

Időjárás előrejelző mérőállomás Raspberry Pi 3 Raspbian Linux alapon

Horváth Milán
Informatikai Kar
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
horvathmilan96@gmail.com

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

Absztrakt— A Raspberry Pi megjelenésével rengeteg testreszabható projekt elkészülhet, amelyek különféle szenzorokat illetve egyéb kompatibilis eszközöket használnak. A Raspberry Pi 3 sokoldalúságát kihasználva ebben a projektben egy hőmérséklet- és páratartalom érzékelő segítségével automatizálódik a LED-ek megvilágításának egy része, míg a másik része a python programozás előnyeit kihasználva, a napi maximum szerint választja meg színét. A különböző hőmérséklet intervallumok különböző színeket eredményeznek. A LED-ek „láthatatlanságát” egy fából készített felhő alak fedí el, melynek formája utal a használati területre, illetve dizájn szempontjából is sokkal esztétikusabb.

Kulcsszavak—Hőérzékelő, Raspberry, SSH, ütemezés, digitális LED, python

I. BEVEZETŐ

A Raspberry Pi első sorban az elektronika és a programozás összefűzését elősegítő oktatási és hobbi célokra tervezett eszköz. Egyik nagy előnye az alacsony fogyasztása, illetve az ingyenesen telepíthető rendszerek. A Raspberry olyan ingyenes Linux disztribúciókat képes futtatni, mint például a Raspbian [1] vagy az Ubuntu MATE [2]. Ezek a rendszerek grafikus felületű operációs rendszerek, amelyek rengeteg alapvető alkalmazással és optimalizációval rendelkeznek.

A telepített rendszerek egyik nagy előnye, hogy mivel ezek ingyenes és nyílt operációs rendszerek, ezért nem szükséges a különféle jogosultságokkal foglalkozni, ugyanis a telepített rendszert könnyedén használhatja a felhasználó rendszergazdaként, amelyet a Raspbian úgynevezett *superuser*-ként hív. Ahhoz, hogy programokat, utasítások *superuser*-ként hajtson végre a rendszer, a futtatandó utasítás előtt a „*sudo*” kapcsolót kell alkalmazni.

A Raspberry Pi 3-ra egy Raspbian nevű operációs rendszer kerül, melynek segítségével könnyedén futtathatjuk a kívánt programot, illetve nagyon egyszerűen tudjuk azt automatizálni, esetleg az eredményeket eltárolni későbbi, statisztikai kimutatásokhoz.

Ennek a projektnek a tervezési és létrehozási folyamatához a Debreceni Egyetem, Informatikai Tanszék laborja biztosított helyet. A kutatás/fejlesztésnek a Debreceni Egyetem adott otthont [8].

II. TERVEZÉSI FÁZIS

A projekt tervezésének első szempontja a digitális LED címezhetőségének a kihasználása, így több területet is figyelembe lehet venni a hőmérsékleti adatok felhasználóval való közlésekor.

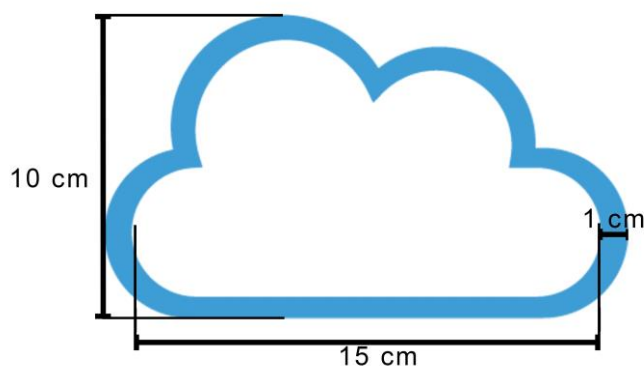
Az első szempont szerint az úgynevezett belső hőmérséklet, vagy aktuális hőmérséklet mérése volt a cél, melynek során az adott helyiség hőmérsékleti adatait egy DHT 11 hőmérséklet- és páratartalom érzékelő modul segítségével kapjuk meg.

A második szempont a külső napi hőmérsékleti maximum, illetve minimum meghatározás volt a cél, melyeket a Python programozási nyelv segítségével könnyen megkaphatunk.

A digitális LED segítségével mind a két értéket ki lehet „közvetíteni” a fényekre. A terv szerint a 12 LED-ből az első 6 az aktuális, vagy benti hőmérséklet szerint fogja választani színét, a második 6 LED pedig a külső, napi maximum hőmérséklet szerint.

A felhő alakja méretének megtervezésekor ezt figyelembe véve készült el a minta, ami alapján létrejött a LED-eket eltakaró dizájn doboz. Méreteit tekintve: 10cm x 15cm x 1cm

Az elkészült minta az 1. ábrán látható.



1. ábra A doboz méreteinek terve

III. A FELHŐ FELÉPÍTÉSE

A doboz elkészítésénél figyelembe véve a LED szalag szélességét, ami 1 cm, több darabra volt szükség, ugyanis 0,7 cm vastagságú fa lapból kerültek kivágásra az elemek.

A 4 külön egység elkészítése után ezek összeillesztése, illetve összeragasztása volt a következő lépés. 3 rész összeragasztása után a LED szalag elhelyezése következett, melynél figyelembe véve a külön színeződést, 6-6 LED külön „tartományba” került, így megelőzve az esetleges nagy színkeveredést, illetve a 3 csatlakozási pontnak egy 0,7 cm átmérőjű furat készült, melyeken egy 3 erű kábel csatlakozik a LED szalagra.

A LED szalag elhelyezése után annak meggyőződése érdekében, hogy minden LED a tervezett szempontok szerint világít, illetve, hogy a hőmérsékleti adatoknak megfelelően választja meg a szín kombinációkat a program tesztelésre került. A tesztelés során a kapott hőmérséklet, illetve a napi maximum megállapítása után a program a helyes értékeket kiküldve, a hozzá tartozó színekben világította meg az első 6, illetve a második 6 LEDet, ahogy a 2. ábrán látható.



2. ábra A program tesztelése az elhelyezett LED szalaggal

Ahhoz, hogy a LED szalag ne legyen látható, olyan megoldásra volt szükség, amelynél a fények átszűrődnek. A legegyszerűbb megoldásnak a pauszpapír tűnt, amely a felhő alakjára szabva, majd az utolsó elemet rá helyezve került végső állapotába. A száradás után egy újabb tesztre került sor, amely során a már kész állapotban lévő felhő lett megvilágítva az adott hőmérsékleti adatok szerint, illetve az esztétikusabb kinézet érdekében az egyes elemek összecsiszolásra kerültek, amely az 3. ábrán látható.

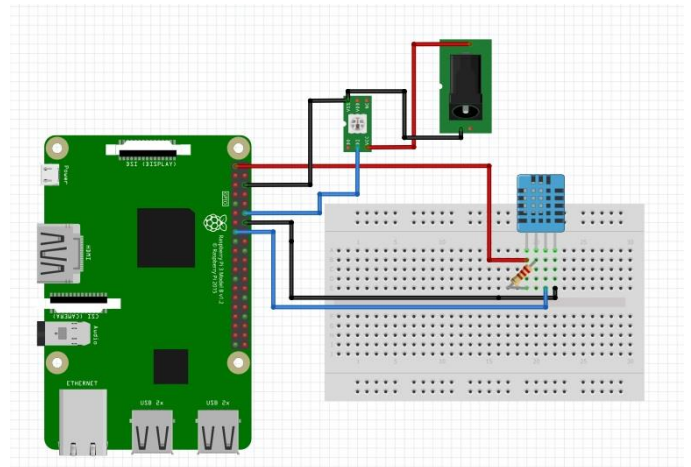


3. ábra Az elkészült felhő alakú doboz

Ez után következhetett a program automatizált és dokumentált működésének ellenőrzése, amely során létrejött dokumentációkból, értékekből könnyen statisztikai adatokat lehet készíteni.

IV. MEGÉPÍTETT ÁRAMKÖR

Az áramkör Fritzing-ben [3] került megtervezésre, aminek fő része a Raspberry Pi 3 model B, amely a DHT 11 Hőmérséklet- és páratartalom szenzort, illetve a WS2812 típusú LED szalagot köti össze, illetve vezérli ezek működését. A Raspberry a megírt Python program segítségével a DHT 11-en keresztül megkapja az aktuális helyiségben lévő hőmérsékletet, és egy úttal a napi minimum és maximum hőmérsékletet is, majd ez alapján választja ki, hogy melyik színnel világítsa meg a felhőt. Az 4. ábrán látható a megtervezett áramkör fizikai kapcsolása.



4. ábra Fizikai kapcsolás

A DHT 11 szenzor egy 4 lábás verzió, amely az alábbi lábkiosztással rendelkezik:

1. Vcc (+)
2. Signal
3. Not Used
4. Ground (-)

A helyes bekötéshez az 1-es (Vcc), illetve 2-es (Signal) láb közé egy 4,7K-s ellenállás szükséges, illetve egy fontos része a kapcsolásnak, hogy a szenzor, a LED szalag, illetve a táp is közös földdel rendelkezzen.

A szenzor kapacitív páratartalom mérővel és termisztorral rendelkezik, nincs szükség analóg portra a működéséhez. 2 másodpercenként van lehetőség egy új mérésre. 3.3-5V közötti tápfeszültség javasolt az üzemeltetéséhez. A környezeti hőmérséklet intervalluma 0-50 Celsius közötti, +-2 fok pontossággal.

A relatív páratartalom 20-95% közötti mérését 5% pontossággal képes elvégezni az eszköz.

A WS2812 típusú címezhető LED kihasználtságának érdekében került a programba az a funkció, amely a napi

hőmérsékleteket meghatározza, ugyanis így érvényesül a címezhetőség elve.

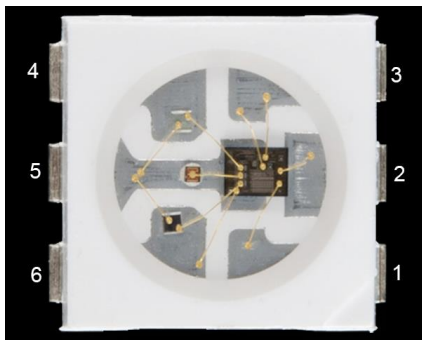
A LED szalag és a Raspberry összekapcsolásánál nincs szükség tranzisztorra, illetve ellenállásra a hagyományos típusú SMD 5050 LED szalaghoz képest.

A WS2812 típusú LED helyes lábkiosztása, illetve paraméterei az 1. táblázatban láthatóak, amely segítségével könnyedén megérthető a digitális LED működésének egyszerűsége.

1. TÁBLÁZAT WS2812 LÁBKIOSZTÁS

Láb	Név	Funkció
1	DO	Adat kimenet
2	DI	Adat bemenet
3	VCC	Vezérlő tápfeszültség
4	NC	Nem használt
5	VDD	LED tápfeszültsége
6	VSS	Közös pont (GND)

A táblázatba foglalt adatok, illetve lábkiosztás megértéséhez a 5. ábrán látható felépítés van segítségül.



5. ábra WS2812 LED felépítése

V. TESZTELÉS

A folyamat első lépése a DHT 11 tesztelése volt, mely a Raspberry-hez csatlakoztatva zajlott. A legegyszerűbb tesztelése a szenzornak, hogy valós értékeket kapunk-e, hogy futtatjuk a programot úgy, hogy a szenzor egy olyan helyen van ahol nagyjából tudjuk a hőmérsékletet. Ebben a projektben fontos a folyamatos valós információ, ezért a szenzor úgy került tesztelésre, hogy egy olyan helyen ahol volt információ a hőmérsékletről, folyamatos programfuttatás történt. Ez nem manuálisan történt, hanem a Raspbian rendszeren lévő CRON [7] segítségével.

A CRON egy olyan „alkalmazás”, amely segítségével ütemezhetjük a programjaink futásait, megadhatjuk, hogy hány percnként, óránként, naponta, havonta fusson le a programunk. A tesztelési folyamat során ezt kihasználva történt a programfuttatás.

A CRON-t lehetőség van grafikus felülettel, illetve parancssorból használni. A grafikus felület telepítéséhez egy parancsablak megnyitása után a következő utasítást kell megadnunk a Raspbianra: „*sudo apt-get install gnomeschedule*”. A telepítés befejeztével a program *ScheduLED Tasks* néven érhető el a rendszeren.

A nem grafikus felületű CRON-t a „*crontab -e*” paranccsal hívhatjuk meg a Raspbian termináljában. A CRON első megnyitásakor a rendszer 3 szövegszerkesztési lehetőséget ajánl fel, amely közül a nano-t érdemes választani. Ezután megadhatjuk mely programot szeretnénk futtatni és mikor, illetve akár több program együttes futtatására is lehetőség van.

Ebben az esetben a megírt program futtatása történt minden 5. percben, illetve az aktuális dátum és időpont került hozzáfűzésre.

A tesztelés során az alábbi parancs került a CRON-ba:

```
* /5 * * * * sudo date >> result.txt
```

```
* /5 * * * * sudo python /home/pi/AdafruitDHT.py 11 22
```

```
>> result.txt
```

Az alap utasítás 5 * karakterrel kezdődik majd az utasítás következik az alábbi formában: * * * * * *parancs*

- Első * karakter: perc (0-59)
- Második * karakter: óra (0-23)
- Harmadik * karakter: hónap adott napja (1-31)
- Negyedik * karakter: hónap (1-12)
- Ötödik * karakter: a hét napja (0-7)

Az első parancs 5 percnként fut le, rendszergazdai jogosultsággal és az aktuális dátumot fűzi hozzá a *result.txt* tartalmához, így hitelesítve az időpontot.

A második parancs szintén 5 percnként fut le, rendszergazdai jogosultsággal és a megírt programot futtatja le a szükséges parancssori argumentumokkal, majd a kimenetet hozzáfűzi a *result.txt* tartalmához, így megkapva a hőmérsékleti adatokat, ahogy a 6. ábrán látható.

```
pi@raspberrypi: ~
GNU nano 2.7.4 File: /tmp/crontab.qEqQIf/crontab
*/5 * * * * sudo date >> result.txt
*/5 * * * * sudo python /home/pi/AdafruitDHT.py 11 22 >> result.txt

pi@raspberrypi: ~
GNU nano 2.7.4 File: result.txt
Thu Apr 5 23:10:01 CEST 2018
Temperature: 28.0 celsius Humidity: 24.0%
Todays high is: 20 celsius
Todays low is: 8 celsius
Thu Apr 5 23:15:01 CEST 2018
Temperature: 28.0 celsius Humidity: 23.0%
Todays high is: 20 celsius
Todays low is: 8 celsius
Thu Apr 5 23:20:01 CEST 2018
Temperature: 29.0 celsius Humidity: 22.0%
Todays high is: 20 celsius
Todays low is: 8 celsius
```

6. ábra A CRON eredményei

A projekt másik fontos részének, a LED szalagnak a tesztelése egy mintaprogram segítségével történt, amely során minden LED a programnak megírt utasításoknak megfelelően világított, működött.

VI. AZ ESZKÖZ MŰKÖDÉSI ELVE, PROGRAM MŰKÖDÉSE

Ebben a részben megismerkedünk az eszköz működési elvével, és a Raspberry-n futtatott program működési elvével.

A Raspberry előnye, hogy nem feltétlen szükséges képernyő, illetve beviteli eszközök használata. A hálózatunkra

csatlakozva könnyen kapcsolatot tudunk teremteni számítógépünk és a Raspberry között.

Ehhez egy úgynevezett *PuTTY* [4] program szükséges, amely segítségével SSH-n keresztül, IP cím megadásával csatlakozni tudunk egy terminálon keresztül a Raspbian rendszerünkhöz, így könnyedén kezelhető az eszköz, illetve módosítható a program.

A csatlakozás folyamán szükség lesz egy felhasználói névre, amely a „*pi*” karaktersorozat, illetve első csatlakozáskor az alapértelmezett jelszót kell megadni, amely a „*raspberry*” karaktersorozat. Első csatlakozás után érdemes a jelszót megváltoztatni biztonsági okokból. Ezt egy parancsablak megnyitása után a „*passwd*” kulcsszó megadása után tehetjük meg, amely beírása után meg kell adnunk az aktuális, jelen esetben az alapértelmezett jelszót, majd beírható az új jelszó, illetve egy „*enter*” lenyomásával megerősíthető.

Az alap koncepció a kinti hőmérséklet szerinti megvilágítás volt, mely később ki lett egészítve a már említett DHT 11-es szenzorral.

A program futtatásakor a szoftver „lekéri” a napi maximum és minimum hőmérsékletet a *yahoo* segítségével, majd a DHT 11 segítségével megállapítja az aktuális hőmérsékletet.

Ezután ezen adatok kiértékelése következik, amely során az aktuális értékhez hozzárendeli az adott RGB színek kódját a program. A szoftver működésének folyamatát a 7. ábra mutatja, amely UMLet [5] segítségével készült.



7. ábra Folyamatábra

A program helyes működéséhez szükség van az *Adafruit_Python_DHT* [6] könyvtárra, amely a digitális LEDek programozását könnyíti meg.

Ha esetleg más méretű a doboz vagy kevesebb – több LEDet szeretnénk használni, könnyen testre szabhatjuk programunkat, ugyanis a program elején lévő definíciónál a *LED_COUNT* változót a 12-es értékről átírva megadhatjuk egyéni LED darab számunkat.

A programon számos egyénre szabható lehetőség van még, ugyanis megadhatjuk azt, hogy melyik várost tekintse

tartózkodási helyünknek, illetve azt, hogy Celsiusban vagy Fahrenheitben kapjuk meg a napi értékeket.

A *where text="város, wi"* utasításnál a város szót kell helyettesítenünk a kívánt várossal, így a programunk az itt megadott helység adatait veszi figyelembe.

A második sorban lévő *u = „c”* utasításnál a *c* jelentése a Celsius, ennek elhagyása Fahrenheit formátumot eredményez.

A program két függvénnyel rendelkezik, amelyek felelnek a LED szalagot színeinek megjelenítéséért. Az első függvény – *inside* – az első 6 db LED megvilágításáért felel az aktuális hőmérséklet függvényében, a programban 0-5 közötti számokkal indexeljük, ugyanis az indexelés 0-tól kezdődik és nem 1-től.














A második függvény – *outside* – az utolsó 6 LED, vagyis a 6 és 11-es indexű LEDeket világítja meg a napi maximum hőmérséklet szerint.

A függvényeket tartalmazó utasításokat az Adafruit könyvtár tartalmazza. A függvényekben szereplő utasítások és funkcióik:

- *setPixelColor(uint16_t n, uint32_t c)*: Adott sorszámú pixel színmegadása színkóddal
- *Color(uint8_t r, uint8_t g, uint8_t b)*: Színkód származtatása RGB komponensekből
- *time.sleep(wait_ms/500.0)*: A LEDek közötti színváltás idejének megadása
- *setBrightness(uint8_t b)*: Fényerősség beállítása (ez csak egyszer hívható meg)
- *show(void)*: Megjelenítés az előzőleg beállított adatokkal

Miután programunk rendelkezik a megfelelő hőmérsékleti adatokkal, következő lépésként kiértékeli azokat. *if* elágazások segítségével történik a kiértékelés, amely létrejön mind az aktuális, mind a napi maximum hőmérséklet színénél. Miután az elágazások között a program megtalálta az értékek megfelelő intervallumot, beállítja a LED szalag adott LEDjeit a megfelelő, intervallumhoz hozzárendelt színekre. A meghatározott színeket a már korábban említett *show(void)* utasítással kapja meg a LED szalag.







A program megírásakor létrejött egy intervallum táblázat, ami kimutatja, hogy milyen hőmérsékleti értékek között milyen színben kell világítania a LED szalagnak. A színek meghatározása a hőmérséklet szerint történtek, illetve minden szín tesztelésre került, abból a szempontból, hogy a LED szalag képes-e megvilágítani az adott színt, amely az adott hőmérsékletre lett rendelve. A program írása közben meghatározott intervallumok, valamint a LED szalag által támogatott színek kódok a 8. ábrán láthatóak.

temp	color	color code
>40		255 0 8
[35:39]		255 70 0
[30:34]		255 161 0
[25:29]		255 234 0
[20:24]		25 255 0
[15:19]		0 255 64
[10:14]		0 255 149
[5:9]		0 255 215
[0:4]		0 166 255
[-5:-1]		0 3 255
[-10:-6]		104 35 173
[-15:-11]		64 7 102
<-16		27 3 43

8. ábra Intervallumok és színek kódok

Az intervallumok és a hozzá rendelt színek kódok megértését az alábbi táblázat foglalja magába, amely megmutatja, hogy egy véletlenszerű hőmérséklet esetén mely színek kódokat alkalmazza a program a LED szalagra.

2. TÁBLÁZAT MAGYARÁZAT

Aktuális hőmérséklet	Szín	Színkód
24°		R: 25 G: 255 B:0
36°		R: 255 G: 70 B: 0
2°		R: 0 G: 166 B: 255
Napi maximum		
-5°		R: 0 G: 3 B: 255
nagyobb mint 40°		R: 255 G: 0 B: 8
12°		R: 0 G: 255 B: 149

A teljes egészében összeszerelt és üzembe helyezett eszköz megfelelően és stabilan működött. A 9. ábra mutatja az eszköz működését, megvilágítását kész állapotban.



9. ábra Az elkészült projekt

VII. ÖSSZEZÉS

A projekt összességében a megtervezettek szerint készült el, mind fizikailag, mind elektronikailag.

A Raspberry Pi 3 model B segítségével létrejött az elképzelés és sikeres lett a címezhető LED kihasználtsága. Az eszköz könnyen automatizálható a már említett CRON segítségével, így személyre szabottan lehetséges a program futtatásának időpontja, amelyet használva nincs szükség a program folyamatos manuális futtatására.

Esetleges fejlesztési lehetőség lehet például egy hálózaton több kliens időjárás állomás, vagy olyan helyzetekben, amikor a külső hőmérséklet nem számít, a DHT 11 segítségével bizonyos helyiségekben vagy tároló helyeken hőmérsékleti statisztika készítése, esetleg kritikus helyzetekben (adott helyiség hőmérséklete túl magas, vagy túl alacsony) értesítés küldése hálózaton keresztül a megfelelő személynek.

VIII. KÖSZÖNETNYÍLVÁNÍTÁS

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 köszönetemet nyilvánítani Dr. Godó Zoltánnak az Informatikai Kar Információ Technológia Tanszék adjunktusának a labor rendelkezésre bocsájtását.

IX. HIVATKOZÁSOK

- [1] Raspbian, (2018, Március). [Online]. Available: <https://www.raspberrypi.org/downloads/raspbian/>
- [2] Ubuntu MATE 14.04 LTS, (2014, November 11). [Online]. Available: <https://ubuntu-mate.org/blog/ubuntu-mate-trusty-final-release/>
- [3] fritzing, (2016, Június 2). [Online]. Available: <http://fritzing.org/download/>
- [4] PuTTY, (2017, Június 08). [Online]. Available: <https://www.putty.org/>
- [5] UMLet, (2017). [Online]. Available: <http://www.umlet.com/>
- [6] Adafruit_Python_DHT, (2017, Március 1). [Online]. Available: https://github.com/adafruit/Adafruit_Python_DHT
- [7] CRON, (2017, Március 1). [Online]. Available: <https://www.rackhost.hu/tudasbazis/webtarhely/cron/>
- [8] 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.