

Universitate Politehnica Timișoara

DOCUMENTAȚIE PROIECT

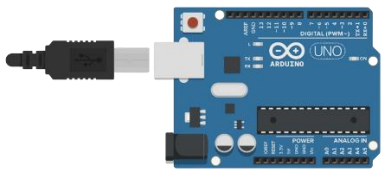
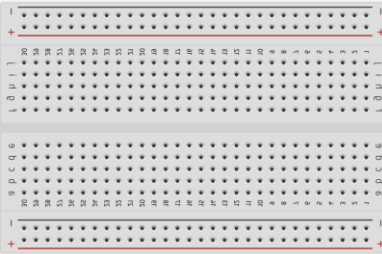
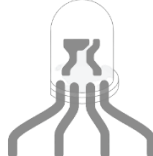


- ORGANIZAREA CALCULATOARELOR –

tinkercad.com/proiect-organizarea-calculatoarelor

Augustinov Thomas
Boticean Ștefan-Andrei

24.05.2023

1. Tabelul cu componentele utilizate în proiect:

	Plăcuța Arduino (Arduino UNO)	Tensiune de alimentare: 5V (USB) sau 7-12V (DC) Microcontroler: ATmega328P Porturi digitale: 14 (0-13)
	Breadboard	Simplifică asamblarea pieselor. Alimentată cu 5V.
	Led RGB	Tensiune de alimentare: 2.0 - 3.3 V
	LED galben	Tensiune de alimentare: 2.0 - 3.3 V
	Rezistor	Utilizat pentru ajustarea intensității curentului. 220 ohmi

2. Testarea fiecărui component în parte:

a. Plăcuța Arduino:

Se verifică dacă placa Arduino Uno primește tensiunea corectă de alimentare și funcționează corect. Apoi se conectează placa la un computer prin cablul USB și se verifică dacă este detectată. Se poate încărca un program simplu de testare, cum ar fi "Blink" pentru a asigura funcționarea corectă a plăcii.

b. Rezistență

Pentru a testa o rezistență trebuie folosit un multimetru setat pe modul de măsurare a rezistenței. Urmează conectarea cele două sonde ale instrumentului la capetele rezistenței și citirea valorii afișate. Aceasta ar trebui să se apropie de valoarea nominală a rezistenței.

c. Led RGB

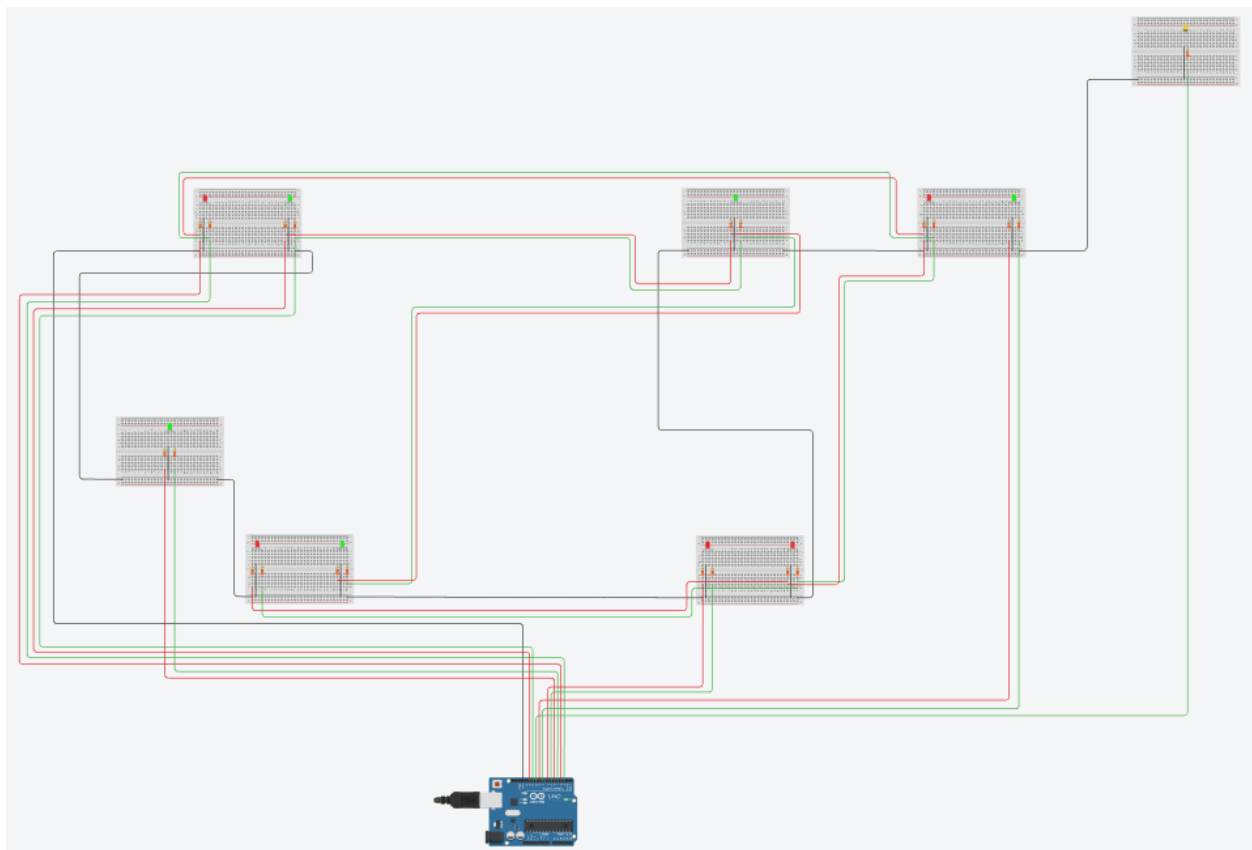
Pentru a testa fiecare LED RGB, trebuie utilizate rezistențe de 220-330 ohmi la fiecare pin R,G,B, conectate apoi rezistențele la porturi digitale ale plăcuței Arduino și conectat catodul la masă. Trebuie scris un program simplu pentru a aprinde fiecare culoare

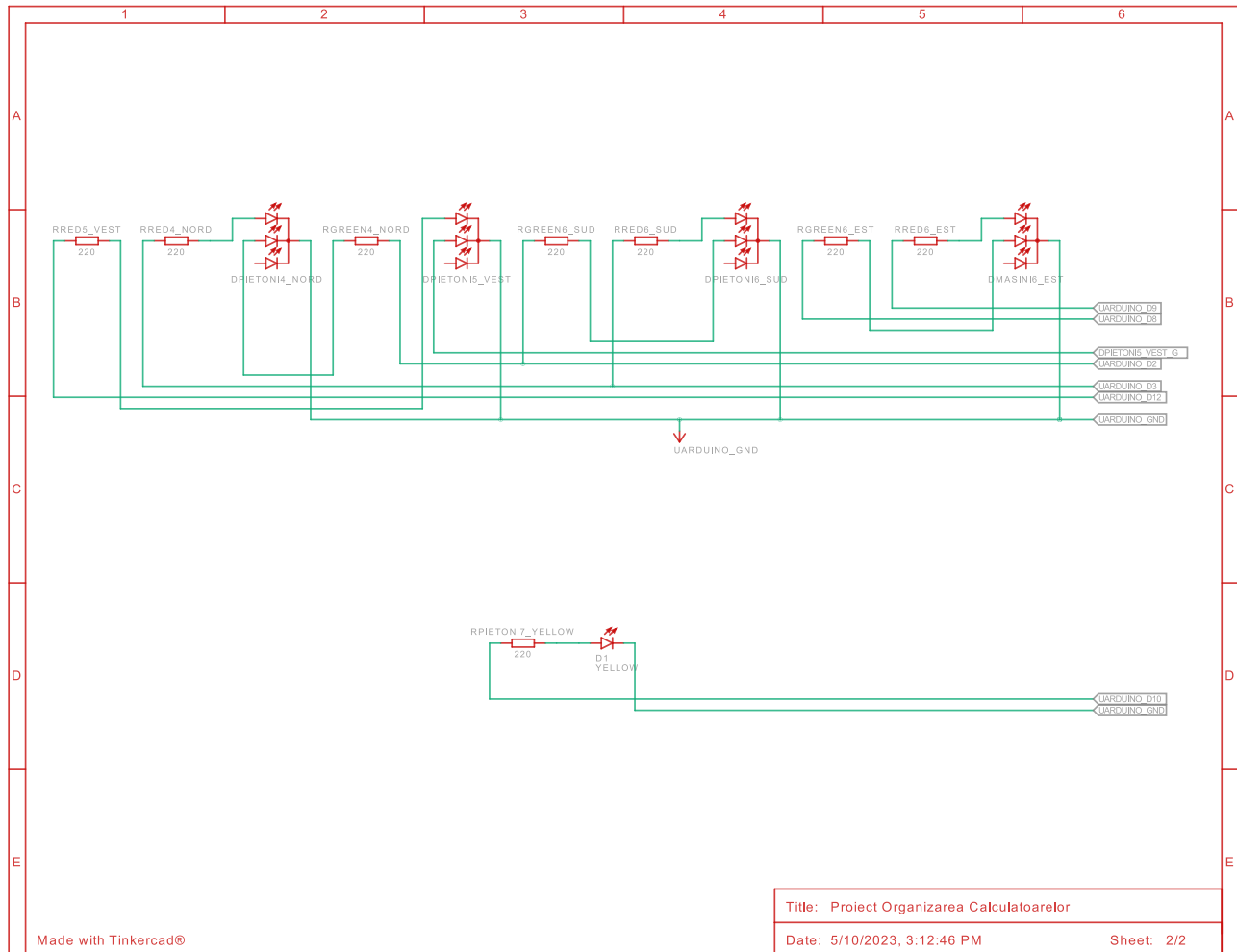
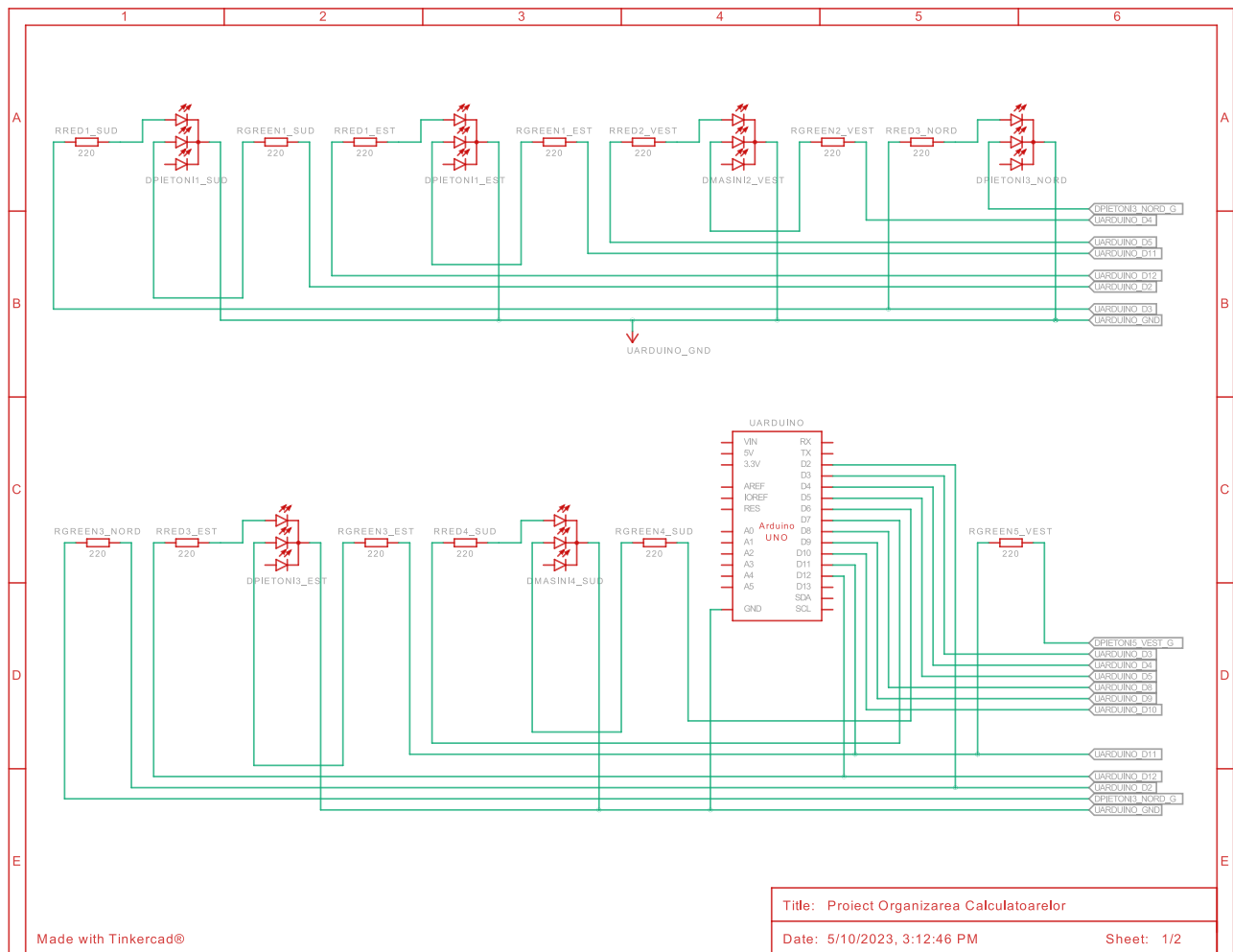
în parte (R, G și B) și trebuie asigurat că fiecare culoare funcționează corect.

d. Led Galben

Pentru a testa Led-ul Galben, trebuie utilizată o rezistență de 220-330 ohmi la anodul led-ului apoi aceasta împreună cu catodul led-ului conectate la porturi digitale ale plăcuței Arduino. Această testare poate lua loc la testarea plăcuței Arduino cu același program "Blink".

3. Arhitectura Hardware a sistemului





4. Arhitectura Software

- Explicarea pe scurt a codului și a funcțiilor folosite

setup() - Această funcție configurează toate pinii ca ieșiri și inițializează comunicarea serială. Este folosită pentru a pregăti placa Arduino și componentele pentru utilizare.

RGB_color() - Această funcție setează valorile red (roșu) și green (verde) pentru LED-urile RGB conectate la pinii specificați prin red_id și green_id.

digitalWrite_lights() - Această funcție setează starea LED-urilor în funcție de valorile stateRed, stateGreen și stateYellow.

RGB_color_lights() - Această funcție aplică valorile red (roșu) și green (verde) tuturor LED-urilor RGB utilizând funcția RGB_color().

blink() - Această funcție face ca LED-urile să clipească de 5 ori, alternând între starea HIGH și starea LOW, pentru pinii red_id și green_id.

blink_pietoni() - Această funcție face LED-urile pietonilor să clipească de 5 ori, alternând între starea HIGH și starea LOW, pentru pinul green_id.

blink_secondary() - Această funcție face LED-urile secundare să clipească de 5 ori, alternând între starea HIGH și starea LOW, pentru pinii green_id1, red_id2, și green_id2.

blink_yellow_light_5seconds() - Această funcție face LED-ul galben să clipească pentru o perioadă de aproximativ 5 secunde.

blink_yellow_light_1second() - Această funcție face LED-ul galben să clipească pentru o perioadă de aproximativ 1 secundă.

secondary() - Această funcție controlează secvența de iluminare a LED-urilor secundare.

primary() - Această funcție controlează secvența de iluminare a LED-urilor principale. În timpul rulării, verifică dacă există caractere noi disponibile în buffer-ul serial și acționează în consecință.

setBlink() - Această funcție face toate LED-urile să clipească, verificând în același timp dacă a fost primit un caracter nou în buffer-ul serial.

setDiagnostic() - Această funcție oprește toate LED-urile.

setRed() - Această funcție setează toate LED-urile roșii la starea HIGH.

setYellow() - Această funcție setează toate LED-urile galbene la starea HIGH.

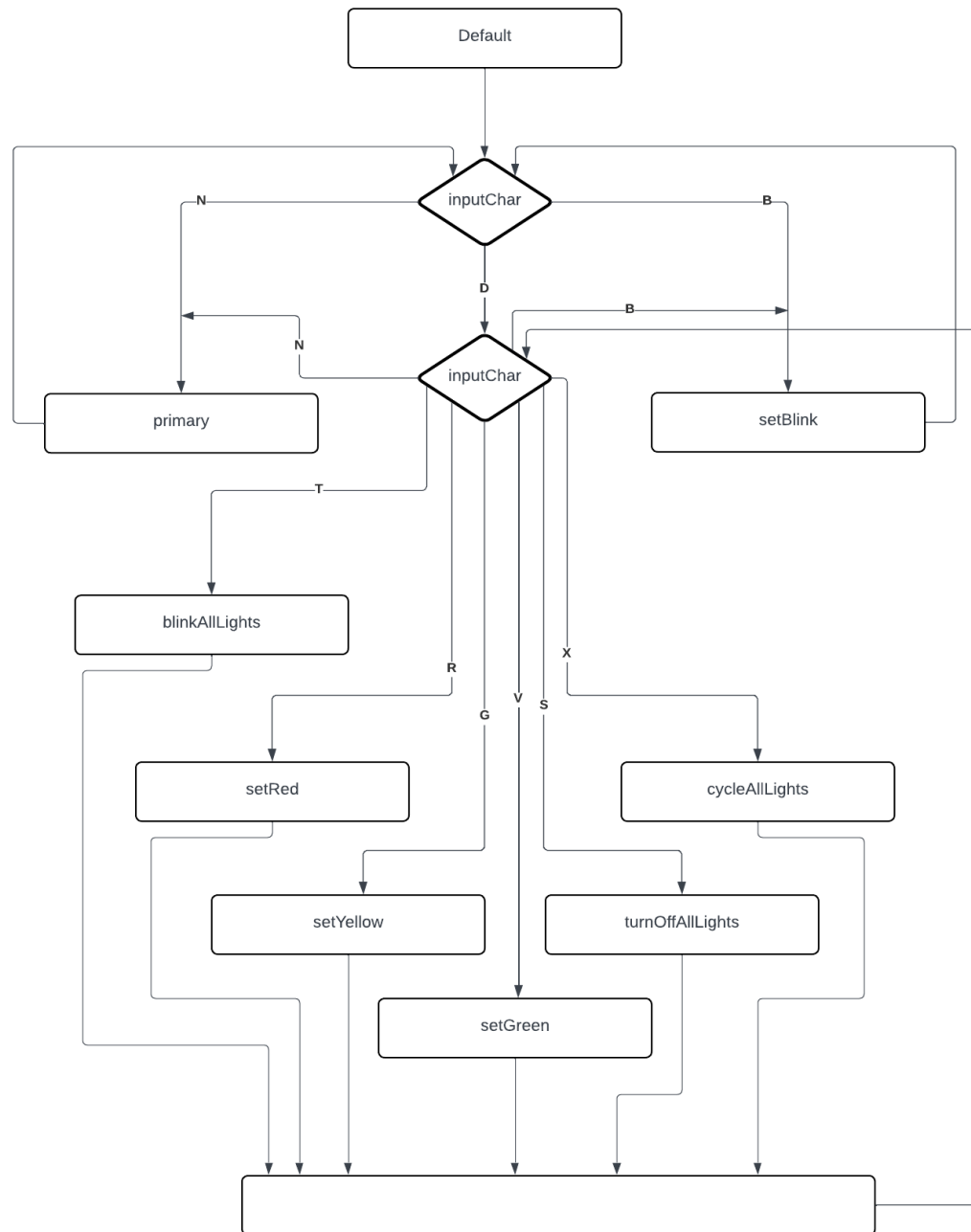
setGreen() - Această funcție setează toate LED-urile verzi la starea HIGH.

blinkAllLights() - Această funcție face toate LED-urile să clipească în funcție de intensitățile intensityRed și intensityGreen specificate.

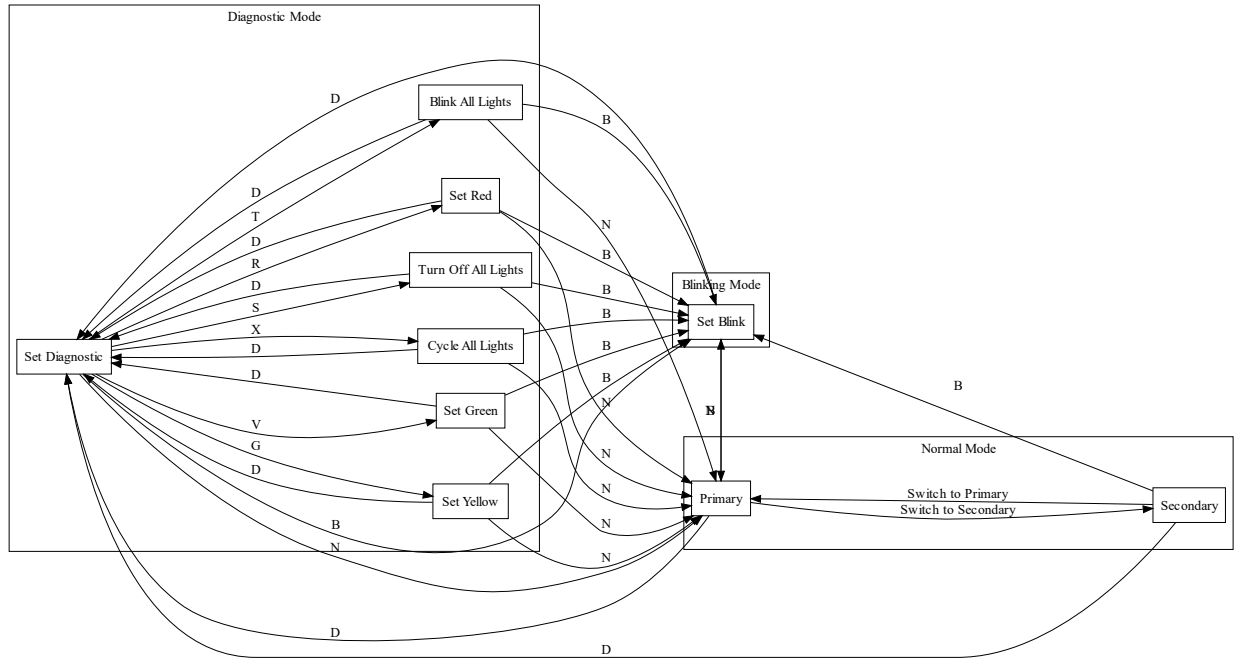
- Metode de a ajunge la rezultatul dorit

Pentru a ajunge la versiunea finală a proiectului am implementat pe rând fiecare stare în care se poate afla el, cu ajutorul funcțiilor descrise mai sus.

- Schema logică



- Diagrama de stare



5. Concluzii

1. Ce a fost mai greu de implementat?

Unul dintre cele mai provocatoare aspecte a fost gestionarea încorporării întârzierilor pentru a asigura succesiunea și sincronizarea corectă a sistemului de semaforizare.

2. Optimizarea codului.

O optimizare efectuată codului a fost a funcției `loop()`.

Înainte de optimizare (codul: <https://pastebin.com/tne3wM7G>):

Codul verifică dacă există date disponibile pe portul serial și citește un caracter dacă există. Apoi trece printr-o secvență de condiții `if` pentru a verifica ce caracter a fost primit și apelează funcția corespunzătoare pe baza caracterului primit.

O problemă majoră cu această abordare este că există un număr mare de repetări ale aceleași secvențe de cod:

```
if (inputChar == ...)
```

și necesitatea evaluării fiecărei condiții.

După optimizare (codul: <https://pastebin.com/RNh5zgAD>):

Codul verifică în continuare dacă există date disponibile pe portul serial și citește un caracter. Cu toate acestea, instrucțiunile repetate if au fost înlocuite cu instrucțiuni switch, care sunt mai eficiente și mai lizibile.

O instrucțiune switch verifică valoarea unei variabile și sare la primul caz de potrivire, prin urmare, nu trebuie să evalueze toate condițiile.

Optimizarea ajută la îmbunătățirea lizibilității și mentenanței codului, deoarece oferă o distincție clară între acțiunile care trebuie întreprinse în fiecare stare.