Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică şi Microelectronică
Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 6.2
La disciplina “Internetul Lucrurilor”
Tema: Automate Finite - Semafor

A efectuat: st. gr. SI-211 Adrian Chihai
A verificat: Valentina Astafi

Chișinău – 2024

1 Definire Problemă

Sa se realizeze o aplicatie ce va implementa Automatele finite dupa cum urmeaza:

1. Proiectare Automat Finit aplicatie Semafor.

2 Descrierea funcțiilor programului

Funcțiile principale care sunt utilizate sunt: setup() - utilizată pentru inițializarea LED-urilor și a seriei și loop() - utilizată pentru a emite starea curentă, a întârzia un timp și a schimba starea pe baza stării curente. În figuria 2.1 este tabelul de tranziție pentru automatul finit, iar în figura 2.2 - diagrama acestuia.

Number	Current state	Next State for T
1	goS	waitS
2	waitS	goW
3	goW	waitW
4	waitW	goS

Fig. 2.1. Tabel de tranziție pentru semafor

Următoarea figură este diagrama de stări pentru semafor:

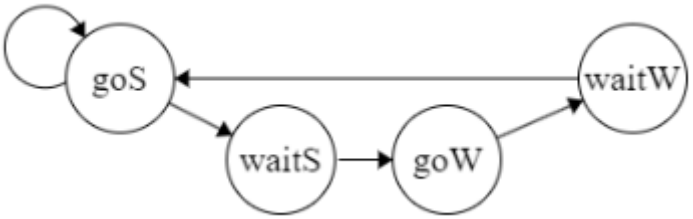

Fig. 2.2. Diagrama pentru semafor

Figura 2.3 reprezintă funcția setup din program.

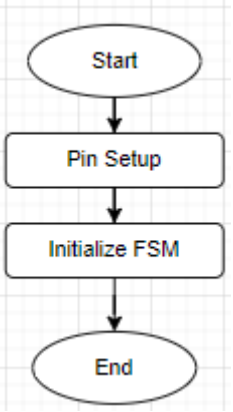

Fig. 2.3. Funcția *setup*

Figura 2.4 prezintă logica implementată în funcția de buclă a programului buton-LED.

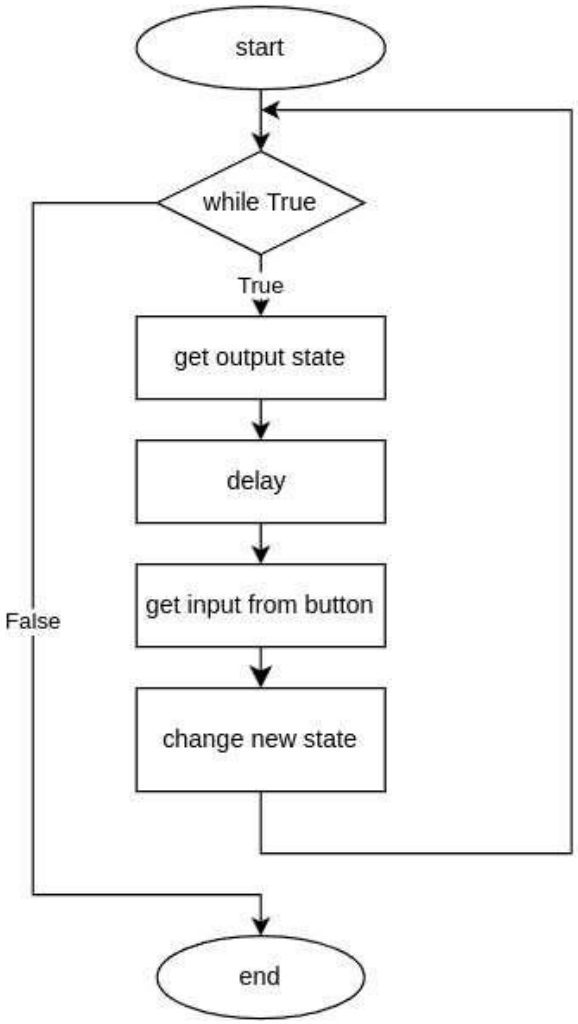

Fig. 2.4. Funcția de loop pentru semafor

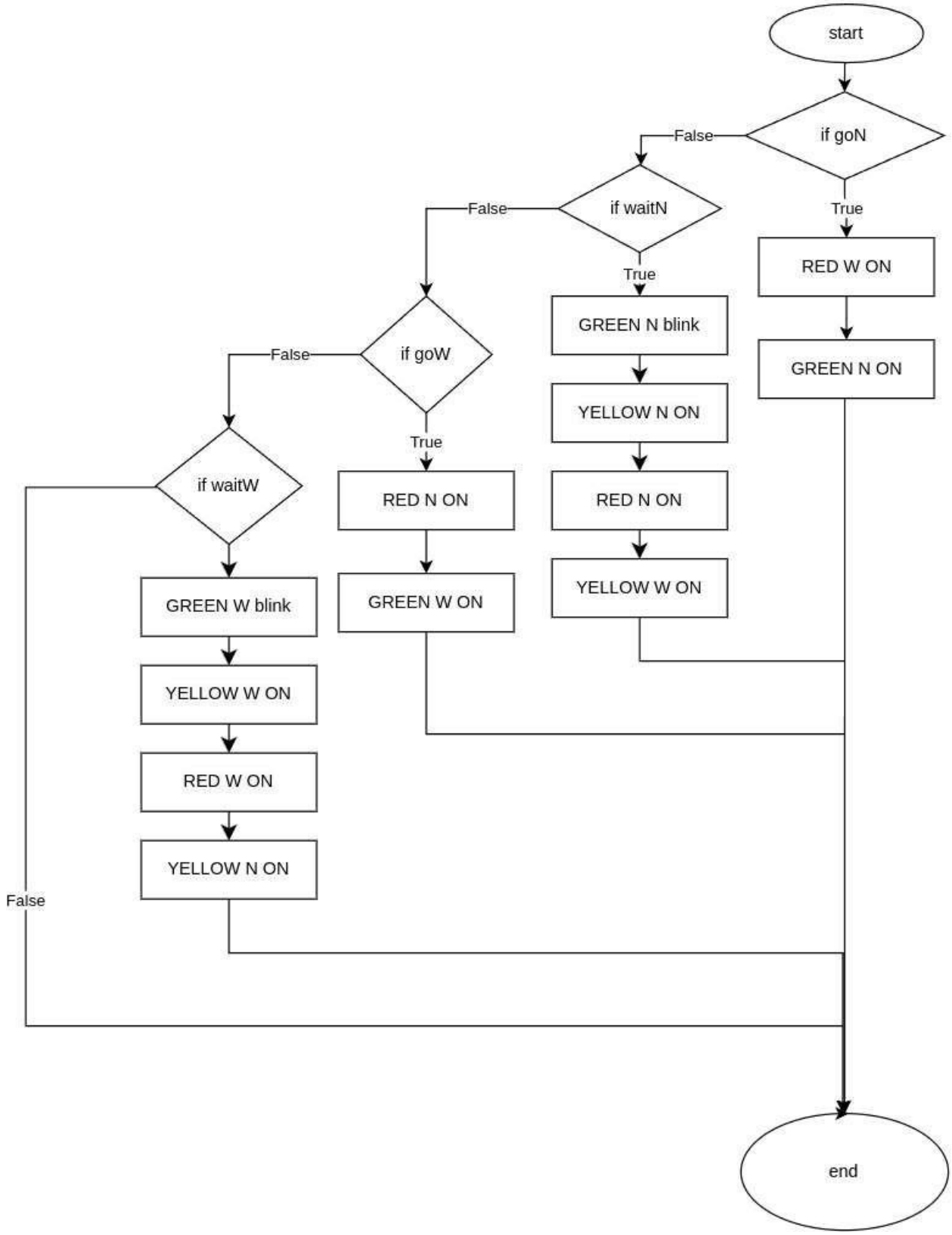

Fig. 2.5. *setOutput* function

3 Circuitul elaborat

Figura 2.6 reprezintă schema electrică asamblată a circuitului semafor :

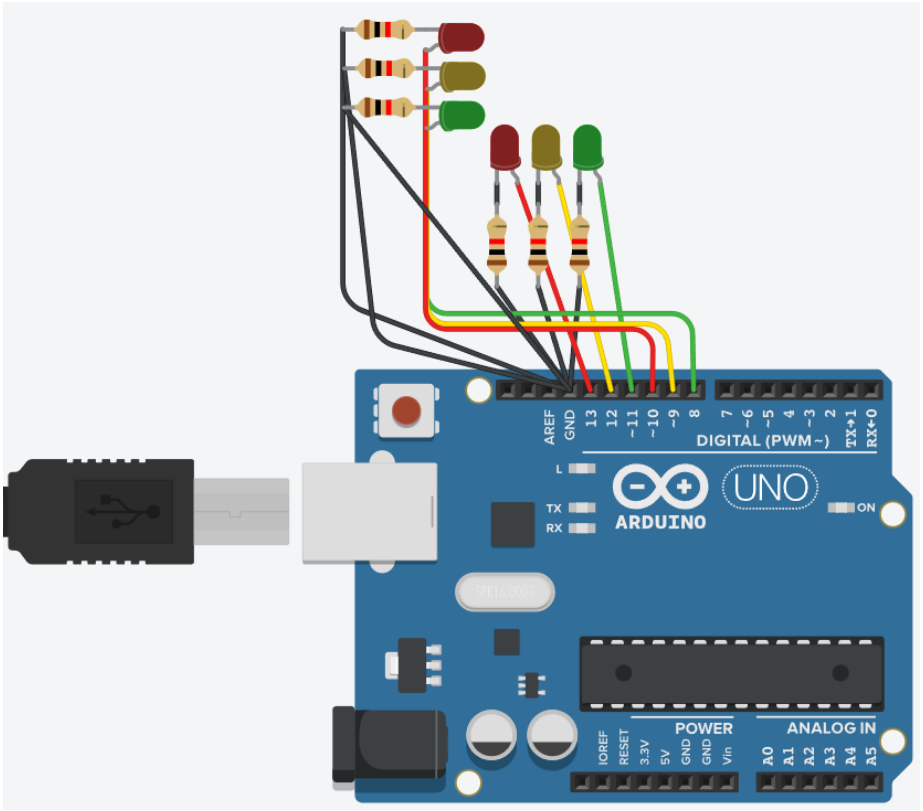

Fig. 2.6. Circuitul elaborat semafor

În figura următoare este rulat circuitul semafor.

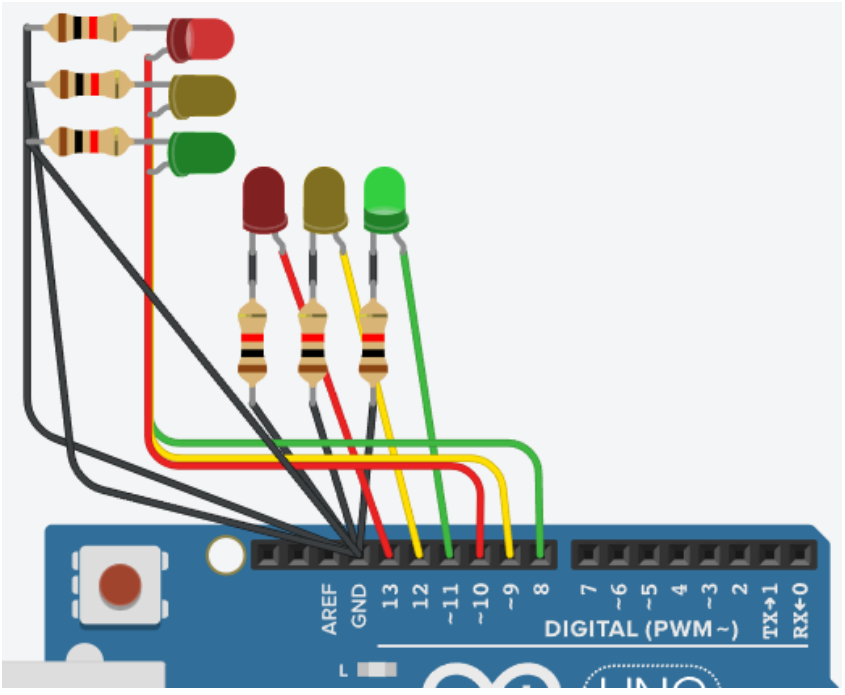

Fig. 2.7.Circuitul semafor rulat

Concluzie

În cadrul acestui laborator, am aplicat cunoștințele despre automatele finite, dezvoltând un sistem funcțional pentru controlul unui semafor. Am utilizat structuri de date pentru a gestiona tranzițiile între stări și a asigura un flux logic al aplicației. Prin integrarea LED-urilor și a butoanelor, am implementat tranziții între stările "Go", "Wait" și "Stop", simulând comportamentul real al unui semafor.

Implementarea funcției SetOutput a simplificat controlul LED-urilor, iar utilizarea tabelei de stări a făcut sistemul ușor de extins. Acest laborator a consolidat înțelegerea automatelor finite și aplicarea lor practică, oferind o bază pentru proiecte mai complexe în domeniul Internetului Lucrurilor.

Anexa 1

```
#define BAUD_RATE 9600

#define SOUTH_STATE HIGH
#define WEST_STATE HIGH

#define WEST_GREEN_PIN 8
#define WEST_YELLOW_PIN 9
#define WEST_RED_PIN 10

#define SOUTH_GREEN_PIN 11
#define SOUTH_YELLOW_PIN 12
#define SOUTH_RED_PIN 13

#define NONE_REQ 0b00
#define SOUTH_REQ 0b01
#define WEST_REQ 0b10
#define BOTH_REQ 0b11

#define GO_SOUTH_STATE 0
#define WAIT_SOUTH_STATE 1
#define GO_WEST_STATE 2
#define WAIT_WEST_STATE 3

#define GO_SOUTH_OUT 0b100001
#define WAIT_SOUTH_OUT 0b100010
#define GO_WEST_OUT 0b001100
#define WAIT_WEST_OUT 0b010100

#define GO_SOUTH_DELAY 3000
#define WAIT_SOUTH_DELAY 1000
#define GO_WEST_DELAY 3000
#define WAIT_WEST_DELAY 1000
#define RED_TO_GREEN_YELLOW_DELAY 1000
#define BLINK_DELAY 500
#define BLINK_COUNT 3

struct State {
  unsigned long Out; // 6-bit pattern to output
  unsigned long Time; // delay in 10ms units
  unsigned long Next[4]; // next state
};

typedef const struct State STyp;

STyp FSM[4]={
  {GO_SOUTH_OUT, GO_SOUTH_DELAY,{GO_SOUTH_STATE, WAIT_SOUTH_STATE, GO_SOUTH_STATE,
WAIT_SOUTH_STATE }},
```

```
  {WAIT_SOUTH_OUT, WAIT_SOUTH_DELAY, {GO_WEST_STATE, GO_WEST_STATE, GO_WEST_STATE,
GO_WEST_STATE }},
  {GO_WEST_OUT, GO_WEST_DELAY,{GO_WEST_STATE, GO_WEST_STATE, WAIT_WEST_STATE,
WAIT_WEST_STATE }},
  {WAIT_WEST_OUT, WAIT_WEST_DELAY, {GO_SOUTH_STATE, GO_SOUTH_STATE, GO_SOUTH_STATE,
GO_SOUTH_STATE }}};

int FSM_State = GO_SOUTH_STATE;

int GetInput(void) {
  int southButton = SOUTH_STATE;
  int westButton = WEST_STATE;
  if (southButton && westButton)
    return BOTH_REQ;
  else if (westButton)
    return WEST_REQ;
  else if (southButton)
    return SOUTH_REQ;
  else
    return NONE_REQ;
}

void SetOutput(int out) {
  int ledState;
  ledState = (out & (1 << 5)) ? HIGH : LOW;
  digitalWrite(WEST_RED_PIN, ledState);
  ledState = (out & (1 << 4)) ? HIGH : LOW;
  digitalWrite(WEST_YELLOW_PIN, ledState);
  ledState = (out & (1 << 3)) ? HIGH : LOW;
  digitalWrite(WEST_GREEN_PIN, ledState);
  ledState = (out & (1 << 2)) ? HIGH : LOW;
  digitalWrite(SOUTH_RED_PIN, ledState);
  ledState = (out & (1 << 1)) ? HIGH : LOW;
  digitalWrite(SOUTH_YELLOW_PIN, ledState);
  ledState = (out & (1 << 0)) ? HIGH : LOW;
  digitalWrite(SOUTH_GREEN_PIN, ledState);
}

void setup() {
  Serial.begin(BAUD_RATE);
  pinMode(WEST_GREEN_PIN, OUTPUT);
  pinMode(WEST_YELLOW_PIN, OUTPUT);
  pinMode(WEST_RED_PIN, OUTPUT);

  pinMode(SOUTH_GREEN_PIN, OUTPUT);
  pinMode(SOUTH_YELLOW_PIN, OUTPUT);
  pinMode(SOUTH_RED_PIN, OUTPUT);

  FSM_State = GO_SOUTH_STATE;
}
```

```
void loop() {
  int FSM_Output = FSM[FSM_State].Out;
  SetOutput(FSM_Output);
  delay(FSM[FSM_State].Time);

  if (FSM_State == WAIT_SOUTH_STATE || FSM_State == WAIT_WEST_STATE) {
    digitalWrite(WEST_YELLOW_PIN, HIGH);
    digitalWrite(SOUTH_YELLOW_PIN, HIGH);
    delay(RED_TO_GREEN_YELLOW_DELAY);
    digitalWrite(WEST_YELLOW_PIN, LOW);
    digitalWrite(SOUTH_YELLOW_PIN, LOW);
  }

  if ((FSM_State == GO_WEST_STATE && (FSM_Output & (1 << 3))) ||
      (FSM_State == GO_SOUTH_STATE && (FSM_Output & (1 << 0)))) {
    for (int i = 0; i < BLINK_COUNT; i++) {
      digitalWrite(FSM_State == GO_WEST_STATE ? WEST_GREEN_PIN : SOUTH_GREEN_PIN,
HIGH);
      delay(BLINK_DELAY);
      digitalWrite(FSM_State == GO_WEST_STATE ? WEST_GREEN_PIN : SOUTH_GREEN_PIN, LOW);
      delay(BLINK_DELAY);
    }
  }

  int FSM_Inputs = GetInput();
  FSM_State = FSM[FSM_State].Next[FSM_Inputs];
}
```