Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/33.

Hőérzékelő & páratartalom mérő mood lamp tervezése és vezérlése Android Linux alapú alkalmazással

Nemes Arnold
Debreceni Egyetem,
Informatikai Kar
Debrecen, Magyarország
arnold.nemes96@gmail.com

Beatrix Papp
London South Bank
University
School of Law and Social
Sciences,
London, United Kingdom
pappb@lsbu.ac.uk

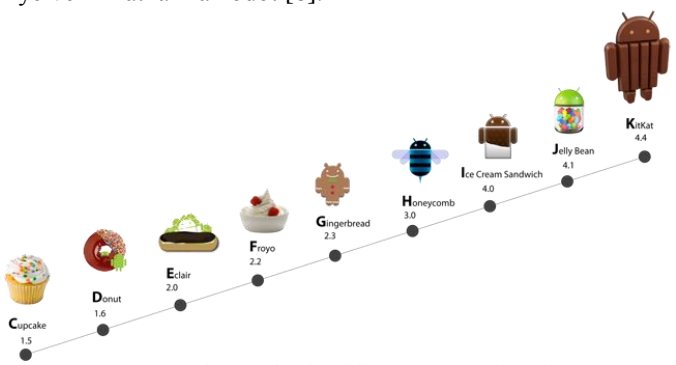
Erdei Timotei István
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
timoteierdei@eng.unideb.hu

***Absztrakt*—Napjainkban az okos telefonok világszerte elterjedtek. Az okos eszközök kezdetben még csak szerénynek mondható hardware specifikációkkal rendelkeztek, mára azonban nem ritkák a 8 magos 4gb RAM – al szerelt szériák sem, melyek alkalmasak kutatási és fejlesztési célokra. Ebben a projektben, egy olyan egyedi készítésű hőmérséklet és páratartalom érzékelésre alkalmas „mood lamp" került megalkotásra, mely Bluetooth segítségével vezérelhető szinte minden Android Linux operációs rendszerrel ellátott okos telefonról. Amit a HC-05 Bluetooth module is lehetővé tesz, melyet egy Arduino Nano vezérel, valamint a DHT11 szenzort is, illetve a mobil applikáció utasításait is kezeli. A lámpa kialakításánál, nagy hangsúly volt fektetve az esztétikára is.**

Kulcsszavak—Arduino; DHT11 sensor; RGB LED strip; MIT App Inventor; Bluetooth module; távvezérlés

I. BEVEZETŐ

Az egyre rohamosabban fejlődő világunkban az okostelefonok térhódítása folyamatosan növekszik, már-már mondhatni, némelyik készülék komolyabb hardverrel rendelkezik, mint számos asztali számítógép. Az Android Linux mobil operációs rendszer, elsősorban érintőképernyős okos telefonokra, táblagépekre lett tervezve. A fejlesztők Java nyelven írhatnak rá kódot [6].



1. ábra: Android mobil operációs rendszerek [7]

Az első Androidos telefon 2008 októberében jelent meg Az Android –os rendszerek egyik legnagyobb előnye, hogy Open-Source minősítésűek, így szabadon terjeszthetőek, módosíthatóak harmadik fél által. Jelenleg a az Android 8.1 Oreo-nál jár az Os verziója. A kutatás/fejlesztésnek a Debreceni Egyetem adott otthont [14].

II. TERVEZÉSI SZEMPONTOK

Nagyon sok vezeték nélküli, Bluetooth vezérlésű eszközzel találkozhatunk napjainkban, hangszóró, monitor, IP kamera stb.

Viszont a soron következő projectben megvalósított kimondottan ilyen típusú távvezérlésű hangulat/éjjeli lámpa még nem hódított teret [1].

A tervezési szempontok között szerepelt, hogy mindenféleképpen egy modern, a mai világba illeszkedő lámpa megvalósítása legyen a cél. Maga a külső fa elemet használ, illetve plexi lapokból lett megvalósítva, melyek darabszámát, és méreteit az I. táblázat tartalmazza.

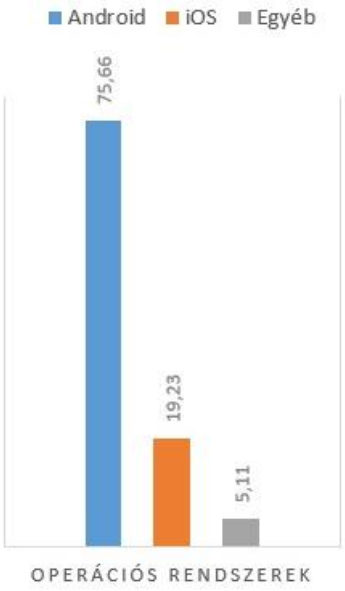
I. TÁBLÁZAT

Darabszám (db)	Anyag	Méret (mm)
1	Fa	130 x 130 x 60
4	Plexi	130 x 130 x 12
5	Fa	130 x 130 x 20

Az első 6cm magas „doboz" szolgál a vezérlés tárolására, illetve az adapter csatlakozó elhelyezése is ebbe került. Majd ezekre felváltva a további 5db fa és 4 db plexi lap jön, melyek közepét egy 54mm átmérőjű kör fúróval kellett kifúrni, hogy ezekbe a 40mm átmérőjű fa henger beleférjen, amire a LED csík lett felragasztva. Azonban az említett „doboz"-t követő első fa lap nem a közepén, hanem a jobb felső sarokban lett kifúrva. Ennek okai a következők:

1. A vezérlésből szükséges kábelek ezen a lyukon keresztül lettek felvezetve.
2. Az említett fa henger ezen a fa lapon lett rögzítve, az esetleges mozgások elkerülése érdekében.

A belső tervezési szempontok között szerepelt, hogy mindenképp Android platformra célszerű az alkalmazást megírni, mivel a világon ez a legelterjedtebb mobiltelefon operációsrendszer, ahogyan ezt a StatCounter 2018. Áprilisi statisztikája is mutatja (1. ábra).



2. ábra: A mobil operációs rendszerek elterjedtsége 2018 [8]

II. A RENDSZER FIZIKAI FELÉPÍTÉSE

Az egész lámpa vezérlése egy Arduino Nano [12] segítségével történik, ami az Arduino UNO miniatürizált verziója. SMD ATmega328 mikrokontrollert tartalmaz, melynek órajele 16MHz, valamint 14db digitális ki és bemenettel rendelkezik, ezekből 8db analóg bemenet. Az eszköz akár 20V árammal is képes működni, azonban a LED csíkhoz elég volt mindösszesen 5V-nyi áramfeszültség, és 2A. Ezt egy Akyga AK-TB-02-es adapter segítségével kapja meg az Arduino, amire a HC-05-ös Bluetooth modulnak is szüksége van [2].

A Bluetooth modul az Arduinon kivűl több termékkel is kompatibilis, bemeneti feszültsége 3,6V-6V-ig terjed. 6 tű-vel rendelkezik a modul, azonban ebből csak 4-re volt szükség mégpedig: RXD, TXD, GND, VCC bemenetekre. Továbbá 3 tranzisztorra, melyek a gyenge villamos jelek erősítésére, valamint feszültség stabilizálás céljára alkalmazhatóak.

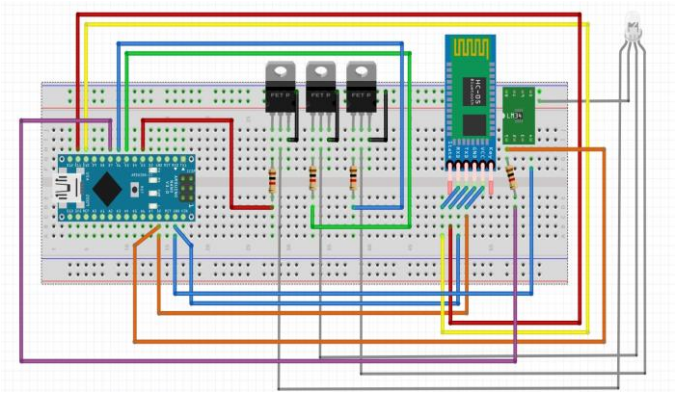
Végül a DHT11 hőmérséklet és páratartalom szenzor. A könnyű jelfeldolgozásának köszönhetően viszonylag sok kész alkalmazási szoftverminta érhető el. Egy másodperces válasz idővel rendelkezik, ez természetesen állítható, ez a leggyorsabb válasz idő, amit képes küldeni. A hőmérséklet mérési tartománya 0-tól 50 celsius fok, de ez +/- 1 fokkal eltérhet. A mérési páratartománya 20RH%-tol 90RH%-ig mér, itt is meg lehet egy +/- 2RH% eltérés, mivel nem hajszálpontos a szenzor. A működési feszültségtartománya: 3.3-5V [9].

Az egész egy próba NYÁK-ra, -amely különálló forrszemekkel rendelkezik- lett ráforrasztva.

A LED csíkhoz eredetileg tartozott egy 4,5V-os áramellátás, illetve egy kis kontroller, amivel a színeket, és a funkciókat lehet változtatni. Azon az az áramerőforrás nem volt elegendő a távvezérlésre alkalmas lámpához, ezért ezt a már említett adapter váltotta fel. A kontroller szintén eltávolításra került, mert a különböző színek, valamint effektek megírása az Arduino ingyenesen beszerezhető (1.8.5 verziószámú) IDE-jében történt [10].

III. MEGÉPÍTETT ÁRAMKÖR

Az áramkör Fritzingbe [11] lett megvalósítva, ami tartalmazza az Arduino Nano panelt, ami a Bluetooth modult és a DHT11 szenzort, valamint az RGB LED csík összekapcsolását végzi [3]. A 3. ábrán látható a megvalósított próbapaneles áramkör.



3. ábra: Próbapaneles áramkör

A led csík 4 kimenete (+, R, G, B) tehát a föld, piros, zöld, kék. Mivel anód lába van a LED csíknak, így az Arduino 5V-jára lett rákötve, a három szín pedig a tranzisztorok középső lábára. A bal oldali lábakra egy 1Kohm ellenállás, amit az Arduino D3, D5, és D6 lábaira lettek forrasztva piros, zöld, kék sorrendben. A jobb oldali lábak pedig földelve lettek, amik egységesen az Arduino GND-jéhez lettek forrasztva. A bluetooth modul 4 kimenete a következő sorrendben kerültek felhasználásra. Az RXD láb, az Arduino D11-hez, a TXD pedig a D10-hez lett kapcsolva. A GND az Arduino GND-jére, a VCC pedig az Arduino 5V-jára lett forrasztva. Az utóbbi 2 bemenet ki lett vezetve oldalra, hogy ebbe a DC csatlakozó pozitív és negatív oldala csatlakozzon, így az adaptert csak egy csatlakozóba kell bedugni, hogy áramellátást kapjon az egész.

A hőmérséklet és páratartalom mérő szenzor négy lába közül, csak három kerül felhasználásra. Szemből nézve, balról jobbra a következő sorrendben. VCC (+) láb, az Arduino 5V-jára, egy 1Kohm-os ellenálláson keresztül, a második, Signal (jel küldő) láb az Arduino D7-es lábához lett kivezetve, még a negyedik, egyben utolsó láb a földelés (-) amely az Arduino GND-jéhez csatlakozik.

IV. ARDUINO PROGRAM TERVEZÉSE

A programozásra a már említett Arduino integrált fejlesztői környezetében került sor, ami egy egyszerűsített C nyelven történik [4]. Álltalában az Arduino kódok két részből állnak, egy setup(), és egy loop() eljárásból. A setup() eljárás

DOI: 10.17667/riim.2018.si/33.

jellegzetessége, hogy a program indulásakor csak egyszer fut le, feladata, hogy a program futását előkészítse, jellemzően a változók, konstansok deklarálást tartalmazza. Ennek az eljárásnak mindenféleképpen szerepelnie kell a kódban, még akkor is, ha nem tartalmaz semmilyen utasítást. Mindenek előtt meg kellett hívni a SoftwareSerial.h könyvtárat, hogy minden Serial-t áttudjunk írni egy mySerial függvényre. Erre azért volt szükség, mert ha a megírt programot fel szeretnénk tölteni az Arduino Nano-ra, akkor egy hibaüzenetet kaptunk, és a bluetooth modul kivétele után tudtuk a feltöltést megvalósítani. Miután alkalmaztuk a mySerial-t már nemvolt ilyen felesleges lépésekre szükségünk.

Továbbá szükség volt a dht.h könyvtárra is, hogy a szenzort megfelelően tudjuk használni a könyvtár által használható függvények, parancsok segítségével.

A setup() első sorában az Adruino – Bluetooth modul kommunikációhoz egy soros port kerül definiálásra, ami nélkülözhetetlen. A paraméterül adott 9600-as érték „baud rate” (átviteli sebesség) ami a másodpercenként küldött/fogadott jelek számát adja meg. A következő sorokban a pinMode() utasítással az első paraméterként megadott „lábakat” állítjuk be OUTPUT-ra azaz kimenetre. Ezek még a setup() eljárás előtt deklarálva lettek, mivel a D3, D5, D6 lábakra lettek kötve a színek így a 3, 5, 6 konstanst kapták. Az analogWrite funkció segítségével a LED csík kikapcsolt állapotba kerül, azaz a redPin, greenPin, bluePin is 0V feszültséget kap. Azonban ha alapból azt szeretnénk, hogy fehéren világítson, ha áram alá helyezzük, akkor mindegyik analogWrite funkció második paraméterét 255-re kellene állítani, ezzel 5V feszültséget adva, mindegyik színnek.

```
#include <SoftwareSerial.h>
#include <dht.h>

dht DHT;
#define DHT11_PIN 7
SoftwareSerial mySerial(10, 11);

int redPin = 3;
int greenPin = 5;
int bluePin = 6;
char color=0;

void setup() {
  mySerial.begin(9600);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  analogWrite(redPin,0);
  analogWrite(greenPin,0);
  analogWrite(bluePin,0);
}
```

4. ábra: Arduino kód include, setup() eljárás, illetve deklarációs része

Ezután jön a loop() eljárás, ahol megvizsgáljuk, hogy elérhető e a bluetooth modul, ha igen, akkor a color nevű változóba beolvasunk egy karaktert, ha ez nem 0, akkor a megfelelő if-es utasítást fogja végrehajtani. Ezek után, a dht könyvtár egy függvénye, a DHT.read11() következik, amely annak a „lábnak” a számát várja, amire rákötöttük a szenzor jel lábát, ami jelen esetben a 7. Ezzel a függvénnyel olvassuk be, a szenzor által mért hőmérsékletet, illetve páratartalmat.

```
void loop() {
  if(mySerial.available() > 0){
    color = mySerial.read();
    if(color != '0'){
      mySerial.println(color);
    }
  }

  DHT.read11(DHT11_PIN);
  delay(4000);
```

5. ábra: A loop() eljárásból egy részlet

A színek beállítása a következő módon történik. Egy if feltételben megvizsgáljuk, hogy a már korábban deklarált color változó milyen értéket kapott az Android alkalmazástól. Minden színhez illetve effektekhez egy ahhoz jellemző betű lett hozzá rendelve, mint azt a II. táblázat is mutatja.

II. TÁBLÁZAT

Color	Character
Off	n
Red	r
Green	g
Blue	b
White	w
Orange	o
Violent	v
Cyan	c

Az egyes színek megadásához egy RGB kód táblából kellett kiválasztani a megfelelő értékeket, majd a három szín második paraméterében 0-tól 255-ig behelyettesíteni azt. Ezzel megmondva a programnak, hogy melyik szín mekkora áramerősséget kapjon, így kialakítva a kívánt színt. Az egyes funkciók/effektek megírása már nagyobb probléma. A „Fade” effektnél, gyakorlatilag az összes szín lejátszódik, amire egy RGB led képes. Mindhárom szín értéke a 0 és 255 skálán végigmegy, ami pirosból indul, azután kékbe megy át, majd zöldbe, és végül újra piros. Ez három darab változóval, és for ciklusokkal lett meg oldva. Kezdetben a piros kapja folyamatosan növekvő áramot, azaz a ciklus végén az 5V-ot. Majd ehhez folyamatosan 30 ms elteltével növeljük a kék áramerősségét, mindaddig, amíg 255 nem lesz. Ezután a piros erősségét csökkentjük, ilyenkor a lámpa teljesen kék színben világít. Majd a zöld értékét növeljük, ezután a kék áramerősséget csökkentjük folyamatosan, ezután vállnak teljesen zölddé a LED-ek. Végül a pirosat növeljük, és a zöldet csökkentjük, így végül újra pirosat kapunk, ahogyan az 6. ábrán látható csúszka mutatja, balról, jobbra.

6. ábra: Az „átmenet” funció szín skálája [13]

DOI: 10.17667/riim.2018.si/33.

A folyamatos 30ms-os késleltetés miatt, az átmenet lassan jön létre, hogy minden szín kivehető legyen a folyamat alatt.
Továbbá a mért hőmérséklet szerint is tud világítani a lámpa, amely 6 féleképpen lehetséges. A 6 intervallum a III. táblázatban látható.

III. TÁBLÁZAT

Celsius (C)	R,G,B érték
C <= -16	0,0,112
-15 <= C <=0	0,0,204
1 <= C <= 10	0,204,255
11 <= C <= 20	255,255,0
21 <= C <= 30	255,0,0
C >= 31	255,0,255

Mivel a hőmérsékletet folyamatosan 4 másodpercenként frissíti a program, így ha a telefonról a „h” betűt küldő gombot nyomjuk meg, akkor 4 sec elteltével, az a szín fog világítani, amely intervallumba a hőmérséklet beleesik.

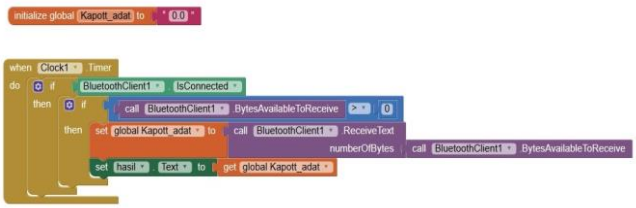
```
if(color == 'h'){
  if(DHT.temperature <= -16){
    analogWrite(redPin,0);
    analogWrite(greenPin,0);
    analogWrite(bluePin,112);
  }
  if(DHT.temperature >= -15 && DHT.temperature <=0){
    analogWrite(redPin,0);
    analogWrite(greenPin,0);
    analogWrite(bluePin,204);
  }
  if(DHT.temperature >= 1 && DHT.temperature <= 10){
    analogWrite(redPin,0);
    analogWrite(greenPin,204);
    analogWrite(bluePin,255);
  }
```

7. ábra: Hőmérséklet szerinti világítás Arduino kódja (részlet)

IV. LINUX MOBIL APPLIKÁCIÓ TERVEZÉSE

Az applikáció az MIT App Inventor segítségével lett készítve [5]. Ez egy online felület ahol könnyen lehet mobil applikációt készíteni, mind a dizájn, mind a gombok programozása terén. Minden szín egy külön gombbal rendelkezik, illetve az effektek is egy-egy gomb segítségével aktivizálhatók. Mivel az applikáció Bluetooth-on küldi a jeleket az Arduinonak ezért fontos volt egy úgynevezett „BluetoothClient”, ami egy „non visible components”, a megfelelő működés érdekében. Szükség van egy ListPicker-re amivel a HC-05 modulhoz tudunk csatlakozni. Ennek az első blokknak köszönhetően, ha megnyitjuk a Bluetooth listát látni fogjuk a „link” klienseket. A következő blokk, a csatlakozó blokk, aminek segítségével megvalósítjuk a BT modul és a mobil alkalmazás közötti kapcsolatot.

Ahhoz, hogy a hőmérséklet a telefonon folyamatosan frissülni tudjon, és 4 másodpercenként kiírja az aktuális értéket, inicializálni kellett egy globális változót melynek kezdőértéke 0. Ezek után (a non visible clock timernek köszönhetően) már csak vizsgálni kell, hogy a buetooth kliens csatlakozva van-e, ha igen, meghívja a .BytesAvailableToRecive blokkot (ha az nagyobb, mint 0), majd kiírja az aktuális értéket a Label mezőbe.

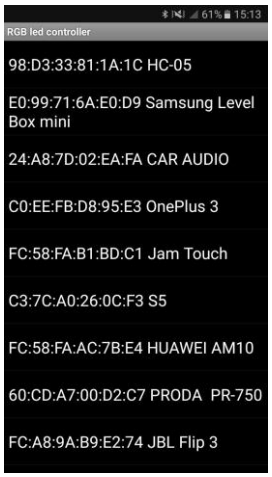


8. ábra: MIT App Inventor blocks (részlet)

A további blokkok a már korábban említett karaktereket küldik az Arduino, hogy az feldolgozza, és ez által a loop() eljárásban lévő megfelelő if utasítás lejátszódjon. Ehhez minden gombra, egy szövegküldő blokkot kell készíteni. Tehát a piros gomb lenyomásakor az applikáció egy „r” karaktert küld, ezáltal a color változó az „r” értéket kapja, mindaddig, amíg ezt meg nem változtatjuk egy másik gomb lenyomásával.

V. RGB LED CONTROLLER

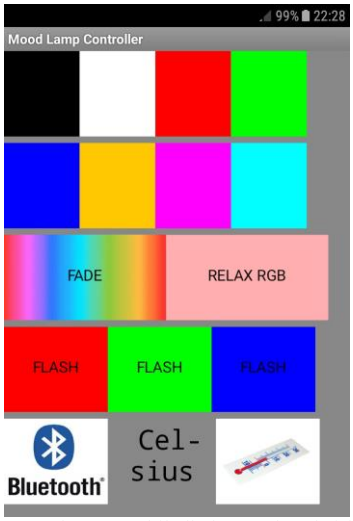
Az applikáció az „RGB led controller” nevet kapta. Kezelése a következőképpen néz ki. A bluetooth bekapcsolása után, megnyitjuk az applikációt, majd a Bluetooth ikonnal ellátott gombra kattintva behozzuk a korábban, már párosításra került eszközöket. Ezek között lesz a HC-05 modul, ami előtt a MAC címe látható, mint minden párosított eszköznél, ahogyan azt a következő screenshot is mutatja.



9. ábra: Az alkalmazás Bluetooth párosítása a HC-05 modullal.

A kiválasztás után a kapcsolódás egyből megtörténik, ez onnan is észrevehető, hogy a próba NYÁK-ra forrasztott Bluetooth modul lassabban villog. A csatlakozást követően, a Celsius helyére automatikusan írja a hőmérséklet értékét, 4 másodperces frissítéssel, a hőmérő gombra nyomva, pedig megjelenik az adott intervallumba eső hőmérséklet színe.

Továbbá a telefonon látható gombok megérintésével tudjuk változtatni a színeket. A kezelőfelület a következő ábrán látható.



10. ábra: A mobil alkalmazás kezelőfelülete.

VI. Mood Lamp Tesztelés

Mivel a vezérléshez muszáj hozzá férni, ezért a „doboz” amiben elhelyezkedik, csavarokkal lett rögzítve, a felette lévő fenyő illetve plexi lapok, Fix All ragasztóval, ami minden felületre alkalmas. A henger alakú rúd ipari fa ragasztóval lett beleragasztva egy 3mm mély kimart lyukba, és erre lett az 1 méter hosszúságú RGB led csík felragasztva. Mivel a csík hátulja öntapadós ezért egy védőfólia levétele után könnyen ragasztható bármilyen felületre, mint az látható a következő ábrán. A ragasztás után érdemes több órára szorítóba hagyni, hogy az jól meg kössön. Ezek után az esztétikus kinézet elnyerése érdekében, egy finom összecsiszolásra van szükség, hogy a fenyő, illetve a plexi lapok teljesen passzoljanak.

Mind ezek előtt a tesztelés sem maradhatott ki. Az egész vezérlés, eleinte csak egy próba panelről lett működtetve, hogy a DHT11-es szenzort kitapasztalva, a lehető legjobb működést érjük el, (mivel a frissítési idő, hogyha 4 másodperc alatt volt, kissé zavaros jeleket adott vissza) továbbá a megfelelő Arduino program tesztelése akadálymentesen történjen. A nyáklapra való ráforrasztás után, amikor már az adapter csatlakozója is a helyére került, eljött a végső tesztelés pillanata. A hőmérséklet szenzort 36 celsius fokig lett tesztelve, tökéletesen működött. A LED csík minden színben erősen világított, nem voltak, led pislákolások és hasonló gyenge áramforrásra utaló jelek. A lámpa több órás működés után is tökéletesen működik, kisebb melegedések érzékelhetők. A bluetooth kapcsolat létrejötte után akár több mint 25 méteres távolságból is vezérelhető a lámpa tökéletesen.

VII. Összegzés

A több színben pompázó hangulatlámpa megépítése, távvezérlése megvalósításra került, mely bluetooth kapcsolat segítségével tökéletesen irányítható. Üzembe helyezés, az utolsó simítások után, 10. ábra.

11. ábra: Az elkészült „Mood lamp”.

Pontosság szempontjából, a maximális távolságból vezérelve a lámpa késése maximum 0,3 másodperc, ez azonban nem befolyásolja a lámpa üzemszerű működését, továbbá a hőmérséklet szenzor mérései néhány celsius fokkal eltérhet a valós hőmérséklettől.

Következtetésképpen elmondhatjuk, hogy a távvezérlés megbízhatóan, pontosan, és biztonságosan működik. A mobil eszköz biztosítja a távirányítást végző ember számára a mobilitást.

VIII. Köszönetnyílvánitás

A publikáció elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.
Szeretném köszönetem kifejezni a projekt elkészítése alatt adott hasznos tanácsaiért és észrevételeiért Erdei Timotei Istvánnak, a Debreceni Egyetem oktatójának, valamint Bodnár Zoltánnak a Certa Kft. dolgozójának, aki az áramkör megépítéséhez adott hasznos tanácsokat, és utasításokat.

IX. Hivatkozások

[1] U. Shah, S. Shah, „Arduino BLINK Blueprints”, Packt Publishing, 2016.
[2] P. Mashkov, T. Pencheva, B. Gyoch, „Lamp with rigid LED strip – Design and thermal management”, Electronics Technology (ISSE), 2011.
[3] Arduino, (2017, December 17). [Online]. Available: https://www.arduino.cc/
[4] Fritzing, (2017, December 17). [Online]. Available: fritzing.org/home/
[5] MIT App Inventor, (2017, December 17). [Online]. Available: appinventor.mit.edu/explore/
[6] Android history, (2017, December 17). [Online]. Available: https://www.android.com/history/#/marshmallow

[7] Android history version, (2017, December 17). [Online]. Available: http://1.bp.blogspot.com/-cIs2nafXk2w/UreWZh6QJvI/AAAAAAAAT5o/cP5LM0R4BsY/s1600/innovationm-android-release-history.png

[8] StarCounter, (2017, December 17). [Online]. Available: https://starcounter.com/

[9] DHT11, (2017, December 17). [Online]. Available: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf

[10] Arduino IDE, (2017, December 17). [Online]. Available: https://www.arduino.cc/en/Main/Software

[11] Fritzing, (2017, December 17). [Online]. Available: http://fritzing.org/home/

[12] Arduino NANO, (2017, December 17). [Online]. Available: https://www.arduino.cc/en/Main/Products

[13] Rapid Tabels, (2017, December 17). [Online]. Available: https://www.rapidtables.com/

[14] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „A Novel Design of an Augmented Reality Based Navigation System & its Industrial Applications," 15th IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era Budapest, Hungary, 6 – 7 June, 2017 - Organised by: MTA SZTAKI – Hungarian Academy of Sciences - Institute for Computer Science and Control.