­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Контрольна робота

з курсу «Математичне моделювання в САПР»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 10

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2015

# Практична 1

### Найменування об'єкту розробки, та область застосування:

Регульоване перехрестя - система світлофорів для регульованого перехрестя, включає в себе дві головні дороги і чотири пішохідних переходи.

### Підстава для розробки та назва проектної організації:

Підставою служить виконання лабораторних робіт з курсу "Математичне моделювання в САПР".

### Мета розробки:

Даний об’єкт розробляється з метою керування світлофорами на регульованих перехрестях.

### Технічні вимоги, які включають:

Склад об'єкту та вимоги до його конструктивного виконання:

* Об’єкт складається з плати (Arduino), вбудованої батареї, яка складається з чотирьох елементів живлення типу АА, платою розширення портів вводу/виводу.
* Розміри замка :194х68х34, вхідна ручка: 136х70х56 мм.
* Вага 300гр

### Логічне представлення

Діаграмма нижче показує логічне розташування системи світлофорів.



**Дорожній Світлофор** – компонента системи яка включає в себе пару(чи більше) синхронізованих світлофорів для одного напрямку дороги.

**Контроллер** – компонента керування системою світлофорів.

### Алгоритм роботи

Діаграма нижче показує алгоритм роботи світлофора. Всі світлофори стартують з червоного кольору, і перемикають на зелений тільки якщо інша група світлофорів має червоний колір.

## 

# Практична 2

Мережа Петрі для двох взаємопов’язаних груп світлофорів. Головна особливість це не допуск одночасного зеленого свфтла.

### 

Система провалідована за допомогою tokengame

# Практична 3

Реалізація складається з двох компонент.

1. proto.js – прототип на мові JavaScript. Використовується для протипіювання і тестування рішення. Використовується DSL.
2. lc.ino – Arduino C реалізаця DSL, що в свою чергу генереється з proto.js

Лістінг proto.js

var assert = require('assert');

var c1 = 'Y';

var c2 = 'Y';

var p1 = null;

var L = 2000;

var S = 1;

var specs = [

{ prec: {c1: 'Y', c2: 'Y', p1: 'r'}, action: {c1: 'R', c2: 'Y'}, delay: S },

{ prec: {c1: 'R', c2: 'Y', p1: 'r'}, action: {c1: 'R', c2: 'G'}, delay: L },

{ prec: {c1: 'R', c2: 'G', p1: 'r'}, action: {c1: 'R', c2: 'Y', p1: 'R'}, delay: S },

{ prec: {c1: 'R', c2: 'Y', p1: 'g'}, action: {c1: 'Y', c2: 'Y'}, delay: L },

{ prec: {c1: 'Y', c2: 'Y', p1: 'g'}, action: {c1: 'Y', c2: 'R'}, delay: S },

{ prec: {c1: 'Y', c2: 'R', p1: 'g'}, action: {c1: 'G', c2: 'R'}, delay: L },

{ prec: {c1: 'G', c2: 'R', p1: 'g'}, action: {c1: 'Y', c2: 'R', p1: 'G'}, delay: S },

{ prec: {c1: 'Y', c2: 'R', p1: 'r'}, action: {c1: 'Y', c2: 'Y'}, delay: L }

];

var match = function(item){

var prec = item.prec;

if (prec.c1 === c1 && prec.c2 === c2){

if (prec.p1){

if ((prec.p1 === 'r' && p1 != 'R') || (prec.p1 === 'g' && p1 != 'G')){

return true;

}

} else {

return true;

}

}

return false;

};

for(var i = 0; i < 100; i++){

var spec = specs.filter(match)[0];

assert(spec);

var action = spec.action;

c1 = action.c1;

c2 = action.c2;

assert(!(c1 == 'G'&& c2 == 'G'));

if (action.p1){

p1 = action.p1;

}

console.log(c1, c2);

}

// Print C specs

specs.forEach(function(x, i){

var r = 'SPEC spec' + (i+1) + ' = SPEC{' + x.prec.c1 + ', '+ x.prec.c2;

r += ', ' + ((x.prec.p1) ? x.prec.p1 : '\_');

r += ', ' + x.action.c1 + ', '+ x.action.c2;

r += ', ' + ((x.action.p1) ? x.action.p1 : '\_');

r += ', ' + x.delay;

r += '};';

console.log(r);

});

Лістінг lc.ino

#include "Tlc5940.h"

const byte \_ = 0;

const byte R = 1;

const byte Y = 2;

const byte G = 3;

const byte r = 11;

const byte g = 13;

struct SPEC {

byte pred\_c1;

byte pred\_c2;

byte pred\_p1;

byte action\_c1;

byte action\_c2;

byte action\_p1;

int delay;

};

const int LONG\_DELAY = 2000;

const int SHORT\_DELAY = 1;

SPEC spec1 = SPEC{ Y, Y, r, R, Y, \_, SHORT\_DELAY };

SPEC spec2 = SPEC{ R, Y, r, R, G, \_, LONG\_DELAY };

SPEC spec3 = SPEC{ R, G, r, R, Y, R, SHORT\_DELAY };

SPEC spec4 = SPEC{ R, Y, g, Y, Y, \_, LONG\_DELAY };

SPEC spec5 = SPEC{ Y, Y, g, Y, R, \_, SHORT\_DELAY };

SPEC spec6 = SPEC{ Y, R, g, G, R, \_, LONG\_DELAY };

SPEC spec7 = SPEC{ G, R, g, Y, R, G, SHORT\_DELAY };

SPEC spec8 = SPEC{ Y, R, r, Y, Y, \_, LONG\_DELAY };

const byte SPECS\_LENGTH = 8;

SPEC\* specs[SPECS\_LENGTH] = { &spec1, &spec2, &spec3, &spec4, &spec5, &spec6, &spec7, &spec8 };

int c1 = Y;

int c2 = Y;

int p1 = \_;

void setup()

{

Tlc.init();

}

void light(byte l, byte n){

byte channel = (n \* 3) + (l - 1);

Tlc.set(channel, 4095);

}

void loop()

{

int i;

// Find matching spec

SPEC\* spec;

for(i = 0; i < SPECS\_LENGTH; i++){

if (specs[i]->pred\_c1 == c1 && specs[i]->pred\_c2 == c2){

if (specs[i]->pred\_p1 != \_){

if ((specs[i]->pred\_p1 == r && p1 != R) || (specs[i]->pred\_p1 == g && p1 != G)){

spec = specs[i];

break;

}

} else {

spec = specs[i];

break;

}

}

}

// Apply action

c1 = spec->action\_c1;

c2 = spec->action\_c2;

if (spec->action\_p1 != \_){

p1 = spec->action\_p1;

}

// Light!

Tlc.clear();

light(c1, 0);

light(c2, 1);

Tlc.update();

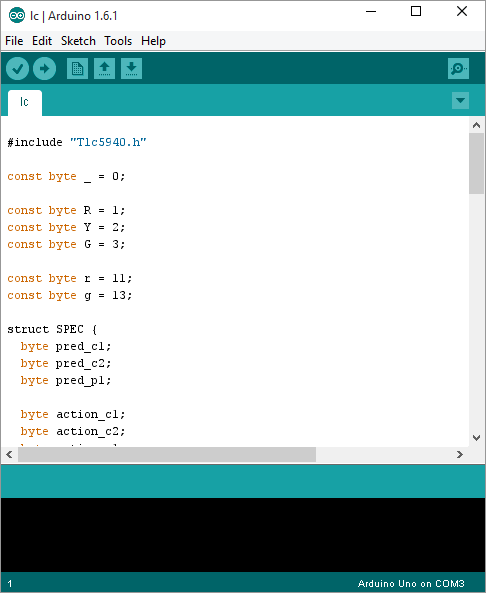
delay(spec->delay);

}

Повна версія кода, разом з файлами проекта - https://github.com/chaliy/studies-octo-adventure/tree/master/lp/c3\_2/mm

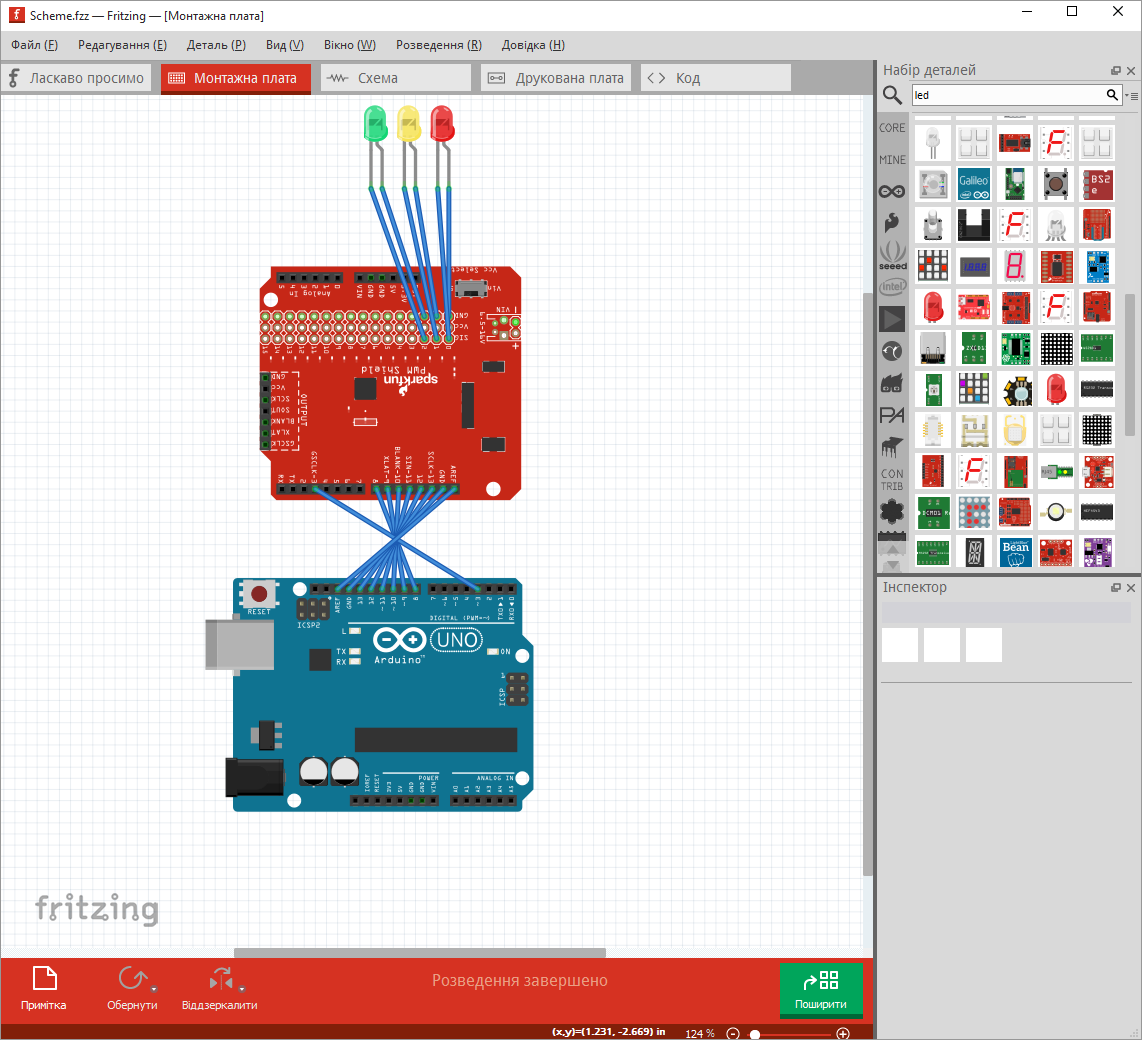
## Результат

Програма в середовищі Arduiono.

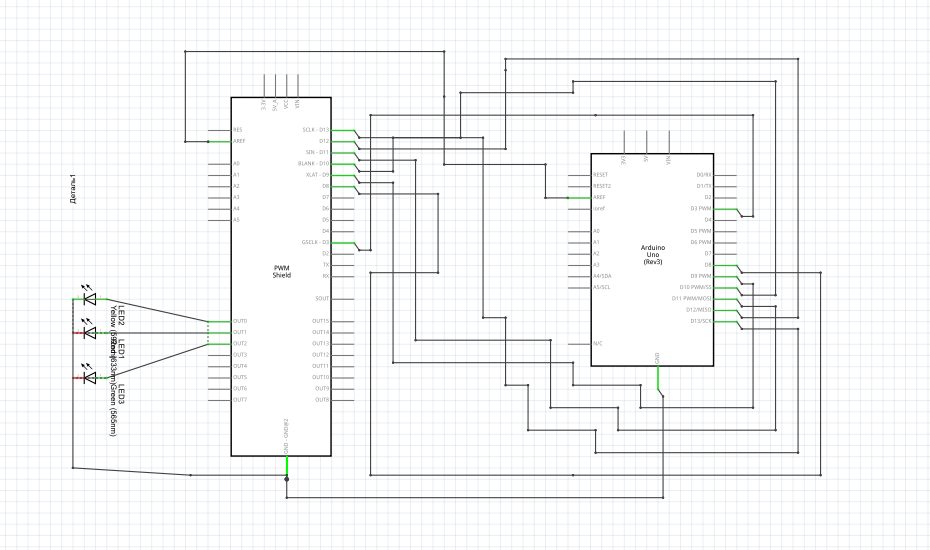


# Практична 4

### Монтажна плата



### Схема електрична принципіальна



Відео прототипа - http://1drv.ms/1eB5jBV