Factor de umplere = duty cycle

R1: Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicatiei, placa emite un sunet de frecvență 400Hz și are unul dintre cele 2 LED-uri aprins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului crește cu 200Hz. LED-ul care inițial era aprins, este controlat de butonul 1, astfel: la prima apăsare a butonului 1, LED-ul începe să pâlpâie de 10 ori pe secundă, fiind stins doar a 10-a parte din timp (în timp ce pâlpâie). La următoarele apăsări pe butonul 1, LED-ul va pâlpâi tot de 10 ori într-o secundă, dar timpul cât e stins se dublează. După multiple apăsări, LED-ul va rămâne stins.

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 1 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 1? …. (explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că butonul 2 a fost apăsat de 3 ori, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsării de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R2: Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicației, placa emite un sunet de frecvență 1600Hz și are unul dintre cele 2 LED-uri stins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului scade cu 200Hz. LED-ul care inițial era stins, este controlat de butonul 1, astfel: la prima apăsare a butonului 1, LED-ul începe să pâlpâie de 10 ori pe secundă, fiind aprins doar a 10-a parte din timp (în timp ce pâlpâie). La următoarele apăsări pe butonul 1, LED-ul va pâlpâi tot de 10 ori într-o secundă, dar timpul cât e aprins se dublează. După multiple apăsări, LED-ul va rămâne aprins.

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 1 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 1? ….(explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că butonul 2 a fost apăsat de 3 ori, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsării de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R3: Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicatiei, placa emite un sunet de frecvență 800Hz și are unul dintre cele 2 LED-uri aprins. Apoi, la fiecare apăsare pe butonul 1, frecvența sunetului crește cu 100Hz. LED-ul care inițial era aprins, este controlat de butonul 2, astfel: la prima apăsare a butonului 2, LED-ul începe să pâlpâie de 5 ori pe secundă, fiind stins doar a 10-a parte din timp (în timp ce pâlpâie). La următoarele apăsări pe butonul 2, LED-ul va pâlpâi tot de 5 ori într-o secundă, dar timpul cât e stins se dublează. După multiple apăsări, LED-ul va rămâne stins.

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 2 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 2? …. (explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că butonul 1 a fost apăsat de 4 ori, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsării de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R4: Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicației, placa emite un sunet de frecvență 1200Hz și are unul dintre cele 2 LED-uri stins. Apoi, la fiecare apăsare pe butonul 1, frecvența sunetului scade cu 100Hz. LED-ul care inițial era stins, este controlat de butonul 2, astfel: la prima apăsare a butonului 2, LED-ul începe să pâlpâie de 5 ori pe secundă, fiind aprins doar a 10-a parte din timp (în timp ce pâlpâie). La următoarele apăsări pe butonul 2, LED-ul va pâlpâi tot de 5 ori într-o secundă, dar timpul cât e aprins se dublează. După multiple apăsări, LED-ul va rămâne aprins.

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 2 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 2? ….(explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că butonul 1 a fost apăsat de 4 ori, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsării de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R5: Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicatiei, placa emite un sunet de frecvență 1400Hz și are unul dintre cele 2 LED-uri aprins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului scade cu 200Hz. LED-ul care inițial era aprins, este controlat de butonul 1, astfel: la prima apăsare a butonului 1, LED-ul începe să pâlpâie de 3 ori pe secundă, fiind stins doar a 20-a parte din timp (în timp ce pâlpâie). La următoarele apăsări pe butonul 1, LED-ul va pâlpâi tot de 3 ori într-o secundă, dar timpul cât e stins creste de 3 ori. După multiple apăsări, LED-ul va rămâne stins.

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 1 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 1? …. (explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că butonul 2 a fost apăsat de 3 ori, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsării de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R6: Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicației, placa emite un sunet de frecvență 300Hz și are unul dintre cele 2 LED-uri stins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului creste cu 200Hz. LED-ul care inițial era stins, este controlat de butonul 1, astfel: la prima apăsare a butonului 1, LED-ul începe să pâlpâie de 3 ori pe secundă, fiind aprins doar a 20-a parte din timp (în timp ce pâlpâie). La următoarele apăsări pe butonul 1, LED-ul va pâlpâi tot de 3 ori într-o secundă, dar timpul cât e aprins se tripleaza. După multiple apăsări, LED-ul va rămâne aprins.

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 1, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 1 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 1? ….(explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că butonul 2 a fost apăsat de 3 ori, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsării de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R7: **Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicației, placa emite un sunet de frecvență 300Hz și are unul dintre cele 2 LED-uri stins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului creste cu 200Hz, iar la fiecare apăsare a butonului 1, frecventa scade cu 200Hz. LED-ul care inițial era stins, se va aprinde acum cu o frecventa egala cu a 100-a parte din frecventa semnalului aplicat pe buzzer. De asemenea, in functie de numarul de apasari, factorul de umplere al semnalului aplicat pe led va incepe de la un factor de umplere 1/10 si va creste cu numarul de apasari de (orice) buton cu inca 1/10.**

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 2 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 2? ….(explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că s-au apăsat in ordine: butonul 2 de 2 ori, apoi butonul 1, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsărilor de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R8: **Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicației, placa emite un sunet de frecvență 1400Hz și are unul dintre cele 2 LED-uri stins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului scade cu 200Hz, iar la fiecare apăsare a butonului 1, frecventa creste cu 300Hz. LED-ul care inițial era stins, se va aprinde acum cu o frecventa egala cu a 100-a parte din frecventa semnalului aplicat pe buzzer. De asemenea, in functie de numarul de apasari, factorul de umplere al semnalului aplicat pe led va incepe de la un factor de umplere 9/10 si va scadea cu numarul de apasari de (orice) buton cu inca 1/10.**

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 2 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 2? ….(explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că s-au apăsat in ordine: butonul 2 de 2 ori, apoi butonul 1, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsărilor de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

R9: **Pe placa de test cu ESP-12 din laborator, s-a implementat următoarea aplicație: la pornirea aplicației, placa emite un sunet de frecvență 300Hz și are unul dintre cele 2 LED-uri stins. Apoi, la fiecare apăsare pe butonul 2, frecvența sunetului va creste cu 200Hz, iar la fiecare apăsare a butonului 1, frecventa va scadea cu 300Hz. LED-ul care inițial era stins, se va aprinde acum (deci dupa o prima apasare pe oricare dintre butoane) cu o frecventa egala cu a 100-a parte din frecventa semnalului aplicat pe buzzer. De asemenea, in functie de numarul de apasari, factorul de umplere al semnalului aplicat pe led va incepe de la un factor de umplere 1/10 si va creste cu numarul de apasari de (orice) buton cu inca 1/10.**

a) Desenați forma de undă a semnalului aplicat pe LED, după prima apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

b) Desenați forma de undă a semnalului aplicat pe LED, după a doua apăsare a butonului 2, specificând durata cât LED-ul e aprins și durata cât LED-ul e stins: ….

c) După ce s-a apăsat butonul 2 de 4 ori, care va fi factorul de umplere al semnalului aplicat pe LED-ul controlat de la butonul 2? ….(explicati)

d) Scrieți instrucțiunile prin care implementați efectul de la punctele a)-c):

e) Presupunând că s-au apăsat in ordine: butonul 2 de 2 ori, apoi butonul 1 o data, desenați forma de undă a semnalului aplicat pe buzzer, încă de la începutul aplicației, specificând pe desen momentul apăsărilor de buton.

f) Scrieți instrucțiunile prin care implementați efectul de la punctul e):

Sugestii implementare Aplicatie: - funcțiile de temporizare ***delay, millis*** și ***micros***.

Până acum, am folosit funcţia delay pentru a opri programul pentru o anumită durată de timp (exprimată în milisecunde şi specificată ca parametru al funcţiei -sintaxa funcţiei delay este ***delay(nr_milisec)***). În cazul în care avem nevoie în aplicaţii de o perioadă mai precisă de timp, există ca alternativă la funcţia delay, funcția DelayMicroseconds, care are aceeași funcționalitate ca şi funcţia delay, dar funcționează cu microsecunde (sintaxa funcţiei este ***DelayMicroseconds(nr_microsec)***). Aceste 2 funcţii sunt foarte uşor de utilizat în cadrul aplicaţiilor, dar ambele au dezavantajul că orice altă acţiune nu poate fi efectuată pe durata utilizării lor. Acestea se mai numesc şi "funcții de blocare", deoarece ele împiedică un program să realizeze alte operaţii până când acea sarcină este finalizată. Folosind aceste funcţii, nu se va executa nicio altă sarcină în timpul specificat ca parametru la intrarea în funcţie.

O alternativă la utilizarea funcţiilor delay şi delayMicroseconds o reprezintă utilizarea funcţiei ***millis***, care nu are nici un parametru de intrare, în schimb are parametru la ieşire (returnează numărul de milisecunde care a trecut de la apelul ei). Sintaxa funcţiei millis este: ***unsigned long val = millis().***

	Parametri la intrare	**Parametri la ieşire (returnaţi de funcţie)**
Funcţia *delay*	numărul de milisecunde în care dorim să întrerupem programul	nu va returna nimic
Funcţia *delayMicroseconds*	numărul de microsecunde în care dorim să întrerupem programul	nu va returna nimic
Funcţia *millis*	nu are niciun parametru de intrare	returnează timpul care a trecut

Pe de altă parte, dacă într-o aplicaţie este nevoie ca două sau mai multe acțiuni să fie executate simultan, nu trebuie utilizată funcția delay() deoarece programul va fi blocat la apelul acestei funcţii (orice acțiune va fi oprită pentru perioada de timp specificată în funcție). În exemplul următor, puteți vedea cum să clipiți simultan trei LED-uri la intervale de timp diferite, folosind funcția millis(), efect imposibil de obţinut cu funcția delay().

```
// exemplu de Program Blink 3LEDs with millis pe Arduino
int var[3]={11,12,13};   // variabila pentru controlul pinilor placutei
int val[3]={HIGH,HIGH,HIGH};   // valorile de tensiune de pornire pentru cele 3 LEDuri
int Perioada[3]={1000,1500,2000};   // perioada cu care va functiona fiecare LED
unsigned long Timp[3];  // folosita pentru functia millis
void setup() {
for (int i=0;i<=2; i++)
    {    pinMode(var[i], OUTPUT);  // declararea fiecarui pin ca OUTPUT pentru LED
         Timp[i] = millis();  // se porneste fiecare cronometru
     }
}
void loop() {
        for (int i=0;i<=2; i++)
            if(millis()-Timp[i]>=Perioada[i])   // se reciteste timpul curent si se verifica cu perioada
               {    val[i] = !val[i];   // inversarea valorii aplicate pe LED
                    digitalWrite(var[i],val[i]);   // reflectarea valorii de tensiune pe LED
                    Timp[i]=millis();  // actualizarea valorii timpului
               }
             }
```

În concluzie, funcția millis() permite programarea folosind diferite fire de execuţie în același timp și este mai precisă, în schimb, funcţiile delay() şi delayMicroseconds() sunt recomandate în programe simple numai dacă este necesară o acțiune de blocare a programului.

Exerciţiu propus: Scrieţi un program care să aprindă un singur LED folosind funcţia millis la un anumit interval de timp şi comparaţi codul rezultat cu cel al programului echivalent în care folosiţi funcţia delay sau functia DelayMicroseconds.

Lucrul cu butoanele:

Un buton se foloseşte pentru a conecta/deconecta două puncte dintr-un circuit atunci când este apăsat.

Exemplul următor aprinde LED-ul conectat la pinul 13 al plăcuţei Arduino atunci când se apasă butonul conectat la pinul 2 al plăcuţei.

Butonul simulat în aplicaţie dispune de 2 fire (picioruşe): unul dintre fire este conectat la pinul 2 al plăcuţei şi de asemenea se conectează printr-un rezistor pull-up la masă. Celălalt picior al butonului se conectează la sursa de 5 volți a plăcuţei.

Figura 1.a.

Când butonul este deschis (neapăsat) nu există nicio legătură între cele două picioare ale butonului, deci pinul este conectat la masă (prin rezistorul pull-up) și vom citi starea HIGH în aplicaţie (Figura 1.a). Când butonul este închis (apăsat), se va face o conexiune între cele două picioare ale butonului, conectând pinul la 5 volți, astfel încât vom citi starea LOW în aplicaţie. Comportamentul aplicaţiei la rulare va fi: LED-ul normal stins și aprins atunci când se apasă butonul.

Există posibilitatea să conectăm acest circuit în sens invers, cu un rezistor de pull-down (pin conectat la +5V), menținând starea LOW (când butonul nu este apăsat contact deschis) și mergând în starea HIGH atunci când butonul este apăsat (Figura 1.b). Dacă vom proceda astfel, comportamentul aceleaşi aplicaţii la rulare va fi inversat, cu LED-ul normal aprins și stins atunci când se apasă butonul.

Figura 1.b

! În aplicaţiile folosite în această lucrare, vom folosi întotdeauna primul caz: buton conectat prin rezistor pull-up, în mod normal la masă: buton neapăsat (contact deschis) – citim starea HIGH, buton apăsat (contact închis) – citim starea LOW.

Exemplu program Arduino: vom vedea un exemplu de program care să menţină aprins LED-ul de pe pinul IO13 atât timp cât butonul de pe pinul IO2 este apăsat. Când butonul nu este apăsat, LED-ul să fie stins.

În continuare, să presupunem că dorim să implementăm o aplicaţie prin care **LEDul de pe pinul 13 să fie stins la pornirea aplicaţiei (buton neapăsat) şi să se aprindă la apăsarea butonului.** Deoarece imediat după RESET, boot-loader-ul Arduino face IO13 de sens OUTPUT, nu e neapărat nevoie să-l configurăm; plăcuţa are şi un LED galben (în serie cu o rezistenţă de prescriere a curentului) către masă. Codul va fi scris diferit pentru cele 2 situaţii descrise mai sus, corespunzător Figurii 1.a, respectiv Figurii 1.b.

// Program 1.a – Blink LED on button press (Figura 1.a)

```
#define LEDpin 13
#define BUTON 2
void setup(){ pinMode(BUTON, INPUT);
            }
void loop(){
            if(digitalRead(BUTON) == LOW)       // daca e apasat, e ZERO Logic
                        digitalWrite(LEDpin, HIGH);
            else
                        digitalWrite(LEDpin, LOW);
}
```

// Program 1.b – Blink LED on button press (Figura 1.b)

```
#define LEDpin 13
#define BUTON 2
void setup(){ pinMode(BUTON, INPUT);
            }
void loop(){
            if(digitalRead(BUTON) == HIGH)       // daca e apasat, e UNU Logic
                        digitalWrite(LEDpin, HIGH);
            else
                        digitalWrite(LEDpin, LOW);
}
```

Exerciţiu propus 1: Modificați aplicația 1.a) a.î. să mai adăugați o pereche Led+buton și fiecare pereche să funcționeze diferit (de ex. când butonul 1 e apăsat, LED-ul 1 e stins, iar când butonul 2 e apăsat, LED-ul 2 e aprins).

Exerciţiu propus 2: Modificați aplicația 1.a) a.î. să mai adăugați un LED și programul să funcționeze astfel: la prima apăsare de buton va aprinde LED 1, la următoarea apăsare de buton va stinge LED 1 şi va aprinde LED 2, apoi similar la următoarea apăsare de buton să stingă LED 2 şi să aprindă LED 1, deci reia funcţionarea.

Lucrul cu butoanele – debouncing (filtrare zgomot): Butoanele generează adesea tranziții false de la apăsarea butonului, din cauza contactelor mecanice: aceste tranziții pot fi citite ca apăsări multiple într-un timp foarte scurt, oscilaţii parazite care pot duce la o funcţionare haotică a programului. Exemplul următor demonstrează cum se pot corecta aceste probleme, prin implementarea debuncingului, ceea ce înseamnă să verificăm de două ori într-o perioadă scurtă de timp pentru a ne asigura că butonul este apăsat cu siguranță. Consultaţi şi exemplul de la: https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce care foloseşte funcţia millis.

Programul de mai jos (Program 2.a) încearcă să complementeze starea LED-ului de pe IO13 la fiecare apăsare pe butonul de pe IO2. Totuși, dacă se menține butonul apăsat mai mult timp, aplicația 2.a) nu funcționează corect. Specificați cauza acestei probleme și modificați secvența de cod a.î. să funcționeze după specificațiile impuse de aplicație (Soluţia: Program 2.b).

// Program 2a – Blink LED on button press
(with no debouncing)

```
/* Sa se complementeze starea LED-ului de pe IO13 la fiecare
apasare pe butonul de pe IO2. */
#define LEDpin 13
#define BUTON 2

void setup(){ pinMode(BUTON, INPUT);  }
void loop(){
if(digitalRead(BUTON) == LOW)    // apasat = ZERO Logic
            digitalWrite(LEDpin, digitalRead(LEDpin) ^ 1);
}
```

Soluție: **// Program 2b – Blink LED on button press**
(with debouncing)

```
#define LEDpin 13
#define BUTON 2

void setup(){ pinMode(BUTON, INPUT); }
void loop(){
if(digitalRead(BUTON) == LOW){    // apasat = ZERO Logic
        delay(10); //ms
        if(digitalRead(BUTON) == LOW){
            while(digitalRead(BUTTpin) == LOW){ } // in timp ce butonul e apasat, nu facem nimic
            digitalWrite(LEDpin, digitalRead(LEDpin) ^ 1); // afectam LEDul doar la eliberarea lui
                                    }
                                }
    }
```