**TECHNICAL UNIVERSITY OF MOLDOVA**
**FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS**
**SOFTWARE ENGINEERING AND AUTOMATION DEPARTMENT**

# Report

## Laboratory work nr. 6

**Theme: Finite Automata**

**Student**: **Furdui Alexandru, FAF-192**
**Verified by: Moraru Dumitru, mag. univ. lecturer**

**Chișinău, 2022**

**1. Task of the laboratory work:**

To create an application that will implement Finite Automata as follows:

1. Finite Automata – Button-Led app
2. Finite Automata – Semaphore App

**2. The progress of the work:**

2.1 Bloc diagram of program/s

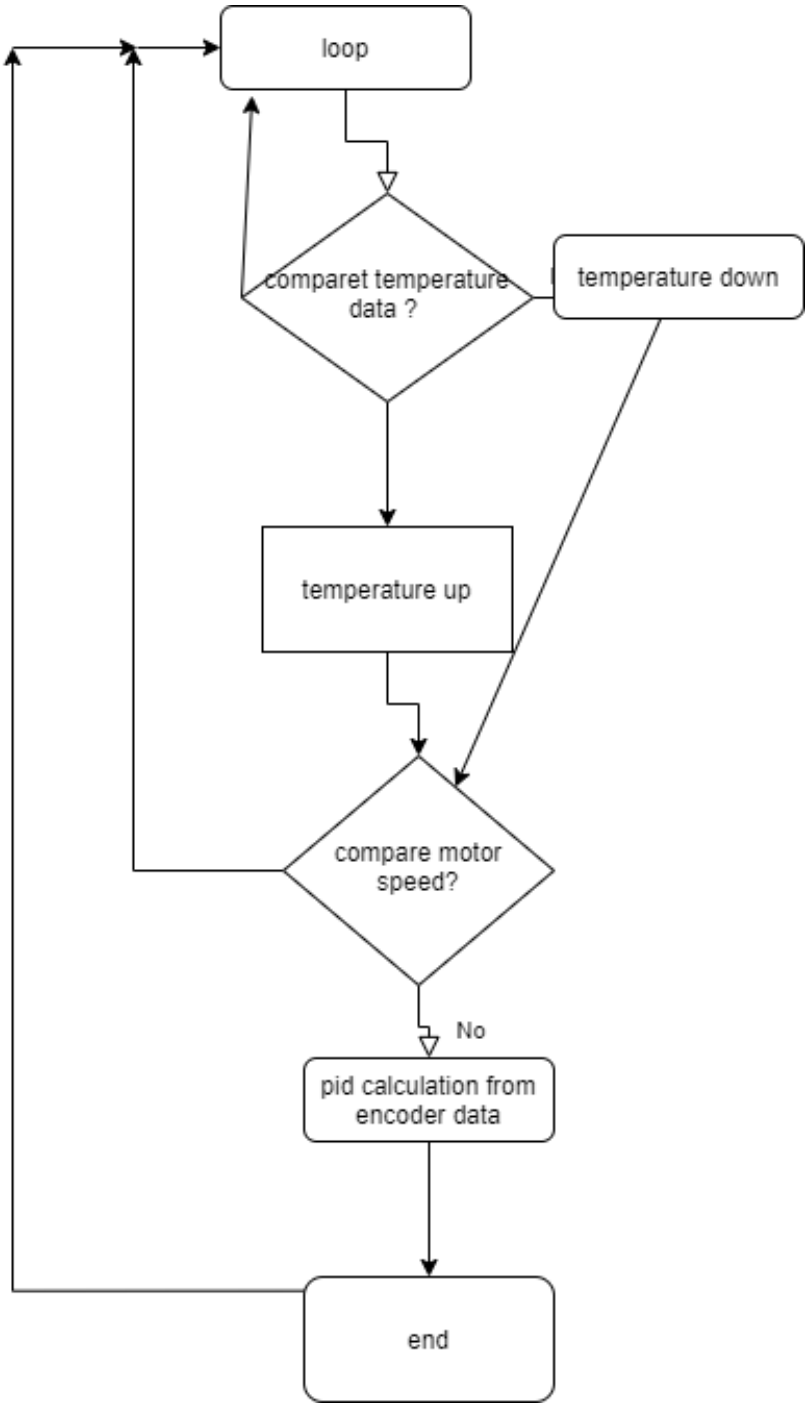
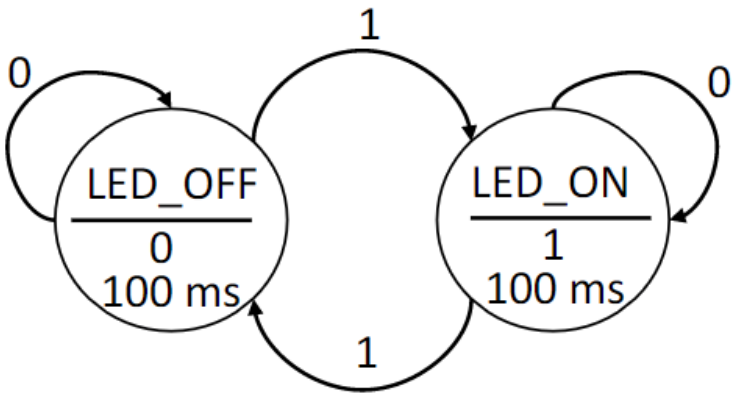



**Figure 1** Bloc diagram for task

2.2 Description of the main tables

a)

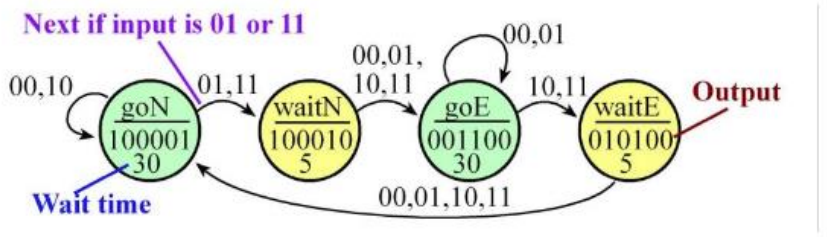



This is the state machine of part a of laboratory work. The system start is state LED_OFF the id of this state is 0 and the time when this state is active is 100 ms after this system listening to input from button. If button is pressed then the input is 1 if is not pressed then the input is 0. System receive input and if is 0 then system call again the same state where it was so it begins a self loop of that state till the button is pressed if button is pressed the is switched to another state that is LED_ON.

| Num | Name | StateId | Delay | Ln=0 | Ln=1 |
|---|---|---|---|---|---|
| 0 | LED_OFF | 0 | 100 ms | LED_OFF | LED_ON |
| 1 | LED_ON | 1 | 100 ms | LED_ON | LED_OFF |

This is how we implement that state machine so we create a structure for every state. StateIdd from table represents id of state delay, is the time that this state is active after that system will change the state depend of the input from button and we put that time to be 100 ms and ln represent an array that we use for switch states so first element of state represent the current state so if button is not pressed the input is 0 then is called the first element of array than is current state and his number is also 0 so is easier to call the state from array and the second element of array is nest state LED_ON.

b)




This is state machine of part b of laboratory work. We have for state goN, waitN, goE and waitE. This state are used for creating a semafor that help to manage traffic from two roads that intersects.Id of state goN is 100001, id of state waitN is 100010, id of state goE is 001100 and id of state waitE is 010100. This id helps us to better understand what state is now and every state have a time when this state is active so for goN and goE is 300 ms and for waitN and waitE is 50 ms. After this time system reveive input and the based on it changing states. System start is state goN so if system receive input 01 or 11 then system is changing to state waitN but if it receive 00 or 10 that is stay in the same state.If in state waitN system receive any type of input then is changing to next state that is goE.If in state goE system receive input 10 and 11 then is changing to waitE but if it receiving state 00 or 01 the stay in the same state. In state waitE if system receive any type of input is changing to goN state.

| Num | Name | Out | Delay | In = 0 | In = 1 | In = 2 | In = 3 |
|---|---|---|---|---|---|---|---|
| 0 | goN | 100 001 | 30 s | goN | waitN | goN | waitN |
| 1 | waitN | 100 010 | 5 s | goE | goE | goE | goE |
| 2 | goE | 001 100 | 30 s | goE | goE | waitE | waitE |
| 3 | waitE | 010 100 | 5 s | goN | goN | goN | goN |

This is how we implement this state machine we make a structure what have all four states.Out in out case is semafor_stateid and represents the id of state.Delay is time when system is active after that depend of input system change state.ln represents an array of how state will change from the current state this array have 4 elements. This array has four elements because are four type of unput that can receive system and for every type of input system is calling a specific state from array. So if system receive 00 then is calling the first element of array that is goN if system receive 01 the system is calling the second element of array that is waitN, if system receive 10 then is calling the third element of array that is goN and if system receive 11 when is calling the four element that is waitN. This more easier to take a state from array because the input is in binary is equal to order number of state in array so 00 = 0 so this is first element, 01=1 this is second element, 10=2 this is third element and 11=3 this is four element. In state waitN for any type of input system will change to state goE.In state goE for input 00 and 01 will stay the same state goE but for input 10 11 will change to state from third and four element of array that is state waitE. In state waitE in every type of input will change system to state goN.

2.3 Simulated or actually assembled electrical schematic

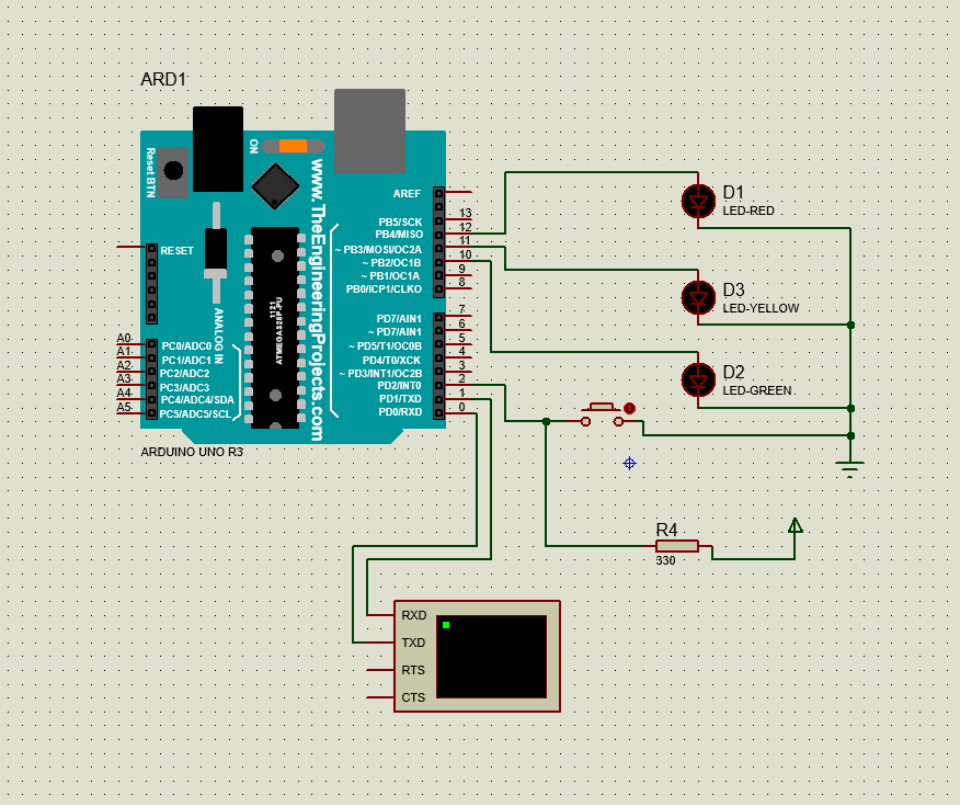



**Figure 2** Simulated assembled electrical schematic

2.4 Screenshots of running the simulation program or photos of the real assembled equipment with screenshots of the “terminal” application on the computer

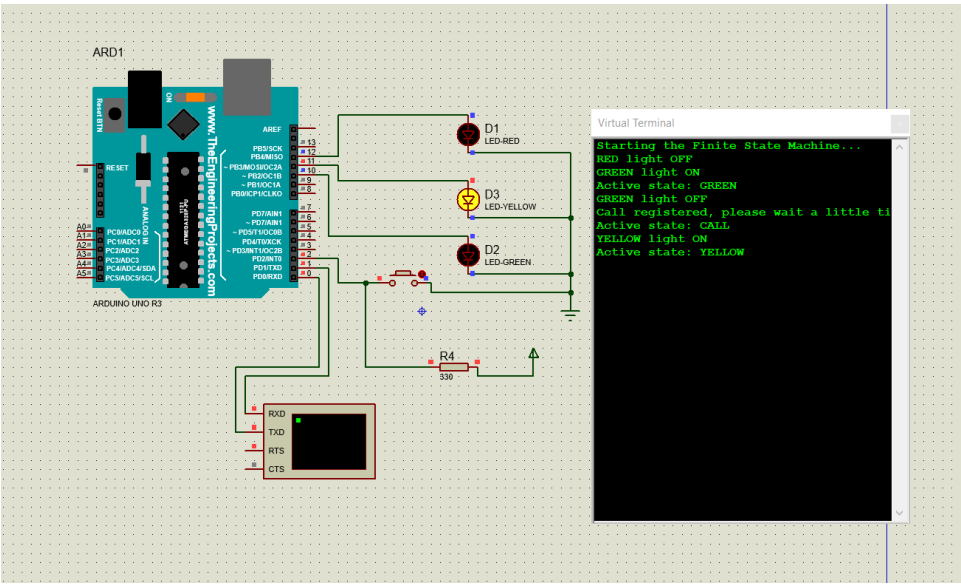


**Figure 3** Running simulation program

## 3. Conclusions:

During this laboratory work I learnt how to make a finite automata in Arduino. This pattern is good to use when you have a system which have a lot of states as this give more control on states and how they change.

**Appendix 1**

**Code:**

```
#include <YA_FSM.h>  // https://github.com/cotestatnt/YA_FSM


const byte BTN_CALL = 2;
const byte GREEN_LED = 12;
const byte YELLOW_LED = 11;
const byte RED_LED = 10;


// Create new FSM
YA_FSM stateMachine;


// State Alias
enum State {RED, GREEN, YELLOW, CALL};


// Helper for print labels instead integer when state change
const char * const stateName[] PROGMEM = { "RED", "GREEN", "YELLOW", "CALL"};


// Pedestrian traffic light -> green ligth ON until button pressed
#define YELLOW_TIME  2000
#define RED_TIME     10000
#define CALL_DELAY   5000
```

```
// Input (trig transition from GREEN to CALL state, the others transitions on timeout)
bool callButton = false;


// Output variables
bool redLed = false;
bool greenLed = false;
bool yellowLed = false;


////////////// STATE MACHINE CALLBACK FUNCTIONS //////////////////
// Define "on entering" callback function (the same for all "light" states)
void onEnter() {
  Serial.print(stateMachine.ActiveStateName());
  Serial.println(" light ON");
}
// Define "on leaving" callback function (the same for all "light"  states)
void onExit() {
  Serial.print(stateMachine.ActiveStateName());
  Serial.println(" light OFF\n");
}
// Define "on enter" for CALL button state
void onEnterCall() {
  Serial.println("Call registered, please wait a little time.");
}


// Setup the State Machine
void setupStateMachine() {
```

```
  // Follow the order of defined enumeration for the state definition (will be used as index)
  // Add States => name,timeout, onEnter cb, onState cb, onLeave cb
  stateMachine.AddState(stateName[RED], RED_TIME, onEnter, nullptr, onExit);
  stateMachine.AddState(stateName[GREEN], 0, onEnter, nullptr, onExit);
  stateMachine.AddState(stateName[YELLOW], YELLOW_TIME, onEnter, nullptr, onExit);
  stateMachine.AddState(stateName[CALL],   CALL_DELAY,   onEnterCall,   nullptr, nullptr);


  stateMachine.AddAction(RED, YA_FSM::N, redLed);         // N -> while state is active red led is ON
  stateMachine.AddAction(GREEN, YA_FSM::S, greenLed);    // S -> SET green led on
  stateMachine.AddAction(YELLOW, YA_FSM::R, greenLed);    // R -> RESET the green led
  stateMachine.AddAction(YELLOW, YA_FSM::N, yellowLed);   // N -> while state is active yellow led is ON


  // Add transitions with related trigger input callback functions
  // In this example it's just a simple lambda function that return state timeout value
  stateMachine.AddTransition(RED, GREEN, [](){return stateMachine.CurrentState()->timeout;} );
  stateMachine.AddTransition(YELLOW,              RED,              [](){return stateMachine.CurrentState()->timeout;}  );
  stateMachine.AddTransition(CALL,              YELLOW,              [](){return stateMachine.CurrentState()->timeout;});
  stateMachine.AddTransition(GREEN, CALL, callButton);
}
```

```
void setup() {
  // Setup Input/Output
  pinMode(BTN_CALL, INPUT_PULLUP);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);

  Serial.begin(9600);
  // while(!Serial) {}  // Needed for native USB port only
  Serial.println("Starting the Finite State Machine...\n");
  setupStateMachine();
}

void loop() {
  // Read inputs
  callButton = (digitalRead(BTN_CALL) == LOW);

  // Update State Machine
  if(stateMachine.Update()){
    Serial.print("Active state: ");
    Serial.println(stateMachine.ActiveStateName());
  }

  // Set outputs
  digitalWrite(RED_LED, redLed);
  digitalWrite(GREEN_LED, greenLed);
  digitalWrite(YELLOW_LED, yellowLed);
```

```
}
```