Cofinanțat prin programul Erasmus+ al Uniunii Europene

## Fișa nr. 9: Utilizarea dronelor și roboților

*Explicații:*

Această activitate este concepută pentru a descoperi aplicabilitatea dronelor în domenii diferite de cele militare sau comerciale, în condițiile în care ele sunt echipate cu cameră sau senzori.

*Conexiuni curriculare*

| Disciplina | Competențe |
|---|---|
| Informatică | Dezvoltarea și aplicarea abilităților analitice ale elevilor, de rezolvare a problemelor, de proiectare și de programare utilizând Arduino. |
| Științe | Organizarea lucrărilor științifice: aprecierea puterii și limitărilor științei, explicarea aplicațiilor cotidiene și tehnologice ale științei; evaluarea implicațiilor personale, sociale, economice și de mediu asociate; luarea deciziilor pe baza evaluării dovezilor și argumentelor |

Resurse necesare:
- dronă / robot
- cameră video
- senzori, cablu USB
- laptop, Arduino Uno

O dronă este definită ca un vehicul aerian fără pilot uman la bord, iar marea lor majoritate sunt proiectate să poarte o cameră foto, ceea ce ridică probleme legate de intimitate. Zborul dronelor poate fi controlat fie în mod autonom de computerele de la bord, fie prin telecomanda unui pilot, dar e nevoie de instruire și aptitudini adecvate pentru a evita accidentele.

Cu ajutorul camerei, drona poate fotografia anumite zone de teren. Imaginile sunt apoi folosite pentru a produce o hartă, care poate fi apoi folosită pentru a lua decizii în vederea gestionării eficiente a zonei respective. Cu ajutorul dronelor pot fi identificate probleme ale mediului care nu sunt evidente la nivelul ochiului (de exemplu: infestări dăunătoare și fungice, probleme legate de irigare, diferențe dintre plante).

## *Exercițiul 1: Arduino (prezentare generală)*

Dronele pot constitui platforme robuste pentru diferiți senzori care să furnizeze informații relevante referitoare la indicatori ai mediului înconjurător. În acest fel, crește utilitatea unei drone, în același timp cu oferirea unor oportunități excepționale de învățare.
Arduino este o platformă electronică de prototipuri open-source pentru crearea de obiecte interactive, ce poate rula pe Windows sau Linux. Faptul că este platformă open-source este benefic pentru începători, deoarece găsesc resurse online pe diferite teme și niveluri de calificare.
Arduino este compus din două părți importante:
- placa Arduino, care este piesa de hardware la care lucrați când construiți obiectele;
- mediul de dezvoltare integrat Arduino, sau IDE, softul pe care îl utilizați pe calculator.
Softul este gratuit și poate fi descărca de pe www.arduino.cc. Placa Arduino este ieftină și tolerantă la greșeli comune. Softul este utilizat pentru a crea programul care se încarcă pe placa Arduino și îi spune ce să facă.

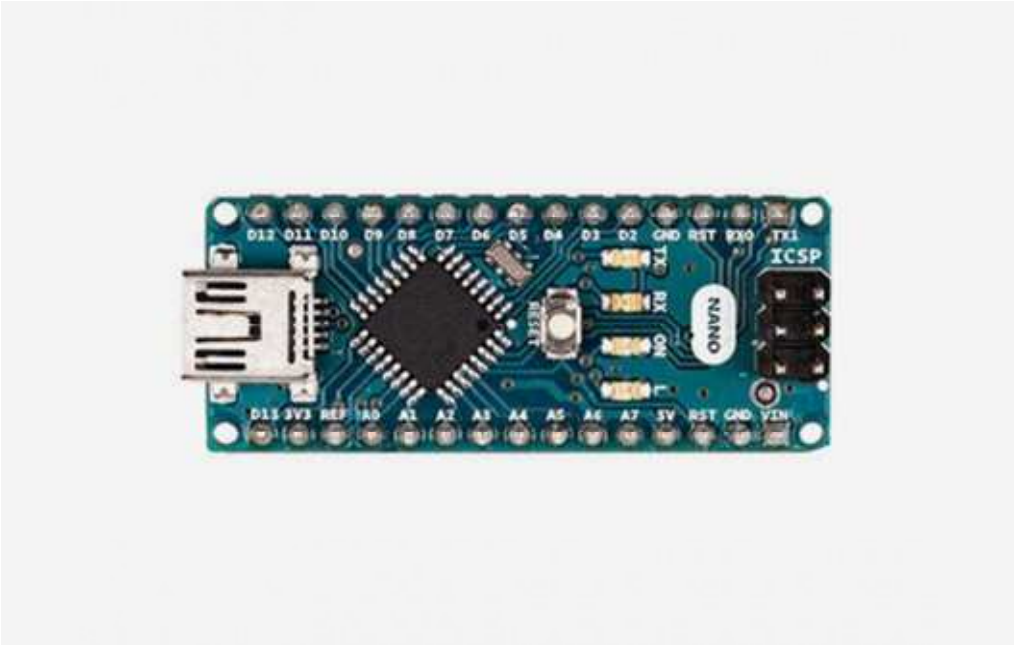



Fig. 1. Placa de dezvoltare Arduino Nano

Placa Arduino este o placă de microcontroler, care are un șir de benzi în partea de sus și de jos cu numeroase etichete. Aceste benzi sunt conectorii, care sunt folosiți pentru a fi atașați la senzori și la dispozitive de acționare.

- 12 pini digitali I / O (pinii D2-D13). Acești pini pot fi fie intrări, fie ieșiri. Intrările sunt folosite pentru a citi informații de la senzori, în timp ce ieșirile sunt folosite pentru a

controla servomotoarele. Intrările digitale pot citi numai una din cele două valori, iar ieșirile digitale pot afișa numai una din cele două valori (HIGH și LOW).

- 8 pini analogici (pinii A0-A7). Pinii de intrare analogici sunt utilizați pentru citirea măsurătorilor de tensiune de la senzorii analogici. Spre deosebire de intrările digitale, care pot distinge doar două niveluri diferite (HIGH și LOW), intrările analogice pot măsura 1.024 niveluri diferite de tensiune.

Placa Arduino poate fi alimentată de la portul USB al computerului sau cu un adaptor (recomandat de 9V).

Ciclul de programare utilizând Arduino are următorii pași:

1. Conectați placa la un port USB de pe computer.
2. Scrieți codul în C pe care doriți să fie executat de Arduino.
3. Compilați codul pentru a verifica dacă totul este în regulă
4. Încărcați această codul pe placă prin conexiunea USB și așteptați câteva secunde până când placa va reporni.
5. Placa execută (efectuează) codul pe care l-ați scris.

Arduino s-a născut pentru a preda *Design interactiv* – o disciplină de design care pune prototipul în centru metodologiei sale, pentru a crea experiențe semnificative între oameni și obiecte.

## *Exercițiul 2: Mediul de dezvoltare Arduino IDE*

Programarea oricărui microcontroller Arduino se efectuează în Arduino IDE, platformă dezvoltată special pentru aceste dispozitive. Limbajul de programare utilizat este C, iar șablonul unui program pentru Arduino este prezentat în figura de mai jos.

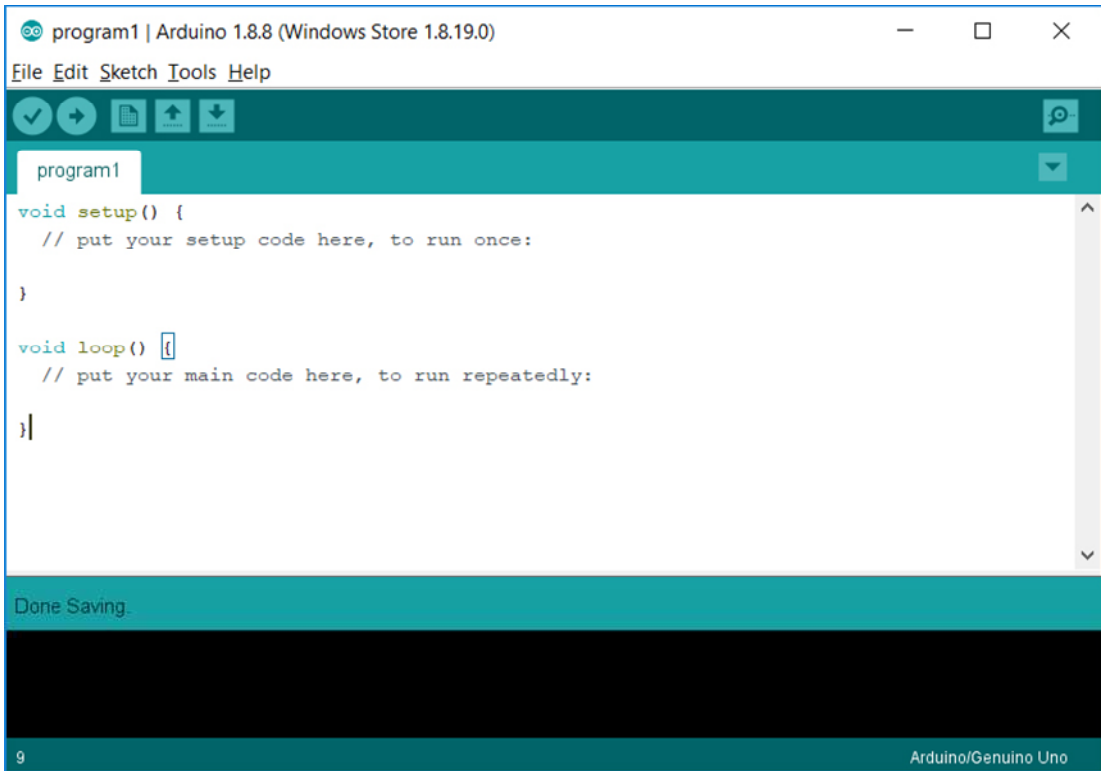



Fig. 2. Mediul de dezvoltare Arduino IDE

Programul conține două funcții principale:

- setup
- loop

Aceste funcții trebuie să existe obligatoriu într-un program pentru Arduino, alte funcții putând fi adăugate ulterior.

Funcția ***setup*** are rolul de a inițializa o singură dată instrucțiunile scrise în cadrul acesteia, de obicei la alimentarea microncontroller-ului Arduino. În cadrul funcției se pot inițializa senzorii, comunicația serială prin USB sau orice alt dispozitiv conectat la pinii Arduino.

Funcția ***loop*** execută continuu instrucțiunile scrise în cadrul acesteia, până la oprirea alimentării microcontroller-ului. Aici se pot prelua și interpreta datele de la senzori și de la alte periferice, cât și transmiterea acestora utilizând portul serial de care beneficiază orice Arduino. Tot aici se poate determina modul de interacțiune al microcontroller-ului cu mediul înconjurător (de exemplu: aprindem ledul verde dacă temperatura are o valoarea mai mare de 20 grade Celsius).
Pentru a încărca programul scris pe Arduino, putem utiliza meniul simplificat al aplicației Arduino IDE care arată precum în figura următoare.


Fig. 3. Meniul mediului de dezvoltare Arduino IDE

Unde:

1. Butonul *Verify*
2. Butonul *Upload*
3. Butonul *New*
4. Butonul *Open*
5. Butonul *Save*
6. Butonul *Serial Monitor*

Butonul *Verify* are rolul de a compila instrucțiunile și de a afișa eventualele erori care au apărut la compilare în partea de jos a programului. În cazul unor erori des întâlnite, acesta vă va arăta și modul prin care le puteți remedia.
Butonul *Upload* încarcă programul scris pe Arduino, verifică compatibilitatea acestuia cu placa și afișează, dacă încărcarea s-a efectuat cu succes, memoria ocupată din microcontroller de către programul actual (*Upload* realizează implicit funcția *Verify*, astfel nu veți putea încărca un program cu erori pe placă).
Butonul *New* creează un nou fișier cod sursă (având extensia .ino).
Butonul *Open* este utilizat pentru deschiderea unui fișier cod sursă cu extensia .ino.
Butonul *Save* salvează programul scris într-o locație pe care o specificați.
Butonul *Serial Monitor* deschide portul de comunicație serial al microcontroller-ului. Dacă acesta nu a fost inițializat în funcția **setup** a programului, va afișa o eroare (de exemplu: comunicație neinițializată/indisponibilă).

*Exercițiul 3: Scrierea unui program simplu*

Următorul program realizează aprinderea, respectiv stingerea, a două LED-uri de culori diferite (în acest caz verde și galben), în funcție de o valoare tastată de către operator. Codul sursă al acestui program arată astfel:

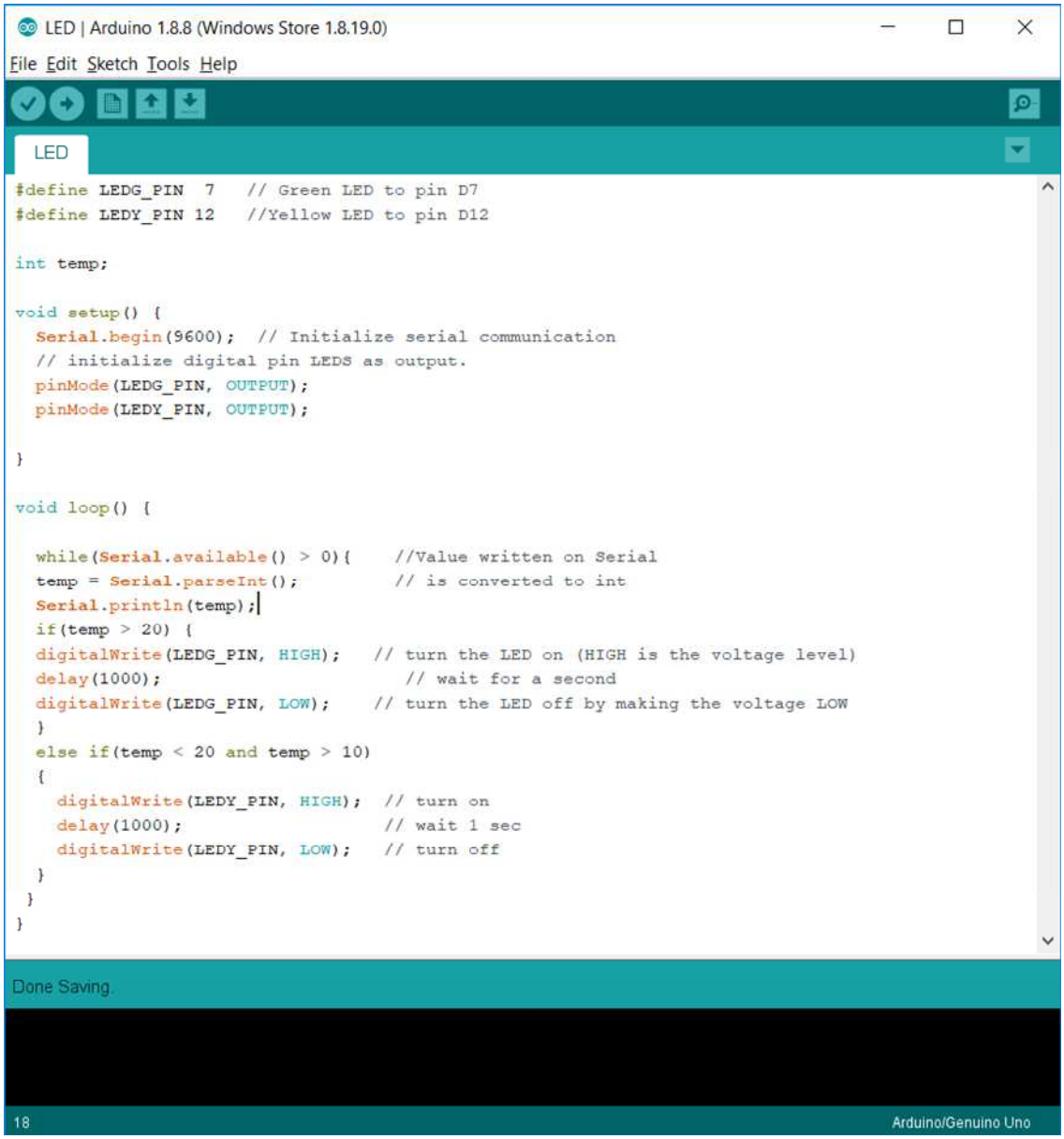



Fig. 4. Codul sursă al programului de aprindere/stingere LED-uri

Cod sursă LED

```
#define LEDG_PIN  7   // Green LED to pin D7
#define LEDY_PIN 12   //Yellow LED to pin D12

int temp;

void setup() {
  Serial.begin(9600);  // Initialize serial communication
  // initialize digital pin LEDS as output.
  pinMode(LEDG_PIN, OUTPUT);
  pinMode(LEDY_PIN, OUTPUT);

}

void loop() {

  while(Serial.available() > 0){    //Value written on Serial
  temp = Serial.parseInt();         // is converted to int
  Serial.println(temp);
  if(temp > 20) {
  digitalWrite(LEDG_PIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LEDG_PIN, LOW);    // turn the LED off by making the voltage LOW
  }
  else if(temp < 20 and temp > 10)
  {
    digitalWrite(LEDY_PIN, HIGH);  // turn on
    delay(1000);                  // wait 1 sec
    digitalWrite(LEDY_PIN, LOW);   // turn off
  }
 }
}
```

Pentru a încărca programul pe o placă Arduino Nano trebuie să urmați următorii pași:

1. Din meniul **Tools** –> Board -> Selectați „Arduino Nano”
2. Din meniul **Tools** -> Processor -> Selectați „ATmega328P (Old Bootloader)”
3. Din meniul **Tools** -> Port -> Selectați portul disponibil
4. Din meniu **Tools** -> Programmer -> Selectați „USBasp”

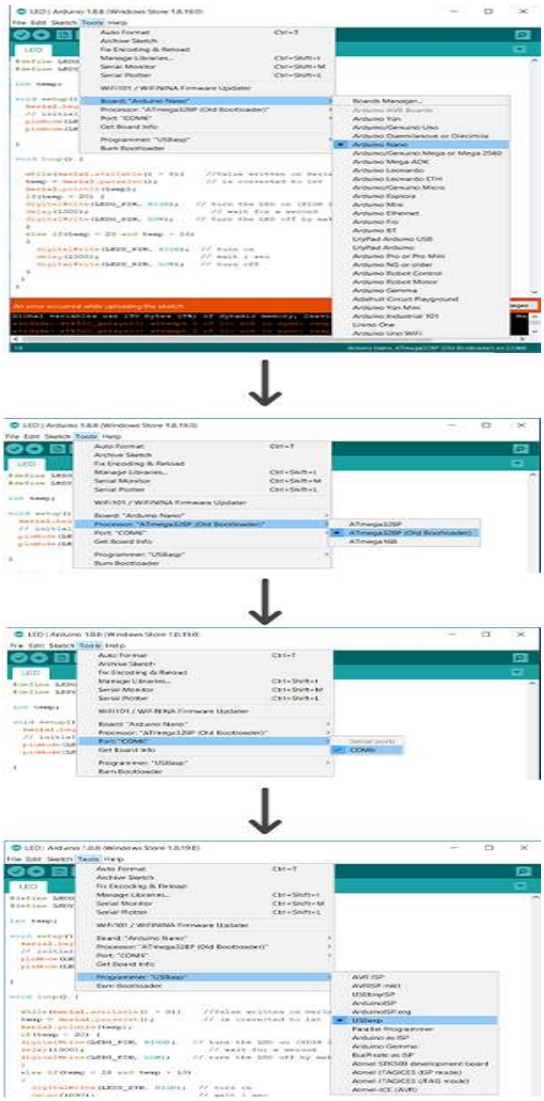


Fig. 5. Pașii de configurare pentru Arduino Nano

Funcționarea programului este următoarea:

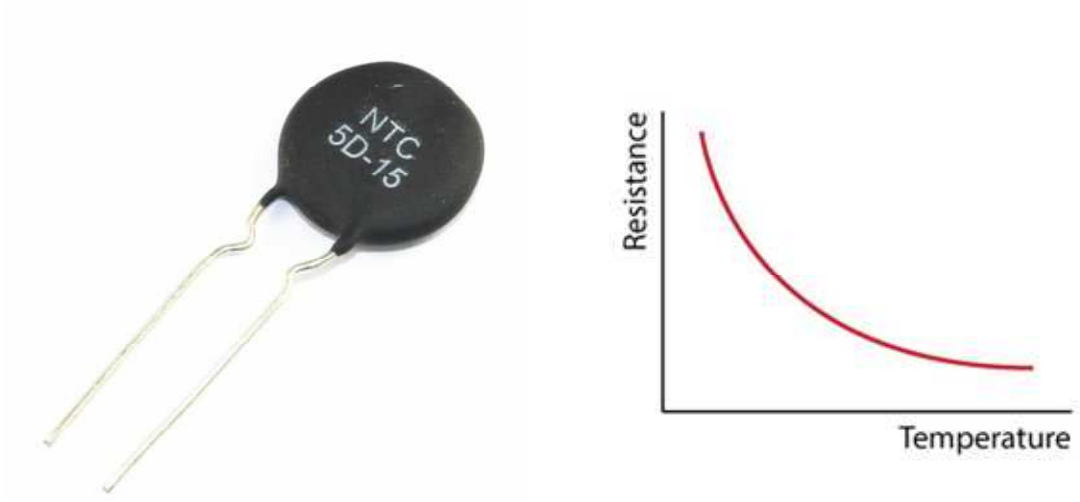
- La alimentarea Arduino este executată funcția **setup** în cadrul acesteia inițializându-se comunicația serială și pinii corespunzători celor două LED-uri (D7, D12) de tip *output* (Arduino furnizează tensiunea de alimentarea a LED-urilor, atunci când este specificat în funcția *loop*)
- Urmează execuția continuă a funcției *loop* ce efectuează verificarea portului serial (dacă a fost introdusă o valoare) urmând ca aceasta:
  - Să fie convertită într-o valoare de tip *int*
  - Să fie afișată din nou pe serial (pentru verificarea conversiei)
  - Să fie comparată, iar în funcție de valoare, să poată fi luată o decizie

## *Exercițiul 4: Utilizarea senzorului de temperatură și umiditate*

Determinarea parametrilor de mediu precum temperatura și umiditatea se poate realiza destul de ușor, prin conversia acestora într-o mărime electrică (cel mai adesea este vorba de tensiune electrică).

În continuare este prezentat modul în care senzorul DHT22 determină acești parametri utilizând componente electronice interconectate.

Pentru determinarea temperaturii, senzorul utilizează un element sensibil de tip termistor. Termistorul are o variație exponențială inversă a rezistenței electrice proprii (rezistența scade cu creșterea temperaturii). O astfel de componentă electrică este prezentată în figura următoare.




A B

Fig. 6. Termistorul. A – Componenta fizică, B- Variația rezistenței cu temperatura

Pentru determinarea umidității, senzorul utilizează un element sensibil de tip capacitiv. Impedanța acestuia variază în funcție de umezeala acumulată pe substratul dintre cei doi electrozi. În figura următoare este reprezentat atât elementul sensibil (1), cât și structura internă (2) a acestuia.

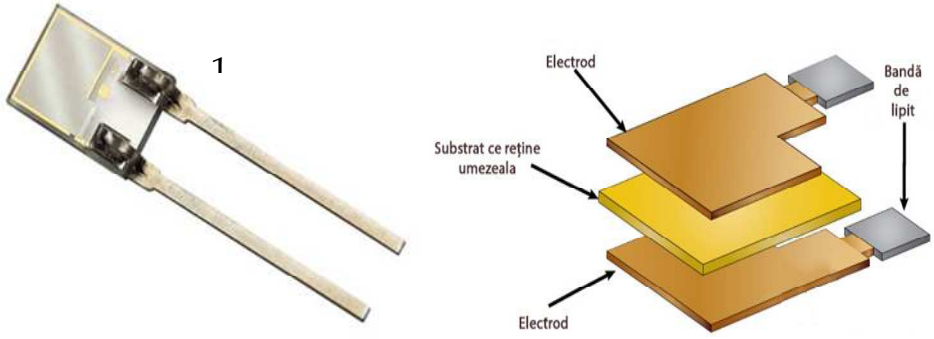



Fig. 7. Element sensibil pentru umiditate

Aceste două componente electronice sunt parte a unui circuit integrat (IC), având rolul de a converti în format digital datele primite de la elementele sensibile (tensiunea electrică) și de a comunica aceste date altui dispozitiv.

| DHT22 pins | |
|---|---|
| 1 | VCC |
| 2 | DATA |
| 3 | NC |
| 4 | GND |


Fig. 8. Senzorul integrat DHT22 de măsurare a

Codul sursă al programului încărcat:

Cod sursă DHT22

```
#include "DHT.h"

#define DHTPIN 7    // Sensor digital pin

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT22 test!");

  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
```

```
 float t = dht.readTemperature();
 // Read temperature as Fahrenheit (isFahrenheit = true)
 float f = dht.readTemperature(true);

 // Check if any reads failed and exit early (to try again).
 if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
 }
 // Compute heat index in Fahrenheit (the default)
 float hif = dht.computeHeatIndex(f, h);
 // Compute heat index in Celsius (isFahreheit = false)
 float hic = dht.computeHeatIndex(t, h, false);

 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(t);
 Serial.print(" *C ");
 Serial.print(f);
 Serial.print(" *F\t");
 Serial.print("Heat index: ");
 Serial.print(hic);
 Serial.print(" *C ");
 Serial.print(hif);
 Serial.println(" *F");
}
```

Veți avea nevoie de o bibliotecă specială pentru un astfel de senzor, iar pentru descărcarea acesteia urmăriți pașii:

Descărcare librarărie DHT

1. Apăsați CTRL + SHIFT + I
2. Scrieți în bara de search „dht” și instalați biblioteca de la *Adafruit* (by Adafruit)
3. Dacă apar probleme la compilare după instalare („Adafruit_Sensor.h not found”), descărcați fișierul lipsă de aici și adăugați-l în locația precizată de adresa erorii (de obicei C:\Users\#user#\Documents\Arduino\libraries\DHT_sensor_library)

Funcționarea programului este următoarea:

- Sunt inițializate portul serial și senzorul (dht.begin)
- Deoarece timpul de răspuns al DHT22-ului este de aprox. 2 secunde, se creează o întârziere prin funcția delay
- Sunt citite valorile furnizate de senzor (umiditate, temperatura în grade Celsius și Fahrenheit), iar dacă acestea sunt diferite de 0, vor fi considerate valide
- Este calculat indexul de temperatură cu ajutorul funcției specifice

- În final, sunt afișate pe portul serial datele obținute

Programul afișează pe portul serial următoarele date:

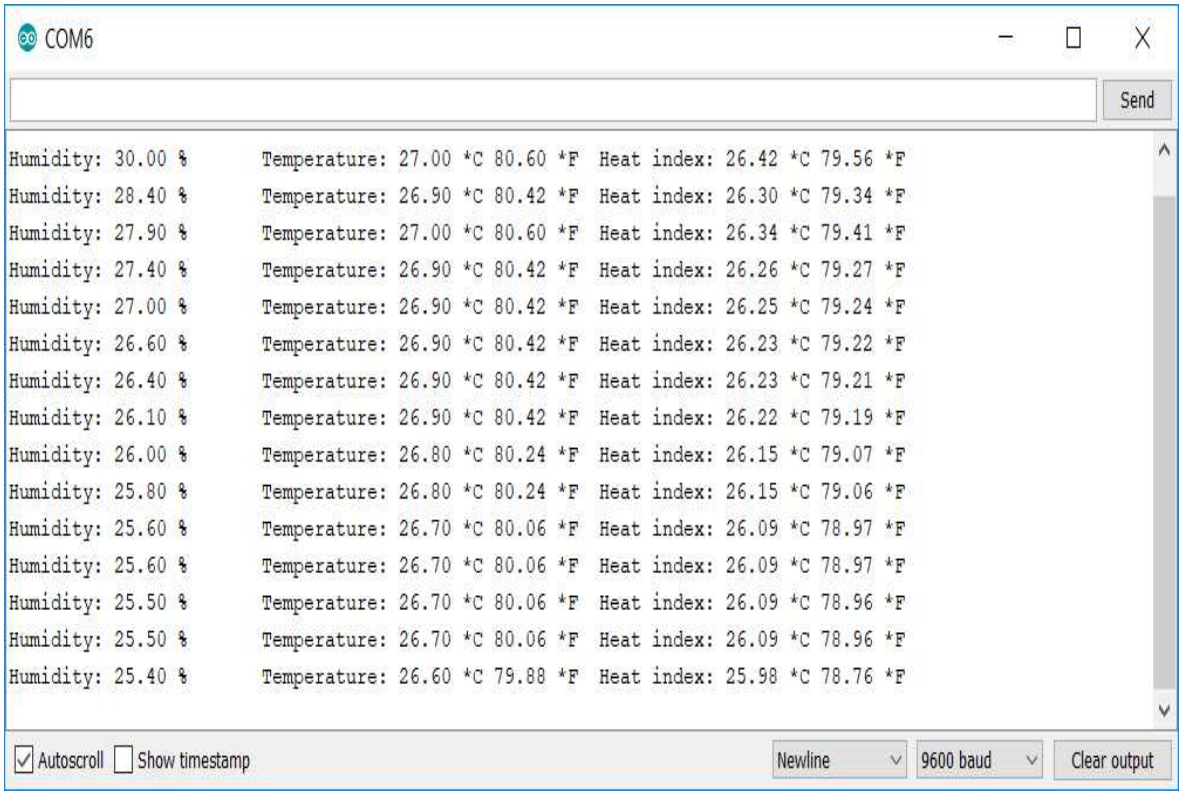



Fig. 9. Portul serial

Senzorul trebuie conectat la Arduino Nano ca în figura de mai jos.

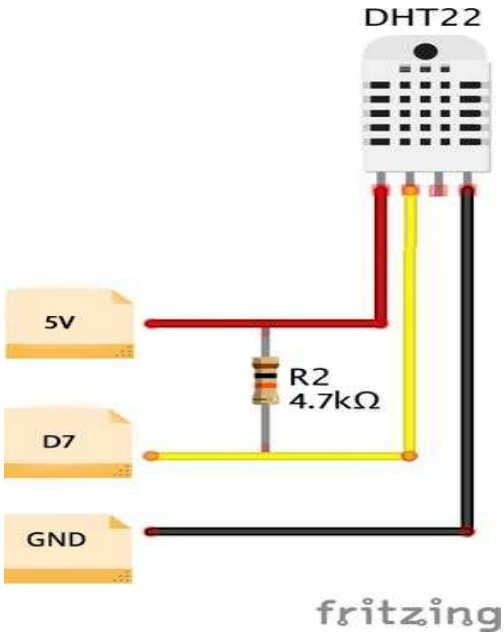



Fig. 10. Schema electrică de conectarea a

## *Exercițiul 5: Stocarea și prelucrarea datelor de la senzori*

Stocarea și prelucrarea datelor are loc pe Arduino, acesta având la dispoziție o memorie de 1KB de tip EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory).
Acest tip de memorie este caracterizată prin faptul că datele stocate nu sunt șterse atunci când alimentarea dispozitivului este oprită.
După încărcarea codului sursă ce efectuează stocarea și prelucrarea datelor, Arduino poate fi alimentat de la bateria dronei, conectând + la pin-ul „$V_{in}$” și – la pin-ul „GND” al acestuia.
Pe lângă conexiunile din schema electrică reprezentată mai jos, pin-ul D9 are atașată o rezistență electrică ce are rol dublu, atât de pull-up (1 logic, când este conectată la 5V), cât și de pull-down (0 logic, când este conectată la GND). Rolul acesteia este de a seta modul de funcționare al ansamblului senzori-Arduino, operatorul alegând prin utilizarea acestui pin modul de scriere/citire al datelor din memoria EEPROM ( 1 = citire, 0 = scriere).

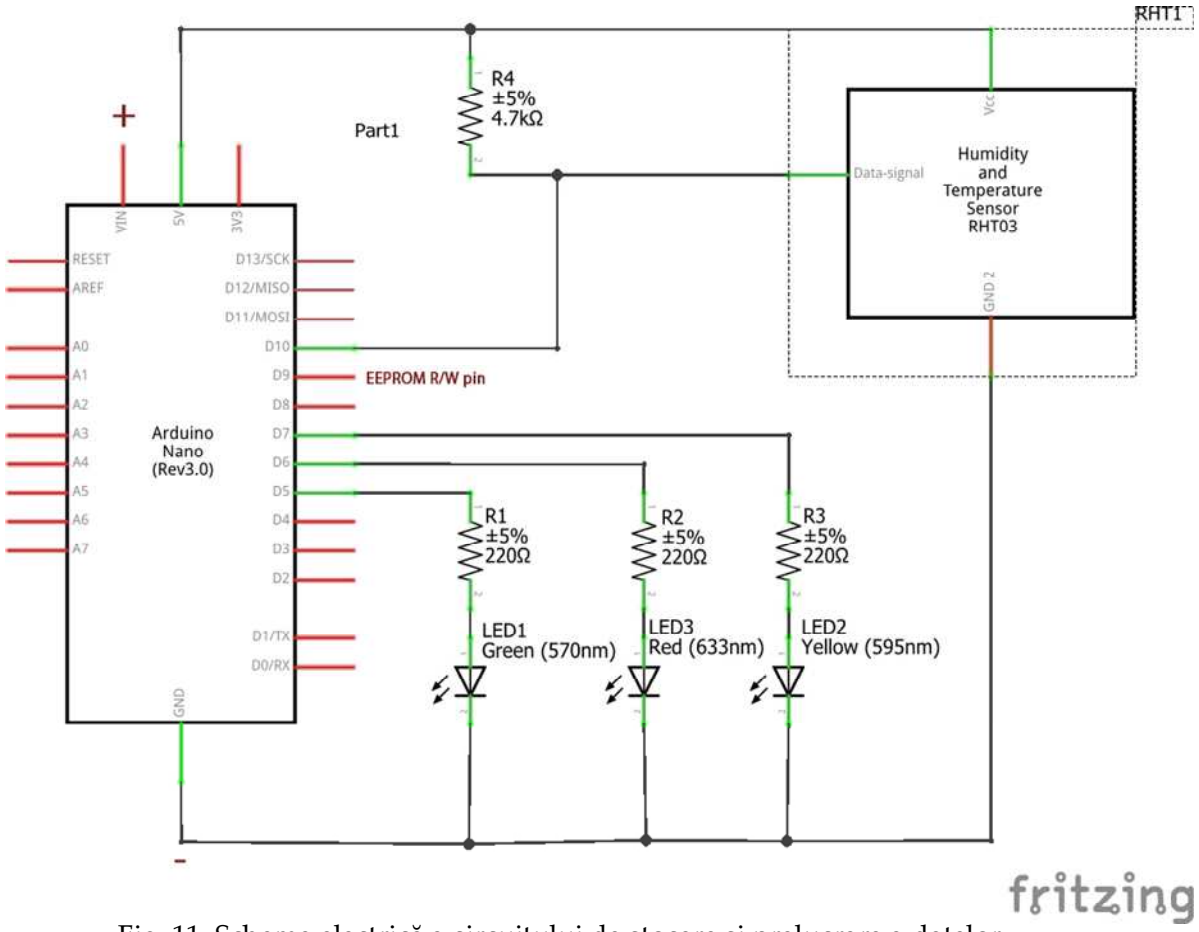



Fig. 11. Schema electrică a circuitului de stocare și prelucrare a datelor

Circuitul fizic este prezentat în imaginile următoare:

.

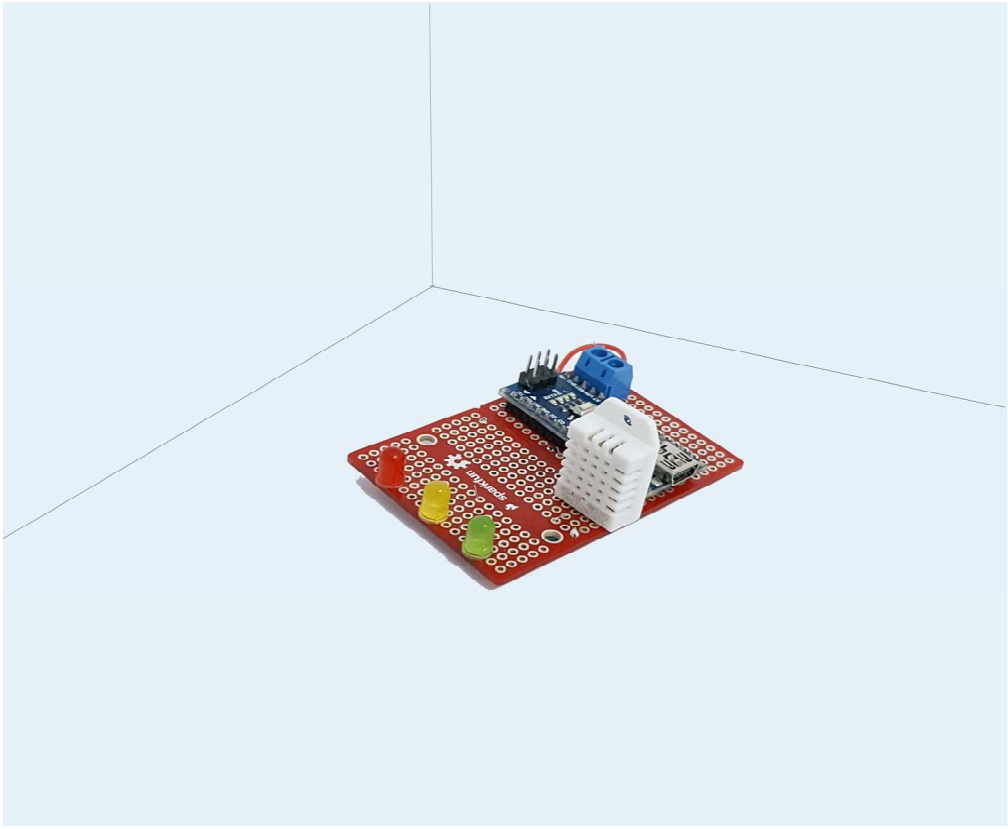


Fig. 12. Montajul de stocare și prelucrare a datelor, privire din stânga

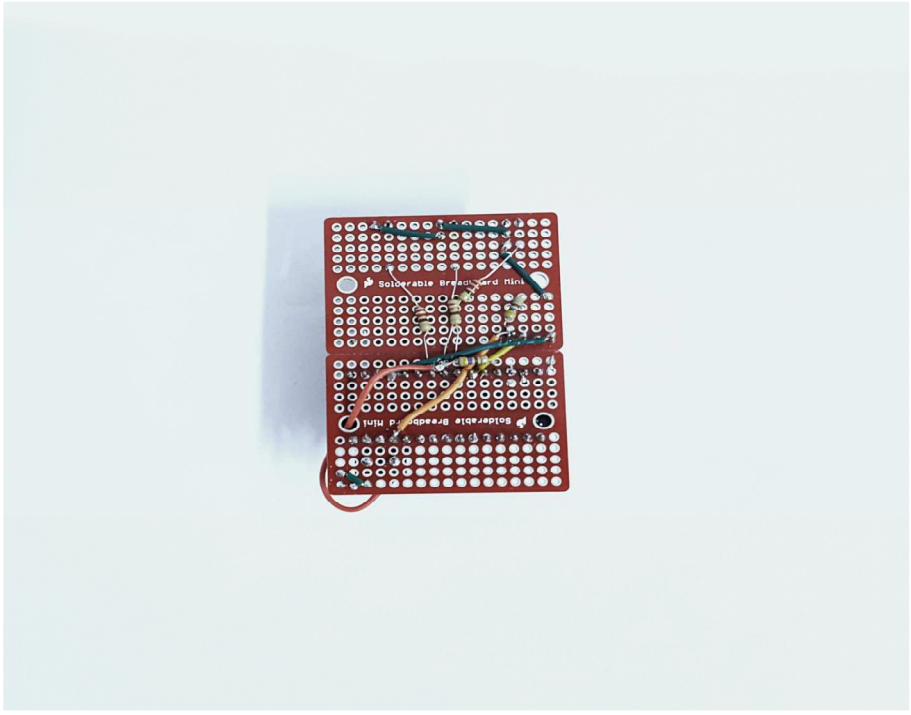


Fig. 13. Montajul de stocare și prelucrare a datelor, privire de jos

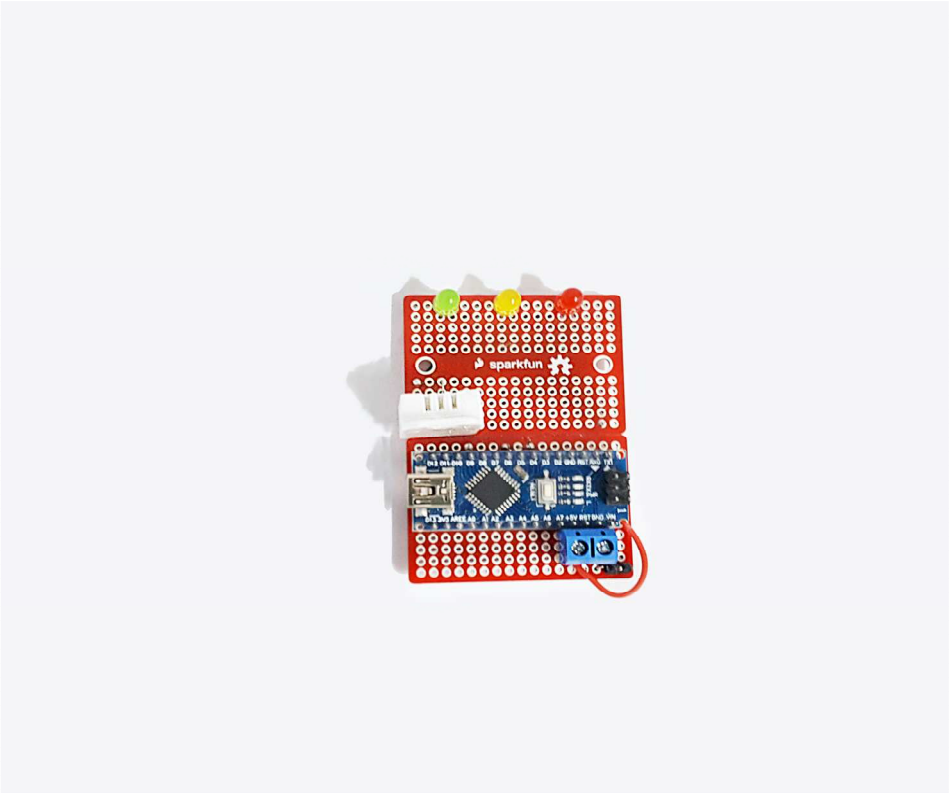



Fig. 14. Montajul de stocare și prelucrare a datelor, privire de sus

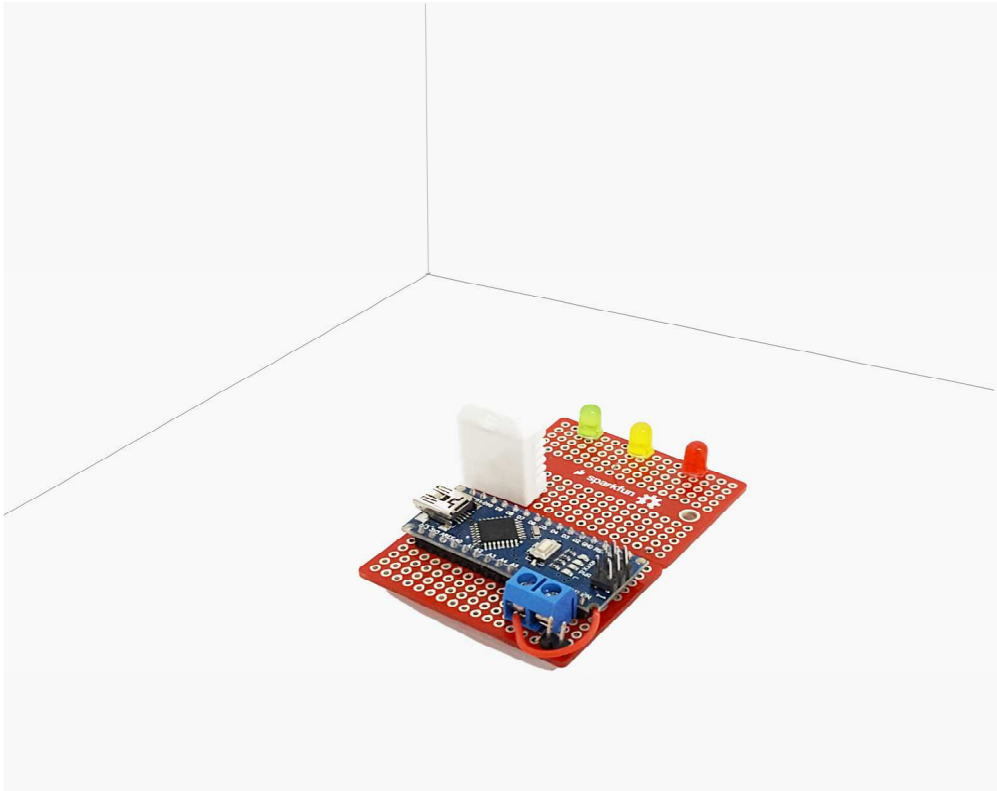


Fig. 15. Montajul de stocare și prelucrare a datelor, privire din stânga