Beltéri hőszabályzó megvalósítása

ESP-32 modullal

Varga Rajmond

Automatizálás IV-B

Tartalom

[**1.Bevezető** 2](#_Toc92133817)

[**2.Az ESP-32-ről:** 2](#_Toc92133818)

[**3. A hőszenzorról:** 3](#_Toc92133819)

[**4. A relé megható áramköre:** 4](#_Toc92133820)

[**5. Alkatrészek bekötése:** 4](#_Toc92133821)

[**6. A weboldal megjelenése és használata:** 5](#_Toc92133822)

[**7. A program ismertetése:** 7](#_Toc92133823)

[**8. A weboldal elérése:** 8](#_Toc92133824)

[**9. Könyvészet:** 9](#_Toc92133825)

# **1.Bevezető**

A projekt célja egy on/of típusú hőszabályzó megvalósítása ESP-32 modullal. A szabályzó egy relé segítségével fog ki/bekapcsolgatni egy gázkazánt.

A szabályzó egy ESP-32-DEVKITC-V4 típusú modulból, egy DHT22 típusú digitális hőszenzorból, valamint egy TM1637 típusú 7-szegmenses kijelzőből fog állni. Az ESP digitális kimenete egy tranzisztort fog meghajtani, ami majd meghajtja a relét.

Ez ESP-n egy web szerver fog futni, és ott lehet beállítani a hőmérsékleti értékeket. A weboldalon amellett, hogy be lehet állítani a jelenlegi hőmérsékletet, azon kívül még kijelzésre kerül a hőmérséklet és a páratartalom is. Programot is állíthatunk be, ahol megadhatjuk, hogy a nap melyik időszakában mennyi legyen az elvárt hőmérséklet.

A hétszegmenses kijelzőn felváltva láthatjuk majd az órát, a mért hőmérsékletet és páratartalmat, valamint a beállított hőmérsékletet.

# **A képen szöveg, elektronika, áramkör látható Automatikusan generált leírás2.Az ESP-32-ről:**

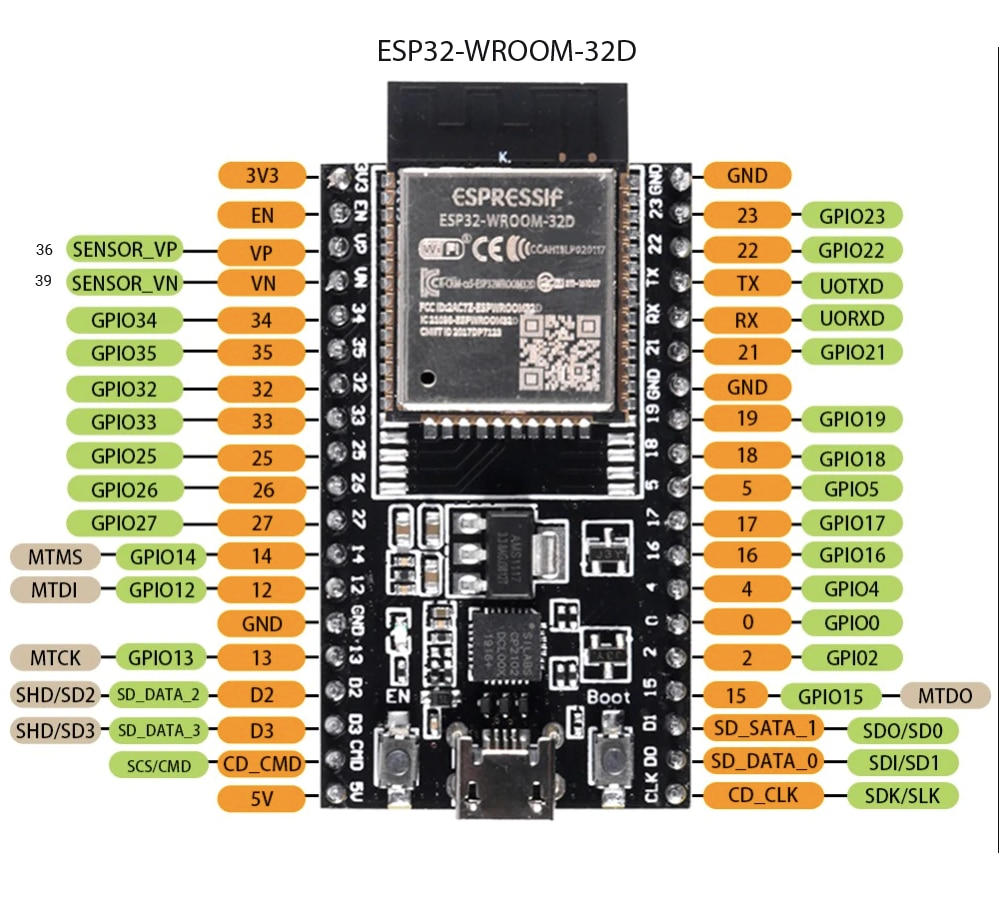
Az ESP-32-DEVKITC-V4 modulon egy ESP-32-WROOM-32D típusú chip található, egy Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), valamint rendelkezik beépített WiFi és Bluetooth modullal is. 4 MB Flash memória található rajta.

Micro USB csatlakozó segítségével táplálható be,van benne beépített feszültségstabilizátor és ugyanezen csatlakozón keresztül programozható.

1. ábra ESP-32

A lapkán egy 40 MHz-es kristály található, valamint -40°C és +85 °C között működtethető.

A modul 38 pinnel rendelkezik, lábkiosztása a [2. ábrán](#a2) található:



. ábra ESP-32-DEVKITC-32D lábkiosztás

# **3. A hőszenzorról:**

 A hőszenzor egy DHT22 típusú hőmérséklet és páratartalom mérő, amely AM2302 néven is ismert. 3,3-5 v tápfeszültséggel lehet betáplálni, és -40°C és +80 °C között mér hőmérsékletet, valamint 0 és 100% páratartalom között. A tradukor benne egy polimer kapacitor.

A szenzor felbontása 16 bit, a mintavételi periódusa nagyobb, mint két másodperc, a hőmérséklet pontossága +-0,5°C, a páratartalom pontossága pedig +-5%.

Három lába van ennek a típusú szenzornak (van olyan változata, ahol 4), egy táp, egy föld és egy adat. A kommunikációs protokollról, amit használ a [[1]](#h1) weboldalon lehet bővebb információkat szerezni.

. ábra DHT22

4. A hétszegmenses kijelző:

A hétszegmenses kijelző egy TM1637 volt. Ez egy egyszerűbb kijelző, 4 számjegyet tud kijelezni, és egy kettőspontot a második és a harmadik közé. A lábkiosztása a [4. ábrán](#a4) látható.

A képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírás

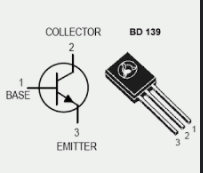
. ábra TM1637

. ábra TM1637 lábkiosztás

A kijelzőnek 4 lába van, egy órajel, egy digitális ki-bemenet, egy tápbemenet és egy föld. 3,3-5V feszültséggel táplálható be, a fényereje állítható, valamint, -10°C és +80 °C hőmérséklet értékek között üzemeltethető. A kommunikációs protokoll, amit használ hasonló az I2C protokollhoz és a [[2]](#h2) weboldalon lehet róla többet megtudni.

# **4. A relé megható áramköre:**

A képen szöveg, térkép, különféle, különböző látható

Automatikusan generált leírás A tranzisztor egy BD 139 tranzisztor volt, a [7. ábrán](#a7) látható kapcsolásban:

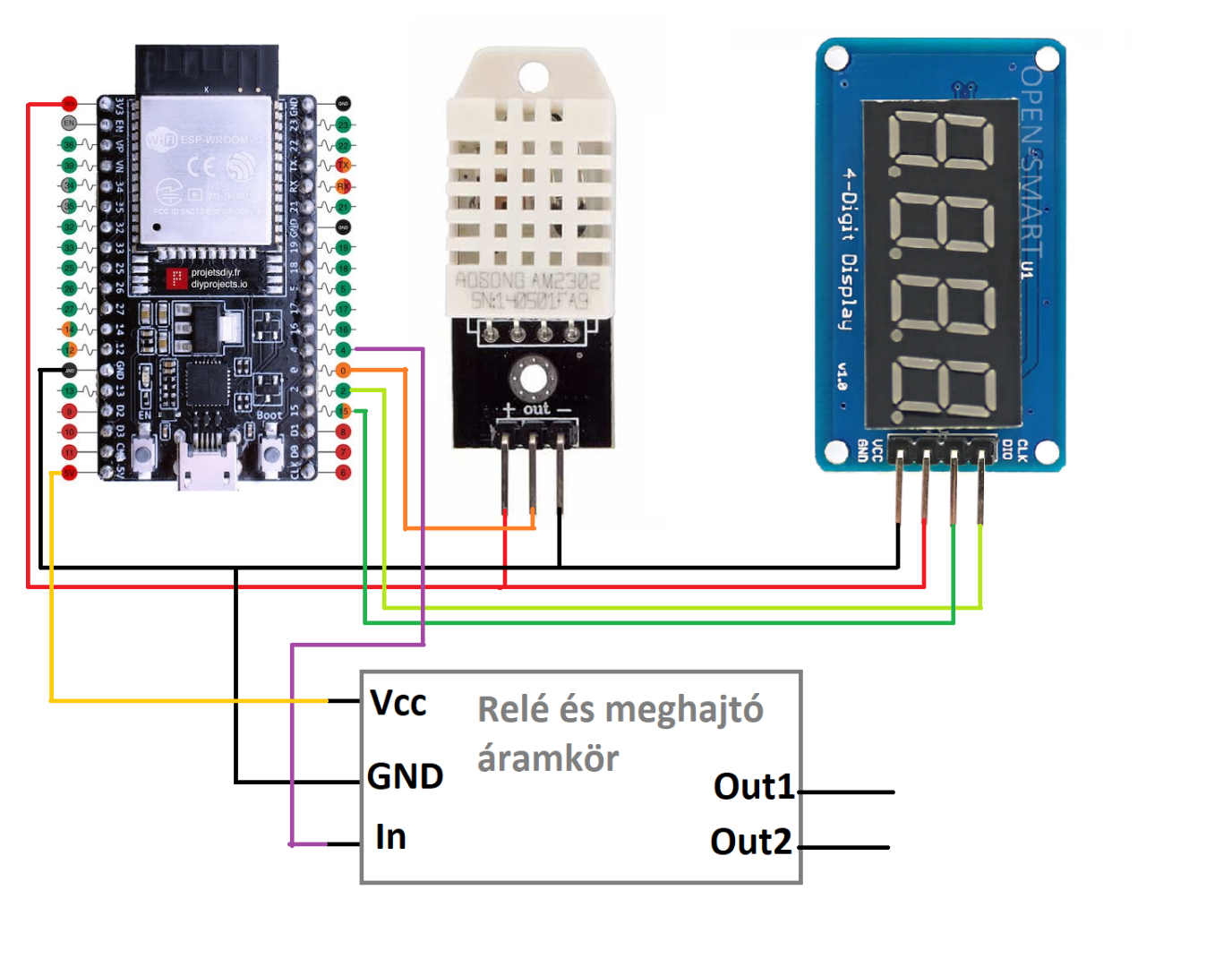
. ábra BD139

Az áramkör egyszerű, amint digitális 1 érkezik a bemeneten, a tranzisztor zár és áram folyik át a relé tekercsein. Egy LED-et is beiktattam, hogy látni lehessen azt, ha a kapcsoló zár, így megkönnyítve a működés nyomon követését.

. ábra meghajtó áramkör

# **5. Alkatrészek bekötése:**

Az alkatrészeket a [8. ábra](#a8) szerint kell összekötni. Az ESP használt lábai módosíthatóak, de akkor fontos, hogy a kódot is módosítani kell.

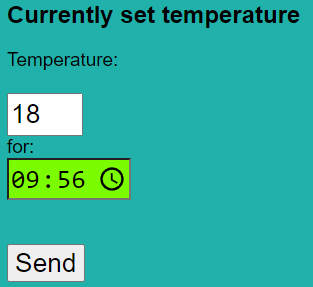


. ábra Alkatrészek bekötése

# **A képen szöveg látható Automatikusan generált leírás6. A weboldal megjelenése és használata:**

A weboldal nagyon egyszerű, HTML nyelven van megírva.

Legfelül kijelzi a jelenlegi hőmérsékletet és páratartalmat, valamint ezek alatt van egy Refresh nevű gomb, amivel frissíthetőek a kiírt adatok. Ez alatt kijelzi azt, hogy a fűtés aktuálisan ki vagy be van e kapcsolva. Ez a [9. ábrán](#a9) tekinthető meg.

 Ezt követően egy olyan mező következik, ahol megadhatom, hogy egy adott időpontig mennyi legyen az elvárt hőmérséklet. Amennyiben ez be van állítva az idő mező háttere zöldre vált, valamint ez felülírja a későbbiekben beírható programot is. Ez legtöbb 24 órát tarthat, ugyanis he a beállított időpont korábbi a jelenleginél akkor a következő nap beállított órájáig fogja tartani a hőmérsékletet. Az alapértelmezetten kijelzett idő itt az ESP-ben jelenleg eltárolt idő, de amennyiben ezt az értéket módosítjuk, az intervallum lejártáig a kívánt idő fog megjelenni. A hőmérséklet alapértelmezetten beírt értéke 18, de módisításkor ez is megváltozik az intervallum lejártáig.

. ábra Weboldal 1

A beállított adatokat itt is, mint mindenhol máshol is a Send gomb megnyomásával küldöm el az ESP-nek.

10. ábra Weboldal 2

A képen szöveg látható

Automatikusan generált leírás Az itt leírtak a [10. ábrán](#a10) tekinthetőek meg.

Ezt követően egy olyan rész következik, ahol az ESP belső óráját állíthatjuk be. Neki tudnia kell a pontos időt, és ezt le is kéri interneten keresztül, de ha nem áll rendelkezésre internetkapcsolat akkor a belső órája itt manuálisan is beállítható. De attól függetlenül, hogy ezt beállítjuk az ESP meg fogja próbálni időnként lekérni a pontos időt és frissíteni fogja ezt az értéket. A mezőbe alapértelmezetten beírva az időt látjuk, ami az ESP-ben jelenleg el van tárolva. Ez a felület a [11. ábrán](#a11) tekinthető meg.

. ábra Weboldal 3

Az utolsó rész az a szakasz, ahol az előre beállított napi hőmérsékleteket adhatjuk meg.

Ez úgy működik, hogy megadhatunk 4 hőmérsékleti értéket 4 időintervallummal. Az ezen intervallumok fontos, hogy 1 napon belül legyenek. Ezeknek a kezdeti és a végső időpontját kell megadni, valamint az azon intervallum idő alatt elvárt hőmérsékletet. Ha 2 ilyen időintervallum egymásra csúszik, akkor az első felülírja az utána következőt. A jelenleg aktív intervallum itt is zöldre vált.

Azon időkre, amikre nem vonatkozik egy intervallum sem, egy megadható alapértelmezett (default) értékre állítja be a szabályzó a hőmérsékletet.

Ennek a megjelenítését a [12.](#a12) és [13. ábrán](#a13) láthatjuk.

A képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leírás

12. ábra Weboldal 4

13. ábra Weboldal 5

# **7. A program ismertetése:**

A programot Arduino IDE környezetben készítettem el, és a kliens-szerver kommunikáció megvalósításának ezen módját a [[3]](#h3) forrásból ismertem meg és használtam fel, módosításokkal.

A probléma megoldásához 4 könyvtárat használtam fel:

1. WiFi.h a kommunikáció megvalósításához.
2. ESP32Time.h az idő számontartásához és lekezeléséhez, aminek az elérhetősége és leírása a [[4]](#h4) linken van.
3. TM1637Display.h az 7 szegmenses kijelzővel való kommunikáció lekezeléséhez, elérhető és a leírása megtalálható az [[5]](#h5) linken.
4. DHT.h ami a DHT22 hőszenzorral való kommunikációt oldja meg. Elérhetősége: [[6]](#h6).

Ezeket követően elnevezem a lapka kimeneteit. Fontos megjegyezni, hogy az általam választott lapka kimeneteinek elnevezése a lapkán nem egyezik meg a kimenetek elnevezéseivel a programozói környezetben. Ezek az elnevezések megtalálhatóak a [[7]](#h7) linken.

Ezeket követően beállítom a WiFi adatait, valamint beállítom a webszerver paramétereit és deklarálom a hozzá szükséges változókat, mint például a header, amibe érkezni fognak az adatok a klienstől.

Megadom az NTP Server elérhetőségét is, ahonnan az ESP lekéri majd a pontos időt, valamint beállítom az ezzel kapcsolatos adatokat, mint például az időzóna.

Itt létrehozom a tömböket és a változókat, amikben az időket, hőmérsékleteket, szabályzó paramétereit és egyéb adatokat tárolok, valamint jelző Flag-eket, amikkel nyomon követem, hogy milyen parancsot hajtok végre. Itt példányosítom az objektumokat az idő és a hőszenzor számára.

A **setup** részben először elindítom a soros kommunikációt a számítógép fele, hogy adatokat küldhessek hibajavítási célokból, inicializálom a kijelző paramétereit, csatlakozok a wifi hálózatra, lekérem és beállítom a pontos időt és végül elindítom a web szervert. Amennyiben nem sikerül itt lekérni az időt beállítok egy kezdeti értéket.

Ezt követően a **loop** szekció következik.

Itt először bekérem a hőmérsékletet és páratartalmat a readTemperature() és readHumidity() függvényekkel, majd lekérem az órát és a percet, amit azután formázok a kívánt formátumba és stringként dolgozok velük.

Amennyiben nem tartok egy adott hőmérsékletet egy megadott óráig, hanem csak a napi programot hajtom végre, az időnek, amit kijelzek ott, ahol ezt a várakozást beállíthatom, beállítom a jelenlegi időt.

Ezt követően a kijelzést oldom meg egy switch-case logika segítségével úgy, hogy 4 másodpercenként váltogatok az aktuális idő, a mért hőmérséklet, mért páratartalom és elvárt hőmérséklet között.

A várakozást úgy oldom meg hogy lekérem a jelenlegi időt másodpercben és ahhoz minden kilégzéskor hozzáadok 4 másodpercet, ha a jelenlegi idő eléri ezt az összeget ismét kijelzek és hozzáadok 4 másodpercet a jelenlegi időhöz.

Ezt követően megnézem, hogy van e kiszolgálandó kliens és ha van akkor lekezelem a kommunikációt és betöltöm a kapott adatokat a headerbe, ami egy string. Ha megérkezett az egész header alkezdhetem feldolgozni.

Mielőtt elkezdem keresni az adatokat a headerben elmentem a jelenlegi elvárt hőmérsékletet és időtartamot, hogy amennyiben nem kapok hasznos adatot akkor is legyen mit kijelezni.

Az adatokat egy sor if-else segítségével nyerem ki a headerből úgy, hogy minden utasításnál más változó nevek szerepelnek benne és ezeket keresem.

Sorban megyek végig a lehetséges utasításokon, és ha megtalálom valamelyik jellegzetes változó nevét akkor kimentem a hasznos adatokat és beállítok egy Flag nevű változót, ami megmondja, hogy milyen parancs jött. Az is egy külön érték a Flag-ben, ha nem jött hasznos adat.

Az időket mindig stringekben, a hőmérsékleteket intekben (int tömbökben) tárolom.

Ezután elküldöm a kliensnek a kijelzendő weboldalt a frissített értékekkel.

A weboldal egyszerű kérdőívként (form típus) kér be mindent majd a send gombokra kattintással küldi el őket a servernek.

Ezt követően kiüresítem a headert és zárom a kommunikációt a klienssel.

A következő lépés az, hogy egy switch-case logikával megnézem, hogy milyen utasítás érkezett. Ha a Flag értéke 1, akkor az utasítás az, hogy tartsa a hőmérsékletet egy meghatározott ideig. Itt az időt átalakítom int típussá és átszámolom másodpercbe. Ha a jelenlegi idő későbbi, mint az óra és perc, ami érkezett, hogy várni kell addig akkor másnap addig vár. Egy if-else logika alakítja át az időt. Ezt követően megnézem, hogy a kiszámolt időpont elérkezett e már, ha nem akkor a változót, ami azt jelzi, hogy hőmérsékletet tartsak egyre állítom, ha pedig igen akkor nullárra és a szabályzás a szerint fog történni, hogy mi a napi program.

Ha a Flag változó 2 akkor az azt jelenti, hogy az időt kell beállítani, és az állítóik be.

Ha ez megvan akkor megnézem, hogy mi kell legyen az elvárt hőmérséklet, ha hőmérsékletet kell tartani egy idő lejártáig akkor az lesz, ha nem akkor egy másik if-else logika eldönti, hogy melyik periódusból válasszak időt, esetlegesen az alapértelmezetten beállított időt ha nem vagyok egy periódusban sem.

Az is itt dől el, hogy a weboldalon melyik cella háttere legyen zöld.

Végül már csak az egyszerű hiszterézises on-off szabályzó van hátra és hogy a Flag változó értékét visszaállítsuk nullára.

# **8. A weboldal elérése:**

A weboldal két módon is elérhető. Az első az, hogy a router által az ESP-nek adott lokális IP címet beírjuk a böngészőbe. A második egy alkalmazás, ami a [[8]](#h8) linken érhető el, és ott nagyon egyszerűen létrehozható egy alkalmazás, ami egy QR kóddal fel is telepíthető egy okostelefonra. Csak annyit kell tenni, hogy be kell húzni egy WebViewer interfészt az alkalmazásba és megnyomni, hogy build. Ott meg fog jelenni a QR kód.

# **9. Könyvészet:**

[1] <http://www.ocfreaks.com/basics-interfacing-dht11-dht22-humidity-temperature-sensor-mcu/>

[2] <https://circuit4us.medium.com/write-spin-code-for-tm1637-and-tm1638-part-i-f5568d2fe05b>

[3] Rui Santos and Sara Santos: Learn ESP32 with Arduino IDE

[4] <https://github.com/fbiego/ESP32Time>

[5] <https://github.com/avishorp/TM1637/blob/master/TM1637Display.h>

[6] <https://github.com/adafruit/DHT-sensor-library>

[7] <https://github.com/wditch/plcLib/issues/13>

[8] <http://ai2.appinventor.mit.edu/>