

lordache Valentin, Cormoș Angel Ciprian, Costea Ilona Mădălina

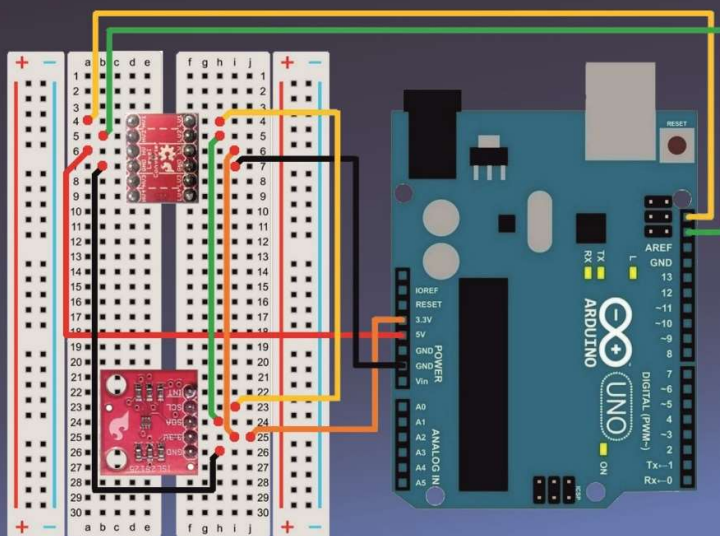
SENZORI

TRADUCTOARE

ȘI ACHIZIȚII DE DATE CU

# ARDUINO UNO

lucrări practice





**Valentin IORDACHE  
Angel Ciprian CORMOȘ  
Iona Mădălina COSTEA**

**SENZORI, TRADUCTOARE  
ȘI ACHIZIȚII DE DATE CU ARDUINO UNO**

**LUCRĂRI PRACTICE**

**EDIȚIE REVIZUITĂ**

**Editura POLITEHNICA PRESS  
București, 2019**

**Copyright ©, 2019, Editura POLITEHNICA PRESS**  
Toate drepturile asupra acestei ediții sunt rezervate editurii.

Adresa: Calea Griviței, nr. 132  
10737, Sector 1, București  
Telefon: 021.402.90.74

*Referenți științifici:*  
prof. univ. dr. ing. **Iulian BĂDESCU**  
conf. dr. ing. **Marius MINEA**

---

**ISBN: 978-606-515-853-5**

---

# PREFAȚĂ

Această carte a fost gândită inițial pentru a fi de folos studenților departamentului Telecomenzi și Electronică în Transporturi din Facultatea Transporturi a Universității Politehnica din București, în care profesază autorii, în parcurgerea aplicațiilor practice din cadrul laboratorului de Senzori, Traductoare și Achiziții de Date, însă modul de prezentare și structurare a informațiilor, precum și utilizarea unor componente și module foarte ușor de procurat, o transformă într-o sursă utilă de informații pentru orice pasionat de electronică.

Cartea conține opt lucrări practice realizate cu placa de dezvoltare Arduino Uno, unele dintre ele cuprinzând mai multe aplicații individuale. Fiecare lucrare conține o parte de introducere în care sunt evidențiate concepte fundamentale utile în înțelegerea modului de utilizare al modulelor și componentelor folosite, prezentarea detaliată a componentelor hardware iar unde este necesar și modul de configurare al acestora, explicarea individuală a instrucțiunilor, funcțiilor și operatorilor folosiți în secvențele de cod, detalierea modului realizare al montajului electronic precum și prezentarea schemei logice a operațiilor și a secvenței de cod complete, cu explicarea rolului fiecărei linii de cod. În plus, fiecare lucrare conține la final un set de exerciții suplimentare și concluzii ce ajută la consolidarea conceptelor abordate în cadrul aplicațiilor.

Tematica abordată în cadrul lucrărilor cuprinde prezentarea unor principii de bază în utilizarea plăcii de dezvoltare Arduino Uno, a principiilor utilizării unor senzori analogici sau digitali de măsurare a parametrilor mediului înconjurător, a distanței și proximității, a nivelului de iluminare, a tensiunii și intensității curentului electric, precum și a modului de realizare a unui ceas în timp real, ce poate fi util în a crea circuite complexe împreună cu celelalte aplicații prezentate.

Fiecare lucrare a fost gândită și structurată astfel încât să cuprindă toate informațiile necesare parcurgerii ei, fără ca cititorul să fie nevoit să consulte lucrările precedente, cu riscul de a repeta anumite elemente deja prezentate în acestea, în ideea păstrării unei cursivități și a ușurinței în realizarea fiecărei aplicații.

Autorii acestei cărți au utilizat pentru testarea aplicațiilor placa de dezvoltare Arduino Uno R3. Cititorul trebuie să țină cont de faptul că utilizarea altor plăci din familia Arduino poate necesita modificări ale unor elemente de hardware sau software, poate duce la nefuncționarea aplicațiilor sau, posibil, la defectarea plăcii de dezvoltare sau modulelor. Toate aplicațiile practice prezentate au fost realizate și testate pentru verificarea funcționării acestora, totuși, autorii nu își asumă responsabilitatea în cazul apariției unor erori sau omisiuni, și nici pentru pagubele rezultate în urma utilizării informațiilor din această carte.

# CUPRINS

<b>PREFATĂ</b> .....	<b>3</b>
<b>CUPRINS</b> .....	<b>5</b>
<b>Lucrarea 1. Principii de bază în utilizarea unei plăci de dezvoltare</b> .....	<b>9</b>
1. Descrierea lucrării .....	9
1.1. Obiectivele lucrării.....	9
1.2. Descriere teoretică.....	9
2. Componente hardware.....	15
3. Componente software.....	18
4. Aplicația 1. LED comandat de un buton .....	20
4.1. Realizarea montajului electronic.....	20
4.2. Schema logică și secvența de cod .....	21
5. Aplicația 2. LED clipitor .....	23
5.1. Realizarea montajului electronic.....	23
5.2. Schema logică și secvența de cod .....	24
6. Aplicația 3. LED pulsator .....	25
6.1. Realizarea montajului electronic.....	25
6.2. Schema logică și secvența de cod .....	26
7. Aplicația 4. Utilizarea unui port analogic.....	28
7.1. Realizarea montajului electronic.....	28
7.2. Schema logică și secvența de cod .....	29
8. Aplicația 5. Citirea gradului de apăsare .....	31
8.1. Realizarea montajului electronic.....	31
8.2. Schema logică și secvența de cod .....	32
9. Exerciții suplimentare și concluzii.....	33
Bibliografie.....	34
<b>Lucrarea 2. Sistem de semaforizare pentru trecere de pietoni</b> .....	<b>37</b>
1. Descrierea lucrării .....	37
1.1. Obiectivele lucrării.....	37
1.2. Descriere teoretică.....	37

2.	Componente hardware.....	40
3.	Componente software.....	44
4.	Aplicație. Sistem de semaforizare pentru trecere de pietoni .....	46
4.1.	Realizarea montajului electronic.....	46
4.2.	Schema logică și secvența de cod .....	48
5.	Exerciții suplimentare și concluzii.....	52
6.	Bibliografie .....	53
<b>Lucrarea 3. Măsurarea parametrilor mediului înconjurător utilizând senzori analogici.....</b>		<b>55</b>
1.	Descrierea lucrării .....	55
1.1.	Obiectivele lucrării.....	55
1.2.	Descriere teoretică.....	55
2.	Componente hardware.....	57
3.	Componente software.....	62
4.	Aplicație. Măsurarea temperaturii și a umidității relative .....	64
4.1.	Realizarea montajului electronic.....	64
4.2.	Schema logică și secvența de cod .....	66
5.	Evaluare .....	70
6.	Bibliografie .....	71
<b>Lucrarea 4. Măsurarea parametrilor mediului înconjurător utilizând senzori digitali.....</b>		<b>73</b>
1.	Descrierea lucrării .....	73
1.1.	Obiectivele lucrării.....	73
1.2.	Descriere teoretică.....	73
2.	Componente hardware.....	75
3.	Componente software.....	81
3.1.	Funcții, comenzi și simboluri utilizate.....	81
4.	Aplicația 1. Măsurarea umidității și temperaturii .....	84
4.1.	Realizarea montajului electronic.....	84
4.2.	Schema logică și secvența de cod .....	86
5.	Aplicația 2. Măsurarea presiunii atmosferice și temperaturii.....	88
5.1.	Realizarea montajului electronic.....	88
5.2.	Schema logică și secvența de cod .....	90



---

6. Exerciții suplimentare și concluzii.....	93
7. Bibliografie.....	94
<b>Lucrarea 5. Măsurarea distanței și a proximității.....</b>	<b>95</b>
1. Descrierea lucrării.....	95
1.1. Obiectivele lucrării.....	95
1.2. Descriere teoretică.....	95
2. Componente hardware.....	99
3. Componente software.....	107
4. Aplicația 1. Măsurarea distanței utilizând un traductor cu unde ultrasonice.....	110
4.1. Realizarea montajului electronic.....	110
4.2. Schema logică și secvența de cod.....	112
5. Aplicația 2. Măsurarea proximității utilizând un traductor cu unde infraroșii.....	115
5.1. Realizarea montajului electronic.....	115
5.2. Schema logică și secvența de cod.....	117
6. Exerciții suplimentare și concluzii.....	120
7. Bibliografie.....	121
<b>Lucrarea 6. Măsurarea nivelului de iluminare.....</b>	<b>123</b>
1. Descrierea lucrării.....	123
1.1. Obiectivele lucrării.....	123
1.2. Descriere teoretică.....	123
2. Componente hardware.....	129
3. Componente software.....	137
4. Aplicația 1. Măsurarea nivelului de iluminare cu un senzor analogic.....	139
4.1. Realizarea montajului electronic.....	139
4.2. Schema logică și secvența de cod.....	140
5. Aplicația 2. Măsurarea nivelului de iluminare cu un senzor digital.....	141
5.1. Realizarea montajului electronic.....	141
5.2. Schema logică și secvența de cod.....	143
6. Aplicația 3. Măsurarea nivelului de iluminare cu un senzor RGB digital.....	145
6.1. Realizarea montajului electronic.....	145
6.2. Schema logică și secvența de cod.....	147

7. Exerciții suplimentare și concluzii.....	149
8. Bibliografie.....	150
<b>Lucrarea 7. Măsurarea tensiunii și intensității curentului electric.....</b>	<b>153</b>
1. Descrierea lucrării.....	153
1.1. Obiectivele lucrării.....	153
1.2. Descriere teoretică.....	153
2. Componente hardware.....	155
3. Componente software.....	163
4. Aplicația 1. Măsurarea tensiunii și a intensității curentului.....	165
4.1. Realizarea montajului electronic.....	165
4.2. Schema logică și secvența de cod.....	167
4.3. Mod de lucru și calibrare.....	170
5. Aplicația 2. Utilizarea unei tensiuni de referință externe.....	172
5.1. Realizarea montajului electronic.....	172
5.2. Schema logică și secvența de cod.....	173
5.3. Mod de lucru și calibrare.....	175
6. Exerciții suplimentare și concluzii.....	175
7. Bibliografie.....	176
<b>Lucrarea 8. Utilizarea unui ceas în timp real.....</b>	<b>177</b>
1. Descrierea lucrării.....	177
2. Componente hardware.....	178
3. Componente software.....	184
4. Realizarea montajului electronic.....	186
5. Schema logică și secvența de cod.....	188
6. Exerciții suplimentare și concluzii.....	193
7. Bibliografie.....	194

# Lucrarea 1. Principii de bază în utilizarea unei plăci de dezvoltare

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Cunoașterea principalelor caracteristici ale plăcii Arduino Uno R3.
- Cunoașterea modului de utilizare/programare a porturilor de intrare/ieșire analogice sau digitale al plăcii Arduino Uno R3.
- Crearea și testarea de circuite simple ce utilizează senzori și traductoare.

### 1.2. Descriere teoretică

#### ✓ **Introducere**

**Arduino Uno R3** este o placă open-source de dezvoltare ce folosește un microcontroler ATMEL pe 8 biți de tip ATmega328. Cu ajutorul ei se pot realiza nenumărate proiecte electronice cu aplicații în aproape orice domeniu, beneficiind și de suport din partea unei comunități online din ce în ce mai mare.

Caracteristicile unei plăci Arduino sunt următoarele [1]:

- Tensiune de operare: 5 V
- Tensiune de alimentare recomandată: 7-12 V
- Port USB 2.0
- Pini intrare/ieșire digitală: 14 (dintre care 6 pot funcționa în mod PWM)
- Pini intrare analogică: 6
- Curent de ieșire pentru un pin digital: 40 mA
- Curent de ieșire pentru pinul de 3,3 V: 50 mA
- Memorie flash (ATmega328): 32 KB (0,5KB pentru boot loader)

- Memorie SRAM (ATmega328): 2 KB
- Memorie EEPROM (ATmega328): 1 KB
- Frecvența ceasului: 16 MHz

Placa poate fi alimentată fie prin portul USB, fie dintr-o sursă externă prin conectorul de alimentare. Poziționarea și denumirea tuturor porturilor și pinilor plăcii se pot vedea în figura următoare.



Figura 1. Placa de dezvoltare Arduino Uno R3

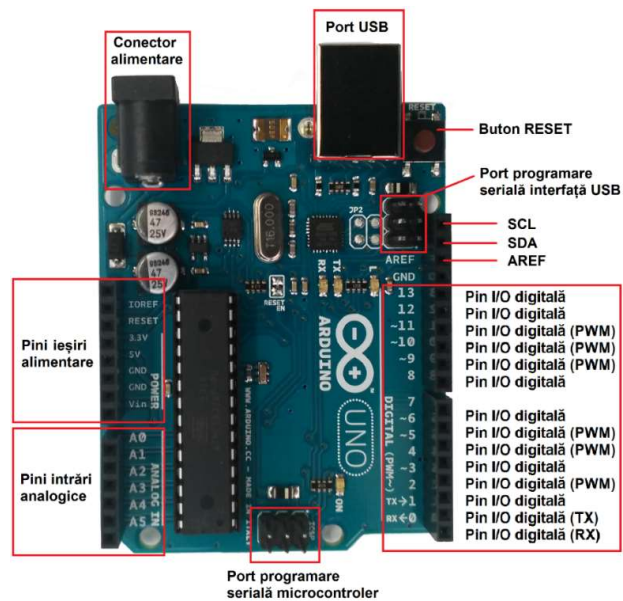


Figura 2. Porturi ale plăcii de dezvoltare Arduino Uno R3

În cadrul lucrării vor fi utilizate următoarele porturi:

- Pin de alimentare 5 V: tensiune furnizată de sursa internă a plăcii (a se evita utilizarea acestui pin pentru alimentarea unor module externe mari consumatoare de curent).
- GND: pin de masă.
- Pin intrare analogică
- Pin intrare/ieșire digitală: 5 V, maxim 40 mA.
- Pin ieșire digitală PWM: 5 V, maxim 40 mA. PWM (engl. Pulse-Width Modulation) presupune variația controlată a formei tensiunii de ieșire prin schimbări rapide ale valorii logice din 1 în 0 (frecvența semnalului este de aprox. 490 Hz), în funcție de un factor de umplere (valoarea acestuia poate lua valori între 0 și 255). Astfel se poate genera un semnal de putere variabilă și simula obținerea unor tensiuni analogice între 0 și 5 V folosind un port digital [2].

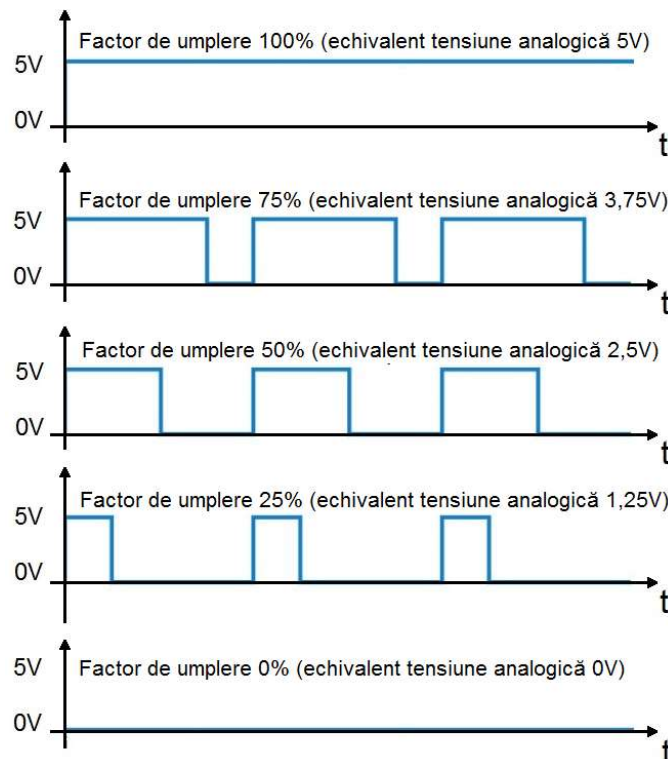


Figura 3. Variații ale tensiunii la o ieșire de tip PWM

Placa Arduino poate fi extinsă prin atașarea unor module numite *shield* ce se pot conecta direct la pini externi ai plăcii (module GPS, Wi-Fi, LCD - Figura 4, touchscreen, control motoare, etc.), aceștia putând fi accesați în continuare deoarece majoritatea modulelor *shield* au extensii ale lor.



Figura 4. Utilizarea unui *shield* LCD pe o placă de tip Arduino Uno R3

Pentru realizarea montajelor electronice ce necesită componente externe (și dacă nu se dorește utilizarea componentelor montate pe un cablaj, de exemplu în cazurile prototipurilor) se poate folosi o placă de testare numită *breadboard* (Figura 5 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

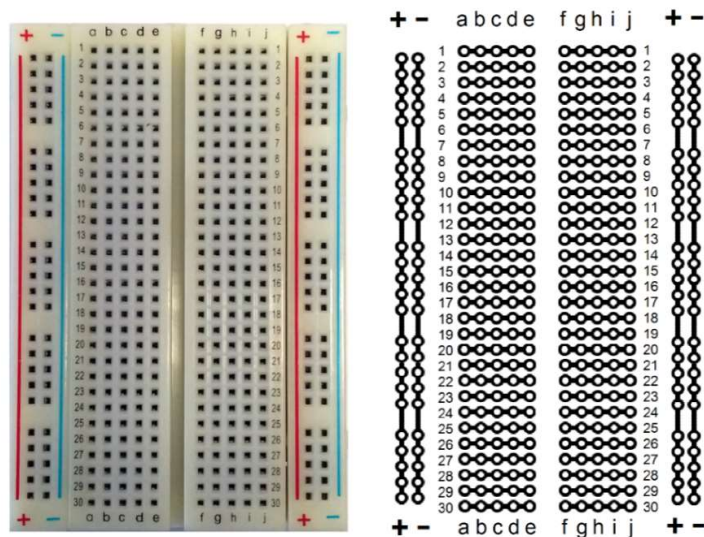


Figura 5. *Breadboard*-ul și conexiunile interne

**Programarea microcontrolerului** se face într-un limbaj derivat din C++. Programul necesar se numește Arduino IDE și se poate descărca de pe web-situl producătorului [3]. După conectarea plăcii și instalarea și lansarea programului asigurați-vă că este selectată placa Arduino și portul corespunzător (în meniul *Tools*, sub-meniurile *Board* și *Port*).

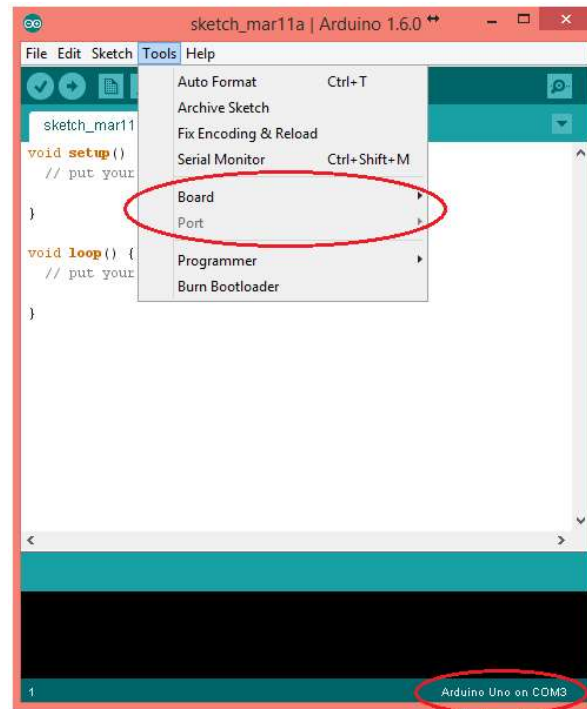


Figura 6. Interfața grafică a programului Arduino IDE

Cele mai utilizate butoane din aplicația software vor fi butonul de compilare (verificarea de erori a programului), butonul de încărcare (încărcarea programului în microcontrolerul plăcii Arduino) și butonul pentru *Serial Monitor* (afișarea datelor trimise de placă către calculator).

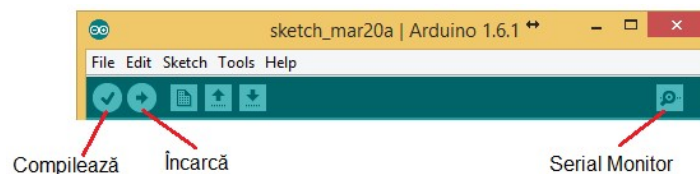


Figura 7. Butoane și meniuri ale programului Arduino IDE

### ✓ **Descrierea aplicațiilor**

În cadrul acestei lucrări vor fi realizate cinci aplicații practice.

#### **Aplicația 1. LED comandat de un buton**

Această aplicație va folosi un pin de intrare digitală pentru conectarea unui buton și un pin de ieșire digitală pentru conectarea unui LED. Scopul este de a realiza schimbarea stării LED-ului din stins în aprins în momentul sesizării apăsării butonului.

#### **Aplicația 2. LED clipitor**

Această aplicație va folosi un pin de ieșire digitală pentru conectarea unui LED. Scopul este de a aprinde și stinge succesiv LED-ul, pe o durată de timp nelimitată, pre-definind perioadele de timp cât LED-ul stă aprins sau stins.

#### **Aplicația 3. LED pulsator**

Această aplicație va folosi un pin de ieșire digitală PWM pentru conectarea unui LED. Scopul este de a aprinde și stinge LED-ul, pe o durată de timp nelimitată, variind intensitatea luminii emise de acesta între 0 și maxim într-o anumită perioadă de timp predefinită.

#### **Aplicația 4. Utilizarea unui port analogic**

Această aplicație va folosi un pin de intrare analogică pentru conectarea unui potențiomtru în configurație de divizor rezistiv de tensiune reglabil. Scopul este de a afișa pe ecran (pe *Serial Monitor*) valorile digitale furnizate de placa Arduino proporțional cu tensiunea de la intrarea analogică.






#### **Aplicația 5. Citirea gradului de apăsare**

Această aplicație va folosi un pin de intrare analogică pentru conectarea unui traductor de presiune. Scopul este de a afișa pe ecran (pe *Serial Monitor*) valorile măsurate de traductor proporțional cu apăsarea exercitată asupra acestuia, la intervale de timp predefinite.



## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

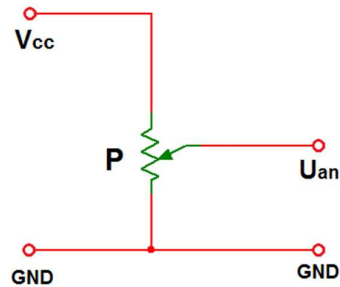
Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Modul LED</i>	LED + rezistor	1	
<i>Fir de legătură</i>	Tată-Tată	5	
<i>Modul Buton</i>	Buton + rezistor	1	
<i>Potențiomtru</i>	50 k $\Omega$ liniar	1	
<i>Modul Traductor de presiune rezistiv</i>	FSR 406 + rezistor	1	

### ✓ Observații

**Modulul LED** conține pe lângă un LED și un rezistor corect dimensionat (modul de dimensionare este prezentat în Lucrarea 2).

**Modulul Buton** este folosit pentru a sesiza apăsarea și a comanda, în acest caz, starea unui LED. Acest traductor se poate înlocui și cu orice alt tip de buton împreună cu o rezistență de 10 k $\Omega$ , așa cum se va vedea în paragraful următor.

**Potențiometrul** este folosit pentru a crea un divizor rezistiv de tensiune reglabil, cu scopul de a aplica pe intrarea analogică a plăcii Arduino o tensiune variabilă în domeniul 0 ... V<sub>CC</sub>. Ca urmare, valoarea măsurată de placă va fi disponibilă sub forma unei valori digitale ce variază în domeniul 0 ... 1023, proporțional cu tensiunea aplicată.



**Figura 8. Divizor rezistiv de tensiune reglabil**

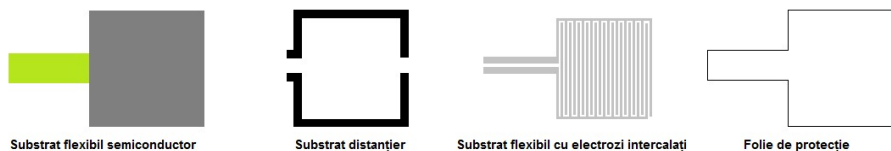
Domeniul de valori digitale se obține în urma conversiei tensiunii analogice într-un semnal digital, lucru realizat de placa Arduino cu ajutorul convertorului analogic-digital intern pe 10 biți.

Valoarea tensiunii analogice se poate măsura cu un voltmetru sau se poate aproxima în funcție de valoarea digitală furnizată de placă, astfel:

$$U_{an} = V_{CC} \frac{val\_dig}{1023} \quad (1)$$

Tensiunea de alimentare  $V_{CC}$  este teoretic de 5 V. Deoarece valoarea reală diferă, va fi necesară o calibrare. Astfel, se va măsura tensiunea  $V_{CC}$  furnizată de placă, folosind un voltmetru, iar valoarea măsurată se va scrie în program la declararea variabilei  $V_{CC}$ .

**Modulul traductor de presiune rezistiv** sesizează gradul de apăsare, bazându-se pe utilizarea unui rezistor sensibil la presiune (FSR 406), valoarea măsurată de placa Arduino fiind disponibilă sub forma unei valori digitale ce variază între 0 și 1023.



**Figura 9. Construcția internă a senzorului de presiune**

Senzorul de presiune este realizat din trei substraturi (vezi Figura 9), având o rezistență foarte mare între electrozi ( $>10 \text{ M}\Omega$ ) atunci când nu se exercită nici o presiune. Creșterea presiunii aplicate senzorului are ca efect

realizarea unui contact electric între substraturile conductoare și determină astfel scăderea valorii rezistenței la bornele acestuia (vezi Figura 10). Valoarea rezistenței depinde nu numai de forța aplicată ci și de flexibilitatea, dimensiunile și forma obiectului care aplică presiunea [4].

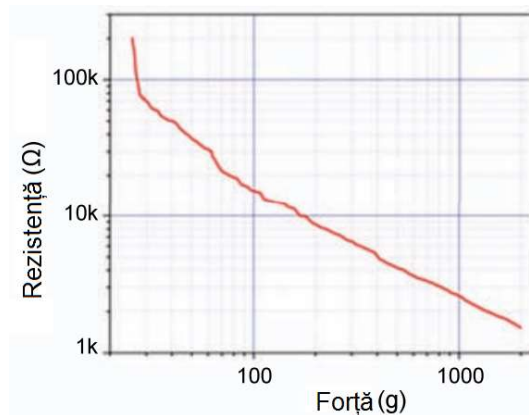


Figura 10. Variația rezistenței sensorului de presiune în raport cu forța de apăsare [4]

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5\text{ V}$ .

Atât modulul traductor de presiune cât și modulul buton conțin, pe lângă senzor, și un rezistor R conectat între pinul de ieșire al modulului (OUT) și masă (GND).

Pentru buton R se numește rezistor de coborâre la 0 (engl. *Pull down* [5], vezi Figura 11) și are rolul de a păstra valoarea logică 0 la ieșirea modulului atunci când nu se exercită presiune pe senzor sau nu este apăsat butonul. Stabilirea unui nivel logic sigur (0 în acest caz) împiedică apariția aleatorie a unei valori 0 sau 1 la intrarea digitală a plăcii Arduino datorită unor eventuale zgomote electrice [6].

Pentru traductorul de presiune (Figura 11) R îndeplinește atât funcția de rezistor de coborâre la 0 (pentru cazul în care nu se exercită nici o forță asupra sensorului, deci la bornele acestuia rezistența este foarte mare) cât și de componentă a unui divizor rezistiv de tensiune (cu ajutorul acestuia se generează la ieșire o tensiune analogică proporțională cu rezistența sensorului în momentul exercitării unei forțe).

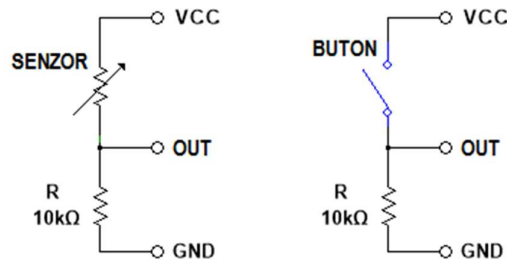


Figura 11. Utilizarea rezistorului R

### 3. Componente software

**int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

**const** are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

**unsigned int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, fără semn (de la 0 până la 65.535).

**boolean variabilă = valoare** stabilește o valoare pentru o variabilă de tip logic.

**float variabilă = valoare** stabilește o valoare pentru o variabilă de tip real în virgulă mobilă pe 32 de biți, cu semn (de la -3.4028235E+38 până la 3.4028235E+38). Numărul total de digiți afișați cu precizie este 6 – 7 (include toți digiții, nu doar cei de după virgulă).

**void setup()** este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

**void loop()** este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

`pinMode(pin, mod)` configurează pinul digital specificat ca intrare sau ca ieșire.

`if(condiție) {instrucțiune/i} else {instrucțiune/instrucțiuni}` testează îndeplinirea sau nu a unei condiții.

`for(inițializare, condiție, increment) {instrucțiune/ instrucțiuni }` repetă un bloc de instrucțiuni până la îndeplinirea condiției.

`digitalWrite(pin, valoare)` scrie o valoare HIGH sau LOW pinului digital specificat.

`digitalRead(pin)` citește valoarea pinului digital specificat.

`analogRead(pin)` citește valoarea pinului analogic specificat.

`analogWrite(pin, valoare)` scrie o valoare (între 0 și 255), a factorului de umplere pentru semnalul de tip PWM, pinului digital de tip PWM specificat.

`delay(ms)` pune în pauză programul pentru o durată de timp specificată în milisecunde.

`Serial.begin(viteză)` stabilește rata de transfer a datelor pentru portul serial în biți/secundă (BAUD).

`Serial.print(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial.

`Serial.println(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial, adăugând după datele afișate și trecerea la o linie nouă.

`==` semnifică *egal cu*.

## 4. Aplicația 1. LED comandat de un buton

### 4.1. Realizarea montajului electronic

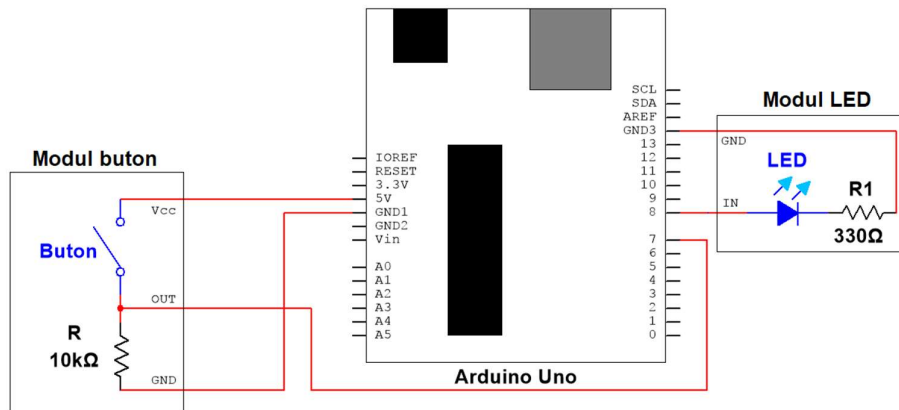


Figura 12. Schema de principiu pentru aplicația 1

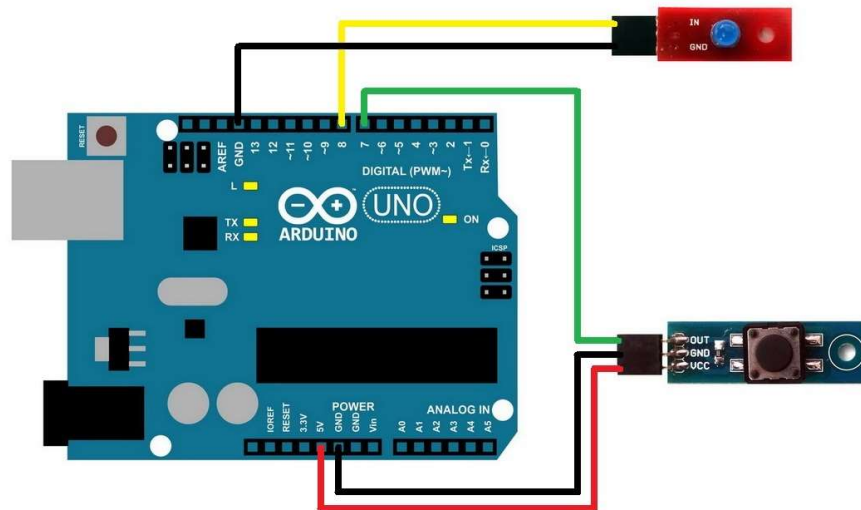
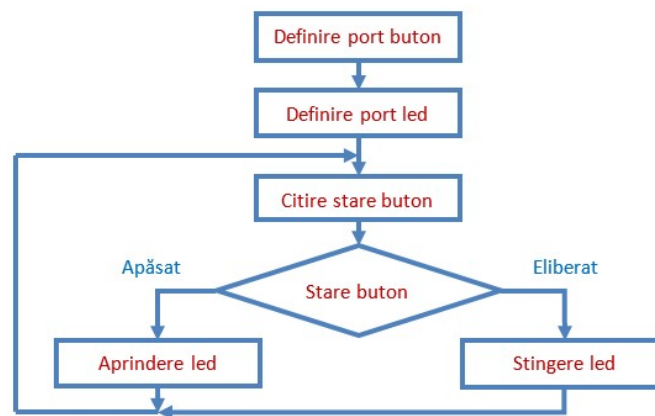


Figura 13. Realizarea conexiunilor electrice pentru aplicația 1

Se realizează următoarele conexiuni:

- Pinul digital 8 de pe placa Arduino se conectează cu un fir la pinul IN al *modulului LED*;
- Pinul GND (digital) de pe placa Arduino se conectează cu un fir la pinul GND al *modulului LED*;
- Pinul digital 7 de pe placa Arduino se conectează cu un fir la pinul OUT al *modulului Buton*;
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al *modulului Buton*;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul VCC al *modulului Buton*.

#### 4.2. Schema logică și secvența de cod



```
const int buton = 7;
//definirea variabilei buton corespunzătoare portului digital 7 unde va fi
conectat pinul OUT al modulului buton
const int led = 8;
//definirea variabilei led corespunzătoare portului digital 8 unde va fi
conectat pinul IN al modulului LED
void setup() {
  pinMode(buton, INPUT);
  //se declară pinul buton ca fiind de intrare
  pinMode(led, OUTPUT);
  //se declară pinul led ca fiind de ieșire
}

void loop() {
```

```
boolean stareButon = digitalRead(buton);
    //se declară variabila de tip boolean stareButon ce ia valoarea logică a
    stării pinului buton
if (stareButon == HIGH) {
    //daca starea pinului buton este 1 logic (buton apăsat)
    digitalWrite(led, HIGH);
    //atunci scrie valoarea 1 logic pe pinul led (aprinde led)
}
else {
    //altfel
    digitalWrite(led, LOW);
    //scrie valoarea 0 logic pe pinul led (stinge led)
}
}
```



## 5. Aplicația 2. LED clipitor

### 5.1. Realizarea montajului electronic

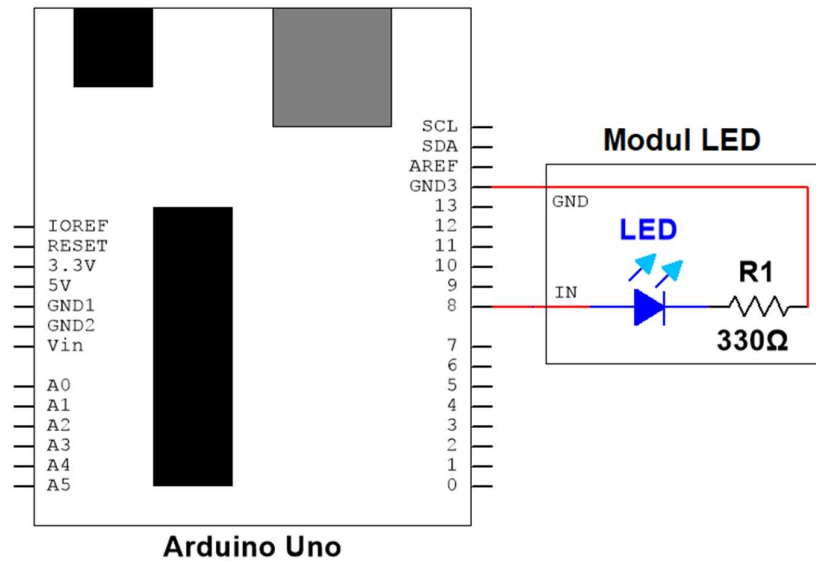


Figura 14. Schema de principiu pentru aplicația 2

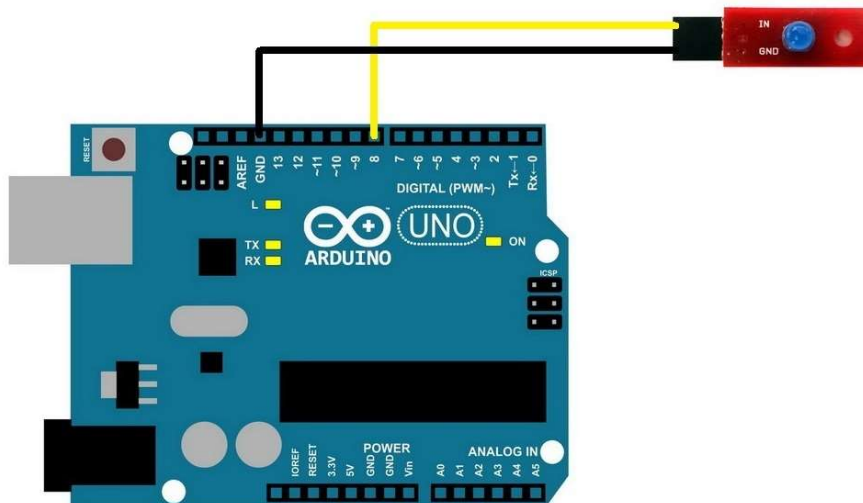
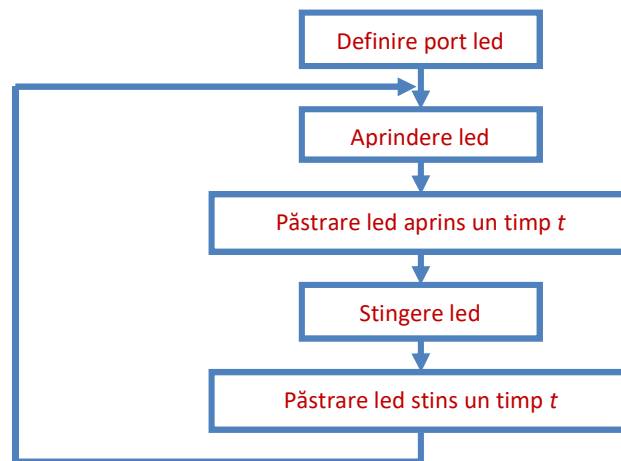


Figura 15. Realizarea conexiunilor electrice pentru aplicația 2

Se realizează următoarele conexiuni:

- Pinul digital 8 de pe placa Arduino se conectează cu un fir la pinul IN al *modulului LED*;
- Pinul GND (digital) de pe placa Arduino se conectează cu un fir la pinul GND al *modulului LED*.

## 5.2. Schema logică și secvența de cod



```
const int led = 8;
    //definirea variabilei led corespunzătoare portului digital 8 unde va fi
    //conectat pinul IN al modulului LED
void setup() {
    pinMode(led, OUTPUT);
    //se declară pinul led ca fiind de ieșire
}

void loop() {
    digitalWrite(led, HIGH);
    //scrie valoarea 1 logic pe pinul led (aprinde led)
    delay(1000);
    //întârzie 1000ms (păstrează led aprins 1 secundă)
    digitalWrite(led, LOW);
    //scrie valoarea 0 logic pe pinul led (stinge led)
    delay(1000);
    //întârzie 1000ms (păstrează led stins 1 secundă)
}
```

## 6. Aplicația 3. LED pulsator

### 6.1. Realizarea montajului electronic

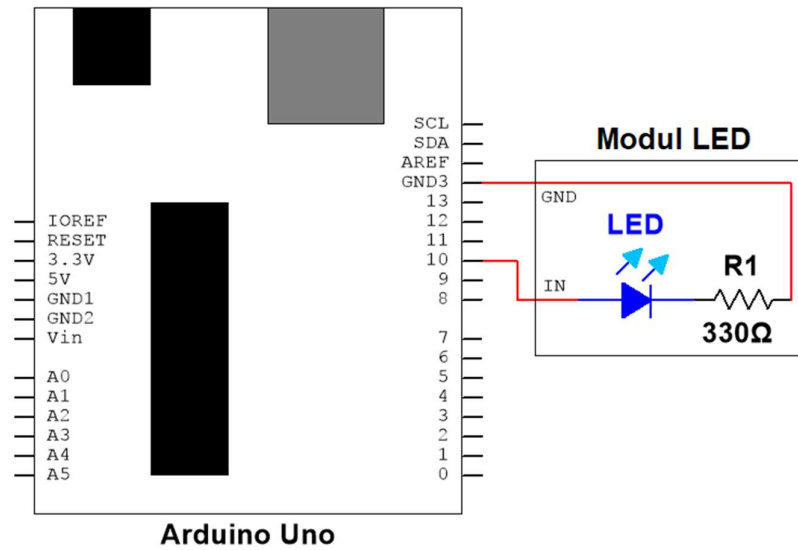


Figura 16. Schema de principiu pentru aplicația 3

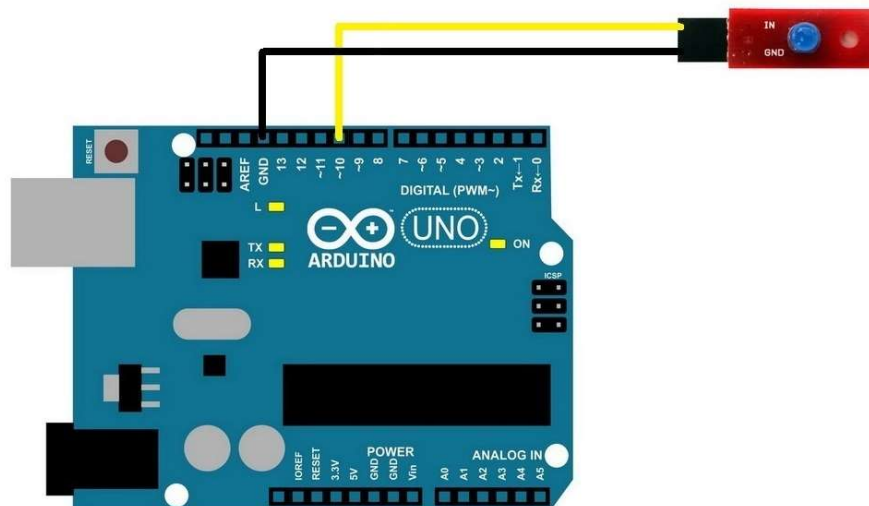
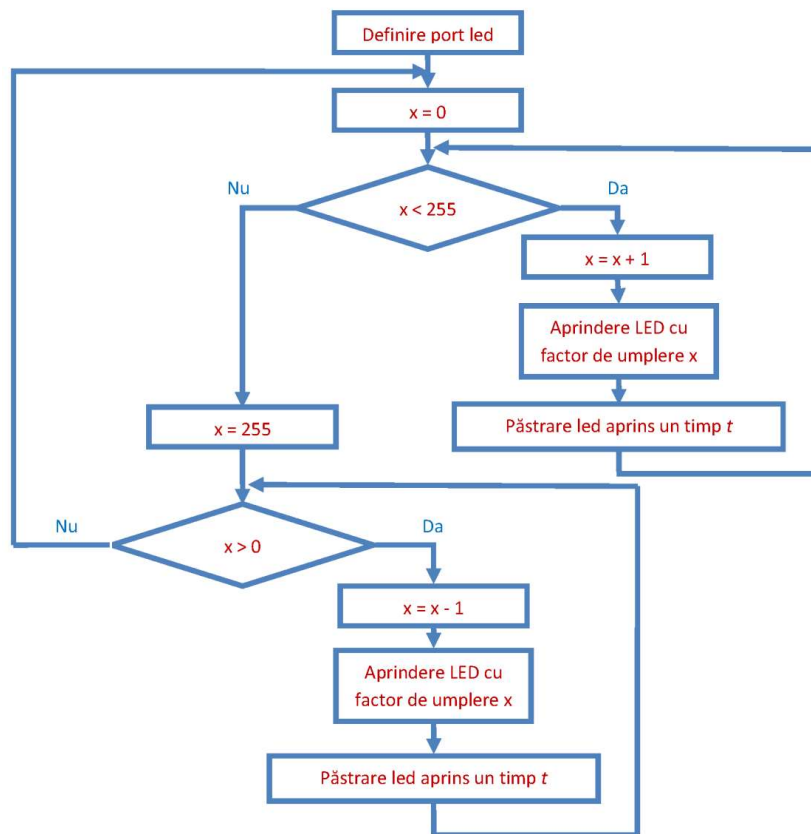


Figura 17. Realizarea conexiunilor electrice pentru aplicația 3

Se realizează următoarele conexiuni:

- Pinul digital 10 (PWM) de pe placa Arduino se conectează cu un fir la pinul IN al *modulului LED*;
- Pinul GND (digital) de pe placa Arduino se conectează cu un fir la pinul GND al *modulului LED*.

## 6.2. Schema logică și secvența de cod



```

const int led = 10;
//definirea variabilei led corespunzătoare portului digital PWM 10 unde
va fi conectat pinul IN al modulului LED

void setup() {
  pinMode(led, OUTPUT);
  //se declară pinul led ca fiind de ieșire

```

```
}  
  
void loop() {  
  for (int x=0; x<255; x=x+1) {  
    //variază valoarea lui x crescător de la 0 până la 254  
    analogWrite(led, x);  
    //scrie valoarea analogică cu factorul de umplere x pe pinul led  
    delay(20);  
    //întârzie 20ms  
  }  
  for (int x=255; x>0; x=x-1) {  
    //variază valoarea lui x descrescător de la 255 pana la 1  
    analogWrite(led, x);  
    //scrie valoarea analogică cu factorul de umplere x pe pinul led  
    delay(20);  
    //întârzie 20ms  
  }  
}
```

## 7. Aplicația 4. Utilizarea unui port analogic

### 7.1. Realizarea montajului electronic

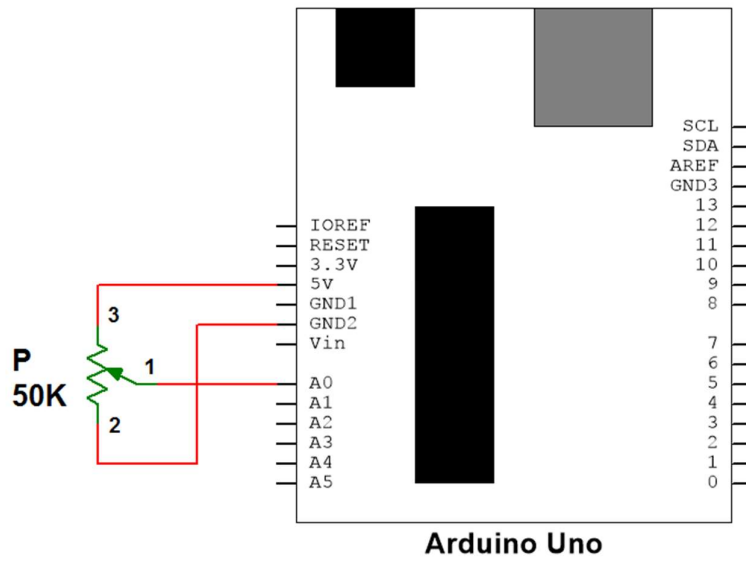


Figura 18. Schema de principiu pentru aplicația 4

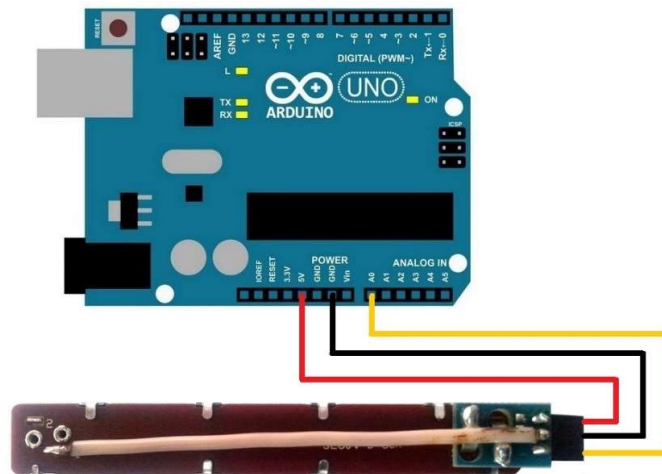
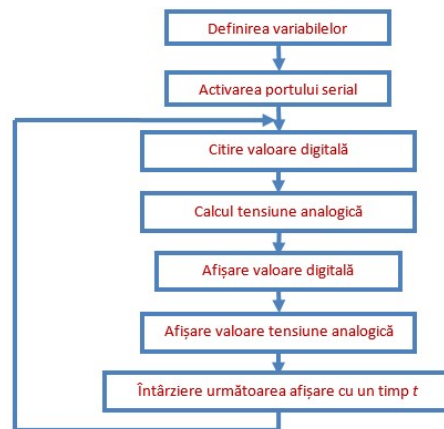


Figura 19. Realizarea conexiunilor electrice pentru aplicația 4

Se realizează următoarele conexiuni:

- Pinul analogic A0 de pe placa Arduino se conectează cu un fir la pinul 1 (cursorul) potențiometrului;
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul 2 al potențiometrului;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul 3 al potențiometrului.

## 7.2. Schema logică și secvența de cod



```
float Vcc = 5;
    //definirea variabilei U corespunzătoare tensiunii de alimentare

void setup() {
    Serial.begin(9600);
    //activează ieșirea portului serial cu rata de 9600 baud
}

void loop() {
    const int val_dig = analogRead(0);
    //se declară variabila constantă de tip întreg val_dig ce ia valoarea citită la
    intrarea analogică A0
    float Uan = Vcc*val_dig/1023;
    //calculul tensiunii analogice de la intrarea A0 – vezi formula (1)
    Serial.print("Valoare digitala: ");
    // afișează pe monitorul serial textul din paranteză
    Serial.println(val_dig);
}
```

```
    //afișează pe monitorul serial valoarea digitală citită
    Serial.print("Tensiune analogica: ");
    //afișează pe monitorul serial textul din paranteză
    Serial.print(Uan,3);
    //afișează pe monitorul serial valoarea calculată a tensiunii analogice, cu
    //trei zecimale
    Serial.println(" V");
    //afișează pe monitorul serial textul din paranteză
    Serial.println();
    //afișează pe monitorul serial o linie fără conținut
    delay(1000);
    //întârzie următoarea afișare cu 1000 ms
}
```



## 8. Aplicația 5. Citirea gradului de apăsare

### 8.1. Realizarea montajului electronic

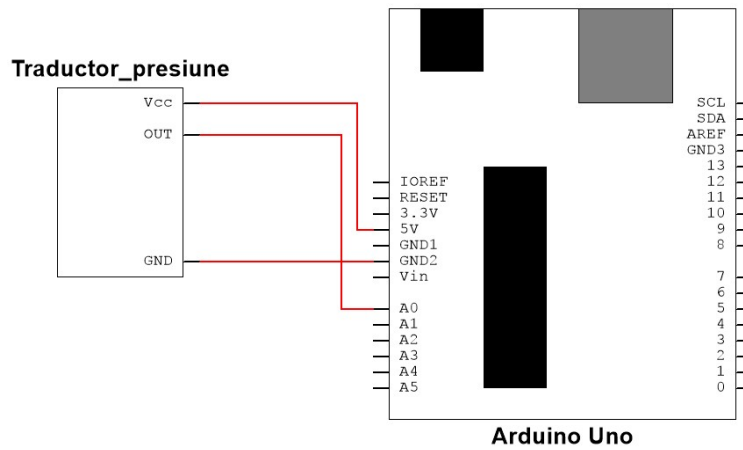


Figura 20. Schema de principiu pentru aplicația 5

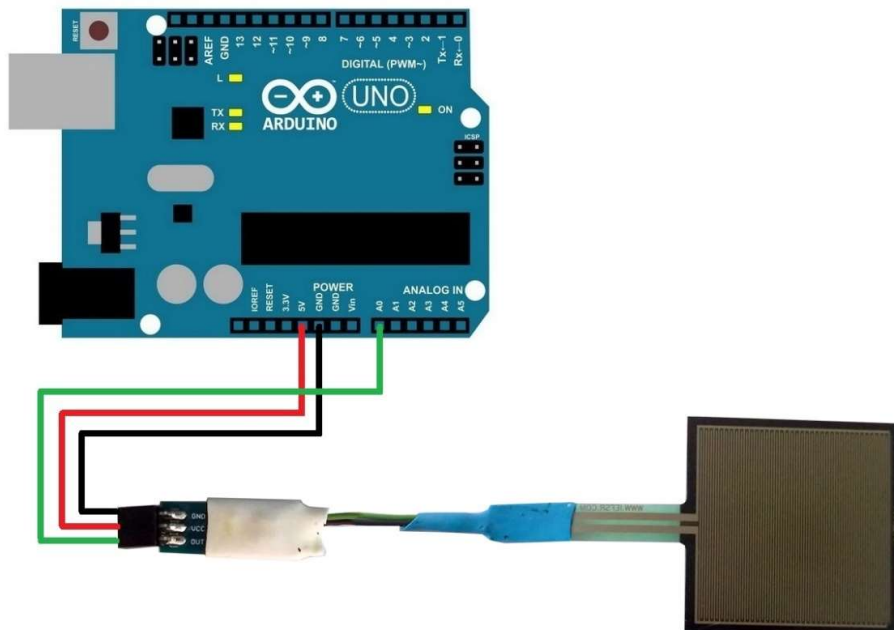
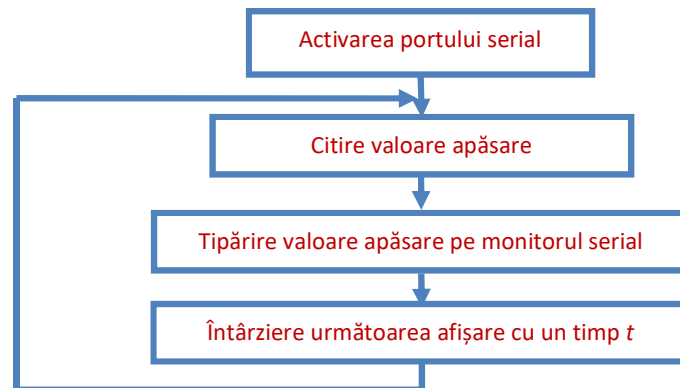


Figura 21. Realizarea conexiunilor electrice pentru aplicația 5

Se realizează următoarele conexiuni:

- Pinul analogic A0 de pe placa Arduino se conectează cu un fir la pinul OUT al *modulului traductor de presiune*;
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al *modulului traductor de presiune*;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul VCC al *modulului traductor de presiune*.

## 8.2. Schema logică și secvența de cod



```
void setup() {  
  Serial.begin(9600);  
  //activează ieșirea portului serial cu rata de 9600 baud  
}  
  
void loop() {  
  const int valoareApasare = analogRead(A0);  
  //se declară variabila constantă de tip întreg valoareApasare ce ia valoarea  
  citită la intrarea analogică A0  
  Serial.print("Valoare apăsare: ");  
  //afișează pe monitorul serial textul din paranteză  
  Serial.println(valoareApasare, DEC);  
  //afișează pe monitorul serial valoarea citită, codată în sistem zecimal  
  delay(1000);  
  //întârzie următoarea afișare cu 1000 ms  
}
```

## 9. Exerciții suplimentare și concluzii

### Aplicația 1. LED comandat de un buton

1. Să se modifice secvența de cod astfel încât la apăsarea butonului led-ul să se stingă, iar la eliberarea lui led-ul să se aprindă.
2. Să se modifice secvența de cod astfel încât la o apăsare a butonului led-ul să treacă în stare opusă stării actuale (să se stingă dacă era aprins, respectiv să se aprindă dacă era stins).

### Aplicația 2. LED clipitor

1. Să se calculeze frecvența cu care clipește led-ul.
2. Să se modifice secvența de cod astfel încât led-ul să clipească cu frecvența de 10Hz.
3. Să se modifice secvența de cod astfel încât led-ul să stea aprins 2 secunde și stins 0,5 secunde.

### Aplicația 3. LED pulsator

1. Să se calculeze perioada de timp la care se repetă un ciclu led aprins/led stins (bucla *loop*).
2. Să se modifice secvența de cod astfel încât perioada de timp la care se repetă un ciclu led aprins/led stins să fie de 2,55 secunde.

### Aplicația 4. Utilizarea unui port analogic

1. Să se afișeze pe monitorul serial poziția cursorului potențiometrului în procente.
2. Să se conecteze un LED pe una din ieșirile digitale PWM și să se varieze intensitatea lunii emise de acesta cu ajutorul potențiometrului.

### Aplicația 5. Citirea gradului de apăsare

1. Să se introducă în aplicație un led care să se aprindă atunci când valoarea apăsării este mai mică decât 50 și să se stingă atunci când este mai mare sau egală cu 50.
2. Să se introducă în aplicație un led care să clipească dacă valoarea apăsării este mai mare decât 100 și să se stingă atunci când este mai mică sau egală cu 100.

Placa de dezvoltare Arduino are intrări/ieșiri digitale și analogice ce permit conectarea diferitelor module externe care pot fi comandate sau care pot genera comenzi. În cazul acestei lucrări modulele de comandă sunt: Modulul Buton, Potențiometrul și Modulul traductor de presiune, acestea având rolul de a transmite comenzi către placa de dezvoltare și de a constitui o interfață om-mașină cu sistemul Arduino. Modulul comandat este modulul LED, iar acesta, prin intermediul plăcii de dezvoltare, Arduino va primi comenzi de la utilizatori.

Modulul Buton va genera comenzi digitale: apăsarea butonului va genera 1 logic (+5 V) iar starea opusă va genera 0 logic (0 V – masă). Potențiometrul și modulul traductor de presiune vor genera comenzi analogice. Ieșirea fiecăruia va genera o tensiune variabilă iar aceasta va fi conectată la una dintre intrările analogice ale plăcii de dezvoltare Arduino. Semnalul analogic de comandă (sau semnalul achiziționat de către placa de dezvoltare Arduino) va fi convertit în semnal digital de către aceasta pentru a putea fi prelucrat. Conversia este făcută de un convertor analogic-digital (ADC).

Semnalele digitale sau analogice achiziționate pe una dintre liniile digitale sau analogice ale plăcii de dezvoltare Arduino vor fi convertite în valori digitale, iar aceste valori vor fi utilizate ca variabile în programele scrise pentru placa de dezvoltare.

Anumite variabile cu care lucrează programele dezvoltate pentru placa Arduino vor trebui trimise către liniile digitale sau de tip PWM de ieșire. Acestea vor avea un semnal electric corespondent la ieșirea/linia către care au fost trimise.

## **Bibliografie**

[1] „*Arduino Uno*,” 11.03.2015.

<http://www.arduino.cc/en/Main/arduinoBoardUno>.

[2] „*Arduino PWM*,” 11.03.2015. <http://www.arduino.cc/en/Tutorial/PWM>.

- [3] „*Arduino IDE*,” 11.03.2015. <http://arduino.cc/en/Main/Software>.
- [4] „*Interlink Electronics - FSR® 400 Series Data Sheet*,” 25.03.2015.  
[http://www.interlinkelectronics.com/datasheets/Datasheet\\_FSR.pdf](http://www.interlinkelectronics.com/datasheets/Datasheet_FSR.pdf).
- [5] „*Arduino Playground*,” 10.03.2015.  
<http://playground.arduino.cc/CommonTopics/PullUpDownResistor>.
- [6] **M. McRoberts**, *Beginning Arduino, 2nd Edition*, Apress, 2013.



# Lucrarea 2. Sistem de semaforizare pentru trecere de pietoni

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează senzori și traductoare.
- Realizarea unei aplicații practice de implementare a unui sistem de semaforizare pentru trecere de pietoni.

### 1.2. Descriere teoretică

#### ✓ Introducere

Scopul sistemelor de semaforizare este de a da prioritate de trecere unei anumite categorii de participanți la trafic. Cei mai vulnerabili dintre aceștia sunt pietonii, iar semaforizarea trecerilor pentru aceștia este necesară nu numai în intersecții ci și zone cu căi rutiere aglomerate sau în care un număr mare de pietoni traversează des strada (centre comerciale, școli, etc.).

Fiecare sistem de semaforizare este caracterizat de o succesiune de faze care însumate formează un ciclu de semaforizare (orice secvență completă a indicației semaforului sau intervalul de timp de la afișarea culorii verde pentru o fază până la afișarea culorii verde pe faza următoare [1]).

Alocarea timpilor pentru fazele de semaforizare trebuie să permită pietonilor să traverseze strada dintr-o singură deplasare, fără ca aceștia să fie nevoiți să se oprească sau să se întoarcă.

Succesiunea fazelor de semaforizare și alocarea timpilor pentru acestea diferă în funcție de țară, tipul trecerii de pietoni, lungimea acesteia, caracteristici locale specifice, etc. Un exemplu similar cu semaforizarea folosită în România se poate găsi în tabelul următor [2]:

Faza	Semafor vehicule	Semafor pietoni	Durață
A	Verde	Roșu	20-60 secunde
B	Galben	Roșu	3 secunde
C	Roșu	Roșu	1-3 secunde
D	Roșu	Verde	4-7 secunde (+2 secunde)
E	Roșu	Verde clipitor	0-2 secunde
F	Galben clipitor	Verde clipitor	6-18 secunde
G	Galben clipitor	Roșu	1-2 secunde

O metodă de îmbunătățire a siguranței și fluidizare a traficului în zona trecerilor pentru pietoni (adaptarea timpilor de traversare, reducerea timpilor de așteptare) o reprezintă implementarea de sisteme de semaforizare la care apariția indicației verde pentru pietoni se face numai pe bază de comandă, eliminând astfel ciclurile de semaforizare pentru pietoni pe durata cărora nu intenționează nimeni să traverseze.

Cea mai utilizată soluție pentru preluarea unei comenzi de la pietoni este folosirea unui buton de comandă. Se pot folosi însă și diverse tipuri de senzori de detecție ce nu necesită intervenție din partea pietonilor, existând anumite categorii de pietoni ce pot avea probleme în utilizarea butoanelor de comandă (copii, bătrâni, persoane cu dizabilități). În plus, pe baza datelor furnizate de senzori, se poate stabili când toți pietonii au traversat, astfel încât să se poată realiza o micșorare a timpului de verde pentru aceștia.

Aceste tipuri de senzori pot fi:

- Senzori de presiune amplasați pe trotuar, înainte de trecerea pentru pietoni (pot avea orice formă sau culoare, pot face diferența între pietoni și alte greutăți statice cum ar fi gheața sau zăpada, folosesc tehnologii bazate pe bucle inductive, putând fi conectate cu ușurință la sistemele de management al traficului), cum ar fi SMARTPED [3] sau PEDXPAD [4].
- Camere video și software pentru detecția pietonilor (C-Walk / SafeWalk [5]) ce supraveghează anumite zone predefinite și sunt capabile să facă diferența între pietoni care traversează, așteaptă să traverseze sau se apropie de trecerea de pietoni.

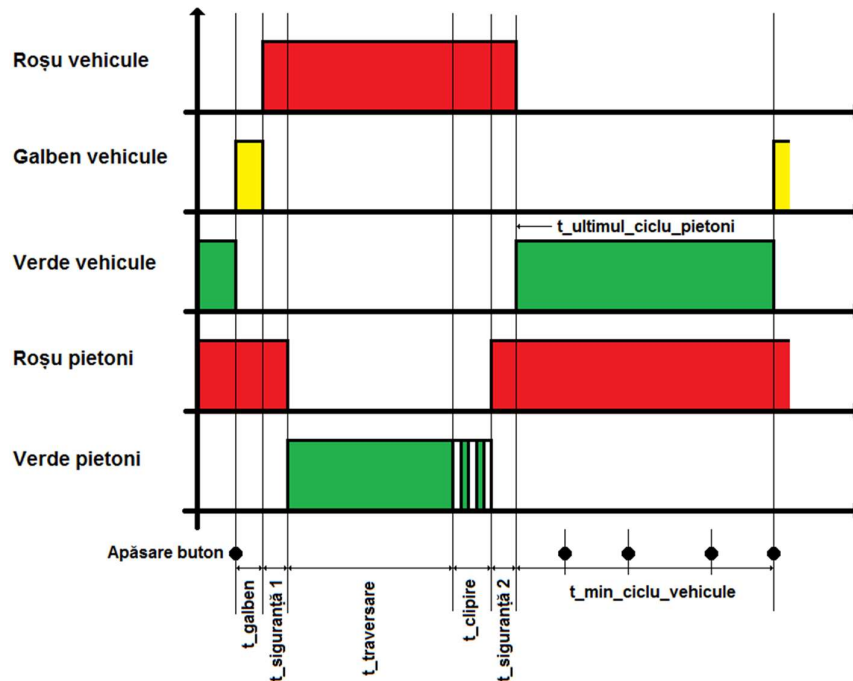


✓ **Descrierea aplicației**

Scopul acestei aplicații este de a realiza un circuit electronic care să comande un sistem de semaforizare a unei treceri pentru pietoni, ce va include un semafor pentru vehicule, unul pentru pietoni și o metodă de comandă a schimbării indicației semafoarelor, la cererea pietonilor. Pentru aceasta se va folosi fie un buton pe care pietonul trebuie să-l apese, fie un senzor de presiune care să detecteze prezența acestuia. Aplicația va permite și schimbarea cu ușurință a timpilor de semaforizare.

Sucesiunea operațiilor se poate observa în Figura 1.

Astfel, semaforul va afișa în mod continuu culoarea verde pentru vehicule și culoarea roșie pentru pietoni, până la apariția unei comenzi din partea pietonilor. În acel moment culoarea verde pentru vehicule se schimbă în galben și apoi în roșu.



**Figura 1. Succesiunea culorilor la semafoare**

După un timp de siguranță, necesar asigurării eliberării intersecției de către vehicule, se aprinde culoarea verde pentru pietoni. După terminarea

timpului de traversare prestabilit, culoarea verde pentru pietoni clipește o perioadă de timp, apoi se stinge și se aprinde culoarea roșie.

După un timp de siguranță, necesar asigurării eliberării intersecției de către pietoni, se aprinde culoarea verde pentru vehicule.






Este necesară și implementarea unui timp minim impus al ciclului de semaforizare pentru vehicule, astfel încât cererile pietonilor să nu fie deservite prea des una după cealaltă, lăsând astfel prea puțin timp circulației vehiculelor.




Pentru aplicația de laborator vor fi folosite LED-uri, însă utilizarea unor relee poate permite comanda unor elemente luminoase mai puternice alimentate la tensiuni mai mari.

LED-urile și butonul vor fi conectate la porturi digitale ale plăcii Arduino, iar senzorul de presiune va folosi unul din porturile analogice.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>LED</i>	5 mm, roșu	2	
<i>LED</i>	5 mm, galben	1	
<i>LED</i>	5 mm, verde	2	
<i>LED</i>	5 mm, alb	1	
<i>Rezistor</i>	Valoarea se calculează	6	

Fir de legătură	Tată-Tată	10	
Modul Buton	Buton + rezistor	1	
Modul Traductor de presiune rezistiv	FSR 406 + rezistor	1	

✓ **Observații**

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 2 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

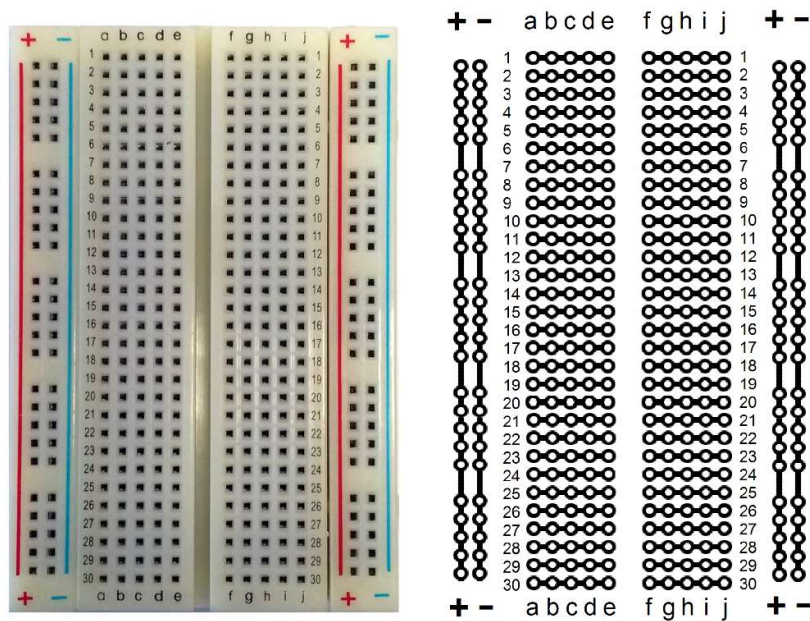
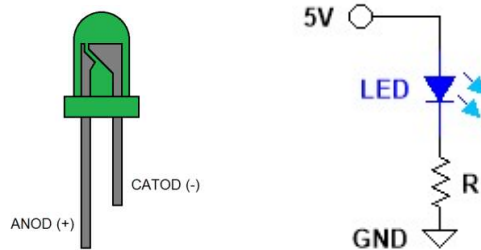


Figura 2. Breadboard-ul și conexiunile interne

**Rezistorul** se va monta în serie cu LED-ul, așa cum se vede în Figura 3, având rolul de limitare a curentului prin acesta, LED-ul funcționând la o tensiune mai mică (în general 1,5 – 3 V) decât cea furnizată de portul de ieșire digital ale plăcii Arduino (5 V).



**Figura 3. Utilizarea rezistorului de limitare a curentului prin LED**

Formula de calcul a valorii rezistenței (aplicând legea lui Ohm) este următoarea:

$$R = (V_S - V_F) / I_F \quad (1)$$

unde:  $V_S = 5 \text{ V}$  (tensiunea furnizată de portul de ieșire digital al modulului Arduino Uno)

$V_F$  (tensiunea pe dioda LED în conducție) se regăsește în specificațiile tehnice ale diodei LED (foaia de catalog).

$I_F$  (curentul prin dioda LED în conducție) se regăsește în specificațiile tehnice ale diodei LED (foaia de catalog).

Dacă nu se cunoaște producătorul unei diode LED se poate utiliza un potențiomtru în locul rezistorului și se reglează acesta până când LED-ul produce iluminarea dorită, apoi se măsoară valoarea rezistenței și se înlocuiește potențiomtrul cu un rezistor fix (**Atenție!** Trebuie măsurat și curentul prin LED pentru a nu depăși valoarea maximă ce poate fi furnizată de portul de ieșire al plăcii Arduino, adică 40 mA).

**Modulul traductor de presiune rezistiv** sesizează gradul de apăsare, bazându-se pe utilizarea unui rezistor sensibil la presiune (FSR 406), valoarea măsurată de placa Arduino fiind disponibilă sub forma unei valori digitale ce variază între 0 și 1023. Acesta conține un senzor și un rezistor de coborâre la 0.

Senzorul de presiune este realizat din trei substraturi (vezi Figura 4), având o rezistență foarte mare între electrozi ( $> 10 \text{ M}\Omega$ ) atunci când nu se exercită nici o presiune. Creșterea presiunii aplicate senzorului are ca efect realizarea unui contact electric între substraturile conductoare și determină astfel scăderea valorii rezistenței la bornele acestuia (vezi Figura 5). Valoarea rezistenței depinde nu numai de forța aplicată ci și de flexibilitatea, dimensiunile și forma obiectului care aplică presiunea [6].

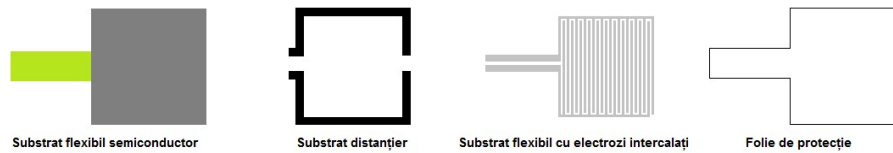


Figura 4. Construcția internă a senzorului de presiune

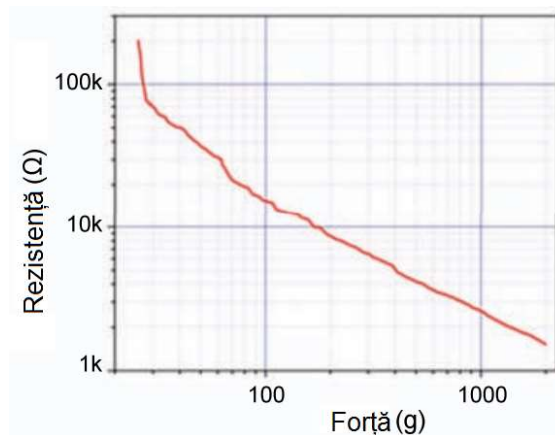


Figura 5. Variația rezistenței senzorului de presiune în raport cu forța de apăsare [6]

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5\text{ V}$ .

**Modulul Buton** este folosit pentru a sesiza apăsarea și a comanda, în acest caz, schimbarea culorii semaforului pentru pietoni. Acest traductor se poate înlocui și cu orice alt tip de buton împreună cu o rezistență de  $10\text{ k}\Omega$ , așa cum este menționat în paragraful următor.

Atât *modulul traductor de presiune* cât și *modulul buton* conțin, pe lângă senzor, și un rezistor conectat între pinul de ieșire al modulului (OUT) și masă (GND). Acesta se numește rezistor de coborâre la 0 (engl. *Pull down* [7], vezi Figura 6) și are rolul de a păstra valoarea logică 0 la ieșirea modulului atunci când nu este aplicată presiune pe senzor sau nu este apăsat butonul. Stabilirea unui nivel logic sigur (0 în acest caz) împiedică apariția aleatorie a unei valori 0 sau 1 la intrarea digitală a plăcii Arduino datorită unor eventuale zgomote electrice [8].

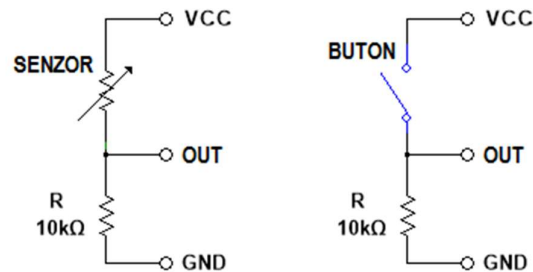


Figura 6. Utilizarea rezistorului de coborâre la 0

### 3. Componente software

**int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

**const** are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

**unsigned int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, fără semn (de la 0 până la 65.535).

**boolean variabilă = valoare** stabilește o valoare pentru o variabilă de tip logic (0 sau 1).

**long variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 32 de biți, cu semn (de la -2.147.483.648 până la 2.147.483.647).

**unsigned long variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 32 de biți, fără semn (de la 0 până la 4.294.967.295).

**void setup()** este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

`void loop()` este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

`pinMode(pin, mod)` configurează pinul digital specificat ca intrare sau ca ieșire.

`if(condiție) {instrucțiune/i} else {instrucțiune/instrucțiuni}` testează îndeplinirea sau nu a unei condiții.

`for(inițializare, condiție, increment) {instrucțiune/ instrucțiuni }` repetă un bloc de instrucțiuni până la îndeplinirea condiției.

`digitalWrite(pin, valoare)` scrie o valoare HIGH sau LOW pinului digital specificat.

`digitalRead(pin)` citește valoarea pinului digital specificat.

`analogRead(pin)` citește valoarea pinului analogic specificat.

`delay(ms)` pune în pauză programul pentru o durată de timp specificată în milisecunde.

`millis()` este o funcție ce returnează ca valoare numărul de milisecunde trecute de când a început executarea secvenței de cod.

`Serial.begin(viteză)` stabilește rata de transfer a datelor pentru portul serial în biți/secundă (BAUD).

`Serial.print(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial.

`Serial.println(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial, adăugând după datele afișate și trecerea la o linie nouă.

`==` semnifică *egal cu*.

`&&` semnifică *și*.

## 4. Aplicație. Sistem de semaforizare pentru trecere de pietoni

### 4.1. Realizarea montajului electronic

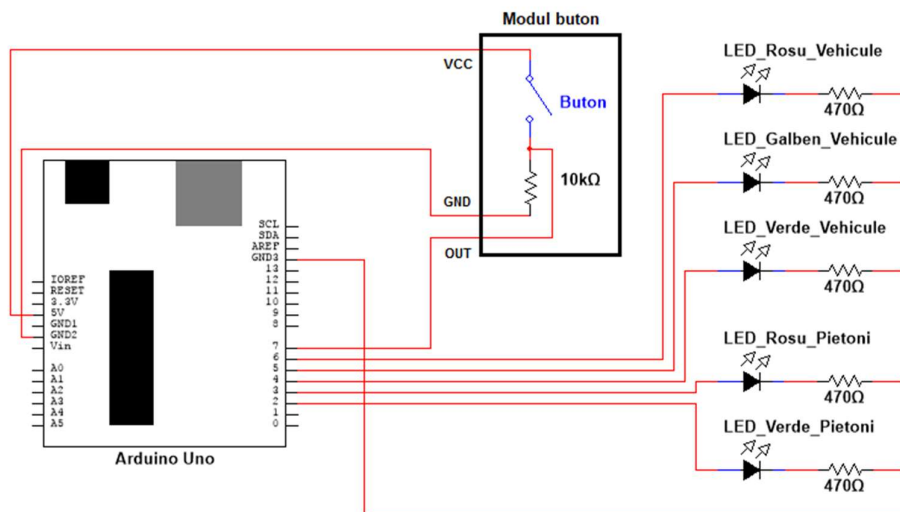


Figura 7. Schema de principiu

Se realizează următoarele conexiuni:

- Se amplasează LED-urile pe *breadboard* conectându-se anodul la ieșirea corespunzătoare a plăcii Arduino, catodul la un pin al rezistorului de limitare a curentului și celălalt pin al acestuia la coloana GND marcată cu ”-”;
- Pinul digital 2 de pe placa Arduino se conectează cu un fir la anodul LED-ului verde al semaforului pentru pietoni;
- Pinul digital 3 de pe placa Arduino se conectează cu un fir la anodul LED-ului roșu al semaforului pentru pietoni;
- Pinul digital 4 de pe placa Arduino se conectează cu un fir la anodul LED-ului verde al semaforului pentru vehicule;
- Pinul digital 5 de pe placa Arduino se conectează cu un fir la anodul LED-ului galben al semaforului pentru vehicule;
- Pinul digital 6 de pe placa Arduino se conectează cu un fir la anodul LED-ului roșu al semaforului pentru vehicule;



- Pinul GND (digital) de pe placa Arduino se conectează cu un fir la bara de ”-” a breadboard-ului;
- Pinul digital 7 de pe placa Arduino se conectează cu un fir la pinul OUT al Modulului Buton;
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al Modulului Buton;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul VCC al Modulului Buton.

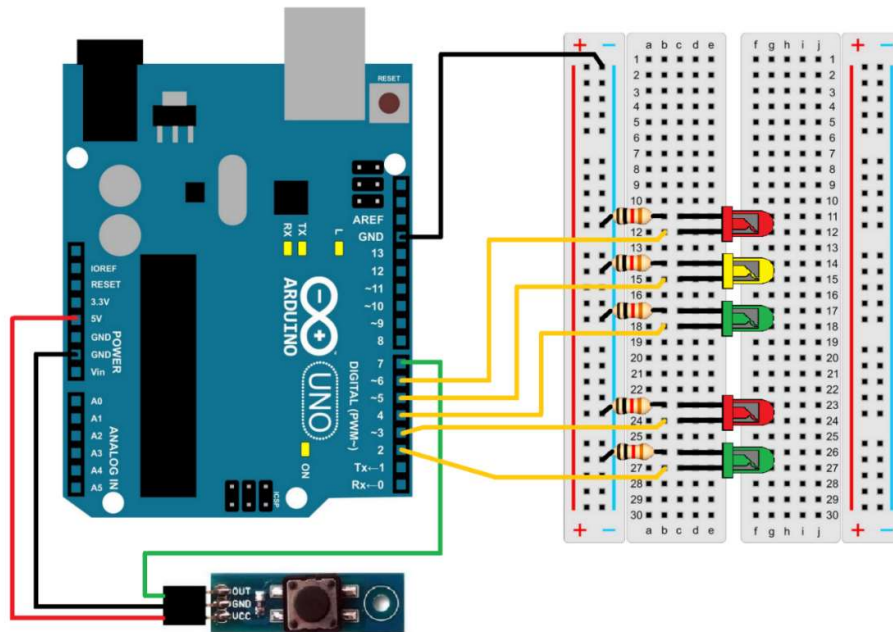
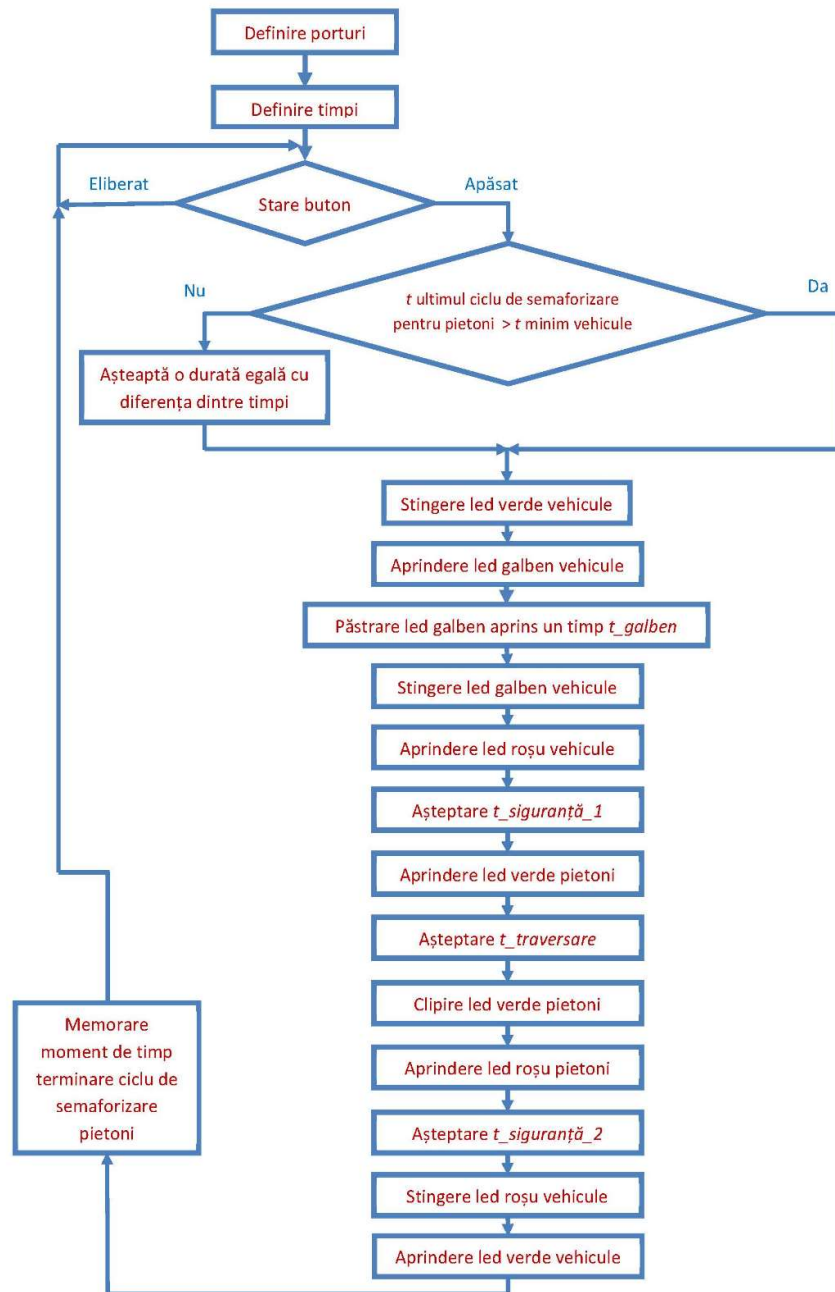


Figura 8. Realizarea conexiunilor electrice

## 4.2. Schema logică și secvența de cod



```
const int led_verde_pieton = 2;
    //definirea variabilei led_verde_pieton corespunzătoare portului digital 2
    unde va fi conectat anodul LED-ului verde pentru pietoni
const int led_rosu_pieton = 3;
    //definirea variabilei led_rosu_pieton corespunzătoare portului digital 3
    unde va fi conectat anodul LED-ului roșu pentru pietoni
const int led_verde_vehicule = 4;
    //definirea variabilei led_verde_vehicule corespunzătoare portului digital
    4 unde va fi conectat anodul LED-ului verde pentru autovehicule
const int led_galben_vehicule = 5;
    //definirea variabilei led_galben_vehicule corespunzătoare portului digital
    5 unde va fi conectat anodul LED-ului galben pentru autovehicule
const int led_rosu_vehicule = 6;
    //definirea variabilei led_rosu_vehicule corespunzătoare portului digital 6
    unde va fi conectat anodul LED-ului roșu pentru autovehicule
const int buton = 7;
    //definirea variabilei buton corespunzătoare portului digital 7 unde va fi
    conectat pinul OUT al modului buton
const int t_galben = 2000;
    //definirea timpului de galben ce va avea valoarea de 2 secunde
const int t_siguranta_1 = 2000;
    //definirea timpului de siguranță 1 ce va avea valoarea de 2 secunde
const int t_siguranta_2 = 2000;
    //definirea timpului de siguranță 2 ce va avea valoarea de 2 secunde
const int t_traversare = 4000;
    //definirea timpului de traversare ce va avea valoarea de 4 secunde
const int t_intermitent = 400;
    //definirea timpului de bază (jumătate de perioadă) pentru verde
    intermitent la pietoni
unsigned long t_min_ciclu_vehicule = 10000;
    //definirea timpului minim impus pentru un ciclu de semaforizare pentru
    vehicule
unsigned long t_ultimul_ciclu_pietoni = 0;
    //definirea valorii inițiale corespunzătoare momentului de timp al
    ultimului ciclu de semaforizare pentru pietoni

void setup() {
    pinMode(led_verde_pieton, OUTPUT);
    //se declară pinul led_verde_pieton ca fiind de ieșire
    pinMode(led_rosu_pieton, OUTPUT);
    //se declară pinul led_rosu_pieton ca fiind de ieșire
    pinMode(led_verde_vehicule, OUTPUT);
    //se declară pinul led_verde_vehicule ca fiind de ieșire
    pinMode(led_galben_vehicule, OUTPUT);
```

```
    //se declară pinul led_galben_vehicule ca fiind de ieșire
pinMode(led_rosu_vehicule, OUTPUT);
    //se declară pinul led_rosu_vehicule ca fiind de ieșire
pinMode(buton, INPUT);
    //se declară pinul buton ca fiind de intrare
digitalWrite(led_verde_vehicule, HIGH);
    //se aprinde led_verde_vehicule (starea inițială)
digitalWrite(led_rosu_pieton, HIGH);
    //se aprinde led_rosu_pieton (starea inițială)
}

void loop() {
    boolean stare_buton = digitalRead(buton);
    //se declară că variabila de tip boolean stareButon ia valoarea logică a
    //pinului buton
    if (stare_buton == HIGH && millis() - t_ultimul_ciclu_pietoni <
        t_min_ciclu_vehicule) {
        //dacă starea butonului este 1 logic (buton apăsat) ȘI timpul scurs de la
        //sfârșitul ultimului ciclu de semaforizare pentru pietoni este mai mic decât
        //timpul minim impus pentru un ciclu de semaforizare pentru vehicule
        delay(t_min_ciclu_vehicule - (millis() - t_ultimul_ciclu_pietoni));
        //atunci așteaptă un timp egal cu diferența dintre cei doi timpi
        initiere_ciclu_pietoni();
        //și apoi execută funcția initiere_ciclu_pietoni
    }
    if (stare_buton == HIGH && millis() - t_ultimul_ciclu_pietoni >
        t_min_ciclu_vehicule) {
        //dacă starea butonului este 1 logic (buton apăsat) ȘI timpul scurs de la
        //sfârșitul ultimului ciclu de semaforizare pentru pietoni este mai mare
        //decât timpul minim impus pentru un ciclu de semaforizare pentru
        //vehicule
        initiere_ciclu_pietoni();
        //atunci execută funcția initiere_ciclu_pietoni
    }
}

void initiere_ciclu_pietoni() {
    //definirea funcției initiere_ciclu_pietoni
    digitalWrite(led_verde_vehicule, LOW);
    //scrie valoarea 0 logic pe pinul led_verde_vehicule (stinge
    //led_verde_vehicule)
    digitalWrite(led_galben_vehicule, HIGH);
    //scrie valoarea 1 logic pe pinul led_galben_vehicule (aprinde
    //led_galben_vehicule)
    delay(t_galben);
}
```

```
//păstrează led_galben_auto aprins o perioadă egală cu timpul de galben
digitalWrite(led_galben_vehicule, LOW);
//scrie valoarea 0 logic pe pinul led_galben_vehicule (stinge
led_galben_vehicule)
digitalWrite(led_rosu_vehicule, HIGH);
//scrie valoarea 1 logic pe pinul led_rosu_vehicule (aprinde
led_rosu_vehicule)
delay(t_siguranta_1);
//așteaptă timpul de siguranță 1, până la aprinderea led_verde_pieton
digitalWrite(led_rosu_pieton, LOW);
//scrie valoarea 0 logic pe pinul led_rosu_pieton (stinge led_rosu_pieton)
digitalWrite(led_verde_pieton, HIGH);
//scrie valoarea 1 logic pe pinul led_verde_pieton (aprinde
led_verde_pieton)
delay(t_traversare);
//păstrează led_verde_pieton aprins o perioadă egală cu timpul de
traversare
for (int x=0; x<5; x=x+1) {
//execută conținutul buclei "for" de 5 ori
digitalWrite(led_verde_pieton, LOW);
//scrie valoarea 0 logic pe pinul led_verde_pieton (stinge
led_verde_pieton)
delay(t_intermitent);
//păstrează led_verde_pieton aprins pentru un timp t_intermitent
digitalWrite(led_verde_pieton, HIGH);
//scrie valoarea 1 logic pe pinul led_verde_pieton (aprinde
led_verde_pieton)
delay(t_intermitent);
//păstrează led_verde_pieton stins pentru un timp t_intermitent
}
digitalWrite(led_verde_pieton, LOW);
//scrie valoarea 0 logic pe pinul led_verde_pieton (stinge
led_verde_pieton)
digitalWrite(led_rosu_pieton, HIGH);
//scrie valoarea 1 logic pe pinul led_rosu_pieton (aprinde
led_rosu_pieton)
delay(t_siguranta_2);
//așteaptă timpul de siguranță 1, până la aprinderea led_verde_auto
digitalWrite(led_rosu_vehicule, LOW);
//scrie valoarea 0 logic pe pinul led_rosu_vehicule (stinge
led_rosu_vehicule)
digitalWrite(led_verde_vehicule, HIGH);
//scrie valoarea 1 logic pe pinul led_verde_vehicule (aprinde
led_verde_vehicule)
```

```
t_ultimul_ciclu_pietoni = millis();  
    //se memorează momentul de timp al sfârșitului ultimului ciclu de  
    semaforizare pentru pietoni  
}
```

## 5. Exerciții suplimentare și concluzii

1. Să se adauge un LED care să se aprindă în momentul apăsării butonului și să stea aprins până la finalul ciclului de semaforizare pentru pietoni.
2. Să se înlocuiască butonul cu un traductor de presiune rezistiv, astfel încât cererea pietonilor să se realizeze automat la sesizarea prezenței acestora. Pentru aceasta se va testa mai întâi traductorul pentru a determina valoarea de prag a presiunii ce corespunde cu detecția unui pieton (se va utiliza o secvență de cod similară cu cea prezentată în lucrarea 1, aplicația 5).
3. Să se modifice secvența de cod astfel încât timpul de traversare pentru pietoni să depindă de numărul de pietoni prezenți. Acest lucru se va simula prin adăugarea consecutivă de greutate pe senzorul de presiune și stabilirea unor valori de prag.

Valoarea logică sau booleană este de tip 0 sau 1 (Fals sau Adevărat) și ea este transmisă între diversele componente și echipamente electronice prin intermediul unor semnale electrice digitale (nivelul de tensiune 0 V are semnificația 0 logic iar nivelul de tensiune de +5 V are semnificația 1 logic). Este foarte importantă înțelegerea modului de reprezentare a diferitelor variabile în medii diferite. Astfel, o variabilă booleană poate fi reprezentată pe ecran cu caracterele 0 sau 1 sau cu șirurile de caractere *TRUE* sau *FALSE*, în timp ce pentru aceleași variabile transmise către porți logice sau intrări digitale se va utiliza un semnal electric cu nivel de amplitudine variabil în funcție de valoarea logică transmisă.

Un alt aspect deosebit de important pentru proiectarea și realizarea unor circuite sau echipamente care funcționează în cadrul unor sisteme ce sunt legate direct de siguranța circulației (exemplul semafoarelor) îl reprezintă proiectarea acestora pentru a atinge un anumit grad de fiabilitate iar, la defectare, să se asigure trecerea răspunsului fals al circuitului sau echipamentului în răspuns eronat (se vor considera trei stări pe care le poate avea un circuit: starea normală de funcționare – când răspunsul circuitului este cel proiectat; starea defectat cu răspuns fals – când circuitul prin

defectare permite efectuarea unor comenzi care să conducă la accidente – exemplu verde antagonist la semafor; și starea defectat cu răspuns eronat – când circuitul prin defectare conduce la îngreunarea circulației și creșterea unor timpi de așteptare – exemplu, galben intermitent la semafoare).

Intrările analogice ale plăcii de dezvoltare Arduino preiau semnalul de la modulele conectate la aceasta și le transmit către convertorul analogic-digital, mai departe semnalele achiziționate de placa de dezvoltare fiind prelucrate digital de către componentele acesteia.

Prin apăsarea butonului de către pietoni trebuie trimisă o cerere pentru a permite trecerea acestora. Cererea trebuie analizată de către sistem în contextul asigurării unui flux normal atât pentru vehicule cât și pentru pietoni. Semaforul pentru pietoni nu va trece în starea permisivă imediat ce pietonul apasă pe buton ci, în funcție de fazele de semaforizare din acel moment, după un timp care va rezulta din execuția algoritmului de tratare a cererii astfel încât cele două fluxuri, de vehicule și de pietoni, să se deruleze în condiții normale.

Un alt aspect important îl constituie calibrarea sistemului, respectiv determinarea și adaptarea pragurilor și nivelurilor semnalelor electrice pentru definirea unor stări ale sistemului sau componentelor acestuia. De exemplu, parametrii electrici ai modulului traductor de presiune se pot modifica în timp, acest lucru necesitând calibrări ale sistemului în funcție de noile valori ale acestora.

## 6. Bibliografie

- [1] „Newparts,” 30.03.2015.  
<http://www.newparts.info/2013/05/caracteristicile-intersecțiilor.html>.
- [2] **Department for Transport, Great Britain**, *The Design of Pedestrian Crossings*, The Stationery Office, 1995.
- [3] „Traffic Safety Corp.,” 27.03.2015.  
<http://www.xwalk.com/pages/pedx.htm>.

- [4] „*3i Innovation Ltd.*,” 27.03.2015.  
<http://3iinnovation.com/s2w/smartped-pedestrian-detection/>.
- [5] „*FLIR Systems*,” 30.03.2015. <http://www.flir.fr/>.
- [6] „*Interlink Electronics - FSR® 400 Series Data Sheet*,” 25.03.2015.  
[http://www.interlinkelectronics.com/datasheets/Datasheet\\_FSR.pdf](http://www.interlinkelectronics.com/datasheets/Datasheet_FSR.pdf).
- [7] „*Arduino Playground*,” 10.03.2015.  
<http://playground.arduino.cc/CommonTopics/PullUpDownResistor>.
- [8] **M. McRoberts**, *Beginning Arduino*, 2nd Edition, Apress, 2013.



# Lucrarea 3. Măsurarea parametrilor mediului înconjurător utilizând senzori analogici

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează senzori și traductoare.
- Utilizarea de module electronice tip *shield*.
- Realizarea unei aplicații practice de măsurare a valorilor temperaturii și umidității relative, de calcul al indicelui de confort termic și afișarea lor pe un ecran LCD.

### 1.2. Descriere teoretică

#### ✓ Introducere

Temperatura este o mărime fizică ce caracterizează starea termică a unui mediu sau a unui corp. Cele mai utilizate scări de măsurare a temperaturii sunt scara Celsius (punctul de îngheț al apei este 0 °C, punctul de fierbere este 100 °C) și scara Fahrenheit (punctul de îngheț al apei este 32 °F, punctul de fierbere este 212 °F). Relația între cele două este următoarea:  $t_C[^\circ\text{F}] = t_F[^\circ\text{C}] \times 1,8 + 32$ .

Umiditatea atmosferică reprezintă cantitatea de vapori de apă din aer. Umiditatea relativă a aerului este relația proporțională dintre umiditatea momentană la o temperatură anume și umiditatea maximă posibilă la aceeași temperatură și se măsoară în procente. Ea nu poate depăși 100% deoarece surplusul se elimină prin condensare. Atunci când cantitatea vaporilor de apă este constantă, scăderea temperaturii determină creșterea valorii umidității relative (aerul devine mai umed), iar creșterea temperaturii determină scăderea valorii umidității relative (aerul devine mai uscat).

Indicele de confort termic este utilizat pentru a descrie temperatura aparentă resimțită de corpul uman și se calculează în funcție de temperatura aerului și umiditatea relativă, după următoarea formulă:

$$ICT = (t_c \cdot 1,8 + 32) - (0,55 - 0,0055 \cdot h)[(t_c \cdot 1,8 + 32) - 58] \quad (1)$$

unde  $t_c$  este temperatura ( $^{\circ}\text{C}$ ) iar  $h$  este umiditatea relativă (%) [1].

Valoarea indicelui de confort termic se interpretează astfel:

- $ICT \leq 65$ , stare de confort;
- $65 < ICT < 80$ , stare de alertă;
- $ICT \geq 80$ , stare de disconfort.

#### ✓ **Descrierea aplicației**

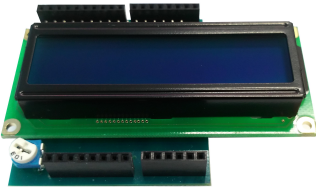

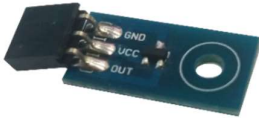
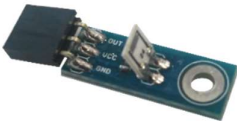
Scopul acestei aplicații este de a realiza un circuit electronic care să măsoare temperatura și umiditatea relativă a mediului înconjurător utilizând senzori analogici, să realizeze calculul indicelui de confort termic și să le afișeze numeric pe un ecran LCD.

Pentru măsurarea temperaturii se va folosi un traductor bazat pe un circuit integrat specializat, de precizie, LM50. Traductorul furnizează la ieșire o tensiune analogică ce va fi aplicată uneia din intrările analogice ale plăcii Arduino, având avantajul unei caracteristici liniare a variației tensiunii de ieșire în raport cu temperatura. Pe baza tensiunii analogice de la intrare placa va furniza o valoare digitală corespunzătoare, valoare ce va fi folosită pentru a calcula și afișa temperatura măsurată.

Pentru măsurarea umidității se va folosi un senzor rezistiv SYH-2R (rezistența la bornele acestuia variază în funcție de umiditate) împreună cu un rezistor fix, pentru a forma un divizor rezistiv de tensiune care să furnizeze plăcii Arduino o tensiune analogică ce variază în funcție de umiditatea măsurată. Pe baza tensiunii analogice de la intrare placa va furniza o valoare digitală corespunzătoare, valoare ce va fi folosită pentru a determina și afișa umiditatea măsurată.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>LCD Shield</i>	Afișare pe 2 rânduri a câte 16 caractere	1	
<i>Fir de legătură</i>	Tată-Tată	8	
<i>Modul traductor de temperatură</i>	LM50	1	
<i>Modul traductor de umiditate</i>	SYH-2R	1	

### ✓ Observații

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 1 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

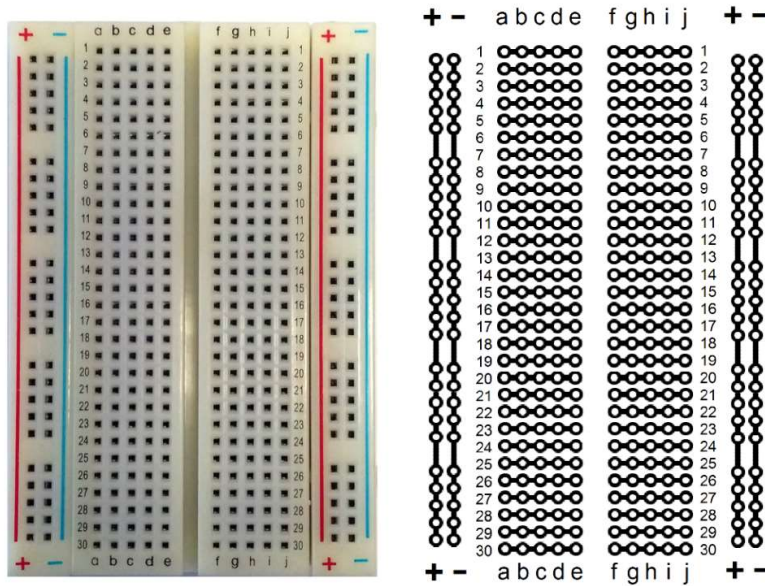


Figura 1. Breadboard-ul și conexiunile interne

**Traductorul de temperatură** măsoară temperatura mediului ambiant, bazându-se pe utilizarea unui senzor de temperatură de precizie (LM50). Senzorul poate măsura temperaturi între  $-40\text{ }^{\circ}\text{C}$  și  $+125\text{ }^{\circ}\text{C}$ , tensiunea de ieșire fiind proporțională cu temperatura în grade Celsius și variind cu pași de  $10\text{ mV}/^{\circ}\text{C}$ . Având în vedere că senzorul măsoară și temperaturi negative, fără să fie necesară o sursă de tensiune negativă, pentru temperatura de  $0\text{ }^{\circ}\text{C}$  tensiunea de la ieșire NU este  $0\text{ V}$ , ci are valoarea de  $500\text{ mV}$  [2]. Pe baza variației acestei tensiuni (ideal între  $0,1$  și  $1,75\text{ V}$ ), aplicată unuia din porturile analogice, placa Arduino furnizează o valoare digitală ce variază între  $21$  și  $359$  ( $103$  pentru temperatura de  $0\text{ }^{\circ}\text{C}$ ). Precizia senzorului este de  $\pm 3\text{ }^{\circ}\text{C}$  la temperatura camerei și de  $\pm 4\text{ }^{\circ}\text{C}$  de-a lungul întregii game de măsurare.

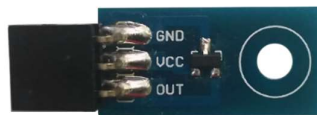


Figura 2. Traductor de temperatură realizat cu senzorul LM50

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5\text{ V}$ .

### Măsurarea temperaturii

Pentru calculul valorii temperaturii măsurate se determină mai întâi valoarea tensiunii analogice ( $U_{temp}$ ) aplicată intrării analogice a plăcii Arduino, pe baza valorii digitale furnizată de aceasta ( $val\_dig\_temp$ ).

$$U_{temp} = \frac{(val\_dig\_temp \cdot V_{CC})}{1023} \quad (2)$$

Ținând cont că temperaturii de  $0^{\circ}\text{C}$  îi corespunde o valoare de  $0,5\text{ V}$  la ieșirea traductorului și că tensiunea de ieșire variază cu  $0,01\text{ V}/^{\circ}\text{C}$ , valoarea temperaturii se poate calcula cu următoarea formulă:

$$temp = \frac{(U_{temp} - 0,5)}{0,01} \quad (3)$$

**Traductorul de umiditate** măsoară umiditatea mediului ambiant, bazându-se pe utilizarea unui senzor de umiditate rezistiv. Senzorul poate măsura umiditatea relativă între  $10\%$  și  $95\%$  iar variația rezistenței de ieșire în funcție de umiditate (măsurate la temperatura de  $25^{\circ}\text{C}$ ) este cea din Figura 4.

Traductorul de umiditate conține, pe lângă senzor, și un rezistor conectat între pinul de ieșire al modului (OUT) și  $V_{CC}$ . Acesta formează, împreună cu senzorul, un divizor rezistiv de tensiune (Figura 3).

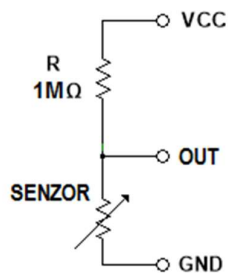
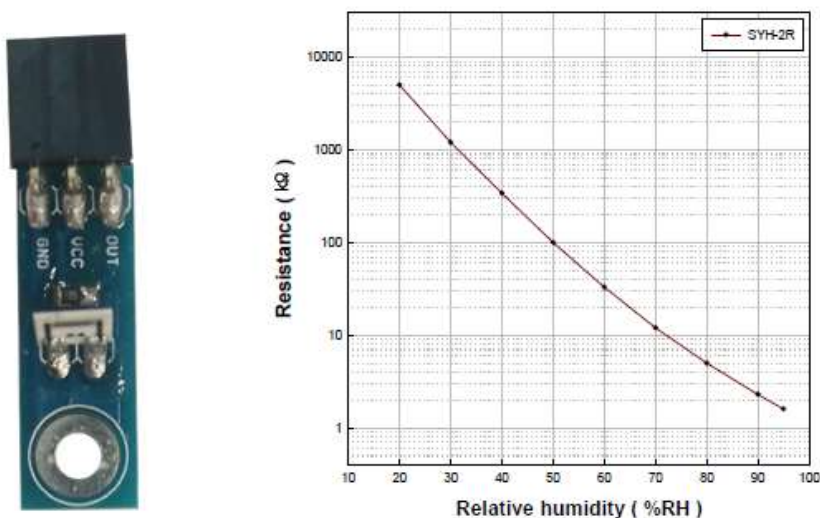


Figura 3. Divizor rezistiv de tensiune



**Figura 4. Traductor de umiditate realizat cu senzorul SYH-2R și caracteristica acestuia [3]**

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5\text{ V}$ .

#### *Măsurarea umidității*

Deoarece producătorul senzorului nu oferă o formulă de calcul pentru umiditate în funcție de rezistența senzorului, determinarea valorii umidității măsurate se va face printr-o metodă mai puțin precisă dar cu rezultate acceptabile pentru o lucrare de laborator demonstrativă. Metoda constă în utilizarea unui program de calculator capabil să digitalizeze grafice aflate sub formă de imagine (în cazul de față a fost utilizat *Plot Digitizer* [4]). Astfel, prin calibrarea inițială a axelor X și Y, se pot citi valorile caracteristicii digitizate dând clic cu mausul pe fiecare punct în parte.

Valorile preluate din grafic se regăsesc în tabelul de mai jos, în coloanele *Rezistență* și *Umiditate*. S-a ales afișarea umidității măsurate cu un pas de 5 % pentru simplificarea secvenței de cod.

Prin utilizarea unui divizor rezistiv de tensiune căruia i se aplică tensiunea  $V_{CC}$ , la ieșire se obține o tensiune proporțională cu variația rezistenței senzorului ( $U_{\text{umid}}$ ) calculată conform formulei:

$$U_{umid} = U_{OUT} = V_{CC} \cdot \frac{R_{senzor}}{R + R_{senzor}} \quad (4)$$

Valoarea digitală furnizată de placa Arduino corespunzătoare fiecărui nivel de tensiune de la intrare se calculează conform formulei:

$$val_{dig\_umid} = \frac{1023 \cdot U_{umid}}{V_{CC}} \quad (5)$$

Astfel se stabilește câte un domeniu de valori digitale care să aproximeze valoarea umidității măsurate (cu pas de 5 %).

Umiditate (%)	Rezistență senzor (Ω)	U_umid (V)	Domeniu val dig_umid	
20	5000	4,166666667	853	1023
25	2400	3,529411765	722	852
30	1210	2,737556561	560	721
35	610	1,894409938	388	559
40	345	1,282527881	262	387
45	170	0,726495726	149	261
50	100	0,454545455	93	148
55	55	0,260663507	53	92
60	33	0,159728945	33	52
65	19,5	0,095635115	20	32
70	12	0,059288538	12	19
75	7,5	0,037220844	8	11
80	5	0,024875622	5	7
85	3,35	0,016694075	3	4
90	2,3	0,011473611	2	
95			0	1

**Shield-ul LCD** permite afișarea de caractere pe un ecran cu cristale lichide, cu iluminare LED. Acesta se montează peste placa Arduino și are conectorii de așa natură încât pinii plăcii vor fi în continuare accesibili.

Ecranul LCD este format din 2 linii a câte 16 caractere, fiecare caracter fiind compus din 5x8 pixeli. Numerotarea coloanelor (caracterelor)

se face de la 0 la 15 (de la stânga la dreapta), iar al rândurilor de la 0 la 1 (de sus în jos).

**Important!** Pentru a funcționa, *shield*-ul utilizează pini digitali ai plăcii Arduino de la 2 până la 7 astfel: pinul 2 – *d7*, pinul 3 – *d6*, pinul 4 – *d5*, pinul 5 – *d4*, pinul 6 – *enable*, pinul 7 – *rs*.

### 3. Componente software

*LiquidCrystal.h* este biblioteca ce conține comenzile pentru *shield*-ul LCD.

*LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)* creează o variabilă *lcd* specificându-se pini digitali folosiți pentru a comanda *shield*-ul LCD.

*int variabilă = valoare* stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

*const* are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

*float variabilă = valoare* stabilește o valoare pentru o variabilă de tip real în virgulă mobilă pe 32 de biți, cu semn (de la -3.4028235E+38 până la 3.4028235E+38). Numărul total de digiți afișați cu precizie este 6 – 7 (include toți digiții, nu doar cei de după virgulă).

*void setup()* este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

*void loop()* este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

*analogRead(pin)* citește valoarea pinului analogic specificat.

*for(inițializare, condiție, increment) {instrucțiune/ instrucțiuni }* repetă un bloc de instrucțiuni până la îndeplinirea condiției.



*switch(variabilă) / case(valoare/domeniu de valori): instrucțiune / break* compară valoarea unei variabile cu valorile specificate în condițiile *case* și execută instrucțiunea atunci când există o potrivire. Comanda *break* determină ieșirea din instrucțiunea *switch*.

*lcd.begin(coloane, rânduri)* inițializează interfața cu ecranul LCD și specifică numărul de rânduri și de coloane al acestuia.

*lcd.setCursor(coloană, rând)* stabilește poziția cursorului LCD. Pentru LCD-ul folosit în această aplicație numărul coloanelor este de la 0 la 15, iar al rândurilor de la 0 la 1.

*lcd.clear()* șterge ecranul LCD și poziționează cursorul în colțul din stânga sus.

*lcd.print()* afișează pe ecranul LCD datele (valori ale unor variabile)/textul dintre paranteze. Pentru a afișa un text este necesar ca acesta să fie plasat între ghilimele ("text"). Pentru a afișa valoarea unei variabile de tip *char*, *byte*, *int*, *long*, sau *string* se scrie numele variabilei și, opțional, baza de numerație a acesteia (*variabilă*, BIN sau DEC sau OCT sau HEX). Pentru a afișa valoarea unei variabile de tip *float* sau *double* se scrie numele variabilei iar după virgulă, numărul de zecimale dorit a se afișa (*variabilă*, *nr. zecimale*).

*delay(ms)* pune în pauză programul pentru o durată de timp specificată în milisecunde.

*++* este folosit pentru a incrementa o variabilă

#### Crearea de caractere personalizate pentru a fi afișate pe LCD

*byte variabilă[nr. valori] = {valori}* stabilește o valoare pentru o variabilă de tip octet, fără semn. În această aplicație variabila definită nu are o singură valoare ci o matrice de valori, care are rolul de a stabili ce pixeli vor fi aprinși (valoare 1) și ce pixeli vor fi stinși (valoare 0) în compoziția unui caracter personalizat (un caracter de pe LCD este format din 5x8 pixeli).

*lcd.createChar(număr, variabilă)* creează un caracter personalizat căruia i se alocă un număr între 0 și 7, având distribuția pixelilor conform variabilei.

*lcd.write(număr)* afișează caracterul din poziția specificată (număr).

În cadrul acestei lucrări, caracterul personalizat va fi cel din Figura 5, reprezentând simbolul pentru grad Celsius.

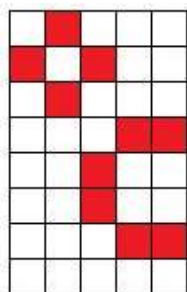
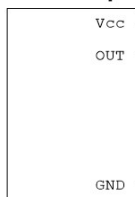


Figura 5. Caracter personalizat

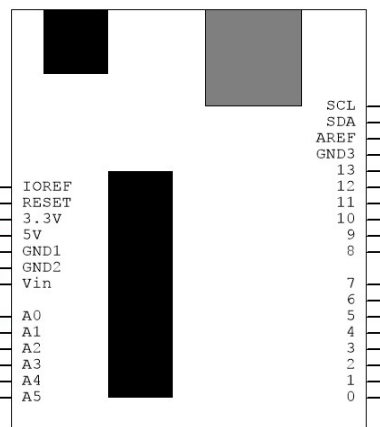
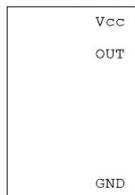
## 4. Aplicație. Măsurarea temperaturii și a umidității relative

### 4.1. Realizarea montajului electronic

#### Traductor temperatură



#### Traductor umiditate



Arduino Uno

Figura 6. Schema de principiu

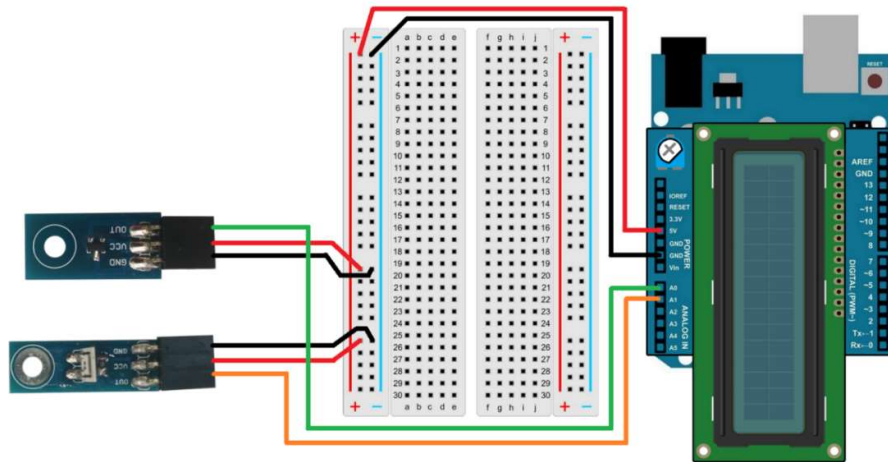
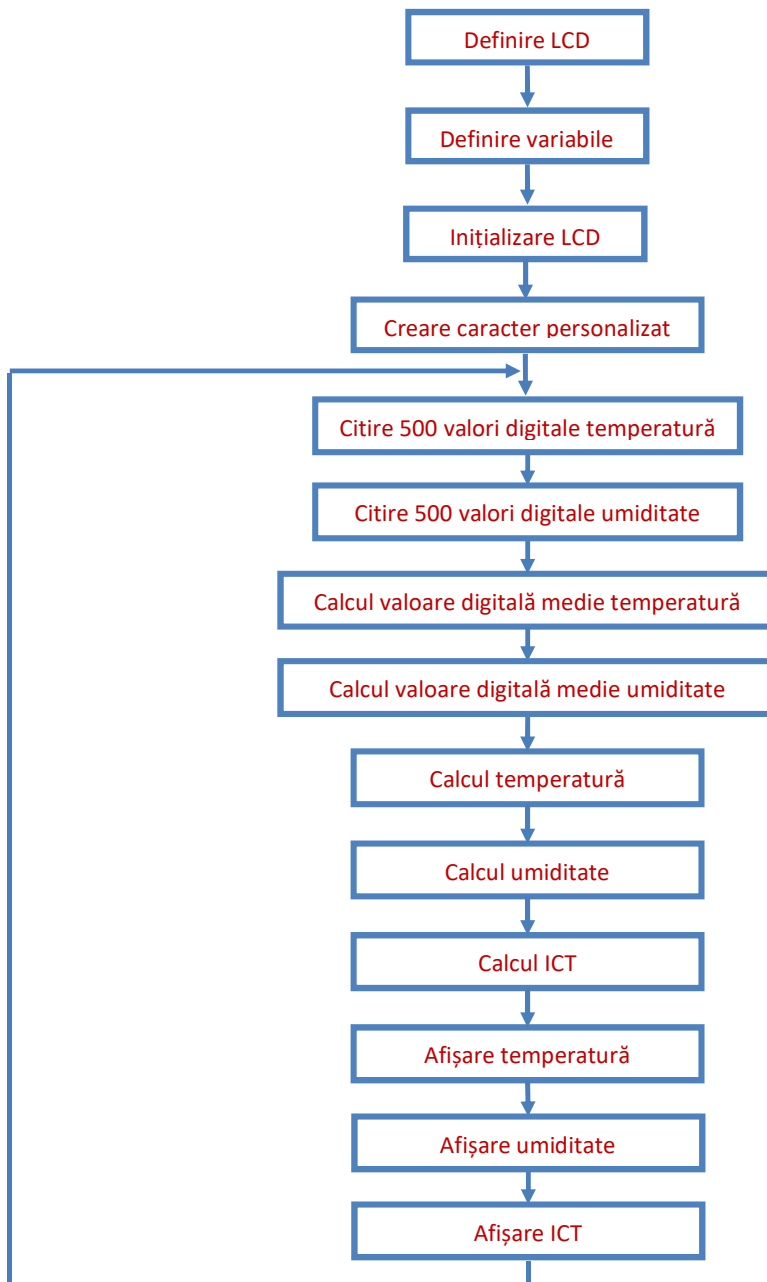


Figura 7. Realizarea conexiunilor electrice

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la bara de minus a *breadboard*-ului;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la bara de plus a *breadboard*-ului;
- Pinul analogic A0 de pe placa Arduino se conectează cu un fir la pinul OUT al modului traductor de temperatură;
- Pinul analogic A1 de pe placa Arduino se conectează cu un fir la pinul OUT al modului traductor de umiditate;
- Pinii GND ai traductoarelor se conectează cu un fir la bara de minus a *breadboard*-ului;
- Pinii Vcc ai traductoarelor se conectează cu un fir la bara de plus a *breadboard*-ului.

#### 4.2. Schema logică și secvența de cod



```
#include <LiquidCrystal.h>
//includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
//inițializarea bibliotecii și a variabilei lcd cu numerele pinilor utilizați de
shield-ul LCD
byte grad[8] = {
//definirea variabilei grad de tip byte, ca fiind o matrice cu 8 rânduri ce au
valorile din acolade
    B01000,
    B10100,
    B01000,
    B00011,
    B00100,
    B00100,
    B00011,
    B00000
};

const int portTemperatura = 0;
//definirea variabilei portTemperatura corespunzătoare portului analogic
A0 unde va fi conectat pinul OUT al senzorului de temperatură
const int portUmiditate = 1;
//definirea variabilei portUmiditate corespunzătoare portului analogic A1
unde va fi conectat pinul OUT al senzorului de umiditate
float temp = 0.0;
//definirea variabilei pentru temperatură
int umid = 0.0;
//definirea variabilei pentru umiditate
float val_dig_temp = 0.0;
//definirea variabilei val_dig_temp ce va avea valoarea digitală
corespunzătoare temperaturii citite
int val_dig_umid = 0.0;
//definirea variabilei val_dig_umid ce va avea valoarea digitală
corespunzătoare umidității citite
float U_temp = 0.0;
//definirea variabilei pentru tensiunea analogică furnizată de traductorul
de temperatură
float ICT = 0.0;
//definirea variabilei pentru ICT (indicele de confort termic)
float Vcc = 5.0;
//definirea variabilei pentru tensiunea Vcc ce va avea valoarea inițială 5V
```

```
void setup(){
  lcd.begin(16, 2);
  //inițializarea interfeței cu ecranul LCD și specificarea numărului de
  //rânduri și de coloane al acestuia
  lcd.createChar(1, grad);
  //crearea caracterului personalizat ce va avea conținutul matricei grad și
  //alocarea poziției 1
}

void loop(){
  for (int i=0;i<500;i++) {
    val_dig_temp = val_dig_temp + analogRead(portTemperatura);
    //citirea valorii digitale corespunzătoare temperaturii de 500 de ori și
    //însurarea tuturor valorilor
    val_dig_umid = val_dig_umid + analogRead(portUmiditate);
    //citirea valorii digitale corespunzătoare umidității de 500 de ori și
    //însurarea tuturor valorilor
    delay(1);
    //întârziere 1 milisecundă
  }
  val_dig_temp = val_dig_temp/500;
  //calculul valorii digitale medii a temperaturii
  val_dig_umid = val_dig_umid/500;
  //calculul valorii digitale medii a umidității
  U_temp = (val_dig_temp * Vcc)/1023;
  //calculul tensiunii analogice echivalentă valorii digitale citite – formula
  // (2)
  temp = (U_temp - 0.5)/0.01;
  //calculul temperaturii – formula (3)
  switch (val_dig_umid) {
    //determinarea valorii umidității în funcție de valoarea digitală citită
    case 0 ... 1:      umid = 95;    break;
    case 2:           umid = 90;    break;
    case 3 ... 4:     umid = 85;    break;
    case 5 ... 7:     umid = 80;    break;
    case 8 ... 11:    umid = 75;    break;
    case 12 ... 19:   umid = 70;    break;
    case 20 ... 32:   umid = 65;    break;
    case 33 ... 52:   umid = 60;    break;
    case 53 ... 92:   umid = 55;    break;
    case 93 ... 148:  umid = 50;    break;
    case 149 ... 261: umid = 45;    break;
    case 262 ... 387: umid = 40;    break;
    case 388 ... 559: umid = 35;    break;
  }
}
```

```
    case 560 ... 721:    umid = 30;    break;
    case 722 ... 852:    umid = 25;    break;
    case 853 ... 1023:   umid = 20;    break;
}
ICT = (temp * 1.8 + 32) - (0.55 - 0.0055 * umid)*((temp * 1.8 + 32) - 58);
//calculul Indicelui de Confort Termic – formula (1)
lcd.clear();
//ștergere conținut ecran LCD și poziționare cursor în colțul din stânga
//sus
lcd.print("t=");
//afișează pe ecranul LCD textul dintre ghilimele
lcd.print(temp,1);
//afișează pe ecranul LCD valoarea variabilei temp, cu o zecimală după
//virgulă
lcd.write(1);
//afișează pe ecranul LCD caracterul personalizat având poziția 1
lcd.print(" h=");
//afișează pe ecranul LCD textul dintre ghilimele
lcd.print(umid);
//afișează pe ecranul LCD valoarea variabilei umid
lcd.print("%");
//afișează pe ecranul LCD textul dintre ghilimele
lcd.setCursor(0, 1);
//mutare cursor pe coloana 1, rândul 2
lcd.print("ICT=");
//afișează pe ecranul LCD textul dintre ghilimele
lcd.print(ICT,1);
//afișează pe ecranul LCD valoarea variabilei ICT, cu o zecimală după
//virgulă
}
```

Deoarece valorile reale diferă de cele teoretice, este necesară o calibrare a circuitului electronic de măsurare realizat, prin modificări în partea de software:

- Tensiunea  $V_{cc}$  are valoarea teoretică de 5 V. Se va măsura tensiunea reală cu ajutorul unui voltmetru, iar valoarea măsurată se va scrie în program la declararea variabilei  $V_{cc}$ .

## 5. Evaluare

1. Să se modifice secvența de cod astfel încât afișarea temperaturii să se facă în grade Fahrenheit.
2. Să se modifice programul astfel încât să se afișeze mesajele „Stare de confort” pentru  $ICT \leq 65$ , „Stare de alertă” pentru  $65 < ICT < 80$  și „stare de disconfort” pentru  $ICT \geq 80$ .
3. Să se modifice programul astfel încât să se afișeze mesajele „Cod roșu” pentru  $ICT \geq 80$  și temperatură mai mare de  $30\text{ }^{\circ}\text{C}$ .
4. Să se modifice programul astfel încât în afară de afișarea temperaturii și umidității instantanee să se calculeze și afișeze valorile medii ale temperaturii și umidității pentru diferite intervale de timp (ex. 30 sec., 1 min, 12 ore, 24 ore, etc.).

Traductoarele au rolul de a converti o variație a unei mărimi fizice în semnal electric. Astfel, la ieșirea traductorului de temperatură se va regăsi o tensiune a cărei valoare este proporțională cu temperatura măsurată. Plașa de variație a tensiunii la ieșirea traductorului este recomandat a fi identică cu cea a intrării analogice a plăcii de achiziție, în general, sau a plăcii de dezvoltare Arduino, în acest caz particular. Acest lucru este important pentru a păstra rezoluția plăcii de dezvoltare, respectiv variația minimă de tensiune sesizată de intrarea analogică a plăcii de dezvoltare. Adaptarea plăcii de variație a tensiunii de la ieșirea traductorului la plașa de variație acceptată de către intrarea plăcii de dezvoltare se face prin intermediul unor circuite de condiționare a semnalelor. Această adaptare este necesară numai în cazul în care se dorește menținerea rezoluției plăcii de dezvoltare.

O rezoluție de 10 biți pentru o intrare analogică corespunde unui număr de 210 (1024 sau 1K) niveluri de tensiune diferite care pot fi sesizate de către intrarea respectivă. Pentru o variație a tensiunii între 0 și 10 V, la o intrare analogică ce este caracterizată de o rezoluție de 10 biți, se pot obține 1024 de niveluri distincte între 0 V și 10 V ceea ce înseamnă o variație minimă detectabilă de 9,765 mV.

Toate aparatele și instrumentele de măsură au nevoie de o calibrare inițială și de calibrare periodică. Această calibrare se efectuează cu ajutorul unui aparat sau instrument de măsură etalon sau prin generarea mărimii măsurate în sistem etalon.

Afișarea informațiilor pe dispozitive LCD este limitată de rezoluția afișorului (sau de numărul de pixeli pe unitatea de suprafață a afișorului), de



memoria disponibilă, de viteza de scriere și de alți parametri, în funcție de aplicațiile la care sunt utilizate aceste afișoare LCD.

## 6. Bibliografie

- [1] **I. B. E. Teodoreanu**, „*Thermal Confort*,” Present Environment and Sustainable Development, nr. 1, pp. 135-142, 2007.
- [2] **Texas Instruments**, „*LM50/LM50-Q1 - Datasheet*,” 1999, revised 2013.
- [3] **Samyoung S&C Co., Ltd.**, „*Humidity sensor SYH-2R series - specifications*”.
- [4] „*Plot Digitizer*,” <http://plotdigitizer.sourceforge.net/>. [Accesat 20 august 2015].



# Lucrarea 4. Măsurarea parametrilor mediului înconjurător utilizând senzori digitali

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează senzori și traductoare.
- Utilizarea de module electronice tip *shield*.
- Realizarea unei aplicații practice de măsurare a valorilor temperaturii, umidității relative, presiunii atmosferice și afișarea lor pe un ecran LCD.

### 1.2. Descriere teoretică

#### ✓ Introducere

Temperatura este o mărime fizică ce caracterizează starea termică a unui mediu sau a unui corp. Cele mai utilizate scări de măsurare a temperaturii sunt scara Celsius (punctul de îngheț al apei este 0 °C, punctul de fierbere este 100 °C) și scara Fahrenheit (punctul de îngheț al apei este 32 °F, punctul de fierbere este 212 °F). Relația între cele două este următoarea:  $t_C[^\circ\text{F}] = t_F[^\circ\text{C}] \times 1,8 + 32$ .

Umiditatea atmosferică reprezintă cantitatea de vapori de apă din aer. Umiditatea relativă a aerului este relația proporțională dintre umiditatea momentană la o temperatură anume și umiditatea maximă posibilă la aceeași temperatură și se măsoară în procente. Ea nu poate depăși 100% deoarece surplusul se elimină prin condensare. Atunci când cantitatea vaporilor de apă este constantă, scăderea temperaturii determină creșterea valorii umidității relative (aerul devine mai umed), iar creșterea temperaturii determină scăderea valorii umidității relative (aerul devine mai uscat).

Presiunea atmosferică reprezintă forța cu care aerul apasă pe o unitate de suprafață a pământului. Presiunea se măsoară în *Newton/metru<sup>2</sup>*

sau *Pascal* ( $1 \text{ N/m}^2 = 1 \text{ Pa}$ ). În cazul presiunii atmosferice, cele mai utilizate unități de măsură sunt *Milibarul* ( $1 \text{ mb} = 100 \text{ Pa} = 100 \text{ N/m}^2$ ) și *Milimetrul coloană de mercur* (presiunea exercitată de o coloană de mercur cu înălțimea de 1 mm,  $1 \text{ mm Hg} = 133,3 \text{ N/m}^2$ ).

Altitudinea este înălțimea unui punct de pe suprafața pământului sau a unui obiect, măsurată pe verticală, în raport cu un nivel de referință, considerat ca fiind nivelul mării. Presiunea atmosferică scade cu creșterea altitudinii și reciproc (aproximativ 10 mb la fiecare 100 m, valabil până la maxim 3000 m), iar pentru a calcula altitudinea în funcție de presiune se poate utiliza formula barometrică internațională [1]:

$$\text{altitudine} = 44330 \cdot \left( 1 - \left( \frac{p}{p_0} \right)^{\frac{1}{5,255}} \right) \quad (1)$$

unde  $p$  este presiunea atmosferică măsurată, iar  $p_0$  este presiunea atmosferică la nivelul mării. Presiunea standard la nivelul mării este de 1013 mb sau 760 mm Hg, însă ea poate varia în funcție de condițiile atmosferice.

### ✓ Descrierea aplicațiilor

Scopul acestor aplicații este de a realiza în final un circuit electronic complex care să măsoare temperatura, umiditatea relativă și presiunea atmosferică din mediul înconjurător utilizând senzori digitali, să realizeze calculul altitudinii și să le afișeze numeric pe un ecran LCD.

De reținut! Pentru realizarea unei stații meteo se poate adăuga ceasul descris în Lucrarea 8 și se recomandă utilizarea unui ecran LCD mai mare.

### Aplicația 1. Măsurarea umidității și temperaturii

Pentru măsurarea umidității se va folosi traductor (DHT22) ce conține un senzor de umiditate de tip capacitiv dar și un senzor de temperatură, datorită necesității de compensare în raport cu temperatura. Valorile umidității relative și a temperaturii măsurate vor fi transmise către placa Arduino printr-o comunicație serială, folosind unul din pinii de intrare/ieșire digitală.

### Aplicația 2. Măsurarea presiunii atmosferice și temperaturii

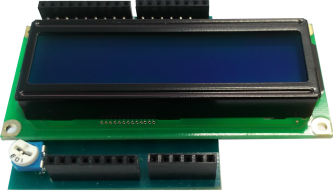


Pentru măsurarea presiunii atmosferice se va folosi un traductor (BMP180) ce conține un senzor de presiune de tip piezo-rezistiv dar și un



senzor de temperatură, datorită necesității de compensare în raport cu temperatura. Valorile presiunii atmosferice și a temperaturii măsurate vor fi transmise către placa Arduino printr-o comunicație serială de tip I<sup>2</sup>C, folosind pinii de date SCL și SDA disponibili pe placa Arduino.

De reținut! Comunicația serială I<sup>2</sup>C (*Inter Integrated Circuit*) este un tip de comunicație *multi-master*, *multi-slave* inventată de Philips Semiconductor special pentru transmiterea de date între circuite integrate de viteză mică și procesoare sau microcontrolere. Magistrala de comunicație este formată din două linii, una pentru transmiterea/recepționarea datelor, SDA (*Serial Data Line*) și una pentru transmiterea/recepționarea semnalului de ceas, SCL (*Serial Clock Line*). Este obligatorie montarea câte unui rezistor de ridicare la 1 pe fiecare dintre cele două linii de date, iar fiecare circuit conectat la o magistrală I<sup>2</sup>C trebuie să aibă o adresă proprie.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>LCD Shield</i>	Afișare pe 2 rânduri a câte 16 caractere	1	
<i>Fir de legătură</i>	Tată-Tată	7	
<i>Rezistor</i>	10kΩ	1	

<i>Traductor de umiditate</i>	DHT22	1	
<i>Traductor de presiune</i>	BMP180	1	

### ✓ Observații

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 1 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

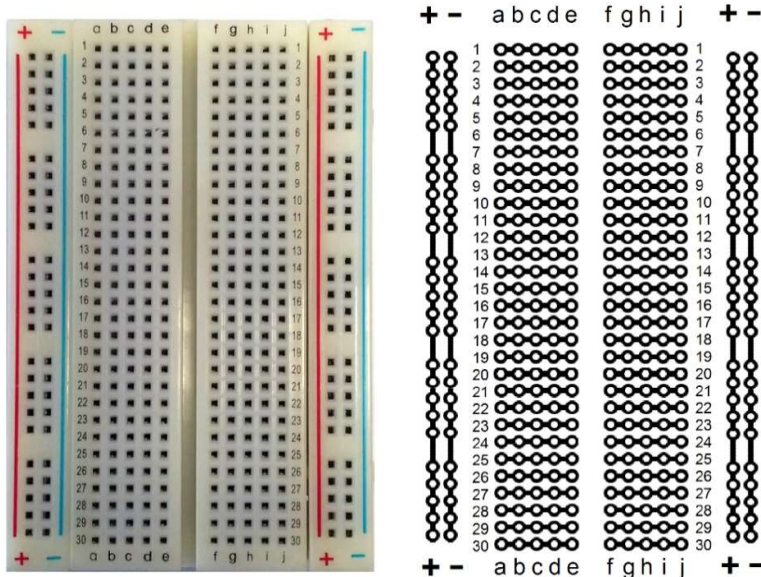


Figura 1. Breadboard-ul și conexiunile interne

*Shield-ul LCD* permite afișarea de caractere pe un ecran cu cristale lichide, cu iluminare LED. Acesta se montează peste placa Arduino și are conectorii de așa natură încât pini plăcii vor fi în continuare accesibili.

Ecranul LCD este format din 2 linii a câte 16 caractere, fiecare caracter fiind compus din 5x8 pixeli. Numerotarea coloanelor (caracterelor) se face de la 0 la 15 (de la stânga la dreapta), iar al rândurilor de la 0 la 1 (de sus în jos).

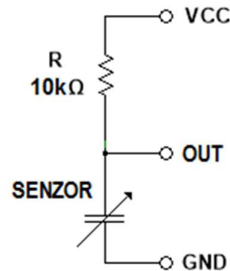
**Important!** Pentru a funcționa, *shield*-ul utilizează pinii digitali ai plăcii Arduino de la 2 până la 7 astfel: pinul 2 – *d7*, pinul 3 – *d6*, pinul 4 – *d5*, pinul 5 – *d4*, pinul 6 – *enable*, pinul 7 – *rs*.

**Traductorul de umiditate** măsoară umiditatea relativă dar și temperatura mediului ambiant, bazându-se pe utilizarea unui senzor de precizie (DHT22) calibrat și compensat în raport cu temperatura. Senzorul este de tip capacitiv și poate măsura umiditatea între 0-100% și temperatura între  $-40\text{ }^{\circ}\text{C}$  și  $+80\text{ }^{\circ}\text{C}$ , furnizând la ieșire un semnal de date digital printr-o conexiune serială. Precizia senzorului este de  $\pm 0,5\text{ }^{\circ}\text{C}$  pentru temperatură și de  $\pm 2\%$  pentru umiditate [2]. Semnificația pinilor este prezentată în Figura 2 (NC semnifică NeConectat).



Figura 2. Traductor de umiditate realizat cu senzorul DHT22

Traductorul necesită utilizarea unui rezistor de  $10\text{ k}\Omega$  între pinul de date și  $V_{CC}$ , cu rol de rezistor de ridicare la 1 (engl. *Pull up* [3], vezi Figura 3). Acesta are rolul de a păstra valoarea logică 1 la pinul de date al traductorului atunci se comută între modurile de intrare sau ieșire, sau când nu există un semnal pe acest pin. Stabilirea unui nivel logic sigur (1 în acest caz) împiedică apariția aleatorie a unei valori 0 sau 1 la intrarea digitală a plăcii Arduino datorită unor eventuale zgomote electrice [4].



**Figura 3. Utilizarea rezistorului de ridicare la 1**

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5\text{ V}$ .

#### *Măsurarea umidității și a temperaturii*

Fiind vorba de un traductor digital, măsurarea și determinarea valorilor umidității și temperaturii se face în mod automat de către acesta, iar pentru afișarea lor este necesară citirea semnalului de date, fără a fi nevoie de alte formule de calcul. Modalitatea de citire a acestuia se poate implementa în secvența de cod folosind instrucțiunile din foaia de catalog a traductorului [2], aceasta necesitând însă mai mult timp și cunoștințe avansate de programare.

Există totuși o variantă mai rapidă de a obține valorile pentru umiditate și temperatură, profitând de faptul că este un senzor foarte des utilizat, prin utilizarea în secvența de cod a unei biblioteci dezvoltate special pentru acest tip de senzor [5], numită *DHT.h* (trebuie descărcate de pe Internet două biblioteci - *Adafruit\_Sensor* și *DHT-sensor-library* - și importate în Arduino IDE folosind tab-ul *Skectch* -> *Import Library...* -> *Add Library...*). Avantajul îl reprezintă implementarea în secvența de cod a numai câțiva pași:

- Stabilirea tipului de senzor.
- Stabilirea pinului digital unde este conectat pinul de date.
- Definierea senzorului.
- Inițializarea senzorului.
- Citirea valorii umidității cu comanda `nume_senzor.readHumidity()` și alocarea ei unei variabile.
- Citirea valorii temperaturii cu comanda `nume_senzor.readTemperature()` și alocarea ei unei variabile.



*De reținut!* DHT22 este un senzor ”lent”, adică nu va reacționa instantaneu la schimbări bruște de temperatură sau umiditate, o citire a acestora putând dura până la 2 secunde sau mai mult [2].

**Traductorul de presiune** măsoară presiunea atmosferică și temperatura mediului ambiant, bazându-se pe utilizarea unui senzor de înaltă precizie și liniaritate (BMP180). Senzorul este de tip piezo-rezistiv și poate măsura presiunea între 300-1100 mb și temperatura între 0-65 °C, furnizând la ieșire un semnal de date digital printr-o conexiune serială de tip I<sup>2</sup>C. Precizia absolută tipică a senzorului este de ±1 °C pentru temperatură și de ±1 mb pentru presiune [1].

Traductorul de presiune conține și două rezistoare de 4,7 kΩ conectate între fiecare dintre pinii de date SCL și SDA, și V<sub>CC</sub>, cu rol de rezistoare de ridicare la 1 (vezi Figura 4). Atunci când o magistrală I<sup>2</sup>C este împărțită de mai multe module, fiecare având câte un set de rezistoare de ridicare la 1, se va păstra un singur set, de exemplu prin eliminarea fludorului de pe jumper-ul SJ1, încercuit cu verde în Figura 4.

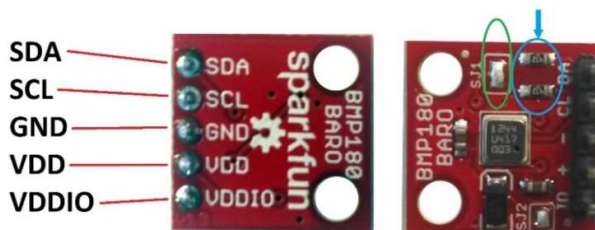


Figura 4. Utilizarea rezistoarelor de ridicare la 1

Pinul VDDIO se utilizează numai în cazul conectării traductorului la microcontrolere care funcționează cu tensiuni mai mici de 3,3 V.

Având în vedere că se utilizează magistrala I<sup>2</sup>C, va trebui ca în secvența de cod să fie inclusă și biblioteca *Wire.h*, disponibilă deja în pachetul de biblioteci preinstalate în Arduino IDE.

**ATENȚIE! Traductorul se va alimenta cu tensiunea V<sub>cc</sub> = 3,3V.**

#### *Măsurarea presiunii atmosferice și a temperaturii*

La fel ca și la traductorul de umiditate, măsurarea și determinarea valorilor presiunii atmosferice și temperaturii se face în mod automat de către acesta, iar pentru afișarea lor este necesară citirea semnalului de date.

Și pentru acest traductor se poate utiliza în secvența de cod o bibliotecă dezvoltată special [6], numită *SFE\_BMP180.h* (biblioteca trebuie descărcată de pe Internet și importată în Arduino IDE folosind tab-ul *Skech* -> *Import Library... -> Add Library...*).

Trebuie reținut că mai întâi se măsoară temperatura și după aceea presiunea, pentru a putea realiza compensarea în raport cu temperatura. Măsurarea presiunii se poate face în patru moduri ( $n = 0, 1, 2, 3$ ), în funcție de acuratețea dorită, prin preluarea a 1, 2, 4, sau 8 eșantioane, timpul de conversie variind între 4,5 și 25,5 ms.

Pașii necesari pentru obținerea valorilor presiunii și temperaturii sunt:

- Definirea senzorului.
- Inițializarea senzorului.
- Începerea măsurării temperaturii cu comanda *nume\_senzor.startTemperature()*.
- Citirea valorii temperaturii cu comanda *nume\_senzor.getTemperature(variabilă\_temp)*.
- Începerea măsurării presiunii cu comanda *nume\_senzor.startPressure(n)*.
- Citirea valorii presiunii cu comanda *nume\_senzor.getPressure(variabilă\_pres,variabilă\_temp)*.

#### *Calculul altitudinii*

Altitudinea este calculată în mod automat dacă se utilizează biblioteca *SFE\_BMP180.h*, citirea valorii acesteia realizându-se cu comanda *nume\_senzor.altitude(variabilă\_pres,p0)*.

De reținut! Pentru presiunea la nivelul mării  $p_0$  se poate utiliza valoarea standard de 1013 mb. Se recomandă totuși folosirea valorii reale, corelată cu condițiile atmosferice (valoare cunoscută de institute meteorologice și uneori disponibilă pe Internet, vezi [7] pentru București).

### 3. Componente software

#### 3.1. Funcții, comenzi și simboluri utilizate

**LiquidCrystal.h** este biblioteca ce conține comenzile pentru *shield*-ul LCD.

**DHT.h** este biblioteca ce conține comenzile pentru senzorul de umiditate.

**Wire.h** este biblioteca ce conține comenzile pentru magistrala I<sup>2</sup>C.

**SFE\_BMP180.h** este biblioteca ce conține comenzile pentru senzorul de presiune.

**LiquidCrystal** *lcd(rs, enable, d4, d5, d6, d7)* creează o variabilă *lcd* specificându-se pini digitali folosiți pentru a comanda *shield*-ul LCD.

**const** are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

**int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

**float variabilă = valoare** stabilește o valoare pentru o variabilă de tip real în virgulă mobilă pe 32 de biți, cu semn (de la -3.4028235E+38 până la 3.4028235E+38). Numărul total de digiți afișați cu precizie este 6 – 7 (include toți digiții, nu doar cei de după virgulă).

**double variabilă = valoare** stabilește o valoare pentru o variabilă de tip real în virgulă mobilă cu dublă precizie față de variabila de tip *float*.

**char variabilă = valoare** stabilește o valoare pentru o variabilă de tip caracter

**void setup()** este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

`void loop()` este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

`if(condiție) {instrucțiune/i} else {instrucțiune/instrucțiuni}` testează îndeplinirea sau nu a unei condiții.

`!=` are semnificația *diferit de*.

`senzor_umiditate.begin()` este o funcție ce inițializează senzorul de umiditate.

`senzor_umiditate.readHumidity()` este o funcție ce citește și oferă valoarea umidității măsurate.

`senzor_umiditate.readTemperature()` este o funcție ce citește și oferă valoarea temperaturii măsurate.

`Wire.begin()` este o funcție care inițializează magistrala I<sup>2</sup>C.

`senzor_presiune.begin()` este o funcție ce inițializează senzorul de presiune.

`senzor_presiune.startTemperature()` este o funcție ce comandă începerea măsurării temperaturii și returnează timpul necesar pentru efectuarea acestei măsurări.

`senzor_presiune.getTemperature(variabilă)` este o funcție ce citește de la traductor, și oferă variabilei, valoarea temperaturii măsurate.

`senzor_presiune.startPressure(n)` este o funcție ce comandă începerea măsurării presiuni și returnează timpul necesar acestei măsurări. *n* poate lua valori între 0 și 3, semnificând modul de măsurare care stabilește numărul de eșantioane.

`senzor_presiune.getPressure(variabilă_pres,variabilă_temp)` este o funcție ce citește și oferă variabilei valoarea presiunii măsurate, compensată cu temperatura specificată prin *variabilă\_temp*.

`senzor_presiune.altitude(variabilă_pres,variabilă_p0)` este o funcție ce citește valoarea altitudinii calculată în funcție de cele două variabile.

*lcd.begin(coloane, rânduri)* inițializează interfața cu ecranul LCD și specifică numărul de rânduri și de coloane al acestuia.

*lcd.setCursor(coloană, rând)* stabilește poziția cursorului LCD. Pentru LCD-ul folosit în această aplicație numărul coloanelor este de la 0 la 15, iar al rândurilor de la 0 la 1.

*lcd.clear()* șterge ecranul LCD și poziționează cursorul în colțul din stânga sus.

*lcd.print()* afișează pe ecranul LCD datele (valori ale unor variabile)/textul dintre paranteze. Pentru a afișa un text este necesar ca acesta să fie plasat între ghilimele ("text"). Pentru a afișa valoarea unei variabile de tip *char*, *byte*, *int*, *long*, sau *string* se scrie numele variabilei și, opțional, baza de numerație a acesteia (*variabilă*, BIN sau DEC sau OCT sau HEX). Pentru a afișa valoarea unei variabile de tip *float* sau *double* se scrie numele variabilei iar după virgulă, numărul de zecimale dorit a se afișa (*variabilă*, *nr. zecimale*).

*delay(ms)* pune în pauză programul pentru o durată de timp specificată în milisecunde.

#### Crearea de caractere personalizate pentru a fi afișate pe LCD

*byte variabilă[nr. valori] = {valori}* stabilește o valoare pentru o variabilă de tip octet, fără semn. În această aplicație variabila definită nu are o singură valoare ci o matrice de valori, care are rolul de a stabili ce pixeli vor fi aprinși (valoare 1) și ce pixeli vor fi stinși (valoare 0) în compoziția unui caracter personalizat (un caracter de pe LCD este format din 5x8 pixeli).

*lcd.createChar(număr, variabilă)* creează un caracter personalizat căruia i se alocă un număr între 0 și 7, având distribuția pixelilor conform variabilei.

*lcd.write(număr)* afișează caracterul din poziția specificată (număr).

În cadrul acestei lucrări, caracterul personalizat va fi cel din Figura 5, reprezentând simbolul pentru grad Celsius.

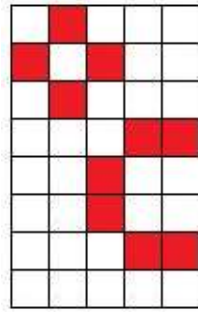


Figura 5. Caracter personalizat

## 4. Aplicația 1. Măsurarea umidității și temperaturii

### 4.1. Realizarea montajului electronic

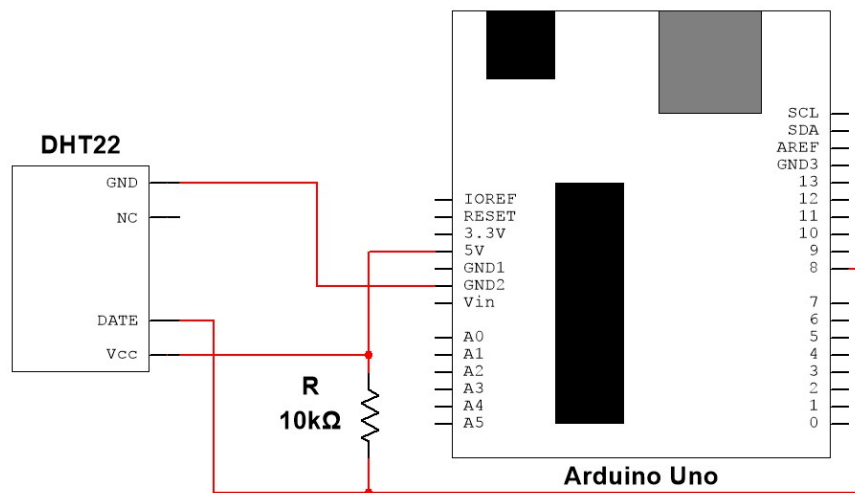


Figura 6. Schema de principiu pentru aplicația 1

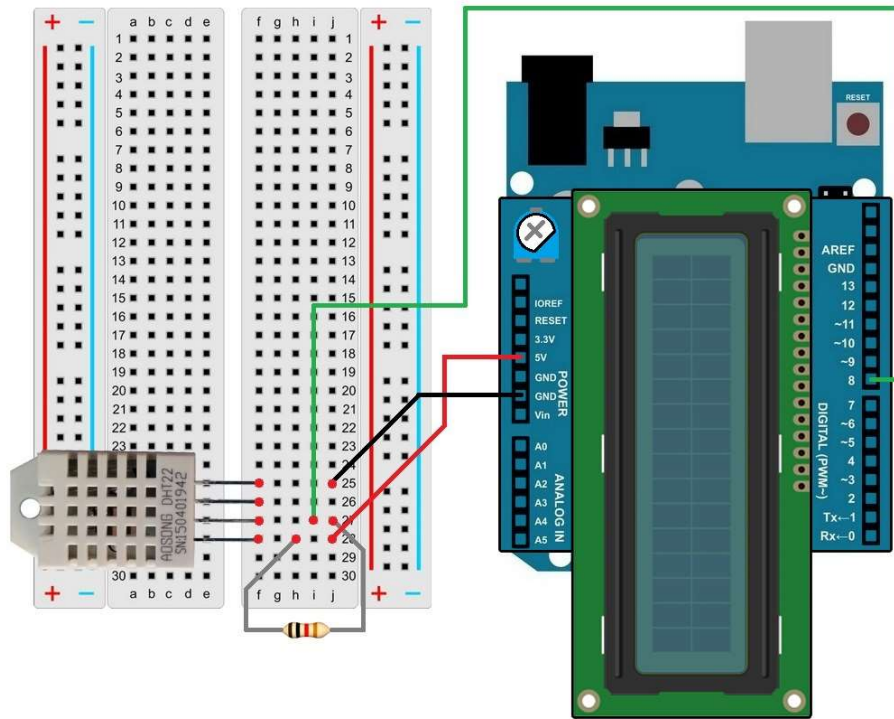
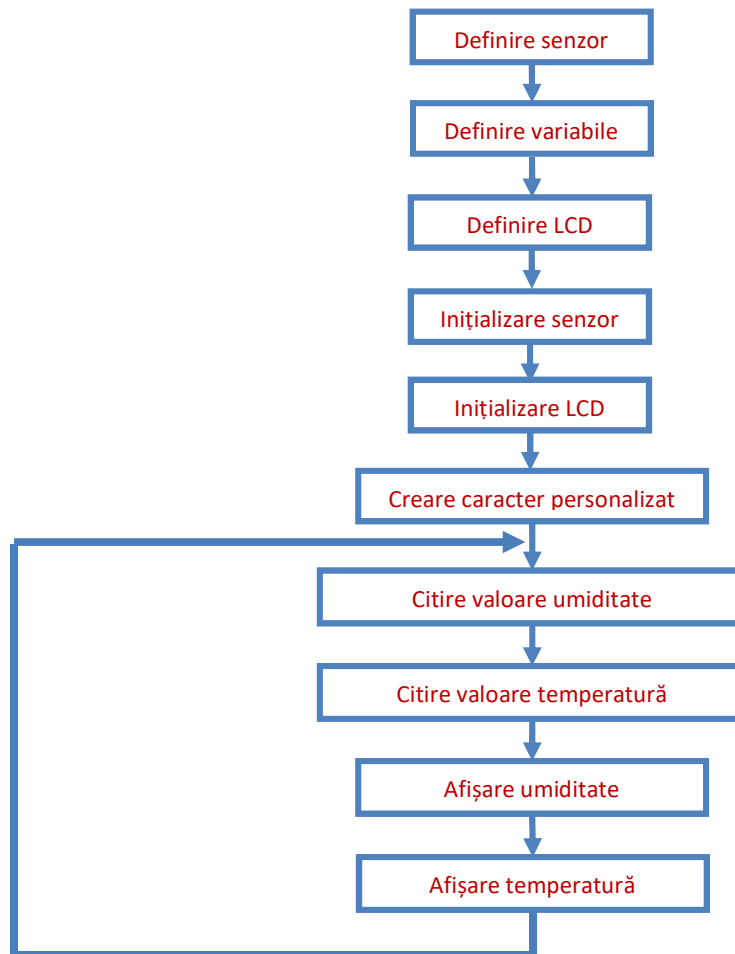


Figura 7. Realizarea conexiunilor electrice pentru aplicația 1

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului de umiditate DHT22;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul V<sub>CC</sub> al traductorului de umiditate DHT22;
- Pinul digital 8 de pe placa Arduino se conectează cu un fir la pinul DATE al traductorului de umiditate DHT22;
- Între pinii V<sub>CC</sub> și DATE ai traductorului de umiditate DHT22 se conectează un rezistor, cu rol de ridicare la 1, având valoarea de 10 k $\Omega$ .

#### 4.2. Schema logică și secvența de cod



```
#include <DHT.h>
//includerea în program a bibliotecii comenzilor pentru senzorul de
umiditate
char tip_senzor = DHT22;
//definirea tipului de senzor
const int pin_senzor = 8;
//definirea variabilei pin_senzor corespunzătoare portului digital 8 unde
va fi conectată ieșirea de date a senzorului de umiditate
DHT senzor_umiditate(pin_senzor, tip_senzor);
```



```
        //definirea senzorului de umiditate
float umid;
        //definirea variabilei umiditate
float temp;
        //definirea variabilei temperatură
#include <LiquidCrystal.h>
        //includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
        //inițializarea bibliotecii și a variabilei lcd cu numerele pinilor utilizați de
        shield-ul LCD
byte grad[8] = {
        //definirea variabilei grad de tip byte, ca fiind o matrice cu 8 rânduri ce au
        valorile din acolade
    B01000,
    B10100,
    B01000,
    B00011,
    B00100,
    B00100,
    B00011,
    B00000
};

void setup(){
    senzor_umiditate.begin();
        //inițializarea senzorului de umiditate
    lcd.begin(16, 2);
        //inițializarea interfeței cu ecranul LCD și specificarea numărului de
        rânduri și de coloane al acestuia
    lcd.createChar(1, grad);
        //crearea caracterului personalizat ce va avea conținutul matricei grad și
        alocarea poziției 1
}

void loop(){
    umid = senzor_umiditate.readHumidity();
        //citirea valorii umidității
    temp = senzor_umiditate.readTemperature();
        //citirea valorii temperaturii
    lcd.clear();
        //ștergere conținut ecran LCD
    lcd.print("Umid = ");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(umid,1);
}
```

```

        //afișează pe ecranul LCD valoarea variabilei umid, cu o zecimală după
        virgulă
    lcd.print("% ");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.setCursor(0, 1);
        //mutare cursorul pe coloana 1, rândul 2
    lcd.print("Temp = ");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(temp,1);
        //afișează pe ecranul LCD valoarea variabilei temp, cu o zecimală după
        virgulă
    lcd.write(1);
        //afișează pe ecranul LCD caracterul personalizat având poziția 1
    delay(1000);
        //întârzie 1 secundă
    }

```

## 5. Aplicația 2. Măsurarea presiunii atmosferice și temperaturii

### 5.1. Realizarea montajului electronic

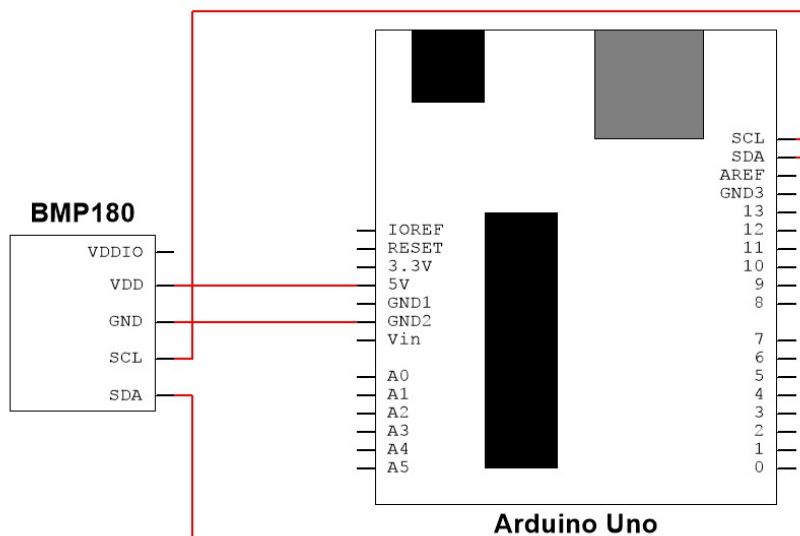


Figura 8. Schema de principiu pentru aplicația 2

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului de presiune BMP180;
- Pinul 3.3V (power) de pe placa Arduino se conectează cu un fir la pinul VDD al traductorului de presiune BMP180;
- Pinul SCL de pe placa Arduino se conectează cu un fir la pinul SCL al traductorului de presiune BMP180;
- Pinul SDA de pe placa Arduino se conectează cu un fir la pinul SDA al traductorului de presiune BMP180.

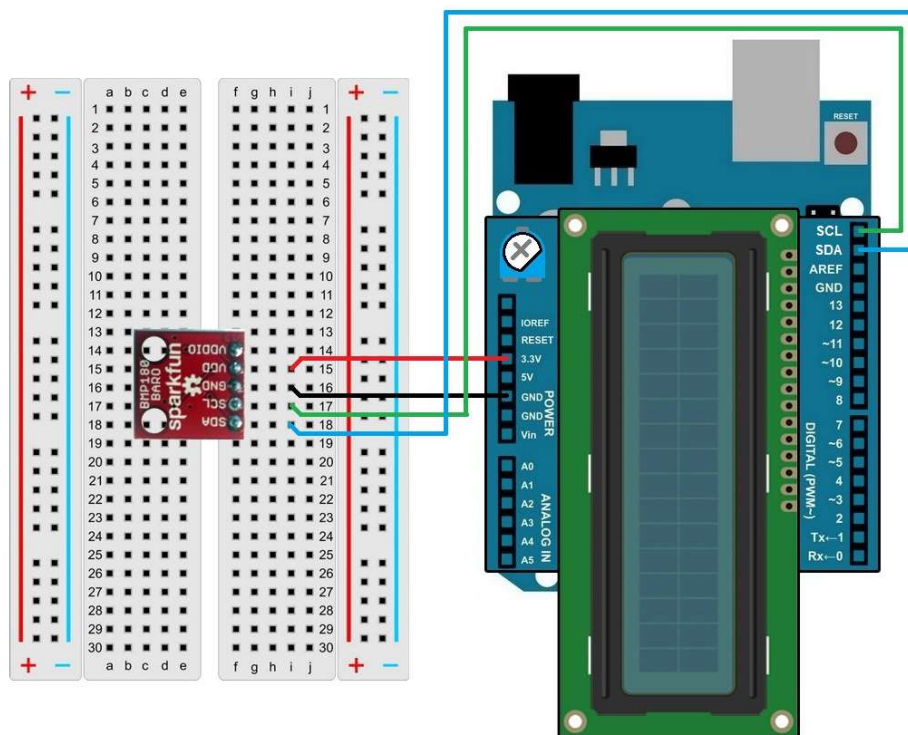
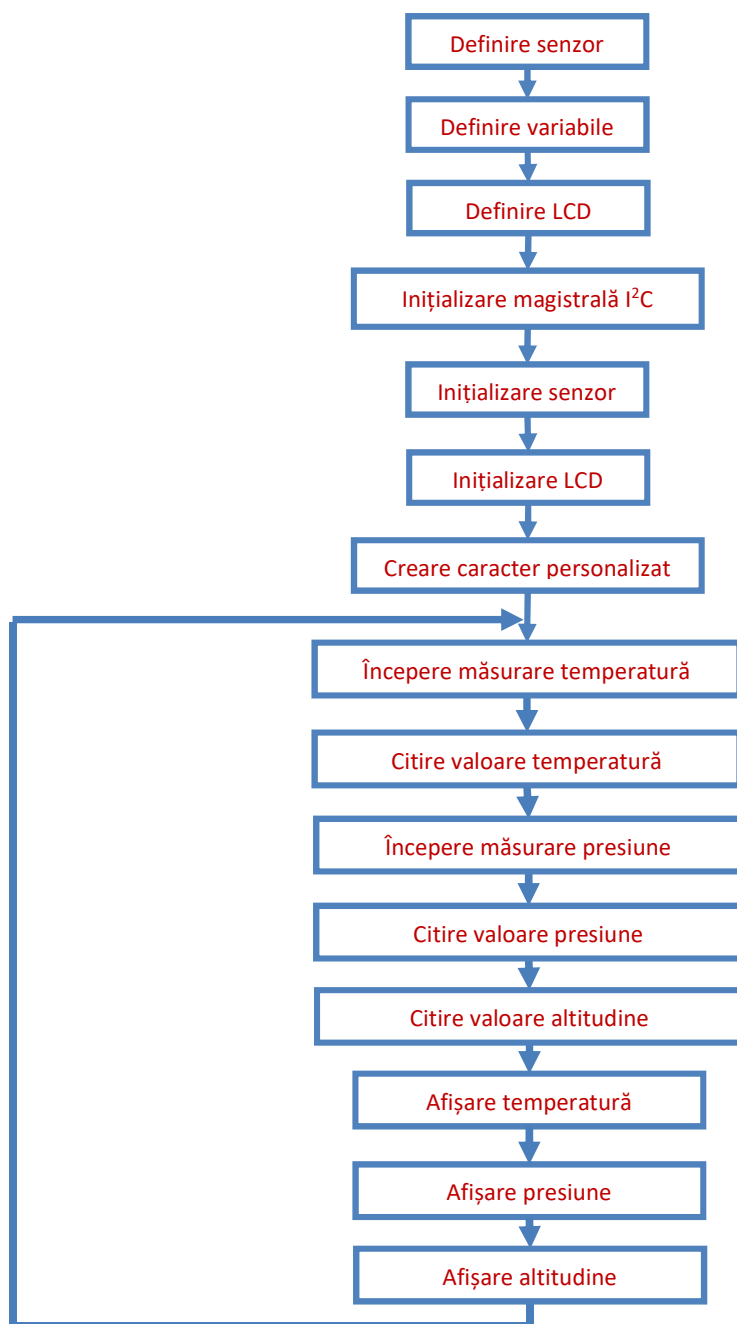


Figura 9. Realizarea conexiunilor electrice pentru aplicația 2

## 5.2. Schema logică și secvența de cod



```
#include <Wire.h>
//includerea în program a bibliotecii comenzilor pentru magistrala I2C
#include <SFE_BMP180.h>
//includerea în program a bibliotecii comenzilor pentru senzorul de
//presiune
SFE_BMP180 senzor_presiune;
//definirea senzorului de presiune
double pres, temp, alt;
//definirea variabilelor pentru presiune, temperatură și altitudine
double p0 = 1013;
//definirea variabilei p0, presiunea la nivelul mării, vezi secțiunea 2 a
//lucrării
#include <LiquidCrystal.h>
//includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
//inițializarea bibliotecii și a variabilei lcd cu numele pinilor utilizați de
//shield-ul LCD
byte grad[8] = {
//definirea variabilei grad de tip byte, ca fiind o matrice cu 8 rânduri ce au
//valorile din acolade
    B01000,
    B10100,
    B01000,
    B00011,
    B00100,
    B00100,
    B00011,
    B00000
};

void setup(){
    Wire.begin();
//inițializarea magistralei I2C
    senzor_presiune.begin();
//inițializarea senzorului de presiune
    lcd.begin(16, 2);
//inițializarea interfeței cu ecranul LCD și specificarea numărului de
//rânduri și de coloane al acestuia
    lcd.createChar(1, grad);
//crearea caracterului personalizat ce va avea conținutul matricei grad și
//alocarea poziției 1
}
```

```
void loop(){
  int status;
  //se definește variabila status de tip întreg
  status = senzor_presiune.startTemperature();
  //începerea măsurării temperaturii, funcția returnând timpul necesar
  if (status != 0) {
    //dacă timpul necesar măsurării este diferit de zero
    delay(status);
    //se așteaptă timpul respectiv
    senzor_presiune.getTemperature(temp);
    //se alocă variabilei temp valoarea temperaturii măsurate
  }
  status = senzor_presiune.startPressure(3);
  //începerea măsurării presiunii (se specifică numărul de eșantioane dorit),
  funcția returnând timpul necesar
  if (status != 0) {
    //dacă timpul necesar măsurării este diferit de zero
    delay(status);
    //se așteaptă timpul respectiv
    senzor_presiune.getPressure(pres,temp);
    //se alocă variabilei pres valoarea presiunii măsurate, compensată cu
    temperatura temp
  }
  alt = senzor_presiune.altitude(pres,p0);
  //se alocă variabilei alt valoarea altitudinii calculate în funcție de
  presiunea măsurată și de p0
  lcd.clear();
  //ștergere conținut ecran LCD
  lcd.print(temp,1);
  //afișează pe ecranul LCD valoarea variabilei temp, cu o zecimală după
  virgulă
  lcd.write(1);
  //afișează pe ecranul LCD caracterul personalizat având poziția 1
  lcd.print(" ");
  //afișează pe ecranul LCD textul dintre ghilimele
  lcd.print(pres,1);
  //afișează pe ecranul LCD valoarea variabilei pres, cu o zecimală după
  virgulă
  lcd.print("mb");
  //afișează pe ecranul LCD textul dintre ghilimele
  lcd.setCursor(0, 1);
  //mutare cursorul pe coloana 1, rândul 2
  lcd.print("Alt=");
  //afișează pe ecranul LCD textul dintre ghilimele
```

```
lcd.print(alt,1);  
    //afișează pe ecranul LCD valoarea variabilei alt, cu o zecimală după  
    virgulă  
lcd.print("m");  
    //afișează pe ecranul LCD textul dintre ghilimele  
delay(1000);  
    //întârzie 1 secundă  
}
```

## 6. Exerciții suplimentare și concluzii

1. Să se modifice secvența de cod astfel încât să se afișeze o avertizare „Cod roșu” la depășirea temperaturii de 30 °C și a umidității de 80%.
2. Să se elimine din secvența de cod funcția de citire automată a valorii altitudinii și să se înlocuiască cu formula de calcul prezentată în introducere (formula (1)).
3. Să se grupeze cele două aplicații în una singură, alegând unele date ce vor fi afișate pe ecranul LCD și altele pe monitorul serial.

Pentru majoritatea limbajelor de programare este foarte utilă construirea unor biblioteci (*library*) de funcții sau de programe care au ca scop simplificarea scrierii unor aplicații software prin utilizarea unor funcții (în special cele des utilizate) care sunt pre definite.

O variabilă de tip *float* sau simplă precizie este caracterizată prin faptul că sunt alocați 4 octeți (32 biți) care vor fi utilizați astfel: primul bit va fi bitul de semn, următorii 8 biți vor fi necesari codării exponentului și ultimii 23 biți vor fi utilizați pentru codificarea fracției. Variabila de tip *double* va fi reprezentată pe 8 octeți (64 biți) care vor fi utilizați astfel: 1 bit pentru semn, 11 biți pentru exponent și 52 biți pentru fracție.

Reprezentarea datelor și informațiilor în medii diferite, precum și abstractizarea și codificarea anumitor stări necesită utilizarea unor tipuri diferite de date. Tipul datelor este ales și pe baza unor criterii privind optimizarea programelor scrise. Variabilele de tip *double* necesită un volum mai mare de date stocate de cât cele de tip *float* și timp mai mare pentru prelucrare.

Afișoarele de tip LCD (*Liquid Crystal Display* – Afișaj cu cristale lichide) sunt afișoare des utilizate în automatizările pentru transporturi datorită consumului redus de energie și a fiabilității relativ ridicate a acestora.

## 7. Bibliografie

- [1] **Bosch Sensortec**, „*BMP180 Digital pressure sensor - Datasheet*,” 2013.
- [2] **Aosong Electronics Co.,Ltd**, „*Digital-output relative humidity & temperature sensor/module - DHT22*”.
- [3] „*Arduino Playground*,” 10 03 2015.  
<http://playground.arduino.cc/CommonTopics/PullUpDownResistor>.
- [4] **M. McRoberts**, *Beginning Arduino, 2nd Edition*, Apress, 2013.
- [5] **Adafruit**, „*DHT Sensor Library*,” <https://github.com/adafruit/DHT-sensor-library>. [Accesat 24 08 2015].
- [6] **SparkFun**, „*BMP180 Breakout Arduino Library*,”  
[https://github.com/sparkfun/BMP180\\_Breakout\\_Arduino\\_Library](https://github.com/sparkfun/BMP180_Breakout_Arduino_Library).  
[Accesat 24 08 2015].
- [7] **Weather Forecast**, „*Bucharest Henri Coandă International Airport*,”  
<http://www.weather-forecast.com/weather-stations/Bucharest-Otopeni-Airport>. [Accesat 25 08 2015].



# Lucrarea 5. Măsurarea distanței și a proximității

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează senzori și traductoare.
- Utilizarea de module electronice tip *shield*.
- Realizarea unei aplicații practice de măsurare a distanței sau a proximității față de un obiect.

### 1.2. Descriere teoretică

#### ✓ Introducere

Ultrasunetele sunt unde sonore având frecvența mai mare decât limita superioară a domeniului de sensibilitate a urechii umane (20 Hz ... 20 KHz), adică mai mare de 20kHz. Măsurarea distanței cu ajutorul ultrasunetelor presupune generarea unor valuri de unde sonore de către un emițător. Prezența unui obstacol în calea undelor duce la reflectarea acestora către senzor (Figura 1).

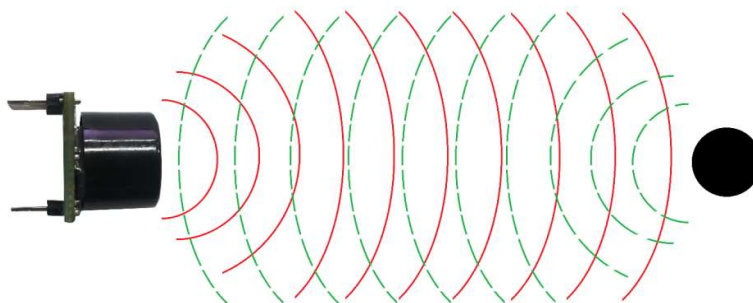


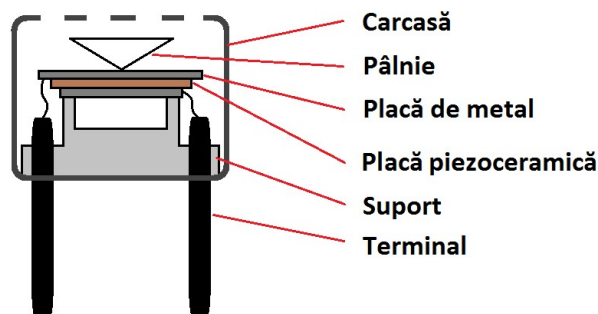
Figura 1. Măsurarea distanței cu ajutorul ultrasunetelor

Traductorul folosit pentru emiterea și recepționarea undelor ultrasonice măsoară durata de timp scursă de la emiterea până la recepția unei unde sonore, durată ce este proporțională cu distanța până la obstacol.

Cei mai utilizați senzori pentru astfel de dispozitive sunt cei bazați pe efectul piezoelectric:

- Efectul direct: aplicarea unei vibrații (unde sonore) pe suprafața unui material piezoceramic are ca efect apariția unei tensiuni electrice pe aceasta.
- Efectul invers: aplicarea unei tensiuni electrice pe suprafața unui material piezoceramic are ca efect dilatarea sau contractarea acestuia, ducând la emisia unor unde sonore.

Modul de construcție al unui astfel de dispozitiv este ilustrat în Figura 2.



**Figura 2. Structura unui senzor ultrasonic**

Elementul piezoelectric este format dintr-o placă de metal și o placă piezoceramică, iar pâlnia are dublu rol, de a radia eficient undele generate de elementul piezoelectric și de a concentra undele recepționate către partea centrală a acestuia.

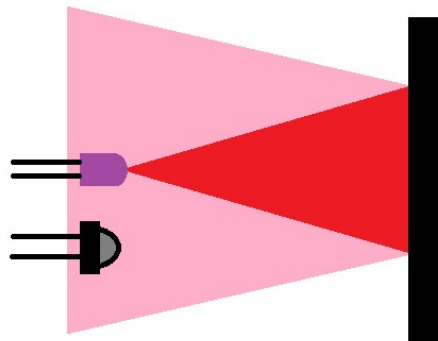
Obiectele realizate din sticlă, metal, lemn, hârtie, beton, etc. reflectă aproape în totalitate undele ultrasonice, făcând ușoară detectarea lor, pe când materialele absorbante precum textile, lână, etc. fac mai dificilă detectarea. De asemenea, mai greu de detectat sunt și obiectele cu forme neregulate, undele fiind reflectate în alte direcții și nu spre senzor [1].

O altă problemă a măsurării distanțelor cu ajutorul ultrasunetelor este atenuarea puterii undelor proporțional cu distanța parcursă de acestea și în

funcție de frecvența undelor ultrasonice (cu cât frecvența este mai mare, cu atât undele sunt atenuate mai rapid).

Utilizarea undelor infraroșii pentru a măsura distanțe se bazează pe același procedeu, emiterea unei unde luminoase și măsurarea timpului până când aceasta ajunge înapoi la traductor datorită reflectării de către un obstacol.

Traductorul este format din două componente, un LED care emite lumină în spectrul infraroșu și o fotodiodă care recepționează undele reflectate (Figura 3). Pentru a evita interferențele cu alte surse de lumină externe, semnalul luminos emis de către LED este modulată.



**Figura 3. Măsurarea distanțelor cu ajutorul undelor infraroșii**

La fel ca și în cazul măsurării distanțelor cu ultrasunete, și măsurarea cu ajutorul undelor infraroșii este afectată atenuarea puterii undelor în funcție de distanța parcursă de acestea și de proprietățile reflectante ale obstacolelor detectate (obiectele deschise la culoare reflectă lumina mai mult decât cele închise la culoare).

Avantaje și dezavantaje:

- Traductoarele ce utilizează unde infraroșii permit măsurarea de distanțe mici (spre deosebire de cele ce utilizează unde ultrasonice, care nu pot măsura distanțe sub anumită valoare minimă), dar au precizie mai mică.
- Traductoarele ce utilizează unde infraroșii au preț mai mic decât cele ce utilizează unde ultrasonice.
- Traductoarele ce utilizează unde infraroșii au timpi de răspuns mai mici decât cele ce utilizează unde ultrasonice.

### ✓ Descrierea aplicațiilor

Scopul acestor aplicații este de a realiza circuite electronice care să măsoare distanța/proximitatea până la un obstacol prin diferite metode și să o afișeze numeric pe un ecran LCD.

#### Aplicația 1. Măsurarea distanței utilizând un traductor cu unde ultrasonice

Pentru măsurarea distanței cu ajutorul ultrasunetelor se va folosi un traductor (LV-MaxSonar-EZ0) bazat pe un senzor de tip piezoceramic. Traductorul furnizează la ieșire mai multe tipuri de semnale, dintre care vor fi utilizate doar două, unul de tip PWM și unul analogic.

Semnalul digital de ieșire, de tip PWM, va fi aplicat uneia din intrările digitale ale plăcii Arduino. Pe baza factorului de umplere al acestuia se va calcula și afișa valoarea măsurată a distanței.

Semnalul analogic de ieșire va fi aplicat uneia din intrările analogice ale plăcii Arduino. Pe baza tensiunii analogice de la intrare placa va furniza o valoare digitală corespunzătoare, valoare ce va fi folosită pentru a calcula și afișa distanța măsurată.

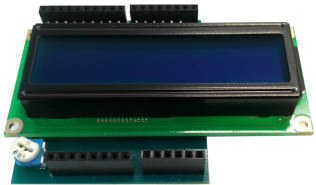



#### Aplicația 2. Măsurarea proximității utilizând un traductor cu unde infraroșii

Pentru măsurarea proximității cu ajutorul undelor infraroșii se va folosi un traductor (VCNL4000) ce conține un LED ce emite unde infraroșii și o fotodiodă ca receptor. Valoarea măsurată va fi transmisă către placa Arduino printr-o comunicație serială de tip I<sup>2</sup>C, folosind pinii de date SCL și SDA disponibili pe aceasta.

De reținut! Comunicația serială I<sup>2</sup>C (*Inter Integrated Circuit*) este un tip de comunicație *multi-master*, *multi-slave* inventată de Philips Semiconductor special pentru transmiterea de date între circuite integrate de viteză mică și procesoare sau microcontrolere. Magistrala de comunicație este formată din două linii, una pentru transmiterea/recepționarea datelor, SDA (*Serial Data Line*) și una pentru transmiterea/recepționarea semnalului de ceas, SCL (*Serial Clock Line*). Este obligatorie montarea câte unui rezistor de ridicare la 1 pe fiecare dintre cele două linii de date, iar fiecare circuit conectat la o magistrală I<sup>2</sup>C trebuie să aibă o adresă proprie.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>LCD Shield</i>	Afișare pe 2 rânduri a câte 16 caractere	1	
<i>Fir de legătură</i>	Tată-Tată	10	
<i>Traductor ultrasonic de distanță</i>	LV-MaxSonar-EZ0	1	
<i>Traductor de proximitate și ambient</i>	VCNL-4000	1	

### ✓ Observații

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 4 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

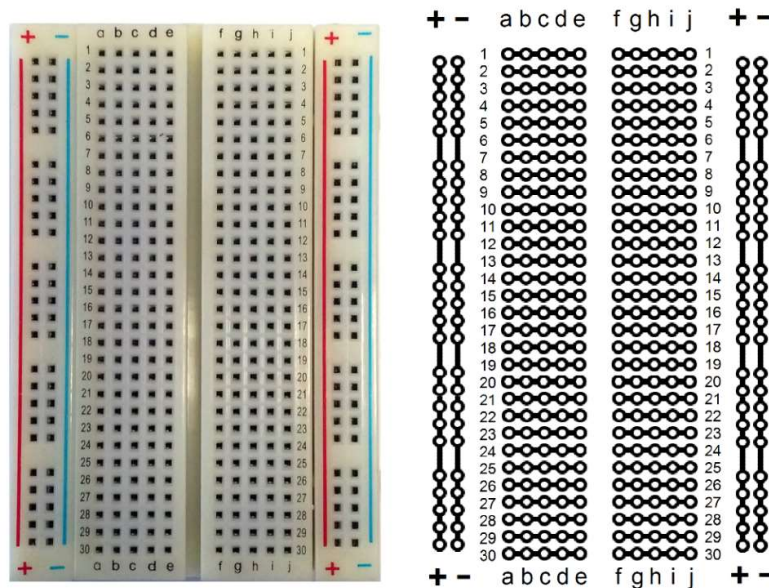


Figura 4. Breadboard-ul și conexiunile interne

**Traductorul ultrasonic de distanță** măsoară distanța până la obiectele aflate în fața acestuia folosind unde ultrasonice cu frecvența de 42 kHz, fiind bazat pe un senzor de tipul LV-MaxSonar-EZ0. Senzorul poate măsura distanțe între 0 și 6,45 m (254 inch), cu o rezoluție de 2,54 cm (1 inch). Deși senzorul poate detecta obiecte între 0 și 15 cm (6 inch), datorită unor constrângeri de natură fizică distanța nu poate fi măsurată cu exactitate iar indicația senzorului pentru toată această plajă de valori va fi de 15 cm [2].

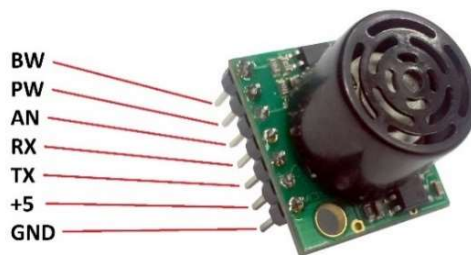


Figura 5. Traductor ultrasonic de distanță realizat cu senzorul LV-MaxSonar-EZ0

Traductorul dispune de trei tipuri de ieșiri de date:

- O ieșire de tip PWM (Pulse-Width Modulation, pinul PW) ce presupune variația controlată a formei tensiunii de ieșire prin schimbări rapide ale valorii logice din 1 în 0, în funcție de un factor de umplere (durata de timp cât semnalul are valoarea 1 logic). Distanța poate fi calculată ținând cont că factorul de umplere variază cu  $147 \mu\text{s}/\text{inch}$ .
- O ieșire analogică (pinul AN) a cărei tensiune variază cu  $V_{CC}/512$  per inch, adică  $9,8 \text{ mV}/\text{inch}$  dacă traductorul este alimentat cu o tensiune de 5 V.
- O ieșire serială (pinul TX) ce furnizează date în mod asincron în formatul RS232.

Ceilalți pini au următoarele roluri:

- Pinul BW activează ieșirea serială TX dacă i se aplică 0 logic sau dacă este neconectat. Dacă i se aplică 1 logic pinul TX trimite un puls în loc de date seriale (util pentru înscrierea mai multor traductoare atunci când senzorii sunt prea apropiați și pot apărea interferențe).
- Pinul RX este folosit pentru a comanda o măsurare a distanței, dacă i se aplică 1 logic sau dacă este neconectat. Dacă i se aplică 0 logic, traductorul va opri măsurarea.

Traductorul măsoară distanța începând de la partea din față a senzorului, așa cum este reprezentat în Figura 6.



**Figura 6. Măsurarea distanței cu traductorul LV-MaxSonar-EZ0**

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5 \text{ V}$ .

*Măsurarea distanței utilizând ieșirea PWM a traductorului*

Pentru calculul valorii distanței măsurate se determină mai întâi valoarea duratei unui impuls logic PWM aplicat intrării digitale a plăcii Arduino, folosind funcția *PulseIn* (funcție ce măsoară durata de timp a valorii 1 logic sau 0 logic a unui semnal digital), valoarea furnizată de funcție fiind exprimată în microsecunde. Știind că factorul de umplere al unui impuls (durata cât acel impuls are valoarea 1 logic) variază cu 147  $\mu\text{s}/\text{inch}$  valoarea distanței se poate calcula cu formula:

$$\text{distanța}_{pw} = \frac{\text{durata\_impuls}}{147} \text{ [inch]} \quad (1)$$

Deoarece 1 inch = 2,54 cm, pentru a exprima distanța în metri formula va fi următoarea:

$$\text{distanța}_{pw} = \frac{\text{durata\_impuls}}{147} \cdot \frac{2,54}{100} \text{ [m]} \quad (2)$$

*Măsurarea distanței utilizând ieșirea AN a traductorului*

Tensiunea analogică de la această ieșire a traductorului variază cu un prag de aproximativ 9,8 mV/inch (mai exact  $\frac{V_{CC}}{512}$  per inch) dacă senzorul este alimentat cu tensiunea de 5 V. Tensiunea de la ieșire este citită de portul analogic al plăcii Arduino, iar aceasta furnizează o valoare digitală corespunzătoare (*val\_dig\_an*) în domeniul 0 – 1023.

Prin urmare, fiecare variație a distanței măsurate cu un inch (512 valori) este tradusă de către placa Arduino într-o variație din 2 în 2 a valorii digitale corespunzătoare:

$$\text{val\_inch} = \frac{\text{val\_dig\_an}}{2} \text{ [inch]} \quad (3)$$

Deoarece 1 inch = 2,54 cm, pentru a exprima distanța în metri formula va fi următoarea:

$$\text{distanța}_{an} = \text{val\_inch} \cdot \frac{2,54}{100} \text{ [m]} \quad (4)$$

De reținut! Măsurătorile pot fi influențate de erori temporare, iar pentru a obține o valoare cât mai stabilă și corectă a distanței se vor face mai multe măsurători (în cazul de față 5). Se poate utiliza ca rezultat valoarea



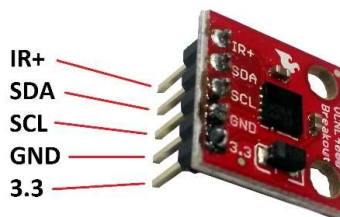
medie (adunarea tuturor valorilor și împărțirea sumei totale la 5) sau valoarea mediană (aceasta se obține prin sortarea celor 5 valori în ordine crescătoare și alegerea ca rezultat a celei din mijloc), aceasta din urmă fiind o metodă mult mai sigură de eliminare a valorilor eronate și alegere a celei corecte.

**Traductorul de proximitate** măsoară proximitatea până la obstacolele din apropierea acestuia, până la maxim 200 mm cu o rezoluție de 16 biți, fiind bazat pe un senzor de tipul VCNL4000 [3]. Datele de ieșire sunt puse la dispoziție printr-o magistrală de tip I<sup>2</sup>C.

Traductorul nu poate fi utilizat pentru măsurarea exactă a distanței datorită faptului că citirea acesteia nu este liniară, precum și din cauza reflectării diferite a undelor de către obiecte de culori diferite aflate la aceeași distanță de senzor. Astfel, traductorul poate fi utilizat numai pentru a determina gradul de apropiere sau depărtare a unui obiect față de acesta.

Lungimea de undă a luminii emise are o valoare de vârf de 890 nm, iar pentru modulația acesteia se poate alege între patru valori ale frecvenței purtătoare: 390,625 kHz, 781,25 kHz, 1,5625 MHz sau 3,125 MHz.

Curentul prin dioda LED poate fi stabilit la valoarea optimă prin programare între 10 mA și 200 mA, cu pas de 10 mA, în funcție de distanța necesară a fi măsurată.



**Figura 7. Traductor de proximitate realizat cu senzorul VCNL4000**

Pini traductorului au următoarele roluri:

- Pinul IR+ este folosit pentru alimentarea diodei LED, fiind necesară o tensiune de 5 V.
- Pini SDA și SCL sunt utilizați pentru conectarea la magistrala I<sup>2</sup>C.
- Pinul GND este folosit pentru legătura la masă.
- Pinului 3.3 este utilizat pentru alimentarea traductorului, fiind necesară o tensiune de 3,3 V.

În ceea ce privește necesitatea utilizării rezistoarelor de ridicare la 1 pe fiecare dintre cele două linii de date, dacă se utilizează un singur dispozitiv pe magistrală atunci nu sunt necesare rezistoare externe, cele incluse (și activate automat) în microcontrolerul de pe placa Arduino fiind suficiente [4].

**Atenție! Traductorul se va alimenta cu tensiunea  $V_{CC} = 3,3\text{ V}$ .**

#### *Măsurarea proximității*

Deoarece citirea datelor de la traductor se face printr-o magistrală I<sup>2</sup>C va trebui ca în secvența de cod să fie inclusă și biblioteca *Wire.h*, disponibilă deja în pachetul de biblioteci preinstalate în Arduino IDE. Traductorul trebuie mai întâi inițializat și programat, apoi valorile măsurătorilor se pot citi din registrele interne folosind funcții specifice utilizării magistralei I<sup>2</sup>C.

Traductorul este un dispozitiv de tip *slave* și are o adresă fixă, 13h (0x13). Are 12 registre de 8 biți accesibile, dar numai o parte vor fi utilizate în această aplicație [3].

Programarea traductorului se face prin scrierea de date în registrele 83h, 89h și 8Ah, conform celor ce urmează.

Stabilirea valorii curentului prin dioda LED se face prin scrierea registrului 83h. Semnificația fiecărui bit al acestuia este următoarea:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Fuse prog ID		Valoare curent LED					
<b>Descriere</b>							
Fuse prog ID		Informație folosită pentru setare inițială. Biți doar pentru citire.					
Valoare curent LED		Valoare curent = Valoare biți (în zecimal) x 10 mA, maxim 200 mA.					

În acest registru se va scrie valoarea 20, pentru a obține curentul maxim prin dioda LED.

Stabilirea frecvenței de modulare a semnalului de măsurare a proximității se face prin scrierea registrului 89h. Semnificația fiecărui bit al acestuia este următoarea:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Indisponibili					Frecvență semnal		
<b>Descriere</b>							
Frecvență semnal	Biți care stabilesc frecvența de modulare. Sunt posibile patru valori: 0 = 3,125 MHz, 1 = 1,5625 MHz, 2 = 781,25 kHz (Implicit), 3 = 390,625 kHz						

În acest registru se va scrie valoarea 2 corespunzător frecvenței de 781,25 kHz.

Ajustarea timpilor modulatorului presupune stabilirea unei valori pentru întârzierea de timp semnalul emis și evaluarea semnalului recepționat, precum și a unei valori pentru timpul mort în evaluarea semnalului recepționat.

Aceste setări se fac registrului 8Ah. Semnificația fiecărui bit al acestuia este următoarea:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Timpul de întârziere al modulației			Indisponibili		Timpul mort al modulației		
<b>Descriere</b>							
Timpul de întârziere al modulatorului	Stabilirea unui timp de întârziere între semnalul emis de LED și evaluarea semnalului recepționat, în vederea compensării întârzierilor datorate LED-ului și a fotodiodei. Reglarea corectă permite optimizarea nivelului semnalului măsurat.						
Timpul mort al modulatorului	Stabilirea unui timp mort în evaluarea semnalului recepționat, necesar pentru reducerea posibilelor efecte perturbatoare.						

În acest registru se va scrie valoarea 129, conform indicațiilor producătorului [3].

Traductorul nu realizează măsurători continue după ce este inițializat ci așteaptă cereri de măsurare din partea unui dispozitiv *master*.

Comanda unei măsurători se realizează prin scrierea de date în registrul 80h. Semnificația fiecărui bit al acestuia este următoarea:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
config_lock	als_data_rdy	prox_data_rdy	als_od	prox_od	Indisponibili		
Descriere							
config_lock	Bit doar pentru citire. Valoarea este 1.						
als_data_rdy	Bit doar pentru citire. Când valoarea este 1 logic, datele măsurătorii luminii ambientale sunt disponibile în registrele 85h și 86h. După ce acestea sunt citite, bitul este trecut în 0 logic.						
prox_data_rdy	Bit doar pentru citire. Când valoarea este 1 logic, datele măsurătorii proximității sunt disponibile în registrele 87h și 88h. După ce acestea sunt citite, bitul este trecut în 0 logic.						
als_od	Scrierea valorii 1 logic în acest bit activează o măsurătoare la cerere pentru lumina ambientală. După terminarea măsurării datele sunt disponibile în registrele 85h și 86h, iar bitul 7 este trecut în 1 logic.						
prox_od	Scrierea valorii 1 logic în acest bit activează o măsurătoare la cerere pentru proximitate. După terminarea măsurării datele sunt disponibile în registrele 87h și 88h, iar bitul 6 este trecut în 1 logic.						

În acest registru se va scrie valoarea rezultată din următorul calcul:

$$\text{continut\_registru\_80h} | 0x08 \quad (5)$$

Mai exact se realizează operația SAU logic între biții existenți în registrul 80h și octetul 00001000 (adică 0x08 în hexazecimal) pentru a se păstra conținutul registrului, mai puțin bitul 4, care va trece în 1 logic. Astfel se realizează comanda de măsurare a proximității.

După comanda de măsurare se așteaptă terminarea acesteia și punerea la dispoziție a datelor. Se verifică permanent bitul 6, urmărindu-se schimbarea valorii acestuia din 0 în 1 logic, cu ajutorul următorului calcul:

$$\text{continut\_registru\_80h} | 0x20 \quad (6)$$

Mai exact se realizează operația ȘI logic între biții existenți în registrul 80h și octetul 00100000 (adică 0x20 în hexazecimal), iar cu ajutorul unei bucle *while* se va urmări apariția unui rezultat diferit al calculului (adică modificarea bitului 6).

Valoarea rezultată a măsurătorii va fi stocată în două registre interne de câte 8 biți fiecare, având adresele 87h și 88h. Citirea datelor din acestea se realizează în felul următor:

- Se citește registrul 87h și se mută biții de date cu 8 biți spre stânga (de exemplu xxxxxxxx devine xxxxxxxx00000000).
- Se citește registrul 88h și se realizează operația SAU logic între datele citite curent și datele citite anterior (yyyyyyyy | xxxxxxxx00000000 = xxxxxxxxyyyyyyyy). Rezultatul format din 16 biți de date reprezintă valoarea proximității măsurate.

**Shield-ul LCD** permite afișarea de caractere pe un ecran cu cristale lichide, cu iluminare LED. Acesta se montează peste placa Arduino și are conectorii de așa natură încât pinii plăcii vor fi în continuare accesibili.

Ecranul LCD este format din 2 linii a câte 16 caractere, fiecare caracter fiind compus din 5x8 pixeli. Numerotarea coloanelor (caracterelor) se face de la 0 la 15 (de la stânga la dreapta), iar al rândurilor de la 0 la 1 (de sus în jos).

**Important!** Pentru a funcționa, *shield*-ul utilizează pinii digitali ai plăcii Arduino de la 2 până la 7 astfel: pinul 2 – *d7*, pinul 3 – *d6*, pinul 4 – *d5*, pinul 5 – *d4*, pinul 6 – *enable*, pinul 7 – *rs*.

### 3. Componente software

**LiquidCrystal.h** este biblioteca ce conține comenzile pentru *shield*-ul LCD.

**Wire.h** este biblioteca ce conține comenzile pentru magistrala I<sup>2</sup>C.

**LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)** creează o variabilă *lcd* specificându-se pinii digitali folosiți pentru a comanda *shield*-ul LCD.

**const** are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

**int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

`unsigned int variabilă = valoare` stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, fără semn (de la 0 până la 65.535).

`byte variabilă` stabilește o variabilă de tip octet, fără semn.

`float variabilă = valoare` stabilește o valoare pentru o variabilă de tip real în virgulă mobilă pe 32 de biți, cu semn (de la -3.4028235E+38 până la 3.4028235E+38). Numărul total de digiți afișați cu precizie este 6 – 7 (include toți digiții, nu doar cei de după virgulă).

`void setup()` este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

`void loop()` este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

`pinMode(pin, mode)` configurează pinul digital specificat ca intrare sau ca ieșire.

`Wire.begin()` este o funcție care inițializează magistrala I<sup>2</sup>C.

`Wire.beginTransmission(adresă)` este o funcție care deschide magistrala I<sup>2</sup>C în modul de transmitere/primire de date către/de la circuitul cu adresa specificată.

`Wire.endTransmission()` este o funcție care încheie transmiterea/primirea de date.

`Wire.write(byte(pointer_registru))` este o funcție care stabilește registru de unde se va începe operația de citire de date.

`Wire.requestFrom(adresă, n)` este o funcție care deschide magistrala I<sup>2</sup>C în modul de citire de date (un număr de  $n$  registre) de la circuitul cu adresa specificată.

`Wire.read()` este o funcție care citește datele registru cu registru și le oferă ca rezultat.

`analogRead(pin)` citește valoarea pinului analogic specificat.

*pulseIn(pin, valoare, expirare)* este o funcție ce returnează durata de timp în microsecunde a unui impuls logic, de valoare 0 (LOW) sau de valoare 1 (HIGH), recepționat pe un pin. Se poate stabili și un timp de expirare în microsecunde pentru a evita măsurarea impulsurilor prea lungi.

*for(inițializare, condiție, increment) {instrucțiune/ instrucțiuni }* repetă un bloc de instrucțiuni până la îndeplinirea condiției.

*while(expresie)* este o buclă executată în mod continuu și infinit până când expresia devine falsă.

*lcd.begin(coloane, rânduri)* inițializează interfața cu ecranul LCD și specifică numărul de rânduri și de coloane al acestuia.

*lcd.setCursor(coloană, rând)* stabilește poziția cursorului LCD. Pentru LCD-ul folosit în această aplicație numărul coloanelor este de la 0 la 15, iar al rândurilor de la 0 la 1.

*lcd.clear()* șterge ecranul LCD și poziționează cursorul în colțul din stânga sus.

*lcd.print()* afișează pe ecranul LCD datele (valori ale unor variabile)/textul dintre paranteze. Pentru a afișa un text este necesar ca acesta să fie plasat între ghilimele ("text"). Pentru a afișa valoarea unei variabile de tip *char*, *byte*, *int*, *long*, sau *string* se scrie numele variabilei și, opțional, baza de numerație a acesteia (*variabilă*, BIN sau DEC sau OCT sau HEX). Pentru a afișa valoarea unei variabile de tip *float* sau *double* se scrie numele variabilei iar după virgulă, numărul de zecimale dorit a se afișa (*variabilă*, nr. zecimală).

*return valoare* are rolul de a termina execuția unei funcții și de a returna o valoare.

*delay(ms)* pune în pauză programul pentru o durată de timp specificată în milisecunde.

*variabilă << n* este un operator care mută biții variabilei cu un număr de n poziții spre stânga.

*++* este folosit pentru a incrementa o variabilă

*&* este operatorul ȘI logic.

*|* este operatorul SAU logic.

/ este operatorul de împărțire care, în cazul împărțirii a două numere întregi, oferă rezultatul fără rest.

! este operatorul boolean NU (negație).

## 4. Aplicația 1. Măsurarea distanței utilizând un traductor cu unde ultrasonice

### 4.1. Realizarea montajului electronic

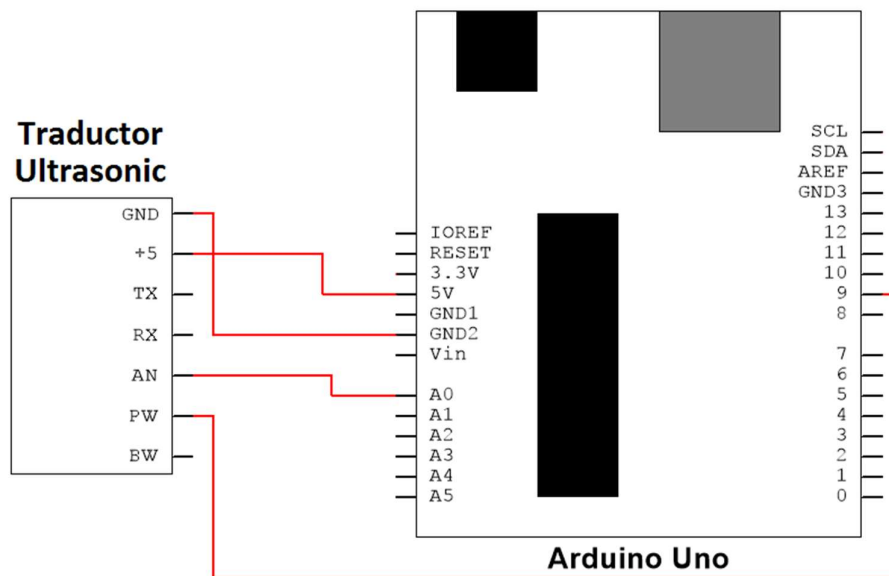


Figura 8. Schema de principiu pentru aplicația 1



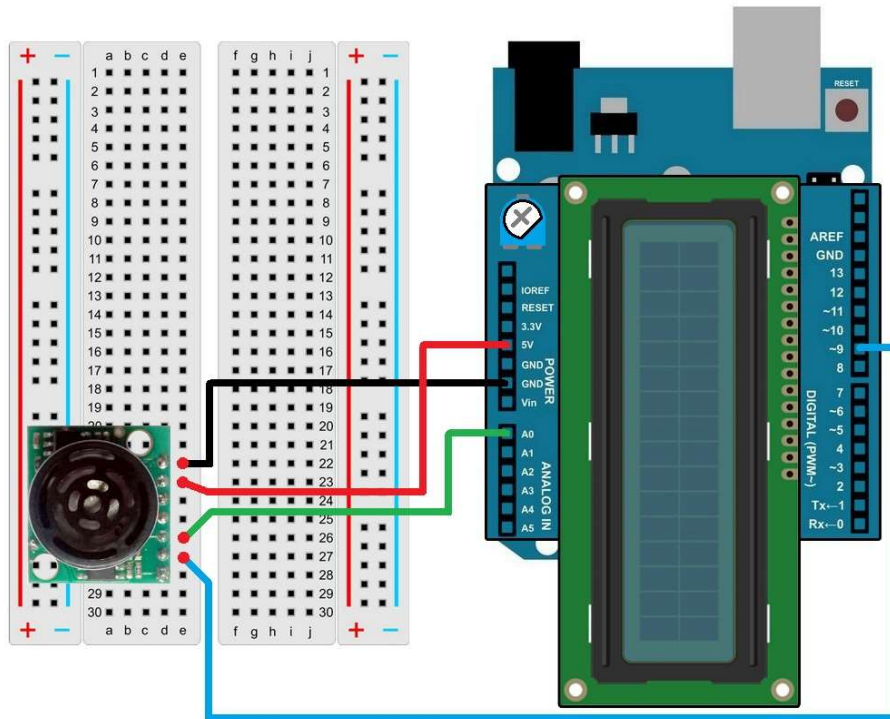
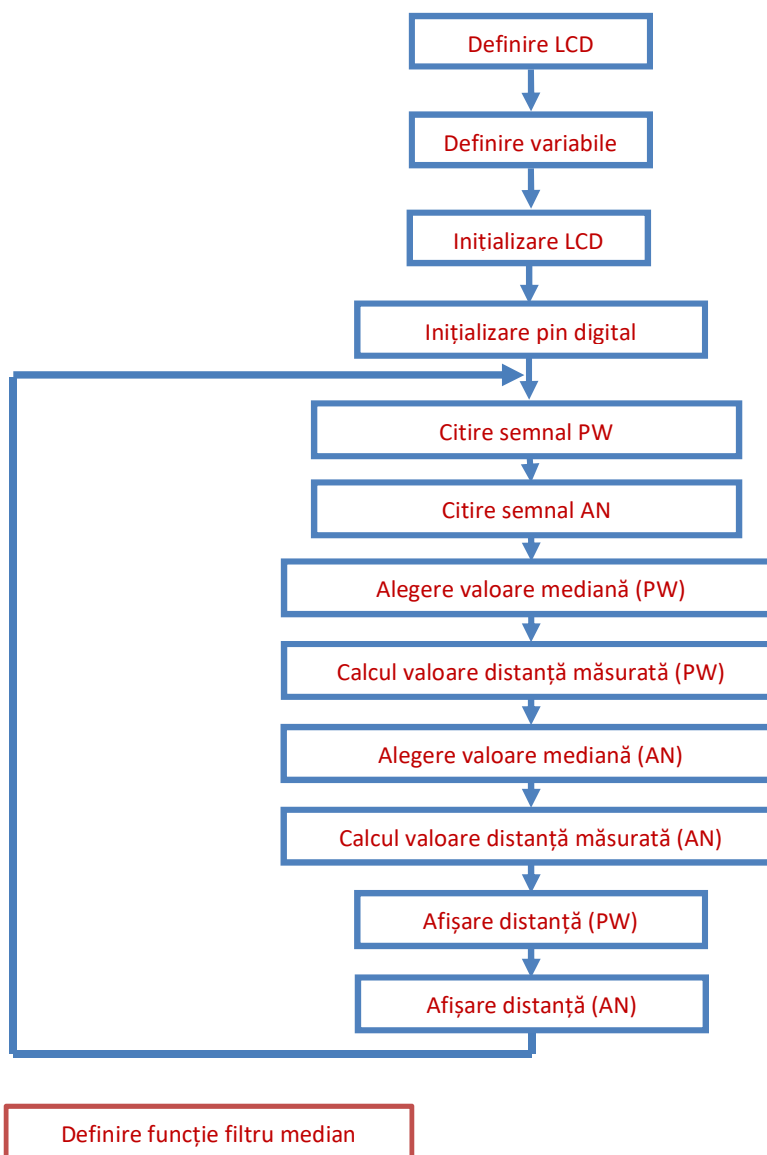


Figura 9. Realizarea conexiunilor electrice pentru aplicația 1

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului ultrasonic;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul +5 al traductorului ultrasonic;
- Pinul analogic A0 de pe placa Arduino se conectează cu un fir la pinul AN al traductorului ultrasonic;
- Pinul digital 9 de pe placa Arduino se conectează cu un fir la pinul PW al traductorului ultrasonic.

#### 4.2. Schema logică și secvența de cod



```
//Autor al programului de bază: Bruce Allen, 23.07.2009
#include <LiquidCrystal.h>
    //includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
    //inițializarea bibliotecii și a variabilei lcd cu numerele pinilor utilizați de
    shield-ul LCD
const int pwPin = 9;
    //definirea variabilei pwPin corespunzătoare portului digital 9 unde va fi
    conectată ieșirea PW a traductorului
const int anPin = 0;
    //definirea variabilei anPin corespunzătoare portului analogic A0 unde va
    fi conectată ieșirea AN a traductorului
unsigned int durata_impuls, val_inch;
    //definirea variabilelor pentru durata unui puls și pentru valoarea distanței
    în inch
float distanta_an, distanta_pw;
    //definirea variabilelor pentru distanțele calculate
int dim_matrice = 5;
    //definirea variabilei ce stabilește numărul de elemente al unei matrice
int element_median;
    //definirea variabilei elementului median al unei matrice
int valoare_pw[] = {0, 0, 0, 0, 0};
    //definirea unei matrice cu 5 elemente pentru valorile transmise prin
    semnal digital
int valoare_an[] = {0, 0, 0, 0, 0};
    //definirea unei matrice cu 5 elemente pentru valorile transmise prin
    semnal analogic

void setup() {
    lcd.begin(16, 2);
        //inițializarea interfeței cu ecranul LCD și specificarea numărului de
        rânduri și de coloane al acestuia
    pinMode(pwPin, INPUT);
        //se declară pinul pwPin ca fiind de intrare
    element_median = dim_matrice/2;
        //calculul valorii poziției elementului median al matricei
}

void loop() {
    //Citirea semnalelor de date
    for(int i = 0; i < dim_matrice; i++) {
        //buclă cu ajutorul căreia se populează cu valori cele două matrice
        valoare_pw[i] = pulseIn(pwPin, HIGH);
            //citirea valorii unui puls și stocarea ei ca element al matricei valoare_pw
    }
```

```
    valoare_an[i] = analogRead(anPin);
        //citirea valorii digitale a semnalului analogic și stocarea ei ca element al
        //matricei valoare_an
    delay(10);
}

//Calculul distanței utilizând semnalul digital
sortare(valoare_pw, dim_matrice);
    //se sortează valorile consecutive ale matricei ce conține duratele
    //pulsurilor recepționate de la traductor
durata_impuls = valoare_pw[element_median];
    //se extrage din matrice valoarea corespunzătoare elementului median
distanța_pw = ((durata_impuls/147)*2.54)/100;
    //se calculează distanța măsurată în metri, în funcție de valoarea mediană
    //a duratei unui puls – formula (2)

//Calculul distanței utilizând semnalul analogic
sortare(valoare_an, dim_matrice);
    //se sortează valorile consecutive ale matricei ce conține valorile digitale
    //ale semnalului recepționat de la traductor
val_inch = valoare_an[element_median]/2;
    //se extrage din matrice valoarea corespunzătoare elementului median și
    //se împarte la 2 pentru a obține distanța măsurată în inch – formula (3)
distanța_an = (val_inch*2.54)/100;
    //se calculează distanța măsurată în metri – formula (4)

//Afișarea valorilor distanțelor măsurate
lcd.clear();
    //ștergere conținut ecran LCD
lcd.print("Dist_pw = ");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.print(distanța_pw,3);
    //afișează pe ecranul LCD valoarea variabilei distanța_pw, cu 3 zecimale
    //după virgulă
lcd.print("m");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.setCursor(0, 1);
    //mutare cursor pe coloana 1, rândul 2
lcd.print("Dist_an = ");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.print(distanța_an,3);
    //afișează pe ecranul LCD valoarea variabilei distanța_an, cu 3 zecimale
    //după virgulă
lcd.print("m");
```

```

        //afișează pe ecranul LCD textul dintre ghilimele
        delay(500);
    }

//Filtru median creat de Bill Gentles, 12.11.2010
//Sortează elementele unei matrice în ordinea crescătoare a valorilor lor
void sortare(int *a, int n) {
for (int i = 1; i < n; ++i)
{
    int j = a[i];
    int k;
    for (k = i - 1; (k >= 0) && (j < a[k]); k--)
    {
        a[k + 1] = a[k];
    }
    a[k + 1] = j;
}
}
}

```

## 5. Aplicația 2. Măsurarea proximității utilizând un traductor cu unde infraroșii

### 5.1. Realizarea montajului electronic

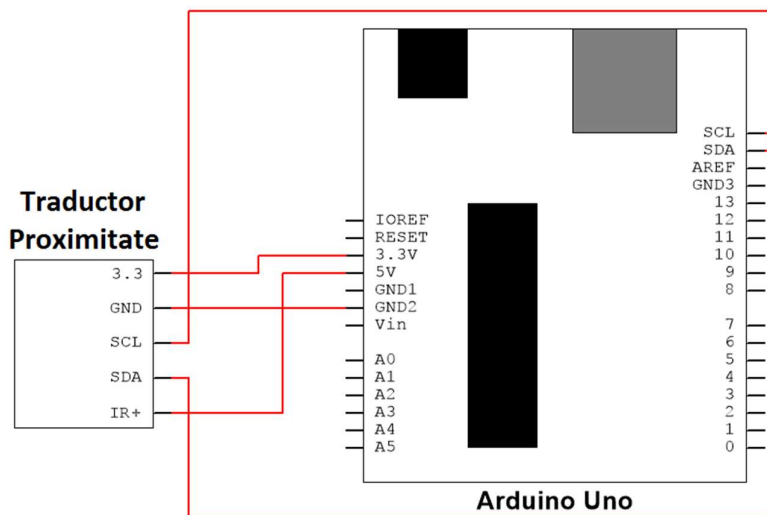
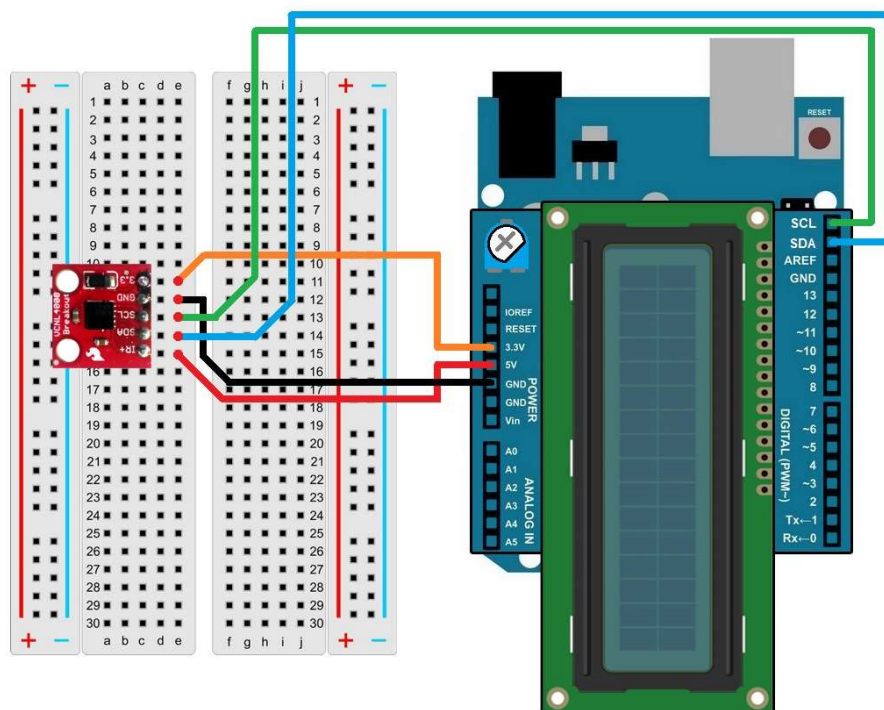


Figura 10. Schema de principiu pentru aplicația 2

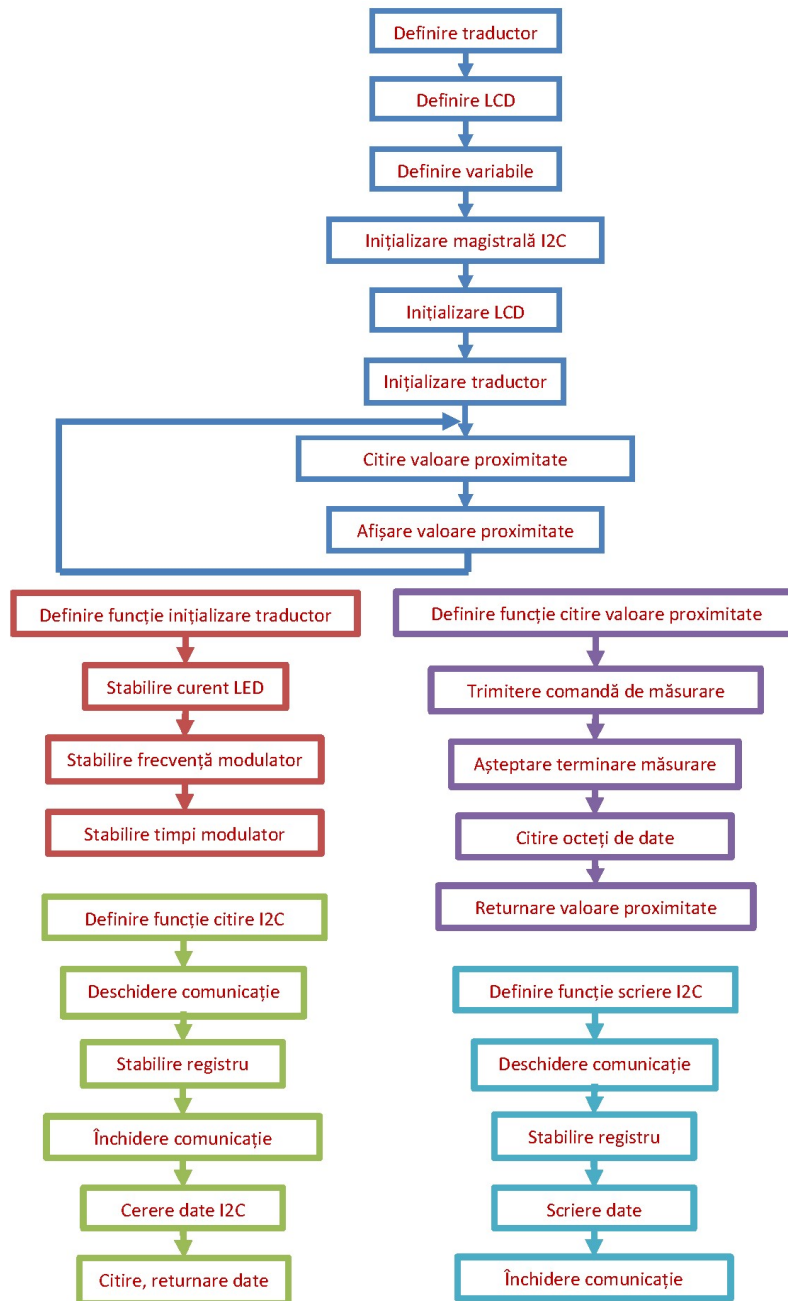


**Figura 11. Realizarea conexiunilor electrice pentru aplicația 2**

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului de proximitate;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul IR+ al traductorului de proximitate;
- Pinul 3.3V (power) de pe placa Arduino se conectează cu un fir la pinul 3.3 al traductorului de proximitate;
- Pinul SCL de pe placa Arduino se conectează cu un fir la pinul SCL al traductorului de proximitate;
- Pinul SDA de pe placa Arduino se conectează cu un fir la pinul SDA al traductorului de proximitate.

5.2. Schema logică și secvența de cod



```
//Autor al programului de bază: Jim Lindblom, 30.08.2011
#include <Wire.h>
    //includerea în program a bibliotecii comenzilor pentru magistrala I2C
#include <LiquidCrystal.h>
    //includerea în program a bibliotecii comenzilor pentru LCD
#define ADRESA_TRADUCTOR 0x13
    //definirea variabilei corespunzătoare adresei traductorului
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
    //inițializarea bibliotecii și a variabilei lcd cu numerele pinilor utilizați de
    shield-ul LCD
unsigned int val_proxy;
    //definirea variabilei pentru valoarea măsurată a proximității

void setup(){
    lcd.begin(16, 2);
        //inițializarea interfeței cu ecranul LCD și specificarea numărului de
        rânduri și de coloane al acestuia
    Wire.begin();
        //inițializarea magistralei I2C
    initializareTraductor();
        //inițializarea traductorului
}

void loop(){
    val_proxy = citesteProximitate();
        //citire valoare proximitate
    lcd.clear();
        //ștergere conținut ecran LCD
    lcd.print("Valoare proxy:");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.setCursor(0, 1);
        //mutare cursorul pe coloana 1, rândul 2
    lcd.print(val_proxy);
        //afișează pe ecranul LCD valoarea variabilei val_proxy
    delay(300);
}

void initializareTraductor(){
//Parametrii de inițializare ai traductorului
    scrieOctet(0x83, 20);
        //stabilirea curentului prin dioda LED la valoarea de 200 mA
    scrieOctet(0x89, 2);
        //stabilirea frecvenței de modulație a semnalului la valoarea de 781.25
        kHz
```



```
scrieOctet(0x8A, 0x81);
    //stabilirea timpilor modulatorului
}
unsigned int citesteProximitate(){
    byte temp = citesteOctet(0x80);
    //se citește registrul 80h
    scrieOctet(0x80, temp | 0x08);
    //se dă comanda de măsurare a proximității - formula (5)
    while (!(citesteOctet(0x80)&0x20));
    //se așteaptă până când s-a terminat măsurarea și datele sunt disponibile -
    //formula (6)
    unsigned int val_proximitate = citesteOctet(0x87) << 8;
    //se citește primul octet de date și se mută biții cu 8 poziții spre stânga
    val_proximitate = citesteOctet(0x88) | val_proximitate;
    //se citește al doilea octet de date și se adaugă la primul octet translatat
    return val_proximitate;
    //se returnează valoarea pe 16 biți a măsurătorii
}

void scrieOctet(byte registru, byte date){
    //Funcție care scrie datele specificate în registrul de la adresa specificată
    Wire.beginTransmission(ADRESA_TRADUCTOR);
    //deschide magistrala I2C în modul de transmitere de date către adresa
    //specificată
    Wire.write(registru);
    //stabilirea registrului de unde va începe următoarea operație
    Wire.write(date);
    //scriere date în registru
    Wire.endTransmission();
    //încheiere transmitere de date
}

byte citesteOctet(byte registru){
    //Funcție care citește datele din registrul de la adresa specificată și
    //returnează conținutul acestuia
    Wire.beginTransmission(ADRESA_TRADUCTOR);
    //deschide magistrala I2C în modul de transmitere de date către adresa
    //specificată
    Wire.write(registru);
    //stabilirea registrului de unde va începe următoarea operație
    Wire.endTransmission();
    //încheiere transmitere de date
    Wire.requestFrom(ADRESA_TRADUCTOR, 1);
}
```

```
    //deschide magistrala I2C în modul de citire de date (un număr de 1
    //registru) de la adresa specificată
byte date = Wire.read();
    //citire valoare proximitate măsurată
return date;
    //returnare valoare proximitate măsurată
}
```

## 6. Exerciții suplimentare și concluzii

1. Să se înlocuiască utilizarea valorii mediane a 5 citiri consecutive a distanței măsurate cu senzorul ultrasonic, cu utilizarea valorii medii a celor 5 citiri.
2. Să se testeze gradul de detecție a proximității unor obiecte de diferite culori, aflate la aceeași distanță de traductor.
3. Să modifice secvențele de cod astfel încât să se vizualizeze datele și pe monitorul serial.
4. Să se testeze gradul de detecție a proximității unor obiecte aflate la distanțe variabile de traductor. Distanța va fi măsurată cu traductorul ultrasonic.

Se utilizează traductoare ultrasonice și cu infraroșii deoarece este important ca acestea să aibă semnalele emise în afara benzilor de frecvență pe care le pot detecta senzorii umani (pentru ochi este vorba de infraroșu – lumina albă poate fi descompusă în lumini colorate elementare (ROGVAIV), lumina roșie având lungimea de undă 620-750 nm și frecvența 400-480 THz – lumina infraroșie, care nu este în spectrul vizibil are lungimea de undă 750 nm – 1 mm și frecvența 300 GHz – 400 THz).

PWM (Pulse Width Modulation) înseamnă modulația impulsurilor în durată și are ca rezultat obținerea unor semnale digitale care au factor de umplere variabil (durata impulsurilor este variabilă în funcție de un parametru). Traductorul de distanță va genera un astfel de semnal PWM iar durata impulsurilor (respectiv, factorul de umplere) se va modifica în funcție de distanța până la obiectul plasat în fața traductorului. Factorul de umplere, respectiv durata impulsului, va fi calculată ca fiind numărul de biți 1 care sunt recepționați pe intrarea digitală având ca referință durata elementară de bit (un exemplu simplificat: dacă durata sau intervalul elementar de bit este a zecea parte din durata unui semnal digital și factorul de umplere al

acestui este de 50%, înseamnă că vor fi 5 biți elementari de 1 pe durata impulsului recepționat).

Adresarea diferitelor componente și circuite electronice este foarte importantă atunci când acestea trebuie să comunice prin același mediu și trebuie să împartă resurse comune. În momentul în care mai multe componente sunt conectate la aceeași magistrală de comunicații trebuie dedicată magistrala acelei componente care are nevoie de comunicații la un moment dat. Acest lucru se face prin adresarea componentei și trecerea acesteia în modul activ de lucru, chiar dacă celelalte componente sunt conectate dar în mod pasiv (de exemplu stare de înaltă impedanță).

## 7. Bibliografie

- [1] **Murata Manufacturing Co., Ltd.**, „*Ultrasonic Sensor - Application Manual*,” 2008.
- [2] **MaxBotix Inc.**, „*LV-MaxSonar-EZ Series Datasheet*,” 2015.
- [3] **Vishay Semiconductors**, „*VCNL4000 Datasheet*,” Rev. 1.6, 24.08.2011.
- [4] **W. Truchsess**, „*Effects of Varying I2C Pull-Up Resistors*,” 18.12.2010.
- [5] **Vishay Semiconductors**, „*Designing VCNL4000 into an Application*,” 11.03.2011.



# Lucrarea 6. Măsurarea nivelului de iluminare

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează senzori și traductoare.
- Realizarea unei aplicații practice de măsurare a nivelului de iluminare folosind senzori digitali și analogici, și afișarea acestuia pe monitorul serial.

### 1.2. Descriere teoretică

#### ✓ Introducere

Oscilațiile câmpurilor magnetice și electrice, orientate perpendicular unul pe celălalt și care se generează reciproc, se numesc unde electromagnetice. Ca și orice oscilație, o mărime care le definește este perioada de repetare a oscilațiilor (u.m. „s”), iar inversa acesteia este frecvența (u.m. „Hz”).

Viteza de propagare a undelor electromagnetice în vid este de este de 299.792.458 m/s (se mai numește - viteza luminii), dar în funcție de mediul pe care acestea îl parcurg, viteza lor poate fi mai mică. Cunoscând viteza de propagare și frecvența undelor, se poate calcula lungimea de undă.

$$\lambda = \frac{v}{f} \quad (\text{u. m. „m”}) \quad (1)$$

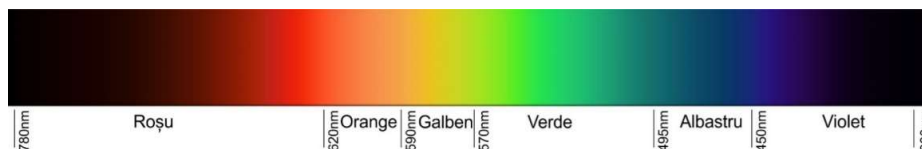
unde  $\lambda$  - este lungimea de undă,  $v$  - este viteza luminii în mediul respectiv și  $f$  - frecvența undelor electromagnetice.

Undele electromagnetice se pot clasifica după frecvența – sau lungimea de undă, astfel:

- Undele electromagnetice de radio-frecvență (de la câțiva Hz la GHz; de exemplu VHF are frecvențe între 30 MHz – 300 MHz și lungimi de undă 10 m – 1 m).
- Microunde (cu frecvențe între 1 GHz și 100 GHz, și lungimi de undă între 300 și 3 mm).
- Radiații Terahertz.
- Infraroșu (cu frecvențe între 300 GHz și 430 THz și lungimi de undă între 1 mm și 700 nm).
- Spectrul vizibil (uman) (cu frecvențe între 430 THz și 790 THz și lungimi de undă între 700 nm și 380 nm).
- Ultraviolet (cu frecvențe între 790 THz și 30 PHz și lungimi de undă între 10 nm și 380 nm).
- Raze X (cu frecvențe între 30 PHz și 30 EHz și lungimi de undă între 10 pm și 10 nm).
- Raze Gama (cu frecvențe mai mari de 30 EHz și lungimi de undă mai mici de 10 pm).

Lumina, percepută ca stimul de ochiul uman, este doar partea de spectru vizibil al undelor electromagnetice. Aceasta este caracterizată de:

- Culoare – este dată de frecvența (se măsoară în Hz) sau lungimea de undă a radiației (se măsoară în nanometri). Culoarea, din punct de vedere tehnic, nu este același lucru cu culoarea percepută de ochiul uman, acesta făcând o sinteză a trei culori elementare: roșu, verde și albastru (RGB – din engl. *red, green, blue*). Spectrul vizibil începe cu culoarea roșu (610-780 nm) și se termină cu culoarea violet (380-424 nm), așa cum se observă în Figura 1.



**Figura 1. Spectrul vizibil**

- Intensitate luminoasă – puterea emisă pe o direcție dată sau puterea transportată de radiație (u.m. „cd” – candela).

$$1 \text{ cd} = 1 \text{ W} / 683 \text{ sr} \text{ („sr” – steradian)}$$

- Fluxul luminos – cantitatea totală de radiație emisă de o sursă (u.m. „lm” – lumenul)

$$1 \text{ lm} = 1 \text{ cd} \cdot 1 \text{ sr}$$

Rezultă că  $1 \text{ cd} = 4\pi \text{ lm} \approx 12,57 \text{ lm}$  – fluxul total al unei surse luminoase punctiforme omnidirecționale cu intensitatea luminoasă de 1 cd.

- Iluminarea – intensitatea luminoasă repartizată pe o suprafață (u.m. „lx” – lux)

$$1 \text{ lx} = 1 \text{ lm/m}^2$$

- Polarizare – planurile de oscilație a undelor electromagnetice.
- Coerență – faza oscilațiilor.

Relațiile între unitățile de măsură enumerate mai sus sunt următoarele:

$$1 \text{ lx} = \frac{1 \text{ lm}}{\text{m}^2} = \frac{1 \text{ cd} \cdot 1 \text{ sr}}{\text{m}^2} = \frac{1 \text{ W}}{683 \text{ sr}} \cdot \frac{1 \text{ sr}}{\text{m}^2} = \frac{1 \text{ W}}{683 \text{ m}^2} \quad (2)$$

Condițiile de iluminare tipice pentru diferite medii sunt prezentate în tabelul următor [1]:

Condiții de iluminare	Intensitate luminoasă
Lună plină	1 lx
Iluminat stradal	10 lx
Iluminat casnic	30 ... 300 lx
Iluminat de birou	100 ... 1000 lx
Iluminat pentru operații medicale	10.000 lx
Lumina soarelui directă	100.000 lx

Pentru detecția sau măsurarea caracteristicilor enumerate se pot utiliza dispozitive sensibile la lumină, cum ar fi: fotorezistoare, fotodiode, celule fotovoltaice și fototranzistori.

*Fotorezistorul* - este o componentă electronică pasivă căreia i se modifică rezistența electrică în funcție de fluxul luminos. Sensibilitatea

fotorezistorului se măsoară în mA/lx la o tensiune constantă și este liniară pentru domenii relativ mari de iluminare, însă depinde mult de culoarea luminii (în funcție de materialul utilizat în construcția fotorezistorului). Funcționarea nu depinde de sensul tensiunii aplicate.

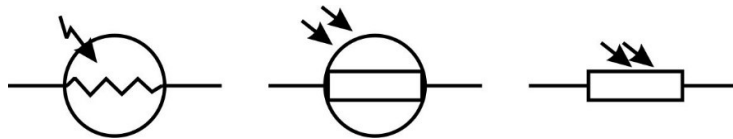


Figura 2. Simboluri utilizate pentru fotorezistor

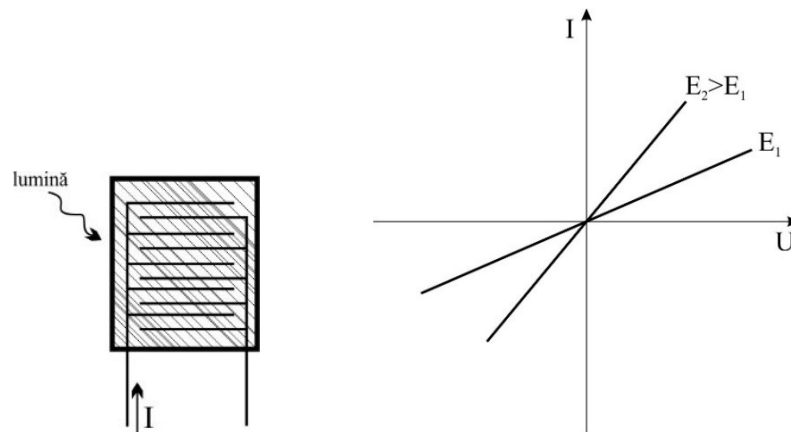


Figura 3. Construcția și caracteristica U-I a unui fotorezistor

*Fotodioda* – este o joncțiune pn și funcționarea se bazează pe efectul fotovoltaic. Prin iluminarea suprafeței active, la bornele diodei va apărea o tensiune electrică de la anod spre catod. O fotodiodă se utilizează polarizată invers, curentul invers fiind denumit curent de iluminare (pentru iluminare zero se numește curent de întuneric). Are o sensibilitate mai bună și un timp de răspuns mai mic decât a fotorezistoarelor.

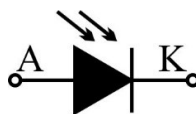


Figura 4. Simbolul utilizat pentru o fotodiodă



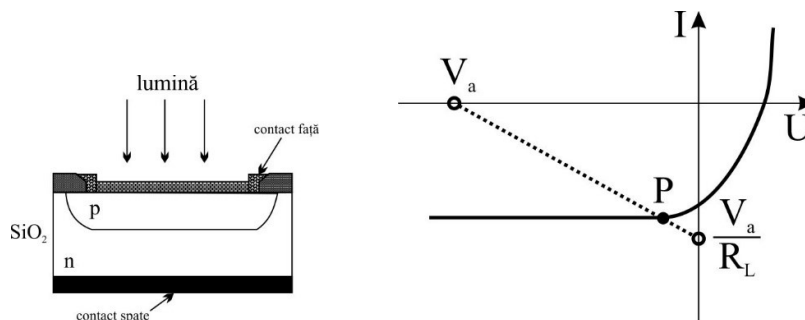


Figura 5. Construcția și caracteristica U-I a unei fotodiode

*Fototranzistorul* – este o combinație de două joncțiuni pn (npn sau pnp), la fel ca tranzistorii bipolari obișnuiți, la care regiunea bază-colector este iluminată. Lumina care cade pe fototranzistor generează curentul de bază necesar polarizării acestuia și duce la apariția unui curent de colector.

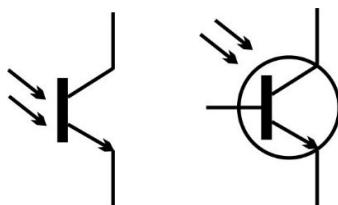


Figura 6. Simboluri utilizate pentru fototranzistor

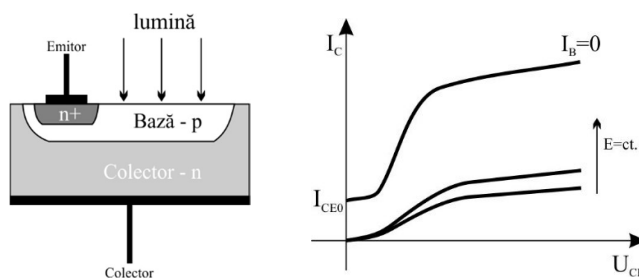


Figura 7. Construcția și caracteristica U-I a unui fototranzistor

✓ **Descrierea aplicațiilor**

Scopul acestor aplicații este de a realiza circuite electronice care să măsoare nivelul de iluminare prin diferite metode și să afișeze numeric valorile pe un ecran LCD sau pe monitorul serial.

### **Aplicația 1. Măsurarea nivelului de iluminare cu un senzor analogic**

Pentru măsurarea nivelului de iluminare se va folosi un traductor bazat pe un senzor (PT15-21C/TR8) de tip fototranzistor. Traductorul furnizează la ieșire o tensiune analogică ce va fi aplicată uneia din intrările analogice ale plăcii Arduino. Pe baza tensiunii analogice de la intrare, placa va furniza o valoare digitală corespunzătoare, valoare ce va fi folosită pentru a afișa pe monitorul serial nivelul de iluminare măsurat.

### **Aplicația 2. Măsurarea nivelului de iluminare cu un senzor digital**

Pentru măsurarea nivelului de iluminare se va folosi un traductor (TSL235R) bazat pe un senzor de tip fotodiodă. Traductorul furnizează la ieșire un semnal digital dreptunghiular având frecvența proporțională cu nivelul de iluminare măsurat de senzor. Placa Arduino va citi această frecvență și va afișa pe monitorul serial valoarea corespunzătoare.





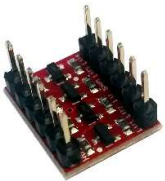
### **Aplicația 3. Măsurarea nivelului de iluminare cu un senzor RGB digital**

Pentru măsurarea nivelului de iluminare RGB se va folosi un traductor bazat pe senzorul ISL29125. Acesta conține o matrice de fotodiode ce descompune lumina în spectre de Roșu, Verde și Albastru (*Red, Green, Blue*) și măsoară nivelul de iluminare pentru fiecare dintre ele. Valorile măsurate vor fi transmise către placa Arduino, printr-o comunicație serială de tip I<sup>2</sup>C, folosind pinii de date SCL și SDA disponibili pe aceasta, iar apoi vor fi afișate pe monitorul serial.

De reținut! Comunicația serială I<sup>2</sup>C (*Inter Integrated Circuit*) este un tip de comunicație *multi-master, multi-slave* inventată de Philips Semiconductor special pentru transmiterea de date între circuite integrate de viteză mică și procesoare sau microcontrolere. Magistrala de comunicație este formată din două linii, una pentru transmiterea/recepționarea datelor, SDA (*Serial Data Line*) și una pentru transmiterea/recepționarea semnalului de ceas, SCL (*Serial Clock Line*). Este obligatorie montarea câte unui rezistor de ridicare la 1 pe fiecare dintre cele două linii de date, iar fiecare circuit conectat la o magistrală I<sup>2</sup>C trebuie să aibă o adresă proprie.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>Fir de legătură</i>	Tată-Tată	10	
<i>Traductor de lumină analogic</i>	PT15-21C/TR8	1	
<i>Traductor de lumină digital</i>	TSL235R	1	
<i>Traductor de lumină RGB digital</i>	ISL29125	1	
<i>Convertor de nivel logic</i>	BSS138	1	

### ✓ Observații

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 8 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

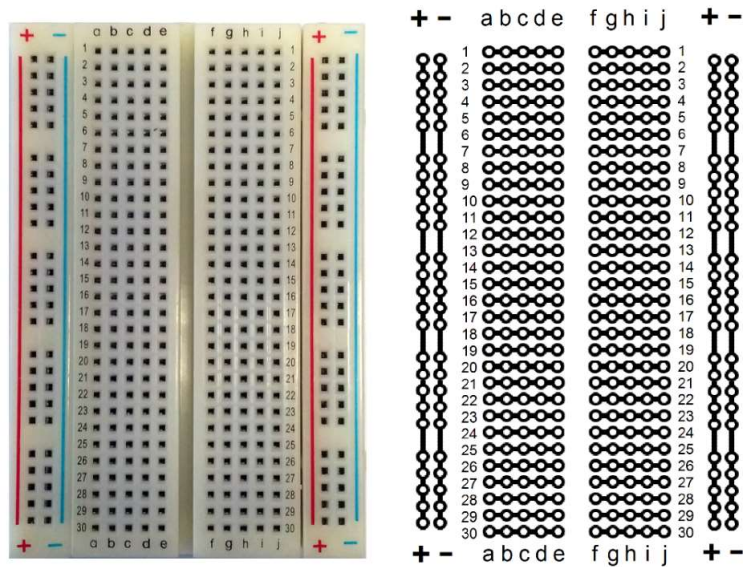


Figura 8. Breadboard-ul și conexiunile interne

**Traductorul de lumină analogic** măsoară nivelul de iluminare bazându-se pe utilizarea unui fototranzistor NPN de tipul PT15-21C/TR8, având timpi mici de răspuns și sensibilitate mare.

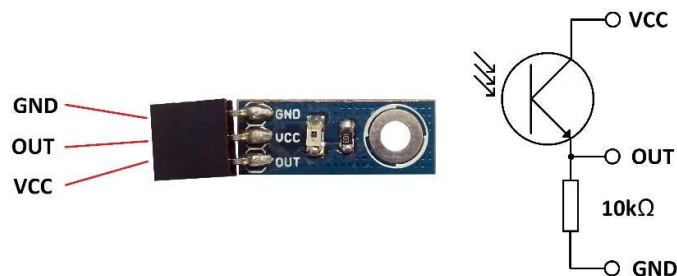


Figura 9. Traductor de lumină analogic

Traductorul conține, pe lângă senzor, și un rezistor conectat între pinul de ieșire al modulului (OUT) și masă (GND). Acesta se numește rezistor de coborâre la 0 (engl. *Pull down* [2], vezi Figura 9) și are rolul de a păstra valoarea logică 0 la ieșirea modulului atunci când nu există lumină. Stabilirea unui nivel logic sigur (0 în acest caz) împiedică apariția aleatorie a unei valori 0 sau 1 la intrarea digitală a plăcii Arduino datorită unor eventuale zgomote electrice [3].

Lățimea de bandă a spectrului luminii ce poate fi detectată de traductor este între 400 și 1100 nm, cea mai mare sensibilitate având-o în jurul valorii de 940 nm [4].

Caracteristica traductorului este liniară, așa cum se poate vedea și din Figura 10, însă acesta nu este etalonat.

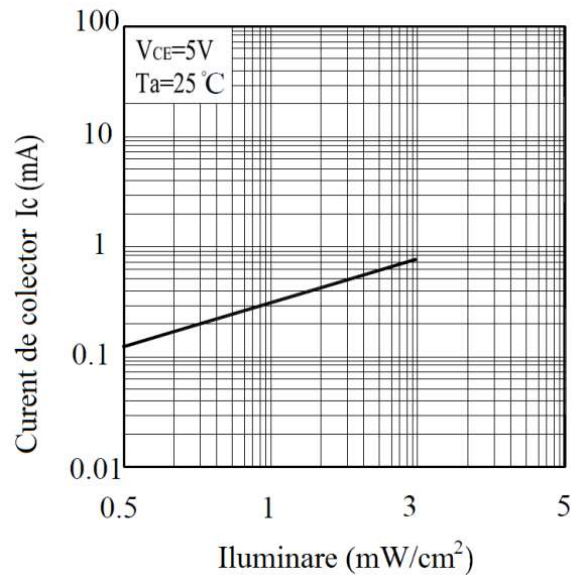


Figura 10. Caracteristica traductorului de lumină analogic [4]

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5 V$ .

#### *Măsurarea nivelului de iluminare*

În funcție de cantitatea de lumină care cade pe senzor placa Arduino va furniza o valoare digitală corespunzătoare între 0 (lipsă lumină) și 1023.

**Traductorul de lumină digital** măsoară nivelul de iluminare bazându-se pe utilizarea unei fotodiode și a unui convertor curent – frecvență, ambele integrate în traductorul de tip TSL235R. Acesta furnizează la ieșire un semnal dreptunghiular având frecvența direct proporțională cu intensitatea luminii ce cade pe fotodiodă. Lățimea de bandă a spectrului luminii ce poate fi detectată de traductor este între 320 și 1050 nm [5].



Figura 11. Traductor de lumină digital

Traductorul conține, pe lângă senzor, și un condensator conectat între pinul VCC (3.3V) și masă (GND) cu rol de decuplare.

Caracteristica traductorului este aproximativ liniară, așa cum se poate vedea și din Figura 12 (cele două axe fiind logaritmice).

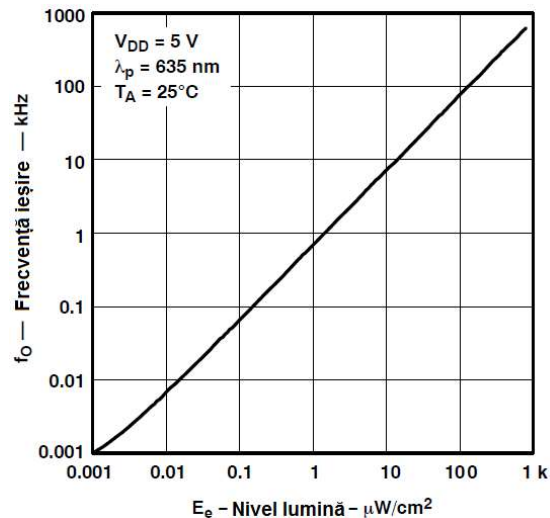


Figura 12. Caracteristica traductorului de lumină digital [5]

**Atenție! Traductorul se va alimenta cu tensiunea  $V_{CC} = 3,3$  V.**

*Măsurarea nivelului de iluminare*

Având în vedere că pentru un nivel de iluminare de  $430 \mu\text{W}/\text{cm}^2$  frecvența tipică a semnalului de ieșire este de 250 kHz [5] și ținând cont de liniaritatea caracteristicii traductorului rezultă următoarea formulă de calcul a nivelului de iluminare:

$$E_e = \frac{430}{250} \cdot f_o = 1,72 \cdot f_o [\mu\text{W}/\text{cm}^2] \quad (3)$$

unde  $f_o$  se măsoară în kHz.

Pentru măsurarea frecvenței semnalului de la ieșirea traductorului cu placa Arduino s-a ales utilizarea întreruperilor, cu ajutorul funcției *attachInterrupt*. Funcția monitorizează unul din pinii digitali de intrare (în cazul plăcii Arduino Uno pot fi utilizați numai pinii 2 și 3) și activează o funcție specială (numită ISR – *Interrupt Service Routine*) atunci când o anumită condiție este îndeplinită (în cazul de față: trecerea din 0 în 1 a nivelului logic aplicat pinului monitorizat).

Calculul frecvenței presupune numărarea trecerilor din 0 în 1 logic ale semnalului de la traductor ce apar într-o perioadă de timp de 1 secundă. Pentru aceasta, funcția ISR va trebui să conțină un simplu contor ce va fi incrementat la fiecare impuls citit.

De reținut! Funcția ISR nu poate avea parametri și nici nu poate să returneze un rezultat, iar în interiorul ei nu pot fi utilizate funcții precum *millis()*, *micros()* sau *delay()* deoarece acestea se bazează pe întreruperi [6].

**Traductorul de lumină RGB digital** de tipul ISL29125 măsoară nivelul de iluminare pentru fiecare din cele trei spectre ale luminii, roșu, verde, albastru, folosind un senzor alcătuit dintr-o matrice de fotodiode. Datele de ieșire sunt puse la dispoziție printr-o magistrală de tip I<sup>2</sup>C, traductorul fiind un dispozitiv de tip *slave*.

Traductorul dispune de două domenii de sensibilitate optică (5,7 mlux ... 375 lux și 0,152 lux ... 10.000 lux) cu rezoluția de 12 sau 16 biți, selectabile prin programarea acestuia. Undele infraroșii sunt respinse, precum și zgomotele cu frecvențe de 50 Hz sau 60 Hz datorate surselor artificiale de lumină [7].

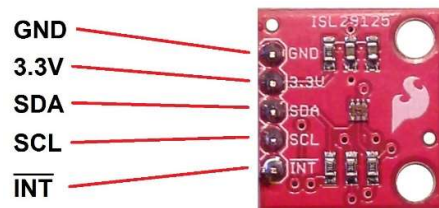


Figura 13. Traductor de lumină RGB

Pini traductorului au următoarele roluri:

- Pinul INT este folosit pentru activarea unei întreruperi.
- Pini SDA și SCL sunt utilizați pentru conectarea la magistrala I<sup>2</sup>C.

- Pinul GND este folosit pentru legătura la masă.
- Pinului 3.3V este utilizat pentru alimentarea traductorului, fiind necesară o tensiune de 3,3 V.

Traductorul conține și două rezistoare de 10 k $\Omega$  conectate între fiecare dintre pinii de date SCL și SDA, și V<sub>CC</sub>, cu rol de rezistoare de ridicare la 1 (vezi Figura 14). Prin eliminarea fludorului de pe jumper-ul încercuit cu verde în Figura 14 se renunță la cele două rezistoare, lucru util atunci când o magistrală I<sup>2</sup>C este împărțită de mai multe module, fiecare având câte un set de rezistoare de ridicare la 1, sau când se utilizează rezistoare externe sau pe cele incluse (și activate automat) în microcontrolerul ATmega328 de pe placa Arduino [8].



Figura 14. Utilizarea rezistoarelor de ridicare la 1

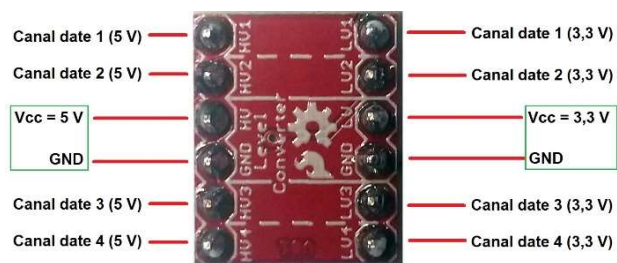
**Atenție! Traductorul se va alimenta cu tensiunea V<sub>CC</sub> = 3,3 V.**

Traductorul de lumină RGB funcționează cu tensiunea de alimentare de 3,3 V, prin urmare conexiunea la magistrala I<sup>2</sup>C va trebui să aibă tot nivel logic 3,3 V. Deoarece placa Arduino utilizează pentru I<sup>2</sup>C un nivel logic de 5 V, pentru a conecta traductorul la aceasta este necesar un convertor de nivel logic.

Convertorul de nivel logic are rolul să separe două circuite care utilizează niveluri logice având tensiuni diferite, în cazul de față 3,3 V (notat LV) și 5 V (notat HV). Convertorul preia valorile logice primite (0 sau 1) și le transmite la ieșire modificând numai valoarea tensiunii semnalului.

Pentru această lucrare se va utiliza un convertor de nivel logic bidirecțional cu patru canale [9], ce poate transfera valorile logice între două circuite în ambele sensuri (Figura 15).





**Figura 15. Convertor de nivel logic bidirecțional cu patru canale**

Vor fi utilizate două dintre canalele de date, unul pentru conexiunea SCL și unul pentru cea SDA, ale magistralei I<sup>2</sup>C.

#### *Măsurarea nivelului de iluminare pentru fiecare componentă RGB*

Deoarece citirea datelor de la traductor se face printr-o magistrală I<sup>2</sup>C va trebui ca în secvența de cod să fie inclusă și biblioteca *Wire.h*, disponibilă deja în pachetul de biblioteci preinstalate în Arduino IDE.

Măsurarea și determinarea valorilor nivelurilor de iluminare pentru componentele RGB se face în mod automat de către acesta, iar pentru afișarea lor este necesară citirea semnalului de date. Pentru acest traductor se poate utiliza în secvența de cod o bibliotecă dezvoltată special [10], numită *SparkFunISL29125.h* (biblioteca trebuie descărcată de pe Internet și importată în Arduino IDE folosind tab-ul *Skech* -> *Import Library...* -> *Add Library...*).

Pașii necesari pentru obținerea valorilor nivelurilor de iluminare sunt:

- Definirea senzorului.
- Inițializarea senzorului.
- Citirea valorii nivelului de iluminare pentru culoarea roșie (*rosu*) folosind funcția *traductor\_RGB.readRed()*.
- Citirea valorii nivelului de iluminare pentru culoarea verde (*verde*) folosind funcția *traductor\_RGB.readGreen()*.
- Citirea valorii nivelului de iluminare pentru culoarea albastră (*albastru*) folosind funcția *traductor\_RGB.readBlue()*.

În funcție de rezoluția convertorului analogic/digital al traductorului, domeniul de valori ce rezultă în urma citirii datelor furnizate de traductor poate fi:

- $0 \dots 2^{12} - 1 = 4.095$ , pentru rezoluția de 12 biți.
- $0 \dots 2^{16} - 1 = 65.535$ , pentru rezoluția de 16 biți.

Aceste valori nu descriu însă intensitatea luminoasă într-un format cunoscut. Două dintre formatele cunoscute, precum și modul de transformare sunt următoarele:

- În mod standard, domeniul de valori ce descrie intensitatea luminoasă este  $0 \dots 1$  (de exemplu  $R = 0,63$ ,  $G = 0,25$ ,  $B = 0,41$ ). Pentru a o afișa astfel se împarte valoarea rezultată în urma citirii la valoarea maximă a domeniului de măsurare (4.095 sau 65.535).
- Un alt domeniu de valori adesea utilizat este  $0 \dots 255$  (de exemplu  $R = 147$ ,  $G = 244$ ,  $B = 85$ ). Pentru a afișa astfel intensitatea luminoasă se împarte valoarea rezultată în urma citirii la  $\frac{4095}{255}$ , respectiv la  $\frac{65535}{255} = 257$ .

De reținut! Biblioteca *SparkFunISL29125.h* are prestabilită rezoluția de 16 biți.

#### *Calculul nivelului general de iluminare*

În afară de formatul RGB, care creează o anumită culoare pe baza a trei culori fundamentale și care diferă de la un sistem la altul în funcție de acuratețea cu care cele trei culori de bază sunt produse, se mai utilizează și formatul XYZ. Acesta este considerat un format general, cu ajutorul căruia poate fi definită orice culoare vizibilă ochiului uman.

Conversia între cele două formate se face cu ajutorul unei matrice care conține coeficienții de transformare [7].

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} C_{XR} & C_{XG} & C_{XB} \\ C_{YR} & C_{YG} & C_{YB} \\ C_{ZR} & C_{ZG} & C_{ZB} \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (4)$$

Valoarea Y a unei culori descrise prin formatul XYZ reprezintă intensitatea luminoasă a acesteia.

Astfel, intensitatea luminoasă măsurată în lux poate fi calculată cu următoarea formulă [7]:

$$Y = E_V = \left( C_{YR} \cdot \frac{rosu}{2^n - 1} + C_{YG} \cdot \frac{verde}{2^n - 1} + C_{YB} \cdot \frac{albastru}{2^n - 1} \right) \cdot domeniu [lux] \quad (5)$$

unde:

- coeficienții  $C$  sunt coeficienți de transformare din formatul RGB în formatul XYZ;
- valoarea intensității luminoase pentru fiecare culoare va fi în domeniul  $0 \dots 1$ , conform celor prezentate anterior (din acest motiv s-a folosit împărțirea la  $2^n - 1$ ,  $n$  fiind rezoluția convertorului analogic/digital al traductorului);
- *domeniul* de măsurare a intensității luminoase poate fi de 375 lux sau de 10.000 lux, în funcție de modul de configurare al traductorului.

De reținut! Biblioteca *SparkFunISL29125.h* are prestabilite rezoluția de 16 biți și domeniul de 10.000 lux.

Există multe spații RGB definite, în funcție de scopul în care acestea sunt utilizate, cel mai cunoscut dintre acestea fiind sRGB. Matricea de transformare a acestuia este [11]:

$$\begin{bmatrix} 0,4125 & 0,3576 & 0,1804 \\ 0,2127 & 0,7152 & 0,0722 \\ 0,0193 & 0,1192 & 0,9503 \end{bmatrix}$$

Prin urmare, cei trei coeficienți utilizați în formula 5 vor avea valorile:  $C_{YR} = 0,2127$ ,  $C_{YG} = 0,7152$ ,  $C_{YB} = 0,0722$ .

### 3. Componente software

**Wire.h** este biblioteca ce conține comenzile pentru magistrala I<sup>2</sup>C.

**volatile** specifică compilatorului să încarce o variabilă din memoria RAM și nu dintr-un registru care este o locație de memorie temporară ce poate fi afectată atunci când se utilizează întreruperi.

**const** are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

**int variabilă = valoare** stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

`unsigned int variabilă = valoare` stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, fără semn (de la 0 până la 65.535).

`unsigned long variabilă = valoare` stabilește o valoare pentru o variabilă de tip întreg pe 32 de biți, fără semn (de la 0 până la 4.294.967.295).

`float variabilă = valoare` stabilește o valoare pentru o variabilă de tip real în virgulă mobilă pe 32 de biți, cu semn (de la  $-3.4028235E+38$  până la  $3.4028235E+38$ ). Numărul total de digiți afișați cu precizie este 6 – 7 (include toți digiții, nu doar cei de după virgulă).

`void setup()` este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

`void loop()` este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

`attachInterrupt(digitalPinToInterrupt(pin), nume_funcție_ISR, mod)` permite utilizarea întreruperilor pentru un pin digital. În cazul plăcii Arduino, pentru utilizarea întreruperilor pot fi folosiți numai pini 2 sau 3. Modurile de activare a întreruperii pot fi: *LOW* (când pinul este în 0 logic), *CHANGE* (când pinul își schimbă valoarea logică), *RISING* (când pinul trece din 0 în 1 logic), *FALLING* (când pinul trece din 1 în 0 logic).

`Serial.begin(viteză)` stabilește rata de transfer a datelor pentru portul serial în biți/secundă (BAUD).

`Serial.print(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial.

`Serial.println(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial, adăugând după datele afișate și trecerea la o linie nouă.

`pinMode(pin, mode)` configurează pinul digital specificat ca intrare sau ca ieșire.

`digitalWrite(pin, valoare)` scrie o valoare HIGH sau LOW pinului digital specificat.

`analogRead(pin)` citește valoarea pinului analogic specificat.

`if(condiție) {instrucțiune/i} else {instrucțiune/instrucțiuni}` testează îndeplinirea sau nu a unei condiții.

`delay(ms)` pune în pauză programul pentru o durată de timp specificată în milisecunde.

`millis()` este o funcție ce returnează ca valoare numărul de milisecunde trecute de când a început executarea secvenței de cod.

`/` este operatorul de împărțire care, în cazul împărțirii a două numere întregi, oferă rezultatul fără rest.

`++` este folosit pentru a incrementa o variabilă

## 4. Aplicația 1. Măsurarea nivelului de iluminare cu un senzor analogic

### 4.1. Realizarea montajului electronic

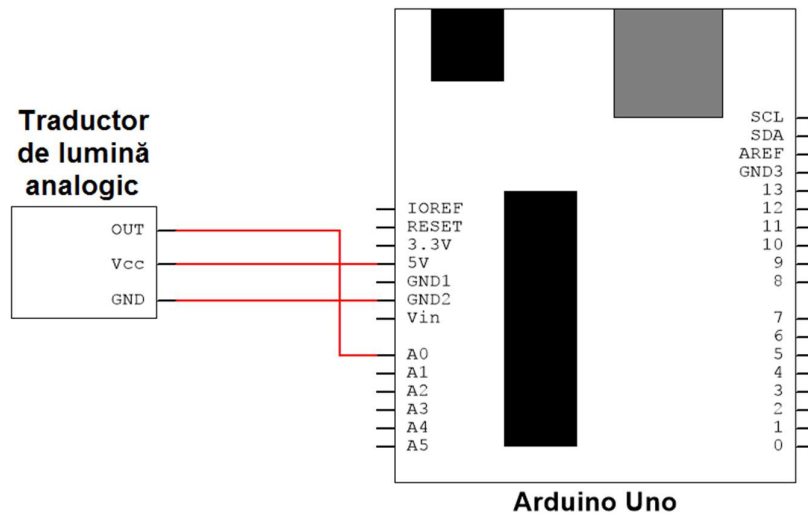
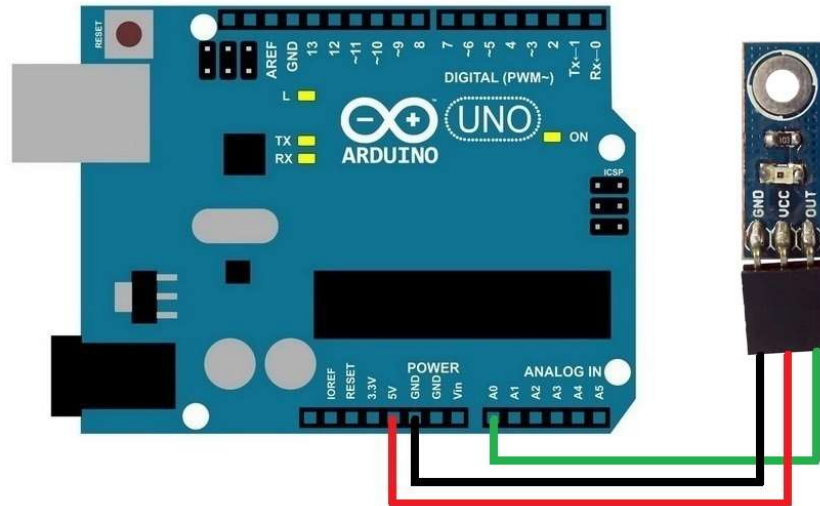


Figura 16. Schema de principiu pentru aplicația 1

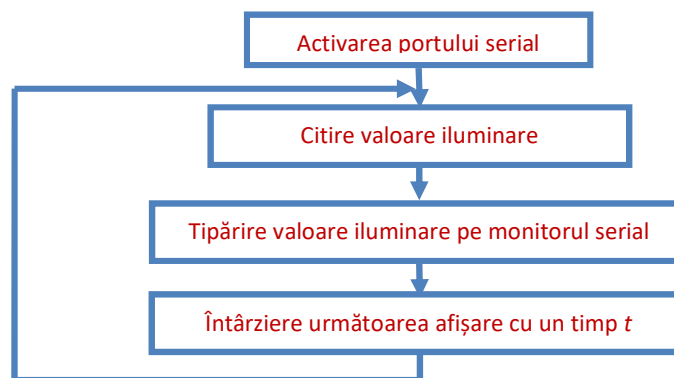


**Figura 17. Realizarea conexiunilor electrice pentru aplicația 1**

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului de lumină analogic;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul VCC al traductorului de lumină analogic;
- Pinul analogic A0 de pe placa Arduino se conectează cu un fir la pinul OUT al traductorului de lumină analogic.

#### 4.2. Schema logică și secvența de cod



```
void setup() {  
  Serial.begin(9600);  
  //activează ieșirea portului serial cu rata de 9600 baud  
}  
  
void loop() {  
  const int valoareIluminare = analogRead(0);  
  //se declară variabila constantă de tip întreg valoareIluminare ce ia  
  //valoarea citită la intrarea analogică A0  
  Serial.print("Valoare iluminare: ");  
  //afișează pe monitorul serial textul din paranteză  
  Serial.println(valoareIluminare, DEC);  
  // afișează pe monitorul serial valoarea citită, codată în sistem zecimal  
  delay(1000);  
  //întârzie următoarea afișare cu 1000 ms  
}
```

## 5. Aplicația 2. Măsurarea nivelului de iluminare cu un senzor digital

### 5.1. Realizarea montajului electronic

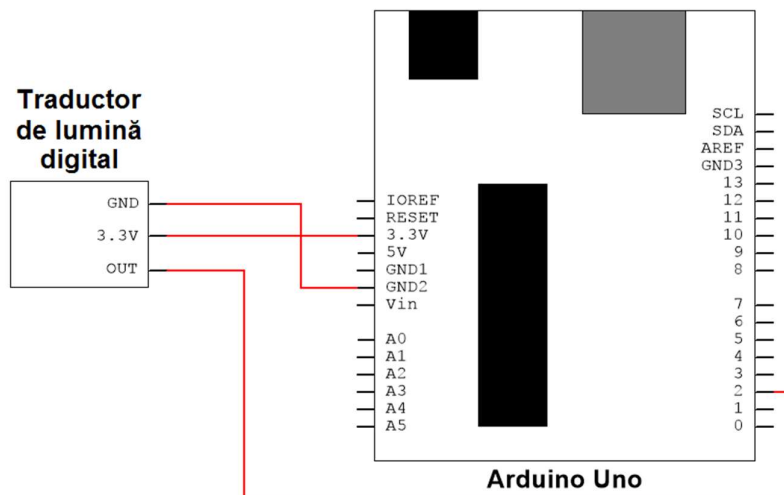
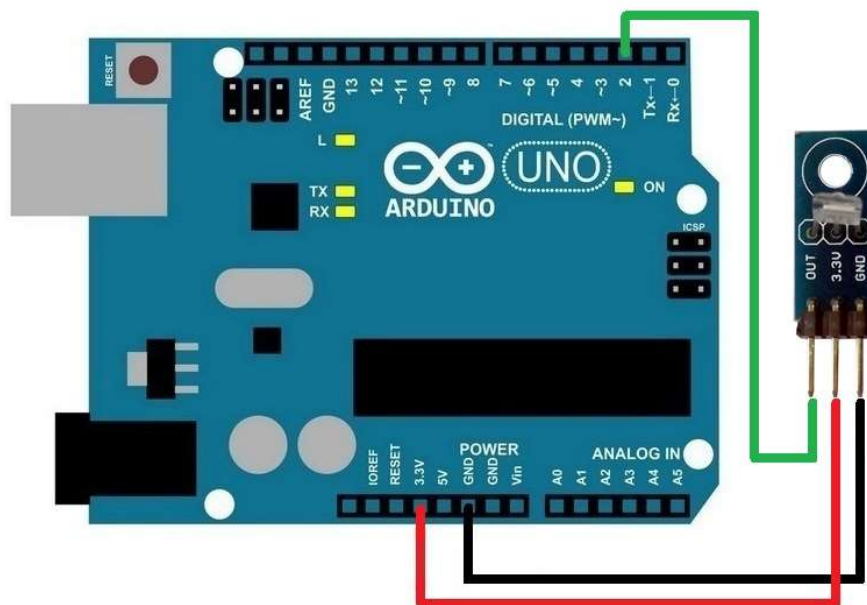


Figura 18. Schema de principiu pentru aplicația 2



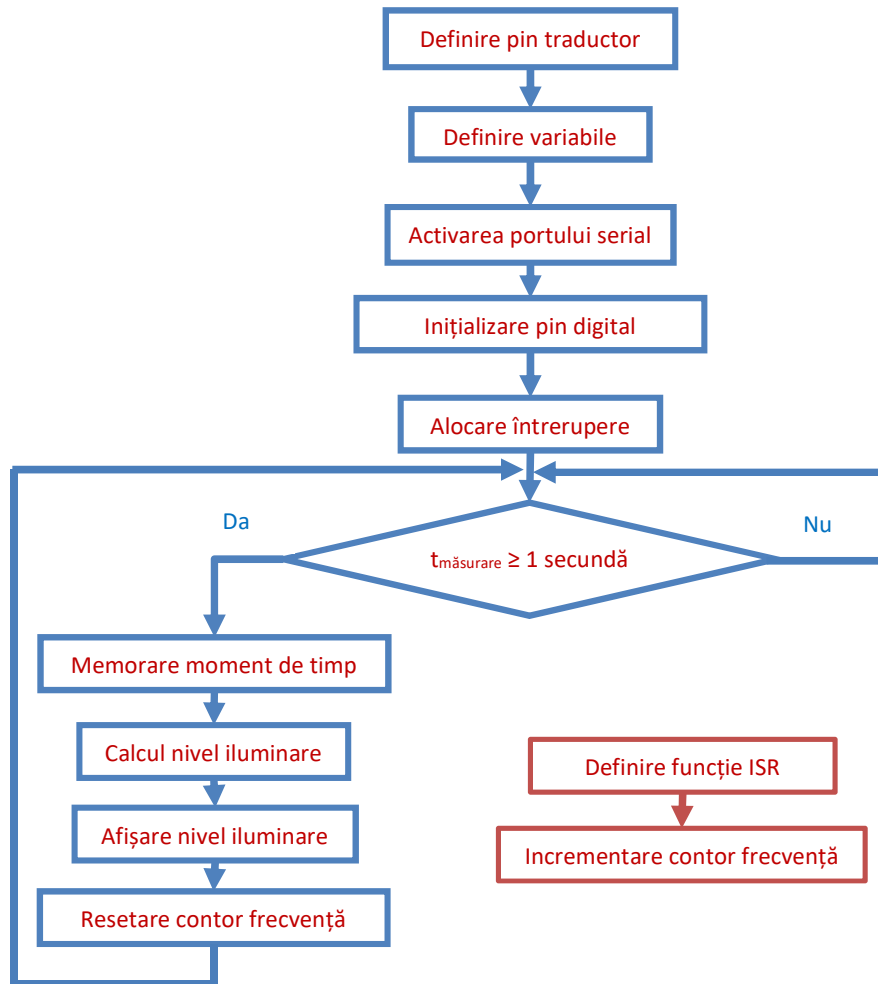
**Figura 19. Realizarea conexiunilor electrice pentru aplicația 2**

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului de lumină digital;
- Pinul 3.3V (power) de pe placa Arduino se conectează cu un fir la pinul 3.3V al traductorului de lumină digital;
- Pinul digital 2 de pe placa Arduino se conectează cu un fir la pinul OUT al traductorului de lumină digital.



5.2. Schema logică și secvența de cod



```

//Autor al programului de bază: Rob Tillaart, 16.05.2011
const int traductor = 2;
    //definirea variabilei traductor corespunzătoare portului digital 2 unde va
    //fi conectată ieșirea OUT a traductorului
volatile unsigned long freqv = 0;
    //definirea variabilei corespunzătoare frecvenței semnalului recepționat
    //de la traductor
unsigned long millis_vechi = 0;
  
```

```
        //definirea variabilei necesare memorării momentului de timp de la care
        //s-a început numărarea impulsurilor
void setup() {
    Serial.begin(115200);
        //activează ieșirea portului serial cu rata de 115200 baud
    pinMode(traductor, INPUT);
        //se declară pinul traductor ca fiind de intrare
    digitalWrite(traductor, HIGH);
        //se scrie valoarea 1 logic pe pinul traductor, pentru a se permite
        //numărarea încă de la primul impuls
    attachInterrupt(digitalPinToInterrupt(traductor), irq, RISING);
        //se alocă o întrerupere pinului traductor și se execută funcția irq atunci
        //când pe acest pin are loc o trecere din 0 în 1 logic
}

void loop() {
    if (millis() - millis_vechi >= 1000)
        //condiție care determină momentul scurgerii unei perioade de 1 secundă
    {
        millis_vechi = millis();
        //memorarea unui nou moment de timp de la care se reîncepe numărarea
        //impulsurilor
        Serial.print("Nivel lumina = ");
        //afișează pe monitorul serial textul din paranteză
        Serial.print(1.72*(frecv/1000));
        //afișează pe monitorul serial valoarea calculată a nivelului de iluminare -
        //formula (3)
        Serial.println(" uW/cm2");
        //afișează pe monitorul serial textul din paranteză
        frecv = 0;
        //resetarea contorului frecv utilizat pentru numărarea impulsurilor
    }
}

void irq() {
    //funcția ISR
    frecv++;
    //incrementarea contorului frecv
}
//Atenție! În colțul din dreapta jos al monitorului serial să fie selectată rata de
115200 baud
```

## 6. Aplicația 3. Măsurarea nivelului de iluminare cu un senzor RGB digital

### 6.1. Realizarea montajului electronic

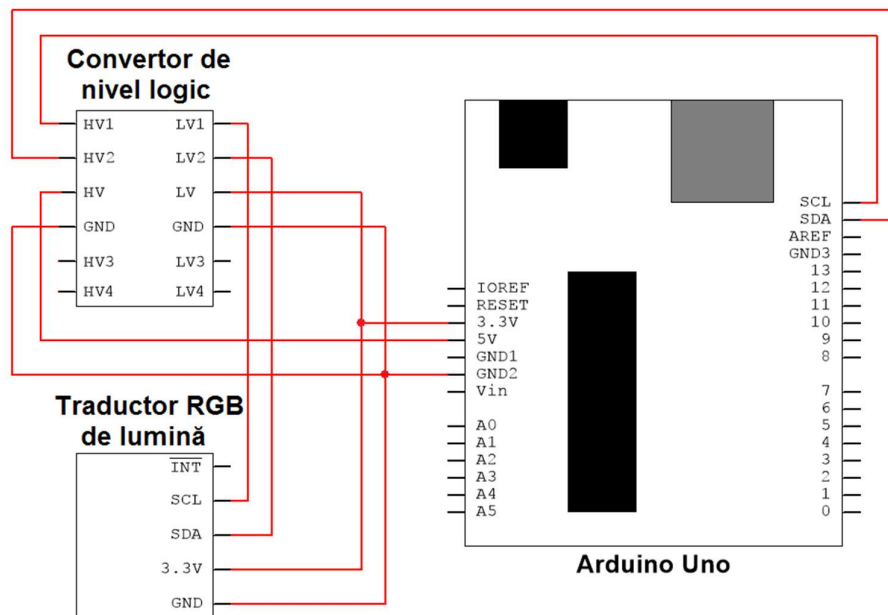


Figura 20. Schema de principiu pentru aplicația 3

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND (LV) al convertorului de nivel logic;
- Pinul GND (HV) al convertorului de nivel logic se conectează cu un fir la pinul GND al traductorului RGB de lumină (este suficientă conectarea în acest mod deoarece pinii GND ai convertorului de nivel logic sunt deja conectați între ei);
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul HV al convertorului de nivel logic;
- Pinul 3.3V (power) de pe placa Arduino se conectează cu un fir la pinul 3.3V al traductorului RGB de lumină;
- Pinul 3.3V al traductorului RGB de lumină se conectează cu un fir la pinul LV al convertorului de nivel logic;

- Pinul SCL al traductorului RGB de lumină se conectează cu un fir la pinul LV1 al convertorului de nivel logic;
- Pinul SDA al traductorului RGB de lumină se conectează cu un fir la pinul LV2 al convertorului de nivel logic;
- Pinul SCL de pe placa Arduino se conectează cu un fir la pinul HV1 al convertorului de nivel logic;
- Pinul SDA de pe placa Arduino se conectează cu un fir la pinul HV2 al convertorului de nivel logic.

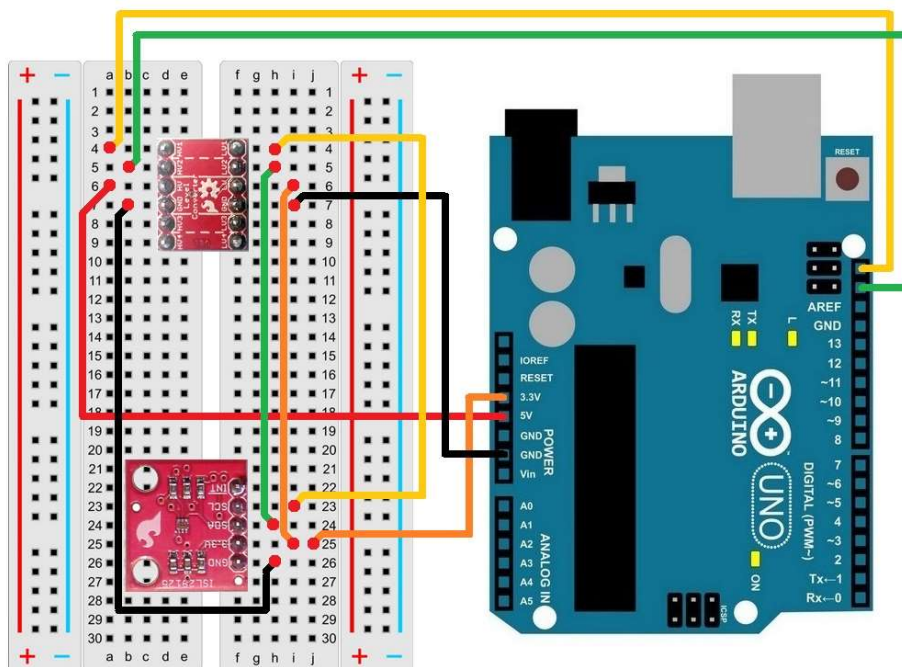
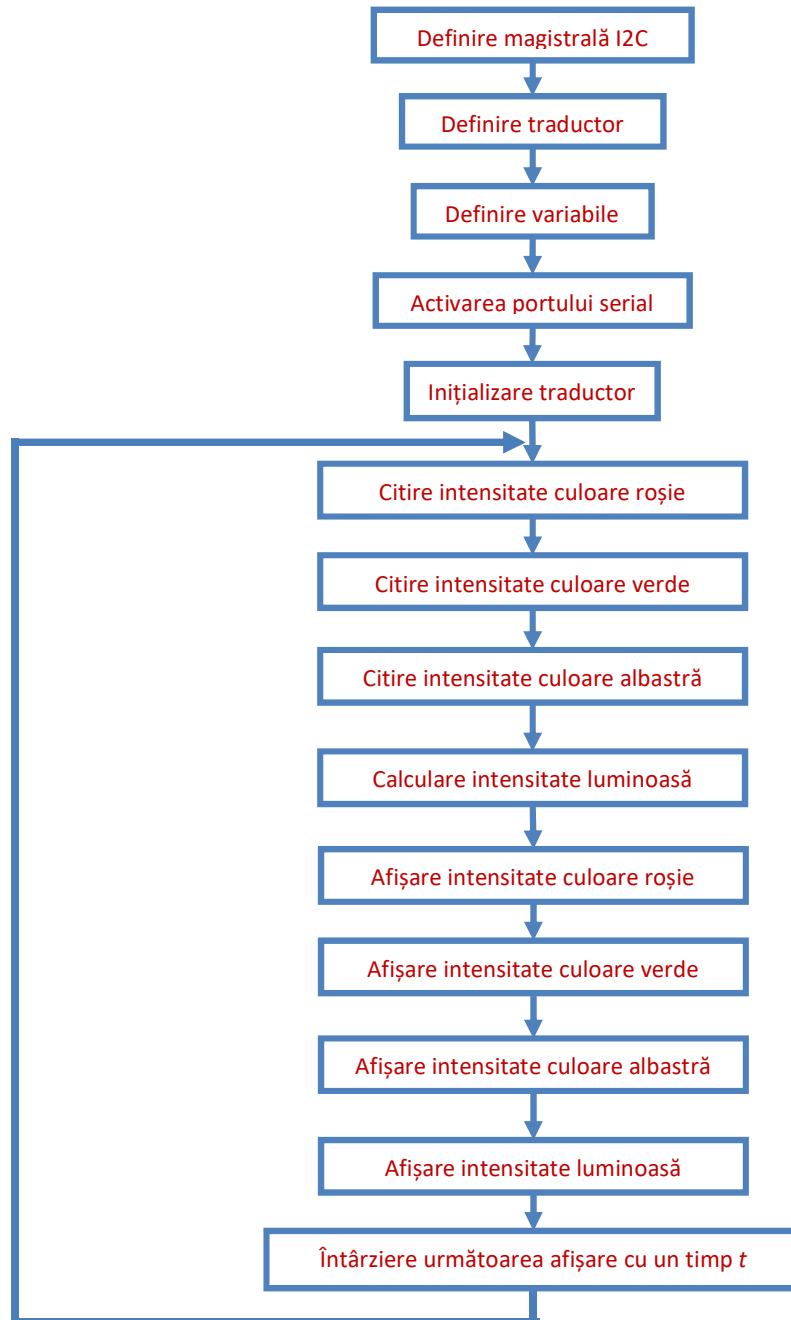


Figura 21. Realizarea conexiunilor electrice pentru aplicația 3

### 6.2. Schema logică și secvența de cod



```
//Autor al programului de bază: Jordan McConnell @ SparkFun Electronics, 11
Apr 2014
#include <Wire.h>
    //includerea în program a bibliotecii comenzilor pentru magistrala I2C
#include "SparkFunISL29125.h"
    //includerea în program a bibliotecii comenzilor pentru traductorul RGB
    digital
SFE_ISL29125 traductor_RGB;
    //definirea traductorului RGB digital ca fiind de tipul ISL29125
float Cyr = 0.2127;
    //definirea variabilei corespunzătoare coeficientului Cyr
float Cyg = 0.7152;
    //definirea variabilei corespunzătoare coeficientului Cyr
float Cyb = 0.0722;
    //definirea variabilei corespunzătoare coeficientului Cyr
float Ev;
    //definirea variabilei corespunzătoare intensității luminoase

void setup() {
    Serial.begin(115200);
        //activează ieșirea portului serial cu rata de 115200 baud
    traductor_RGB.init();
        //inițializare traductor RGB digital
}

void loop() {
    unsigned int rosu = traductor_RGB.readRed();
        //citirea valorii intensității luminii de culoare roșie
    unsigned int verde = traductor_RGB.readGreen();
        //citirea valorii intensității luminii de culoare verde
    unsigned int albastru = traductor_RGB.readBlue();
        //citirea valorii intensității luminii de culoare albastră
    Ev = (Cyr * rosu / 65535 + Cyg * verde / 65535 + Cyb * albastru / 65535) *
10000;
        //calculul valorii intensității luminoase - formula (5)
    Serial.print("Rosu: ");
    Serial.println(rosu / 257);
        //afișarea pe monitorul serial a valorii intensității luminii de culoare roșie
        (domeniul 0 ... 255)
    Serial.print("Verde: ");
    Serial.println(verde / 257);
        //afișarea pe monitorul serial a valorii intensității luminii de culoare verde
        (domeniul 0 ... 255)
    Serial.print("Albastru: ");
```

```
Serial.println(albastru / 257);
    //afișarea pe monitorul serial a valorii intensității luminii de culoare
    albastră (domeniul 0 ... 255)
Serial.print("Intensitatea luminoasa: ");
Serial.print(Ev,0);
Serial.println(" lux");
    //afișarea pe monitorul serial a valorii intensității luminii
Serial.println();
delay(2000);
    //întârzie 2000 ms
}
//Atenție! În colțul din dreapta jos al monitorului serial să fie selectată rata de
115200 baud
```

## 7. Exerciții suplimentare și concluzii

1. Să se modifice secvența de cod pentru aplicația 1 astfel încât să se citească valoarea iluminării de 10 ori într-o secundă și să se afișeze media celor 10 citiri.
2. Să se modifice secvența de cod pentru aplicația 2 astfel încât afișarea intensității luminoase să se facă în  $W/cm^2$ .
3. Să se modifice secvența de cod pentru aplicația 3 astfel încât intensitatea luminoasă pentru fiecare culoare să fie afișată în domeniul 0 ... 1.
4. Să se realizeze un singur montaj electronic care să grupeze cele trei aplicații și să se compare rezultatele obținute de la cele trei traductoare.

Caracteristica de ieșire a traductoarelor este foarte importantă pentru utilizator, aceasta furnizând informații despre dependența dintre mărimea de ieșire și mărimea de intrare. Această caracteristică se dorește a fi liniară și, pe cât posibil, fără fluctuații care țin de parametrii mediului în care funcționează traductorul (de exemplu, temperatura mediului de funcționare). Pentru a obține această liniaritate sunt necesare acțiuni de compensare. Pentru cazurile modificării mărimii de ieșire cu temperatura se fac compensări termice care constau, în majoritatea cazurilor, în adăugarea unor componente care reacționează invers odată cu modificarea temperaturii (față de reacția traductorului necompensat).

Setul de 3 culori de bază RGB (roșu, verde și albastru) este folosit în obținerea luminii colorate în tot spectrul vizibil. Acest lucru se poate face

prin alăturarea a trei surse de lumină care vor emite pe fiecare dintre cele trei culori de bază și vor avea intensități diferite astfel încât, de la o anumită distanță, punctul obținut prin alăturarea celor trei surse de lumină va avea o anumită culoare.

Un alt parametru important al suprafețelor luminate și colorate îl constituie contrastul. Acesta este diferența în luminanță sau culoare ce face ca un obiect sau imaginea lui pe un afișaj se poate distinge față de alte obiecte din același cadru.

## 8. Bibliografie

- [1] **E. F. Schubert**, *Light Emitting Diodes, Second edition*, Cambridge University Press, 2006.
- [2] „*Arduino Playground*,” 10 03 2015.  
<http://playground.arduino.cc/CommonTopics/PullUpDownResistor>.
- [3] **M. McRoberts**, *Beginning Arduino, 2nd Edition*, Apress, 2013.
- [4] **Everlight Electronics Co., Ltd.**, „*PT15-21C/TR8 - Technical Data Sheet*,” 28.04.2003.
- [5] **Texas Advanced Optoelectronic Solutions Inc.**, „*TSL235R - Light-to-frequency converter*,” 2007.
- [6] „*Arduino Reference - attachInterrupt()*,” 21 11 2015.  
<https://www.arduino.cc/en/Reference/AttachInterrupt>.
- [7] **Intersil Americas**, „*ISL29125 - Datasheet*,” 24.01.2014.
- [8] **W. Truchsess**, „*Effects of Varying I2C Pull-Up Resistors*,” 18.12.2010.



- 
- [9] **SparkFun Electronics**, „*Bi-Directional Logic Level Converter Hookup Guide*,” <https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide>. [Accesat 25 11 2015].
- [10] **SparkFun Electronics**, „*ISL29125 RGB Light Sensor Hookup Guide*,” <https://learn.sparkfun.com/tutorials/isl29125-rgb-light-sensor-hookup-guide>. [Accesat 25 11 2015].
- [11] **D. Pascale**, *A review of RGB color spaces*, Montreal, Canada: The BabelColor Company, 2003.



# Lucrarea 7. Măsurarea tensiunii și intensității curentului electric

## 1. Descrierea lucrării

### 1.1. Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează senzori și traductoare.
- Utilizarea de module electronice tip *shield*.
- Realizarea unei aplicații practice de măsurare a valorilor tensiunii și intensității curentului electric (într-un circuit de curent continuu), de calcul al puterii și energiei consumate și afișarea lor pe un ecran LCD.

### 1.2. Descriere teoretică

#### ✓ Introducere

Curentul electric este o mișcare dirijată a sarcinilor electrice ce apare într-un circuit electric format din generator electric, conductori și consumatori. Curentul electric este caracterizat de mai multe mărimi fizice, astfel:

- Tensiunea electrică, notată cu  $U$ , având ca unitate de măsură volt-ul (V), reprezintă diferența de potențial între două puncte ale unui circuit electric și este proporțională cu lucrul mecanic efectuat de către generatorul electric pentru a deplasa o sarcină electrică de la un punct la celălalt.
- Intensitatea curentului electric, notată cu  $I$ , având ca unitate de măsură amper-ul (A), măsoară sarcina electrică ce traversează secțiunea unui conductor în unitatea de timp.
- Puterea electrică, notată cu  $P$ , având ca unitate de măsură watt-ul (W), reprezintă energia furnizată de un generator electric într-o unitate de timp și se calculează cu formula  $P = W/t = U \cdot I$ .

- Energia electrică, notată cu  $E$ , având ca unitate de măsură watt-ora (Wh), măsoară lucrul mecanic necesar pentru a transporta o sarcină electrică  $q$  printr-o secțiune din circuit într-un interval de timp, și se calculează cu formula  $E = U \cdot I \cdot t = P \cdot t$ .

#### ✓ **Descrierea aplicației**

Scopul acestei aplicații este de a realiza un circuit electronic care să măsoare tensiunea aplicată unui consumator (un bec cu incandescență în cazul aplicației prezentate) precum și intensitatea curentului consumat de acesta, să realizeze calculul puterii și energiei consumate și să le afișeze numeric pe un ecran LCD.

Măsurarea tensiunii electrice utilizând placa Arduino Uno se va face prin aplicarea tensiunii de măsurat la una din intrările analogice, deoarece placa conține convertorul analog/digital necesar transformării mărimii fizice analogice în una digitală. Așa cum a mai fost menționat, pentru o variație între 0 V și 5 V la aplicată intrării analogice, placa Arduino furnizează o valoare digitală între 0 și 1023. Atunci când tensiunea analogică poate depăși valoarea de 5 V (cum va fi cazul în această lucrare), situație care poate duce la defectarea microcontrolerului, este necesară utilizarea unui divizor rezistiv de tensiune.

Măsurarea intensității curentului electric utilizând placa Arduino Uno se va face cu ajutorul unui traductor dedicat măsurării acestuia, bazat pe utilizarea unui senzor magnetic Hall.

Efectul Hall (Figura 1) constă în apariția unui câmp electric transversal și a unei diferențe de potențial într-un semiconductor parcurs de un curent electric, atunci când acesta este introdus într-un câmp magnetic, perpendicular pe direcția curentului. Curentul electric ce parcurge materialul semiconductor este influențat de câmpul magnetic, prin urmare tensiunea de la ieșirea senzorului va fi direct proporțională cu intensitatea câmpului magnetic.

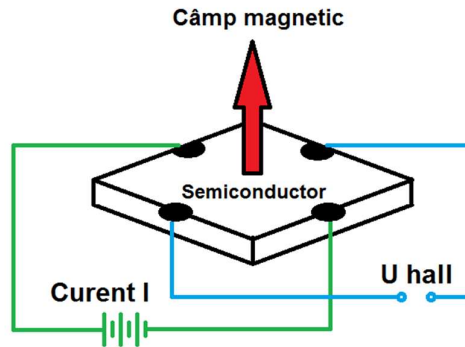
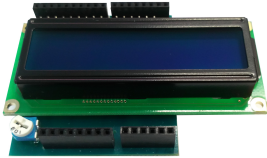








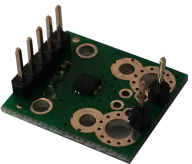
Figura 1. Efectul Hall

Pe suprafața traductorului se află un conductor din cupru ce este străbătut de curentul electric ce se dorește a fi măsurat, acesta generând un câmp magnetic sesizat de senzor și transformat într-o tensiune electrică proporțională (tensiune furnizată la ieșirea traductorului) [1]. Pe baza variației tensiunii de la ieșirea traductorului (între 0 și 5 V), aplicată unuia din porturile analogice, placa Arduino furnizează o valoare digitală ce variază între 0 și 1023.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>LCD Shield</i>	Afișare pe 2 rânduri a câte 16 caractere	1	
<i>Rezistor</i>	10kΩ	1	
<i>Rezistor</i>	1,8kΩ	1	

<i>Rezistor semireglabil</i>	2 k $\Omega$	1	
<i>Rezistor semireglabil</i>	5 k $\Omega$	1	
<i>Bec</i>	12V, 35W	1	
<i>Fir de legătură</i>	Tată-Tată	6	
<i>Fir conexiune</i>	Cu și fără conectori	4	
<i>Traductor de curent</i>	ACS711	1	
<i>Sursă de alimentare reglabilă</i>	0 - 35V, 5A	1	

### ✓ Observații

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 2 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

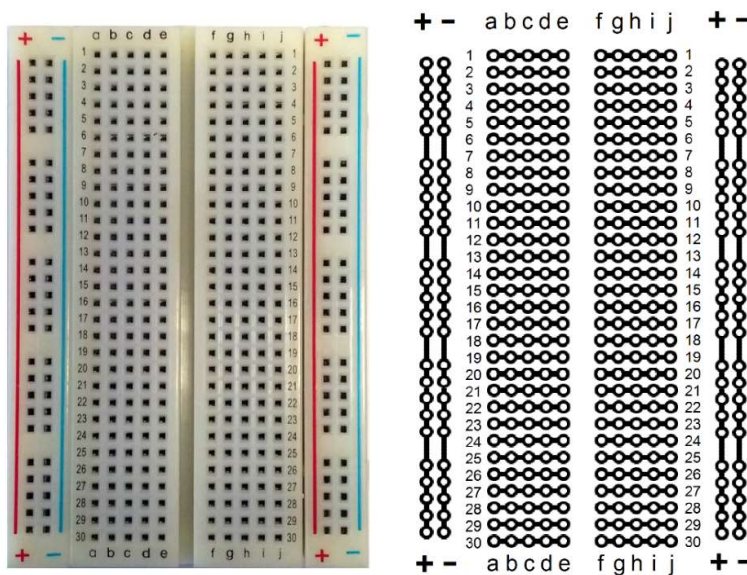


Figura 2. Breadboard-ul și conexiunile interne

**Traductorul de curent** [2] măsoară intensitatea curentului absorbit de consumator, bazându-se pe utilizarea unui senzor magnetic Hall liniar (ACS711EX).



Figura 3. Traductor de curent realizat cu senzorul ACS711EX

Traductorul poate măsura curenți electrici cu intensități între -15,5 A și 15,5 A având o rezistență internă de aproximativ 0,6 mΩ și o izolație electrică pentru tensiuni până la 100 V. Tensiunea de ieșire în repaus (atunci când curentul măsurat este 0 A) este  $V_{cc}/2$ , adică 2,5 V. Această valoare poate varia însă în funcție de valoarea reală a tensiunii  $V_{cc}$  aplicate și de deriva termică a senzorului ( $\pm 5\%$ ), vezi Figura 4.

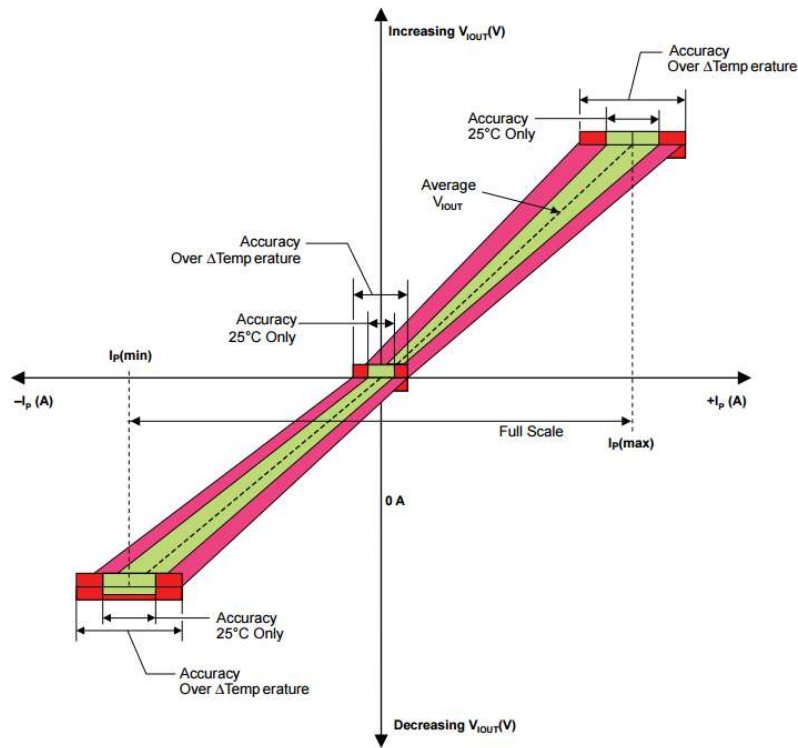


Figura 4. Variația tensiunii de ieșire în funcție de curent pentru senzorul ACS711EX [2]

Schema electronică a traductorului de curent se poate observa în Figura 5.

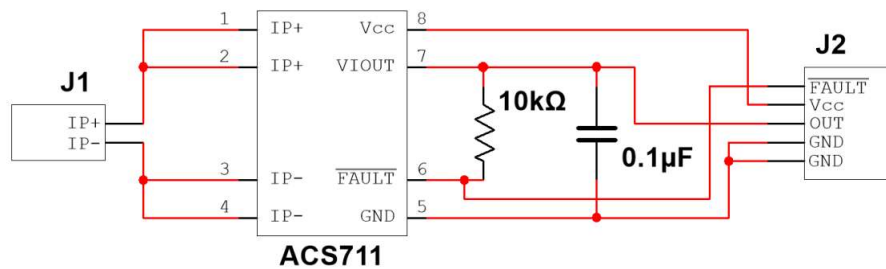


Figura 5. Schema electronică a traductorului de curent

Traductorul se va alimenta cu tensiunea  $V_{CC} = 5$  V.



*Măsurarea intensității curentului cu ajutorul traductorului de curent*

Pentru un curent ce traversează traductorul între -15,5 A și 15,5 A acesta va furniza la ieșire (pe borna OUT) o tensiune între 0 și  $V_{cc}$  (5V). Tensiunea este citită de portul analogic al plăcii Arduino, iar aceasta furnizează o valoare digitală (notată  $val\_dig\_I$ ) în domeniul 0 – 1023, corespunzătoare unui domeniu de intensitate a curentului între -15,5 A și 15,5 A.

Prin urmare, atunci când intensitatea curentului este 0,  $val\_dig\_I$  va avea valoarea 512 (vom nota această valoare cu  $val\_dig\_I0$ ). Astfel, pentru curenți electrici pozitivi, domeniul de valori va fi  $512 \div 1023$  ( $val\_dig\_I0 \div val\_dig\_I_{max}$ ).

Rezultă că rezoluția de măsurare va fi de 15,5 A / 512 valori, adică aproximativ 0,03 A/valoare.

Pentru calculul intensității curentului ne vom raporta la valoarea corespunzătoare unui curent de 1 A,  $val\_dig\_1A$  (pentru a putea ulterior să realizăm o calibrare), astfel:

$$\begin{aligned} val\_dig\_1A &= val\_dig\_I0 + \frac{val\_dig\_I_{max} - val\_dig\_I0 + 1}{I_{max}} \\ &= 512 + \frac{512}{15,5} = 545 \end{aligned} \quad (1)$$

$$I = \frac{(val\_dig\_I - val\_dig\_I0)}{(val\_dig\_1A - val\_dig\_I0)} \quad (2)$$

**Shield-ul LCD** permite afișarea de caractere pe un ecran cu cristale lichide, cu iluminare LED. Acesta se montează peste placa Arduino și are conectorii de așa natură încât pinii plăcii vor fi în continuare accesibili.

Ecranul LCD este format din 2 linii a câte 16 caractere, fiecare caracter fiind compus din 5x8 pixeli. Numerotarea coloanelor (caracterelor) se face de la 0 la 15 (de la stânga la dreapta), iar al rândurilor de la 0 la 1 (de sus în jos).

**Important!** Pentru a funcționa, *shield*-ul utilizează pinii digitali ai plăcii Arduino de la 2 până la 7 astfel: pinul 2 – *d7*, pinul 3 – *d6*, pinul 4 – *d5*, pinul 5 – *d4*, pinul 6 – *enable*, pinul 7 – *rs*.

**Divizorul rezistiv de tensiune** (Figura 6) este utilizat pentru a măsura tensiunea electrică și este necesar deoarece atât tensiunea furnizată de sursa de alimentare de laborator (35 V) cât și tensiunea maximă de funcționare a becului (12 V) depășesc tensiunea maximă suportată la intrările analogice (5 V) ale plăcii Arduino Uno.

Sursa de alimentare va fi reglată astfel încât să nu poată furniza mai mult de 12 V (prin limitarea intensității curentului furnizat la aproximativ 3 A, necesară aprinderii becului la intensitate maximă). Cu toate acestea, pentru protecția plăcii Arduino la dereglări accidentale ale sursei de alimentare, se va calcula divizorul rezistiv astfel încât va furniza la ieșire o tensiune de maxim 5 V ( $V_{cc}$ ) pentru o tensiune aplicată la bornele de intrare de 35 V (chiar dacă prin aceasta se va micșora precizia domeniului de măsurare).

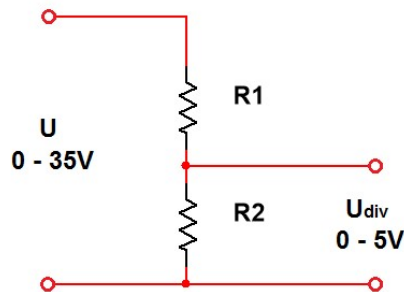


Figura 6. Schema divizorului rezistiv de tensiune

Curentul absorbit de intrarea analogică este neglijabil, prin urmare putem considera că intensitățile curentilor prin  $R_1$  și  $R_2$  sunt egale. Se alege o valoare mică, de ordinul mA. Pentru calculele următoare am ales  $I_{div} = 3$  mA.

$$R_1 + R_2 = \frac{U}{I_{div}} = \frac{35 \text{ V}}{3 \cdot 10^{-3} \text{ A}} = 11,67 \text{ k}\Omega \quad (3)$$

Din formulele divizorului rezistiv de tensiune rezultă:

$$U_{div-max} = V_{cc} = U \frac{R_2}{R_1 + R_2} \Rightarrow R_2 = \frac{5V \cdot 11,67 k\Omega}{35V} = 1,67 k\Omega \quad (4)$$

$$\Rightarrow R_1 = 10 k\Omega$$

Se aleg rezistoarele standard  $R_1 = 10 k\Omega$  și  $R_2 = 1,8 k\Omega$ , cu toleranța de 5%.

Datorită domeniului de toleranță, valorile reale ale rezistoarelor pot varia astfel:

- $R_1$  între  $9,5 k\Omega$  și  $10,5 k\Omega$
- $R_2$  între  $1,71 k\Omega$  și  $1,89 k\Omega$

Pentru a ne asigura că divizorul nu va furniza pe rezistența  $R_2$  o tensiune mai mare decât  $V_{cc}$  refacem calculul rezistenței  $R_1$ , pentru cea mai mare valoare a lui  $R_2$ , adică  $1,89 k\Omega$ .

$$U_{div-max} = V_{cc} = U \frac{R_2}{R_1 + R_2} \Rightarrow R_1 = \frac{(35V \cdot 1,89k\Omega) - (5V \cdot 1,89k\Omega)}{5V} = 11,34 k\Omega \quad (5)$$

Dacă considerăm situația în care  $R_1$  ales are valoarea cea mai mică, adică  $9,5 k\Omega$ , se observă că este necesară o rezistență suplimentară care să compenseze diferența de  $11,34 k\Omega - 9,5 k\Omega = 1,84 k\Omega$ .

Prin urmare se alege soluția înserierii cu rezistorul  $R_1$  a unui rezistor semireglabil  $R$  cu valoarea standard de  $2 k\Omega$ , ce va fi utilizat pentru calibrare.

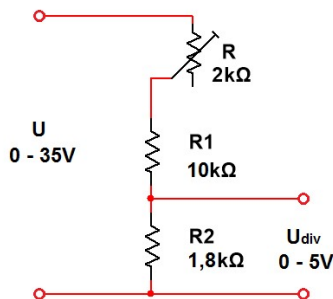


Figura 7. Schema finală a divizorului rezistiv de tensiune

*Măsurarea tensiunii cu ajutorul divizorului rezistiv de tensiune*

Pentru o tensiune aplicată divizorului între 0 și 35 V (sursa de alimentare) acesta va furniza la ieșire (pe bornele rezistorului  $R_2$ ) o tensiune între 0 și  $V_{cc}$  (5 V). Această tensiune este citită de portul analogic al plăcii Arduino care furnizează o valoare (notată  $val\_dig\_U$ ) în domeniul 0 – 1023.

Astfel, rezoluția de măsurare va fi de  $35V / 1024$  valori = 34,18 mV/valoare.

Tensiunea furnizată de divizorul rezistiv se determină astfel:

$$\frac{V_{cc}}{1023} = \frac{U_{div}}{val\_dig\_U} \Rightarrow U_{div} = \frac{V_{cc} \cdot val\_dig\_U}{1023} \quad (6)$$

Tensiunea aplicată divizorului (care se dorește a fi măsurată) se poate determina și ea, astfel:

$$\frac{U}{U_{div}} = \frac{35}{V_{cc}} \Rightarrow U = U_{div} \frac{35}{V_{cc}} \quad (7)$$

*Mărirea rezoluției de măsurare a tensiunii*

Fiecare port de intrare analogic utilizează un convertor analogic-digital pentru a transforma tensiunea analogică citită într-o valoare digitală. Rezoluția convertorului este funcție de numărul de biți utilizați pentru descrierea valorii digitale. Astfel, placa Arduino utilizează 10 biți, deci pentru o tensiune de intrare între 0 – 5 V va furniza 1024 de valori digitale (între 0 – 1023), rezoluția fiind de aproximativ  $5 V / 1024 = 4,88$  mV/valoare.

Pentru a determina valoarea tensiunii analogice de intrare, aceasta este comparată de către convertor cu o tensiune de referință. Valoarea tensiunii de referință este egală cu  $V_{cc}$ , adică 5 V.

În cazul aplicației prezentate, tensiunea maximă ce poate fi măsurată este de 35 V (adică 5 V la ieșirea divizorului de tensiune), dar tensiunea maximă efectivă măsurată nu poate depăși 12 V datorită consumatorului (adică 1,71 V la ieșirea divizorului de tensiune).

În loc să utilizăm tot domeniul de valori digitale 0 – 1023 pentru tensiuni între 0 – 5 V, putem să-l utilizăm pentru a descrie domeniul

0 – 1,71 V. Aceasta se poate realiza furnizarea unei tensiuni de referință din exterior, care să aibă valoarea de 1,71 V. Astfel, rezoluția de măsurare devine  $1,71 \text{ V} / 1024 = 1,67 \text{ mV/valoare}$ .

O tensiune de referință externă se poate aplica plăcii Arduino pe pinul AREF (*Analog REFerence*). Tensiunea poate să provină dintr-o sursă de alimentare externă sau de la o sursă internă (3,3 V sau 5 V) folosind un divizor rezistiv de tensiune pentru a o micșora.

**Atenție!** A nu se utiliza tensiuni de referință externe mai mici de 0 V sau mai mari de 5 V pe pinul AREF!

În cazul aplicației prezentate se va utiliza un rezistor semireglabil (ce poate fi definit ca un divizor rezistiv reglabil) în locul a două rezistoare fixe, pentru a stabili cu precizie cu ajutorul unui aparat de măsură valoarea tensiunii de referință, așa cum se observă în Figura 10.

### 3. Componente software

`LiquidCrystal.h` este biblioteca ce conține comenzile pentru *shield*-ul LCD.

`LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)` creează o variabilă `lcd` specificându-se pinii digitali folosiți pentru a comanda *shield*-ul LCD.

`int variabilă = valoare` stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

`const` are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

`unsigned long variabilă = valoare` stabilește o valoare pentru o variabilă de tip întreg pe 32 de biți, fără semn (de la 0 până la 4.294.967.295).

`float variabilă = valoare` stabilește o valoare pentru o variabilă de tip real în virgulă mobilă pe 32 de biți, cu semn (de la -3.4028235E+38 până la 3.4028235E+38). Numărul total de digiți afișați cu precizie este 6 – 7 (include toți digiții, nu doar cei de după virgulă).

`void setup()` este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

`void loop()` este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

`pinMode(pin, mode)` configurează pinul digital specificat ca intrare sau ca ieșire.

`lcd.begin(coloane, rânduri)` inițializează interfața cu ecranul LCD și specifică numărul de rânduri și de coloane al acestuia.

`Serial.begin(viteză)` stabilește rata de transfer a datelor pentru portul serial în biți/secundă (BAUD).

`if(condiție) {instrucțiune/i} else {instrucțiune/instrucțiuni}` testează îndeplinirea sau nu a unei condiții.

`for(inițializare, condiție, increment) {instrucțiune/ instrucțiuni }` repetă un bloc de instrucțiuni până la îndeplinirea condiției.

`analogRead(pin)` citește valoarea pinului analogic specificat.

`delay(ms)` pune în pauză programul pentru o durată de timp specificată în milisecunde.

`millis()` este o funcție ce returnează ca valoare numărul de milisecunde trecute de când a început executarea secvenței de cod.

`Serial.println(valoare sau variabilă, sistem de numerație)` tipărește date sub formă de caractere ASCII folosind portul serial, adăugând după datele afișate și trecerea la o linie nouă.

`lcd.setCursor(coloană, rând)` stabilește poziția cursorului LCD. Pentru LCD-ul folosit în această aplicație numărul coloanelor este de la 0 la 15, iar al rândurilor de la 0 la 1.

`lcd.clear()` șterge ecranul LCD și poziționează cursorul în colțul din stânga sus.

`lcd.print()` afișează pe ecranul LCD datele (valori ale unor variabile)/textul dintre paranteze. Pentru a afișa un text este necesar ca acesta să fie plasat între ghilimele ("text"). Pentru a afișa valoarea unei variabile de tip *char*, *byte*, *int*, *long*, sau *string* se scrie numele variabilei și, opțional, baza de numerație a acesteia (*variabilă*, BIN sau DEC sau OCT sau HEX). Pentru a afișa valoarea unei variabile de tip *float* sau *double* se scrie numele variabilei iar după virgulă, numărul de zecimale dorit a se afișa (*variabilă*, nr. zecimale).

`analogReference(EXTERNAL)` stabilește tensiunea de referință folosită pentru intrările analogice ca fiind cea furnizată de o sursă din exterior.

`++` este folosit pentru a incrementa o variabilă

## 4. Aplicația 1. Măsurarea tensiunii și a intensității curentului

### 4.1. Realizarea montajului electronic

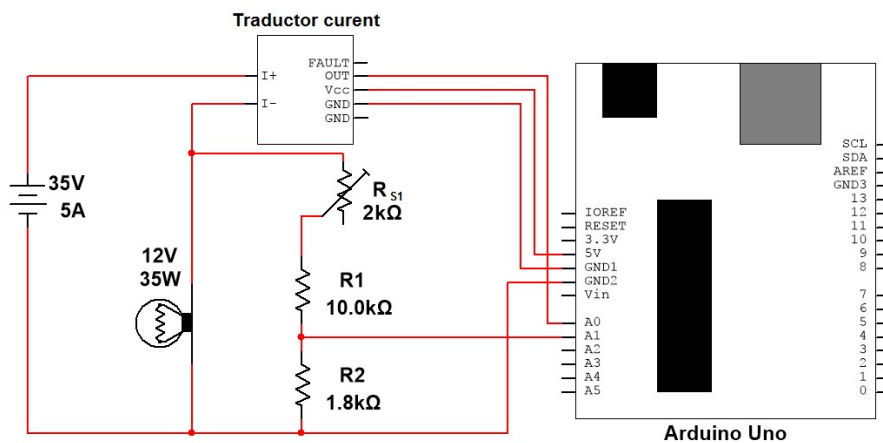
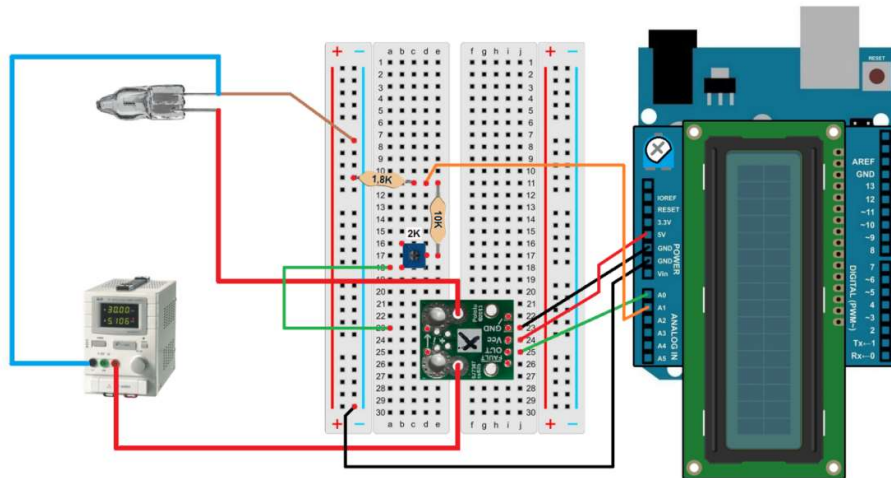


Figura 8. Schema de principiu pentru aplicația 1



**Figura 9. Realizarea conexiunilor electrice pentru aplicația 1**

Se realizează următoarele conexiuni:

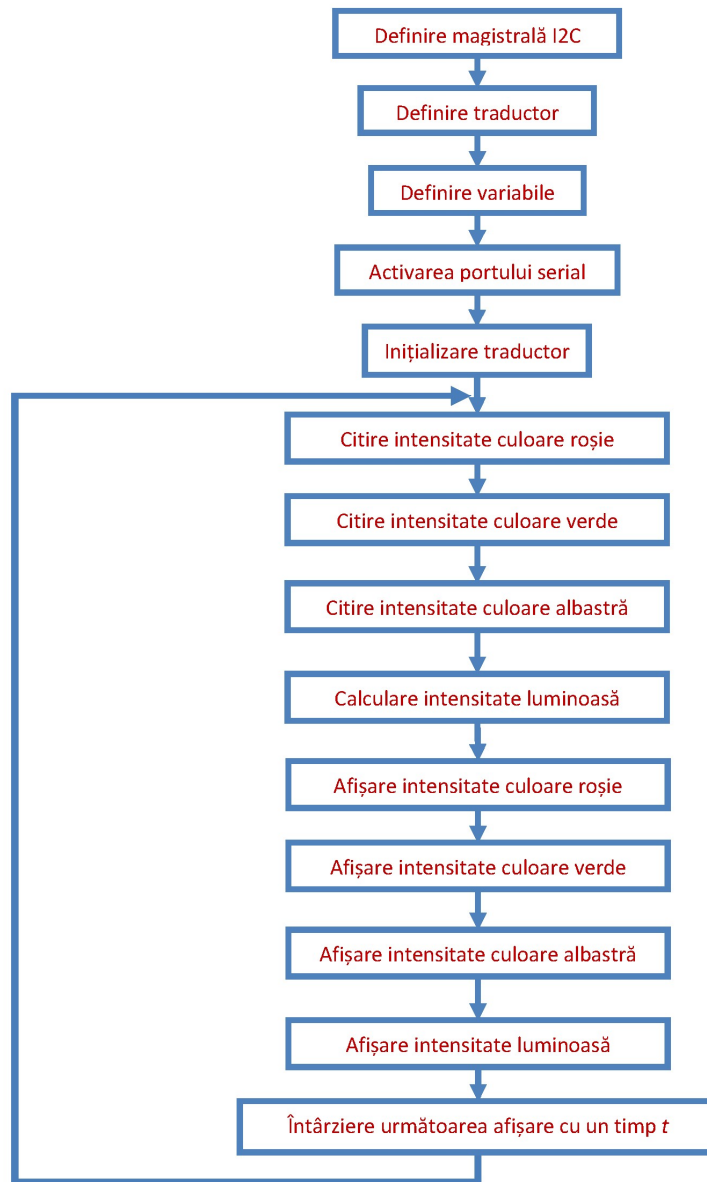
- Se amplasează traductorul de curent pe *breadboard* conectându-se pinii de intrare pentru curentul ce trebuie măsurat pe coloana *d* și pinii de ieșire și alimentare pe coloana *i*;
- Se amplasează rezistoarele pe *breadboard* conform schemei de principiu;
- Pinul de minus al becului se conectează cu un fir la bara de "-" a *breadboard*-ului;
- Pinul analogic A1 de pe placa Arduino se conectează cu un fir la rezistoarele R1 și R2;
- Pinul analogic A0 de pe placa Arduino se conectează cu un fir la pinul OUT al traductorului de curent;
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al traductorului de curent;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul Vcc al traductorului de curent;
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la bara de "-" a *breadboard*-ului.

**Atenție!** A se verifica conectarea corectă și sigură a rezistorului de  $1,8\text{ K}\Omega$  și a firului (maro în schema de mai sus) dintre minusul becului și bara de "-" a *breadboard*-ului. Orice eroare în acest sens poate duce la



apariția pe pinul A1 a unei tensiuni mai mari de 5 V și poate deteriora microcontrolerul.

#### 4.2. Schema logică și secvența de cod



```
#include <LiquidCrystal.h>
//includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
//inițializarea bibliotecii și a variabilei lcd cu numerele pinilor utilizați de
shield-ul LCD
const int intrareCurent = 0;
//definirea variabilei intrareCurent corespunzătoare portului analogic A0
unde va fi conectat pinul OUT al traductorului de curent
const int intrareTensiune = 1;
//definirea variabilei intrareTensiune corespunzătoare portului analogic
A1 unde va fi conectată ieșirea divizorului rezistiv
float Udiv = 0.0;
//definirea tensiunii furnizate de divizorul rezistiv
float I = 0.0;
//definirea variabilei intensitate electrică
float U = 0.0;
//definirea variabilei tensiune electrică
float E = 0.0;
//definirea variabilei energie electrică
float Etot = 0.0;
//definirea variabilei energie electrică totală
float P = 0.0;
//definirea variabilei putere electrică
unsigned long val_dig_I = 0;
//definirea variabilei val_dig_I ce va conține valoarea citită de pe portul
analogic
unsigned long val_dig_U = 0;
//definirea variabilei val_dig_U ce va conține valoarea citită de pe portul
analogic
unsigned long timp = 0;
//definirea variabilei timp cu valoarea inițială 0
float Vcc = 5.0;
//definirea variabilei Vcc ce va avea valoarea inițială 5V
float Uref = 5.0;
//definirea variabilei Vref ce va avea valoarea inițială 5V
float val_dig_I0 = 512;
//definirea variabilei val_dig_I0 ce corespunde inițial valorii mediane
furnizate de traductorul de curent
const int val_dig_1A = 545;
//definirea variabilei val_dig_1A ce corespunde inițial valorii furnizate de
traductorul de curent atunci când este parcurs de un curent de 1A

void setup(){
  lcd.begin(16, 2);
```

```

        //inițializarea interfeței cu ecranul LCD și specificarea numărului de
        rânduri și de coloane al acestuia
    Serial.begin(9600);
        //activează ieșirea portului serial cu rata de 9600 baud
    }

    void loop(){
        val_dig_U = analogRead(intrareTensiune);
            //variabila val_dig_U ia valoarea citită de pe portul analogic 1
        /*for (int i=0;i<500;i++) {
            val_dig_U = val_dig_U + analogRead(intrareTensiune);
            val_dig_I = val_dig_I + analogRead(intrareCurent);
            delay(1);
        }
        val_dig_U = val_dig_U / 500;
        val_dig_I = val_dig_I / 500;*/
        Udiv = (val_dig_U * Uref) / 1023.0;
            //se calculează tensiunea culeasă de la bornele divizorului de tensiune, în
            funcție de valoarea val_dig_U citită pe portul analogic 1 – formula (6)
        U = Udiv * (35 / Vcc);
            //se calculează tensiunea aplicată la bornele divizorului de tensiune
            (tensiunea aplicată becului) – formula (7)
        val_dig_I = analogRead(intrareCurent);
            //variabila val_I ia valoarea citită de pe portul analogic 0
        I = (val_dig_I - val_dig_I0) / (val_dig_1A - val_dig_I0);
            //se calculează intensitatea în A, în funcție de valoarea val_dig_1A
            corespunzătoare unui curent de 1A – formula (2)
        P = U * I;
            //se calculează puterea în W
        E = (P * (millis() - timp)) / 3600000;
            //se calculează energia consumată pe durata ultimei bucle loop
        timp = millis();
            //se modifică referința de timp
        Etot = Etot + E;
            //se calculează energia totală consumată
        Serial.println(val_dig_I);
            //tipărește pe monitorul serial valoarea citită de pe portul analogic 0
            (necesară pentru calibrare)
        lcd.clear();
            //șterge ecranul LCD și poziționează cursorul în colțul din stânga sus
        lcd.print("U=");
            //afișează pe ecranul LCD textul dintre ghilimele
        lcd.print(U,1);
    }

```

```
        //afișează pe ecranul LCD valoarea variabilei U, cu o zecimală după
        virgulă
    lcd.print("V");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(" I=");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(I,2);
        //afișează pe ecranul LCD valoarea variabilei I, cu 2 zecimale după
        virgulă
    lcd.print("A");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.setCursor(0,1);
        //mută cursorul pe coloana 1, rândul 2
    lcd.print("P=");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(P,1);
        //afișează pe ecranul LCD valoarea variabilei P, cu o zecimală după
        virgulă
    lcd.print("W");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(" E=");
        //afișează pe ecranul LCD textul dintre ghilimele
    lcd.print(Etot,2);
        //afișează pe ecranul LCD valoarea variabilei Etot, cu 2 zecimale după
        virgulă
    lcd.print("Wh");
        //afișează pe ecranul LCD textul dintre ghilimele
    delay(500);
        //întârzie 500ms
}
```

### 4.3. Mod de lucru și calibrare

#### *Pasul 1*

Se scrie secvența de cod. Se realizează conexiunile electrice conform desenului din Figura 13 și se încarcă secvența de cod.

Deoarece valorile reale diferă de cele teoretice, este necesară o calibrare a circuitului electronic de măsurare realizat, atât prin modificări în partea de software cât și în partea de hardware:

- Tensiunile  $V_{cc}$  și Uref au valoarea teoretică de 5 V. Se va măsura tensiunea reală cu ajutorul unui voltmetru, iar valoarea măsurată se va scrie în program la declararea variabilelor  $V_{cc}$  și Uref.
- val\_dig\_I0 (valoarea digitală corespunzătoare valorii 0 a intensității curentului măsurat) are valoarea teoretică 512. Cu sursa de alimentare oprită (curent 0 prin consumator) se afișează variabila val\_dig\_I pe monitorul serial, iar valoarea afișată se va scrie în program la declararea variabilei val\_I0.
- val\_dig\_1A (valoarea digitală corespunzătoare valorii de 1A a intensității curentului măsurat) are valoarea teoretică 545. Cu sursa de alimentare pornită, se mărește treptat tensiunea de alimentare până când valoarea intensității curentului prin consumator este de 1 A. Se afișează variabila val\_dig\_I pe monitorul serial, iar valoarea afișată se va scrie în program la declararea variabilei val\_dig\_1A.
- Se rotește rezistorul semireglabil  $R_{S1}$  până când tensiunea afișată pe LCD corespunde cu tensiunea afișată de sursa de alimentare.

### Pasul 2

Se elimină caracterele /\* și \*/ care marchează liniile următoare ca fiind comentarii:

```
for (int i=0;i<500;i++) {  
    val_dig_U = val_dig_U + analogRead(intrareTensiune);  
    val_dig_I = val_dig_I + analogRead(intrareCurent);  
    delay(1);  
}  
val_dig_U = val_dig_U / 500;  
val_dig_I = val_dig_I / 500;
```

și se marchează ca fiind comentariu următoarele linii:

```
//val_dig_U = analogRead(intrareTensiune);  
//val_dig_I = analogRead(intrareCurent);  
//delay(500);
```

## 5. Aplicația 2. Utilizarea unei tensiuni de referință externe

### 5.1. Realizarea montajului electronic

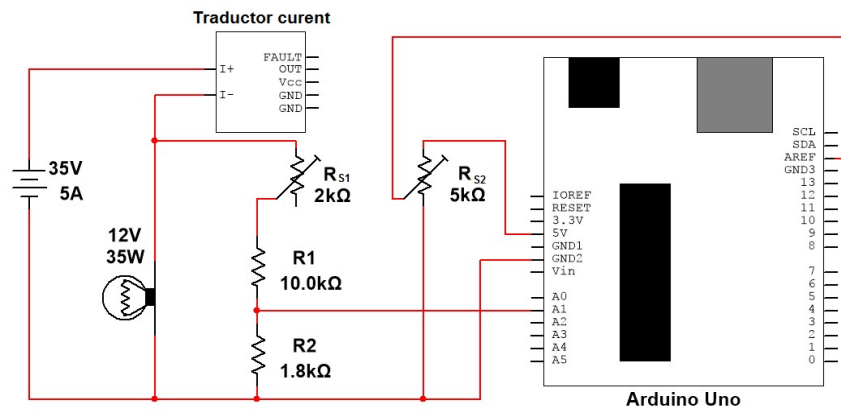


Figura 10. Schema de principiu pentru aplicația 2

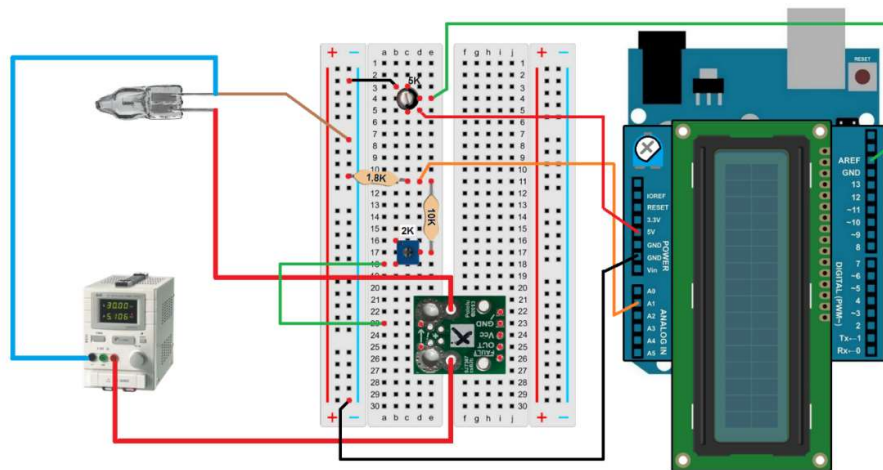


Figura 11. Realizarea conexiunilor electrice pentru aplicația 2

Se realizează următoarele conexiuni:

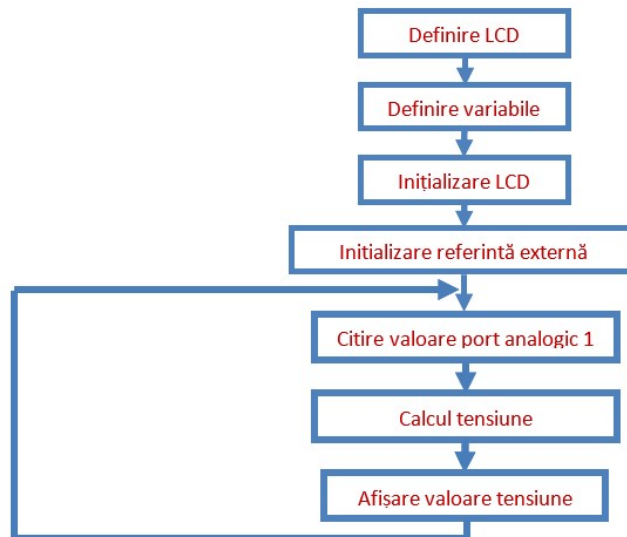
- Se amplasează traductorul de curent pe *breadboard* conectându-se pinii de intrare pentru curentul ce trebuie măsurat pe coloana *d* și pinii de ieșire și alimentare pe coloana *i*;

- Se amplasează rezistoarele pe *breadboard* conform schemei de principiu;
- Pinul de minus al becului se conectează cu un fir la bara de "-" a *breadboard*-ului;
- Pinul analogic A1 de pe placa Arduino se conectează cu un fir la rezistoarele R1 și R2;
- Pinul AREF de pe placa Arduino se conectează cu un fir la cursorul rezistorului semireglabil de 5 K $\Omega$ . Ceilalți doi pini ai acestuia se conectează cu câte un fir la GND, respectiv la pinul 5V de pe placa Arduino.
- Pinul GND (power) de pe placa Arduino se conectează cu un fir la bara de "-" a *breadboard*-ului.

**De reținut!** În cadrul acestei aplicații nu se mai poate utiliza senzorul de curent deoarece acesta necesită o tensiune de referință egală cu  $V_{cc}$ .

**Atenție!** A se verifica conectarea corectă și sigură a rezistorului de 1,8 K $\Omega$  și a firului (maro în schema de mai sus) dintre minusul becului și bara de "-" a *breadboard*-ului. Orice eroare în acest sens poate duce la apariția pe pinul A1 a unei tensiuni mai mari de 5 V și poate deteriora microcontrolerul.

## 5.2. Schema logică și secvența de cod



```
#include <LiquidCrystal.h>
//includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
//inițializarea bibliotecii și a variabilei lcd cu numele pinilor utilizați de
shield-ul LCD
const int intrareTensiune = 1;
//definirea variabilei intrareTensiune corespunzătoare portului analogic
A1 unde va fi conectată ieșirea divizorului rezistiv
float Udiv = 0.0;
//definirea tensiunii furnizate de divizorul rezistiv
float U = 0.0;
//definirea variabilei tensiune electrică
unsigned long val_dig_U = 0;
//definirea variabilei val_dig_U ce va conține valoarea citită de pe portul
analogic
float Vcc = 5.0;
//definirea variabilei Vcc ce va avea valoarea inițială 5V
float Uref = 1.71;
//definirea variabilei Vref ce va avea valoarea inițială 5V

void setup(){
  lcd.begin(16, 2);
  // inițializarea interfeței cu ecranul LCD și specificarea numărului de
rânduri și de coloane al acestuia
  analogReference(EXTERNAL);
  //folosirea tensiunii de referință externe
}

void loop(){
  for (int i=0;i<500;i++) {
    val_dig_U = val_dig_U + analogRead(intrareTensiune);
    delay(1);
  }
  val_dig_U = val_dig_U / 500;
  //se calculează valoarea medie val_dig_U citită pe portul analogic 1
  Udiv = (val_dig_U * Uref) / 1023.0;
  //se calculează tensiunea culeasă de la bornele divizorului de tensiune, în
funcție de valoarea val_dig_U
  U = Udiv * (35 / Vcc);
  //se calculează tensiunea aplicată la bornele divizorului de tensiune
(tensiunea aplicată becului)
  lcd.clear();
  //șterge ecranul LCD și poziționează cursorul în colțul din stânga sus
  lcd.print("U=");
```



```
    //afișează pe ecranul LCD textul dintre ghilimele  
    lcd.print(U,1);  
    //afișează pe ecranul LCD valoarea variabilei U, cu o zecimală după  
    virgulă  
    lcd.print("V");  
    //afișează pe ecranul LCD textul dintre ghilimele  
}
```

### 5.3. Mod de lucru și calibrare

Se scrie secvența de cod. Se realizează conexiunile electrice conform desenului din Figura 11 și se încarcă secvența de cod.

Deoarece valorile reale diferă de cele teoretice, este necesară o calibrare a circuitului electronic de măsurare realizat, atât prin modificări în partea software cât și în partea de hardware:

- Tensiunea  $V_{cc}$  are valoarea teoretică de 5 V. Se va măsura tensiunea reală cu ajutorul unui voltmetru, iar valoarea măsurată se va scrie în program la declararea variabilei  $V_{cc}$ .
- Cu ajutorul unui voltmetru se măsoară tensiunea pe pinul AREF. Se rotește rezistorul semireglabil  $R_{S2}$  până când aceasta are valoarea de 1,71 V.

**Atenție!** Dacă se utilizează o referință externă pe pinul AREF, trebuie obligatoriu setată în secvența de cod referința analogică la EXTERN (comanda `analogReference(EXTERNAL)`) înainte de a apela `analogRead()`. În caz contrar, se va scurtcircuita tensiunea de referință activă (generată intern) și PIN-ul AREF, fiind posibilă deteriorarea microcontrolerului de pe placa Arduino.

## 6. Exerciții suplimentare și concluzii

1. Se observă că valoarea intensității curentului măsurat variază puțin atunci când acesta are valoarea 0 (`val_dig_I0`), datorită derivei termice a senzorului. Să se modifice secvența de cod astfel încât pentru domeniul (`val_dig_I0 - 2`, `val_dig_I0 + 2`) să se afișeze valoarea 0 pentru intensitatea curentului.
2. Care este efectul modificărilor secvenței de cod de la pasul 2?

3. Să se modifice secvența de cod astfel încât afișarea tensiunii măsurate să se facă cu 4 zecimale.

Calibrarea unor aparate sau instrumente de măsură care sunt conectate la un calculator sau placă de dezvoltare se poate face hardware sau software. Calibrarea se face utilizând aparate și instrumente de măsură etalon (sau considerate de către utilizator ca fiind etalon). Pentru calibrarea hardware este necesară modificarea parametrilor care definesc unele componente electronice care fac parte din aparatul sau instrumentul de măsură. Calibrarea software se face prin modificarea valorilor de conversie din programele care prelucrează datele achiziționate sau care sunt furnizate de intrările analogice (ex. convertorul analogic-digital).

Divizorul rezistiv de tensiune are avantajul simplității schemei electrice folosite și a numărului redus de componente electrice și electronice dar are și dezavantajele existenței unui curent permanent de lucru care, pentru a avea o influență cât mai mică asupra sarcinii, trebuie să fie mult mai mare față de curentul pe care îl ia sarcina legată la divizorul rezistiv și a consumului de energie electrică pe durata alimentării circuitului.

## 7. Bibliografie

[1] **Allegro MicroSystems, LLC**, „*ACS711 Datasheet*,” 2013.

[2] „*ACS711EX Current Sensor Carrier*,” **Pololu Robotics & Electronics**, 15 04 2015. <https://www.pololu.com/product/2452>.

# Lucrarea 8. Utilizarea unui ceas în timp real

## 1. Descrierea lucrării

### ✓ Obiectivele lucrării

- Crearea și testarea de circuite de complexitate medie ce utilizează module externe.
- Utilizarea de module electronice tip *shield*.
- Realizarea unei aplicații practice de afișare pe un ecran LCD a unui ceas în timp real.

### ✓ Descrierea aplicației

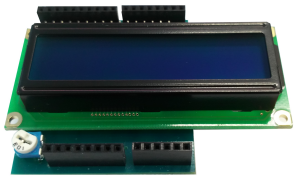


Scopul acestei aplicații este de a înțelege cum poate fi realizat un circuit electronic care să afișeze numeric pe un ecran LCD un ceas în timp real.

Pentru aceasta se va folosi un modul de ceas în timp real realizat cu circuitul integrat DS1307, capabil să furnizeze informații privind ziua lunii, luna, anul, ziua săptămânii, ora, minutele și secunde, printr-o comunicație serială de tip I<sup>2</sup>C, folosind pinii de date SCL și SDA disponibili pe placa Arduino Uno.

De reținut! Comunicația serială I<sup>2</sup>C (*Inter Integrated Circuit*) este un tip de comunicație *multi-master*, *multi-slave* inventată de Philips Semiconductor special pentru transmiterea de date între circuite integrate de viteză mică și procesoare sau microcontrolere. Magistrala de comunicație este formată din două linii, una pentru transmiterea/recepționarea datelor, SDA (*Serial Data Line*) și una pentru transmiterea/recepționarea semnalului de ceas, SCL (*Serial Clock Line*). Este obligatorie montarea câte unui rezistor de ridicare la 1 pe fiecare dintre cele două linii de date, iar fiecare circuit conectat la o magistrală I<sup>2</sup>C trebuie să aibă o adresă proprie.

## 2. Componente hardware

Componentele și modulele electronice utilizate în cadrul lucrării sunt cele din următorul tabel:

Componentă sau modul	Caracteristici	Număr bucăți	Imagine
<i>Arduino Uno</i>		1	
<i>Breadboard</i>	82x52x10 mm	1	
<i>LCD Shield</i>	Afișare pe 2 rânduri a câte 16 caractere	1	
<i>Fir de legătură</i>	Tată-Tată	4	
<i>Ceas în timp real</i>	DS1307	1	

### ✓ Observații

În această lucrare, pentru realizarea montajului electronic folosind componente externe se va utiliza o placă de testare de tip *breadboard* (Figura 1 – în partea din dreapta sunt simbolizate legăturile electrice între pini).

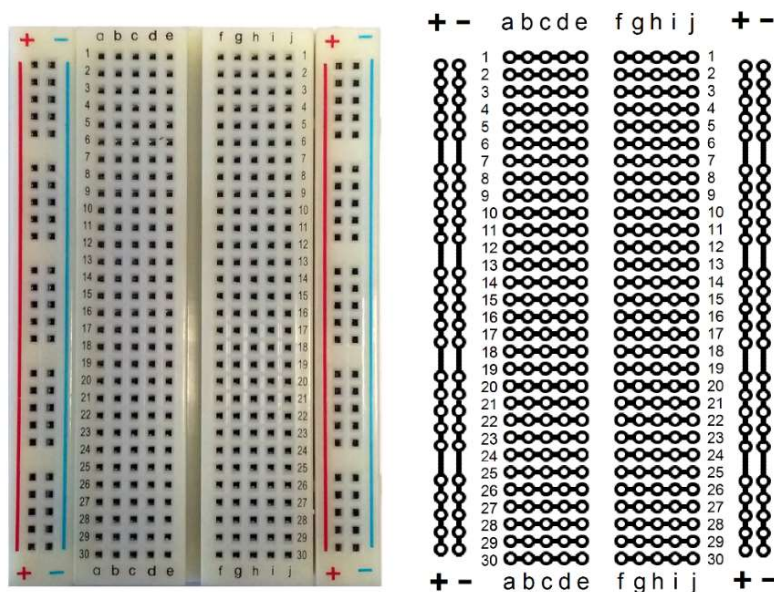


Figura 1. Breadboard-ul și conexiunile interne

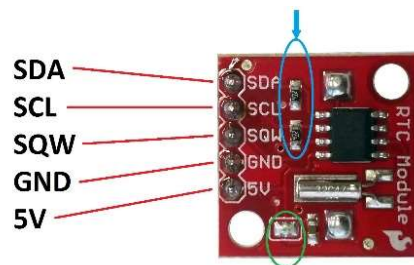
**Shield-ul LCD** permite afișarea de caractere pe un ecran cu cristale lichide, cu iluminare LED. Acesta se montează peste placa Arduino și are conectorii de așa natură încât pinii plăcii vor fi în continuare accesibili.

Ecranul LCD este format din 2 linii a câte 16 caractere, fiecare caracter fiind compus din 5x8 pixeli. Numerotarea coloanelor (caracterelor) se face de la 0 la 15 (de la stânga la dreapta), iar al rândurilor de la 0 la 1 (de sus în jos).

**Important!** Pentru a funcționa, *shield*-ul utilizează pinii digitali ai plăcii Arduino de la 2 până la 7 astfel: pinul 2 – *d7*, pinul 3 – *d6*, pinul 4 – *d5*, pinul 5 – *d4*, pinul 6 – *enable*, pinul 7 – *rs*.

**Ceasul în timp real** furnizează informații precum ziua lunii, luna, anul, ziua săptămânii, ora, minutele și secunde. Este realizat cu un circuit integrat specializat (DS1307) și folosește un circuit oscilant pe bază de cristal de cuarț cu frecvența de 32,768 kHz, furnizând la ieșire un semnal de date digital printr-o conexiune serială de tip I<sup>2</sup>C. Formatul orei poate fi setat între 12 ore cu bit pentru AM/PM sau 24 de ore, iar numărul de zile ale lunilor, precum și corecția pentru anii bisecți se face în mod automat [1]. Pentru ziua din săptămână ceasul furnizează cifre de la 0 până la 6 ce corespund intervalului de duminică până sâmbătă.

Modulul de ceas în timp real conține și două rezistoare de  $4,7\text{ k}\Omega$  conectate între fiecare dintre pinii de date SCL și SDA, și  $V_{CC}$ , cu rol de rezistoare de ridicare la 1 (vezi Figura 14). Atunci când o magistrală I<sup>2</sup>C este împărțită de mai multe module, fiecare având câte un set de rezistoare de ridicare la 1, se va păstra un singur set, prin eliminarea fludorului de pe jumper-ul încercuit cu verde în Figura 14.



**Figura 2. Utilizarea rezistoarelor de ridicare la 1**

Pinul SQW generează, când este activat, un semnal dreptunghiular cu o frecvență ce poate fi aleasă dintre patru valori: 1 Hz, 4 kHz, 8 kHz sau 32 kHz.

Având în vedere că se utilizează magistrala I<sup>2</sup>C, va trebui ca în secvența de cod să fie inclusă și biblioteca *Wire.h*, disponibilă deja în pachetul de biblioteci preinstalate în Arduino IDE.

Modulul de ceas în timp real se va alimenta cu tensiunea  $V_{CC} = 5\text{ V}$ .

### **Scrierea (setarea) datelor ceasului în timp real**

Deoarece nu există un mod de sincronizare automată a ceasului cu un ceas extern, setarea corectă a datelor ceasului se face manual de către utilizator.

În cazul ceasului în timp real nu se va mai utiliza o bibliotecă suplimentară. Datele vor fi scrise direct în registrele interne ale ceasului folosind funcții specifice utilizării magistralei I<sup>2</sup>C, pașii fiind următorii:

- Activarea magistralei I<sup>2</sup>C pentru scriere, menționând adresa ceasului.
- Setarea adresei primului registru de unde se va începe scrierea datelor.

- Scrierea datelor în registre. Datorită faptului că datele sunt stocate în registre în format BCD (zecimal codat binar), acestea vor fi convertite în prealabil din format zecimal în BCD.
- Închiderea magistralei I<sup>2</sup>C pentru scriere.

Conținutul registrelor interne de stocare a datelor este prezentat în tabelul următor [1]:

Adresă	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Funcție	Domeniu
00h	CH	Zeci Secundă			Unități Secundă				Secundă	00-59
01h	0	Zeci Minut			Unități Minut				Minut	00-59
02h	0	12	PM/AM	Zeci Oră	Unități Oră			Oră	1-12 +AM/PM	
		24	Zeci Oră						00-23	
03h	0	0	0	0	0	Unități Zi săptămână		Zi săptămână	01-07	
04h	0	0	Zeci Zi lună		Unități Zi lună			Zi lună	01-31	
05h	0	0	0	Zeci Lună	Unități Lună			Lună	01-12	
06h	Zeci An				Unități An			An	00-99	
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	-
08h-3Fh									RAM 56x8	00h-FFh

Observații:

- Dacă bitul 8 al registrului 00h este setat în 1 se dezactivează oscilatorul (ceasul nu mai funcționează) cu scopul de a minimiza curentul consumat (datele ceasului trebuie actualizate la o nouă folosire).
- Dacă bitul 7 al registrului 02h este setat în 0, afișarea orelor se face în formatul de 24 de ore.
- Dacă bitul 7 al registrului 02h este setat în 1, afișarea orelor se face în formatul de 12 de ore, bitul 6 furnizând în acest caz informația AM (valoarea 0) sau PM (valoarea 1).

*Transformarea datelor din zecimal în BCD*

Codarea de tip BCD alocă fiecărei cifre din sistemul zecimal (0, 1,...9) un cod binar de patru biți (0000, 0001,...1001). Dacă un număr din sistemul zecimal are  $n$  cifre, prin codare BCD îi va corespunde un cod format din  $n \cdot 4$  biți (de exemplu numărul 19 în zecimal se scrie ca fiind 0001 1001 în BCD).

Registrele interne ale ceasului stochează date pe 8 biți, codate BCD, adică prin decodare vom obține un număr zecimal format din două cifre (cea a zecilor și cea a unităților).

Placa Arduino preia datele ce trebuie scrise în registrele ceasului în zecimal (așa cum sunt scrise de utilizator) și le transmite în cod binar, fără să știe că ceasul le va interpreta ca fiind codate în BCD. Astfel, după exemplul de mai sus, numărul 19 (0001 1101 în BCD) va fi transmis în binar ca 0001 0011 și va însemna pentru ceas 13. Prin urmare, înainte de a fi transmise, datele trebuie transformate din zecimal în BCD.

Transformarea din zecimal în BCD se face în secvența de cod și presupune realizarea următoarelor operații:

- Se împarte numărul inițial la 10. Împărțirea a două numere întregi oferă rezultatul fără rest, adică cifra zecilor ( $19 / 10 = 1$ , adică 0000 0001). Urmează translatarea biților rezultatului cu 4 poziții către stânga (codul 0000 0001 va deveni 0001 0000) prin care se păstrează practic cei patru biți care reprezintă cifra zecilor, și se aduc pe poziția cifrei zecilor în formatul BCD).
- Efectuarea operației *modulo* (returnează restul împărțirii a două numere întregi) între numărul inițial și 10 ( $19 \% 10 = 9$ ), rezultatul fiind cifra unităților din numărul inițial (0000 1001).
- Operarea unui SAU logic între cele două rezultate va duce la obținerea codului 0001 1001, adică 19 în BCD.

Formula de calcul finală, conform celor de mai sus, este următoarea:

$$octet_{BCD} = \left( \left( \left( \frac{valoare_{zecimal}}{10} \right) \ll 4 \right) | (valoare_{zecimal} \% 10) \right) \quad (1)$$

ATENȚIE! În secvența de cod, variabila care va conține `octetBCD` trebuie declarată ca fiind de tipul fără semn (de exemplu *byte*), prezența bitului de semn generând erori (vezi [2]).

### Citirea datelor ceasului în timp real

Datele vor fi citite direct din registrele interne ale ceasului folosind funcții specifice utilizării magistralei I<sup>2</sup>C, pașii fiind următorii:



- Activarea magistralei I<sup>2</sup>C pentru scriere, menționând adresa ceasului.
- Setarea adresei primului registru de unde se va începe citirea datelor.
- Închiderea magistralei I<sup>2</sup>C pentru scriere.
- Activarea magistralei I<sup>2</sup>C pentru citire, menționând adresa ceasului și numărul de registre (octeți în acest caz) din care vor fi preluate date.
- Citirea datelor din registre și alocarea lor unor variabile. Datorită faptului că datele sunt stocate în registre în format BCD (zecimal codat binar), acestea vor fi convertite în prealabil în format zecimal.

De reținut! Atunci când se alege formatul de 12 ore, din octetul corespunzător orei se extrag biții 1...5 care conțin informația despre oră și bitul 6 care conține informația despre AM sau PM.

#### *Transformarea datelor din BCD în zecimal*

Placa Arduino recepționează fiecare octet de date interpretându-le ca pe un număr clasic ce folosește toți cei 8 biți, fără să știe că de fapt ele sunt codate BCD. Astfel combinația 0001 1001 va însemna în zecimal 25 (și nu 19 cum ar trebui).

Transformarea din BCD în zecimal se face în secvența de cod și presupune realizarea următoarelor operații:

- Translatarea biților octetului de date cu 4 poziții către dreapta (codul 0001 1001 va deveni 0000 0001; se păstrează practic cei patru biți care reprezintă cifra zecilor) și înmulțirea cu 10 (rezultatul final va fi  $1 \cdot 10 = 10$ ).
- Operarea unui ȘI logic între octetul de date și un octet cu valoarea 0000 1111 ( $0001\ 1001 \& 0000\ 1111 = 0000\ 1001 = 9$ ; se păstrează practic cei patru biți care reprezintă cifra unităților).
- Se adună rezultatele operațiilor de mai sus ( $10 + 9 = 19$ ).

Formula de calcul finală, conform celor de mai sus, este următoarea:

$$valoare_{zecimal} = \left( (octet_{BCD} \gg 4) * 10 \right) + (octet_{BCD} \& 1111) \quad (2)$$

ATENȚIE! În secvența de cod, variabila care va conține octetul<sub>BCD</sub> trebuie declarată ca fiind de tipul fără semn (de exemplu *byte*), prezența bitului de semn generând erori (vezi [2]).

### 3. Componente software

*LiquidCrystal.h* este biblioteca ce conține comenzile pentru *shield*-ul LCD.

*Wire.h* este biblioteca ce conține comenzile pentru magistrala I<sup>2</sup>C.

*LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)* creează o variabilă *lcd* specificându-se piniile digitali folosiți pentru a comanda *shield*-ul LCD.

*const* are semnificația de constantă modificând comportamentul unei variabile. Variabila va deveni de tip *Read-only* adică valoarea ei nu va putea fi schimbată.

*int variabilă = valoare* stabilește o valoare pentru o variabilă de tip întreg pe 16 biți, cu semn (de la -32.768 până la 32.767).

*byte variabilă* stabilește o variabilă de tip octet, fără semn.

*void setup()* este o funcție (care nu returnează date și nu are parametri) ce rulează o singură dată la începutul programului. Aici se stabilesc instrucțiunile generale de pregătire a programului (setare pini, activare porturi seriale, etc.).

*void loop()* este principala funcție a programului (care nu returnează date și nu are parametri) și este executată în mod continuu atâta timp cât placa funcționează și nu este resetată.

*if(condiție) {instrucțiune/i} else {instrucțiune/instrucțiuni}* testează îndeplinirea sau nu a unei condiții.

*Wire.begin()* este o funcție care inițializează magistrala I<sup>2</sup>C.

*Wire.beginTransmission(adresă)* este o funcție care deschide magistrala I<sup>2</sup>C în modul de transmitere/primire de date către/de la circuitul cu adresa specificată.

`Wire.endTransmission()` este o funcție care încheie transmiterea/primirea de date.

`Wire.write(byte(pointer_registru))` este o funcție care stabilește registrul de unde se va începe operația de citire de date.

`Wire.requestFrom(adresă, n)` este o funcție care deschide magistrala I<sup>2</sup>C în modul de citire de date (un număr de  $n$  registre) de la circuitul cu adresa specificată.

`Wire.read()` este o funcție care citește datele registru cu registru și le oferă ca rezultat.

`lcd.begin(coloane, rânduri)` inițializează interfața cu ecranul LCD și specifică numărul de rânduri și de coloane al acestuia.

`lcd.setCursor(coloană, rând)` stabilește poziția cursorului LCD. Pentru LCD-ul folosit în această aplicație numărul coloanelor este de la 0 la 15, iar al rândurilor de la 0 la 1.

`lcd.clear()` șterge ecranul LCD și mută cursorul în colțul din stânga sus.

`lcd.print()` afișează pe ecranul LCD datele (valori ale unor variabile)/textul dintre paranteze. Pentru a afișa un text este necesar ca acesta să fie plasat între ghilimele ("text"). Pentru a afișa valoarea unei variabile de tip *char*, *byte*, *int*, *long*, sau *string* se scrie numele variabilei și, opțional, baza de numerație a acesteia (*variabilă*, BIN sau DEC sau OCT sau HEX). Pentru a afișa valoarea unei variabile de tip *float* sau *double* se scrie numele variabilei iar după virgulă, numărul de zecimale dorit a se afișa (*variabilă*, nr. zecimală).

`Serial.begin(viteză)` stabilește rata de transfer a datelor pentru portul serial în biți/secundă (BAUD).

`Serial.println("text")` tipărește textul sub formă de caractere ASCII folosind portul serial, adăugând după acesta și trecerea la o linie nouă.

`return valoare` are rolul de a termin execuția unei funcții și de a returna o valoare.

`delay(ms)` pune în pauză programul pentru o durată de timp specificată în milisecunde.

*variabilă*  $\gg n$  este un operator care mută biții variabilei cu un număr de  $n$  poziții spre dreapta.

*variabilă*  $\ll n$  este un operator care mută biții variabilei cu un număr de  $n$  poziții spre stânga.

$\&$  este operatorul ȘI logic.

$|$  este operatorul SAU logic.

$\%$  este operatorul *modulo* care calculează și oferă restul împărțirii între două numere întregi.

$/$  este operatorul de împărțire care, în cazul împărțirii a două numere întregi, oferă rezultatul fără rest.

$==$  semnifică *egal cu*.

#### 4. Realizarea montajului electronic

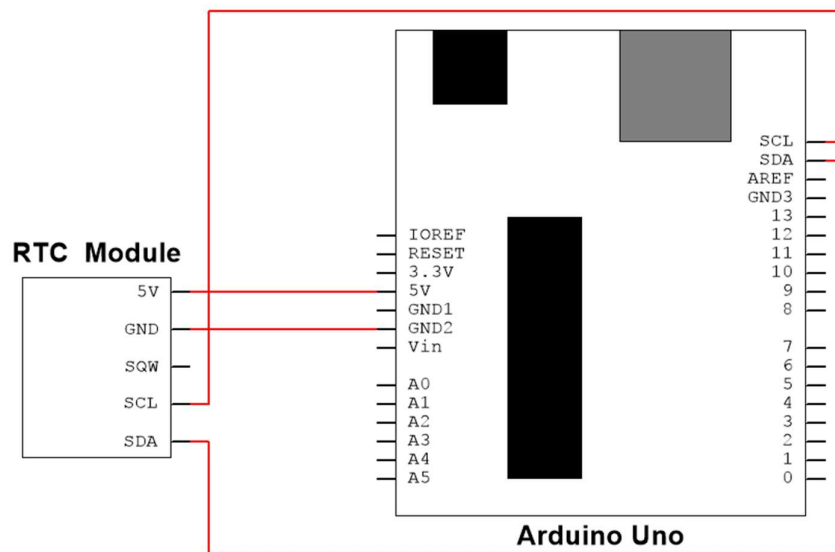


Figura 3. Schema de principiu

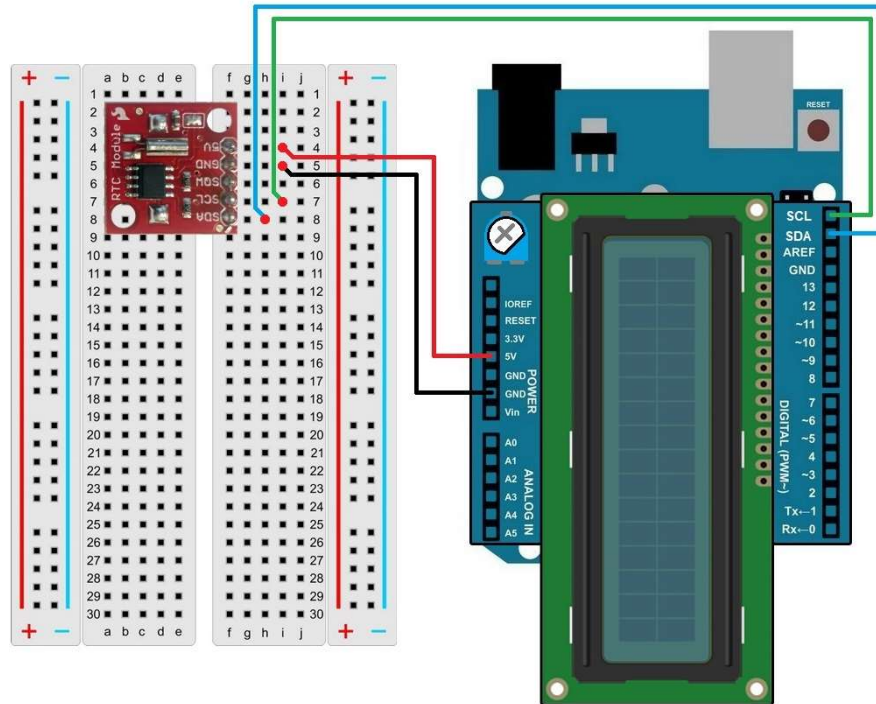


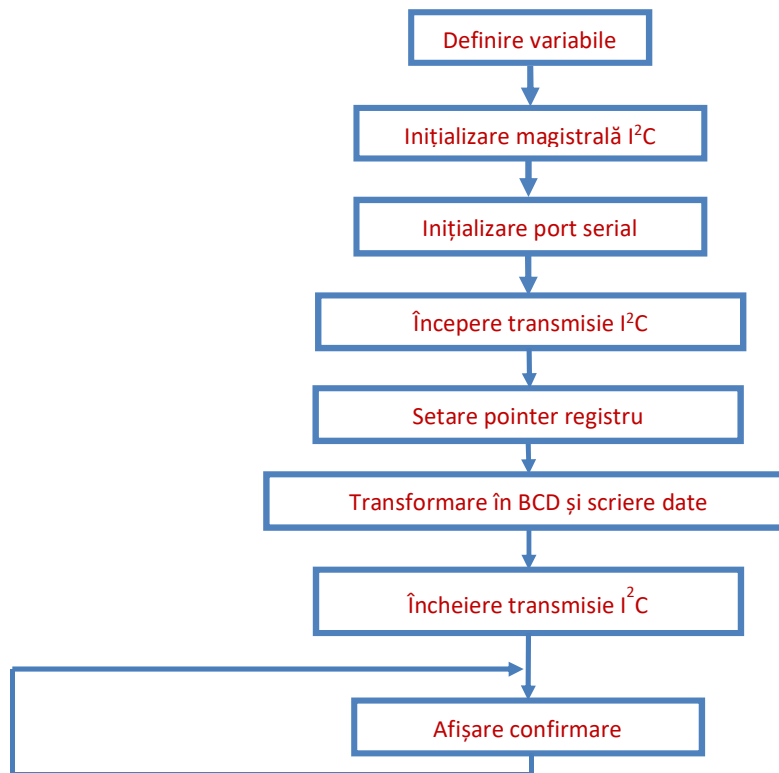
Figura 4. Realizarea conexiunilor electrice

Se realizează următoarele conexiuni:

- Pinul GND (power) de pe placa Arduino se conectează cu un fir la pinul GND al ceasului în timp real;
- Pinul 5V (power) de pe placa Arduino se conectează cu un fir la pinul 5V al ceasului în timp real;
- Pinul SCL de pe placa Arduino se conectează cu un fir la pinul SCL al ceasului în timp real;
- Pinul SDA de pe placa Arduino se conectează cu un fir la pinul SDA al ceasului în timp real.

## 5. Schema logică și secvența de cod

### Scrierea (setarea) datelor ceasului în timp real



```
#include <Wire.h>
//includerea în program a bibliotecii comenzilor pentru magistrala I2C
const int ADRESA_CEAS = 0x68;
//definirea variabilei corespunzătoare adresei ceasului
int secunda = 10;
//definirea variabilelor corespunzătoare datelor ceasului
int minut = 29;
int ora = 4;
int ziSaptamana = 5;
int ziLuna = 27;
int luna = 8;
int an = 15;

void setup(){
```

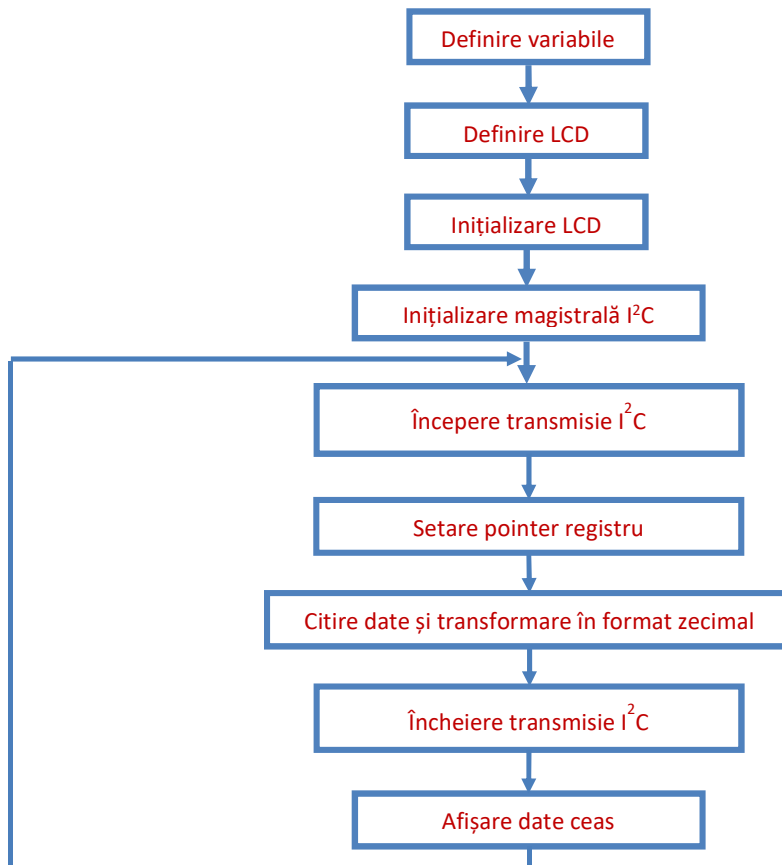
```

Wire.begin();
    //inițializarea magistralei I2C
Serial.begin(9600);
    //activează ieșirea portului serial cu rata de 9600 baud
Wire.beginTransmission(ADRESA_CEAS);
    //deschide magistrala I2C în modul de transmitere de date către adresa
    //specificată
Wire.write(byte(0x00));
    //stabilirea registrului de unde se va începe operația de scriere de date
Wire.write(ZecinBCD(secunda));
    //scriere valoare secunde, după conversie din zecimal în BCD
Wire.write(ZecinBCD(minut));
    //scriere valoare minute, după conversie din zecimal în BCD
Wire.write(ZecinBCD(ora));
    //scriere valoare oră, după conversie din zecimal în BCD - pentru
    //formatul de 24 ore
//Wire.write(ZecinBCD(ora) | 0b1100000);
    //scriere valoare oră, după conversie din zecimal în BCD - pentru
    //formatul de 12 ore, setare bit 6 în 1 pentru PM (sau 0 pentru AM) și
    //setare bit 7 în 1
Wire.write(ZecinBCD(ziSaptamana));
    //scriere valoare zi săptămână, după conversie din zecimal în BCD
Wire.write(ZecinBCD(ziLuna));
    //scriere valoare zi lună, după conversie din zecimal în BCD
Wire.write(ZecinBCD(luna));
    //scriere valoare lună, după conversie din zecimal în BCD
Wire.write(ZecinBCD(an));
    //scriere valoare an, după conversie din zecimal în BCD
Wire.endTransmission();
    //încheiere transmitere de date
}

void loop(){
    //conținutul buclei loop are rolul de a informa în privința terminării
    //procedurii de actualizare a ceasului
Serial.println("Ceasul a fost actualizat");
    //tipărește textul dintre ghilimele pe monitorul serial
delay(10000);
    //întârzie 10 secunde
}
byte ZecinBCD(byte valoare) {
    return ( ((valoare/10)<<4) | (valoare%10) );
    //funcție de conversie a datelor din zecimal în BCD - formula (1)
}

```

## Citirea datelor ceasului în timp real



```

#include <Wire.h>
    //includerea în program a bibliotecii comenzilor pentru magistrala
    I2C
const int ADRESA_CEAS = 0x68;
    //definirea variabilei corespunzătoare adresei ceasului
int secunda, minut, ora, ziSaptamana, ziLuna, luna, an;
    //definirea variabilelor pentru datele furnizate de ceas
//byte AMPM;
#include <LiquidCrystal.h>
    //includerea în program a bibliotecii comenzilor pentru LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
  
```



```
//inițializarea bibliotecii și a variabilei lcd cu numerele pinilor  
utilizați de shield-ul LCD
```

```
void setup(){  
  Wire.begin();  
  //inițializarea magistralei I2C  
  lcd.begin(16, 2);  
  //inițializarea interfeței cu ecranul LCD și specificarea numărului  
  de rânduri și de coloane al acestuia  
}  
  
void loop(){  
  Wire.beginTransmission(ADRESA_CEAS);  
  //deschide magistrala I2C în modul de transmitere de date către  
  adresa specificată  
  Wire.write(byte(0x00));  
  //stabilirea registrului de unde se va începe operația de citire de date  
  Wire.endTransmission();  
  //încheiere transmitere de date  
  Wire.requestFrom(ADRESA_CEAS, 7);  
  //deschide magistrala I2C în modul de citire de date (din 7 registre)  
  de la adresa specificată  
  secunda = BCDinZec(Wire.read());  
  //citire valoare secunde, conversie din BCD în zecimal și alocare  
  variabilă  
  minut = BCDinZec(Wire.read());  
  //citire valoare minute și conversie din BCD în zecimal și alocare  
  variabilă  
  ora = BCDinZec(Wire.read());  
  //citire valoare oră și conversie din BCD în zecimal și alocare  
  variabilă  
  //ora = Wire.read();  
  //citire valoare oră  
  //AMPM = ora & 0b100000;  
  //păstrare bit 6 din valoare oră (dacă este 0 atunci este AM)  
  //ora = BCDinZec(ora & 0b11111);  
  //păstrare biți 1..5 din valoare oră (datele efective ale orei) și  
  actualizare variabilă oră  
  ziSaptamana = BCDinZec(Wire.read());  
  //citire valoare zi săptămână și conversie din BCD în zecimal și  
  alocare variabilă
```

```
ziLuna = BCDinZec(Wire.read());
    //citire valoare zi lună și conversie din BCD în zecimal și alocare
    //variabilă
luna = BCDinZec(Wire.read());
    //citire valoare lună și conversie din BCD în zecimal și alocare
    //variabilă
an = BCDinZec(Wire.read());
    //citire valoare an și conversie din BCD în zecimal și alocare
    //variabilă

lcd.clear();
    //ștergere conținut ecran LCD
lcd.print(ziLuna);
    //afișează pe ecranul LCD valoarea variabilei ziLuna
lcd.print("-");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.print(luna);
    //afișează pe ecranul LCD valoarea variabilei luna
lcd.print("-");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.print(an + 2000);
    //afișează pe ecranul LCD valoarea variabilei an
lcd.setCursor(0, 1);
    //mutare cursorul pe coloana 1, rândul 2
lcd.print(ora);
    //afișează pe ecranul LCD valoarea variabilei ora
lcd.print(":");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.print(minut);
    //afișează pe ecranul LCD valoarea variabilei minut
lcd.print(":");
    //afișează pe ecranul LCD textul dintre ghilimele
lcd.print(secunda);
    //afișează pe ecranul LCD valoarea variabilei secunda
//if (AMPM == 0) {
    //dacă AMPM = 0 se afișează AM, altfel se afișează PM
//  lcd.print(" AM");
// }
// else {
//  lcd.print(" PM");
```

```
// }
    delay(1000);
        //întârzie 1 secundă
    }

byte BCDinZec(byte valoare) {
    return (((valoare>>4)*10) + (valoare&0b1111));
        //funcție de conversie a datelor din BCD în zecimal - formula (2)
    }
```

## 6. Exerciții suplimentare și concluzii

1. Să se modifice secvența de cod astfel încât lunile, orele, minutele sau secunde de la 0 până la 9 să fie afișate sub forma 00, 01, ... 09.
2. Să se modifice secvența de cod astfel încât să se afișeze pe ecran și ziua din săptămână, sub formă prescurtată (lun, mar, mie, etc.)
3. Să se scrie un program care să ofere funcția de timer – afișarea unui mesaj predefinit la atingerea momentului de timp prestabilit de către utilizator.
4. Să se scrie un program prin intermediul căruia să fie efectuate anumite operații de citire sau scriere la porturile plăcii de dezvoltare la anumite momente de timp (pentru simularea ceasului de sincronizare utilizat în funcționarea automatelor de trafic și a instalațiilor de semaforizare).

Ceasul sau semnalul de ceas este foarte important în sincronizarea circuitelor electronice sau în abordarea secvențială a etapelor unui proces industrial (în cazul transporturilor se poate considera managementul traficului ca fiind un astfel de proces). Transmisiile între diferitele componente ale unui sistem pot fi sincrone sau asincrone. Transmisiile sincrone se fac prin utilizarea unui ceas comun de către toate echipamentele care transmit date în cadrul unui sistem. Transmisiile asincrone se fac prin includerea unei informații de sincronizare în mesajele sau datele transmise (în special pentru definirea intervalului de bit).

Sincronizarea diferitelor componente ale unui sistem de management al traficului se poate face prin utilizarea semnalelor de ceas din sistemele GNSS (*Global Navigation Satellite Systems*), cum ar fi GPS, sau prin

utilizarea unui canal de ceas dedicat. Sincronizarea este foarte importantă împreună cu menținerea unei referințe comune de timp pentru toate echipamentele sincronizate (momentul  $t = 0$ ).

Semnalul de ceas este important și în dezvoltarea echipamentelor pe baza circuitelor integrate digitale acesta având atât rolul de a sincroniza toate circuitele existente cât și de a asigura abordarea secvențială a programului sau logicii implementate în respectivul echipament.

## 7. Bibliografie

- [1] **Maxim Integrated**, „*DS1307, I2C Real-Time Clock - Datasheet*,” 2008.
- [2] **Arduino Reference**, „*Bitshift Operators*,”  
<https://www.arduino.cc/en/Reference/Bitshift>. [Accesat 25 08 2015].