Blinky - Cub LED 4x4x4 cu animații și timer

Priboi Ioana-Mădălina 332CB

Introducere

Blinky este un cub LED 4x4x4 controlat de un microcontroller ATmega328P (Arduino Nano), care servește drept dispozitiv de birou pentru studenți. Proiectul combină o parte creativa, de jocuri de lumini, care acționeaza ca stress reliever, si o parte practica, de cronometrare a timpului de învățat sau a pauzelor.

Cubul are două moduri de funcționare, selectabile printr-un buton:

- **Modul Animații** - ruleaza în buclă jocuri de lumini 3D, viteza luminilor poate fi ajustată de utilizator printr-un potențiometru
- **Modul Timer** - utilizatorul selectează un interval de timp prin butoane, LCD-ul afișeaza timpul rămas, iar LED-urile indică vizual progresul numărătorii inverse. Expirarea timpului e semnalizată printr-un semnal sonor dat de buzzer.

Ideea a pornit de la un cub LED mult mai mare și complex (văzut pe Tik Tok), și mi-am dorit să fac și eu ceva asemănător, doar că pe o scară mai mică. După aceea, mi-a venit ideea de timer, fie pentru studiu (tehnica Pomodoro), fie ca timer general (la gătit, de exemplu)

Proiectul este util mai ales pentru studenti, pentru ca arată bine pe birou, alungă plictiseala, e un stress reliever bun si contribuie la un stil de învățat mai organizat.

Descriere generală

Proiectul are la bază un microcontroller ATmega328P (Arduino Nano), care acționeaza ca unitate centrală și coordonează toate modulele hardware si software.

Module hardware de intrare:

- **BTN_MODE** - schimbă modul de funcționare a cubului, între Animații și Timer
- **BTN_UP** - incrementează timpul setat în modul timer
- **BTN_DOWN** - decrementează timpul setat în modul timer
- **BTN_START/STOP** - pornește și oprește timerul
- **Potențiometru** - controlează viteza animațiilor în modul Animații

Module hardware de ieșire:

- **Cub LED 4x4x4** - afișeaza jocurile de lumini și progresul vizual al timer-ului prin cele 64 de led-uri, multiplexate pe 4 straturi
- **LCD 1602 I2C** - afișează modul curent de funcționare și timpul rămas
- **Buzzer pasiv** - semnalizează sonor expirarea timpului

Module software:

- **FSM** - gestionează stările sistemului și tranzițiile dintre ele
- **Driver LED** - implementează multiplexarea celor 4 straturi ale cubului
- **Motor Animații** - rulează jocurile de lumini 3D, cu viteza controlată din potențiometru
- **Motor Timer** - gestionează numărătoarea inversă, actualizează LCD-ul și declanșează alerta sonora la final

Interacțiunea modulelor:

Utilizatorul pornește cubul si selectează modul de funcționare prin apăsarea butonului BTN_MODE. Modul curent este afișat pe LCD, iar FSM-ul gestionează tranziția între stările sistemului.

În **modul Animații**, FSM-ul acționează Motorul de Animații, care gestionează jocurile (pattern-urile) de lumini. Driver LED-ul preia aceste pattern-uri si le afișează pe cub prin multiplexarea celor 4 straturi, trimițând datele catre shift registers prin SPI și comutând straturile prin tranzistoare. Viteza animațiilor e controlată prin Potențiometru, a cărui valoare e citită continuu prin ADC.

În **modul Timer**, utilizatorul setează durata dorită (în pasi de 10 secunde) prin butoanele BTN_UP si BTN_DOWN, iar LCD-ul afișează timpul setat. La apăsarea BTN_START/STOP, Motorul Timer pornește numărătoarea inversă și actualizează LCD-ul la fiecare 250ms. LED-urile indică vizual timpul rămas. La expirarea timpului, FSM-ul trece in starea de “alertă” ⇒ Driver LED-ul afișează animația corespunzătoare pe cub, iar Buzzer-ul emite un semnal sonor prin PWM. Timer-ul poate fi oprit oricand prin BTN_START/STOP.

Hardware Design

Listă de componente

Componentă	Cantitate	Rol
Arduino Nano V3 CH340 (ATmega328P)	1	Microcontroller
LED 5mm albastru	64	Cubul LED 4×4×4
Sârmă cupru	~2m	Structura cubului LED
Shift register 74HC595	2	Controlul coloanelor cubului
Tranzistor NPN 2N2222	4	Comutarea straturilor cubului
Rezistor 330 Ohm	16	Limitare curent LED-uri
Rezistor 1k Ohm	4	Baza tranzistoarelor NPN
LCD 1602 cu modul I2C	1	Afișarea modului și timer-ului
Buzzer pasiv 5V	1	Alertă sonoră la expirarea timpului
Potențiometru 10k Ohm	1	Controlul vitezei animațiilor
Buton tactil 6×6×6mm	4	BTN_MODE, BTN_UP, BTN_DOWN, BTN_START/STOP

Comutator ON/OFF	1	Pornirea/oprirea cubului LED
Alimentator 5V 2A	1	Alimentare externă
Condensator ceramic 100nF	2	Stabilizare alimentare pt shift register
Condensator electrolitic 100uF	1	Stabilizare alimentare generală pe breadboard
Placă perforată	1	Montaj cub LED
Breadboard 830 puncte	1	Prototipare și testare
Fire Dupont	multe	Conexiuni între componente

Schema electrică

Creierul proiectului este plăcuța Arduino Nano V3 CH340 (ATmega328P). Microcontroller-ul controlează cubul LED prin două registre de deplasare 74HC595, folosite pentru cele 16 coloane ale cubului, și prin patru tranzistoare NPN 2N2222, folosite pentru comutarea celor 4 straturi

Cubul este alcătuit din 64 de LED-uri albastre, organizate ca matrice 4x4x4. Coloanele verticale sunt anoduri comune, iar layerele orizontale sunt catoduri comune. Astfel, pentru aprinderea unui LED, se setează coloana corespunzătoare prin shift registere și se activează stratul corespunzator prin tranzistorul asociat.

Cele două shift registere 74HC595 sunt conectate în lanț. Microcontroller-ul trimite datele serial pe pinul D11, semnalul de clock este pe D12, iar latch-ul este pe D10. Primul shift register controlează coloanele C1-C8, iar al doilea coloanele C9-C16. Fiecare coloană este conectată printr-o rezistența de 330 ohm pentru limitarea curentului prin LED-uri.

Straturile cubului sunt comutate separat cu tranzistoare NPN. Bazele tranzistoarelor sunt conectate la pinii D2-D5 prin rezistențe de 1k ohm, emitorii sunt legați la GND, iar colectorii sunt conectați la catodurile comune ale straturilor. Prin activarea rapidă a straturilor, cubul generează animații 3D prin multiplexare.

Interfața utilizatorului este formată dintr-un LCD 1602 cu modul I2C, 4 butoane, un potentiometru și un buzzer pasiv. LCD-ul folosește magistrala I2C pe pinii A4 și A5, potentiometrul este citit analogic pe A0 (ADC), iar buzzerul este conectat pe D6 pentru generarea semnalului sonor

Pini folosiți

Pin Arduino	Port / Pin ATmega	Net Label	Componenta	Rol	De ce l-am ales
D0	PD0	-	-	RX	Rezervat pentru comunicatia USB/Serial
D1	PD1	-	-	TX	Rezervat pentru comunicatia USB/Serial
D2	PD2	LAYER1_CTRL	Tranzistor TR1	Control layer L1 (jos)	Pini digitali consecutivi pt controlul layerelor
D3	PD3	LAYER2_CTRL	Tranzistor TR2	Control layer L2	Pini digitali consecutivi pt controlul layerelor

D4	PD4	LAYER3_CTRL	Tranzistor TR3	Control layer L3	Pini digitali consecutivi pt controlul layerelor
D5	PD5	LAYER4_CTRL	Tranzistor TR4	Control layer L4 (sus)	Pini digitali consecutivi pt controlul layerelor
D6	PD6 / OC0A	BUZZER	Buzzer pasiv	Semnal sonor	Singurul pin OC0A al Timer0 - necesar pentru generarea tonului in hardware
D7	PD7 / PCINT23	BTN_MODE	Buton MODE	Schimbare mod	Pin digital cu suport PCINT
D8	PB0 / PCINT0	BTN_UP	Buton UP	Crestere timer	Pin digital cu suport PCINT
D9	PB1 / PCINT1	BTN_DOWN	Buton DOWN	Scadere timer	Pin digital cu suport PCINT
D10	PB2	SR_LATCH	74HC595 - RCLK	Latch	L-am grupat cu D11-D12 pt controlul shift registerelor
D11	PB3	SR_DATA	74HC595 - SER	Data	L-am grupat cu D10-D12 pt controlul shift registerelor
D12	PB4	SR_CLOCK	74HC595 - SRCLK	Clock	L-am grupat cu D10-D11 pt controlul shift registerelor - am evitat D13 (LED built-in)
D13	PB5 / SCK	-	-	-	L-am evitat, pentru ca este legat la LED-ul built-in
A0	PC0 / ADC0	POT	Potentiometru 10k	Citire analogica pt viteza animatii	Pin ADC
A1	PC1 / PCINT9	BTN_START_STOP	Buton START/STOP	Pornire/oprire timer	Pin analogic folosit ca digital, cu suport PCINT
A2	PC2	-	-	-	-
A3	PC3	-	-	-	-
A4	PC4 / SDA	LCD_SDA	LCD I2C - SDA	Date I2C	Pinul SDA hardware al Arduino Nano
A5	PC5 / SCL	LCD_SCL	LCD I2C - SCL	Clock I2C	Pinul SCL hardware al Arduino Nano

Am incercat sa pastrez o impartire cat mai logica: am grupat pinii aceleiasi componente pe acelasi port, ca sa fie mai usor de urmarit atat schema, cat si codul. Straturile sunt pe PORTD (PD2-PD5), shift registerele pe PORTB (PB2-PB4), iar I2C pe PORTC (PC4-PC5). Pentru butoane am verificat ca toti pinii au suport PCINT pe ATmega328P si sunt distribuiti pe porturi diferite, astfel fiecare buton are ISR-ul sau separat (BTN_MODE, BTN_UP/DOWN si BTN_START_STOP sunt pe porturi diferite (PORTD, PORTB, PORTC), deci fiecare port are ISR-ul sau. BTN_UP si BTN_DOWN impart acelasi ISR (PCINT0_vect), le disting in cod dupa starea pinului). Asa pot detecta apasarile fara sa blochez programul principal, iar multiplexarea cubului si timerul pot rula in paralel

Poze si teste hardware

Test shift registere

Inainte de a incepe sa lucrez la cub, am verificat placa Arduino Nano si cele doua shift registere 74HC595 folosind 16 LED-uri conectate direct pe breadboard. In aceasta etapa am testat daca datele trimise serial ajung corect la iesirile registrelor si daca pot controla individual coloanele.

Sablon pentru pozitionarea LED-urilor

Pentru a pastra LED-urile aliniate, am realizat un sablon din carton pentru fiecare layer. Am desenat o matrice 4×4 si am facut gauri in punctele de intersectie, astfel incat fiecare LED sa stea pe aceeasi pozitie in toate straturile. Distanta dintre doua LED-uri vecine este de 2.5 cm, iar gaurile au diametrul de 5 mm, cat sa intre LED-urile fara sa se miste prea mult.

Layer de test

Am realizat mai intai un layer de test pe sablonul din carton. Acest layer nu a fost folosit in cubul final, ci doar pentru antrenament si pentru a verifica daca distantele sunt bune, daca LED-urile stau drept si daca iese ok structura

Dupa lipire, am testat LED-urile individual, ca sa ma asigur ca fiecare LED se aprinde corect si ca nu exista scurtcircuite intre conexiuni

Realizarea layerelor

Dupa layerul de test, am facut separat cele 4 layere ale cubului. Am lipit catozii LED-urilor din acelasi layer impreuna, iar anozii i-am lasat liberi momentan, pentru formarea coloanelor verticale, ulterior

Asamblarea cubului final

Am suprapus cele 4 layere, tot la o distanta de 2.5mm intre ele, si am lipit coloanele verticale, adica anozii LED-urilor.

Astfel am obtinut structura finala a cubului LED 4x4x4. Am conectat toate straturile la GND comun si testat fiecare coloana sa vad ca se aprind corect toate LED-urile

Test cub LED

Dupa lipirea cubului, l-am conectat la breadboard si am verificat daca LED-urile se aprind corect. Am testat coloanele si layerele pe rand, ca sa ma asigur ca nu exista lipituri gresite sau conexiuni intrerupte. In poza se vede cubul in timpul testarii, cu cateva LED-uri aprinse.

Test periferice

Am testat mai intai LCD-ul, ca sa verific alimentarea si comunicatia I2C. Mesajul afisat pe ecran confirma ca LCD-ul functioneaza si poate fi folosit pentru afisarea modului curent si a timerului

Am conectat apoi potentiometrul, butoanele si buzzerul. Butoanele merg folosite pentru schimbarea modului si controlul timerului, potentiometrul pentru viteza animatiilor, iar buzzerul pentru semnalul sonor de final.

Observatie: Montajul este inca pe breadboard, deci arata destul de messy momentan. In etapa urmatoare il voi muta pe placa perforata, pentru un montaj mai stabil si mai curat

Software Design

Stadiul actual

Toate modulele sunt implementate si functionale. Aplicatia ruleaza complet in ambele moduri: Animatii (6 animatii: Rain, Plane Scan, Expand, Spiral, Firework, Sparkle) si Timer (countdown cu alerta sonora si vizuala la expirare).

Mediu de dezvoltare

Proiectul este dezvoltat in PlatformIO cu compilatorul avr-gcc, scris in C pur (avr-libc), cu acces direct

la registrele ATmega328P, fara framework Arduino. Ca model principal am folosit laboratoarele rezolvate, adaptand implementarile la hardware-ul specific proiectului. Acolo unde laboratoarele nu acopereau o functionalitate (ex. protocolul LCD peste I2C), am folosit resurse suplimentare: datasheet-ul HD44780, datasheet-ul ATmega328P si referinta Matiasus/HD44780_PCF8574 de pe GitHub.

Biblioteci folosite

Nu am folosit biblioteci externe. Singurele headere incluse sunt cele din avr-libc, conform indicatiilor laborantului:

- **avr/io.h** → definitiile registrelor si pinilor ATmega328P (DDRD, PORTB, OCR1A etc.)
- **avr/interrupt.h** → suport pentru ISR-uri si functiile sei()/cli()
- **util/delay.h** → _delay_ms() si _delay_us() pentru pulsurile de timing ale LCD-ului si beep-ul de pornire
- **stdint.h** → tipuri de date cu dimensiune fixa (uint8_t, uint16_t, uint32_t etc.)
- **stdlib.h** → rand()/srand() pentru generarea numerelor aleatoare in animatii

Functionalitati din laborator

Laborator	Functionalitate	Unde e folosita in proiect
Lab 2	Systicks (uptime_ms)	timer_get_ticks() - baza de timp globala
Lab 2	Pattern non-blocant (SYSTICKS_PASSED)	TICKS_PASSED - ADC la 50ms, LCD la 250ms, animatii la speed_ms
Lab 2	PCINT (ex2.c)	Detectia apasarii butoanelor prin ISR, fara polling
Lab 3	Timer1 CTC (Timer1_init_systicks)	Intrerupere la 1ms pentru systicks si multiplexarea cubului
Lab 3	Timer0 CTC	Generarea tonului buzzerului prin toggle hardware pe OC0A
Lab 4	ADC	Citirea potentiometrului si maparea la viteza animatiei
Lab 5	SPI (transfer bit cu bit, MSB first)	Bit-bang SPI pentru trimiterea datelor la shift registere
Lab 5	LCD HD44780 (mod 4-bit, nibble transfer)	Controlul LCD-ului prin I2C si adaptorul PCF8574
Lab 6	TWI/I2C	Comunicatia cu modulul LCD prin magistrala I2C

Scheletul proiectului

Proiectul este impartit in module independente, fiecare cu rolul lui:

- **timer.c** → systicks, ISR la 1ms, multiplexare cub la 125Hz, countdown
- **spi.c** → bit-bang SPI pentru shift registere (74HC595)
- **led_cube.c** → bufferul cubului (cube_state[]), afisarea unui strat

- **adc.c** → citirea potentiometrului, mapare la viteza cu histereza
- **buttons.c** → initializare PCINT, ISR-uri pe 3 vectori, debounce prin systicks
- **buzzer.c** → Timer0 CTC, toggle hardware pe OC0A (PD6)
- **i2c.c** → TWI master (start, write, read_ack, read_nack, stop)
- **lcd.c** → protocol HD44780 4-bit peste I2C prin PCF8574
- **animations.c** → 6 animatii non-blocante (Rain, Plane Scan, Expand, Spiral, Firework, Sparkle)
- **main.c** → FSM cu doua moduri, bucla principala non-blocanta

State machine

Am implementat un FSM simplu in main.c cu doua moduri:

- **MODE_ANIMATION** - ruleaza animatiile, BTN_UP/DOWN schimba animatia curenta, potentiometrul controleaza viteza
- **MODE_TIMER** - countdown configurat cu BTN_UP/DOWN in pasi de 10 secunde (minim 10s, maxim 99 minute si 50 secunde), pornit/oprit cu BTN_START_STOP

Tranzitia intre moduri se face prin BTN_MODE. La comutare resetez cubul, frame-ul local si opresc buzzerul. La revenirea in MODE_ANIMATION re-initializez animatiile cu un seed nou din ADC.

In MODE_TIMER am adaugat si o sub-stare de alerta (alert_active), activata la expirarea countdown-ului: LED-urile afiseaza un pattern aleator la fiecare 80ms si buzzerul suna continuu, pana la apasarea BTN_START_STOP.

Interactiunea dintre module

ISR-ul din timer.c se declanseaza la fiecare 1ms. Incrementeaza systicks, comuta stratul activ al cubului (citind cube_state[]) si decrementeaza countdown-ul in modul Timer. Restul modulelor nu stiu de ISR - pur si simplu scriu in cube_state[], iar ISR-ul afiseaza ce gaseste acolo.

Bucla principala din main.c este complet non-blocanta. La fiecare iteratie citesc timestamp-ul curent si execut fiecare task doar daca a trecut intervalul sau: ADC la 50ms, LCD la 250ms, animatii la viteza setata din potentiometru. Am folosit macroul TICKS_PASSED din Lab 2, care functioneaza corect si la overflow pe 32 biti datorita scaderii fara semn:

```
#define TICKS_PASSED(last, interval) \
    ((uint32_t)(systicks - (last)) >= (interval))
```

Animatiile sunt non-blocante: animations_update() verifica daca a trecut period_ms de la ultimul frame si returneaza imediat daca nu. Fiecare animatie are starea ei interna resetata la schimbarea animatiei curente.

BTN_UP si BTN_DOWN impart acelasi vector de intrerupere (PCINT0_vect), pentru ca sunt amandoua pe PORTB. Le disting in interiorul ISR-ului dupa starea fiecarui pin. PCINT se declanseaza si la eliberare, asa ca verific ca pinul e LOW inainte sa setez flag-ul, si ca au trecut cel putin 50ms de la ultima apasare valida - debounce fara condensatoare externe.

Validare: am testat fiecare modul separat inainte de integrare. Am aprins fiecare LED individual (0-63) ca sa verific maparea coloanelor, am testat LCD-ul cu un mesaj de confirmare, am verificat butoanele cu debounce am testat buzzerul la mai multe frecvente si am testat si timer-ul pentru a ma asigura ca este precis.

Calibrarea potentiometrului

Potentiometrul de 10k este conectat la A0 (ADC0) si citit cu referinta AVcc (5V) si prescaler 128. Mapez valoarea bruta (0-1023) la intervalul de viteza 50-500ms prin aritmetica pe intregi, fara float:

```
new_speed = ANIM_SPEED_MIN +
    (uint16_t)((uint32_t)(ANIM_SPEED_MAX - ANIM_SPEED_MIN)
               * (1023 - adc_val) / 1023);
```

Am inversat sensul intentionat: potentiometru la maxim ⇒ animatie rapida (50ms), potentiometru la minim ⇒ animatie lenta (500ms). Mi s-a parut mai intuitiv asa.

Pentru a evita tremuratul cauzat de zgomotul ADC, am aplicat o histereza de 10ms: actualizez viteza doar daca noua valoare difera cu mai mult de 10ms fata de ultima returnata.

La pornire, valoarea ADC a potentiometrului este folosita si ca seed pentru srand(), astfel incat secventa animatiilor aleatoare sa varieze in functie de pozitia potentiometrului.

Optimizari

Citire atomica a systicks - systicks are 32 biti, iar AVR citeste cate 8 biti odata. Daca ISR-ul modifica variabila la mijlocul citirii, obtin o valoare corupta. Dezactivez intreruperile pe durata citirii, salvand si restaurand SREG:

```
uint8_t sreg = SREG;
cli();
t = systicks;
SREG = sreg;
```

Evitarea ghosting-ului - inainte de a trimite datele unui nou strat prin SPI, sting toate straturile. Fara asta, LED-urile stratului precedent raman aprinse pe durata transferului si apar lumini parazite pe straturi gresite.

Actualizare LCD fara flash - am implementat lcd_print_line() sa scrie caracter cu caracter si sa completeze cu spatii pana la 16 caractere, in loc sa apelez lcd_clear(). Comanda clear dureaza 2ms si produce un flash vizibil la fiecare actualizare.

Evitarea printf - am implementat num_to_str() si strcopy() manual in main.c. Includerea printf trage cateva sute de bytes din biblioteca standard, risipiti din cei 32KB de flash disponibili

Elemente de noutate

Elementul principal de noutate fata de alte proiecte asemanatoare de pe net este combinarea a doua functionalitati practice distincte pe acelasi hardware: animatii 3D interactive si un timer countdown functional, comutabile in timp real prin acelasi buton.

In plus, animatia Firework sincronizeaza sunetele cu vizualul: racheta urca cu note ascendente (E4, A4, D5, G5), explozia e insotita de un sunet la 1200Hz, iar scanteile care se sting sunt insotite de note descrescatoare. Totul e realizat prin Timer0 CTC cu toggle hardware pe OC0A, fara ISR dedicat pentru sunet.

Viteza animatiilor este controlata continuu si in timp real din potentiometru, iar intreaga logica este non-blocanta: ISR-ul de multiplexare ruleaza independent la 125Hz, indiferent de ce face programul principal.

Rezultate Obtinute

Proiectul functioneaza conform planului initial. Cubul 4x4x4 afiseaza animatii prin multiplexare:

- **Rain** - picaturi cad aleator pe coloane diferite, de sus in jos
- **Plane Scan** - un strat intreg se aprinde si se deplaseaza de jos in sus
- **Expand** - LED-urile se extind din centrul cubului spre exterior, apoi se contracta
- **Spiral** - coloanele se aprind in spirala, din interior spre exterior, crescand si scazand
- **Firework** - o racheta urca layer cu layer, explodeaza si lasa scantei, sincronizat cu sunete ascendente din buzzer
- **Sparkle** - 8 LED-uri aleatoare clipesc haotic pe tot cubul

In modul Timer, cubul afiseaza vizual cat timp a ramas: LED-urile se sting treptat, proportional cu scurgerea timpului. La expirare, cubul clipeste haotic si buzzerul suna continuu pana la apasarea butonului de stop.

LCD-ul afiseaza in timp real numele animatiei curente si viteza in modul animatie, respectiv timpul ramas formatat MM:SS in modul timer.

Concluzii

Mi-a placut sa lucrez la acest proiect, mai ales partea de hardware - e ceva mai diferit fata de celelalte materii si am gasit-o mult mai interesanta decat ma asteptam. Sa lipesc LED-uri, sa trag fire si sa vad cubul prinzand forma a fost destul de tare

Sunt multumita de rezultatul final si il voi folosi pe Blinky cu siguranta in sesiune, ca sa imi tina companie la invatat. Dupa sesiune il voi da cadou nepotelului meu, asa ca sper sa ii placa si lui animatiile.

Demo

- Blinky party
- Demo Blinky LED Cube - YouTube

Download

Repo GitHub: Cod sursa - GitHub

Bibliografie/Resurse

Resurse hardware

- ATmega328P Datasheet
- Arduino Nano ATmega328P - Optimus Digital
- 74HC595 Datasheet
- HD44780 Datasheet
- PCF8574 Datasheet
- 2N2222 Datasheet
- Asamblare cub LED 4x4x4 - inspiratie si referinta

Resurse software

- AVR-libc Documentation
- PlatformIO Documentation
- HD44780 over PCF8574 - referinta implementare I2C LCD
- LED Cube 4x4x4 cu shift registers - referinta testare componente
- Laboratoare 2-6 PM

Export to PDF