény mérésére, azonban azt nem tudtam, hogy hogyan is fogom kinyerni belőlük az adatokat. Azt se tudtam, hogy milyen kommunikációs technológiát veszek igénybe az adatok felhőbe juttatására.

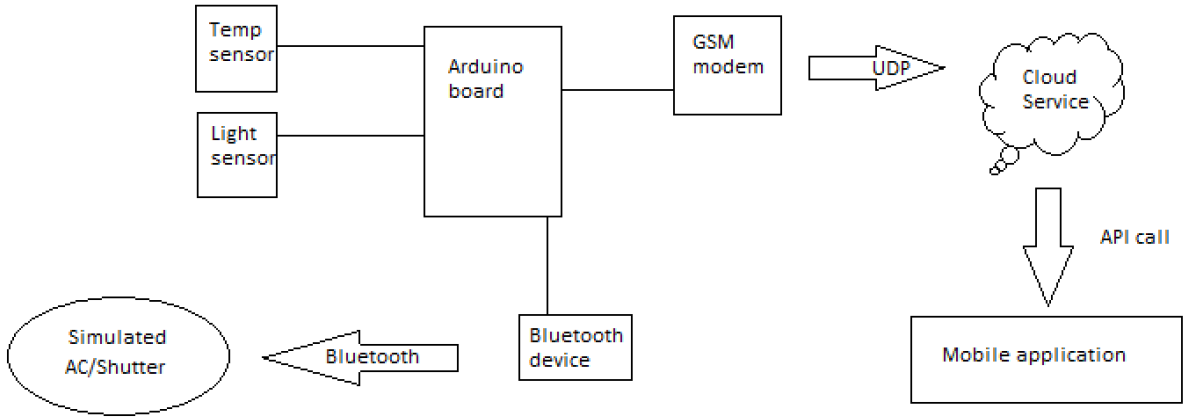
Kis kutatás után realizáltam, hogy szükségem lesz egy mikrokontroller (microcontroller unit (**MCU**)) által vezérelt board-ra, amihez egy hőmérséklet szenzor és egy látható fényérzékelő is van csatlakoztatva. A mikrokontrollerre tudtam, hogy kelleni fog egy könyvtárat írnom, ami kiolvassa a szenzorok adatait, valamint tárolásra alkalmas formátumba szervezi őket.

Ezt követően találtam ki, hogy egy felhő szolgáltatásban el lehetne tárolni az adatokat, mert az egyszerűen hozzáférhető lenne és csak egy API hívást kéne kezdeményeznem ahhoz, hogy lekérjem ezeket az adatokat a felhőből.

Az adatok lekérését és megjelenítését egy mobilos applikáción, lehetőség szerint Androidon keresztül terveztem megvalósítani. Azért döntöttem így, mert Androidos applikációkkal foglalkoztam már korábban, továbbá használtam már egy grafikonos kiegészítőt is. Úgy gondoltam, hogy az lenne a leginkább felhasználóbarát megközelítés, ha egy grafikonon jeleníteném meg az adatokat, mert azt szinte minden felhasználó megérti.



A tervezés következő lépése abból állt, hogy kitaláljam mit tegyen a felhasználó akkor, amikor a hőmérséklet az optimális zónán kívülre esik. Ezek a tevékenységek a légkondi hűtésének és fűtésének elindítása és kikapcsolása, valamint a redőnyök fel- és lehúzása. Az ötlet az volt, hogy valamilyen eszközzel szimulálom az AC és a roló a működését, ezzel lépne kapcsolatba a board Wifi-s vagy Bluetooth-os vezetékmentes kapcsolaton keresztül. Ezen a ponton úgy döntöttem, hogy csak a Bluetooth-os irányt valósítom meg. Ehhez azonban szükségem volt egy olyan modulra, ami a mikrokontrollert és az emulált okos eszközöket rádiós módon összekapcsolja. Erről látható egy elvi blokkdiagram az alábbi képen.

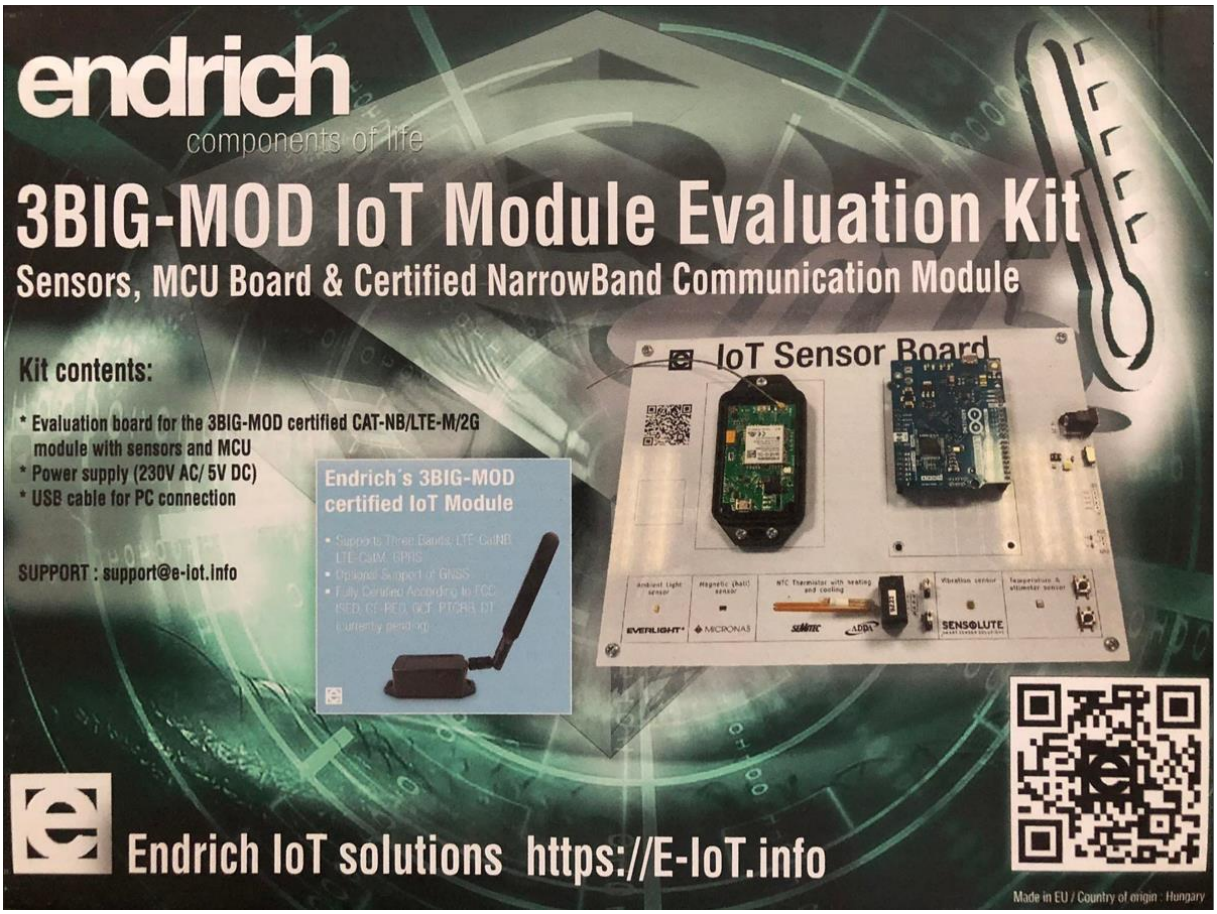


Amint az a végleges tervből is látszik, a feladat elvégzéséhez szükségem volt egy okos board-ra, amihez kapcsolódik egy hőmérséklet szenzor, egy fényérzékelő, valamint egy Bluetooth-os kommunikációra alkalmas eszköz. Továbbá szükségem volt egy GSM modemre is, ahhoz, hogy az okos szenzorokat az Internethez tudjam kapcsolni.

Összefoglalva, a terveknek megvalósításához szükségem volt egy IoT végpontra (szenzor+kommunikáció+vezérlés), egy ingyenes felhő alapú adatbázis szolgáltatásra egy jól működő API-val, aminek segítségével lekérhettem az adatokat. El kellett készítenem a szenzorok adatgyűjtéséhez és a GSM modem működtetéséhez szükséges könyvtárat, létre kellett hoznom egy mobil applikációt ezeknek az adatoknak a megjelenítésére, valamint egy légkondicionálót és egy redőnyt szimuláló eszközt is.

Technológiai háttér

A technológiai háttér megteremtésekor egy olyan lehetőségem adódott, hogy az alább látható Endrich gyártású IoT board birtokába jutottam és szabadon felhasználhattam. A board lelke egy Arduino Leonardo típusú mikrokontroller. Ez az eszköz a magja az egész koncepciónak, ugyanis ez felel az adatok kinyeréséért a szenzorokból, rajta keresztül lehet az adatokat formálni és tovább küldeni a GSM modemnek, ami a felhőbe továbbítja azokat.

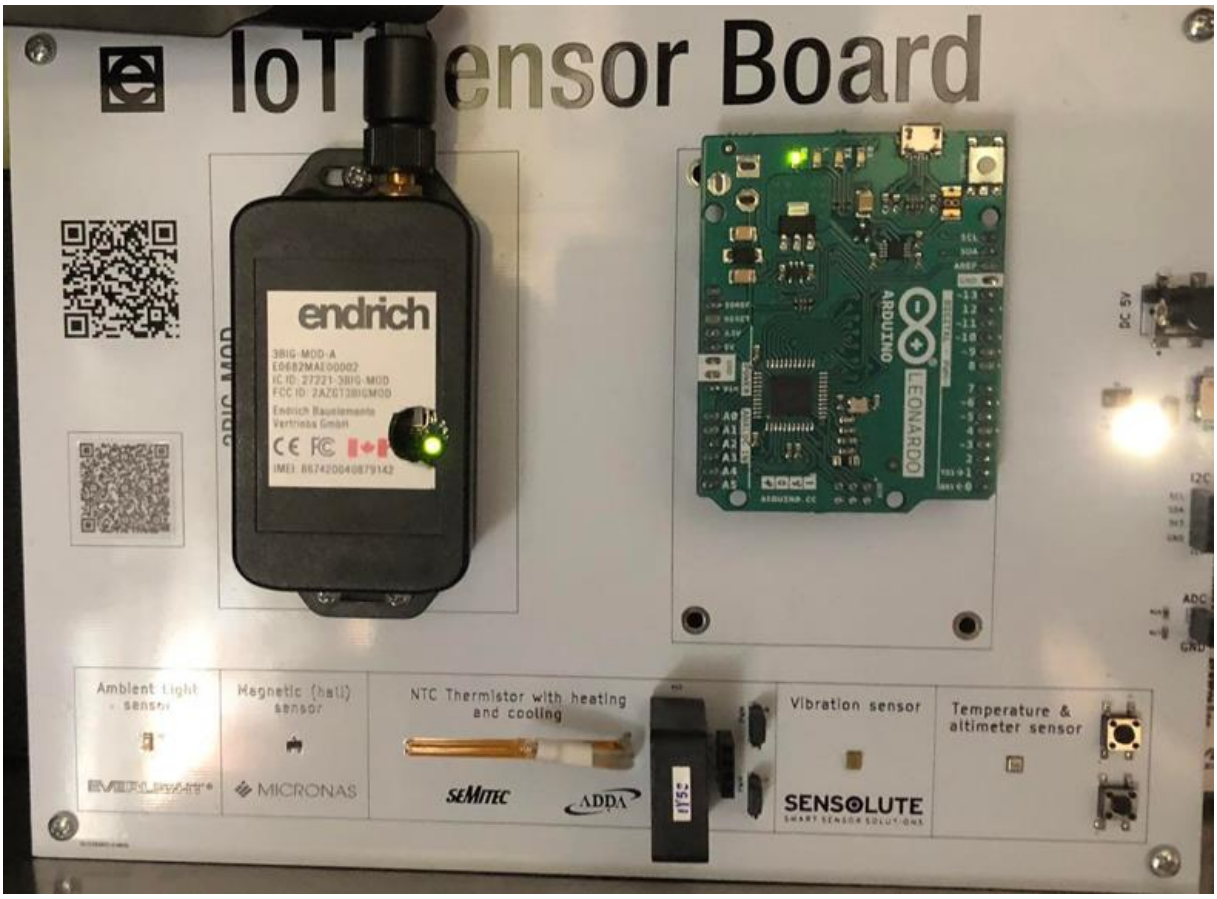


Ehhez a mikrovezérlőhöz van hozzácsatlakoztatva I^2C-n keresztül egy digitális hőmérő, és egy a látható fény tartományában működő fényérzékelő. Ez utóbbiból a mikrokontroller analóg bemenetén keresztül feszültségméréssel tudjuk kinyerni az adatokat (analog-digital converter (**ADC**)). Az adatokat a mikrokontrollerre írt C++ könyvtár kezeli, ennek segítségével formázhatjuk a felhőszolgáltatásnak küldendő üzenetet (JSON). A formázást követően a programnak tovább kell tudnia küldeni az adatokat a GSM modulnak, ami AT parancsok segítségével továbbítja azokat a központi felhőbe.

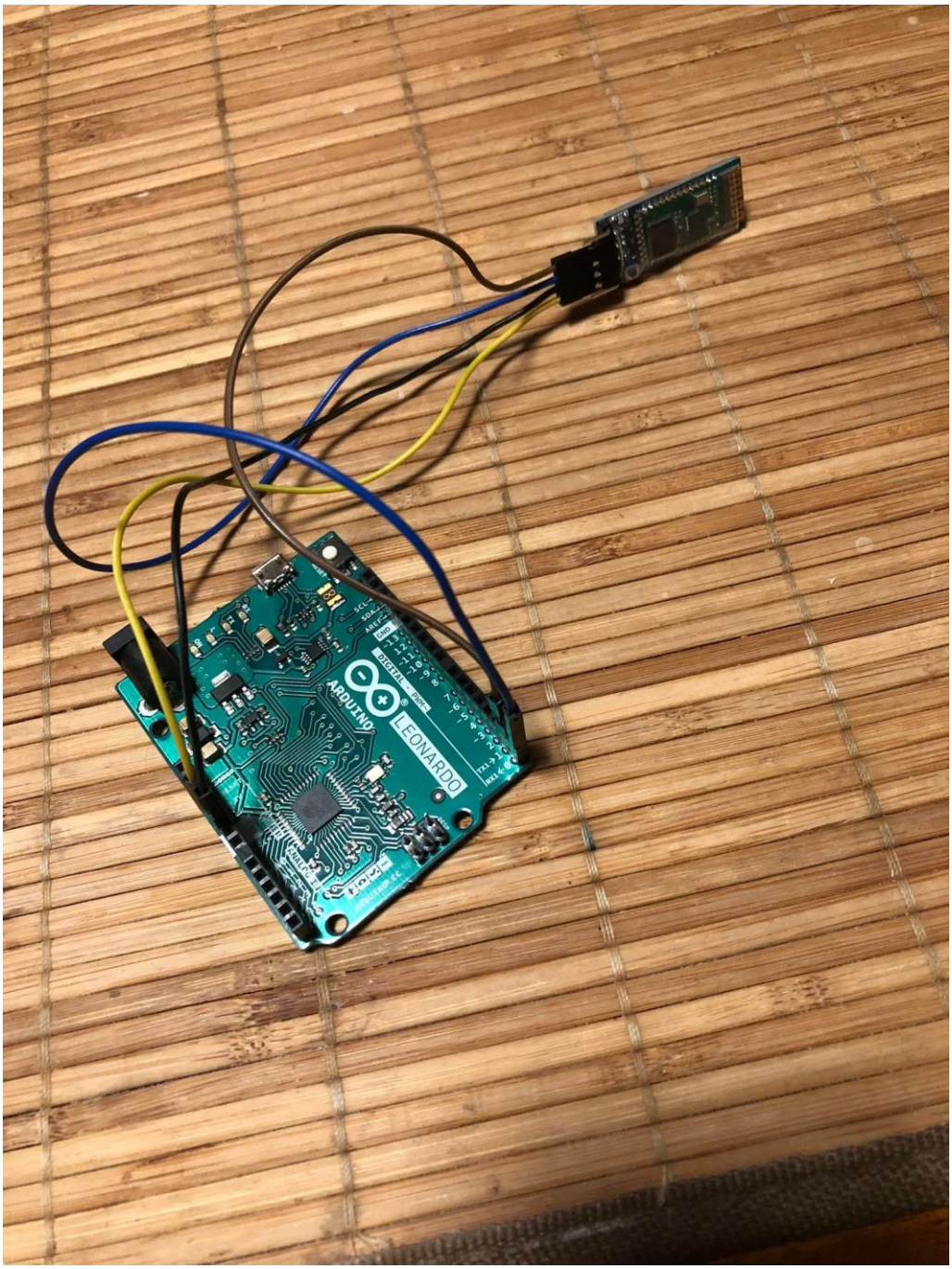
Ezen a board-on található egy MA510-es GSM modul, ami pontosan kielégíti a projekthez szükséges elvárásokat. A modem rádiós kapcsolatot három sávon képes a GSM szolgáltató felé létesíteni, NB-IoT, LTE-M és 2G technológiák felhasználásával, és a hálózatra regisztrálva UDP kapcsolatot létesít a felhőszolgáltató szerverével. Az adatok JSON formátumban kerülnek feladásra. A modem a GSM hálózaton csak az adatküldés idejére van jelen, ezzel is csökkentve a terhelést és minimalizálva a fogyasztást. Felhő alapú szolgáltatásként az E-IOT CDB központi Cloud Server-t használtam. A szerverről az adatokat az Endrich API-kal tudtam lekérni.

Az adatok megjelenítéséhez egy Androidos applikáció fejlesztésére volt szükségem, amelyben le tudom kérni az adatokat a felhő szolgáltatásból az API-n keresztül és ezeket meg tudom jeleníteni egy felhasználóbarát, diagram alapú grafikus felületen.

Az alábbi képen a mér működésben lévő IoT board látszik. Folyamatosan küldi az adatokat a felhőbe két perces intervallumonként.



A feladatnak a második fele az otthoni okos eszközök vezérlésének szimulációja. Ehhez sajnosnem tudtam ezt a board-ot használni, ugyanis nem rendelkezik további UART-al (soros port), hogy a Bluetooth modult rá lehessen kötni. Ahhoz, hogy működjön, egy másik Arduino Leonardo mikrokontrollerre volt szükségem, amin egy újonnan írt C++ programkód fut. A légkondicionáló és redőny berendezés modellezésére pedig egy Android telefonon futó Bluetooth terminál emulátort használtam, ami képes fogadni a Bluetooth modul által küldött adatokat.



A projekt megvalósítása

A projekt megvalósításának az első lépése az volt, hogy beszerezzem az IoT board-ot és megértsem annak működését. Ezt követően tudtam csak elkezdeni a forráskód írását. A forráskód határozza meg, hogy miként működjön az MCU. Az én esetemben a következő funkciók megvalósítására volt szükség: adatok kinyerése a szenzorokból, az adatok JSON-ba való konvertálása, UDP kapcsolat létrehozása és adatok továbbítása a szerverre.

Az adatok kinyerését mind a digitális hőmérséklet szenzor, mind az analóg fényszenzor esetében hasonló függvényekkel kiviteleztem. Az első esetben egy beépített könyvtári függvény segítségével kértem le az adatokat a szenzorból, a második esetben pedig az Arduino C++ programnyelvében létező analogRead paranccsal, feszültségméréssel nyertem ki a fényerővel arányos jelet, és alakítottam át fényerő adattá.

```
int AmbientLightSensor::readSensor(int input_pin){
  return analogRead(input_pin);
}

int AmbientLightSensor::getIntensity(){
  int raw_data = readSensor(A2);
  return (int)raw_data*0.263; //converts to lux
}
```

```
float TemperatureSensor::readSensor(){
  // TE I2C Pressure/altitude and temperature sensor library
  MS5637 barometricSensor;
  barometricSensor.begin();
  return barometricSensor.getTemperature();
}

float TemperatureSensor::getTemperature(){
  return TemperatureSensor::readSensor();
}
```

A következő feladatom az volt, hogy megvalósítsam az adatok küldésének folyamatát. Ezt az MA510 osztályban végeztem el. Itt különböző függvényeket írtam, amelyekre szükségem volt a modem működtetéséhez.

A szerverre való feltöltéshez egy a szolgáltató által előírt formátumú JSON üzenetre volt szükség, ami tartalmazza a modem IMEI számát, a SIM kártya CCID-ét, valamint a feladásához szükséges UDP kapcsolat létesítéséhez szükség volt a távoli szerver IP címére is, melyet DNS rekord feloldásával a cloud.e-iot.info URL-ből kellett kiolvasnom. A fentiek megvalósításához egy komplett könyvtárat írtam. Az ehhez tartozó kódrészlet az alábbi képen látható.

```
String MA510::getIMEI(){
  String imei = "";
  Serial1.println("AT+CGSN");
  delay(1000);
  while(Serial1.available() > 0){
    if(Serial1.available() > 0){
      imei += (char)Serial1.read();
    }
  }
  int idx = imei.indexOf("AT+CGSN");
  String final = imei.substring(idx+10, idx+25);

  Serial.print("IMEI: |");
  Serial.print(final);
  Serial.println("|");

  return final;
}
```

```
String Sensor::createJSON(String myimei, String mymsgRef, String mygpsData){
  AmbientLightSensor am_light_sensor;
  TemperatureSensor temp_sensor;
  String myJSON="";
  String myTEMP = temp_sensor.convertToPayloadData("T", 6, 100);
  Serial.print("|");Serial.print(myTEMP);Serial.println("|");
  myJSON = "{\"IMEI\": \""+myimei+"\" , \"msgref\" : \""+mymsgRef+"\", \"payload\" :\"|"+am_light_sensor.convertToPayloadData("A", 6, 100)+"|"+myTEMP+"|\",\"gpsdata\" :\""+mygpsData+"\"}";
  return myJSON;
}
```

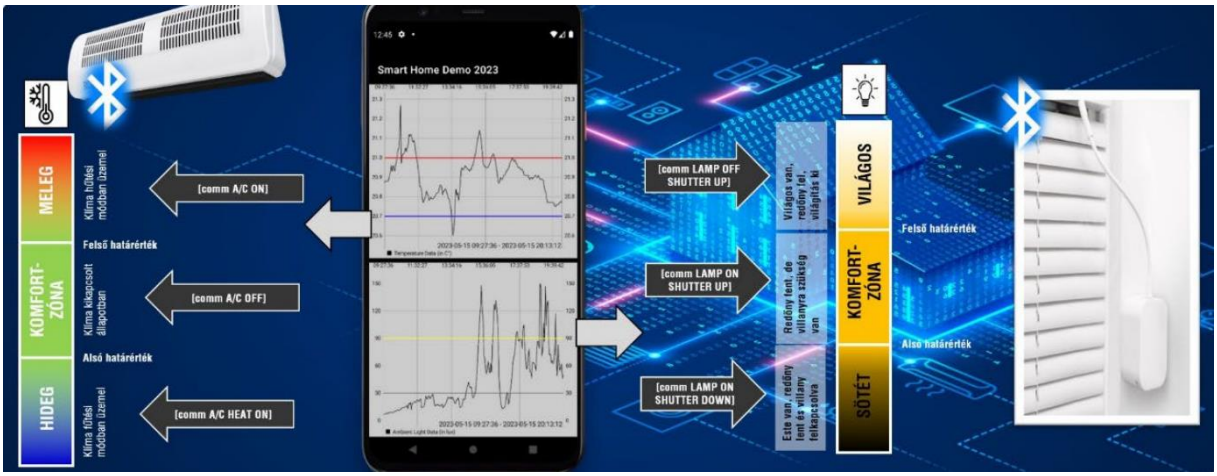
Így már rendelkeztem a megfelelő formátumú üzenettel, melyben a szenzor adatok a payload mezőben helyezkednek el. A teljes üzenetet bináris formátumba kellett alakítani a szerverre való küldéséhez. Ehhez először fel kellett építenem egy UDP kapcsolatot, majd a formázott JSON-t átküldeni rajta (AT parancsok). Ezután már csak le kellett zárni a kapcsolatot, hogy a modem lekapcsolódjon a hálózatról a terhelés csökkentése és az energiatakarékosság miatt.

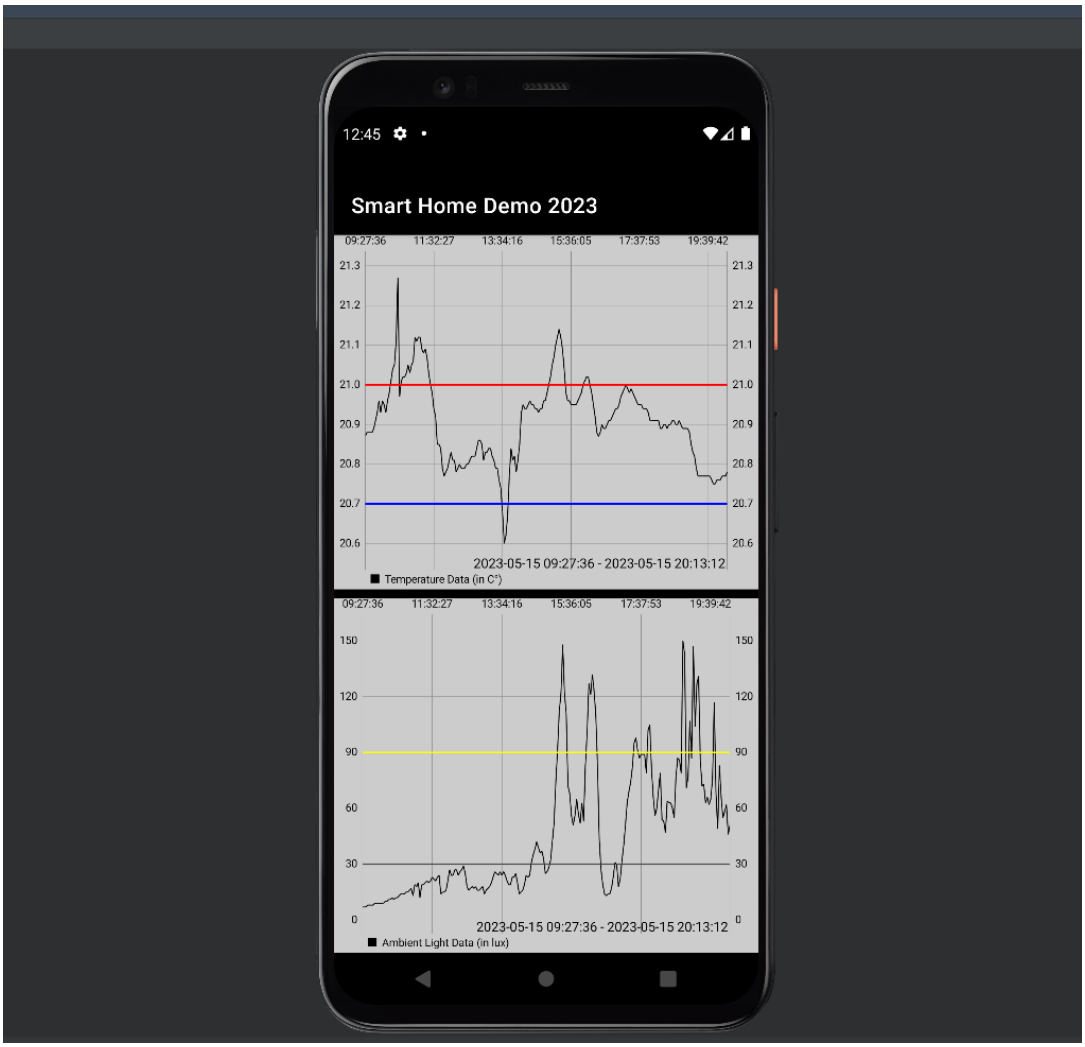
Ezen a ponton az IoT board programkódja teljes mértékben elkészült, így már csak annyi teendőm volt vele, hogy elhelyezzem egy szoba egy adott pontján és hagyjam dolgozni.

A következő lépés a projekt kivitelezésében az volt, hogy az adatokat lekérjem és megjelenítsem egy Android applikációban. Ehhez először meg kellett valósítanom a

szerveren meghívott API kérésre adott válasz modelljét. Ezek a modellek tárolják el a lekért adatokat. Ezt követően a modellekből kiválasztottam azokat az adatokat, amelyekre szükségem volt (hőmérséklet, fényerősség, mérések időpontjai), majd betöltöttem őket a diagramokba. Található mind a két diagramon két-két konstans érték is, ezek jelentik az optimális hőmérséklet és fényerősség mértékét. A repository-ban található egy videó részlet is a telefonos applikáció működéséről.

Az utolsó lépés az volt, hogy a légkondicionáló és a redőny szimulációját elkészítsem. Ezt úgy tettem meg, hogy egy új Arduino Leonardo MCU-t összekötöttem egy Bluetooth modullal soros porton keresztül (transmit/receive (**Tx/Rx**)), majd írtam rá egy egyszerű programot. A program úgy működik, hogy amikor a felhasználó az Android telefonon futó Bluetooth terminál emulátorba beírja a hőmérséklet (pl: **T/t23**) és fényerő (pl: **L/l39**) adatokat, akkor a telefon a Bluetooth kapcsolaton keresztül ezeket átküldi az Arduino számára, ami ezeket kiértékelve parancsokat küld ugyanezen a csatornán vissza a szimulált okos eszközök számára. Ilyen parancs lehet például, hogy induljon el a légkondicionáló hűtési vagy fűtési üzemmódja, vagy hogy menjen fel vagy le a roló.





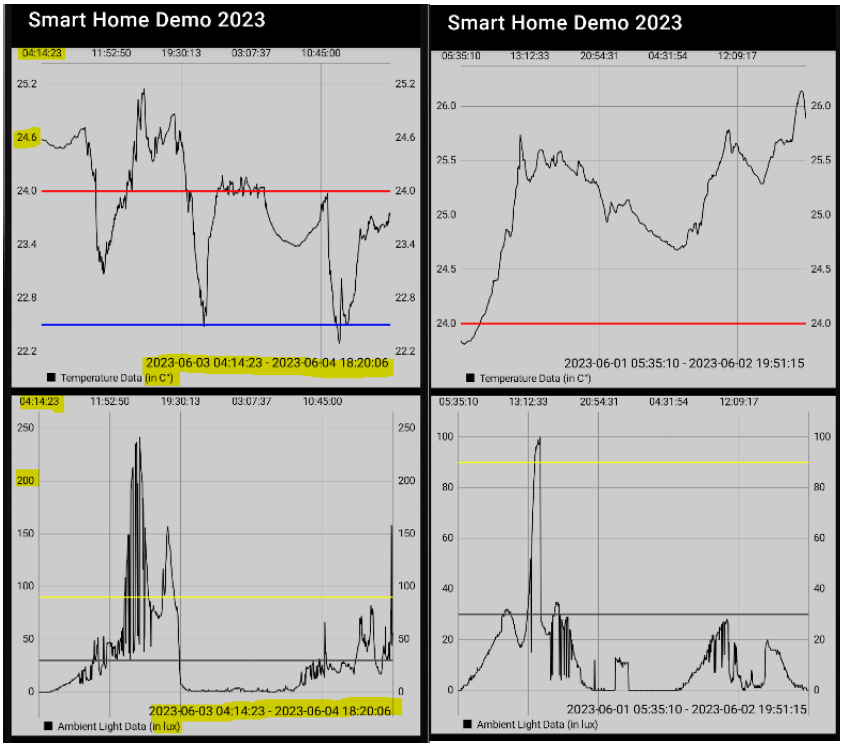
Felhasználás és működés

Ahhoz, hogy működésbe tudjon lépni a projekt, a felhasználónak el kell helyeznie valahol az IoT boardot, majd egy tápegységet kell rá kötnie, amely folyamatos áramellátást nyújt a készüléknek. Ezt követően az eszköz működik magától, egészen addig, amíg van áramellátása, vagy ki nem kapcsolja a felhasználó. A mikrokontroller a C++ forráskódban meghatározott módon leolvassa az adatokat a szenzorokból, majd átkonvertálja azokat JSON formátumba. Ezután a GSM modem létrehozza az UDP kapcsolatot a szerverrel, átküldi neki az üzenetet és végezetül lebontja a kapcsolatot. A küldés nagyjából két perces intervallumokban történik.

Jelen esetben az Androidos applikáció az emulátorban fut, de ezt fel lehetne tölteni egy felhasználó készülékére is. Ha a felhasználó elindítja az appot két grafikon jelenik meg a

képernyőn. A felső grafikonról leolvasható a mért hőmérsékletek értéke, míg az alsóról a fényerősség (utolsó 1000 adatig). Mind a két diagramon a mérés időpontjának a függvényében (X-axis) jelennek meg az adatok óra:perc:másodperc formátumban. Ezen felül a diagram alsó részén látható a mérésnek az intervalluma is. Az alsó kép pillanatában az első adatot 2023.05.15-én mérte a board délelőtt fél tíz fele. A diagramokon továbbá található két-két konstans vonal is. Ezek a vonalak jelzik az általam definiált optimális fényerősség és hőmérséklet tartományokat, amiken kívül valamilyen cselekvést kell végrehajtania a szimulált eszközöknek, mint például hűtési funkció elindítása az AC-n.

A légkondicionáló berendezést és a redőnyöket egy szimuláción keresztül tudja irányítani a felhasználó. Ahogy azt már írtam korábban is, egy külön Arduino Leonardo mikrokontroller vezérli ezt a szimulációt. A felhasználásra való készülékre le van töltve egy Bluetooth terminál emulátor, amibe, ha a felhasználó ír be valós hőmérséklet vagy fényerősség adatokat a korábban definiált módon, akkor válaszként megkapja a tevékenységet az MCU-tól, ami automatikusan működne a valóságban. Erre a mikrokontrollerre is írtam egy C++ forráskódot, amiben ellenőrzöm, hogy a megadott határértékekhez képest mennyire helyezkedik el a készülékről bevitt adat, és ezekhez az esetekhez definiáltam a már korábban említett cselekvéseket.




```
  if (actualTemperature > highTempTreshold){
    Serial.println("AirCo Cooler ON, too warm here");  // For displaying on local Serial port
    Serial1.println("AirCo Cooler ON , too warm here"); // Sending data to the HC-05 BT module
    Serial1.flush();
  } else {
    if (actualTemperature < lowTempTreshold) {
      Serial.println("AirCo Heater ON, too cold here");  // For displaying on local Serial port
      Serial1.println("AirCo Heater ON, too cold here"); // Sending data to the HC-05 BT module
      Serial1.flush();
    } else {
      Serial.println("AirCo OFF, convenient here");  // For displaying on local Serial port
      Serial1.println("AirCo OFF , convenient here"); // Sending data to the HC-05 BT module
      Serial1.flush();
    }
  }
}
```

Nehézségek a kivitelezés során

A projekt megalkotása során számos nehézséggel néztem szembe. Az egyik legnagyobb kihívás számomra az volt, hogy megismerkedjek egy olyan hardverrel, aminek sem a működéséről, sem a működtetéséről nem tudtam sok mindent a projekt kezdete előtt. Számomra minden eszköz és szenzor, ami a hardverre volt csatlakoztatva, teljesen idegen volt. Mivel ennek a feladatnak a hardver a legfontosabb része, ugyanis enélkül nem lehetne adatokkal dolgozni, különösen sok energiát kellett belefektetnem az eszközök megismerésébe. Miután megértettem mi hogyan is működik szembesültem a következő nehézséggel: a mikrokontroller fejlesztésével.

A vezérlő kódjának megírásával töltöttem el a legtöbb időt, mivel arra törekedtem, hogy minden elképzelésemet meg tudja valósítani a működése. A hardverre történő programozásnak is megvoltak a saját kihívásai, ugyanis itt nem volt elég csak a szoftver helyes működésére odafigyelni, még a hardvernek az igényeit és képességeit is figyelembe kellet vennem. A programozás közben kerültem szembe a projekt legsúlyosabb problémájával, azzal, hogy nem volt elég soros port az IoT board-on ahhoz, hogy a Bluetooth modult hozzá tudjam kötni.

Emiatt az egész feladatot újra kellett gondolnom, és elő kellett jönnöm egy másik megoldással, aminek a működése nagyon hasonló az eredetileg elképzelt működéshez. Ez az oka annak, hogy nem tudom automatikusan működtetni a szimulációt, hanem manuálisan be kell írnom a hőmérséklet és fényerősség értékeket a telefon Bluetooth terminál emulátorába.

A változtatásnak is megvoltak a maga kihívásai. Először is kellett hozzá egy új Arduino Leonardo mikrokontroller, amire egy külön C++ kódot kellett írnom. Miután rácsatlakoztattam a Bluetooth modult egy régi Android telefonnal tudtam hozzá csatlakozni. Azonban, ha battery pack-et kötöttem az MCU-hoz, akkor folyamatosan leállt a vezérlő. Ez amiatt történt, hogy a kontrollernek olyan kicsi volt az áramfelvétele, hogy a battery pack azt hitte, hogy nincs hozzácsatlakoztatva semmi, így lekapcsolta az áramellátást. Emiatt a szimuláció futtatása sokkal nehezebbé vált, és több időt is vett igénybe.

Az Android fejlesztésben annyi kihívás volt, hogy még nem foglalkoztam igazán gráfok megjelenítésével. A diagrammok adatokkal való feltöltése bizonyult az egyik legnehezebb feladatnak.

Felhasznált forrásaim

E-IoT platform HARDWARE & SOFTWARE USER manual: https://e-iot.info/

Magyar Elektronika 2023 június: Alkalmazási példa – Okosotthon, világítás, roló- és klímavezérlés (Endrich-es publikáció)

Arduino Leonardo: https://docs.arduino.cc/hardware/leonardo

API hívás: http://cloud.e-iot.info:82/iot-demo/getdata-ios.php?IMEI=867420040879142

Android charts: https://github.com/PhilJay/MPAndroidChart