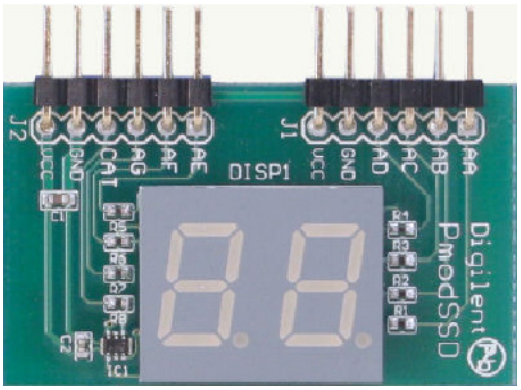
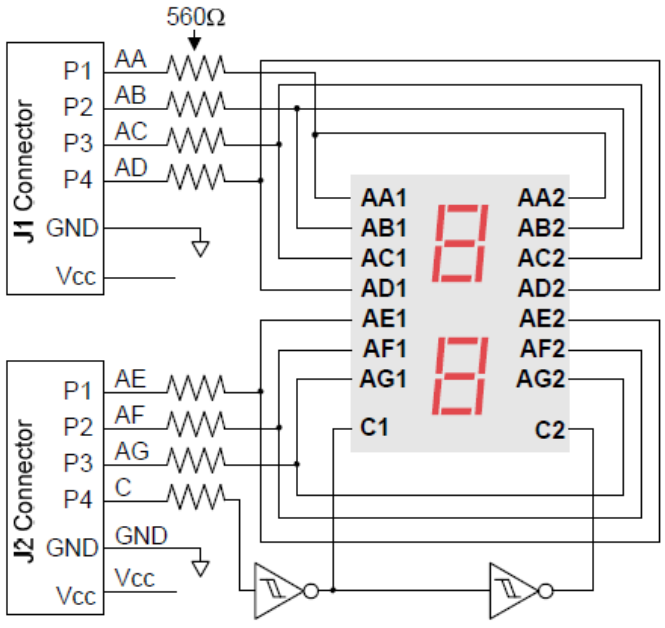
Laborator 2 – aplicații cu module simple de intrare şi ieşire.

1. Afisorul cu 2x7 segmente.

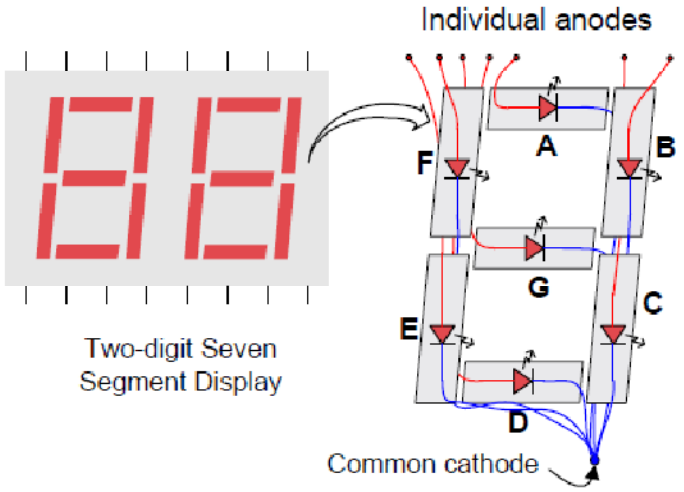
Modulul PmodSSD (Seven Segment Display), din figura de mai jos, oferă posibilitatea afișării a două caractere. Fiecare segment este un LED, care se aprinde dacă există combinația corectă de tensiuni pe anodul si catodul acestuia (anodul HIGH, catodul LOW).



Pentru a economisi numărul de pini de semnal necesari, cele două cifre nu pot fi aprinse simultan. Selecția dintre cele doua cifre o face semnalul CAT (C pe schema de mai jos). Dacă acest semnal este 0, se aprind segmentele cifrei unităților (cifra din dreapta), iar daca semnalul CAT este 1, se aprind segmentele cifrei din stânga.



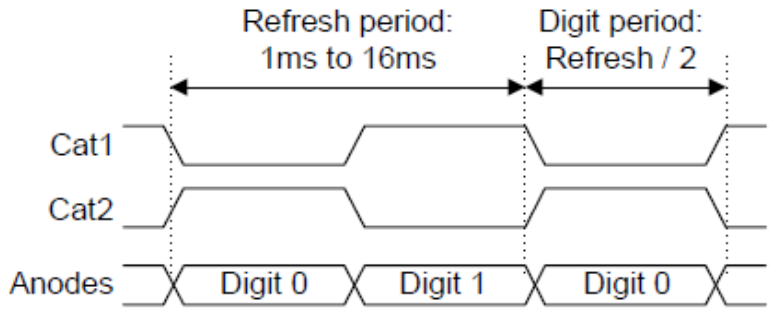
Semnalele notate cu litere AA…AC corespund anozilor LED-urilor segment. Corespondența acestor semnale este indicata de figura de mai jos:



De exemplu, pentru afişarea cifrei 3 va fi nevoie ca segmentele să primească următoarele nivele logice:

G	F	E	D	C	B	A
1	0	0	1	1	1	1

Pentru afişarea unui număr de două cifre, trebuie comutat foarte rapid între cele doua blocuri, folosind semnalul CAT. Următoarea diagramă de timp ilustrează procesul:



Pentru realizarea funcţionalităţii acestei diagrame de timp, se foloseşte următorul pseudocod:

```
Loop:
        AA…AG = cod_cifra_unitati
        CAT=0
        Delay()
        AA…AG=cod_cifra_zeci
        CAT=1
        Delay()
Goto loop
```

2. Utilizarea porturilor microcontrollerului pentru Intrare/Iesire

Arduino, prin funcțiile sale digitalRead şi digitalWrite, ascunde mecanismul prin care microcontrollerul face aceste operații. Mai mult, aceste funcții introduc întârzieri, care pot fi semnificative atunci când e nevoie de transferul datelor pe mai mulți biți.
Orice microcontroller este conectat cu exteriorul prin porturi de intrare/ieşire. AVR ATmega 2560 este un microcontroller pe 8 biți, deci şi porturile lui au 8 biți. Fiecare port are asociate trei registre (x va fi înlocuit cu A,B,C,D, …, în funcție de portul folosit):

– DDRx – registrul de direcție
– PORTx – registrul pentru date de ieşire
– PINx – registrul pentru date de intrare

Registrul de direcție DDRx

DDRx (Data Direction Register) configurează direcția datelor pe pinii portului (dacă un bit din port va fi folosit pentru intrare, sau pentru ieşire). Un 0 scris pe un bit din DDRx face ca pinul corespunzător din port să fie pin de intrare, iar un bit setat la 1 face ca pinul corespunzător să fie pin de ieşire.

Exemple:

- Pentru a configura toți pinii portului A ca intrare:
  DDRA = 0b00000000;
- Pentru a configura toți pinii portului A ca ieşire:
  DDRA = 0b11111111;
- Pentru a configura jumătatea inferioară a portului B ca ieşire, şi jumătatea superioară ca intrare:
  DDRB = 0b00001111;

Registrul de intrare PINx

PINx (Port IN) se foloseşte pentru citirea datelor de la pini configurați ca intrare. Pentru a se putea citi date, aceşti pini trebuie configurați ca intrare, setând biții din DDRx la zero.
Exemplu: citirea datelor din portul A

```
DDRA = 0;
char a = PINA;
```

Registrul de ieşire PORTx

Registrul PORTx este folosit pentru a transmite date de la microcontroller la perifericele conectate la pinii portului x. Pentru ca datele să fie vizibile la ieşire, biții corespunzători din registrul de direcție DDRx trebuie să fie configurați cu valoarea 1.
Exemplu: aprinderea din 2 in 2 a 8 led-uri conectate la portul A

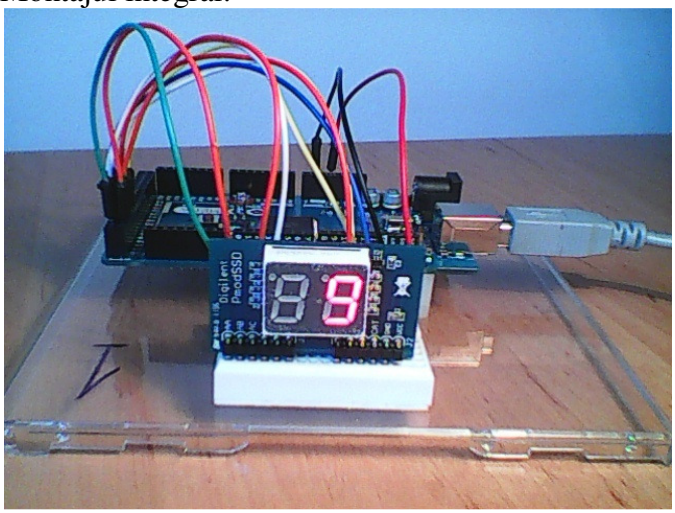
```
DDRA = 0xFF
PORTA = 0b10101010
```

3. Conectarea afișorului 7segmente la placa Arduino Mega 2560

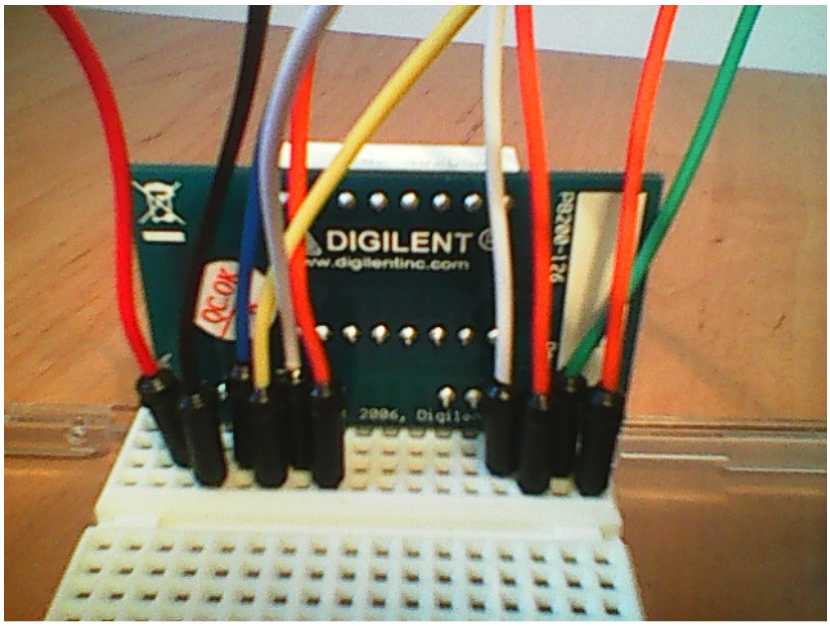
Montajul se face cu placa deconectată de la PC. Primul pas este introducerea afişorului în breadboard, apăsând ferm, dar fără a produce lovituri. Apoi vom conecta firele de alimentare, in semi-coloanele corespunzătoare VCC si GND pentru conectorul al doilea (J2) al afișorului. Celălalt capăt al firelor de alimentare îl vom conecta la pinii 5V si GND ai plăcii Arduino. Nu este nevoie sa aplicam tensiune si la conectorul J1.

Pentru pinii de semnal, corespunzători anozilor afișorului, AA…AG, şi pentru catodul CAT, vom conecta pe rând fire, pe care le vom conecta la placa Arduino la pinii digitali 22, 23, … 29, corespunzători biților 0… 7 ai portului A.

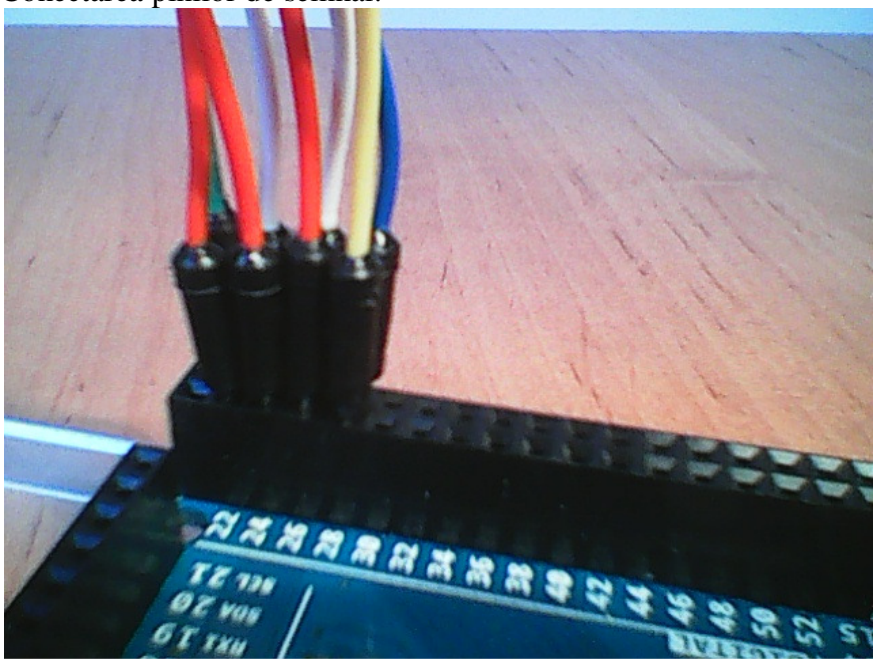
Montajul integral:



Detaliu cu firele conectate, in spatele afișorului:



Conectarea pinilor de semnal:



Atenție! Primii doi pini de pe conectorul plăcii Arduino sunt pini de alimentare! Pinii digitali încep cu a doua pereche! Mare atenție la poziția acestora!

Creați un nou proiect (sketch) Arduino, şi introduceți următorul cod:

```
// Afisare pe SSD
// conectat la PORTA
```

```
// Tabelă de valori (LUT) cu codurile pentru fiecare cifră de la 0 la 9. Fiecare bit corespunde unui LED, 1 înseamnă LED aprins, 0 înseamnă LED stins.

const unsigned char ssdlut[] = {0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111};
// dimensiunea LUT-ului
const int lutsize = 10;

int cpos = 0; // pozitia curenta
int cdigit = 0; // prima cifra din cele doua
unsigned char outvalue = 0;

void setup() {
  // activare PORTA ca iesire
  DDRA = 0b11111111;
}

void loop() {

  outvalue = cdigit>0 ? 0x80 : 0; // care catod il aprindem ? (00000000 sau 10000000)
                                  // catodul este legat la bitul 7 din portul A, prin această operație punem
                                  // bitul 7 pe 1 sau pe 0, alternativ, urmand ca ceilalti biti sa fie atasati
                                  // prin OR logic, in linia de cod urmatoare

  PORTA = (ssdlut[cpos] | outvalue); // facem OR intre valoarea din LUT si catodul selectat

  cpos++; // incrementam pozitia curenta

  if (cpos>=lutsize) { // daca am ajuns la capat
    cpos = 0; // revenim la zero
    cdigit^=1; // daca cifra anterioara a fost 0, acum e 1, daca a fost 1, acum e zero (^ = XOR)
  }

 // asteptare 0.5 sec
 delay(500);
}
```

Acest program va afișa cifre de la 0 la 9 pe primul element, apoi același lucru pe al doilea element.

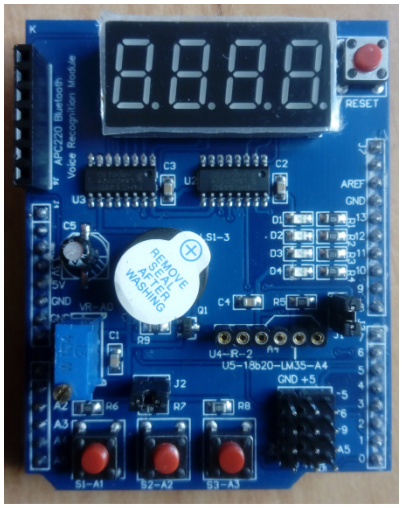
4. Shield-ul pentru învățare – Arduino Learning Shield

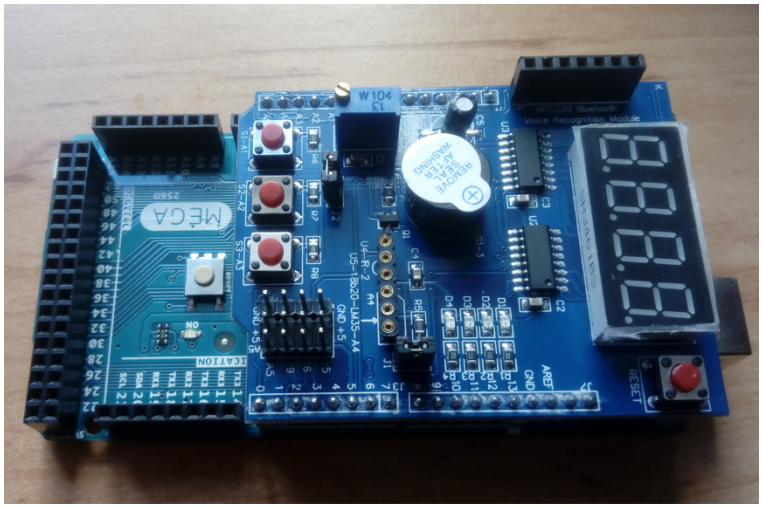
Shield-urile sunt PCB-uri (plăci de circuit imprimat – Printed Circuit Board) care pot fi plasate deasupra plăcilor Arduino, extinzându-le astfel capabilitățile. Există o varietate foarte mare de astfel de shield-uri, precum: XBee shield, SD-Shield, H-Bridge shield, etc. Pe PCB-urile Shield-urilor, anumite fire sunt trase către baretele de pini care urmează sa fie introduse în Arduino.

Aşadar, trebuie să fim atenţi la pinii folosiţi de componenta electronică, pentru a o putea mapa corect în program şi pentru a nu avea conflicte în utilizarea acestor pini.

În această lucrare de laborator vom utiliza un shield de învăţare, prezentat in figura de mai jos. Acesta dispune de următoarele resurse:

- 1 afişor cu 7 segmente, cu patru cifre
- 3 butoane (+ buton de reset)
- 4 leduri
- 1 potenţiometru pentru generare de semnal analogic
- 1 generator de sunet (buzzer)





Schema electrica a acestui shield se poate descarca de aici:
https://ardushop.ro/en/index.php?controller=attachment&id_attachment=40

Resursele shield-ului pot fi accesate astfel:

Butoanele sunt conectate la pinii A1, A2 si A3. Aceste butoane dispun de rezistenţe pull-up pe shield, astfel încât în starea liberă pe pini va fi citit nivelul logic 1, iar la apăsarea butonului se va citi nivelul logic 0. Nu este nevoie de activarea rezistorilor Pull-Up interni.

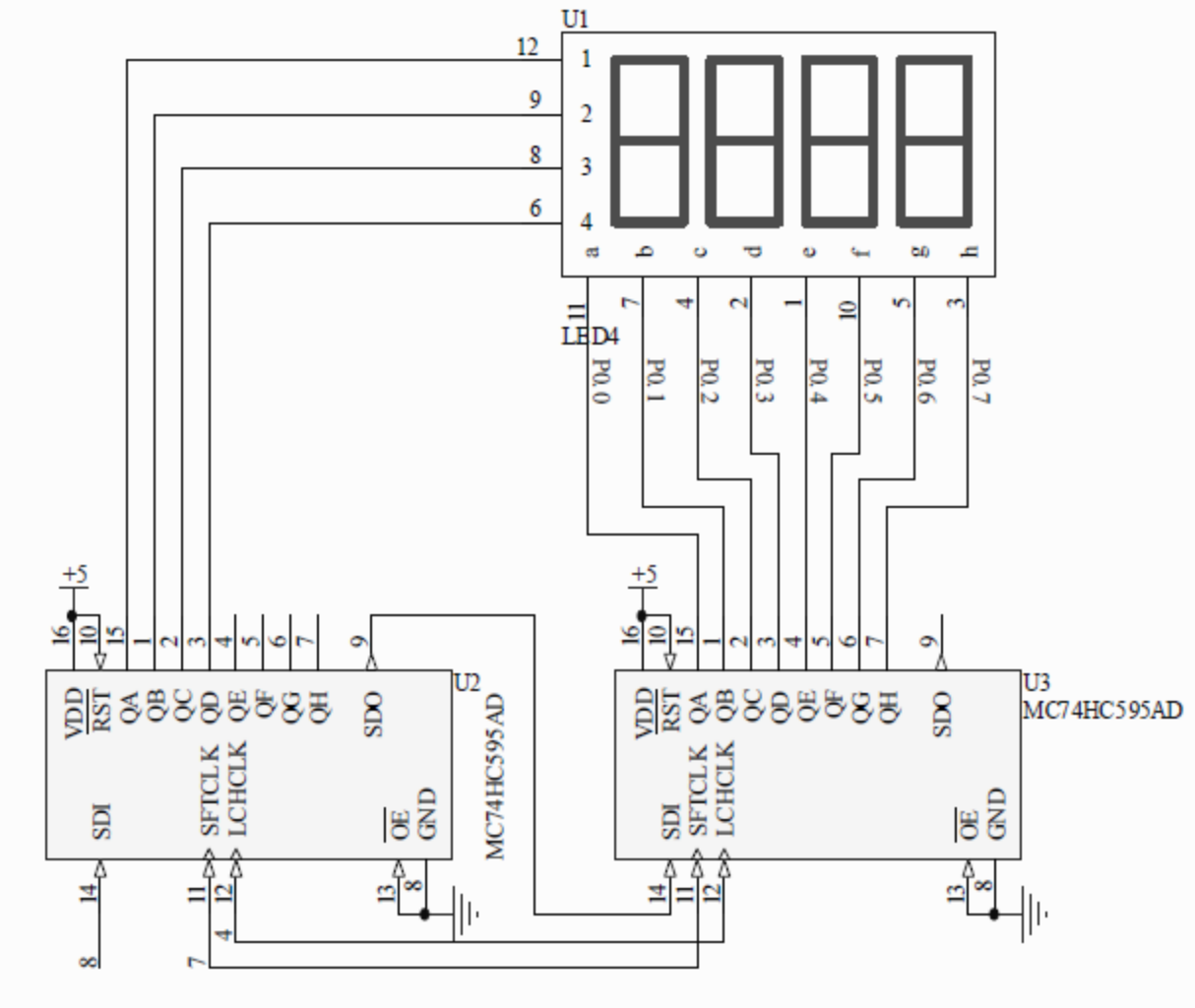
Led-urile sunt conectate cu anodul la VCC, si **catozii la pinii 10, 11, 12 si 13**. Astfel, pentru aprinderea unui LED trebuie scris nivelul logic 0 pe pinul corespunzător.

Afişorul 4x7 segmente este organizat în modul anod comun, şi catozi separaţi pentru fiecare segment de cifră, spre deosebire de PMod SSD, unde segmentele au catodul comun, si anozii separaţi. Astfel, pentru aprinderea unei cifre trebuie setat anodul cifrei pe nivelul logic 1, şi catozii segmentelor care trebuie aprinse pe zero.

Pentru comanda celor patru cifre, vom avea deci nevoie de 4 semnale pentru anozi, şi opt semnale pentru catozi (7 segmente, şi punctul). Aceste semnale nu sunt disponibile direct pentru utilizator, ci sunt conectate la doi regiştri de deplasare, conform schemei din figura de mai jos. Regiştrii de deplasare sunt legaţi în serie, ieşirea registrului de anozi fiind conectată la intrarea registrului pentru catozi. Astfel, pentru scrierea unei configuraţii complete (selecţie cifră activă + selecţia segmentelor care vor fi aprinse), este nevoie **doar de trei semnale**:

- SDI – serial data in, conectat la **pinul 8**

- SFTCLK – shift clock, semnalul de ceas pe care se vor prelua datele, conectat la **pinul 7**
- LCHCLK – latch clock pin, semnalul care va permite introducerea datelor, conectat la **pinul 4**.



Pentru scrierea unei cifre, trebuie ca prima dată să se transmită serial 8 biți conținând configurația catozilor, iar apoi 8 biți pentru configurația anozilor (din care doar 1 bit trebuie să fie 1, pentru cifra activă).

Următorul exemplu ilustrează utilizarea afişorului 4x7 segmente:

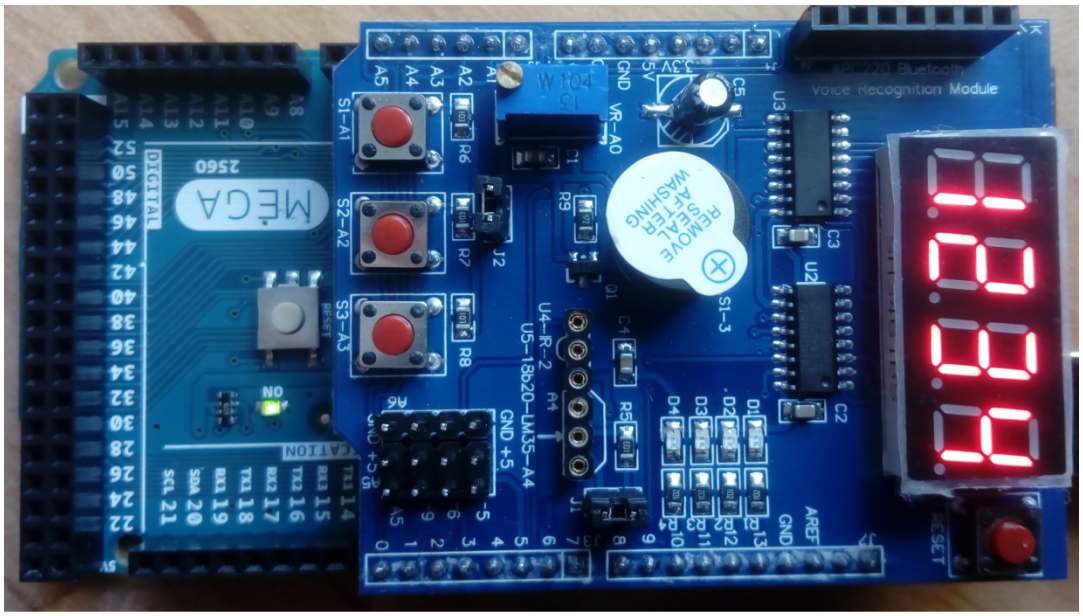
```
int latchPin = 4;
int clockPin =7;
int dataPin = 8; // Pinii SSD


const unsigned char ssdlut[] = {0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111};
const unsigned char anodelut[]  = {0b00000001, 0b00000010, 0b00000100, 0b00001000};

const unsigned char digits[] = {1,2,3,4}; // Numărul afişat va fi 1234. Modificați aici pt alt număr
```

```
void setup ()
{

  pinMode(latchPin,OUTPUT);
  pinMode(clockPin,OUTPUT);
  pinMode(dataPin,OUTPUT); // Cei trei pini pentru registrii de deplasare, configurați ca iesire

}

void loop()
{

    for(char i=0; i<=3; i++) // pentru fiecare din cele 4 cifre
    {

      unsigned char digit = digits[i];  // cifra curentă
      unsigned char cathodes = ~ssdlut[digit]; // catozii cifrei curente, vom nega valoarea din LUT

    digitalWrite(latchPin,LOW); // vom activa semnalul latch pentru a permite scrierea
    shiftOut(dataPin,clockPin,MSBFIRST, cathodes); // serializam octetul anozilor
    shiftOut(dataPin,clockPin,MSBFIRST, anodelut [i] );  // serializam octetul anozilor
    digitalWrite(latchPin,HIGH); // dezactivam semnalul latch
    delay(2);      // asteptare
    }

}
```

Analiza codului de mai sus:

- Utilizăm acelaşi LUT ca în exemplul precedent, dar aici vom aplica o operație de negare pe biți, pentru a ilustra faptul că acum se activează catozii, care necesită semnal 0.
- Arduino oferă funcția shiftOut, care transmite un octet în mod serial, pe pinul dataPin, generând şi un semnal de ceas pe pinul clockPin. Această funcție poate fi utilizată cu oricare doi pini digitali. Există două moduri de a serializa un octet: începând cu bitul 7 (MSBFIRST), sau începând cu bitul 0 (LSBFIRST). În acest caz, e nevoie să se aleagă varianta MSBFIRST, deoarece e important ca biții catozilor să corespundă cu segmentele cifrelor.

Rezultatul programului:



Lucru individual

1) Realizați montajul şi rulați primul exemplu din acest document.
2) Completați LUT-ul cu valori pentru numere hexazecimal (A, B, C, D, E). Testați programul cu numere de acest fel.
3) Adaptați programul pentru afișarea oricărui număr de la 0 la 99 pe afișor, in format decimal. Țineți cont că numărul nu este acelaşi lucru cu reprezentarea lui ca cifre separate pe jumătăți de octet (acest lucru este însă valabil in cazul numerelor hexazecimale). De exemplu, numărul 16 este reprezentat in binar ca 00010000, care va duce la afişarea numărului '10'. Pentru afişarea corecta a numerelor zecimale, trebuie făcuți următorii paşi:
- aflați cifra zecilor, câtul împărțirii cu 10
- aflați cifra unităților, restul împărțirii cu 10

CZ = numar div 10
CU = numar mod 10 = numar – (CZ*10)

4) Rulați al doilea exemplu, care utilizează Learning Shield.
5) Modificați exemplul 2, pentru a afişa orice număr de 4 cifre, în format zecimal. Numărul va fi dat ca **int**, nu ca cifre separate. Faceți ca numărul să se incrementeze periodic.
6) Utilizați butoanele de pe Learning Shield. Folosiți un buton pentru incrementarea numărului afişat, şi alt buton pentru decrementare.
7) Folosiți afişorul de 4 cifre, şi butoanele (Learning Shield), pentru a realiza un cronometru ce numără secundele (2 cifre) şi sutimile de secundă (2 cifre). Folosiți un buton pentru start, unul pentru stop, şi unul pentru reset. **Indicație: folosiți funcția millis() .**