Recent Innovations in Mechatronics (RIiM) Vol. 4. (2017). No. SI.
DOI: 10.17667/riim.2017.si/3.

Épületautomatizálási rendszer kiépítése Arduinoval és ESP8266 modullal

Nagy Zoltán Simon
Villamosmérnöki és Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
nagyzoltansimon@gmail.com

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

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

***Absztrakt* – Ezen kutatás keretein belül egy épületautomatizálási rendszer készült el, mely alapját Arduino Pro Mini mikrokontrollerek és ESP-01 Wi-Fi modulok adják. Az elkészült rendszer reluxát és hangulatvilágítást vezérel illetve mérési adatokat gyűjt.**

Kulcsszavak –Arduino, Pro Mini, ESP8266, ESP-01, reluxavezérlés, világítás.

I. BEVEZETŐ

A projekt készítése során egy olyan épületautomatizálási rendszer alapját szerettem volna kiépíteni, mely később az adott eszközökkel és adott logika mentén könnyen bővíthető. A kezdeti rendszerrel elsősorban a saját házunk nyári túlmelegedési problémájának megoldását tűztem ki célul egy automatikus reluxavezérlési rendszer kifejlesztésével.

A rendszer alapját Arduino panelek adják, melyek ESP8266-os Wi-Fi modulokkal kommunikálnak egymással, és ezáltal egy olyan rendszer építhető ki, mely más épületautomatizálási feladatok ellátására is alkalmas. Az adatok egy központba futnak be, mely azután egy EEPROM-ba menti le azokat. A központ feladata ezen kívül, hogy kapcsolatot tartson a felhasználóval, illetve a többi eszköznek adatokat szolgáltasson.

II. ELŐZMÉNYEK

A. Az Arduino platform bemutatása

Az Arduino története a kétezres évek elejére nyúlik vissza. A platform létrehozásában Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, és David Mellis vett rész az Ivrea városi Interaction Design intézetében. A munkájuk alapját Wiring nyílt forráskódú platform adta, amelyet Hernando Barragan készített el diplomamunkájaként Kolumbiában 2003-ban. Az Arduino platform végül 2005-ben látta meg a napvilágot. A platform két részből áll, egy integrált fejlesztői környezetből (IDE) és magából az Arduino board-ból. A fejlesztői környezet szabadon letölthető az Arduino hivatalos oldaláról. [1]

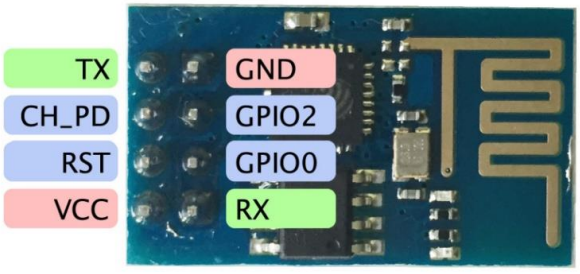
Az Arduinot egy kissé átalakított C nyelvben lehet programozni. A programok két fő részből épülnek fel, ezek a *void setup()* illetve a *void loop()* funkcióhívások. Az első rész a *void setup()* funkcióhívás. Ez a rész a program futása során egyszer megy végbe. Itt lehet megadni a kivezetések inicializálását, a soros vagy I^2C kommunikációs csatornák beállításait vagy olyan műveleteket, amelyeket csak egyszer akarunk lefuttatni. A második rész a *void loop()*, mely folyamatosan ismétlődik, míg a rendszer bekapcsolt állapotban van. Ezen két funkcióhíváson kívüli részeken lehet deklarálni és definiálni változókat, illetve meghívni függvénykönyvtárakat. Az Arduino egyik nagy előnye, hogy a keretrendszer eredetileg is tartalmaz már előre megírt könyvtárakat, amelyet a *#include<könyvtárnév.h>* utasítással tudunk meghívni. [2]

A fejlesztői környezetben található még számos példafeladat, illetve soros monitor, mellyel USB-n keresztül tudunk az Arduino board-nak üzenetet küldeni.

A projekt során én Arduino Pro Mini board-okat használtam. Ezek hasonlóan az Arduino Uno-hoz ATmega328 mikrovezérlővel működnek. Nagy különbség a kettő között, hogy a Pro Mini-t jóval kisebbre tervezték, mint az Uno-t, így például nem tartalmaz USB vagy hálózati csatlakozót. [1]

B. Az ESP8266-os chip bemutatása

Az ESP8266 egy mikrochip, melyet az Espressif Systems nevű sanghaji cég fejlesztett ki. A cég alacsony energiaigényű Wi-Fi illetve Bluetooth modulokat fejleszt, kifejezetten IoT fejlesztésekhez. [3]



1. ábra: ESP-01-es modul [8]

Bár az ESP modulok el vannak látva memóriaegységgel, melyre programot lehet feltölteni, és így mikrokontrollerként is működhet, az általam használt ESP-01-es modul csupán két szabadon programozható I/O porttal rendelkezik, így nem alkalmas összetettebb vezérlési feladatokra.

A modulra Lua nyelven lehet programkódot írni, de használható az Arduino 1.6.4 vagy későbbi kiadású keretrendszere is a felprogramozásra. De a chipet vezérelhetjük AT-s parancsokkal is. [4]

C. Hasonló projekt bemutatása

Az *International Journal of Engineering and Computer Science* című folyóirat 2017. márciusi számában mutatták be épületautomatizálásra alkalmas rendszerüket a *Gnanamani College of Technology* egyetem tanárai. Az általuk kidolgozott rendszer egy Arduino Uno-ra és NodeMCU panelre épül. Az utóbbi panel egy ESP-12 (vagy ESP-12E) modulra épülő mikrovezérlő. [5]

A szerzők által épített rendszer tartalmaz bemeneti egységként két infravörös (PIR) mozgásérzékelő modult és egy LM35-ös hőmérőt. A kimenetként pedig relé modulokat tartalmaz, melyekkel a világítást, tápkábeleket vagy HVAC rendszert lehet kapcsolni. A rendszer automatikusan szabályozta a kimeneteket az érzékelők adatai alapján. Például, ha a belső hőmérséklet meghalad egy kritikus szintet, a vezérlő áramkör meghúzza annak az aljzatnak a reléjét, amelyről a ventilátor van táplálva, így az bekapcsol. A rendszert azonban a felhasználó által is vezérelhetővé tették. A vezérléshez egy egyéni alkalmazást készítettek. Az alkalmazás képes távoli kapcsolatot létesíteni az eszközökkel, vezérelni és monitorozni azokat, vagy ütemezni a feladatokat. A programban van IP és felhasználói hitelesítés funkció, valamint hangvezérléssel is kapcsolható. [5]

Természetesen számtalan fejlesztés létezik ebben a témában az abszolút hobbi projektektől egészen a professzionális vagy ipari fejlesztésekig.

III. AZ ÉPÜLET ÉS A RENDSZER BEMUTATÁSA

Az épület elhelyezkedésének bemutatásához a SektchUp [6] nevű programban készítettem el a ház 3 dimenziós modelljét. A program lehetővé teszi, hogy az elkészült modellt a felhasználó elhelyezze a Google Maps-en, így a valós épülethez lehet viszonyítani a modell tájolását. A modell a 2. ábrán látható.

2. ábra: Az épület modellje

Az ábrán az épület keleti illetve déli oldala látható. A keleti oldalon 5 olyan ablak található, melyeken a nap már kora reggeltől kezdve besüt, ezzel nyáron már reggel felmelegítve a lakást.

A rendszer elsődleges feladata az, hogy megoldja ezt a problémát azzal, hogy az ablakokba szerelt reluxákat vezéreli úgy, hogy nyáron reggelente a direkt napsugárzást nem engedi be, de a nap többi szakában az indirekt sugárzást igen. Természetesen a rendszernek úgy kell vezérelnie a reluxákat, hogy azok télen a direkt sugárzást is beengedjék.

A rendszer a következőképpen épül fel: A központi egység fő feladata az adatgyűjtés és az alegységek összekapcsolása, illetve a felhasználó ezen keresztül tudja felügyelni illetve vezérelni a rendszert. Az alegységek ellátnak vezérlési feladatokat, illetve mérési adatokat továbbítanak a központ felé.

IV. A KÖZPONTI EGYSÉG BEMUTATÁSA

A központi egységben az Arduino és ESP8266-os modulokon kívül a következő eszközök találhatóak: RTC modul, EEPROM, hall-elemes áramerősség-mérő modul, termisztor, LED-sor.

A. A központ és az alegységek közötti kommunikáció

A központi ESP8266-os *acces point* (elérési pont) módban létrehoz egy hálózatot és rajta egy szervert. A többi alegység erre a hálózatra kapcsolódik, és ők is létrehoznak egy-egy szervert. Hogy az IP-címek átfedése ne okozzon problémát, mindegyik szervert más-más port-on hozzák létre. Mindig a központi egység kezdeményezi a kapcsolatot kliensként és az alegység zárja le az üzenetváltások végén. Az ESP8266-os chipek többszörös kapcsolatra is képesek, összesen négy csatornán keresztül tudnak kapcsolódni. Az üzenetváltások végén azért szükséges a lezárás, mert anélkül csupán négy kis egységgel tudna a központ kapcsolatot tartani. Továbbá gyakoriak a hibák a kommunikációban, ha túl sokáig van nyitva tartva ugyanaz a csatorna.

Az üzenetváltások során felhasznált fő utasítások a következőek:

- **AT+CWMODE=mód:** Az utasítással adjuk meg, hogy az ESP milyen módban működjön. (1: Station, 2: Acces point, 3: Sation + acces point)
- **AT+CWJAP=”Hálózat_neve”,”Jelszó”:** Az utasítással adjuk meg, hogy a modul melyik hálózathoz kapcsolódjon.
- **AT+CIPMUX=mód:** Az utasítással lehet beállítani, hogy egy vagy több csatornán keresztül lehessen kommunikálni. (0: egyszeres kapcsolat, 1: többszörös kapcsolat).
- **AT+CIPSERVER=mód, port:** Elindítja vagy törli a szervert az adott port-on. (0: Törlés, 1: Indítás)

- **AT+CIPSTART=id,"mód","cím",port:** Létrehoz egy kapcsolatot az adott című szerverrel. Többszörös kapcsolat esetében mindig meg kell adni a csatorna ID-számát (1-4). A mód lehet TCP vagy UDP.

- **AT+CIPSEND=id, üzenet_hossza:** Elküldi az üzenetet az adott csatornán. Az üzenet hosszát az elküldendő karakterek adják plusz két byte a \r\n.

- **AT+CIPCLOSE=id:** Az ESP bontja a kapcsolatot a megadott csatornán. (id=5 esetén az összes csatornát zárja).

Az Arduino és az ESP-01-es modul soros interfészen keresztül kommunikál egymással. Így nincs szükség külön programkönyvtárra vagy az ESP8266-os chip felprogramozására. Az Arduino egyszerűen kiküldi az AT utasításokat az UART portjain keresztül. Tekintettel kell lenni azonban arra, hogy az ESP8266-os chip 3,3 V feszültségszinten működik, az 5 V károsítaná. Ezért a táplálása csak 3,3 V-ról működhet. Az általam épített rendszerben a 3,3 V-ot egy LD1117V33V sorozatszámú feszültségszabályzó integrált áramkör állítja elő a modul számára. Mivel a projektben használt Arduino Pro Mini 5 V-os jelszinttel dolgozik, ezért az Arduino Tx (kimenet) és az ESP Rx (bemenet) portja közé egy feszültségosztó kapcsolást tettem, hogy az ESP-01-es modult ne károsítsa az 5 V. Visszafelé nem okoz problémát a jelszint különbség (ESP Tx →Arduino Rx), mivel az Arduino a 3,3 V-ot még logikai 1 szintnek veszi.

B. A központ és a felhasználó közötti kommunikáció

A felhasználói kommunikáció megoldásának alapját egy fejlesztői oldalon [7] található cikk adta. Az irányítás a következőképpen zajlik.:

Egy HTML-kódot a felhasználó a saját készülékére (telefon vagy számítógép) menti. A HTML-fájl mellé le kell tölteni a jQuery függvénykönyvtárat is a JavaScript működéséhez, majd a készülékkel rácsatlakozni az ESP-modul által létrehozott Wi-Fi hálózatra (vagy arra a hálózatra, amelyre az ESP modul is csatlakozik). Bármely gombra kattintva a program rácsatlakozik az ESP által létrehozott szerverre úgy, hogy az elérési útvonalba az adott gomb *id* számát írja.

```
OK
0,CONNECT

+IPD,0,316:GET /?pin=11 HTTP/1.1
Host: 192.168.4.1
Connection: keep-alive
Accept: */*
Origin: null
User-Agent: Mozilla/5.0 (Windows NT 1(
Accept-Encoding: gzip, deflate, sdch
Accept-Language: hu-HU,hu;q=0.8,en-US;

0,CLOSED
```

3. ábra: Üzenetfogadás a telefontól

Az ESP-modul a *3. ábrán* látható adatokat küldi ki az Arduino felé. Látható, ahogy a soros monitoron kiírta az ESP a megnyomott gomb számát. Az Arduinoval a *Serial.find(„pin=")* utasítással a beolvassuk az idézőjelben lévő szövegig a kapott adatokat, majd a *Serial.parseInt()* utasítással beolvassuk az utána található számot, mely a gombnak az *id* száma. Ebben az esetben azért nem használható a *Serial.read()* utasítás, mert akkor a szám jelének a bináris kódját olvassa be a program, amelyet utána decimális számmá alakítva egy teljesen más számot kapunk. Az Arduino kapott szám alapján dönti el, hogy milyen utasítást kell végrehajtania.

A HTML-kódon stilisztikai változtatásokat hajtottam végre, illetve az adatok beolvasásához létrehoztam egy *iframe*-keretet, melybe az oldal az ESP által kiküldött adatokat jeleníti meg. Az *iframe*-utasítás a *http://192.168.4.1:80/pin=0* –s szerverre kapcsolódik rá. Mint ahogy látható, az elérési útban a *pin=0* –s utasítás található, tehát a rendszer úgy érzékeli, mintha csak egy gombot nyomott volna meg a felhasználó, melyre elküldi az adott szerverre a mérési adatokat.

C. EEPROM-ok kezelése

A központ minden órában begyűjti az érzékelők adatait, így a felhasználói oldalon megjelenített adatok a legutoljára beolvasott adatok. Ez azt jelenti, hogy az adatok nem feltétlenül frissek. Azonban a HTML-oldalon található egy adatbegyűjtés gomb is, amellyel frissíteni lehet a mérési adatokat. Ez eltart néhány másodpercig, de az adatbegyűjtés után, ha a felhasználó frissíti az oldalt, akkor már az új, friss adatok jelennek meg. Az adatgyűjtésekből származó értékeket a rendszer lementi az egyik EEPROM-ba a mérés időpontjával együtt.

A projektben 24LC256 számú EEPROM-chipeket használtam. Az egyik chipbe a program a felhasználói utasításokat rögzíti, a másikba pedig az érzékelőadatokat.

Adatrögzítés a chipre, amelyen az érzékelők adatai kerülnek tárolásra, minden órában történik. Az adatok beírása meghatározott sorrendben megy végbe. A sorrend betartása azért is

fontos, mivel a kiolvasás során is ugyanebben a sorrendben kell az adatokat visszakapni. A sorrend a következő:

1. Időpont (hónap, nap, óra); 3 byte
2. Belső hőmérséklet; 1 byte
3. Belső páratartalom; 1 byte
4. Belső fényerősség; 1 byte
5. Külső hőmérséklet; 2 byte
6. Külső fényerősség; 1 byte
7. Központi egység hőmérséklete; 2 byte

Mivel az adatok mentése során a programnak pontosan tudnia kell, hogy mely területre írhat, az EEPROM első két byte tárterületén egy integer típusú változó kerül tárolásra, mely megadja, hogy utoljára hányadik címre történt mentés.

Az adatok kiolvasása az Arduino keretrendszer soros monitorja segítségével történik. Az adatok beolvasására külön programot írtam, mivel az eredeti vezérlőprogram már így is nagyon nagy volt. Miután az eszköz rá a számítógéphez lett csatlakoztatva, a soros monitoron keresztül elküldött *„ok”* utasítással lehet elindítani az adatok letöltését.

D. Tápellátás

A rendszert egy 12 V-os 6,6 Ah-s ólomsavas akkumulátor táplálja, mely utántöltéséről egy napelem modul gondoskodik. A napelem, mely az akkumulátort táplálja, egy amorfkristályos napelem modul, melyet az általam tervezett tetőkampóval lett a *2. ábrán* látható *A* betűvel megjelölt tetőfelületre felszerelve.

4. ábra: A felszerelt napelem

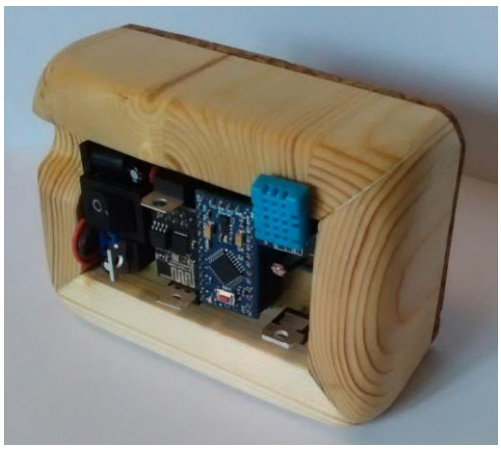
Az akkumulátor töltését egy Conrad töltésszabályozó vezéreli. A modul három csatlakozóval van ellátva. Eggyel a napelemhez, eggyel az akkumulátorhoz csatlakozik, egyre pedig fogyasztót lehet kötni. A modul védi az akkumulátort túltöltés (14,5 V) illetve mélykisütés (11 V) ellen is. Maximálisan 10 A kimeneti áramot képes leadni. Az adatlapja szerint a fogyasztó felé 12 V-os feszültséget biztosít, azonban tapasztalataim szerint ez az érték gyakran 12 V fölé emelkedik, főleg, ha napsütéses időben a napelem nagy feszültséget ad le. Így a központi modulba egy 12 V-os feszültségszabályzó kapcsolást is terveztem.

5.ábra: A felszerelt központi egység

A töltésszabályzóhoz más fogyasztókat is lehet kapcsolni, például egy USB adapter modult, mellyel telefonok vagy más USB csatlakozóval rendelkező eszközök is tölthetőek. A központi egységre egy deszkalap borítást helyeztem. A szobában felszerelt központi egység a töltésszabályzóval és az akkumulátorral az *5. ábrán* látható.

V. AZ ALEGYSÉGEK BEMUTÁSA

A rendszert úgy terveztem, hogy az eddig kiépített technika alapján könnyen lehessen alegységekkel bővíteni, melyek a központi egység Wi-Fi hálózatára kapcsolódva kommunikálni tudnak azzal. Ezek az alegységek elláthatnak utána mérési vagy vezérlési feladatokat is, mint például a belső hőmérséklet mérése vagy világításvezérlés. Az általam készített kis mobil mérőállomás a *6. ábrán* látható.



6. ábra: Mérőállomás

A mobilitás érdekében a modulok két darab 2000 mAh lítiumion celláról vannak táplálva. A cellák egyenként 3,7 V névleges feszültséggel rendelkeznek. A lítium cellák nagyon érzékenyek a túltöltésre és a mélykisütésre is, azaz a cellákat nem szabad 4,2 V fölé tölteni illetve 3 V alá meríteni.

A mélykisütés elleni védelemről egy komparátor üzemmódban működő LM358-as műveleti erősítő gondoskodik. Ez a típusú

műveleti erősítő tipikusan aszimmetrikus táplálásra lett tervezve. A referenciafeszültséget egy LF50CV feszültségszabályzó integrált áramkör biztosítja, mely 5 V-os feszültséget ad a kimeneti lábra, amely a műveleti erősítő invertáló bemenetére van bekötve. Az akkumulátor feszültségét egy feszültségosztó kapcsolás redukálja úgy, hogy mikor a két akkumulátor összfeszültsége 6,4 V, akkor 5 V legyen a kimenet, mely a műveleti erősítő nem invertáló bemenetére van kötve. A műveleti erősítő kimeneti lába egy az áramkörbe épített tranzisztorhoz kapcsolódik. Így, ha a két cella összfeszültsége 6,4 V alá csökken, a komparátor a lekapcsolja a kimeneti feszültséget, mely hatására a tranzisztor megszakítja az áramkört.

A cellák töltését egy MCP73822 jelű töltésszabályzó integrált áramkör vezérli. Ezt a típust jellemzően 2 cellás töltésre tervezték 8,4 V-os töltőfeszültséggel.

Az Arduino programban a DHT11-es mérőmodul működtetéséhez először meg kell hivatkozni a *DHT.h* függvénykönyvtárat, majd a *dht.setup(pin)* utasítással meg kell adni, hogy az Arduino mely bemenetére van kötve a szenzor. A I/O portnak mindenképpen PWM képesnek kell lennie. Miután az Arduino beolvasta a hőmérséklet- illetve a páratartalom-értékeket, beolvassa az A0-ás analóg portján a fotoellenálláson eső feszültséget. A kapott értékeket továbbítja a központi egységnek a korábban látott módon úgy, hogy központi egység elküldi annak az értéknek a kódszámát, amelyet be akar olvasni. Majd az alegység visszaküldi a mért értéket úgy, hogy elé írja a megnevezését (pl.: *tben=22*). Ekkor a központ kiolvassa a „tben=” karakterekig a soros porti buffert, majd a *Serial.parseInt()* utasítással az értéket. A kommunikáció végén a mérőegység zárja a kapcsolatot.

VI. RELUXAVEZÉRLÉS

A projekt során felhasznált, hagyományos bambusz reluxát egy ablakra helyeztem fel. Mivel a reluxát a külső ablakkeretre rögzítettem, nyáron nem jut be az ablakon direkt napsugárzás, ami melegítené a szobát.

A reluxalemezek szögének a változtatásához egy 28BYJ-48 típusú léptetőmotort szereltem fel. Ez a motor csomagban vezérlő-shielddel együtt kapható. A shield egy ULN2003APG chipre épül, melyet 5-12 V-os feszültséggel lehet táplálni. A reluxa házának közepén található egy tengely, mely forgatásával lehet a lamellákat dönteni. Ehhez a tengelyhez csatlakozik a motor egy fogaskerekes áttétellel. A tápellátáson kívül 4 bemenete van a shield-nek, melyekre az Arduino azon kimeneteivel kell csatlakozni, amelyekkel a motorban lévő tekercseket akarjuk vezérelni. A vezérlő áramkör a reluxa házának ablak felőli oldalára lett felrögzítve a *7. ábrán* látható módon.

7. ábra: Reluxavezérlő áramkör

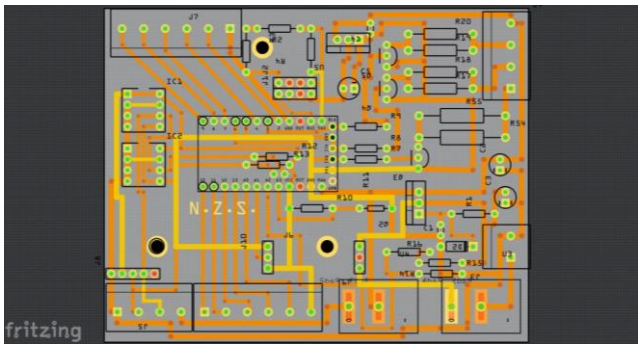
A motorvezérlő shield-et egy relé kapcsolja, így csak akkor kerül áram alá, ha az Arduino meghúzza a relét. A reluxa házban lévő tengelyéhez csatlakozik még egy 100 kΩ-os potenciométer, mely segítségével az Arduino be tudja olvasni, hogy a reluxa lamellák épp ilyen állásban vannak. Az ablak mellé felhelyeztem egy mérőállomást, mely a külső hőmérsékletet és fényerősséget méri.

Ezzel az egységgel a központ I^2C soros buszon keresztül kommunikál.

VII. NYOMTATOTT ÁRAMKÖR KÉSZÍTÉSE

A nyomtatott áramkörök elkészítését a központi egység áramkörének az elkészítésével szeretném bemutatni. A nyomtatott áramköri terveket a *Fritzing* [9] nevű open source programmal készítettem el.

A tervezés során felhasználó először is össze tudja rakni az elemeket az általa tervezett áramkörhöz egy virtuális próbapanelen. Az áramkör tesztelhető, mivel programot is meg lehet adni a vezérléséhez. Ezután össze lehet állítani a kapcsolási rajzot és a nyomtatott áramköri tervet. A program automatikusan megjelöli az összeköttetéseket.

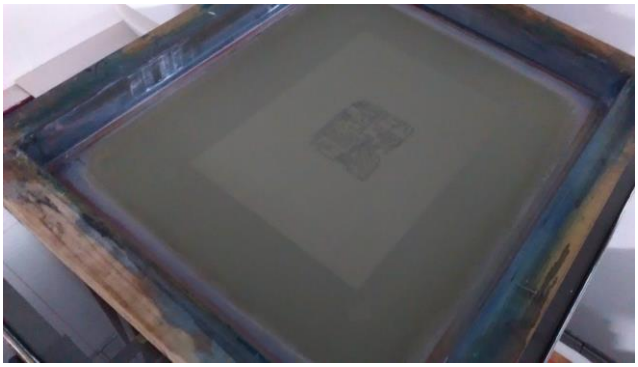


8. ábra: Nyomtatott áramköri terv

Tervezés során a lehető legkisebb lemezre törekedtem, amellett, hogy az alkatrészek ne is kerüljenek túl közel egymáshoz, a hőtermelődés miatt.

A NYÁK-lemezt szitanyomtatással készítettem el édesapám segítségével, az ő grafikai műhelyében.

A szitanyomtatás technológia lényege a következő: Fa keretekre egy igen sűrű szövésű szitát feszítenek ki, mely utána be lesz vonva két réteg emulzióval. Az emulzió két komponensből áll, egy bázisanyagból és egy fényiniciátor-anyagból. Egy bizonyos hullámhosszúságú fényre az emulzió „kicserződik", azaz a bázisanyag polimerré válik, amelyet többé már nem old a víz. Ezután a szitából, csak speciális rétegtelenítő anyaggal lehet kimosni. Ahol a fény nem érte az emulziót, ott vízoldható marad, és kimosható a szitából a levilágítás után. A nyomtatás során ezeken a helyeken folyik le a festék a lapra, így kiadva a mintát.



9. ábra: Levilágításhoz előkészített szita

Az alsó és felső oldalról készült terv (PDF formátumban) tintasugaras nyomtatóval ki lett nyomtatva pauszpapírra. A pausz egy erre kialakított üvegdobozra lett elhelyezve, majd erre került a szita. Ezután alulról meg lett világítva a szita úgy, hogy az áramköri terv a szükséges helyeken leárnyékolta. Ezeken a helyeken az emulzió továbbra is vízoldható maradt. A megvilágítás 27 percig tartott 2 db 300 W teljesítményű higanygőzizzóval. A megvilágítás után a szita leárnyékolt részeiből vízzel ki lett mosva az emulzió. A sziták száradása után kezdődhetett maga a nyomtatás.

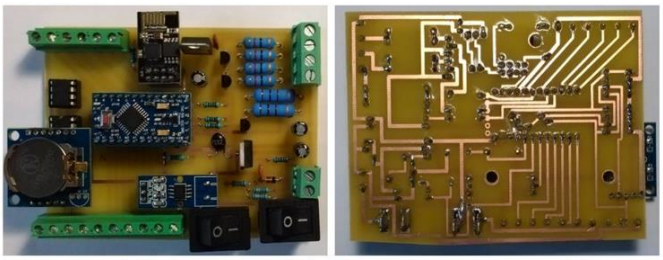
A lemezeket méretre kellett vágni. A vágást egy gérvágó körfűrésszel végeztem. A lemez ezután nedves polírpapírral fel lett csiszolva. A méretre vágott és polírozott lemezekre ezután rá lehetett szitázni az áramköri terveket. A lemezt a szitaasztalra tettük, majd hozzáigazítottuk magát a szitakeretet.

A szitanyomtatás során a festéket egy raklival (rákelkés) húzzák végig a szitán, mely az emulzióval nem fedett részeken lefolyik a munkadarabra, így a rézlemezre rákerült az áramkör mintája.

A NYÁK elkészítésének egyik fő problémája az volt, hogy a nyomtatott áramkör csak kétoldalas lemezen lehetett megvalósítani, ez viszont megnehezítette a forrszemek pontos pozicionálását. A probléma úgy lett megoldva, hogy miután a lemezre az alsó oldali minta felkerült és megszáradt a festék, a rögzítésre szolgáló három lyukat kifúrtam. Ezt egy fúróállványra rögzített kézi fúróval végeztem el. Így a felső oldal mintáját már ezekhez a lyukakhoz lehetett pozicionálni.

Miután a festék a felső oldalon is megszáradt, a lemez közepes sűrűségű $FeCl_3$ oldatban lett kimaratva. A maratás ebben a savban nagyjából 25 percig tartott szobahőmérsékleten. A marási idő csökkenthető a sav melegítésével. A lemezt ezután bő vízzel letisztítottuk. Ekkor még a festék rajta volt a vezetősávokon, így az denaturált szeszbe mártott ecsettel lett eltávolítva. Ezután a denaturált szesz is le lett tisztítva a lemezről, majd a forrszemek kifúrása következett. Ehhez 0,5 mm-es fúrófejet használtam. A sorkapcsok és a billenőkapcsolók helyeit ostorköszörűvel tágítottam ki.

Végül az alkatrészek beültetése és beforrasztása következett. Az elkészült áramkör a *9. ábrán* látható.



10. ábra: Elkészült áramkör (alsó és felső nézet)

VIII. ÖSSZEFOGLALÁS

A szakdolgozatomban leírt rendszer megalkotásával sikerült mélyebb betekintést nyernem az Arduino-programozásba és megismertem az ESP8266-os chip használatát. Az Arduinoval kapcsolatos tapasztalataim is bővültek, mint például, hogy stabilabb működésre képesek, mint ahogyan azt kezdetben gondoltam. Az ESP-01-es modul használatával azonban sok nehézségbe ütköztem. Bár sikerült a hibákat kiküszöbölnöm, sok időbe telt, míg minden problémára megoldást találtam.

A kész rendszer megfelel az általam kitűzött céloknak, mivel alkalmas automatizált működésre, viszont felhasználóként is könnyű vezérelni a rendszer működését.

Elsődleges célként fogalmaztam meg, hogy a kiépített rendszert később tovább tudjam fejleszteni, újabb funkciókkal és eszközökkel bővíteni. Ilyenek például újabb mérőegységek építése, világítást kapcsoló relék beépítése, és nem utolsósorban minden szükséges ablakba automatizált reluxa beépítése.

KÖSZÖNETNYILVÁNÍTÁS

Szeretném megköszönni Sarvajcz-Bánóczy Emese tanárnőnek, hogy elvállta és lelkiismeretesen elvégezte a projekt konzulensi feladatait. Köszönettel tartozom még a családomnak és menyasszonyomnak a támogatásukért. Illetve minden tanáromnak és hallgatótársamnak, akik segítségemre voltak a projekt elkészítésében.

HIVATKOZÁSOK

[1] „Arduino Weboldal," [Online]. Available: https://www.arduino.cc. [Hozzáférés dátuma: 23. 05. 2017.].

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

[3] „Espressif Systems," [Online]. Available: https://espressif.com/en.

[Hozzáférés dátuma: 23. 05. 2017.].

[4] „Github ESP8266 AT parancskészlet," [Online]. Available: https://github.com/espressif/ESP8266_AT/wiki. [Hozzáférés dátuma: 23. 05. 2017.].

[5] J. Chandramohan, R. Nagarajan, K. Satheeshkumar, N. Ajithkumar, P. Gopinath és S. Ranjithkumar, „Intelligent Smart Home Automation and Security System Using Arduino and WI-FI," *International Journal Of Engineering And Computer Science ISSN:2319-7242*, pp. 20694-20698, 03. 03. 2017..

[6] „SketchUp," [Online]. Available: https://www.sketchup.com/. [Hozzáférés dátuma: 23. 05. 2017.].

[7] „AllAboutEE," [Online]. Available: http://allaboutee.com/2015/01/02/esp8266-arduino-led-control-from-webpage/. [Hozzáférés dátuma: 23. 05. 2017.].

[8] „Rhydolabz," [Online]. Available: https://www.rhydolabz.com/wiki/?p=10872. [Hozzáférés dátuma: 23. 05. 2017.].

[9] „Fritzing," [Online]. Available: http://fritzing.org/home/. [Hozzáférés dátuma: 23. 05. 2017.].