

Ministerul Educației, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator Nr.6
Disciplina: IoT
Tema: Automate Finite - Button-LED

A efectuat:

st.gr.TI-212,
Muntean Mihai

A verificat :

asist. univ.
Lupan Cristian

Chișinău 2024

Definirea problemei:

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

1. Proiectare Automat Finit aplicatie Button-Led.
2. Proiectare Automat Finit aplicatie Semafor.

Obiective:

1. Crearea diagramelor Automatelor finite Button-Led / Semafor;
2. Explicarea tabelor de tranziții Button-Led / Semafor;
3. Demonstrarea probelor de implimentare fizică.

INTRODUCERE

Automatele finite sunt modele matematice utilizate pentru a reprezenta și a gestiona comportamentele sistemelor discrete, fiind esențiale în domeniul informaticii și al ingineriei electronice. Aceste modele sunt caracterizate printr-un set finit de stări, tranziții între aceste stări și o logică bine definită care determină cum se schimbă starea în funcție de evenimente sau condiții. În contextul microcontrolerelor (MCU), automatele finite sunt utilizate pentru a implementa funcționalități precum controlul de stări al LED-urilor, butoanelor, semafoarelor sau altor dispozitive periferice. Ele oferă o metodologie structurată pentru dezvoltarea aplicațiilor care implică secvențe logice, facilitând scrierea unui cod clar, ușor de testat și de întreținut.

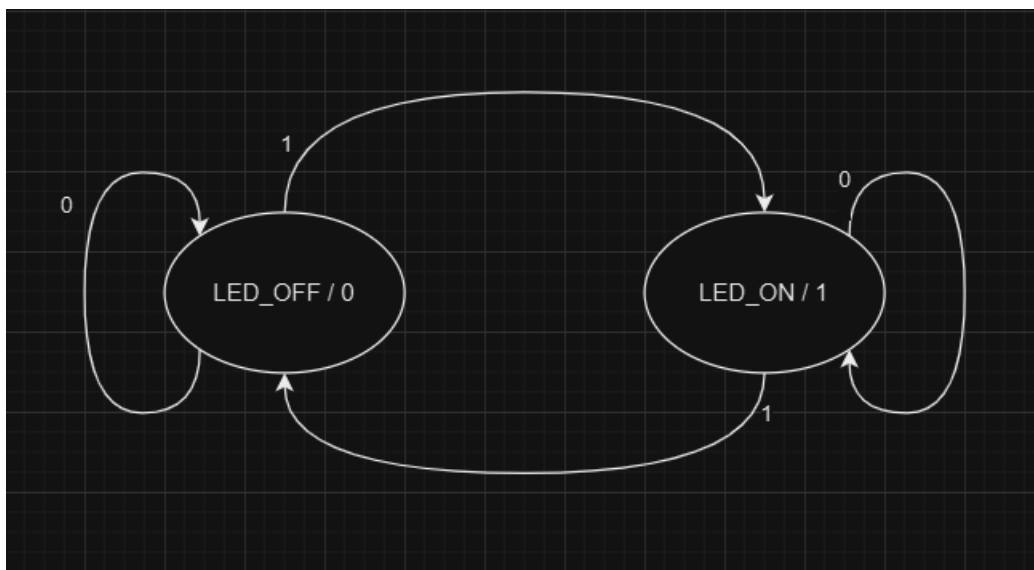


Figura 1 – Diagrama automatului finit Button-Led

Tabelul 1 – Tabel de tranziție Button-Led

Name	Out	Delay	In = 0	In = 1
LED_OFF	0	100 ms	LED_OFF	LED_ON
LED_ON	1	100ms	LED_ON	LED_OFF

Din câte se observă atât în diagramă cât și în tabelul de tranziție a acestui automat finit, pentru Button-Led avem definite 2 stări: când led-ul este aprins și led-ul stins. Tranziția între aceste stări se realizează la apăsarea unui buton. Deci când starea butonului este 1, atunci automatul finit reacționează cu modificarea stării led-ului, și respectiv când butonul nu este apăsat (0) led-ul își păstrează starea. În continuare este prezentat codul sursă pentru acest automat finit:

Cod sursă:

```
#define LED_PIN 13
#define BUTTON_PIN 8
#define LED_ON_STATE HIGH
#define LED_OFF_STATE LOW

typedef struct{
    unsigned long Out;
    unsigned long Time;
    unsigned long Next[2];
} SType;

SType FSM[2] = {
    {0, 10, { LED_OFF_STATE, LED_ON_STATE }},
    {1, 10, { LED_ON_STATE, LED_OFF_STATE }}
};

int FSM_State = LED_OFF_STATE;

void setup() {
    Serial.begin(115200);
    pinMode(LED_PIN, OUTPUT);
    pinMode(BUTTON_PIN, INPUT);

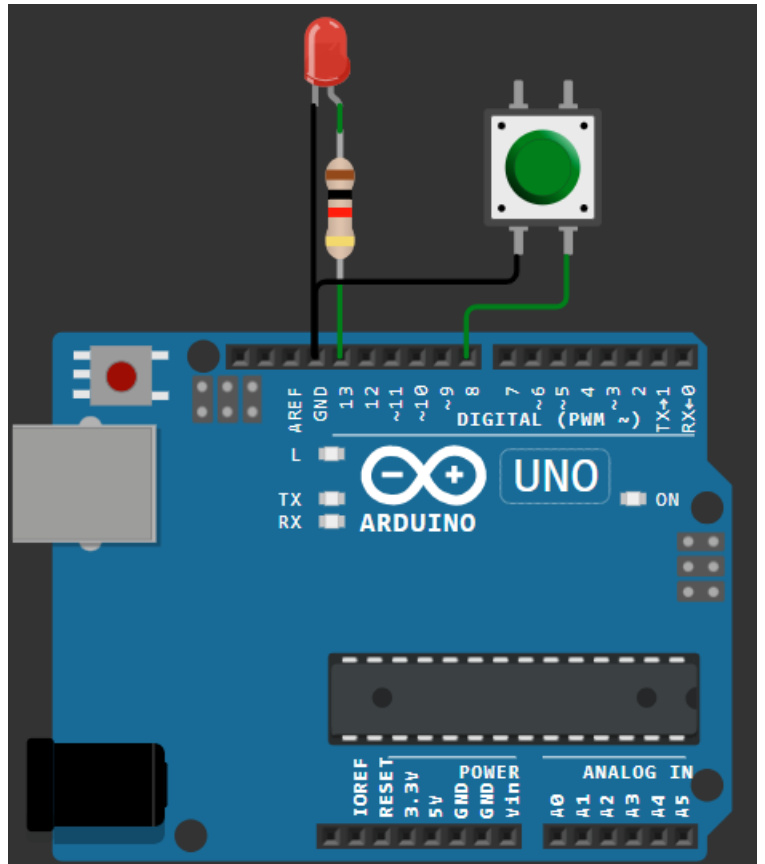
    FSM_State = LED_OFF_STATE;
}

void loop() {
    int output = FSM[FSM_State].Out;
    digitalWrite(LED_PIN, output);

    delay(FSM[FSM_State].Time * 10);

    int input = digitalRead(BUTTON_PIN);
    FSM_State = FSM[FSM_State].Next[input];

    FSM_State == LED_OFF_STATE ?
        printf("Led is OFF\n") :
        printf("Led is ON\n");
}
```



Materiale necesare:

- Placă pe bază de arduino;
- Led;
- Buton;
- Rezistor;
- Cabluri pentru conectare.

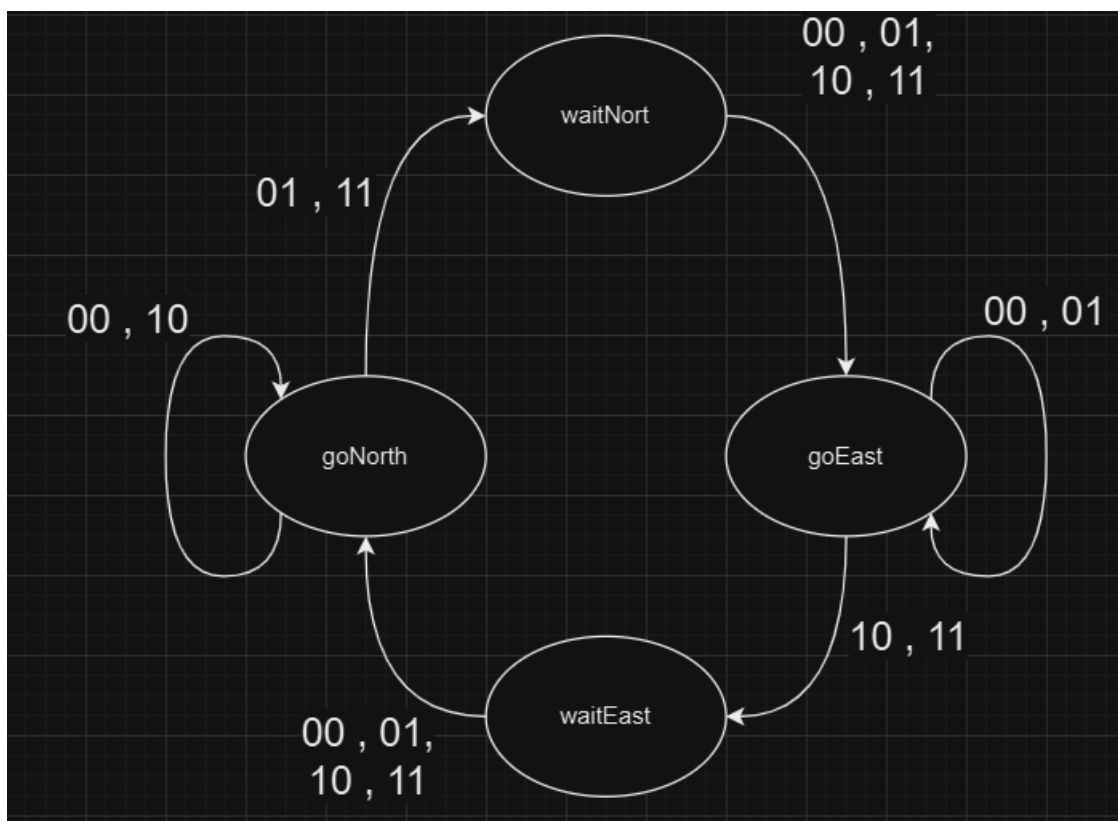


Figura 2 - Diagrama automatului finit Semafor

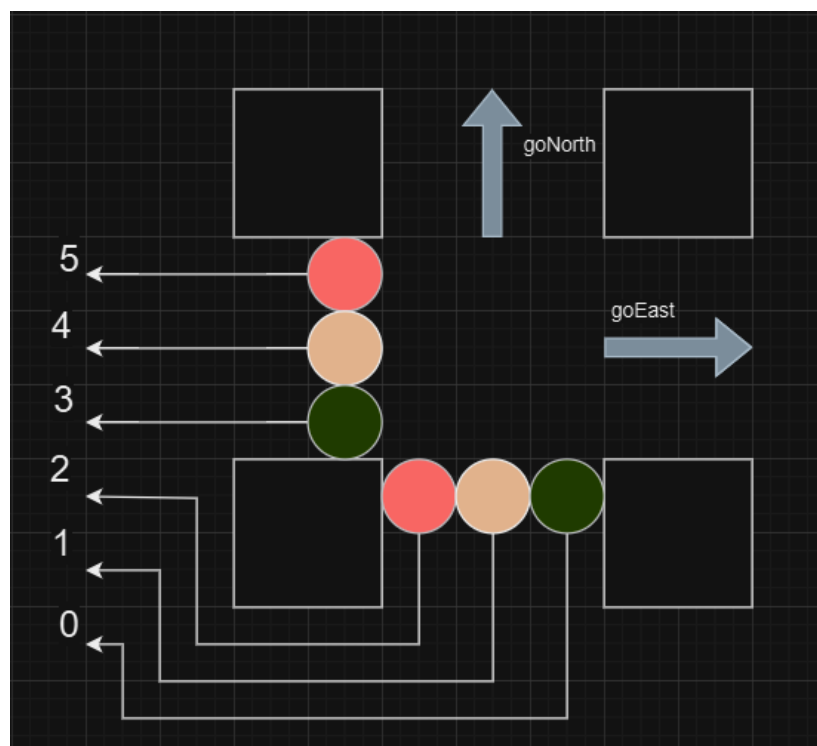


Figura 3 – Exemplificarea cazului

Tabelul 2 – Valoarea biților care descriu starea automatului finit

Bit position	5	4	3	2	1	0
goNorth	1	0	0	0	0	1
waitNorth	1	0	0	0	1	0
goEast	0	0	1	1	0	0
waitEast	0	1	0	1	0	0

Tabelul 3 - Tabel de tranziție Semafor

Name	Out	Delay	In = 0 (00)	In = 1 (01)	In = 2 (10)	In = 3 (11)
goNorth	100 001	300	goNorth	waitNorth	goNorth	waitNorth
waitNorth	100 010	50	goEast	goEast	goEast	goEast
goEast	001 100	300	goEast	goEast	waitNorth	waitNorth
waitEast	010 100	50	goNorth	goNorth	goNorth	goNorth

Pentru automatul finit Semafor avem definite 4 stări. Acesta funcționează pe baza valorii de ieșire plasate pe 6 biți. După cum se observă și în figura 3, fiecare bit descrie starea unui led al semaforului, respectiv primii 3 biți aparțin unui semafor ce dirijează pe orizontală (So), următorii 3 biți semaforului care dă accesul deplasării verticale (Sv).

Stările definite de automatul finit sunt:

1. Deplasarea pe verticală (goNorth / 100 001): în acest caz culoarea So este roșie, iar Sv verde;
2. Oprirea pe verticală (waitNorth / 100010): culoarea So rămâne încă roșie, dar Sv aprinde galben;
3. Deplasare pe orizontală (goEast / 001100): So devine verde, iar Sv este roșu;
4. Oprire pe orizontală (waitEast / 010100): culoarea So devine deja galben, iar Sv încă rămâne roșu.

Tranziția între stările semafoarelor se realizează în baza a două butoane.

1. În cazul în care ne aflăm într-o stare de deplasare pe una din benzi, la apăsarea butonului ce aparține benzii opuse are loc tranziția către starea următoare (se oprește banda curentă);
2. În cazul în care ne aflăm iarăși într-o stare de deplasare, dar nu a fost apăsat nici un buton sau butonul benzii curente, starea semafoarelor rămâne neschimbată;
3. În cazul în care starea descrie o repauză, la expirarea timpului prestabilit, va fi urmată următoarea stare nemijlocit.

Cod sursă:

```
#define NORTH_PIN 9
#define EAST_PIN 2
#define EAST_RED_PIN 3
#define EAST_YELLOW_PIN 4
#define EAST_GREEN_PIN 5
#define NORTH_RED_PIN 6
#define NORTH_YELLOW_PIN 7
#define NORTH_GREEN_PIN 8
#define goN 0    // 0b00
#define waitN 1 // 0b01
#define goE 2    // 0b10
#define waitE 3 // 0b11

int GetInput();
void SetOutput(int out);

typedef struct State {
    unsigned long Out;
    unsigned long Time;
    unsigned long Next[4]; // next state for inputs 0,1,2,3
} SType;

SType FSM[4] {
    {0b100001, 300, { goN, waitN, goN, waitN }},
    {0b010010, 50, { goE, goE, goE, goE }},
    {0b001100, 300, { goE, goE, waitE, waitE }},
    {0b010010, 50, { goN, goN, goN, goN }}
};

int FSM_State = goN;

void setup() {
    Serial.begin(115200);
    // Init Button
    pinMode(NORTH_PIN, INPUT);
    pinMode(EAST_PIN, INPUT);

    // Init LED
    pinMode(EAST_RED_PIN, OUTPUT);
    pinMode(EAST_YELLOW_PIN, OUTPUT);
    pinMode(EAST_GREEN_PIN, OUTPUT);
    pinMode(NORTH_RED_PIN, OUTPUT);
    pinMode(NORTH_YELLOW_PIN, OUTPUT);
    pinMode(NORTH_GREEN_PIN, OUTPUT);

    // Init Initial State
    FSM_State = goN;
}

// The loop function is called in an endless loop
```

```

void loop() {
    // 1. Output Based on current state
    int output = FSM[FSM_State].Out;
    SetOutput(output);
    Serial.print("output: ");
    Serial.println(output);

    // 2. wait for time relevant to state
    delay(FSM[FSM_State].Time * 10);

    // 3. Get Input
    int input = GetInput();
    Serial.print("input: ");
    Serial.println(input);

    // 4. Change state based on input and current state
    FSM_State = FSM[FSM_State].Next[input];
}

int GetInput() {
    int northButton = digitalRead(NORTH_PIN);
    int eastButton = digitalRead(EAST_PIN);

    if (northButton && eastButton)
        return 0b11;
    else if (northButton)
        return 0b10;
    else if (eastButton)
        return 0b01;
    else
        return 0b00;
}

void SetOutput(int out) {
    int ledState;
    ledState = (out & (1 << 5)) ? HIGH : LOW;
    digitalWrite(EAST_RED_PIN, ledState);
    ledState = (out & (1 << 4)) ? HIGH : LOW;
    digitalWrite(EAST_YELLOW_PIN, ledState);
    ledState = (out & (1 << 3)) ? HIGH : LOW;
    digitalWrite(EAST_GREEN_PIN, ledState);
    ledState = (out & (1 << 2)) ? HIGH : LOW;
    digitalWrite(NORTH_RED_PIN, ledState);
    ledState = (out & (1 << 1)) ? HIGH : LOW;
    digitalWrite(NORTH_YELLOW_PIN, ledState);
    ledState = (out & (1 << 0)) ? HIGH : LOW;
    digitalWrite(NORTH_GREEN_PIN, ledState);
}

```

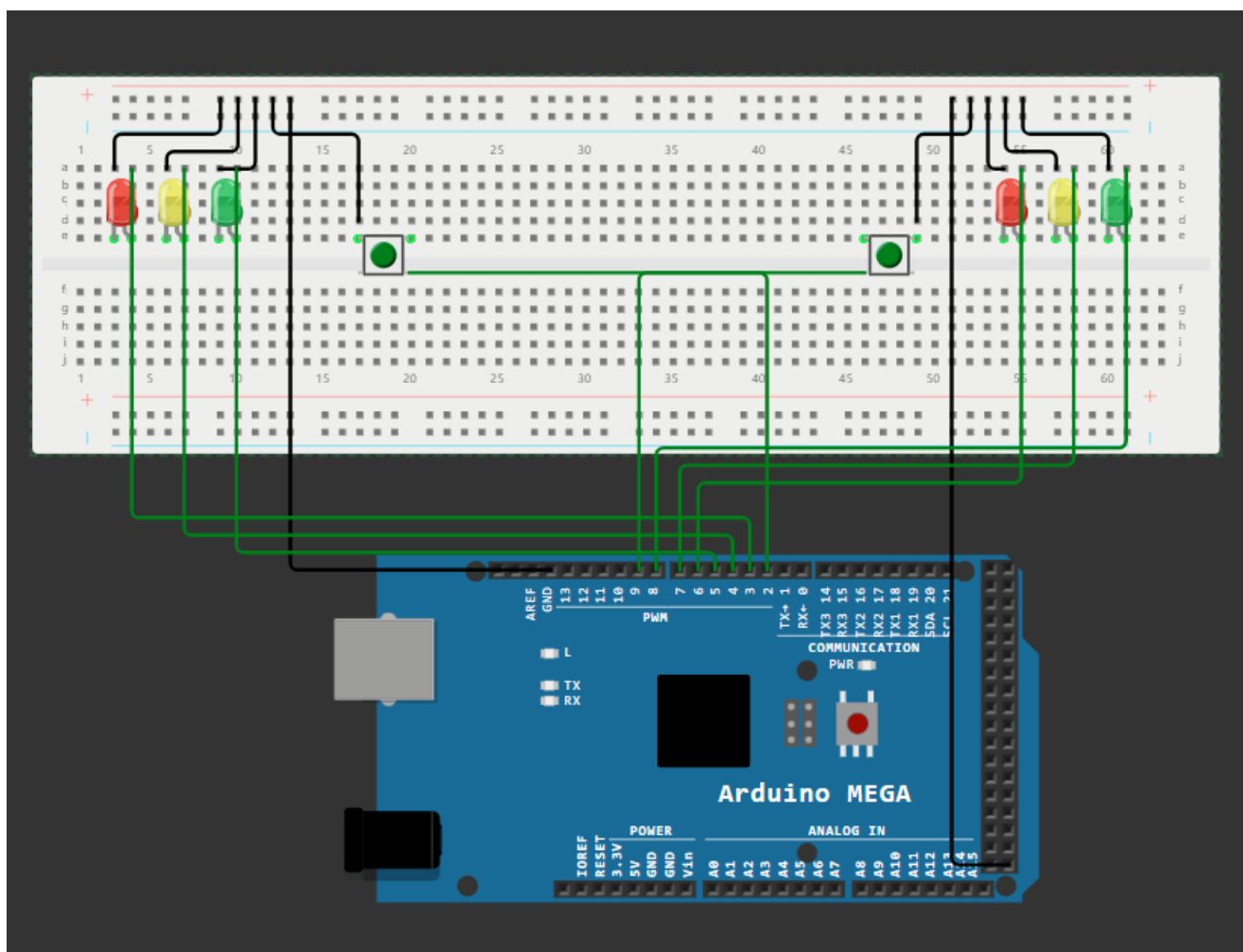



Figura 4 Asamblarea circuitului virtual pentru semafoare

CONCLUZIE

Automatele finite joacă un rol fundamental în dezvoltarea de aplicații eficiente pentru microcontrolere, oferind o metodologie robustă și clară pentru gestionarea comportamentelor secvențiale. Prin organizarea logicii aplicațiilor sub formă de stări și tranziții bine definite, acestea permit implementarea unor sisteme fiabile, scalabile și ușor de întreținut.

În contextul utilizării lor în microcontrolere (MCU), automatele finite se evidențiază prin:

1. Sunt ideale pentru mediile cu resurse limitate, cum ar fi cele întâlnite în sistemele embedded, datorită cerințelor reduse de memorie și procesare.
2. Pot fi utilizate într-o gamă variată de aplicații, de la controlul periferic al hardware-ului până la gestionarea proceselor complexe, cum ar fi protocoalele de comunicare.
3. Asigură un comportament previzibil al sistemului, esențial pentru aplicații critice.

Astfel, automatele finite nu doar că simplifică dezvoltarea aplicațiilor pentru MCU, ci și asigură un echilibru optim între performanță, consum de resurse și complexitate. Această abordare reprezintă o soluție practică și eficientă pentru construcția de sisteme embedded de înaltă calitate.

BIBLIOGRAFII

1. Resursa electronică: https://docs.wokwi.com/?utm_source=wokwi – Regim de acces;
2. Resursa electronică: <https://else.fcim.utm.md/course/view.php?id=343> – Regim de acces;
3. Resursa electronică: <https://forum.arduino.cc/t/serial-print-and-printf/146256/14> - Regim de acces;
4. Proiectul pe GitHub, Resursa electronică: [https://github.com/MunMihai/Anul4/tree/8c7af8c43b8c728ff28e1ee6c75a63f997659015/Semestrul_7/Internetul_Lucrurilor%20\(IoT\)/Laborator/Laborator6](https://github.com/MunMihai/Anul4/tree/8c7af8c43b8c728ff28e1ee6c75a63f997659015/Semestrul_7/Internetul_Lucrurilor%20(IoT)/Laborator/Laborator6) - Regim de acces;

Anexa 1

Fișierul stdinout.h

```
#ifndef _STDINOUT_H
#define _STDINOUT_H

// no need to make an instance of this yourself
class initializeSTDINOUT
{
    static size_t initnum;
public:
    // Constructor
    initializeSTDINOUT();
};

// Call the constructor in each compiled file this header is included in
// static means the names won't collide
static initializeSTDINOUT initializeSTDINOUT_obj;

#endif
```

Fișierul stdinout.cpp

```
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include <stdio.h>
#include "stdinout.h"

// Function that printf and related will use to print
static int serial_putchar(char c, FILE *f)
{
    if (c == '\n') {
        serial_putchar('\r', f);
    }

    return Serial.write(c) == 1 ? 0 : 1;
}

// Function that scanf and related will use to read
static int serial_getchar(FILE *)
{
    // Wait until character is available
    while (Serial.available() <= 0) { ; }

    return Serial.read();
}

static FILE serial_stdinout;

static void setup_stdin_stdout()
```

```

{
    // Set up stdout and stdin
    fdev_setup_stream(&serial_stdinout, serial_putchar, serial_getchar, _FDEV_SETUP_RW);
    stdout = &serial_stdinout;
    stdin  = &serial_stdinout;
    stderr = &serial_stdinout;
}

// Initialize the static variable to 0
size_t initializeSTDINOUT::initnum = 0;

// Constructor that calls the function to set up stdin and stdout
initializeSTDINOUT::initializeSTDINOUT()
{
    if (initnum++ == 0) {
        setup_stdin_stdout();
    }
}

```

Anexa 2

Sarcina 1: diagram.json

```

{
    "version": 1,
    "author": "Mihai Muntean",
    "editor": "wokwi",
    "parts": [
        { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
        {
            "type": "wokwi-led",
            "id": "led1",
            "top": -128.4,
            "left": 99.8,
            "attrs": { "color": "red" }
        },
        {
            "type": "wokwi-resistor",
            "id": "r1",
            "top": -52.8,
            "left": 95.45,
            "rotate": 90,
            "attrs": { "value": "1000" }
        },
        {
            "type": "wokwi-pushbutton",
            "id": "btn1",
            "top": -87,
            "left": 166.2,
            "rotate": 90,
            "attrs": { "color": "green", "bounce": "1" }
        }
    ]
}

```

```

    }
  ],
  "connections": [
    [ "led1:C", "uno:GND.1", "black", [ "v0" ] ],
    [ "led1:A", "r1:1", "green", [ "v0" ] ],
    [ "r1:2", "uno:13", "green", [ "v0" ] ],
    [ "btn1:2.r", "uno:GND.1", "black", [ "v9.6", "h-80.5" ] ],
    [ "btn1:1.r", "uno:8", "green", [ "v19.2", "h-42" ] ]
  ],
  "dependencies": {}
}

```

Sarcina 2: diagram.json

```

{
  "version": 1,
  "author": "Mihai Muntean",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-breadboard", "id": "bb1", "top": -99, "left": -93.2, "attrs": {} },
    { "type": "wokwi-arduino-mega", "id": "mega", "top": 163.8, "left": 73.2, "attrs": {} },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -51.6,
      "left": -63.4,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": -51.6,
      "left": -34.6,
      "attrs": { "color": "yellow" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": -51.6,
      "left": -5.8,
      "attrs": { "color": "limegreen" }
    },
    {
      "type": "wokwi-led",
      "id": "led4",
      "top": -51.6,
      "left": 483.8,
      "attrs": { "color": "limegreen" }
    },
    {
      "type": "wokwi-led",
      "id": "led5",

```

```

    "top": -51.6,
    "left": 455,
    "attrs": { "color": "yellow" }
  },
  {
    "type": "wokwi-led",
    "id": "led6",
    "top": -51.6,
    "left": 426.2,
    "attrs": { "color": "red" }
  },
  {
    "type": "wokwi-pushbutton-6mm",
    "id": "btn1",
    "top": -11.8,
    "left": 86.4,
    "attrs": { "color": "green" }
  },
  {
    "type": "wokwi-pushbutton-6mm",
    "id": "btn2",
    "top": -11.8,
    "left": 364.8,
    "attrs": { "color": "green" }
  }
],
"connections": [
  [ "bb1:17t.d", "bb1:tp.9", "black", [ "v-48", "h-47.2" ] ],
  [ "bb1:6t.a", "bb1:tp.7", "black", [ "v-9.6", "h39.2" ] ],
  [ "bb1:3t.a", "bb1:tp.6", "black", [ "v-19.2", "h58.4" ] ],
  [ "bb1:tp.8", "bb1:9t.a", "black", [ "v0" ] ],
  [ "bb1:tp.10", "mega:GND.1", "black", [ "v0" ] ],
  [ "bb1:tp.43", "bb1:54t.a", "black", [ "v0" ] ],
  [ "bb1:tp.44", "bb1:57t.a", "black", [ "v29.1", "h28" ] ],
  [ "bb1:tp.45", "bb1:60t.a", "black", [ "v19.5", "h47.2" ] ],
  [ "mega:GND.5", "bb1:tp.41", "black", [ "h-31.4", "v-421.75" ] ],
  [ "mega:3", "bb1:4t.a", "green", [ "v-28.8", "h-340.1" ] ],
  [ "mega:4", "bb1:7t.a", "green", [ "v-19.2", "h-301.8" ] ],
  [ "mega:5", "bb1:10t.a", "green", [ "v-9.6", "h-263.5" ] ],
  [ "mega:6", "bb1:55t.a", "green", [ "v-48", "h178" ] ],
  [ "mega:7", "bb1:58t.a", "green", [ "v-57.6", "h216.8" ] ],
  [ "mega:8", "bb1:61t.a", "green", [ "v-67.2", "h258.6" ] ],
  [ "bb1:49t.d", "bb1:tp.42", "black", [ "v-48", "h29.6" ] ],
  [ "btn1:2.r", "mega:2", "green", [ "h0" ] ],
  [ "led1:A", "bb1:4t.e", "", [ "$bb" ] ],
  [ "led1:C", "bb1:3t.e", "", [ "$bb" ] ],
  [ "led6:A", "bb1:55t.e", "", [ "$bb" ] ],
  [ "led6:C", "bb1:54t.e", "", [ "$bb" ] ],
  [ "led5:A", "bb1:58t.e", "", [ "$bb" ] ],
  [ "led5:C", "bb1:57t.e", "", [ "$bb" ] ],
  [ "led4:A", "bb1:61t.e", "", [ "$bb" ] ],
  [ "led4:C", "bb1:60t.e", "", [ "$bb" ] ],

```

```

[ "led2:A", "bb1:7t.e", "", [ "$bb" ] ],
[ "led2:C", "bb1:6t.e", "", [ "$bb" ] ],
[ "led3:A", "bb1:10t.e", "", [ "$bb" ] ],
[ "led3:C", "bb1:9t.e", "", [ "$bb" ] ],
[ "btn1:1.l", "bb1:17t.e", "", [ "$bb" ] ],
[ "btn1:1.r", "bb1:20t.e", "", [ "$bb" ] ],
[ "btn2:1.l", "bb1:46t.e", "", [ "$bb" ] ],
[ "btn2:1.r", "bb1:49t.e", "", [ "$bb" ] ],
[ "btn2:2.l", "mega:9", "green", [ "h0" ] ]
],
"dependencies": {}
}

```