WiFi Thermostat for Central Heating

Autor: Petru-Alexandru Mateescu

Introducere

Scopul proiectului este controlarea unei centrale termice prin intermediul smartphone-ului. Acest lucru se poate realiza cu o aplicație software ce transmite comenzi către un microcontroler prin intermediul datelor mobile sau a rețelei wifi. În cazul în care nu se dorește folosirea telefonului se utilizează postul de comandă local ce dispune de un buton ce declanșează pornirea/oprirea centralei și două butoane ce ajustează temperatura dorită. Acesta dispune și de un display pe care se afișează data și ora, temperatura dorită, temperatura și umiditatea ambientale, stadiul de funcționare al centralei termice (pornit/oprit) și o pictogramă ce evidențiază modul de funcționare automat al termostatului.

Ideea de la care am pornit a fost necesitatea înlocuirii unui termostat inteligent ce nu a mai funcționat și care dispunea de aceleași capabilități de controloare a centralei termice prin intermediul smartphone-ului.

Utilitatea proiectului constă în monitorizarea facilă a temperaturii și umidității și a stadiului de funcționare al centralei termice. De asemenea, se înlocuiește necesitatea pornirii manuale a acesteia prin funcționarea automată a termostatului.

Descriere generală

La prima punere în funcțiune, microcontrolerul se conectează la internet la o rețea de wifi predefinită și afișează un mesaj de confirmare către monitorul serial al calculatorului (dacă acesta este conectat), iar apoi afișează un mesaj de bun venit pe display-ul local și pe display-ul telefonului. Sunt citite temperatura și umiditatea ambientale de la senzorul DHT11 și sunt afișate pe ambele display-uri. Temperatura dorită prestabilită este de 25°C ce se poate modifica ulterior fie de la cele doua butoane de la postul de comandă local fie de la slider-ul de pe aplicația mobilă.

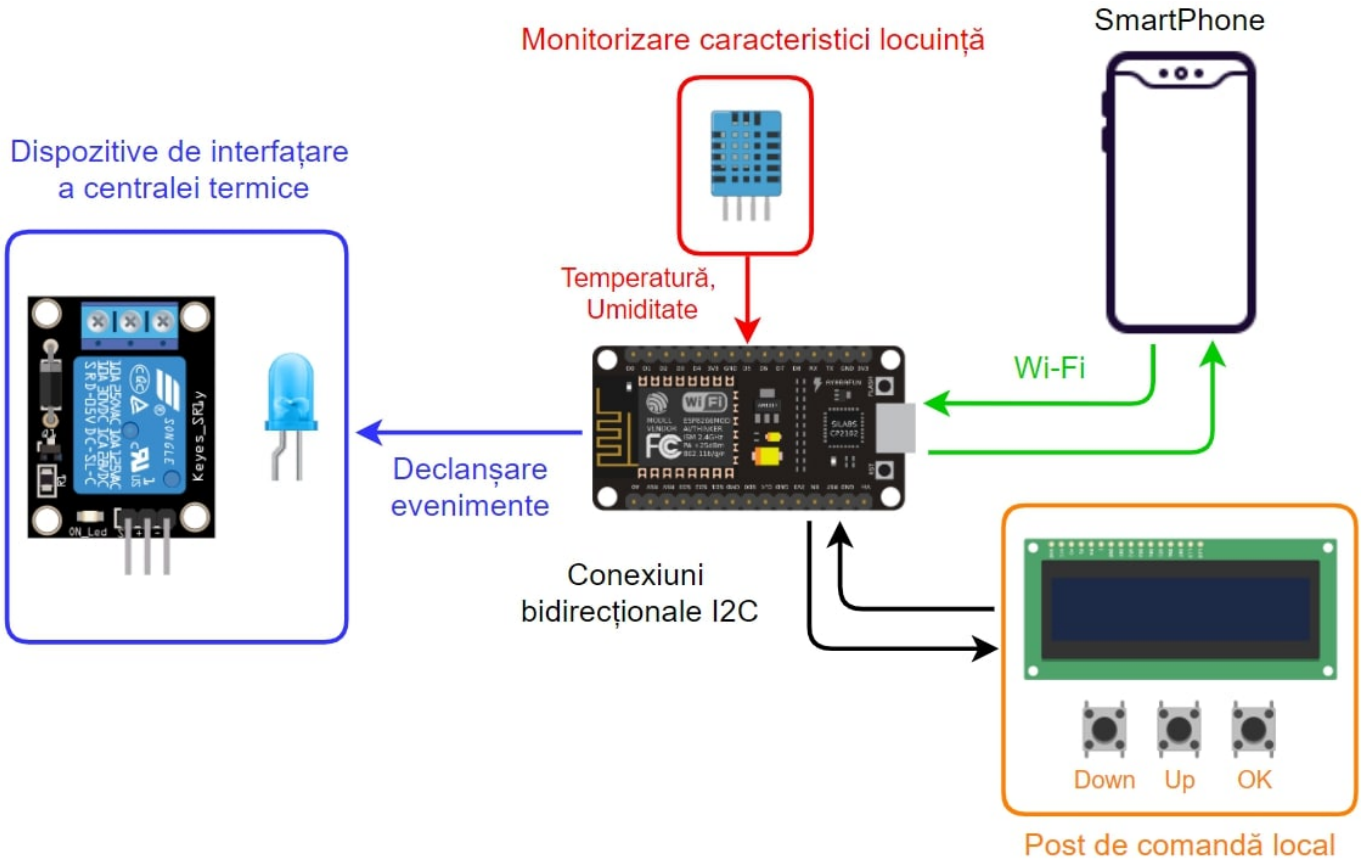
Dacă funcția automată a termostatului nu este activată, atunci, în orice moment, se poate porni sau opri centrala termică fie de la butonul fizic de comanda fie de la butonul aplicației software de pe telefon pană se ajunge la temperatura dorită.

Funcția automată va porni centrala termică daca temperatura ambientală a scazut cu o toleranța prestabilita (2°C) sub temperatura dorită și o va opri daca temperatura ambientală a crescut cu toleranța prestabilită peste temperatura dorită. În aceste momente se va afișa un mesaj sugestiv pe ecranul smartphone-ului (AUTO ON/AUTO OFF) și un simbol de lacăt pe ecranul postului de comandă local. Pentru a anula funcționalitatea automată a termostatului se ajustează temperatura dorită.

Pentru pornirea centralei termice microcontrolerul va cupla un releu, va aprinde un LED și va afișa un mesaj corespunzator pe display-ul local și pe display-ul aplicației smartphone-ului (ON). Pentru oprirea centralei termice se declanșeaza evenimentele opuse: decuplează releul, stinge LED-ul și afișează OFF pe ecrane.

Data și ora se afișează doar pe ecranul postului de comandă local pentru că pe telefon sunt afișate constant în afara aplicației software. Pentru ca acestea să fie setate corect folosesc biblioteca NTPtimeESP.h ce se conectează la serverul local Romaniei prin wifi. Astfel exclud necesitatea folosirii unui modul hardware RTC.

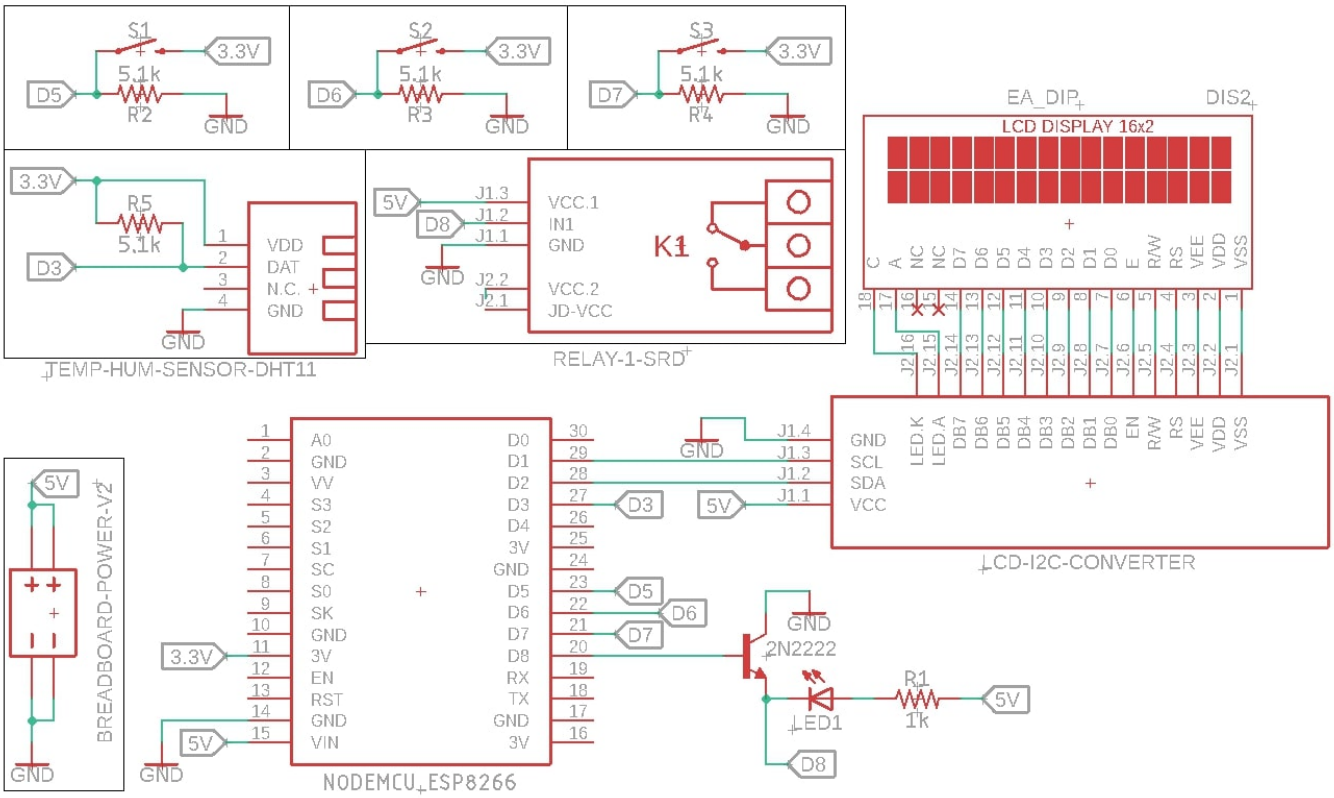
Schema bloc



Hardware Design
Lista de componente

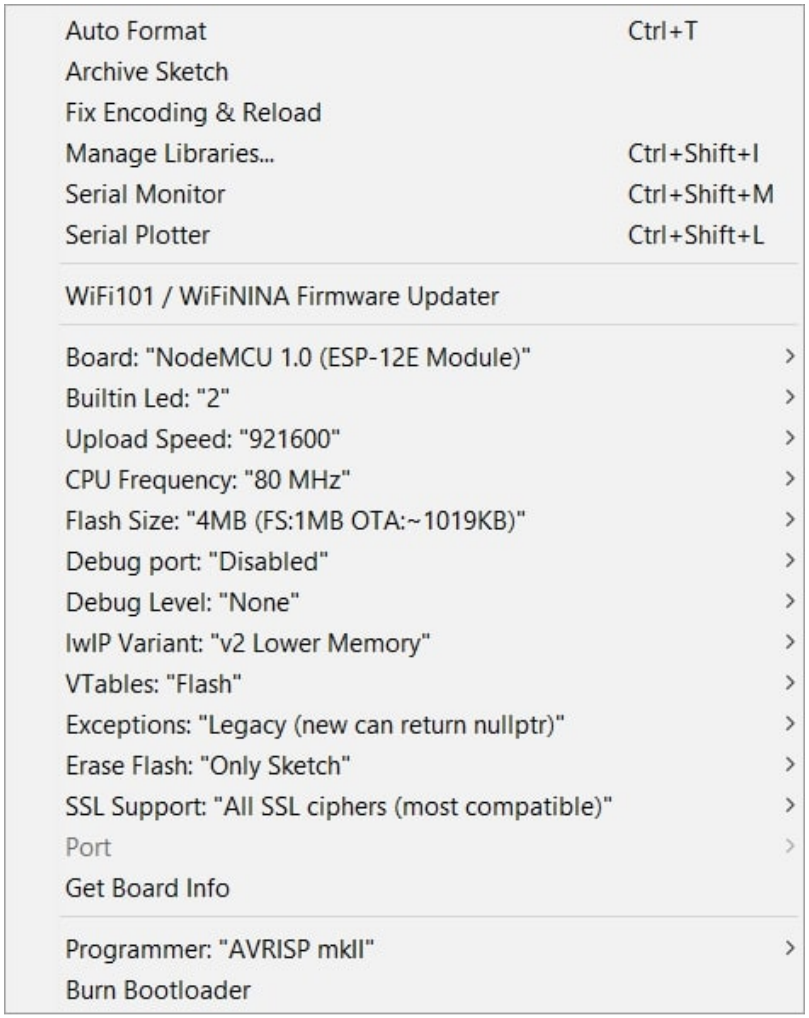
- NodeMCU ESP8266
- Ecran LCD I2C 16x2
- 3 Butoane
- LED 5mm (orice culoare)
- Rezistor 1kΩ
- 4 Rezistori de 5k1Ω
- Tranzistor NPN
- Releu
- Senzor DHT11
- Breadboard
- Fire de legatură
- Sursă de alimentare

Schema electrică



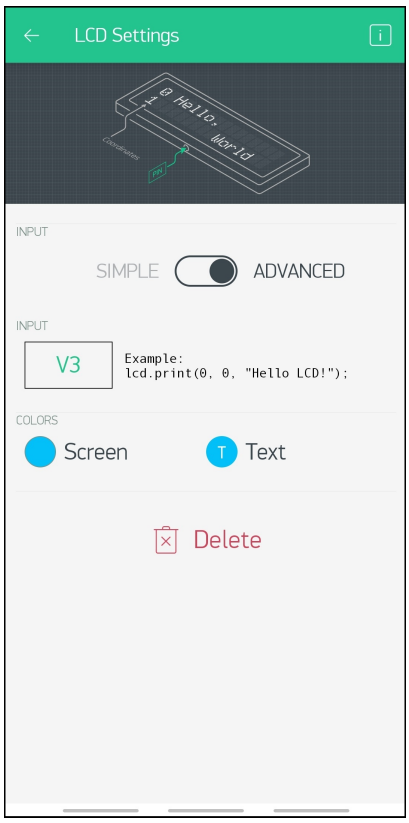
Software Design

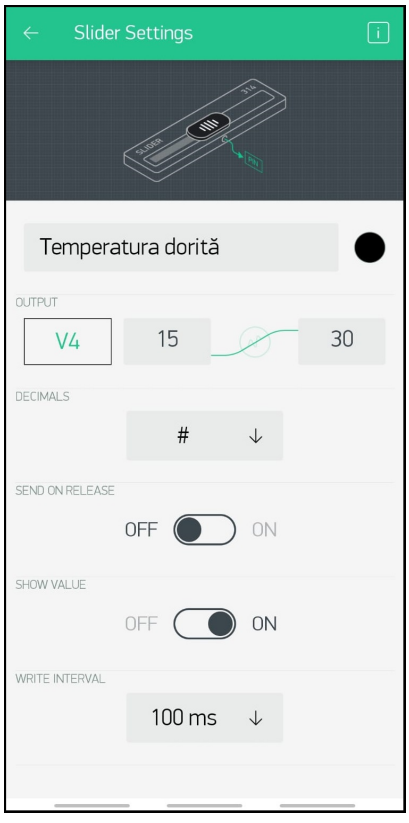
Mediul de programare ales este platforma Arduino IDE cu setările aferente microcontroller-ului NodeMCU ESP8266.



Aplicația aleasă pentru comunicarea dintre µc și smartphone este Blynk, disponibilă atât pe Android cât și pe iOS. Aceasta are o interfață simplă și este ușor de folosit. În cadrul proiectului, aplicația conține trei elemente:

- un ecran pe care sunt afișate temperatura și umiditatea ambientale, stadiul de funcționare al centralei termice (ON / OFF), temperatura dorită și un martor ce semnifică funcționarea automată a termostatului;
- un buton de comandă ce pornește sau oprește centrala termică în modul manual;
- un slider pentru ajustarea temperaturii dorite.





Imaginile anterioare descriu setarile aferente fiecărui element:

- ecranul este setat pe modul advanced, pinul de comunicare cu µc este pinul virtual V3 (folosit și în codul sursă), culoarea de fundal este albastru deschis, iar culoarea textului este albă;
- butonul este denumit „Comandă centrala termică”, pinul de comunicare este GPIO14 (D5), modul de tip push, denumirea „Trimite comanda” este afișată cât timp acesta nu este apăsat, iar denumirea „Comandă trimisă” este afișată cât timp este apăsat, dimensiunea fontului este medie și culoarea acestuia este neagră;

- sliderul este denumit „Temperatura dorită”, are culoarea neagră, pinul de comunicare este pinul virtual V4, valorile exprimate sunt de tip întreg (fără zecimale) cuprinse în intervalul 15-30 (°C), funcția send on release este dezactivată (se trimit valorile la interval de 100ms) și valoarea setată este afișată.

Pentru comunicarea dintre NodeMCU și aplicația Blynk folosesc bibliotecile `Blynk.h`, `ESP8266WiFi.h`, `BlynkSimpleEsp8266.h`, dar și un cod de autorizare (același și în codul sursă și în aplicația software) și elementele de conectare ale µc cu adresa wifi (id-ul și parola):

```
#include <Blynk.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "Introdu_tokenul_de_autorizare";
char ssid[] = "Introdu_ssid-ul_wifi-ului";
char password[] = "Introdu_parola";

Blynk.begin(auth, ssid, password);  // inițializare
Blynk.run();  // menținere conexiune
```

Pentru a exclude necesitatea folosirii unui modul hardware RTC am folosit biblioteca `NTPtimeESP.h`. Aceasta declară o structură (strDateTime) ce stochează datele obținute printr-o cerere de tip GET pe un server ce se ocupă cu un RTC rezervat pentru Romania. Efectul obținut este că se afișează ora exactă în orice moment:

```
#include <NTPtimeESP.h>
NTPtime NTPro("ro.pool.ntp.org");
strDateTime dateTime;
dateTime = NTPro.getNTPtime(1.0, 1);

struct strDateTime {
    byte hour;
    byte minute;
    byte second;
    int year;
    byte month;
    byte day;
    byte dayofWeek;
    boolean valid;
}
```

Pentru declararea și utilizarea senzorului DHT11 am folosit biblioteca `SimpleDHT.h`:

```
#include <SimpleDHT.h>
SimpleDHT11 dht11;  // strunctura necesară comunicării cu senzorul
const int dht_pin = D3;  // declar pinul pe care se primesc datele
dht11.read(dht_pin, &temperature, &humidity, NULL);
```

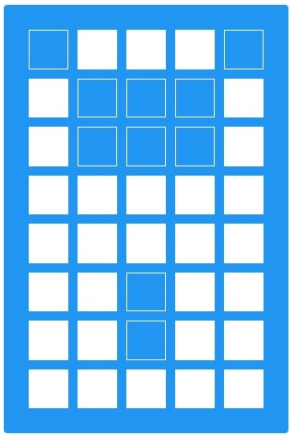
Pentru utilizarea ecranelor LCD (fizic și virtual din aplicația mobilă) și a portului serial (comunicarea dintre µc și calculator) am folosit bibliotecile `Wire.h` si `LiquidCrystal_I2C.h`. Tot aici îmi creez un

caracter special sub formă de lacăt pe care îl afișez când modul automat este în funcțiune:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define BLYNK_PRINT Serial

WidgetLCD blynk_lcd(V3);  // folosesc pinul virtual V3 pentru ecran
LiquidCrystal_I2C lcd (0x27, 16, 2);  // adresa ecranului fizic este 0x27
Serial.begin(115200);

Wire.begin(D2, D1);  // declar pinii pentru protocolul I2C
lcd.begin();
lcd.createChar(0, locked);  // creez un caracter de forma unui lacăt
```



```
byte customChar[] =
{ B01110,
  B10001,
  B10001,
  B11111,
  B11111,
  B11011,
  B11011,
  B11111
};
```

Pentru afișarea informațiilor pe ecrane și monitorul serial mă folosesc de următoarele funcții:

```
lcd.clear();                    blynk_lcd.clear();
lcd.home();
lcd.setCursor(int x, int y);
lcd.print(char *string);        blynk_lcd.print(int x, int y, char *string);
```

Majoritatea funcțiilor din coodul sursă sunt pentru afișarea diverselor date către ecranul LCD fizic, ecranul LCD virtual al aplicației și către monitorul serial al calculatorului (când acesta este conectat):

- `void printWelcomeMsgToLcd()` afișează un mesaj de bun venit, doar după ce µc NodeMCU s-a conectat cu succes la rețeaua de internet furnizată, atât pe ecranul fizic cât și pe cel virtual;
- `void setup()` se ocupă de conectarea µc la rețeaua de internet, afișând mesaje de confirmare către monitorul serial, inițiază comunicarea dintre µc și aplicația mobilă, setează pinii (intrare/ieșire) și citește primele valori ale temperaturii și umidității ambientale:

```
Serial.println("Booted");  // se afișează la pornire/reset
Serial.println("Connecting to Wi-Fi");  // se afișează la pornire/reset

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    // încerc conectarea microcontrolerului la wifi
    Serial.print(".");
    delay(500);
```

```
}

Serial.println("WiFi connected");

Blynk.begin(auth, ssid, password);
```

- `BLYNK_WRITE(V4)` este o funcție ce se apelează automat (doar) atunci când sliderul din aplicația mobilă (de pe pinul V4) și-a modificat valoarea. Funcția actualizează valoarea temperaturii dorite și o afișează pe ambele ecrane:

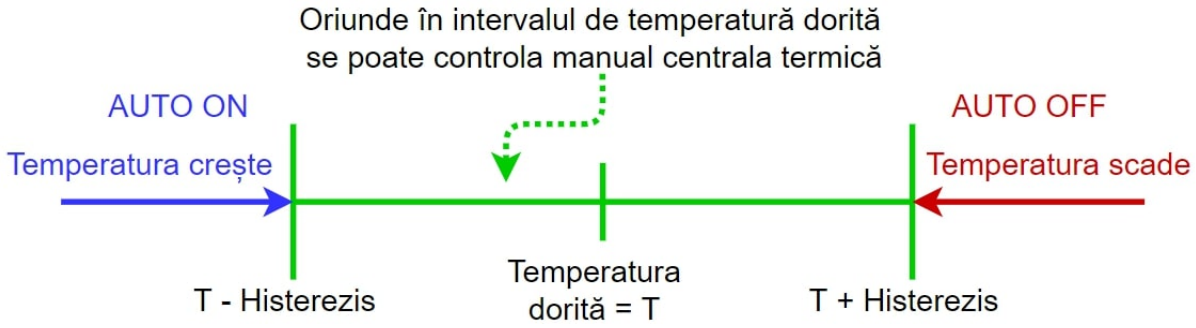
```
desired_temp = param.asInt(); // transform valoarea citită în întreg

blynk_lcd.print(0, 1, "Set to");
blynk_lcd.print(7, 1, desired_temp);
blynk_lcd.print(9, 1, "°C");

if (!printSeconds) {
    // afișez temperatura dorită doar când nu afișez secundele
    lcd.setCursor(12, 0);
    lcd.print(desired_temp);
    lcd.print((char)223);  // simbolul pentru grade °
    lcd.print("C");
}
```

- `void printDesiredTempToLcd()` afișează temperatura dorită pe ecrane (similar cu funcția anterioară), doar că aceasta se apelează constant în funcția `loop()`;
- `void printTimeToLcd(strDateTime dateTime)` afișează data și ora pe ecranul fizic într-un format ușor de citit pentru utilizator. Pe ecranul virtual al aplicației mobile am optat pentru neafișarea acestor informații întrucât sunt afișate mereu pe smartphone;
- `void printStateToLcd()` este funcția în care am implementat comandarea centralei termice atât în modul manual cât și în cel automat. Se afișează mesaje corespunzătoare, se aprinde/stinge martorul LED și se cuplează/decuplează releul. Funcționalitatea manuală pornește/oprește centrala în funcție de apăsarea butonului OK și de stadiul curent al centralei termice doar dacă temperatura ambientală se află în intervalul de temperaturi dorite (temperatura dorită/setată ± histerezis/threshold). Funcționalitatea automată a termostatului pornește dacă temperatura ambientală nu se regăsește în intervalul anterior, iar atunci termostatul va acționa corespunzător pentru a readuce temperatura ambientală în acest interval după cum urmează:

1. dacă temperatura ambientală este sub limita inferioară a intervalului atunci termostatul pornește centrala termică și o blochează în aceast stadiu până ce temperatura ambientală reintră în intervalul dorit sau până ce se micșorează intervalul dorit;
2. dacă temperatura ambientală este peste limita superioară a intervalului atunci termostatul oprește centrala termică și o blochează în aceast stadiu până ce temperatura ambientală reintră în intervalul dorit sau până ce se mărește intervalul dorit.

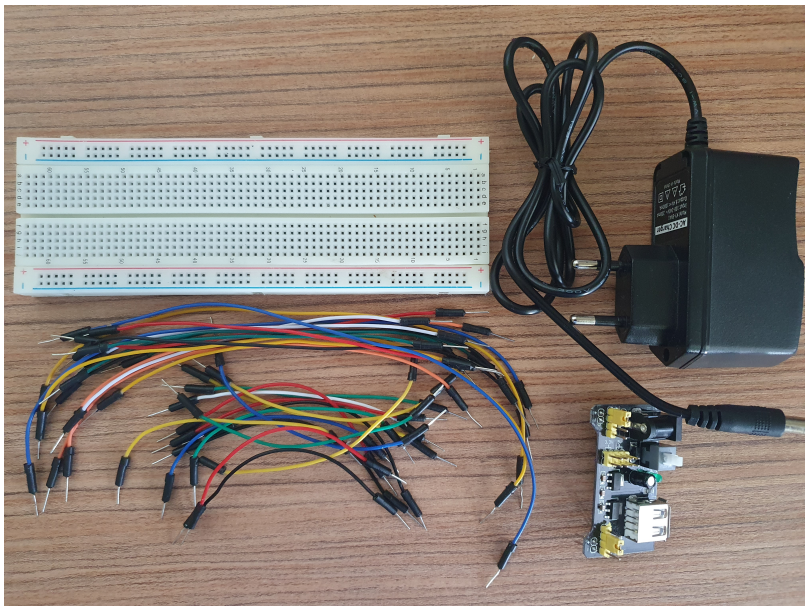


- `void printTempToSerialMonitor()` afișează temperatura și umiditatea ambientale și temperatura dorită către monitorul serial;
- `void printTempToLcd()` afișează temperatura și umiditatea ambientale către cele două ecrane;
- `void loop()` menține conexiunea cu aplicația Blynk, citește data și ora de la server cu comanda `NTPro.getNTPtime(1.0, 1);` și le afișează la monitorul serial și pe LCD-ul fizic, verifică dacă butoanele sunt apăsate (dacă da atunci fie pornește/oprește centrala termică, fie crește/scade temperatura dorită și citește și afișează temperatura și umiditatea de la senzorul DHT11.

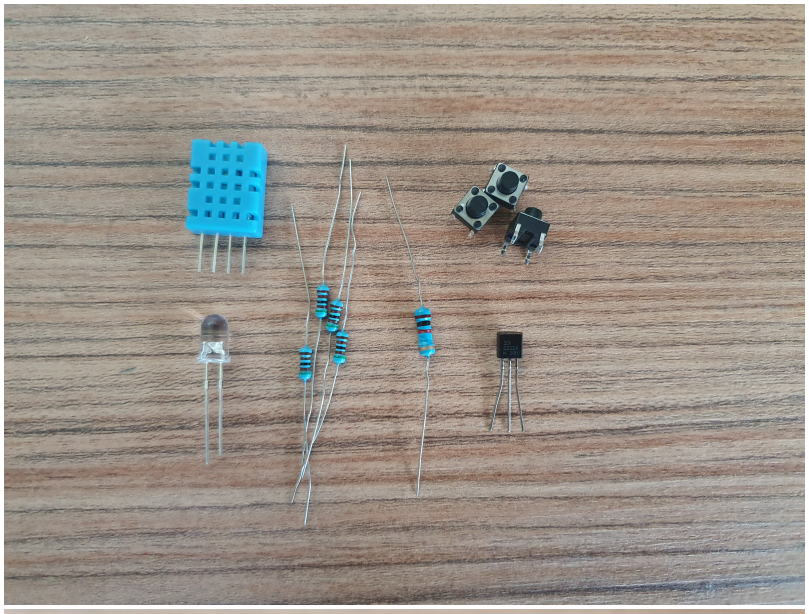
Rezultate Obținute

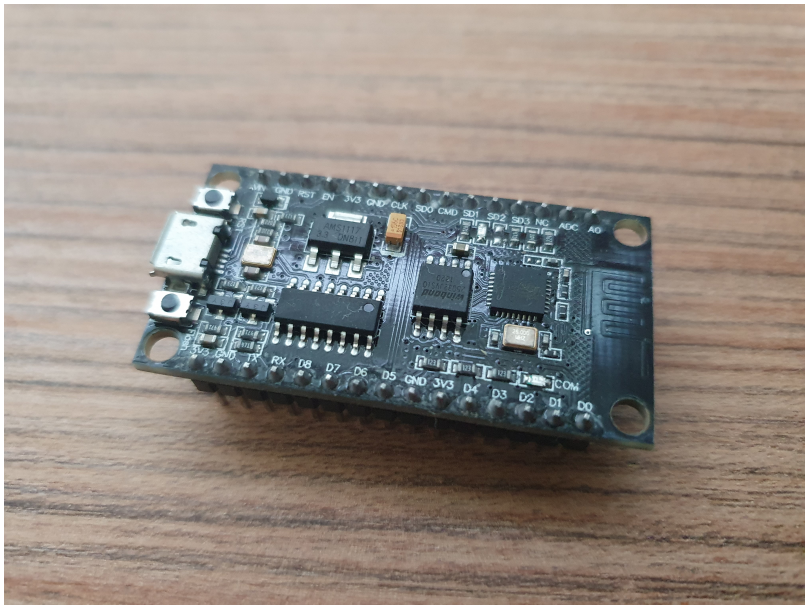
Poze:

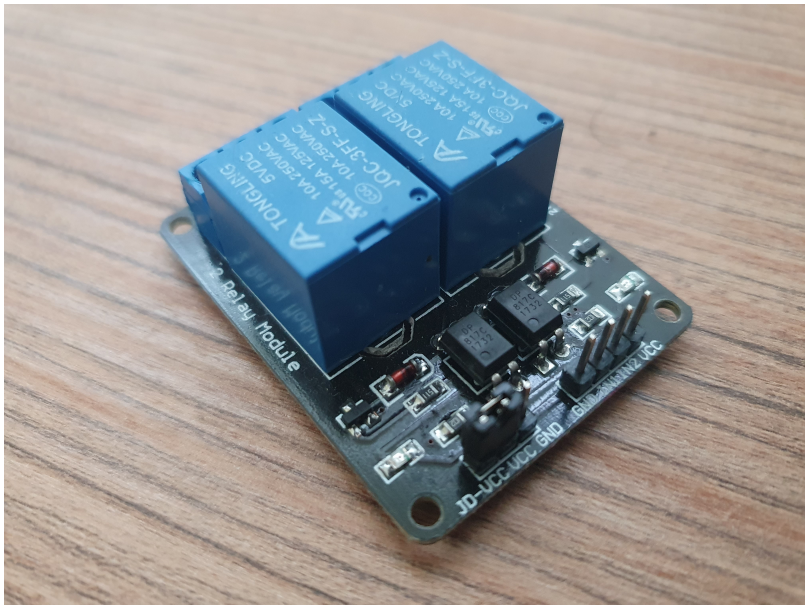
- Componentele folosite (similare cu cele din pozele furnizorului):



Sursa de alimentare furnizează 1000mA la 8,4V curent continuu ce este transformat ulterior la 5V de către sursa de breadboard și distribuit către esp8266, ecran și releu. Restul componentelor funcționează la 3,3V.

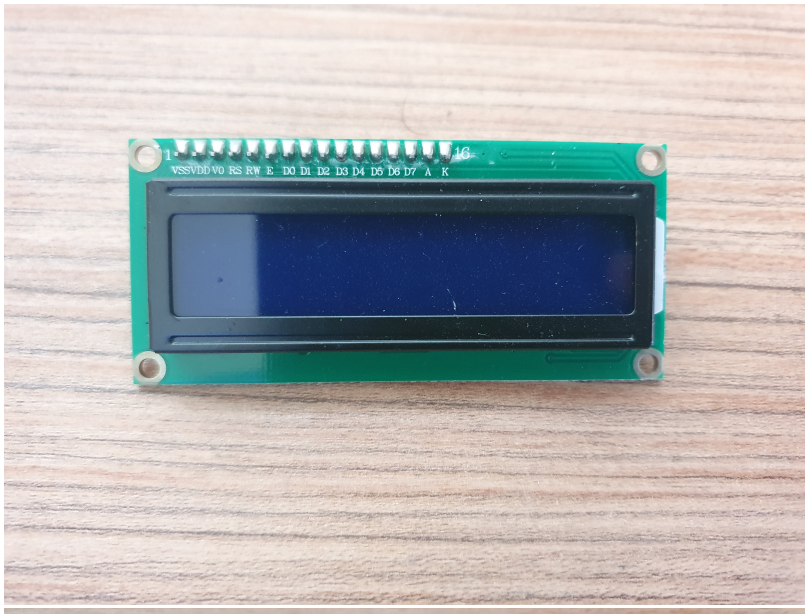


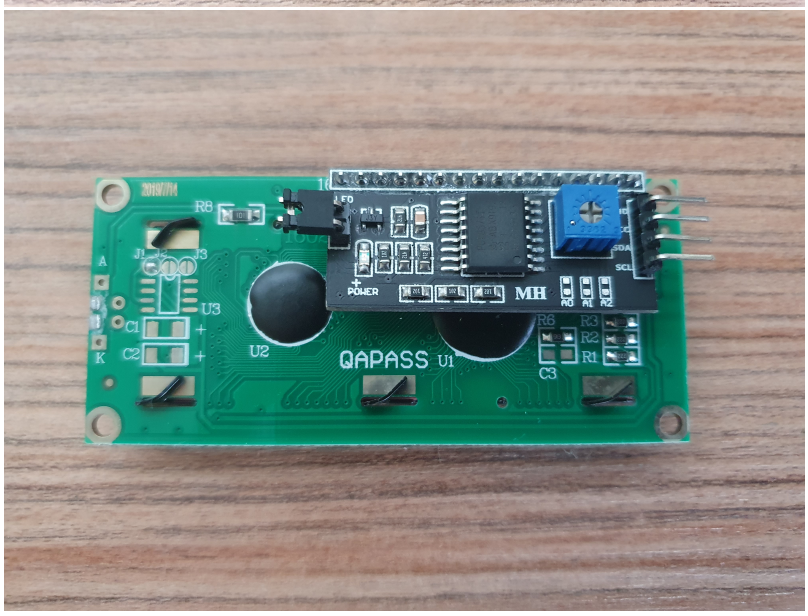




Releul folosit este diferit de cel inițial. Am optat ulterior pentru un releu ce dispune și de optocuplor. La momentul comandării pieselor am găsit doar modul cu două relee, dar voi utiliza doar unul. După

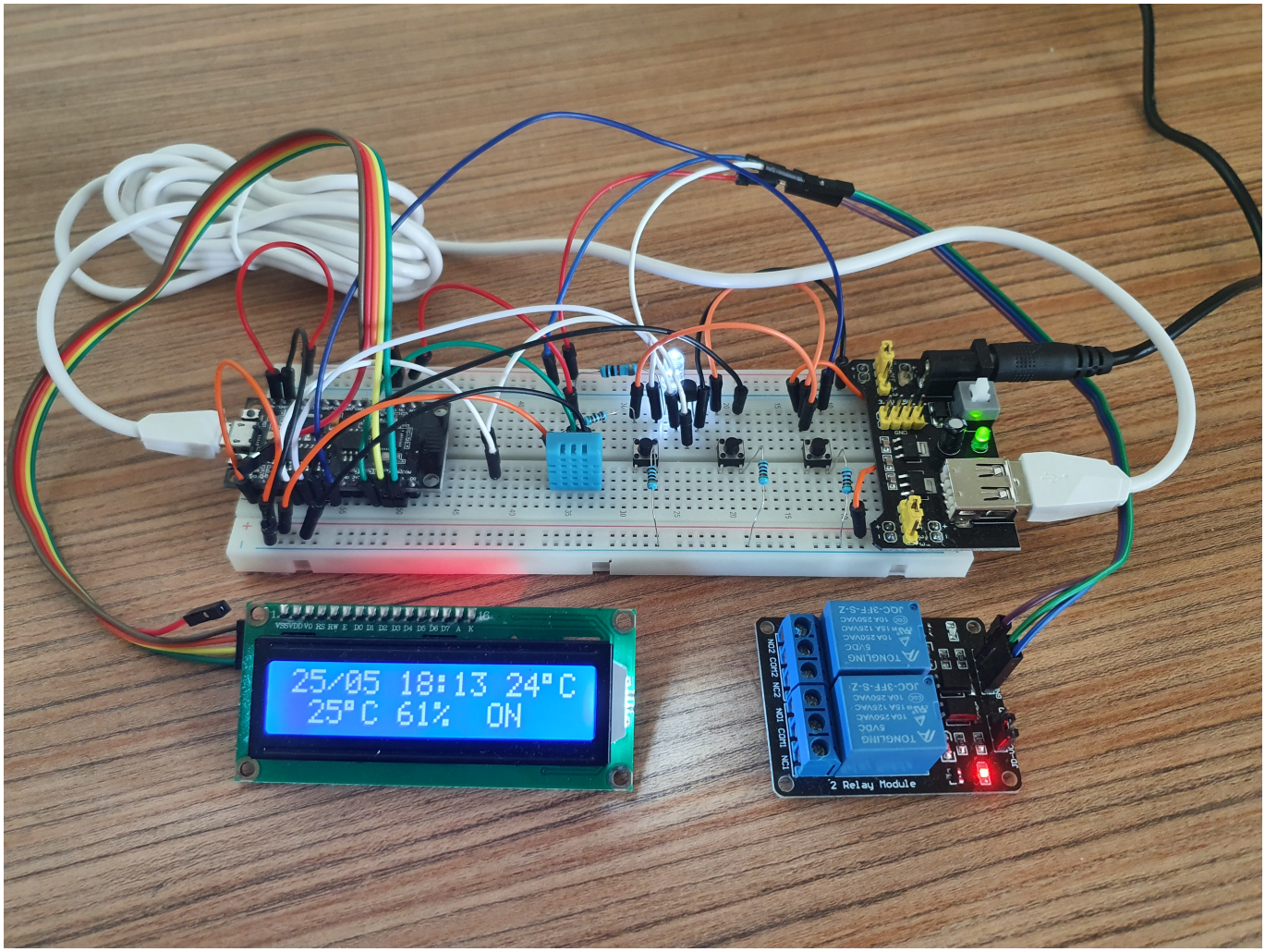
ce am testat releul am concluzionat că acesta cuplează pe un semnal LOW, iar microcontrolerul furnizează un semnal HIGH pentru cuplare (în mod normal pe pinul releului se furnizează un semnal LOW pentru că dacă se resetează microcontrolerul, acesta să nu cupleze releul și să pornească centrala termică accidental). Pentru rezolvarea acestei probleme am folosit un tranzistor ce lasă să treacă semnalul de GND/LOW către releu doar când baza acestuia este pe semnalul HIGH.





Ecranul are deja lipit adaptorul I2C, fiind necesară doar ajustarea luminozității acestuia (se folosește o șurubelniță în cruce ce ajustează potențiometrul albastru de pe spate).

- Ansamblul dispozitivului:



- Cele două mesaje de bun venit:

Mesajele sunt afișate și pe ecranul aplicației de pe smartphone.

Video ce arată funcționalitatea dispozitivului comandat atât de pe smartphone cât și de la postul de comandă local: wifi_central_heating_video.

Concluzii

Proiectul a fost dificil și îl recomand preponderent celor care doresc să accepte o provocare. Am învățat numeroasele lucruri: crearea unei scheme bloc, alcătuirea listei de componente, comandarea componentelor, ajustarea comenzii și a proiectului pentru că furnizorul de componente nu are pe stoc toate elementele necesare, asamblarea componentelor, implementarea de cod sursă atât pe microcontroler cât și pe aplicația de smartphone și altele. După cum spuneam anterior, releul mi-a dat puțin de furcă pentru că în mod firesc acesta trebuia sa cupleze pe semnalul HIGH și să decupleze pe semnalul LOW, lucru ce nu se întâmplă.

Pentru cei neînfricați: *Spor la proiect!*

Download

Documentația procesorului esp8266: esp8266-nodemcu-datasheet.pdf.

Fișierele obținute (codul sursă, schema eagle, biblioteci folosite, schema bloc, intervalul de temperatură dorită): wifi_thermostat_files.zip.

Jurnal

- 12 aprilie - achiziție componente hardware;
- 15 - 18 aprilie - finalizare conectare componente;
- 22 - 23 aprilie - testare funcționalitate componente, începere cod sursă;
- 30 aprilie - finalizare cod sursă, realizare aplicație smartphone;
- 5 - 6 mai - realizare schemă bloc, schemă electrică;
- 14 - 16 mai - completare pagină documentație;
- 24 - 26 mai - finalizare pagină documentație proiect.

Bibliografie/Resurse

Resurse Software:

- Aplicația Blynk cu NodeMCU
- Biblioteca ESP8266WiFi
- Biblioteca SimpleDHT
- Biblioteca Wire
- Biblioteca LiquidCrystal
- Biblioteca NTPtimeESP
- Generator de caractere speciale pentru ecran LCD

Resurse Hardware:

- Documentația NodeMCU ESP8266
- Update de firmware NodeMCU
- Documentație ecran LCD 16x2
- Documentație modul I2C pentru ecran
- Documentație modul releu
- Documentație DHT11
- Documentație sursă pentru breadboard

Documentația proiectului în format pdf: wifi_thermostat_for_central_heating.pdf.