Fejlesztői dokumentáció

Mikroelektromechanikai rendszerek (GKLB_INTM020)

Készítette:

Márton Arnold Levente és Bán Tamás

Tartalom

Fejlesztői dokumentáció 1
1. Bevezető 3
2. Hardver és szoftver leírása 4
2.1 Arduino UNO 4
2.2 Szoftver környezet 6
3. Fejlesztés 8
3.1. Felhasznált eszközök 8
3.2 Felmerülő probléma 10
3.3 Kód részletezése, program működése 10
3.4 Rendszerkövetelmény 11
3.5 Kapcsolási rajz 12
3.5 Ubidots adatbázis, tesztelés 13

1. Bevezető

A téma választás során próbáltunk a meglévő dolgainkból, alkatrészekből összeállítani a feladat teljesítéséhez szükséges rendszert megvalósítani. Az alapot egy Arduino UNO adta, ehhez társult hozzá az alábbi részegységek, amiket a projekt során felhasználtunk:

- Arduino UNO
- B10k analóg potenciométer
- 1 csatornás relé modul
- Wiznet W5100 ethernet shield
- mágneses nyitásérzékelő
- DALLAS DS18B20
- 4 számjegyű 7 szegmenses kijelző
- Breadboard, ellenállások, vezetékek

A feladat egyeztetése során szobatermosztát feladat megvalósítása mellett döntöttünk. A termosztát beállítását egy potméter segítségével valósítottuk meg amit a felhasználó tud szabályozni kézzel. A szoba hőmérsékletét az eszközhöz csatlakoztatott hőmérő mér. Ezt a két értéket a kijelzőn látjuk, jobb oldalon a kívánt hőmérséklet látható, a bal oldalon pedig a szoba hőmérsékletét látjuk. Az értékek felváltva jelennek meg a kijelzőn. Az eszközhöz raktunk kiegészítőket, ilyen például a nyitás érzékelő, ami nem engedi működtetni a fűtést, ha pl.: nyitva maradt ajtó vagy ablak van a házban. Ebben az esetben a relé nem kap áramot, hiába magasabb a cél hőfok a jelenleginél. A relé működését egy visszajelző led segítségével látjuk, hogy be van-e kapcsolva vagy sem. Ez később lehet cserélni 230V-os ra amivel lehet kapcsolni fűtőtestet. Ami tovább okosítja a rendszert, hogy adatbázisba tároljuk a beállított és az adott helységben elhelyezett termosztát által mért értéket. Ebből lehetőségünk van kimutatás készíteni, akár grafikonra kirajzolva látjuk, hogy változik az érték. Emellett rögzíti a fűtés mennyi időt működött, mikor lett bekapcsolva és mikor kapcsolt ki ha elérte a kívánt hőmérsékletet vagy a felhasználó kikapcsolta azt.

2. Hardver és szoftver leírása

Az Arduino egy nyílt forrású fejlesztőplatform, ami mostanában elég elterjedt a hasonló Rasberry fejlesztőplatform mellett. Az Arduino-t pont ilyen egyedi otthoni fejlesztésekhez hozták létre, emellett célközönségként, akár oktatásban is jól kihasználható eszköz. Az Arduino el tud látni akár automatizálási feladatokat is és jó vezérlő szerepet tud játszani okosotthon létrehozásában is.

Ahogy a képen is látható, a mi általunk választott alap az Arduino platform részét képző UNO elektronikai áramköri lap és a szoftveres környezet. Ezen kívül többféle verzióban megtalálható maga az áramkör.

2.1 Arduino UNO

Az UNO verzió talán ez első vezérlők egyike, ami talán mondható a legelterjedtebb verziónak is. Az Arduino UNO áramköri lap az egyik legelterjedtebb a sokféle kártya közül.

Az Arduino lelke a mikrovezérlő, az UNO esetében egy pontosan egy „ATmega328/P” típusú, 8 bit kezelésére képes mikrovezérlőt ölel magába. A legfontosabb perifériák egy általános mikrovezérlőben a következők lehetnek:

- Időzítő áramkörök
- Analóg és digitális be- és kimenetek
- Kommunikációs perifériák stb.

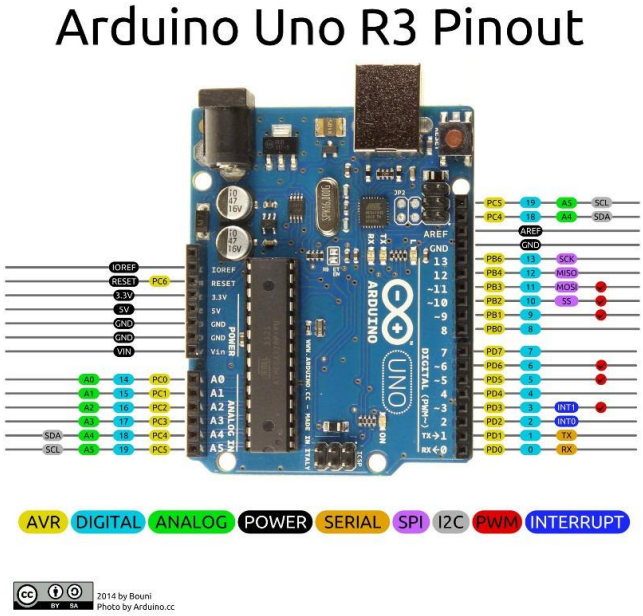
Az Arduino UNO áramköri lap arra szolgál, hogy a lapba beépített mikrovezérlőnek a lábai ki vannak vezetve a könnyebb és kényelmesebb hozzáférhetőség érdekében egy csatlakozókra,

valamint még itt találhatóak meg a mikrovezérlő működéséhez és programozásához szükséges és nélkülözhetetlen alkatrészek.

Ezek a mikrovezérlők jóval kisebb számítási teljesítménnyel rendelkeznek összehasonlítva egy hagyományos számítógéphez képest, így az áram fogyasztásuk és méretük is sokkal kisebb. Mivel a legtöbb esetben nem fut rajtuk operációs rendszer, ezért a rendelkezésre álló erőforrásnak megfelelő valós-idejű feladatokra is alkalmazhatók, azaz egy adott eseményre a mikrovezérlő gyorsan és egy garantált maximális idő alatt képes reagálni.

ATmega328/P mikrovezérlő paraméterei:

- 8 bites architektúra
- 16 MHz-es órajel
- EEPROM (csak olvasható memória): 1 KB
- SRAM (változók tárolására): 2 KB
- A flash (programot tároló ROM): 32 KB



UNO áramköri alkatrészek.

A lábkiosztás akkor szükséges, ha a külső áramköröket is használni akarjuk, ez akkor is fontos lehet, ha a kódból tudjunk hivatkozni az adott kivezetésre. Az Arduino áramköri lap két szélén lévő csatlakozó aljzat melletti számokkal érjük el az adott kivezetést a kódból. Ezek a kivezetések lehetnek digitálisak (0-13-ig) vagy analóg bemenetek (A0-A5-ig). Az A0-A5 kivezetések tudnak lenni akár digitális be- és kimenetek is, ha a kódban úgy állítjuk be az adott kivezetést. Külső áramkörök esetén a fix feszültségű kivezetésekről kényelmesen tudunk

tápfeszültséget adni az áramkör számára. A csatlakozókhoz kényelmesen köthetünk szenzorokat, digitális IC-ket és egyéb külső áramköröket. Ahhoz, hogy ezeket biztonságosan tudjuk használni, előbb meg kell ismernünk a be- és kimenetek működésének korlátait. Az eszköz működési feszültsége 5V, a digitális kimenet maximális terhelése 20mA. A 3.3V -os kimenetből pedig max 50mA tudunk használni.
Az feszültségszintekre és áramkorlátokra nagyon figyelni kell az Arduino használata során. Csak egyforma feszültségszinteken üzemelő áramköröket szabad összekapcsolni, az áramerősségeket pedig a számolás során kapott értéknek megfelelő, sorba kapcsolt ellenállással lehet a maximális megengedett érték alatt tartani.

Az Arduinohoz számos "feltét" áramkör is kapható, ezeket a ráépíthető lapokat Arduino shieldeknek neveznek. Ezeknek az áramköri lapoknak az előnye, hogy egyszerűen, egyetlen mozdulattal rá lehet helyezni az Arduinora és máris kiegészítettük az Arduino-t valamilyen hasznos funkcióval. A kompatibilitásra figyelni kell, mert léteznek 3,3 V tápfeszültséget igénylő shieldek, amik tönkremehetnek az 5 V-os jelszintet használó Arduino UNO-tól.

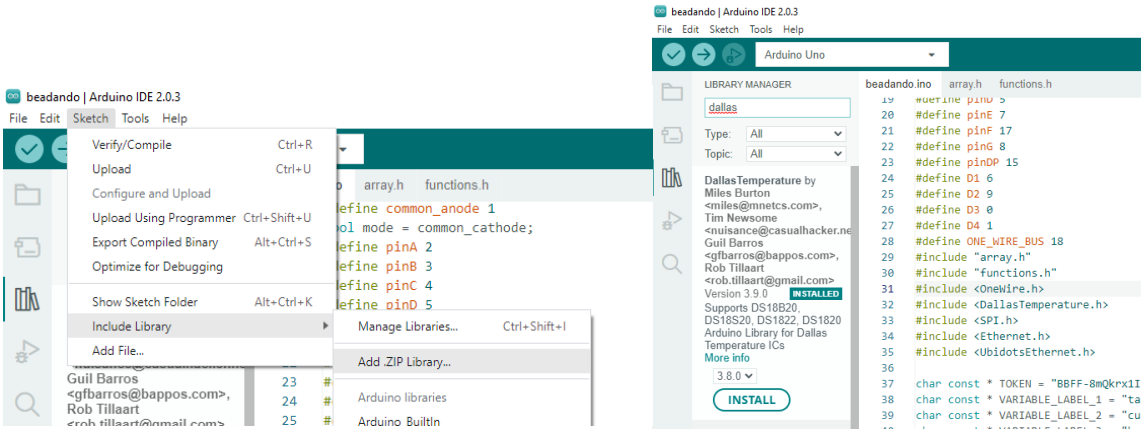
Az ATmega328/P lényegesebb belső perifériái és gyakorlati alkalmazásaik:

- Digitális be- és kimenet: digitális jelek, állapotok beolvasásához és kiadásához
- PWM: vezérelhető kitöltési tényezőjű digitális jelhez
- ADC: analóg feszültségek méréséhez
- UART: soros kommunikáció a számítógéppel, külső digitális áramkörökkel

2.2 Szoftver környezet

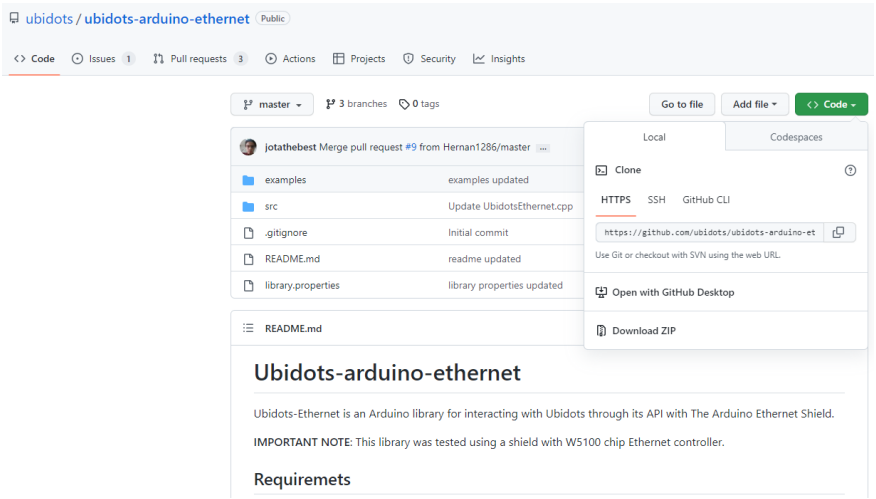
A fejlesztést az Arduino-hoz rendelkezésre álló Arduino IDE-val valósítottuk meg. Az általunk használt verzió a 2.0.3 a modern szerkesztőplatformon (az Eclipse Theia-n) alapul, A frissített felhasználói felület mellett az új IDE legfontosabb fejlesztése a működési sebességének növelése. Ez elsősorban az Arduino nyelvi szerveren belül végrehajtott optimalizációknak köszönhető - ugyanakkor más fejlesztések a programozók számára is gyorsabb haladást tesznek lehetővé. Ilyen például az új kódkiegészítő és kódsegéd, amik nem csak a kódok bevitelét gyorsítják meg, de már gépelés közben jelzik a hibákat.

Library manager megnyitás (3 oldalfül), itt több libary-t kell telepítenünk, Egyiket bemutatva többit pedig felsorolva láthajtuk. Elsőnek kikeressük a dallas, ez ha telepítve van akkor tudjuk a hőmérsékleti adatokat kezelni. Ez a „dallas temperature.h” használatához kell
és ezzel együtt feltelepíti a „onewire.h”-t is. bekerül ebbe az arduino IDE-ba.



Ezen felül telepítenünk kell még az ethernet sield-hez szükséges „ubidots-arduino-ethernet” libery-t is. Ebbe a zip-be van benne az ubidots.h amiben megvan minden függvény megvalósítása, ő fordítja át linké amit el tud küldeni.

„#include <UbidotsEthernet.h>” Összeollózza az értékeket, csinál egy linket+tokent(saját token, én generáltam) és így küldi el ami bekerül az adatbázisba.

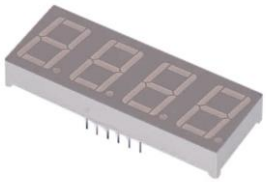


3. Fejlesztés

3.1. Felhasznált eszközök

A fejlesztés során az alább eszközöket kötöttünk össze az Arduino UNO eszközzel.

A hőmérsékleti adatok kijelzésére egy **4 számjegyű 7 szegmenses kijelzőt** építettünk be. Ahogy írtam bevezetőben is, ez mutatja a potméter segítségével beállított cél hőmérsékletet és az aktuálisan mért hőmérsékletet. Ezeket az értékeket felváltva mutatja, a kívánt érték jobbra igazított, az aktuális érték pedig balra láthatjuk a kijelzőn. A kijelző az alábbi képen látható.

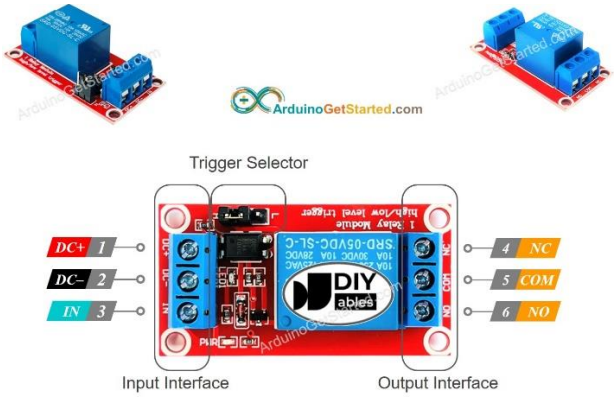


A hőmérséklet mérésére a **DALLAS DS18B20**, ami a valós hőmérsékletet + 0,5 °C elétéréssel tud mérni.

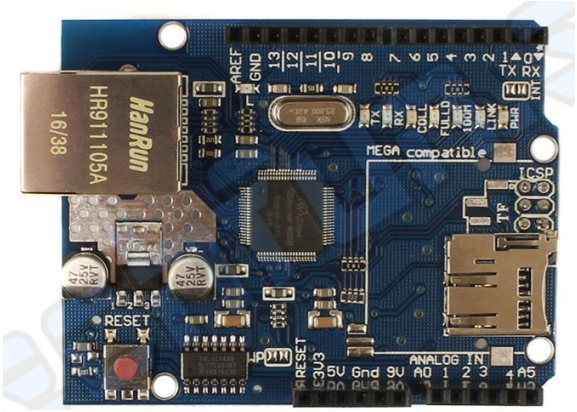
B10k analóg potenciométer segítségével 0-tól 31ig lehet értékeket beállítani, ez a cél hőmérsékletet. Alapból 1024 értéket határoz meg(0-1023), jelen esetben /32-vel osztottuk az értéket, mert számunkra elegendő az így kapott érték

Mágneses nyitásérzékelő, ha nyitást érzékel, kikapcsolt állapotba állítja a relét ehhez 2 vezeték szimpla megszakítás kell.

1 csatornás relé modul ennek a segítségével kapcsolhatjuk ki és be a fűtőtestet, (tesztelés és fejlesztés során egy led-et kapcsoltunk ki-be, ezzel tudtuk szimulálni a fűtés be és kikapcsolását). Ez a relé COM port-jára bekötött 230V-os fázis kapcsolás során a NO-ra továbbítja az áramot, ha a relé ON állapotba kerül. Ez a relé 230V 10A -ig lehet terhelni, így akár egy keringető szivattyú vagy egy kisebb teljesítményű fűtőtest kapcsolására jó.



Arduino UNO hálózati csatlakozását egy shield hozzáadásával oldottuk meg, ami egy **Wiznet W5100 ethernet** shield, amivel hálózati elérést tudjuk biztosítani ahhoz, hogy csatlakozhatunk a vezetékes hálózatra. Ez azért szükséges, hogy adatbázisszerver elérhessük és az adatokat tudjuk felölteni.



3.2 Felmerülő probléma

A fejlesztés során egységről egységre növekvényesen készült a projekt felépítése, mindig bővült a meglévő kész/működő rész, de a végén nem maradt elég szabad PIN láb amit ki lehetett volna vezetékezni, ezután újra tervezés volt szükséges, az összes eltervezett funkció használatához amit a teljes projekt során elterveztük. Nem akartunk egyetlen funkciót sem kihagyni, mivel így lesz tényleg teljes egy termosztát működése

A legnagyobb fejtörést az serial kommunikációról való áttérésnél jött szembe, itt kellett az összes PIN lábkiosztást újra tervezni, hogy az ethernet-hez szükséges 4 láb bekötését meg tudjuk valósítani. A 4 láb, amit ehhez kellett az a már felhasznált (kijelző ezen volt), így már tényleg nem volt más, teljes PIN lábkiosztás újra tervezésére volt szükség. Lehet, ha az elején a teljes projektet vettük volna pappíra és megtervezzük a teljes lábkiosztást akkor nem lett volna szükség ekkora újra tervezésre. Az új PIN kiosztásnál a serial monitor kikapcsolásra került, így tudtuk működtetni az kívánt egységeket.

Így kihasználjuk az összes PIN-t, ami az UNO eszközön található jelen esetben.

3.3 Kód részletezése, program működése

beadando.ino:

A kód elején a 7 szegmenses kijelző részeinek azonosítása található.

Ezután következnek a define-ok, melyek segítenek az arduino pineket lehet azonosítani, így olvashatóbbá teszik a kódot azáltal, hogy nem a pinek-hez tartozó eredeti számát használom, hanem az én általam deklarált konstans nevét.

Ezután jönnek a header fájlok, amik szükségesek az arduino-hoz csatlakoztatott komponensek használatához, majd az ubidots-al kapcsolatos konstansok és végül a változók. Adatbázisként az ubidots online szolgáltatását választottuk, egyetemi projekthez igénybe lehet venni egy ingyenes, csökkentett funkcionalitású fiókot, amivel egy darab privát token-t lehet létrehozni, ezáltal egy eszközhöz számunkra bőven, a célunkhoz elegendő volt.

A void setup() egyszer fut le az arduino bekapcsolása után, itt állíthatjuk be a pineket, szenzorok olvasását, ethernet-et, serial monitort (ezt sajnos inaktiválni kellett, mert a 0 és 1 pinek is használatba kerültek), ahogy erről fentebb a felmerülő problémáknál írtuk.

A void loop() funkció ismétlődik folyamatosan amíg az arduino be van kapcsolva. Itt történik az értékek kiolvasása, kijelzőre írás, relé kapcsolása, az adatok szerverre küldése.

Relé működtetésének a logikája: Ha ablak vagy ajtó nyitást érzékel a rendszer, a relét kikapcsolja, ellenkező esetben megvizsgálja, hogy a potenciométer nullás, azaz kikapcsolt állapotban van-e. Nullás értéknél nem kapcsolja be a relét, nullától eltérő érték esetén megvizsgálja, hogy a beállított kívánt hőmérséklet magasabb-e a jelenlegi hőmérő által mért hőmérsékletnél. Ha magasabb, a relét bekapcsolja, így bekapcsol a fűtőtest. Ha nem magasabb, akkor a relé kikapcsolt állapotba kerül.

array.h:

A kijelző által megjelenített karakterek találhatóak benne.

funcions.h:

A kijelző törléséhez és kijelzőre íráshoz használt funkciókat tartalmazó header fájl.

3.4 Rendszerkövetelmény

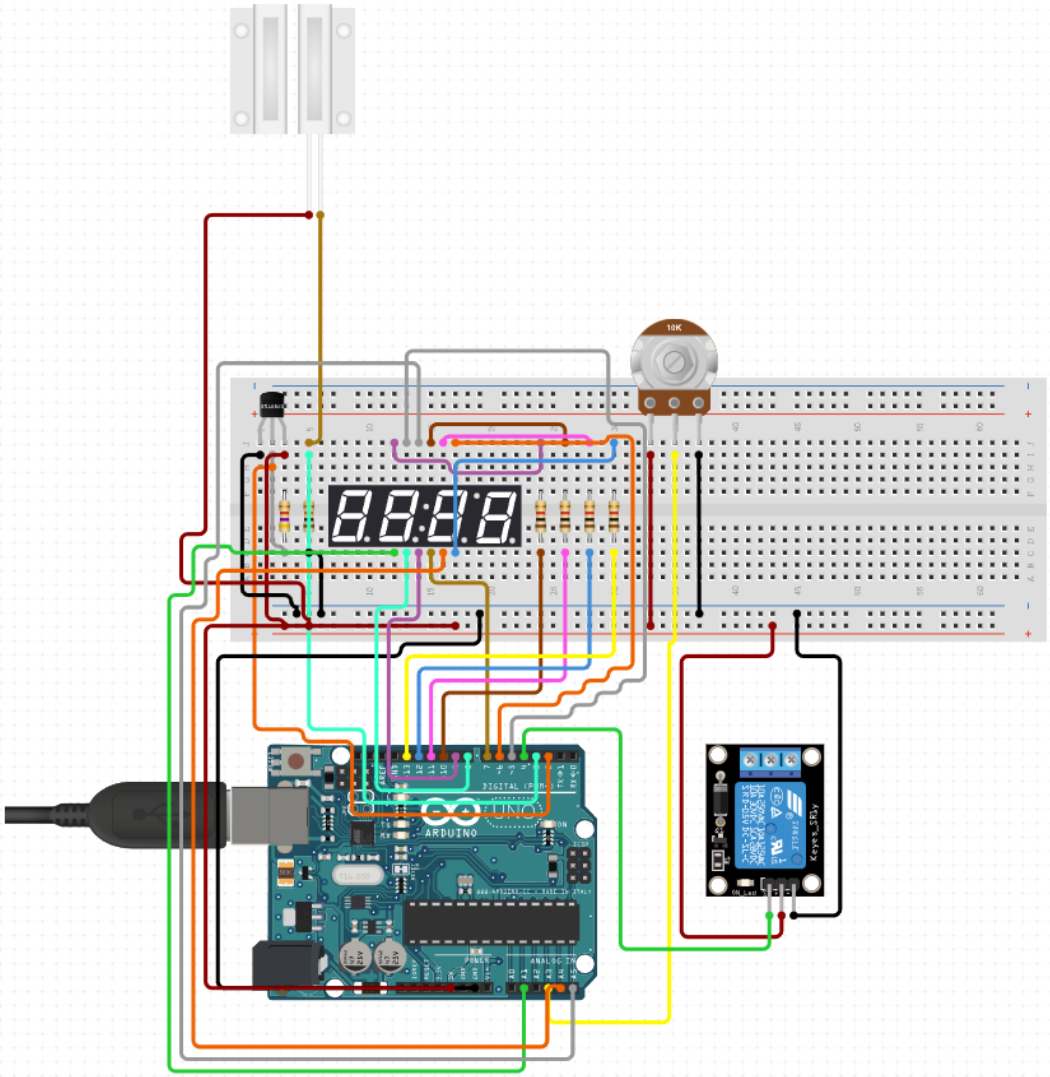
Gépi rendszerkövetelmény:

- Java futtatókörnyezet
- Windows 10 operciós rendszer
- kb 1 GB szabad tárterület
- kb 128MB memória
- kb 3000Mhz 1 magos processzor
- Ubidots web elérés (adatok gyűjtésére)
- legalább USB 2.0

Fejlesztői környezet:

- Git for windows 2.39
- Arduino IDE 2.0.e
- GitHub desktop

3.5 Kapcsolási rajz

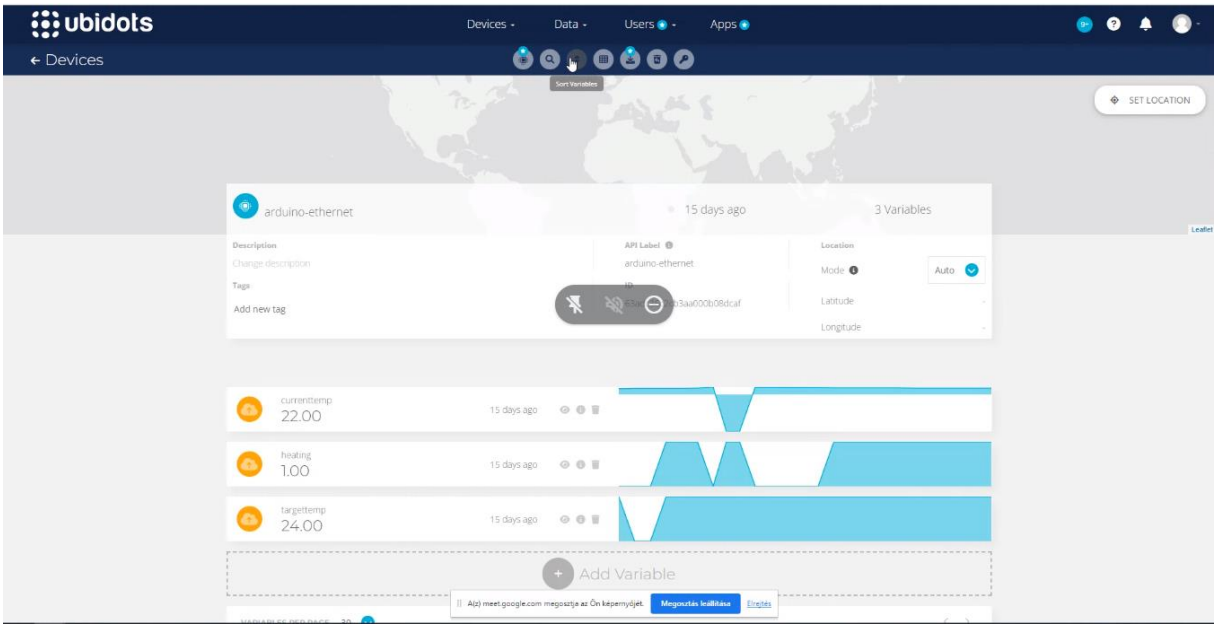


3.5 Ubidots adatbázis, tesztelés

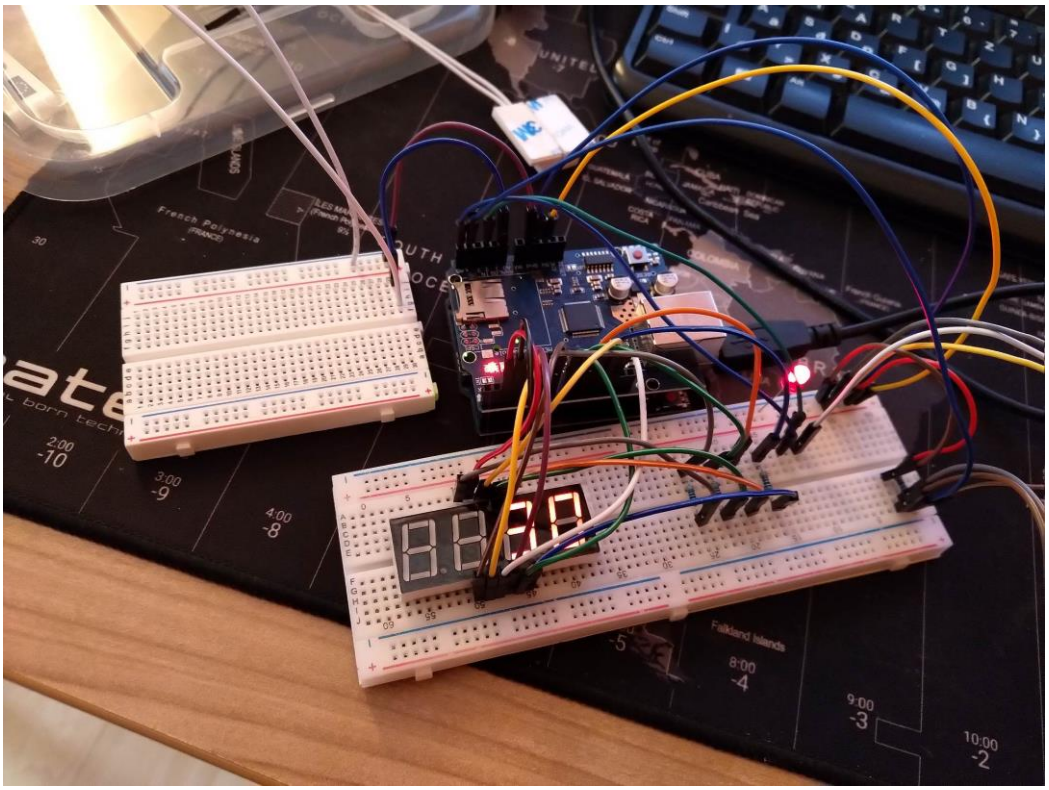
A lenti képen tudjuk visszanézni, mi történt használat során. Első képen például a hőmérséklet szenzor-t megszakítottuk így az érték -127 -es értéket vette fel. Ahogy látható a „currenttemp” 22 °C -ot jelez, ha működése megfelelő, ami valós szobahőmérsékletet jelent.

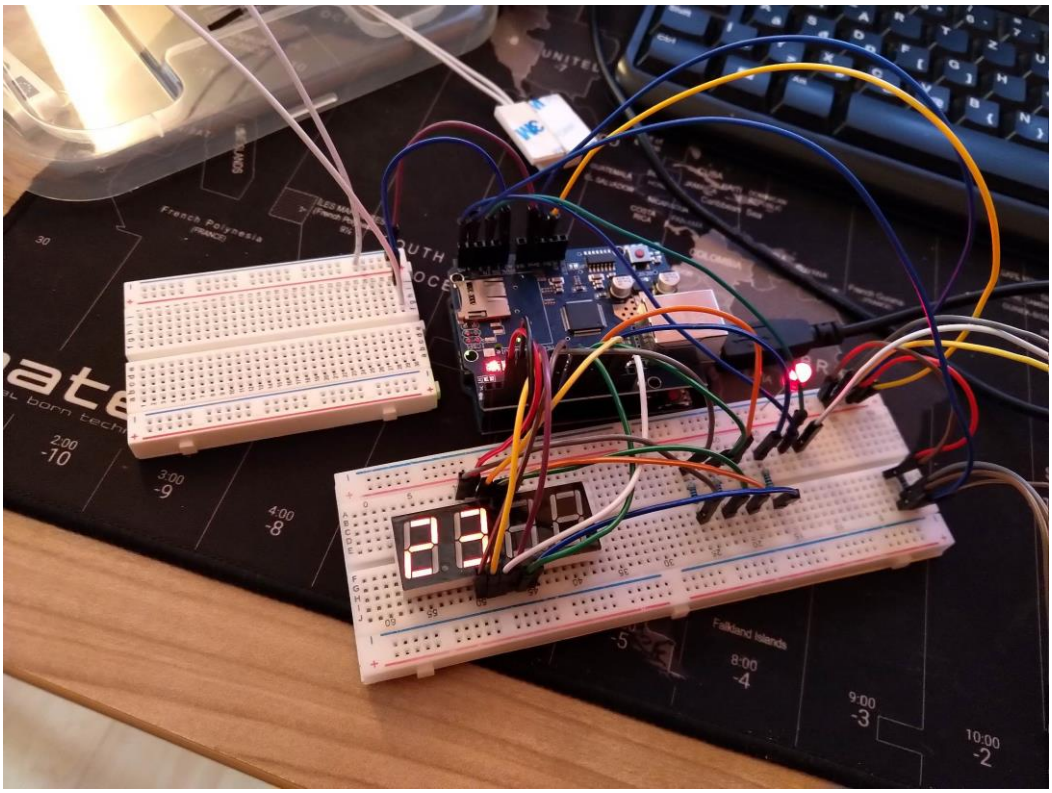
A „heating” -nél ami a második sorban van láthatjuk 1-es értéket vesz fel ha a fűtés működik, 0-ás értéket pedig ha nem működik.

A „targettemp” pedig a beállított hőfokot jelzi, amit a felhasználó állított be.



A lenti képek a projekt fejlesztése során készültek:





Ez a kép pedig a végső állapotot mutatja: