Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/3.

Épületekben használandó mérőműszer megvalósítása Arduino mikrokontrollerrel

Agonás Ágnes Fruzsina
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
agonas.agnes@gmail.com

Sarvajcz-Bánóczy Emese
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
emese.banoczy@eng.unideb.hu

Dr. habil. Husi Géza
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
husigeza@eng.unideb.hu

Absztrakt – Ezen kutatás keretein belül egy épületekben használandó mérőműszer készült el, mely alapját Arduino Mega ADK mikrokontroller adja. Az elkészült műszer épületek különböző paramétereinek mérésére alkalmas, melyek számítógépes adatbázisban kerülnek eltárolásra, ahol különféle Python nyelven megírt programokkal feldolgozhatóak épületenergetikai számítások megkönnyítése végett.

Kulcsszavak –Arduino, , ESP8266, GY-30 (BH1750FVI), GY-BME280, 2,4" TFT LCD, Python, épületenergetika.

I. BEVEZETŐ

Magyarország – és egyben a világ – éves energiafogyasztásának mintegy 40 %-át az épületek teszik ki, energetikai besorolásuk azonban viszonylag bonyolult és nem egységesített folyamat. Az ehhez szükséges jellemzők meghatározása nem objektív módon történik, amelynek hátrányát épület felújításokat célzó pályázatok esetében érzékelhetjük. Mivel a fenntartható fejlődés útja csak úgy követhető, ha minél kevesebb energia befektetésévével érünk el jobb eredményeket, elengedhetetlen a nagy fogyasztású tényezők gazdaságosabbá tétele.

A cél egy olyan Arduino alapú érzékelő rendszer létrehozása, ami képes a főbb komfort és energetikai szempontból meghatározó jellemzők mérésére, az adatok szoftveres feldolgozására és rendszerezésére. A mérni kívánt jellemzők a hőmérséklet, páratartalom, légnyomás és a megvilágítás erőssége. Az adatok továbbítása számítógépre vezeték nélküli kapcsolat útján valósul meg egy ESP8266 Wi-Fi modul segítségével, tárolásuk pedig egy SQLite adatbázisban történik. Innen később különféle Python programok segítségével a mentett adatok kiolvasásra, statisztikai feldolgozásra, megjelenítésre, és energetikai számításokban való felhasználásra kerülnek.

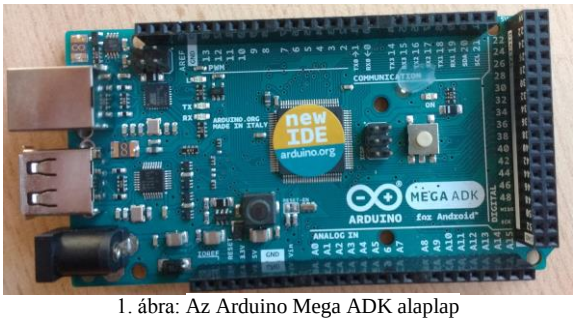
II. ELŐZMÉNYEK

A. Az Arduino-ról általánosan

Az Arduino egy nyílt forráskódú, szabad szoftveres elektronikai fejlesztőplatform. A platform két részből áll: egy integrált fejlesztői környezetből (IDE) és magából az Arduino boardból. Mivel a program nyílt forráskódú, így a fejlesztésébe bárki bekapcsolódhat az egész világon. A fejlesztői környezet bárki által letölthető az Arduino hivatalos oldaláról.

Egy Arduino készülékre szánt program bármilyen nyelven megírható egy olyan szerkesztőbe, amely bináris gépi kódot eredményez a céleszköz számára. Az Arduino fejlesztői környezet (IDE, Integrated Development Environment) egy Java nyelven megírt kereszt-platformos fejlesztői környezet. Egy Arduino fejlesztői környezetben megírt programot vázlatnak (sketch) hívnak. Ezek a fejlesztő számítógépen vannak tárolva szöveges fájlként .ino kiterjesztéssel. Az Arduino IDE a C és C++ nyelveket támogatja. Az így elkészült kódok két fő részből állnak, az első a *setup(felépítés)*: feladata a változók inicializálása, bemeneti és kimeneti pinek meghatározása, könyvtárak meghívása. Ez a rész a program futása során csak egyszer megy végbe. A második a *loop (hurok)*: a setup meghívása után ez a funkció irányítja az alaplapot. Ez a rész folyamatosan ismétlődik egészen addig, ameddig a rendszer bekapcsolt állapotban van. Az Arduino fejlesztői környezet előnyei közé tartozik az is, hogy a keretrendszer alapvetően tartalmaz már előre megírt könyvtárakat, amelyek funkcióit a programozás során fel tudjuk használni. [1]
A projekt megvalósításához Arduino Mega ADK board-ot használtunk, ami egy ATmega2560-on alapuló mikrokontroller alaplap. Tartalmaz külön USB portot, 54 digitális bemeneti/kimeneti pint, 16 analóg bemenetet, 4 UART-ot (hardver soros port), 16 MHz-es kristályoszcillátort, egyéb USB csatlakozást, tápbemenetet, ICSP (In-Circuit Serial Programming) headert és egy újraindító gombot.

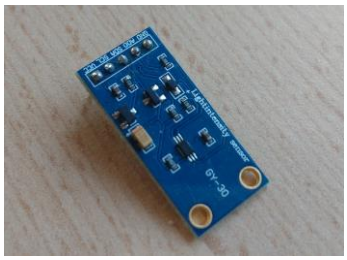
DOI: 10.17667/riim.2018.si/3.



1. ábra: Az Arduino Mega ADK alaplap

B. A GY-30 (BH1750FVI) megvilágításmérő szenzor

A BH1750FVI egy digitális megvilágításmérő szenzor IC I^2C busz interfészre fejlesztve.[2]

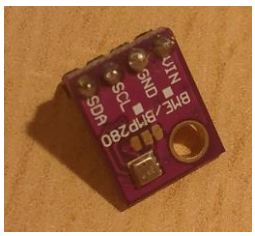


2. ábra: A GY-30 (BH1750FVI) precíziós megvilágításmérő szenzor

Az érzékelő eszköz nagy mérési tartománnyal és magas felbontással rendelkezik, így megfelelően pontos méréseket lehet vele végezni a fényintenzitás meghatározásához. Maga az érzékelő egység egy fotodióda, amely az emberi szemet megközelítő érzékenységgel rendelkezik. Nagy előnye, hogy kétféle I^2C slave cím is kiválasztható rajta, ezért szükség esetén egy buszra két eszköz is csatlakoztatható.

C. A GY-BME280 (nyomás-, pára- és hőmérsékletmérő) kombinált szenzor

A BME280 egy kombinált digitális nyomás- páratartalom- és hőmérsékletmérő szenzor, amely rendelkezik mind I^2C, mind SPI interfésszel.[3]

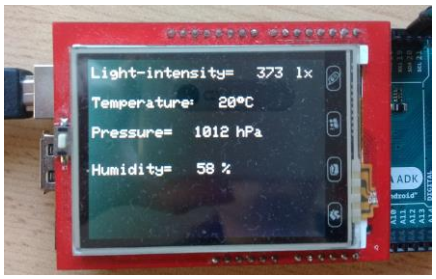


3. ábra: A GY-BME280 kombinált szenzor

Mivel rendelkezik I^2C interfésszel, így az adatok átvitelét a fényérzékelővel együtt erre a buszra sorosan kötve valósítottam meg.

*D. 2,4" TFT LCD (320*240)/SD (SPFD5408) érintőképernyős kijelző*

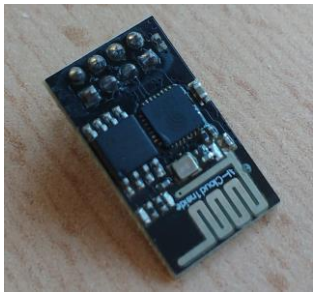
Az eszköz 2,4”-os érintőképernyő funkcióval ellátott kijelzővel rendelkezik, ami rezisztív elven működik. Beépített jelszintillesztőn át közvetlenül csatlakoztatható akár a 3,3 V-os, akár az 5 V-os feszültségű Arduino lapkára is. [4]



4. ábra: Az SPFD5408 érintőképernyős kijelző

E. Az ESP8266 Wi-Fi modul

A modul viszonylag olcsó, nagy hatótávolságú, és rendelkezik mikrokontroller kompatibilitással, azaz csatlakoztatható az Arduino alaplaphoz is.



5. ábra: Az ESP8266 01 Wi-Fi modul

A modul a TCP/IP kommunikációs formát használja. Az TCP/IP egy kommunikációs protokoll, amely az interneten és hasonló számítógépek hálózatokon használatos. Nevét a két legfontosabb protokolljáról, a TCP-ről (Transmission Control Protocol) és az IP-ről (Internet Protocol) kapta. Működése egy négyrétegű elvi modellen alapul: alkalmazási réteg, szállítási réteg, internet réteg és hálózati interfész réteg.

III. A MŰSZER FELÉPÍTÉSE

A mérőrendszer egy érzékelőkből, kijelzőből és Wi-Fi modulból álló áramkör, amelyet az Arduino eszköz lát el megtáplálással. A különböző portokhoz a tesztelési fázis folyamán a próbapanelen keresztül csatlakoztathatók a különböző eszközök. Bekötésükhöz ellenállásokra, a 3,3 V-os tápfeszültség-igény miatt feszültségszabályozóra, és a zajszűrés miatt kondenzátorokra van szükség.

A. Az eszközök bekötése

Az érintőképernyős kijelző egy Arduino shield, amely közvetlenül a tüskesorain keresztül hozzácsatlakoztatható az alaplaphoz. Mivel az Arduinon 2 db 5 V-os tápfeszültség- és 3 db földcsatlakozás található, így nem jelent gondot, hogy a kijelző közvetlenül csatlakozik ezen kimenetekhez. A kijelző adatkimenetei az Arduino digitális I/O portjaihoz, a vezérlő kimenetek az Arduino analóg bemeneteihez csatlakoznak. Az analóg bemenetek digitális I/O portokként is használhatók.

Vezérlő kimenetek a kijelzőn: LCD_RST (reset), LCD_CS (chip select), LCD_RS (register select), LCD_WR (write), LCD_RD (read). Ezeken keresztül lehet utasításokat küldeni arról, hogy milyen feladatot végezzen el a kijelző.

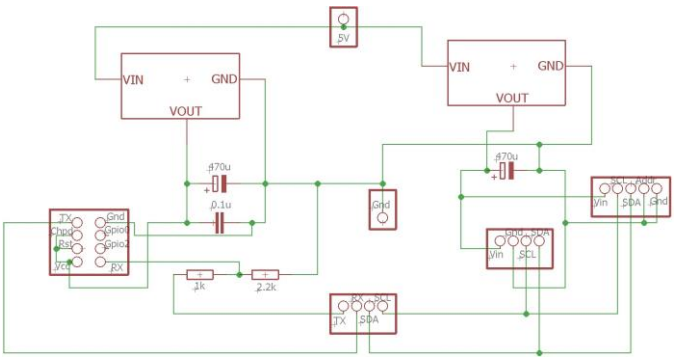
Mindkét érzékelő (a kombinált nyomás-, pára-, és hőmérsékletmérő és a fényerősségmérő) közös I^2C buszon keresztül csatlakozik az Arduinohoz. Ezek a kijelző által nem lefoglalt 5 V-os tápfeszültségre és földre csatlakoznak a 3,3 V-os feszültségszabályozón keresztül. Az SDA és SCL lábaik az Arduino SDA és SCL portjaihoz csatlakoznak. Az adatok küldése úgy valósulhat csak meg ezzel a megoldással, hogy egymástól időben eltérően küldik azokat az alaplapra. Ezekről az eszközökről a mért adatok soros porton keresztül AT parancsok segítségével először a Wi-Fi modulra, majd onnan a számítógépre juttathatók át további feldolgozás és kiértékelés céljából.

A Wi-Fi modul az érzékelőkhöz hasonlóan 3,3 V-os feszültségszabályozón keresztül kötjük a megtáplálásra. Az Arduino kommunikációs portjai a TX (transceive - küldés) és RX (receive - fogadás) portok. Ezeket feszültségosztón keresztül a Wi-Fi RX és TX lábaira kötjük, mivel ezeknek is 3,3 V-os feszültségszinte van szükségük a megfelelő működéshez. Az RX portot mindig a TX lábra, a TX portot pedig mindig az RX lábra kell kötni, mivel a küldés-fogadás kommunikációs folyamata csak ilyen módon valósulhat meg.

B. A próbapanel problémája és a megoldás

A próbapanel alkalmazása azonban a gyakorlatban nem célszerű: rengeteg feleslegesen hosszú kábel használata, melyek könnyen elmozdulnak, kihúzódnak a helyükről, a próbapanel mérete pedig meglehetősen nagy. Ezen problémák megoldására az Eagle nevű tervezőprogram ingyenes verziójában elkészítettük a kifejezetten erre az áramkörre tervezett nyomtatott áramköri lemezt.

Elsőnek a kapcsolási rajz elkészítésére volt szükség, melyhez meg kellett állapítanunk a felhasználandó alkatrészek datasheetjéből, hogy azok milyen lábkiosztással, tokozással, egyéb méretekkel rendelkeznek. Az érzékelők, a Wi-Fi modul, valamint az Arduino lemezhez történő csatlakoztatását tüskesorok beforrasztásával kívánjuk megoldani, melyek hossza a csatlakoztatni kívánt lábak illetve pinek számának megfelelő.

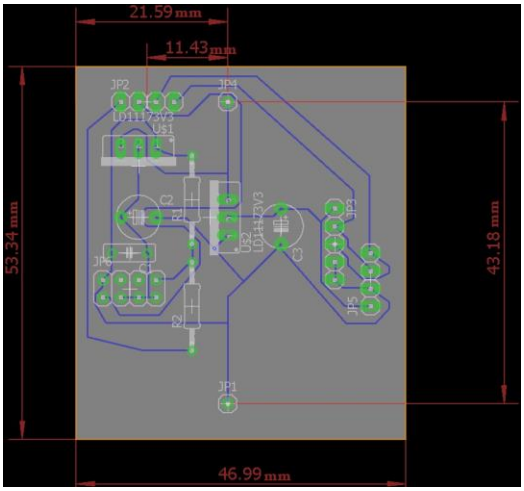


6. ábra A kapcsolási rajz

A kapcsolási rajzon szereplő alkatrészek összeköttetéseit elkészítve következő lépés volt a tényleges NyÁK lemez megtervezése. Itt váltak aktuálissá a fent említett Arduino méretek, melyek alapján az alkatrészek pontos helyükre kerültek a lemezen. A tervezés során a következőket kellett mindenféleképpen szem előtt tartanunk:

- a vezetékek közötti szigetelő rész szélessége (6 milre állítottuk)
- a vezetékek szélessége (10 milre állítottuk)

A tervezés kritériumainak adatait hivatalos NyÁK lemez gyártásával foglalkozó gyártók információi alapján választottam ki aszerint, hogy a megadott minimum értékektől nagyobb értékeket állítsunk be.



7. ábra A NyÁK lemez terv

IV. AZ ARDUINO PROGRAM MŰKÖDÉSE

A. A setup rész

Az Arduino program setup részében történik a mérőműszer kezdeti beállításainak elvégzése. Itt történik az Arduino-hoz csatlakoztatott eszközök inicializálása.

A szenzorok működési beállításai is ebben a részben határozhatók meg. Különféle mérési módok elérhetőek az adott érzékelőkhöz, ezeket a beállító parancsokat az I^2C buszon keresztül juttatja el a program hozzájuk.

A fényérzékelő esetében a *setup* részben a megfelelő műveleti kód *(opcode)* elküldésével kapcsolható be az eszköz, illetve beállítható, hogy milyen módban és milyen felbontással folytassa a mérést.

A kombinált szenzor mérési periódusa hőmérséklet, nyomás, és páratartalommérésből áll a *setup* részben állítható mintavételezési számmal. Egy mérési periódus lezajlása után a nyomás és hőmérséklet adatok átvezethetők egy opcionális IIR szűrőn keresztül, ami kiszűri a rövidtávú ingadozásokat, ennek beállításai szintén az inicializálás során történnek meg.

A Wi-Fi modul inicializálása során megtörténik az IP cím beállítása, a modul kliensként vagy szerverként való működése, valamint a TCP/IP kommunikáció illetve az UART sebességének meghatározása.

B. A mérés folyamata

A mérési folyamat a *loop* végtelen ciklusában fut addig, ameddig a műszer feszültség alatt van. Az aktuális értékeket az LCD kijelző azonnal megjeleníti. A mért adatokat az érzékelők az I^2C buszrendszeren keresztül küldik át az Arduino alaplapnak, melynek működését a következőkben részletezzük:

Az I^2C busz működése:

Az I^2C (Inter-Integrated Circuit) egy kétvezetékes szinkron adatátviteli rendszer integrált áramkörök összekapcsolására, amelyet a Philips Semiconductor cég fejlesztett ki. A két vezeték a következő: SDA (adatvezeték) és SCL (órajel vezeték), mindkettő 7-bites címzéssel rendelkezik.

Az I^2C buszrendszerben kétféle szerepkörű eszköz fordulhat elő. A master generálja az órajelet és kommunikációt kezdeményez a slave-ekkel, a slave pedig fogadja az órajelet és reagál, amikor a master megcímzi.

Amikor a master átvitel módban van, a folyamatot egy start bit küldésével kezdi, amelyet annak a slave-nek a 7-bites címe követ, amellyel a master kommunikációt szeretne kezdeményezni. Ezután következik egyetlen egy bit, amely megadja, hogy a master írni (0) szeretne a slave-be, vagy adatot kiolvasni (1) belőle.

Amennyiben a slave létezik a buszrendszeren, egy ACK bittel fog válaszolni (alacsony szintre teszi az SDA vonalat annak jeléül, hogy tudomásul vette a kommunikációs kezdeményezést). A master ekkor vagy átvitel, vagy fogadás módba vált annak függvényében, hogy az írás vagy az olvasás bit lett kiküldve, a slave pedig ennek komplementereként fog működni (master ír – slave olvas, master olvas – slave ír).

A címzés és az adat bitek elsőként a legnagyobb helyiértékkel rendelkező (MSB) bitet küldik el. A start bitet az SDA vezeték magas szintről alacsony szintre történő átváltása és az SCL vezeték magas szinten léte, a stop bitet az SDA vezeték alacsony szintről magas szintre váltása és az SCL vezeték magas szinten léte jelzi. Az SDA további szintváltásai az SCL alacsony szintjén történnek.

Amennyiben a master írni szeretne, ismétlődően byte-okat küld a slave-nek, amely minden egyes alkalommal egy ACK bittel válaszol. (Ebben az esetben a master átvitel módban, a slave fogadás módban van.) Ha a master olvasni szeretne, akkor ismétlődően byte-okat fogad a slave-től, amelyek után minden egyes alkalommal (kivétel az utolsó byte) egy ACK bitet küld. (Ez esetben a master fogadás módban, a slave átvitel módban van.) A master az átvitelt egy stop bit küldésével szakítja meg.[5]

A fényérzékelő folytonos mérésű magas felbontású módban üzemel. Első lépésként az Arduino elküldi a működési módra vonatkozó utasítást. Ezt követően megtörténik a mérés, melyet az érzékelő 2 byte hosszan küldi vissza a masternek, majd a mért érték kiolvasása következik. [2] Az eredmény kiszámítása abban az esetben, mikor a High Byte (nagyobb helyiértékű byte) "10000011", a Low Byte (kisebb helyiértékű byte) pedig "10010000":

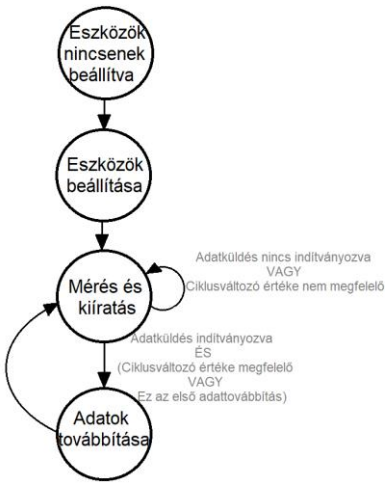
$$(2^{15} + 2^9 + 2^8 + 2^7 + 2^4) / 1{,}2 = 28067 \text{ lx}$$

A BME280 mérési ciklusának megkezdése után az első lépésként a hőmérsékletet, majd a nyomást, végül pedig a páratartalmat méri az eszköz. Ezt követően az eszköz megvizsgálja, hogy az IRR szűrő használata engedélyezve van-e, és amennyiben engedélyezve van, inicializálva van-e. Amennyiben a válasz bármelyikre nemleges, úgy a rendszer inicializálja a szűrőt az ADC (analóg-digitális átalakító) értékeivel. Amennyiben a válasz igen, úgy a rendszer frissíti a szűrő memóriáját az ADC értékek és a szűrő együtthatója segítségével. Ezen folyamatok bármelyikének befejezésével a szűrő memóriája a kimeneti regiszterekre másolódik, a mérési folyamat pedig befejeződik. [3]

C. Az adatok számítógépre való továbbítása

A TCP/IP kommunikációs formát Python programozási nyelv segítségével alkalmaztuk. A Python kétféle módon tud csatlakozni a hálózati rendszerekhez: egy alacsonyabb szintű, amely socketeket használ, és egy magasabb szintű (FTP, http, stb.). A socketek kétirányú kommunikációs csatornák végpontjai, melyek képesek kommunikálni azonos folyamatok között, egy eszközön belül lévő különböző folyamatok között, illetve különböző eszközökben lévő folyamatok között is. A socket kapcsolatban egy szerver és egy vagy több kliens található. Esetünkben egy kliens (számítógép) és egy szerver (a Wi-Fi modulon keresztül az Arduino) van. A klienssel tudunk rácsatlakozni a szerver eszközre, így adatcsere lehetséges köztük. Jelen helyzetben ezek az érzékelők által mért adatok, amelyeket a szerver közvetít a kliens felé. E kapcsolat létrehozásához szükség van a host és a port megadására is. A portra maximálisan 5 kliens csatlakoztatható. Meg kell adni a várható adatcsomag hosszát, valamint lehetséges timeout beállítása is. [6]

Az adatok számítógépre való továbbítása szakaszos ütemű, mivel az épületek jellemzőinek értékei nem változnak túl gyors ütemben. Ezért egy ciklusváltozó bevezetését tartottuk célszerű megoldásnak. Ez minden mérési ciklus végén növeli az értékét, és amikor elér egy bizonyos értéket, akkor indul el az adatküldés folyamata. Ez kb. 3 és fél percenként ismétlődik, így folyamatos mérés mellett viszonylag sok mérési eredményt tudunk továbbítani a Wi-Fi modul segítségével a számítógépre.



9. ábra: A mérés és adattovábbítás állapotgépe

V. A PYTHON PROGRAMOK

A Python egy portábilis, dinamikus, bővíthető, ingyenes nyelv, ami lehetővé teszi a programozas moduláris és objektum orientált megközelítését. A nyelv tervezési filozófiája a futási sebességgel szemben az olvashatóságot és a programozói munka megkönnyítését helyezi előtérbe. Általános célú, azaz széleskörűen alkalmazható szoftverek írására, és magas szintű, azaz jellemzője az absztrakció, elvonatkoztatottság, a könnyebb használhatóság.

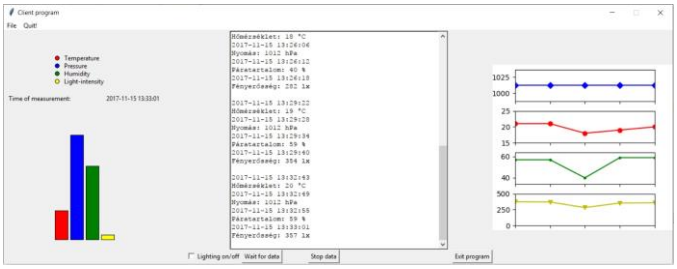
Minden egyes Python programhoz grafikus intefészt (ún. GUI-t) hoztunk létre a felhasználók számára a beépített Tkinter modul segítségével. Ez egy egyszerűen programozható felhasználói interfészt nyújt a Python programokhoz. A modul támogatja a programok igényeit kielégítő Tk widgeteket (eszköztárakat). A Tkinter a Tk (Tcl/Tk GUI eszközkészlet) Python interfésze. [7]

A. *A fogadó program*

A begyűjtött és a számítógépre továbbított adatokat egy SQLite adatbázisban tároljuk. Az SQLite egy olyan könyvtár, amely egy szerver nélküli, zéró konfigurációs SQL adatbázis motort implementál. Az SQLite nyílt forráskódú, így bárki által ingyenesen felhasználható bármilyen célból. A *„Python Standard Library”* nevű Python könyvtár tartalmaz egy *„sqlite3”* modult, amely arra lett kifejlesztve, hogy ezzel az adatbázissal létesítsen kapcsolatot és dolgozzon benne.

A Wi-Fi modul által továbbított adatokat ez a program fogadja és tárolja a számítógépen. Ez lehetséges egy már meglévő adatbázisban, vagy a felhasználó által létrehozott új adatbázisban is. A program gombnyomásra indítja az adatok fogadását, illetve az adatfogadás leállítását is.

A mért értékek háromféle módon vannak megjelenítve a grafikus felület által. Bal oldalon egy oszlopdiagram található, amelynek méretei az aktuális mért értékek és a maximálisan előforduló értékek arányában változnak. A középső részen szöveges formátumban kerülnek megjelenítésre az adatok. Jobb oldalon pedig négy különböző diagramon láthatóak az értékek.

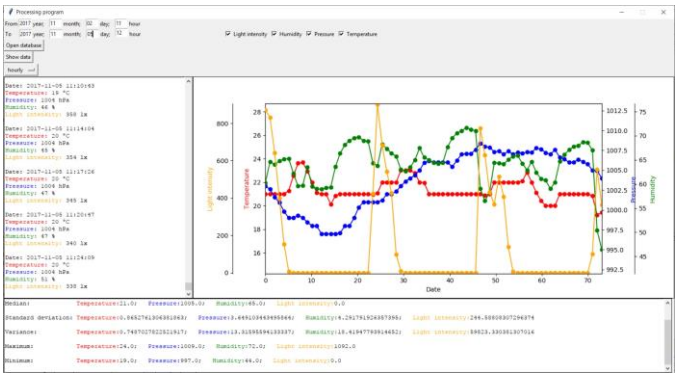


10. ábra A fogadó program grafikus interfésze

B. *A feldolgozó program*

A feldolgozó program a begyűjtött, már adatbázisban tárolt adatokat olvassa ki és dolgozza fel különféle statisztikai módszerekkel. A felhasználó által kiválasztott adatbázisból, és az általa megadott időintervallumok közötti adatok kerülnek kiértékelésre a program által.

A grafikus interfész megjeleníti a kiolvasott adatokat szöveges formátumban is, típustól függően szín szerint is szétválogatva. A különféle mérési eredményeket statisztikai módszerekkel is feldolgozza, ezáltal átlagot, móduszt, mediánt, szórást, szórásnégyzetet, minimum és maximum értéket számolva. Grafikonon is megjelenítésre kerülnek a mért értékek, itt a felhasználó állítja be, hogy milyen időközönként átlagolva (óránkénti, napi, heti, éves) szeretné látni az adatokat, valamint hogy a mért paraméterek közül melyek legyenek rajta a grafikonon. 0-4 paraméter megjelenítése lehetséges tetszőleges kombinációban.



11. ábra A feldolgozó program grafikus interfésze

C. *A kiértékelő program*

Az adatbázisba elmentett adatokat épületenergetikai számításokat végző programmal is feldolgoztunk. Ezen számítások célja az, hogy meghatározzuk vele egy adott épület hőveszteségét [kWh]-ban. Mivel ez csak akkor lehetséges, ha ismerjük az épületet körülvevő külső és fűtetlen terek paramétereit is, mi azonban csak egy mérőeszközzel rendelkezünk, így ezeket a hiányzó adatokat hivatalos mérések alapján manuálisan visszük fel az adatbázisban létrehozott új táblákba.

A kiértékelő program a három Python nyelven írt program közül a leghosszabb, mivel rengeteg adat felhasználótól való bekérése szükséges ahhoz, hogy a számítások pontosan elvégezhetőek legyenek. Összesen 10 darab keretet jelenítünk meg a főablakban, viszont az érdekesség az, hogy nem egyszerre, hanem egymás után. Mivel ennyi adat bekérése egy kereten belül meglehetősen sok lenne, és sokkal összeszedettebb, ha az egy típusba tartozó adatokat egy körön belül olvastatja be a program, ezenfelül a működés szempontból szükséges is, hogy az adatok már a bekérés során feldolgozhatóak legyenek, ezért érdemes ez a módszer alkalmazása.

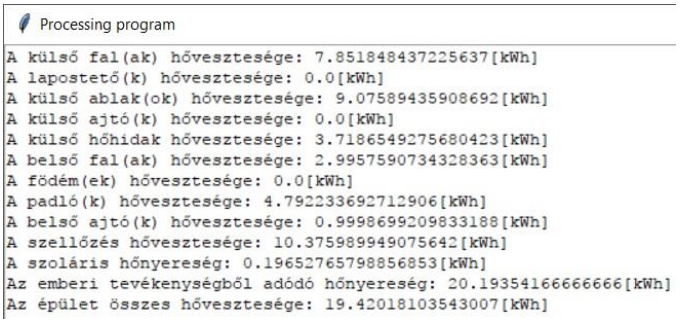
Az első 9 keretben a program elvégzi a felhasználótól való adatok bekérését. Az egyes keretek között a *Tovább* gombbal lehet lépni, amely lenyomásával a program elvégzi a szükséges számításokat, törli az előző keretet, és megjeleníti a következő keretet az összes hozzátartozó résszel. Az utolsó keretben az elvégzett energetikai számítások eredményei olvashatók le.

VI. A PRÓBAMÉRÉS VÉGZÉSE

A tényleges mérést abban egy debreceni társasházi lakásban végeztük el. Azért itt valósítottuk meg a több napos kísérletet, mivel ismerjük a lakás határoló szerkezeteinek tényezőit, valamint állandó felügyelet mellett tudtuk lefolytatni a mérést.

A vizsgálatot 2017.11.02. 11 órától 2017.11.05. 11 óráig végeztük, így összesen 3 napi eredmény áll rendelkezésünkre a feldolgozáshoz. A mérés ideje alatt pontosan fel lett jegyezve a külső nyílászárók kinyitásának, illetve becsukásának időpontja, a lakók otthon tartózkodása, valamint a gázóraállás is, ami arra szolgál, hogy a kiszámított eredményt lehessen mihez viszonyítani. A fűtési időszak előtt megfigyeltük a melegvíz-készítés gázfogyasztásának mértékét, hogy ezt levonhassuk az újonnan leolvasott eredményekből.A számolást a kiértékelő program végzi a bele írt képletek felhasználásával.

A mérés kezdetekor a gázóraállás 2051,582$[m^3]$, befejezéskor pedig 2056,772 $[m^3]$ volt. A két érték különbségét beosztva hárommal megkaptuk a napi átlagos fogyasztást, ami 1,73 $[m^3]$. Ebből levonva az előzetesen megállapított melegvíz fogyasztást (0,99 $[m^3]$) a napi szinten fűtésre elhasznált gáz mennyisége 0,74 $[m^3]$, ami a kazán 90 %-os hatásfoka miatt ténylegesen csupán 0,67 $[m^3]$-nek számít, ez a három napra vonatkozóan nagyjából 2$[m^3]$-nek felel meg. Amennyiben a földgáz fűtőértékét 35700 $\left[\frac{kJ}{m^3}\right]$-nek veszem, úgy az összes fogyasztás 71400 $[W]$ = 19,83 $[kWh]$.



12. ábra A próbamérés során mért értékekkel végzett számítások eredménye

VII. ÖSSZEFOGLALÁS

A méréssel megállapított és a megfigyelésen alapuló eredmény egymáshoz való közelisége meglehetősen feltűnő. Mivel az eszköz csupán próbamérésen esett át mindenféle utólagos kísérletezés, beállítás, finomítás és tapasztalatok nélkül, így valószínűnek tartjuk azt, hogy ekkora pontosságot a gázóra leolvasási hibája, vagy a nem mért adatok manuálisan történő adatbázisba vitele (mivel ezek csupán közelítő értékek) okozhatja. Jól látható azonban, hogy a két érték ezen hibák megléte nélkül is nagyjából azonos lenne, így kijelenthető, hogy a mérőeszköz, valamint a hozzá írt számítógépes programok a kitűzött célok szerint működnek. A kitűzött célt sikerült elérnünk, azaz a mérőeszköz segítségével képesek voltunk meghatározni a lakás gázfogyasztását.

A műszer és a programok kisebb átalakításokkal alkalmazhatóak lennének arra, hogy épületenergetikai vizsgálatok végzését és tanúsítások kiadását megkönnyítsék. Egy másik felhasználási mód az okos épületekben valósulhatna meg. Mivel a műszer viszonylag sok paraméter mérésére képes, így ezeket lehetne különféle beavatkozókkal szabályozni. Az Arduino által küldött adatok ugyanúgy valamilyen számítógépre (akár egy egyedülálló, vagy clusterekbe állított Raspberry Pikre) lennének továbbítva, amely feldolgozná a kapott információt. Lehetőség lenne a rendszerbe manuálisan (pl. okostelefonról, valamilyen alkalmazás segítségével), illetve automatikus módon beavatkozni. Érdemes lenne az adatbázisba mentett adatokat a rendszer különböző tanulási metódusaihoz felhasználni, ami ezáltal a statisztikák kiértékelése alapján képes lenne "megjósolni" az épület várható állapotát, és ennek megfelelően üzemeltetni a beavatkozókat.

További fejlesztési módszer lehet az Arduino helyettesítése egy egyszerű mikroprocesszorral. Bár a processzor felprogramozásával, valamint a hozzá tartozó áramkör megtervezésével jóval több munka lenne, mint a készen kapható mikrokontroller esetében, ám kompenzálna érte az alacsonyabb ár és a kisebb méret.

KÖSZÖNETNYILVÁNÍTÁS

A publikáció elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió támogatásá-

val, az Európai Szociális Alap társfinanszírozásával valósult meg.

HIVATKOZÁSOK

[1] R. Cseh, Arduino Programozási kézikönyv, Budapest, 2011.

[2] "Digital 16bit Serial Output Type Ambient Light Sensor IC" [Online]. Available: http://datasheet.octopart.com/BH1750FVI-TR-Rohm-datasheet-25365051.pdf. [Hozzáférés dátuma: 2017.10.15]

[3] „BME280 Combined humidity and pressure sensor", [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST-BME280_DS001-10.pdf [Hozzáférés dátuma: 15. 10. 2017.].

[4] ILI TECHNOLOGY CORP., "a-Si TFT LCD Single Chip Driver 240RGBx320 Resolution and 262K color Datasheet"

[5] Philips Semiconductors, „Az I²C-busz és használata," 1995.

[6] I. F. Akyildiz – Mehmet Can Vuran, Wireless Sensor Networks, Wiley, 2010.

[7] J. E. Grayson, Python and Tkinter Programming, Greenwich, 2000.