**НП „ИТ КАРИЕРА“**

**Модул 8 - Разработка на софтуер**

**Изготвил:**

Ениз Тургай Хасан

От ПМГ „Акад. Боян Петканчин“

Хасково

**Проект Traffic Lights**

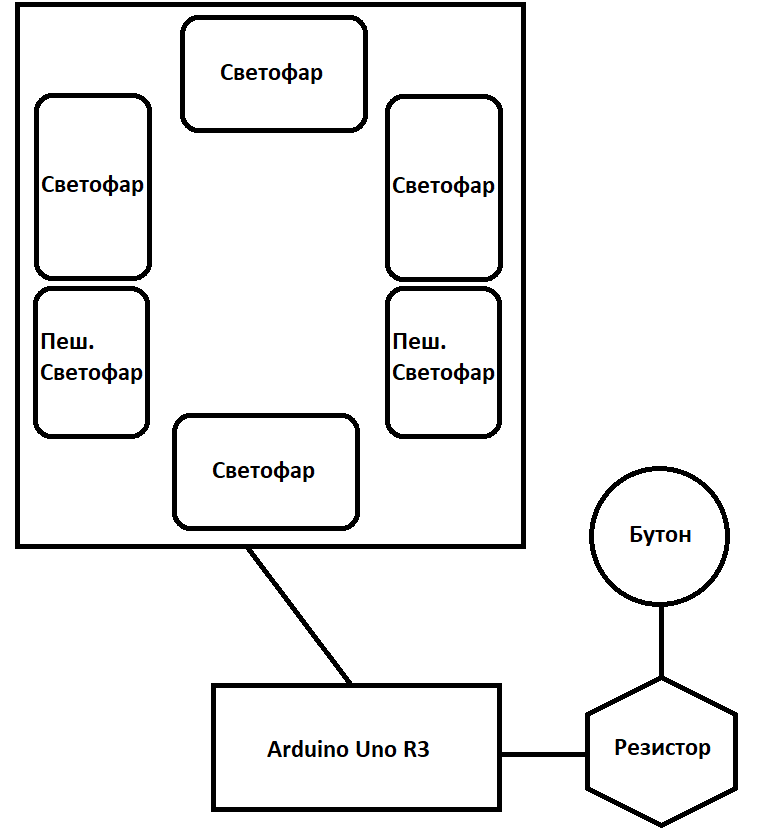
**Описание на проекта:**

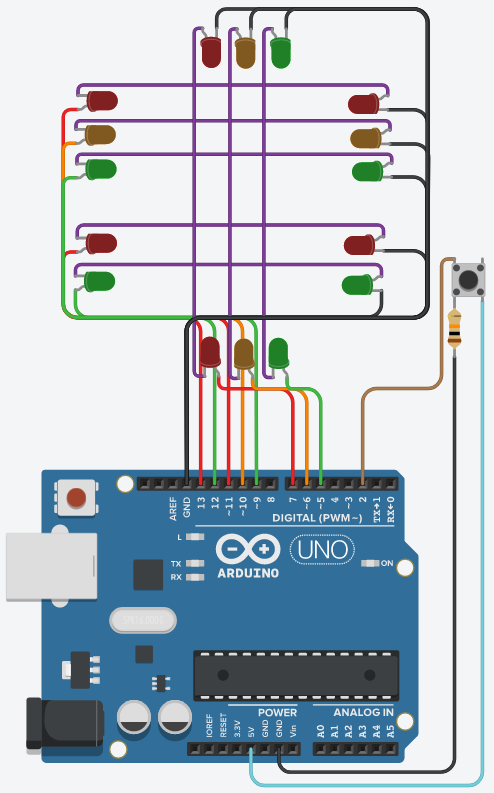
Проектът е разработен в webapp-а Tinkercad, който представлява симулатор на електрически вериги.

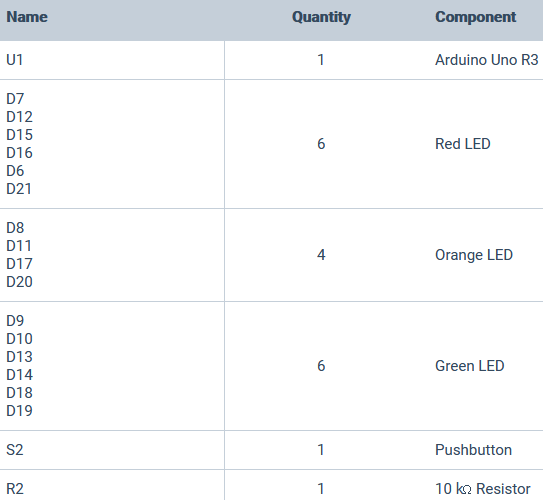
Проектът е симулация на светофари в кръстовище (вкючително пешеходни светофари). Изплозвам Arduino Uno R3, като контролер за смяната на цветовете на светофарите. При натискане на пешеходния бутон Arudino-то променя светлините на светофарите, ката че да позволи на пешеходците да преминат безопастно през кръстовището.

Цялата фунционалност се управлява от компютъра на Arduino-то, което изпълнавя моята програма (сорс код по-отдолу). Тази програма се грижи за правилната смяна на цветовете на светофарите, интервала между тези смени и скоростта им. Също така програмата наблюдава състоянието на пешеходния бутон и променя (или не променя) светофарите спрямо него.

**Блокова схема на проекта**



**Електрическа схема**

**Списък съставни части**

**Сорс код и описание на функционалността**

**Описание:**

За възможно най-лесна работа си предефинирам пиновете за различните видове светофари (спрямо позиция и тип). В глобална променлива запазвам състоянието на пешеходния светофар и бутона.

За улеснение на работата си създавам няколко структури:

* Светлина
* Пешеходен светофар (съдържа две светлини)
* Нормален светофар (съдържа три светлини)

Всяка от тези структури е способна да си промени състоянието, тоест да направи една стъпка напред в състоянието си (на пример при пешеходен светофар: зелено -> червено)

Създавам си няколко глобални фунции които управляват светофарите и наблюдават състоянието на бутона.

В setup() инициализирам структурите и bind-вам светлините на светофарите към техните пинове.

В loop() използвам глобалните фунции за проверка на бутона и сменям състоянието на светофарите спрямо изтеклото време.

**Сорс код:**

bool pedState = false; // true = GREEN : false = RED

int timestep = 10;

int lastButtonState;

int trafficTimer = 12000;

int ped[] = {13, 12};

int hor[] = {11, 10, 9};

int ver[] = {7, 6, 5};

bool curStateLR = false; // false -> RED

struct Light

{

int pin;

int state = LOW;

Light(int pinNumber)

{

pin = pinNumber;

}

void ToggleState()

{

if(state == LOW)

SetState(HIGH);

else if(state == HIGH)

SetState(LOW);

}

void SetState(int newState)

{

if(newState != LOW && newState != HIGH)

{

Serial.println("Invalid state!");

return;

}

digitalWrite(pin, newState);

state = newState;

}

};

struct PedLightArray

{

Light\* lights[2];

PedLightArray(int pins[2])

{

for(int i = 0; i < 2; i++)

lights[i] = new Light(pins[i]);

lights[0]->state = HIGH;

}

void ToggleState()

{

for(int i = 0; i < 2; i++)

lights[i]->ToggleState();

}

};

struct TrafficLightArray

{

int state = 0; // 0 - R, 1 - RY, 2 - YG, 3 - G, 4 - GY, 5 - YG

Light\* lights[3];

TrafficLightArray(int pins[3])

{

for(int i = 0; i < 3; i++)

lights[i] = new Light(pins[i]);

}

void ToggleState()

{

state++;

if(state > 5)

state = 0;

switch(state)

{

case 0:

FullRed();

return;

case 1:

RedToYellow();

return;

case 2:

YellowToGreen();

return;

case 3:

FullGreen();

return;

case 4:

YellowToGreen();

return;

case 5:

RedToYellow();

return;

default:

Serial.println("Invalid State");

break;

}

}

void FullRed()

{

lights[0]->SetState(HIGH);

lights[1]->SetState(LOW);

lights[2]->SetState(LOW);

state = 0;

}

void RedToYellow()

{

lights[0]->SetState(HIGH);

lights[1]->SetState(HIGH);

lights[2]->SetState(LOW);

}

void YellowToGreen()

{

lights[0]->SetState(LOW);

lights[1]->SetState(HIGH);

lights[2]->SetState(HIGH);

}

void FullGreen()

{

lights[0]->SetState(LOW);

lights[1]->SetState(LOW);

lights[2]->SetState(HIGH);

state = 3;

}

};

PedLightArray\* pedLR;

TrafficLightArray\* LR;

TrafficLightArray\* UD;

void setup()

{

pinMode(ped[0], OUTPUT); // 13 - PED RED

pinMode(ped[1], OUTPUT); // 12 - PED GREEN

pinMode(hor[0], OUTPUT); // 11 - MAIN 1 RED

pinMode(hor[1], OUTPUT); // 10 - MAIN 1 ORANGE

pinMode(hor[2], OUTPUT); // 9 - MAIN 1 GREEN

pinMode(ver[0], OUTPUT); // 7 - MAIN 2 RED

pinMode(ver[1], OUTPUT); // 6 - MAIN 2 ORANGE

pinMode(ver[2], OUTPUT); // 5 - MAIN 2 GREEN

pinMode(2, INPUT);

Serial.begin(9600);

lastButtonState = LOW;

pedLR = new PedLightArray(ped);

pedLR->lights[0]->SetState(HIGH);

LR = new TrafficLightArray(hor);

UD = new TrafficLightArray(ver);

LR->FullRed();

UD->FullGreen();

}

bool ButtonPressed()

{

return digitalRead(2) == HIGH;

}

bool ButtonChanged(bool targetState)

{

if(lastButtonState != targetState && ButtonPressed() == targetState)

{

lastButtonState = targetState;

Serial.println("Button changed");

return true;

}

lastButtonState = !targetState;

return false;

}

void ToggleTrafficLightState(bool lr)

{

TrafficLightArray\* traffLight = lr ? LR : UD;

traffLight->ToggleState();

delay(250);

traffLight->ToggleState();

delay(150);

traffLight->ToggleState();

}

void TogglePedState()

{

pedLR->ToggleState();

}

void ToggleTrafficState()

{

if(curStateLR)

TogglePedState();

ToggleTrafficLightState(curStateLR);

delay(2000);

if(!curStateLR)

TogglePedState();

curStateLR = !curStateLR;

ToggleTrafficLightState(curStateLR);

}

void loop()

{

int counter = trafficTimer;

while(counter > 0)

{

delay(timestep);

counter -= timestep;

if(ButtonChanged(true) && !curStateLR)

{

ToggleTrafficState();

Serial.println("Called for button");

return;

}

}

ToggleTrafficState();

}