МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт энергетики и электротехники
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование кафедры)

11.04.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Электронные приборы и устройства
(направленность (профиль)/специализация)

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему «Электронные шашки»

Студент А.В. Николаев _____
(И.О. Фамилия) (личная подпись)

Научный руководитель к.т.н., доцент А.В. Прядилов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

Руководитель программы д.т.н., профессор В.В. Ивашин _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«____»_____2018г.

Допустить к защите

Заведующий кафедрой к.т.н., доцент А.А. Шевцов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«____»_____2018г.

Тольятти 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ…… 3
1 Обзор существующих вариантов электронных досок…….. 4
2 Разработка структурной схемы прибора для изучения динамической индикации и выбор элементов……8
3 Разработка электрической схемы прибора для изучения динамической индикации……13
4 Разработка конструкции прибора для изучения динамической индикации…….. 15
5 Разработка программной части прибора для изучения динамической индикации…….. 17
6 Экспериментальные исследования динамической индикации…37
7 Разработка структурной схемы электронной игры в шашки….. 39
8 Разработка электрической схемы…….. 40
9 Разработка конструкции…… 43
10 Разработка программы…….. 47
ЗАКЛЮЧЕНИЕ……... 79
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ…….80
ПЕРЕЧЕНЬ ЭЛЕМЕНТОВ……84

ВВЕДЕНИЕ

В настоящее время актуальна задача управления большим количеством светодиодов. Для этого используются ШИМ и динамическая индикация. Для того, чтобы собирать качественные приборы, с использованием динамической индикации, нужно знать, как человеческий глаз ее воспринимает.

В данной работе достигаются две цели. Первая – исследование восприятия глазом динамической индикации и ШИМ – регулирования яркости. Вторая – создание электронной игры в шашки.

Задачи работы:

- Разработка структурной схемы прибора, для изучения динамической индикации.
- Разработка электрической схемы.
- Разработка конструкции.
- Написание и отладка программы.
- Экспериментальные исследования динамической индикации и ШИМ.
- Разработка структурной схемы электронной игры в шашки.
- Разработка электрической схемы.
- Разработка конструкции.
-Разработка программы.

1 Обзор существующих вариантов электронных досок

Из существующих решений были найдены только шашки либо полностью электронные, либо с фигурами, игровой процесс которых контролируется компьютером. Эти варианты принципиально отличаются от разрабатываемых в моей работе.

Электронная игра 4tune-G860 [1], представляет из себя небольшой модуль размером примерно с пачку сигарет и представлена на рисунке 1.1.

Рисунок 1.1 - Электронная игра 4tune-G860

Игра заключена в пластиковый корпус, серого цвета. Размеры 11 на 7 на 2,5 сантиметра. Экран сенсорный, со встроенной подсветкой, изготовленный из стекла, в которой расположены мельчайшие проводники.

Игра управляется с помощью стилуса. Запитывается игра от батареек AAA, слот рассчитан на две штуки. Крышка слота закрывается на саморез.

Рисунок 1.2 - Электронная игра 4tune-G860 с обратной стороны

Устройство включает в себя шашки, шахматы и линии. У шашек можно настраивать правила: европейские или русские шашки. Количество ходов ограниченно: можно сделать не более 120 ходов. Играть можно против компьютера или человека. Уровень сложности игры против компьютера настраивается от 1 до 20. Чем выше уровень сложности, тем дольше устройство обдумывает ходы. Помимо этого есть функция подсказки, отмены хода, отсчета времени.

В игре линии задача играющего соединить, выстроить в линию(по горизонтали, вертикали или под угон) 4 свои пешки. При этом пешки можно

ставить (бросать), как на свободное место, так и сверху на свои или пешки противника. Побеждает тот, кто первый построил линию из 4-х своих пешек.

Рисунок 1.3 - Электронная игра 4tune-G860 включенная

Магнитные шашки с дисплеем.

На китайском сайте [2] было найдено еще одно исполнение шашек, которое все же сильно отличается от предложенных в моей работе, представлено на рисунках 1.4, 1.5.

Это устройство включает в себя 8 игр: шашки, шахматы, реверси, 4 в ряд, лиса, гуси, кузнечик, ним, норткот. В отличие от моего устройства, в магнитных шашках присутствуют фигуры, перемещение которых фиксируются с помощью сенсоров, и отображаются на ЖК экране. В устройстве есть функция обучения, для новичков, а так же ест возможность играть против компьютера.

Рисунок 1.4 – Магнитные шашки

Рисунок 1.5 – Магнитные шашки

2 Разработка структурной схемы прибора для изучения динамической индикации и выбор элементов

Главным электрическим элементом в приборе является RGB – светодиод. Для задания различных режимов работы диода и подачи на него различных сигналов, в том числе ШИМ и динамического, используем микроконтроллер. Управление режимами микроконтроллера осуществляется с помощью джойстика. Информация о режиме исследования выводится на LCD – дисплей.

Структурная схема устройства представлена на рисунке 2.1.

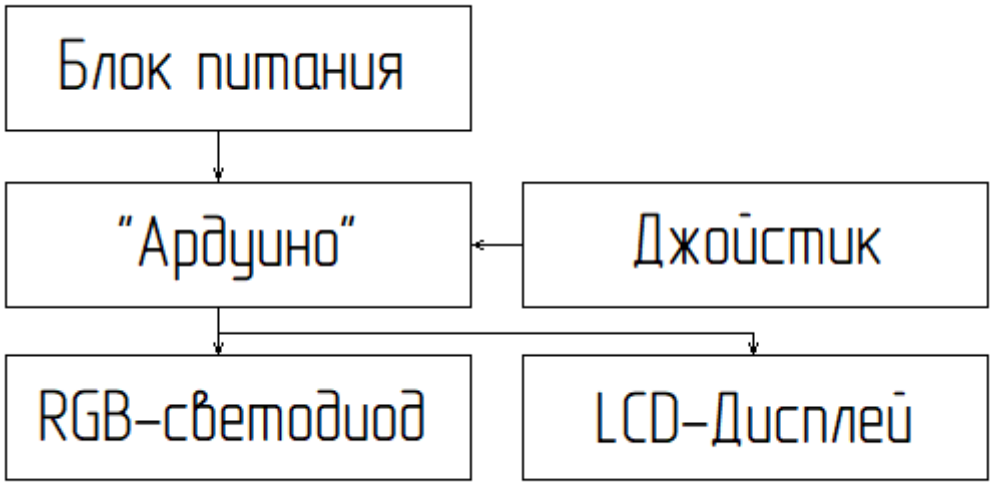


Рисунок 2.1 – Структурная схема

Перечисленные элементы были закуплены в электронных магазинах и на соответствующих сайтах.

RGB – светодиод выбран такой, которым можно управлять с помощью 5 вольт от цифровых выводов микроконтроллера.

LCD – дисплей содержит 2 строчки по 16 символов, чего достаточно для отображения номера режима исследования, а так же различных данных.

В первом режиме он отображает скважность ШИМ – сигнала сравниваемую со стопроцентной скважностью, для определения того, какое изменение скважности будет замечать глаз. Так же он выводит значение цвета, по шкале RGB от 0 до 255.

Во втором режиме дисплей отображает частоту включения диода для изучения динамической индикации, а так же цвета, используемые в данном опыте: красный, зеленый, синий, красный/зеленый, красный/синий, зеленый/синий.

В третьем режиме дисплей отображает подключаемое сопротивление к диоду, режим работы диода: импульсный или постоянный, а так же скважность и цвет.

Джойстик [3] содержит 5 выводов, два из которых используются как питание джойстика, два идут на аналоговый вход платы «Ардуино», и сигнал с них говорит о наклоне по оси X и У. Последний вывод срабатывает на нажатие и идет на цифровой вход платы. Изменение сигналов осей X и У после наклона ручки управления основано на изменение сопротивление двух реостатов, расположенных под прямым углом друг относительно друга.

Необходимо выбрать микроконтроллер для управления схемой. Обратим внимание на платы «Ардуино» [4], [5], на базе AVR микроконтроллеров. Нам подойдет плата «Ардуино УНО», которая имеет необходимое количество выводов для решения наших задач.

Кварцевый генератор УНО частотой 16 МГц, так же имеется разъем USB, разъем для подачи питания с блока питания. Разъем ICSP и кнопка перезагрузки нам не понадобятся. В плате встроен микроконтроллер ATMega 328.

Всего плата имеет 14 цифровых пинов, которые обозначаются D0-D13. Шесть пинов можно использовать как выходы ШИМ [6]. Каждый пин можно запрограммировать на вход или выход, используя команды pin Mode(), digital Write(), digital Read(). Работа пинов настроена на напряжение 5В. У пинов имеется подтягивающий резистор 20 кОм, который можно отключать и подключать программно.

Некоторые пины могут выполнять дополнительные функции. Последовательная шина RX, TX используется для получения (RX) и передачи (TX) данных ТТЛ. Пины 2 и 3 – внешнее прерывание. Шесть выводов, отмеченные специальным символом и идущие под номерами 3, 5, 6, 9, 10, 11 могут использоваться для ШИМ, которая задается функцией analog write. Связь SPI осуществляется выводами 10 – 13 (SS, MOSI, MISO, SCK). Тринадцатый пин подключен к светодиоду, который загорается когда на пине есть сигнал.

Шесть входов платы имеют АЦП, используются как аналоговые пины и обозначаются A0-A5. Эти пины могут выдавать 1024 значения с разрешением 10 бит. Обычно эти пины могут замерить сигнал до 5 В, но верхний предел можно изменить, используя вывод AREF и функции analog reference.

Пин платы 5V может использоваться как питание компонентов платы и внешних компонентов. Пин 3V3 дает напряжение 3,3В. На пин VIN можно подать напряжение от внешнего источника питания и запитывать плату и компоненты. Пин GND – это земля.

Пин AREF задает опорное напряжение для аналоговых входов. Задается функцией analog reference. Пин RESET перезагружает плату когда на него подается логический ноль. Так же имеется кнопка перезагрузки, но пин RESET используется когда нет доступа к кнопке. В этой работе

перезагружать плату будем включая и выключая ее ключем, поэтому этот пин нам не понадобится.

На плату нельзя подавать питание более 20В. Питается плата от 7-12 В. Теоретически при меньшем напряжении плата должна работать нестабильно, но на практике она хорошо работает и от 4-5 В. Максимальный ток через пины равен 40 мА. Частота работы 16 МГц.

Микроконтроллер ATmega328 имеет 32 кБ флэш памяти, 0,5 кБ используется для хранения загрузчика, 2 кБ ОЗУ, 1кБ EEPROM

Источники света, используемые в промышленном машинном видении, обычно используют один из основных цветов, красный, зеленый и синий (RGB), интенсивности которого можно варьировать, регулируя Входной уровень электрического тока. Светлый цвет влияет на работу контрольной машины, а также на циркадные ритмы рабочих на фабриках [7].

Следовательно, светлый цвет важен в производственном процессе, но выбор цвета ограничен на аппаратном уровне при использовании одного источника света. Смешанный источник света может эртеировать различные цвета, комбинируя светодиоды RGB и построенные с использованием оптического микшера, такого как коллиматор, [8] микролинзовая матрица, раковина-смеситель, диффузор, или пучок оптических волокон. Хотя такой смешанный источник света может генерировать различные цвета, спектральные диапазоны мощных светодиодов являются постоянными.

Поэтому вспомогательные цветные светодиоды применяются к смешанному источнику света, когда требуется гибкость спектрального диапазона. В этом случае цветовая комбинация называется «RGB и белый» (RGBW) или «RGB и желтый» (RGBY). Уже введена коммерческая ИС для управления четырехканальным смешанным источником. [9] Алгоритм цветового микширования определяет входной уровень, уровень затемнения смешанного источника света для определенной цели, используя

численный метод. Целью может быть улучшение качества изображения, [10] определение оптического спектра, [11] и наведение желаемой координаты цвета.

Коммерческие светодиоды имеют типичные цвета. Однако эти цвета обычно содержат элементы RGB даже в случае светодиодов RGB. Таким образом, цвет, который генерирует светодиод, состоит из основных цветов и света.

Модель смешивания может быть показана в линейной комбинации с отдельными светодиодами. [12] Линейное отношение формируется с использованием одной из различных цветовых систем координат, таких как (XYZ), цветовая температура и CIE. Линейные формы обычно применялись к смесителям RGB [13, 14], и Sisto предложил расширенную модель с произвольным количеством светодиодов. [15] Модель Sisto может применяться к смешанному источнику света, используя вспомогательные цветные светодиоды для выбранного цвета.

Источник RGBW является наиболее распространенным типом, и его линейные модели обсуждались в отношении проектирования контроллера. Выбранный пользователем цвет имеет три элемента в системе координат цвета, но уровни затемнения для смешанного источника света (Например, RGBW) требуют дополнительных координат [16]. Таким образом, может быть много решений для уровня затемнения.

3 Разработка электрической схемы прибора для изучения динамической индикации

Центральным элементом схемы является плата «Ардуино». Она запитывается от 9 вольт USB – провода, который может идти на компьютер или блок питания.

От пятивольтового выхода платы питается джойстик. Питание подается на соответствующие выводы 5V, GND. Выводы джойстика VRX, VRY идут на аналоговые входы платы, для того, что бы можно было считать положение ручки правления, вывод SW идет на цифровой вход платы, для считывания нажатия.

От платы так же идет питание на дисплей и подается на соответствующие входа: A и K, через резистор для управления яркостью. Управляется дисплей с помощью цифровых выводов «Ардуино», подающих сигнал на входа D4-D7, RS, E. Вход V0 подключается через резистор на землю, и регулирует контрастность. VSS, RW подключаются на землю, VDD на +5 вольт.

Светодиод подключается на цифровые входа, для управления разными цветами. Красный и зеленый цвета подключены на 2 цифровых входа, через разные резисторы.

Схема разработанного прибора представлена на рисунке 3.1.

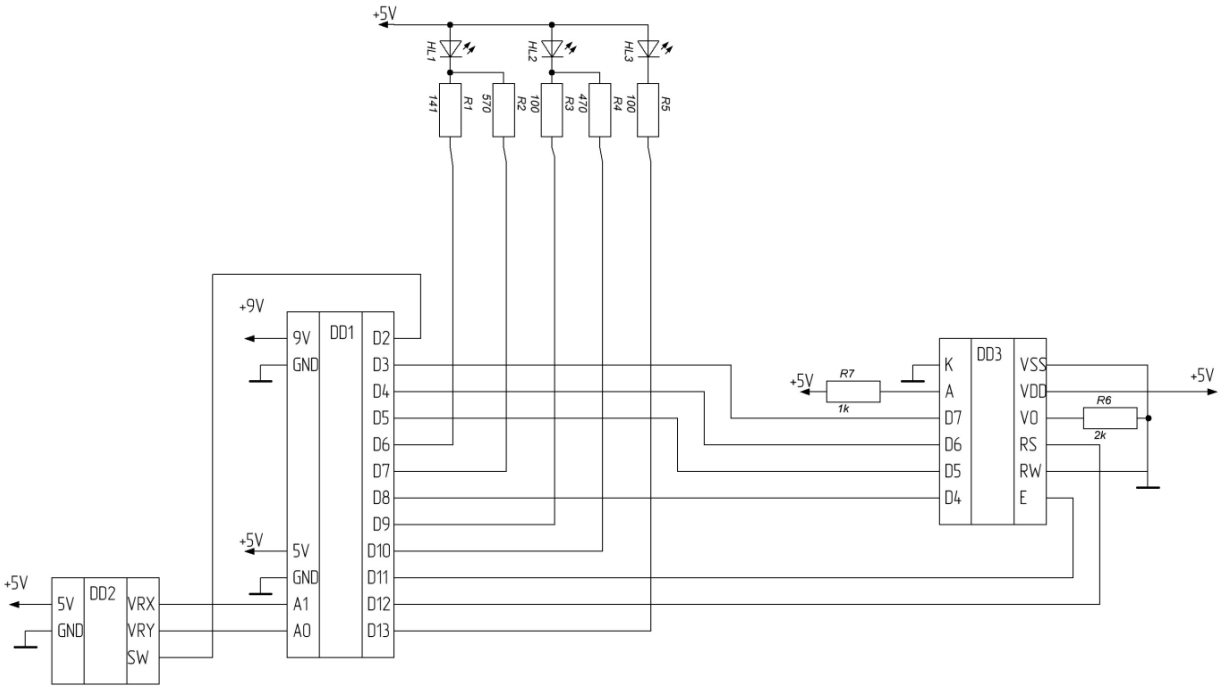


Рисунок 3.1 – Схема прибора для изучения динамической индикации и ШИМ

4 Разработка конструкции прибора для изучения динамической индикации

Главное назначение прибора – демонстративность. Основные требование к конструкции: наглядность, компактность, аккуратность и эстетичность.

Все элементы схемы аккуратно размещаются в пластиковой коробке. Коробка защищает элементы от повреждения, помогает элементам расположиться и закрепиться, а так же придает эстетичность прибору.

Верхняя крышка снимается, что дает доступ к элементам. В коробке проделаны необходимые отверстия для дисплея, светодиода, джойстика и USB – кабеля. Для светодиода сделан рассеиватель.

Элементы в коробке закрепляются с помощью мягкого двухстороннего скотча. Для компактности плата и дисплей располагаются в два яруса, друг над другом. Таким образом, дисплей прилегает к верхней крышке, к которой делается для него прямоугольное отверстие.

Элементы схемы соединяются с помощью специальных проводов и макетной платы, которая позволяет быстро переделывать схему без необходимости пайки.

Собранный прибор представлен на рисунках 4.1, 4.2.

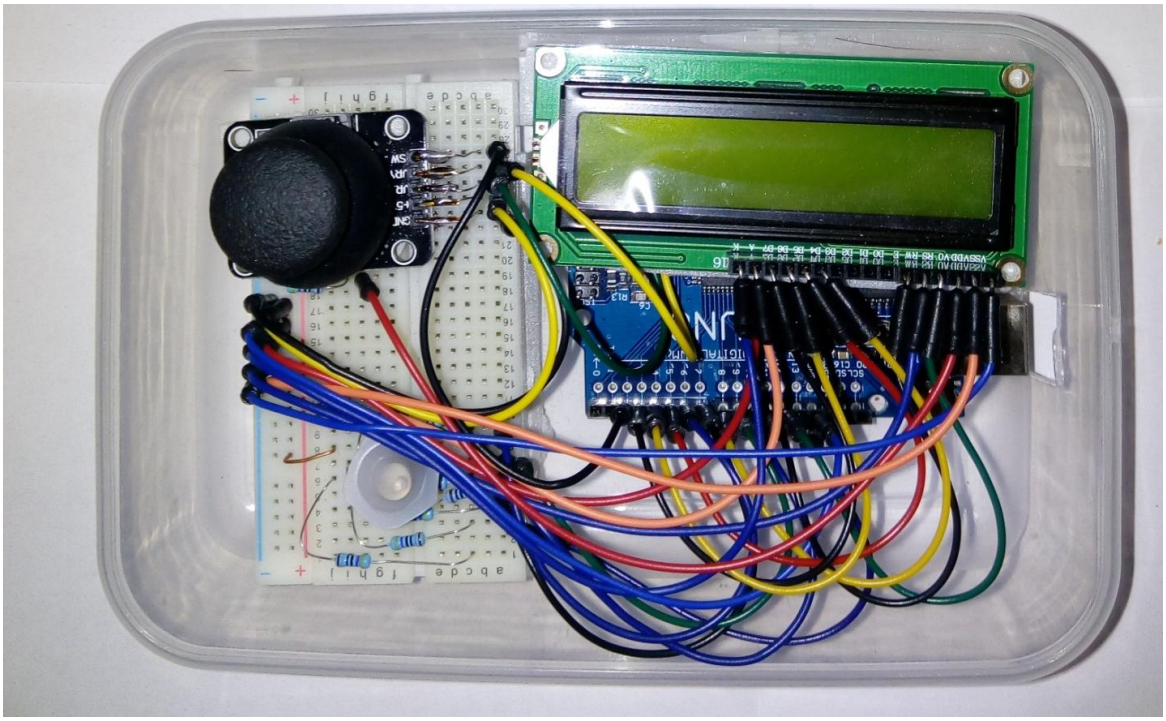
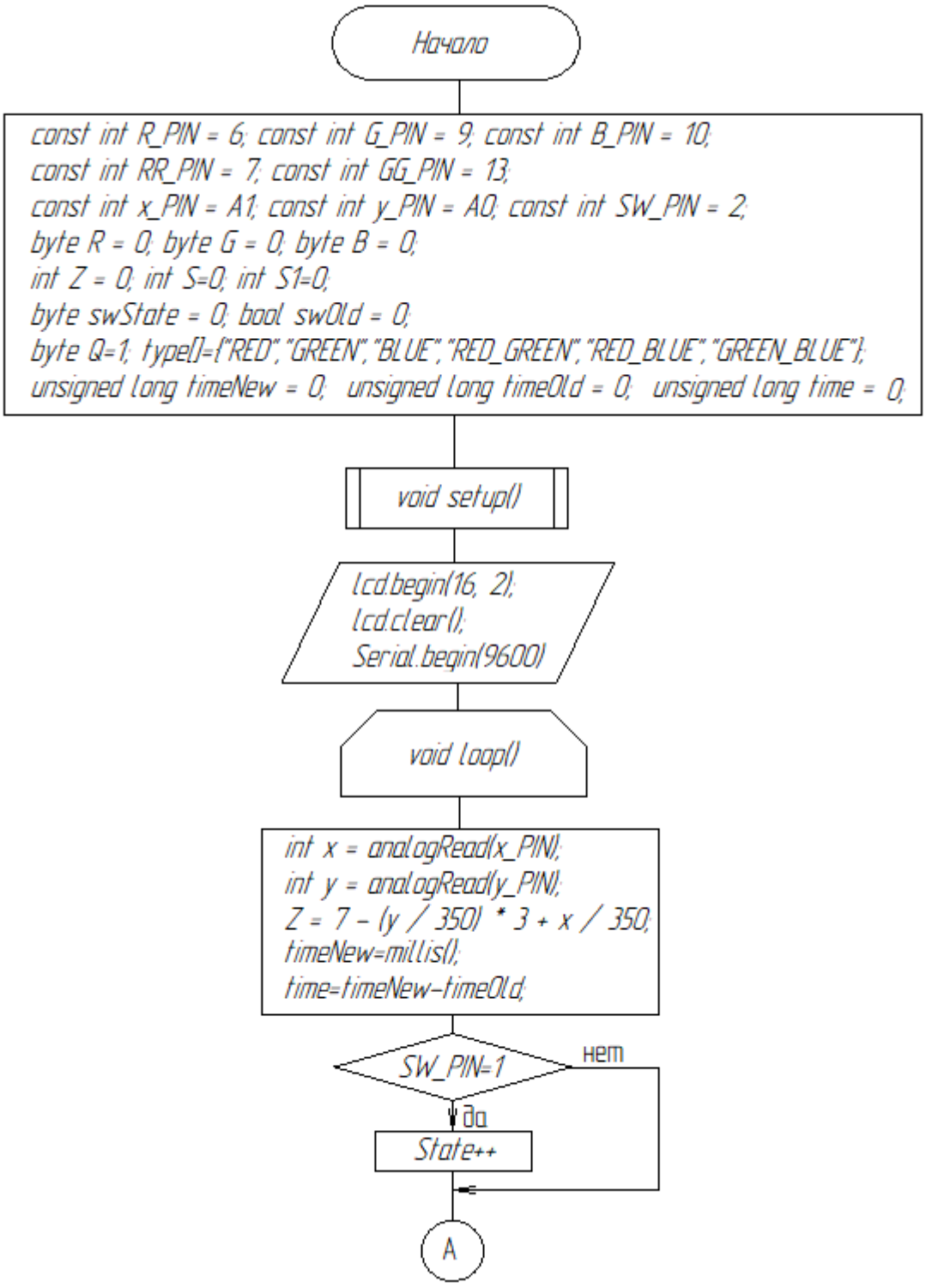


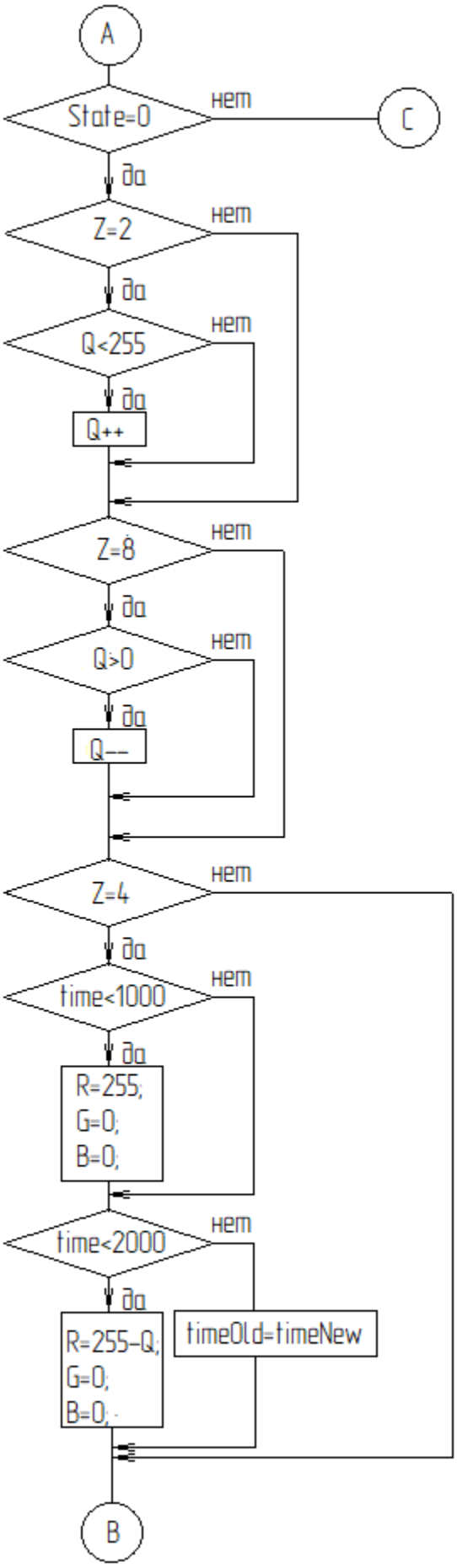
Рисунок 4.1 – Прибор со снятой крышкой

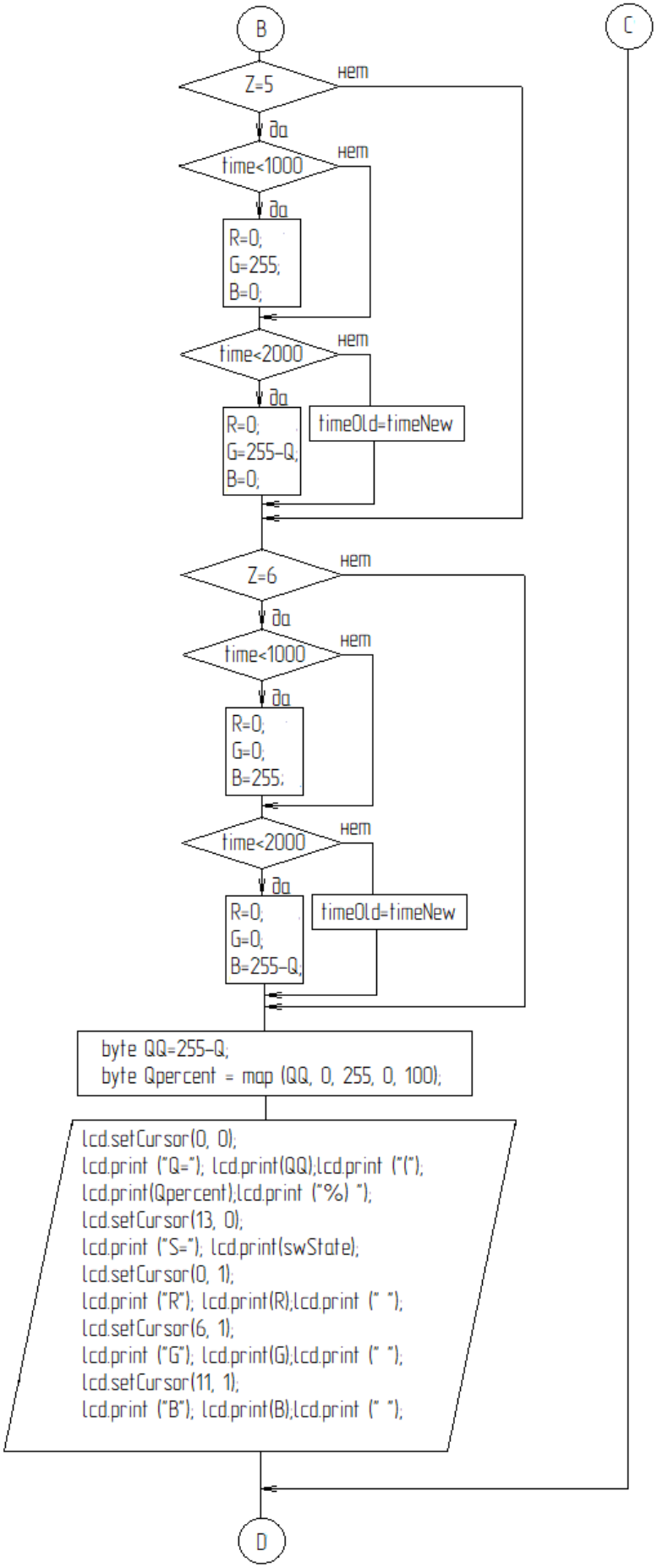
Рисунок 4.2 – Прибор в работающем состоянии

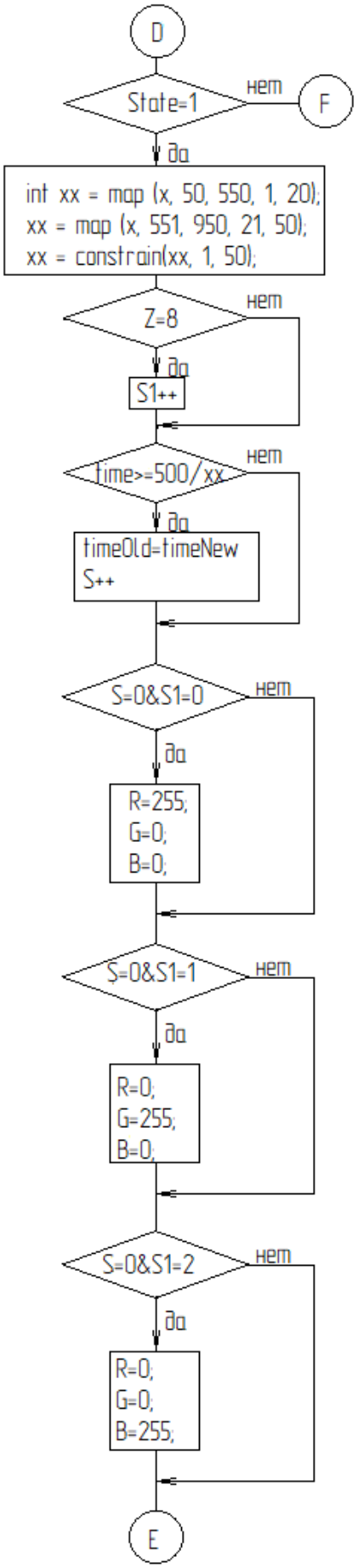
5 Разработка программной части прибора для изучения динамической индикации

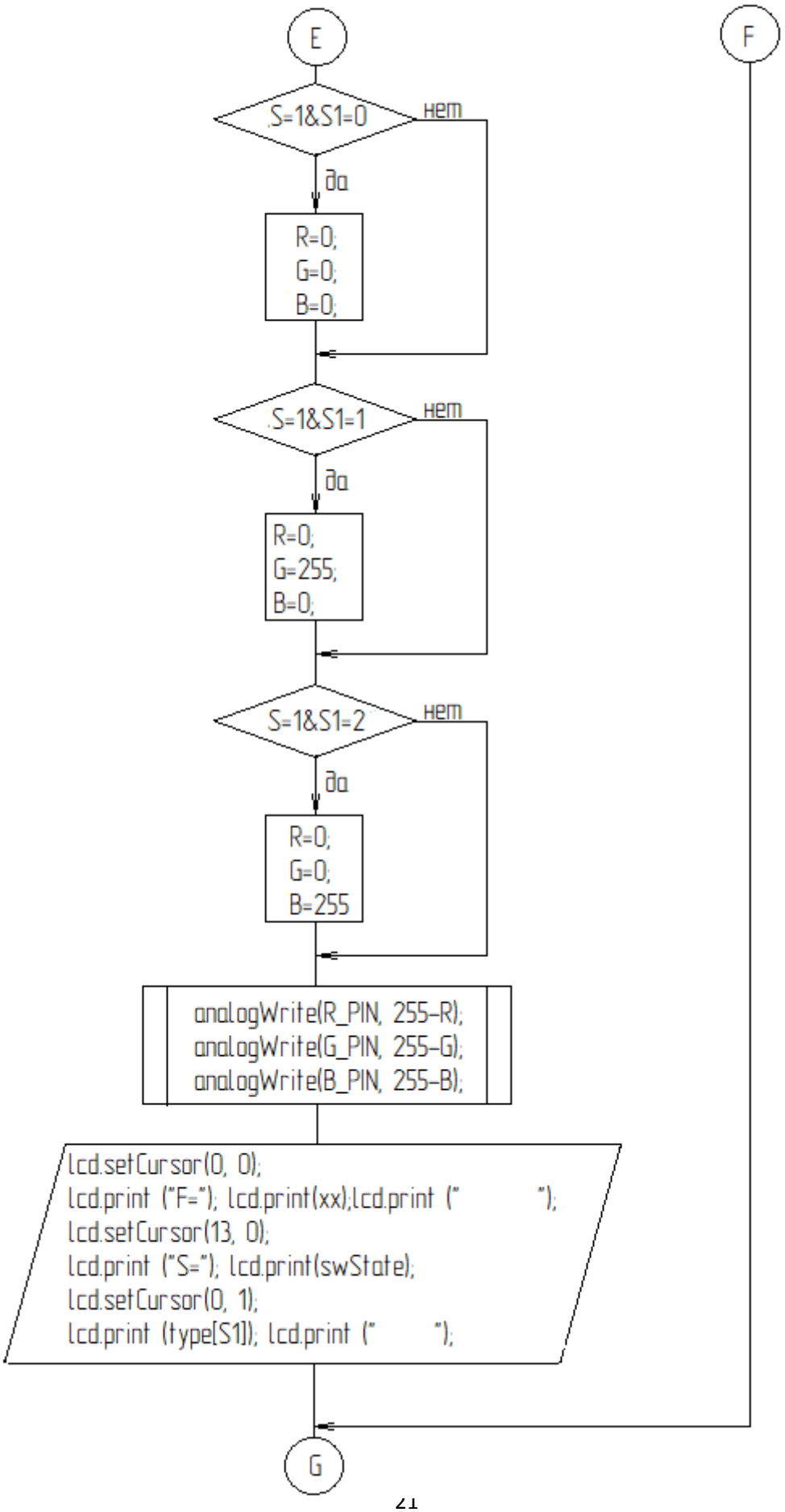
На рисунке 5.1 представлена блок-схема разработанной программы.

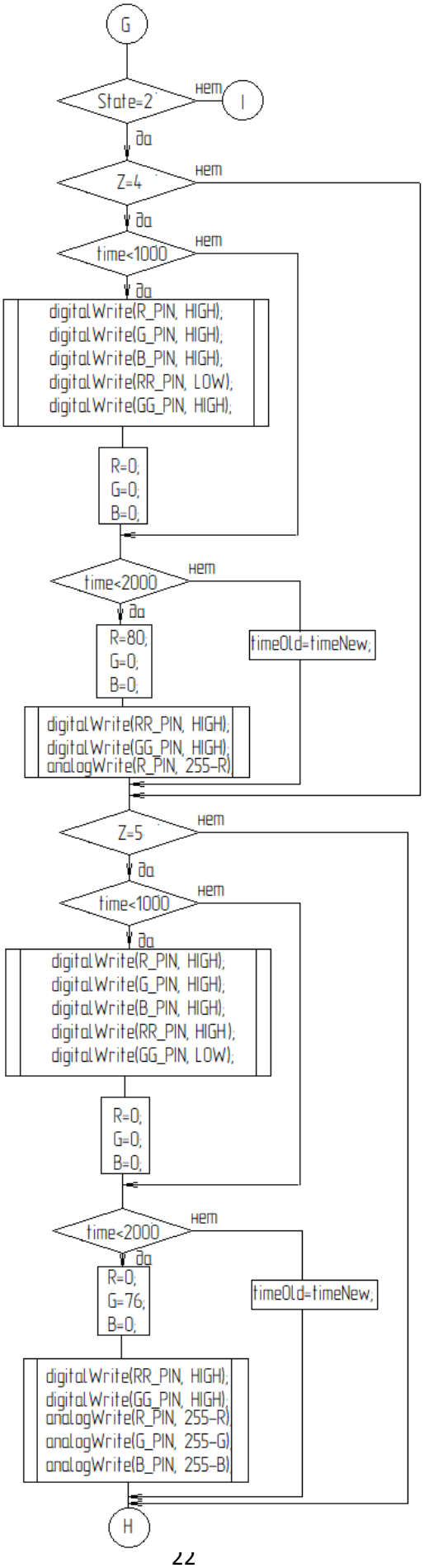



A
State=0
нет
C
да
Z=2
нет
да
Q<255
нет
да
Q++
Z=8
нет
да
Q>0
нет
да
Q--
Z=4
нет
да
time<1000
нет
да
R=255;
G=0;
B=0;
time<2000
нет
да
R=255-Q;
G=0;
B=0;
timeOld=timeNew
B


B
C
Z=5
нет
да
time<1000
нет
да
R=0,
G=255;
B=0,
time<2000
нет
да
R=0;
G=255-Q,
B=0;
timeOld=timeNew
Z=6
нет
да
time<1000
нет
да
R=0,
G=0,
B=255;
time<2000
нет
да
R=0,
G=0,
B=255-Q,
timeOld=timeNew
byte QQ=255-Q;
byte Qpercent = map (QQ, 0, 255, 0, 100);
lcd.setCursor(0, 0);
lcd.print ("Q="); lcd.print(QQ),lcd.print ("(");
lcd.print(Qpercent),lcd.print ("%) ");
lcd.setCursor(13, 0);
lcd.print ("S="); lcd.print(swState);
lcd.setCursor(0, 1);
lcd.print ("R"); lcd.print(R),lcd.print (" ");
lcd.setCursor(6, 1);
lcd.print ("G"); lcd.print(G),lcd.print (" ");
lcd.setCursor(11, 1);
lcd.print ("B"); lcd.print(B),lcd.print (" ");
D


D
State=1
нет
F
да
int xx = map (x, 50, 550, 1, 20);
xx = map (x, 551, 950, 21, 50);
xx = constrain(xx, 1, 50);
Z=8
нет
да
S1++
time>=500/xx
нет
да
timeOld=timeNew
S++
S=0&S1=0
нет
да
R=255;
G=0;
B=0;
S=0&S1=1
нет
да
R=0;
G=255;
B=0;
S=0&S1=2
нет
да
R=0;
G=0;
B=255;
E


E
F
S=1&S1=0
нет
да
R=0;
G=0;
B=0;
S=1&S1=1
нет
да
R=0;
G=255;
B=0;
S=1&S1=2
нет
да
R=0;
G=0;
B=255
analogWrite(R_PIN, 255-R);
analogWrite(G_PIN, 255-G);
analogWrite(B_PIN, 255-B);
lcd.setCursor(0, 0);
lcd.print ("F="); lcd.print(xx);lcd.print (" ");
lcd.setCursor(13, 0);
lcd.print ("S="); lcd.print(swState);
lcd.setCursor(0, 1);
lcd.print (type[S1]); lcd.print (" ");
G


G
State=2
нет
I
да
Z=4
нет
да
time<1000
нет
да
digitalWrite(R_PIN, HIGH);
digitalWrite(G_PIN, HIGH);
digitalWrite(B_PIN, HIGH);
digitalWrite(RR_PIN, LOW);
digitalWrite(GG_PIN, HIGH);
R=0;
G=0;
B=0;
time<2000
нет
да
R=80;
G=0;
B=0;
timeOld=timeNew;
digitalWrite(RR_PIN, HIGH);
digitalWrite(GG_PIN, HIGH);
analogWrite(R_PIN, 255-R);
Z=5
нет
да
time<1000
нет
да
digitalWrite(R_PIN, HIGH);
digitalWrite(G_PIN, HIGH);
digitalWrite(B_PIN, HIGH);
digitalWrite(RR_PIN, HIGH);
digitalWrite(GG_PIN, LOW);
R=0;
G=0;
B=0;
time<2000
нет
да
R=0;
G=76;
B=0;
timeOld=timeNew;
digitalWrite(RR_PIN, HIGH);
digitalWrite(GG_PIN, HIGH);
analogWrite(R_PIN, 255-R);
analogWrite(G_PIN, 255-G);
analogWrite(B_PIN, 255-B);
H

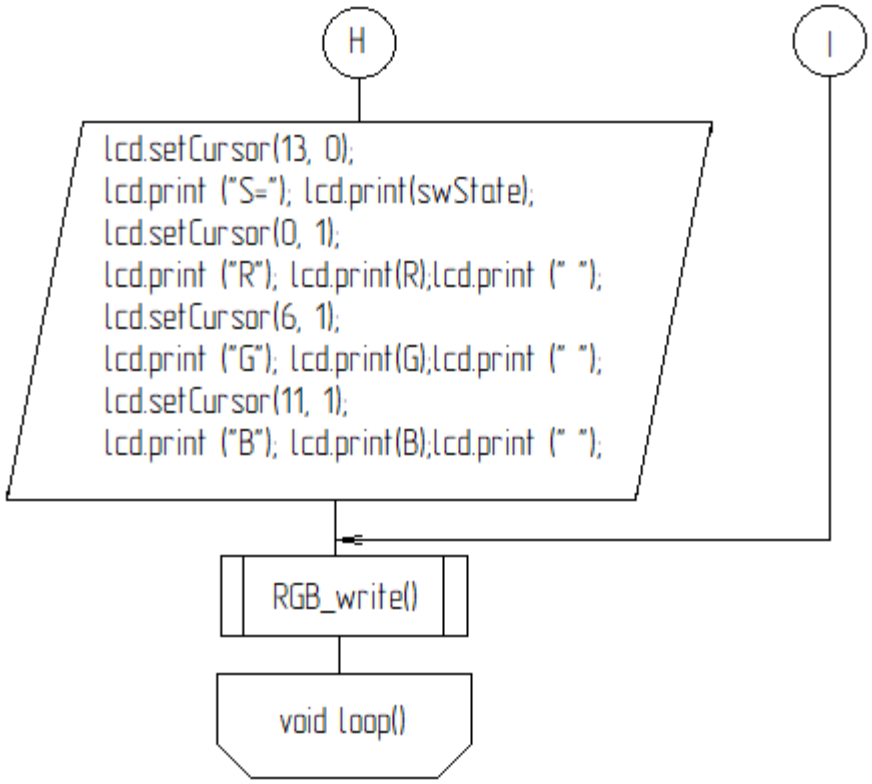


Рисунок 5.1 – Блок-схема программы

Текст программы приведен в конце данного пункта. Программа написана в среде «Ардуино», которую можно скачать с официального сайта [17]. Разработанная программа должна выполнять следующие задачи:

1) Считывать сигнал с джойстика
2) Подавать сигнал на светодиод
3) Выводить данные на дисплей

Программа начинаем с того, что прикрепляем библиотеку Liquid Crystal [18], которая поможет работать с дисплеем.

Далее инициализируются переменные и дисплей. Переменные записывают цифровые или буквенные значения, которые можно использовать обращаясь к ним в алгоритме программы [19].

Для использования переменной, следует ее объявить в начале программы или цикла. Она имеет тип и начальное значение, если указано [20].

Далее идет функция void setup(), выполняющаяся один раз и служащая для установки параметров программы. В ней мы устанавливаем режимы выводов с помощью функции pinMode, включаем подтягивающие резисторы, где нужно, с помощью этой же функции. Задаем режим работы дисплея, на шестнадцать столбцов и две строки, с помощью функции lcd.begin.

В начале функции цикла void loop(), производим вычисления зоны джойстика. Затем считываем нажатие кнопки джойстика, с помощью которого программа будет переключаться между режимами. Далее засекаем время начала цикла. На этом вспомогательные вычисления закончены.

Далее, если кнопка еще не была нажата, то выполняется программа для первого режима. Вычисляется зона, в которой находится ручка управления. Если внизу, то скважность прибавляется, если наверху – то прибавляется. Скважность запоминается, и программа поочередно ее меняет, с выбранной скважности до стопроцентной и обратно, что позволяет глазу оценить изменение. Если ручка управления слева, то эксперимент проводится с красным цветом, если по центру – то с зеленым, если справа – то с синим. Далее следует блок вывода на экран скважности, скважности в процентах, номера эксперимента, и яркости цветов RGB.

При нажатии джойстика режимы исследования переключаются. При втором режиме исследуется динамическая индикация. Цвет для исследования можно выбрать, перемещая ручку вверх.

В начале замеряется время. Благодаря этому, программа с заданной частотой переключает два цвета. Частоту можно регулировать джойстиком от 0 до 50 Гц, перемещая ручку от крайне левого до крайне правого

положения. Далее информация выводится на дисплей: частота, номер режима, используемые цвета.

В третьем опыте, каждую секунду переключаем из импульсного режима работы в постоянный. Для этого в начале сверяем время и определяем положение джойстика. Далее включаем выводы с нужным сопротивлением, и диод с нужной яркостью. Далее сверяем время, если секунда прошла, то переключаем яркость диода и цифровой вывод с сопротивление. Выводим режим работы, режим исследования и значение сопротивления на дисплей.

Этот режим используется для двух цветов: красный и зеленый, которые можно переключать джойстиком.

Текст программы для изучения динамической индикации.

```
#include <LiquidCrystal.h> // для подключения дисплея контакты на Arduino в порядке:
LiquidCrystal lcd(12, 11, 8, 5, 4, 3); // RS, E, DB4, DB5, DB6, DB7

const int R_PIN = 6;
const int G_PIN = 9;
const int B_PIN = 10;
const int RR_PIN = 7;
const int GG_PIN = 13;

const int x_PIN = A1;
const int y_PIN = A0;
const int SW_PIN = 2; //соответствует прерыванию 0

byte R = 0;
byte G = 0;
```

```
byte B = 0;

int Z = 0; //номер зоны, задаваемый джостиком. от 1 (левый верхний) до 9 (правый нижний)
int S=0;
int S1=0;
byte swState = 0; //состояние, переключаемое кнопкой =режим управления цветом
bool swOld = 0; //предыдущее значение кнопки
byte Q=1; //в первом опыте изменение яркости
char* type[]={"RED","GREEN","BLUE","RED_GREEN","RED_BLUE","GREEN_BLUE"};

unsigned long timeNew = 0; //текущее значение времени
unsigned long timeOld = 0; //значение времени, когда переключали светодиод.
unsigned long time = 0;

void setup() {
  pinMode(R_PIN, OUTPUT);
  pinMode(G_PIN, OUTPUT);
  pinMode(B_PIN, OUTPUT);
  pinMode(x_PIN, INPUT);
  pinMode(y_PIN, INPUT);
  pinMode(GG_PIN, OUTPUT);
  pinMode(RR_PIN, OUTPUT);

  pinMode(SW_PIN, INPUT_PULLUP);//Включение подтягивающего резистора
```

```
  lcd.begin(16, 2);// устанавливаем размер (количество столбцов и строк)
  lcd.clear();  //очистка экрана (чтоб стереть не нужные цифры)
  Serial.begin(9600);
 }

 void loop() {
  int x = analogRead(x_PIN);
  int y = analogRead(y_PIN);
  Z = 7 - (y / 350) * 3 + x / 350;

  bool sw = !digitalRead(SW_PIN); //чтение кнопки, подключенной к земле
  if (sw && !swOld) { //нажатие = если кнопка не была нажата, а сейчас нажата
   swState = ++swState % 3; // 0->1->2->3->4->0 //по нажатию кнопки инвертируем состояние swState
   delay (100);//подавление дребезга
  }  swOld = sw;

  timeNew=millis();
  time=timeNew-timeOld;

 if (swState == 0) {
  pinMode(RR_PIN, INPUT);
  pinMode(GG_PIN, INPUT);
  pinMode(R_PIN, OUTPUT);
  pinMode(G_PIN, OUTPUT);
  pinMode(B_PIN, OUTPUT);
   if (Z==2) {
    if (Q<255){
```

```
    Q++;
    delay(100);
  }
}

if (Z==8) {
  if (Q>0){
    Q--;
    delay(100);
  }
}

if (Z==4) {
  if (time<1000){
    R=255;
    G=0;
    B=0;
  } else if (time<2000){
    R=(255-Q);
    G=0;
    B=0;
  } else{
    timeOld=timeNew;
  }
}

 if (Z==5) {
  if (time<1000){
    G=255;
    R=0;
```

```
    B=0;
   } else if (time<2000){
    G=(255-Q);
    R=0;
    B=0;
   } else{
    timeOld=timeNew;
   }
  }

   if (Z==6) {
    if ((time)<1000){
     B=255;
     R=0;
     G=0;
    } else if (time<2000){
     B=(255-Q);
     R=0;
     G=0;
    } else{
     timeOld=timeNew;
    }
   }

byte QQ=255-Q;
byte Qpercent = map (QQ, 0, 255, 0, 100);
lcd.setCursor(0, 0);
lcd.print ("Q="); lcd.print(QQ);lcd.print ("(");
lcd.print(Qpercent);lcd.print ("%) ");
lcd.setCursor(13, 0);
```

```
lcd.print ("S="); lcd.print(swState);
lcd.setCursor(0, 1);
lcd.print ("R"); lcd.print(R);lcd.print (" ");
lcd.setCursor(6, 1);
lcd.print ("G"); lcd.print(G);lcd.print (" ");
lcd.setCursor(11, 1);
lcd.print ("B"); lcd.print(B);lcd.print (" ");
  }

 if (swState == 1) {
  pinMode(RR_PIN, INPUT);
  pinMode(GG_PIN, INPUT);
  pinMode(R_PIN, OUTPUT);
  pinMode(G_PIN, OUTPUT);
  pinMode(B_PIN, OUTPUT);

  int xx = map (x, 50, 550, 1, 20);
  xx = map (x, 551, 950, 21, 50);
  xx = constrain(xx, 1, 50);

    if (Z==8){
      S1=++S1%6;
      delay(200);
    }

    if (time>=(500/xx)) {
      timeOld = timeNew;
      S=++S%2;
    }
```

```
if (S==0&S1==0) {
  R=255;
  G=0;
  B=0;
}
if (S==0&S1==1) {
  R=0;
  G=255;
  B=0;
}
if (S==0&S1==2) {
  R=0;
  G=0;
  B=255;
}
if (S==0&S1==3) {
  R=255;
  G=0;
  B=0;
}
if (S==0&S1==4) {
  R=255;
  G=0;
  B=0;
}
if (S==0&S1==5) {
  R=0;
  G=255;
  B=0;
}
```

```
if (S==1&S1==0){
   R=0;
   G=0;
   B=0;
 }
if (S==1&S1==1){
   R=0;
   G=0;
   B=0;
 }
if (S==1&S1==2){
   R=0;
   G=0;
   B=0;
 }
if (S==1&S1==3){
   R=0;
   G=255;
   B=0;
 }
if (S==1&S1==4){
   R=0;
   G=0;
   B=255;
 }
if (S==1&S1==5){
   R=0;
   G=0;
   B=255;
```

```
    }
  analogWrite(R_PIN, 255-R);
  analogWrite(G_PIN, 255-G);
  analogWrite(B_PIN, 255-B);

lcd.setCursor(0, 0);
lcd.print ("F="); lcd.print(xx);lcd.print ("        ");
lcd.setCursor(13, 0);
lcd.print ("S="); lcd.print(swState);
lcd.setCursor(0, 1);
lcd.print (type[S1]); lcd.print ("      ");
  }

  if (swState == 2) {
    if (Z==4) {
      if (time<1000){
        pinMode(RR_PIN, OUTPUT);
        pinMode(GG_PIN, OUTPUT);
        pinMode(R_PIN, INPUT);
        pinMode(G_PIN, INPUT);
        pinMode(B_PIN, INPUT);
        digitalWrite(R_PIN, HIGH);
        digitalWrite(G_PIN, HIGH);
        digitalWrite(B_PIN, HIGH);
        digitalWrite(RR_PIN, LOW);
        digitalWrite(GG_PIN, HIGH);
        R=0;
        G=0;
        B=0;
      } else if (time<2000){
```

```
    pinMode(RR_PIN, INPUT);
    pinMode(GG_PIN, INPUT);
    pinMode(R_PIN, OUTPUT);
    pinMode(G_PIN, OUTPUT);
    pinMode(B_PIN, OUTPUT);
    digitalWrite(RR_PIN, HIGH);
    digitalWrite(GG_PIN, HIGH);
    R=80;
    G=0;
    B=0;
    analogWrite(R_PIN, 255-R);
  } else{
    timeOld=timeNew;
  }
}

if (Z==5) {
  if (time<1000){
    pinMode(RR_PIN, OUTPUT);
    pinMode(GG_PIN, OUTPUT);
    pinMode(R_PIN, INPUT);
    pinMode(G_PIN, INPUT);
    pinMode(B_PIN, INPUT);
    digitalWrite(R_PIN, HIGH);
    digitalWrite(G_PIN, HIGH);
    digitalWrite(B_PIN, HIGH);
    digitalWrite(RR_PIN, HIGH);
    digitalWrite(GG_PIN, LOW);
    R=0;
    G=0;
```

```
        B=0;
       } else if (time<2000){
        pinMode(RR_PIN, INPUT);
        pinMode(GG_PIN, INPUT);
        pinMode(R_PIN, OUTPUT);
        pinMode(G_PIN, OUTPUT);
        pinMode(B_PIN, OUTPUT);
        digitalWrite(RR_PIN, HIGH);
        digitalWrite(GG_PIN, HIGH);
        R=0;
        G=76;
        B=0;
            analogWrite(R_PIN, 255-R);
            analogWrite(G_PIN, 255-G);
            analogWrite(B_PIN, 255-B);
      } else{
        timeOld=timeNew;
      }
    }


lcd.setCursor(0, 0);
  if (time<1000) {
    if (Z==4) {
      lcd.print ("R=565 CONST");
    } else if (Z==5) {
      lcd.print ("R=468 CONST  ");
    }
  }else{
    if (Z==4) {
```

```
      lcd.print ("R=141 IMP  ");
    }else if (Z==5){
      lcd.print ("R=100 IMP  ");
    }
  }
lcd.setCursor(13, 0);
lcd.print ("S="); lcd.print(swState);
lcd.setCursor(0, 1);
lcd.print ("R"); lcd.print(R);lcd.print (" ");
lcd.setCursor(6, 1);
lcd.print ("G"); lcd.print(G);lcd.print (" ");
lcd.setCursor(11, 1);
lcd.print ("B"); lcd.print(B);lcd.print (" ");
  }
RGB_Write ();
}

void RGB_Write () {
  analogWrite(R_PIN, 255-R);
  analogWrite(G_PIN, 255-G);
  analogWrite(B_PIN, 255-B);
}
```

6 Экспериментальные исследования динамической индикации

Были проведены несколько опытов. В первом опыте оценивалась способность глаза замечать малые изменения яркости. Для этого изменялась скважность ($Q=T/t_и$) ШИМ сигнала (Рисунок 6.1). Для глаза было заметно изменение скважности на 5 % не зависимо от того, какой цвет рассматривался.

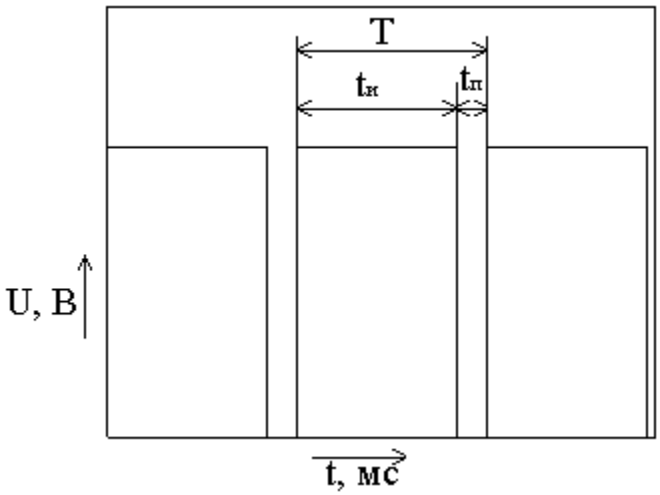


Рисунок 6.1 – ШИМ сигнал, подаваемый на светодиод в первом опыте.

Во втором опыте оценивалось восприятие мерцания светодиода с разными частотами. Человеческий глаз перестает видеть мерцание на частоте 50 Гц и выше. На частоте 25 Гц начинают сливаться цвета, например переключение красного и зеленого на этой частоте уже воспринимаются как желтый.

В третьем опыте сравнивалась яркость светодиода при протекании через него постоянного тока с яркостью при импульсном токе большей амплитуды (ШИМ). Скважность ШИМ и сопротивления подбирались так, чтобы средний ток был одинаков в обоих случаях (Рисунок 6.2). Разница была заметна. Красный цвет при ШИМ-сигнале более яркий, чем при постоянном сигнале. Зеленый цвет наоборот, менее яркий при ШИМ-сигнале, и при этом немного меняется его оттенок.

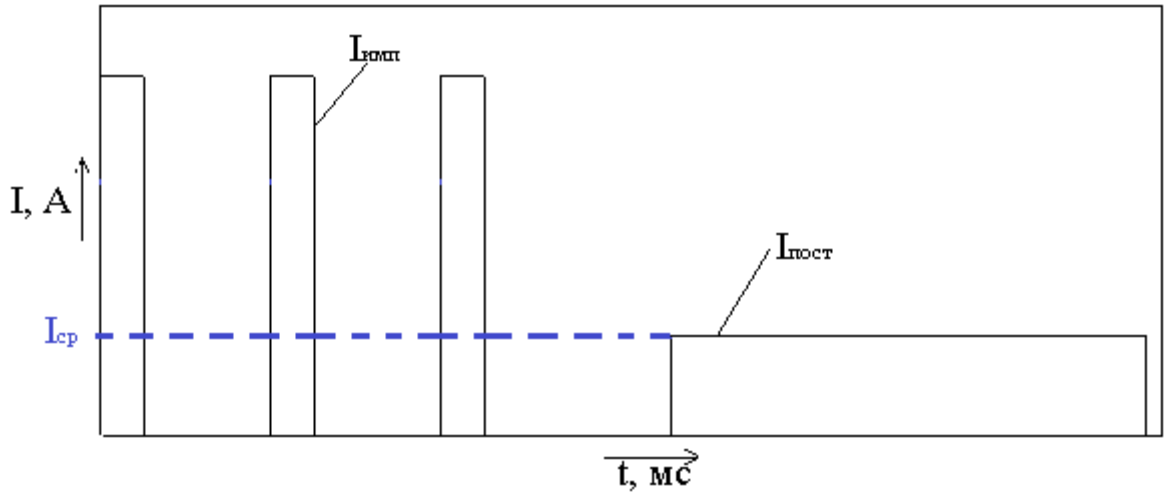


Рисунок 6.2 – Ток светодиода в третьем опыте

7 Разработка структурной схемы электронной игры в шашки

Структурная схема прибора для игры в шашки приведена на рисунке 7.1.

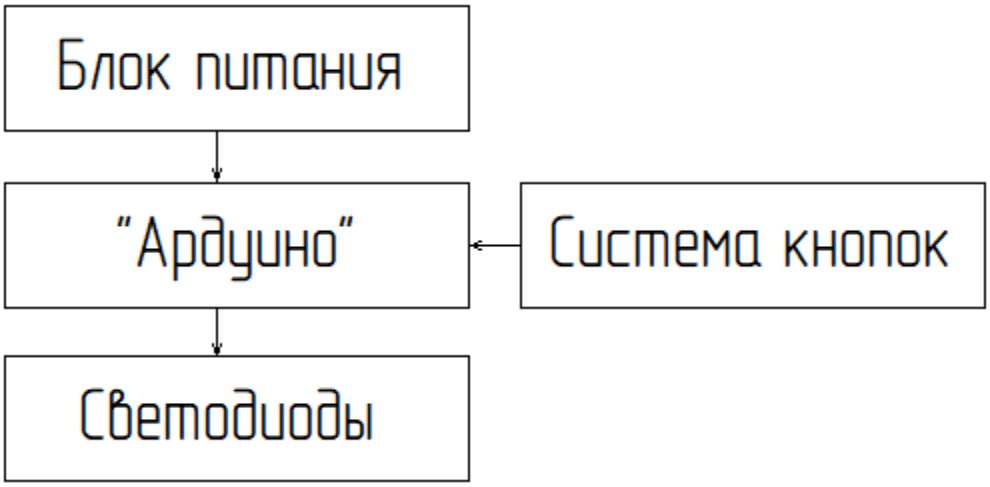


Рисунок 7.1 – Структурная схема прибора

Прибор состоит из блока питания, платы «Ардуино», системы кнопок и индивидуально адресуемых светодиодов.

8 Разработка электрической схемы

Главным управляющим элементом схемы является плата «Ардуино». От платы подается питание на первый светодиод. Далее все 32 светодиода подключаются последовательно друг к другу [21]. Управляются они цифровым выводом от «Ардуино».

8 цифровых выводов платы формируют ряды, к которым присоединяются по 4 ключа одним из контактов [22]. Вторым контактом ключи присоединяются к одному из четырех аналоговых выводов, что формирует матрицу из ключей.

Разработанная схема приведена на рисунке 8.1.

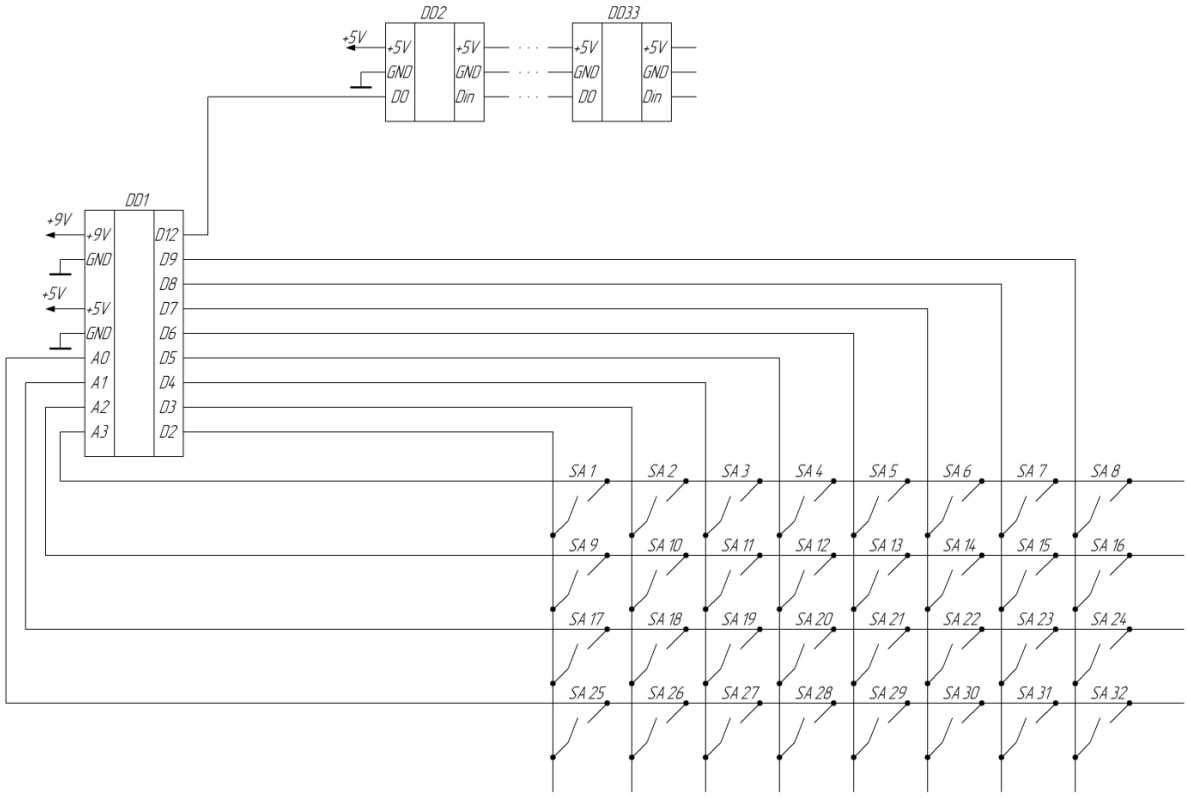


Рисунок 8.1 - Схема шашек

Для автономного питания платы, без проводов и внешних источников питания, используется телефонный аккумулятор. Аккумулятор заряжается от платы, представленной на рисунке 8.2 [23].

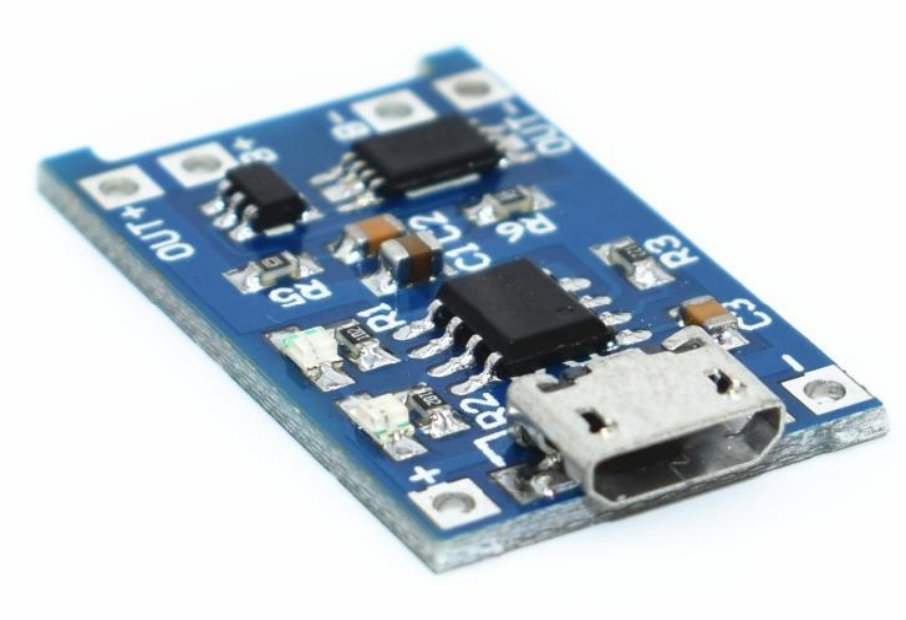


Рисунок 8.2 – Плата для заряда аккумулятора

Эта плата подключается к сети через провод Micro USB. К выходам OUT+, OUT- этой платы, подключается регулятор напряжения, представленный на рисунке 8.3 [24].

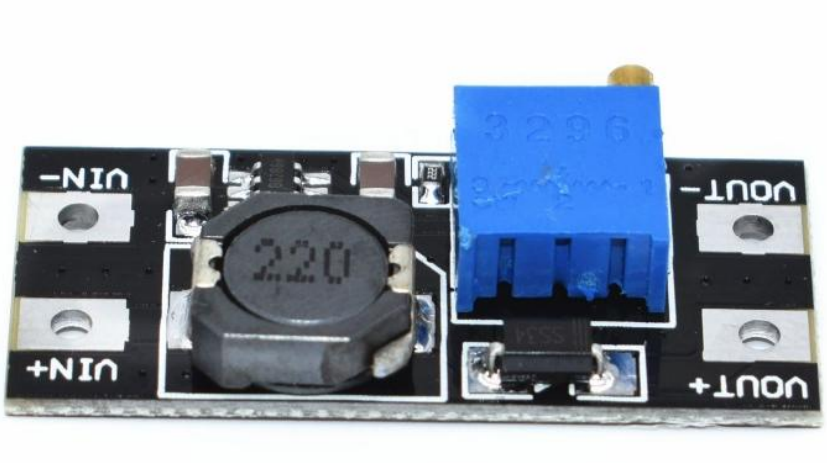


Рисунок 8.3 – Регулятор напряжения

На регуляторе устанавливается напряжение, равное 5В, необходимое для питания платы «Ардуино». Плата подключается к выводам VOUT+, VOUT-. Схема блока питания шашек приведена на рисунке 8.4.

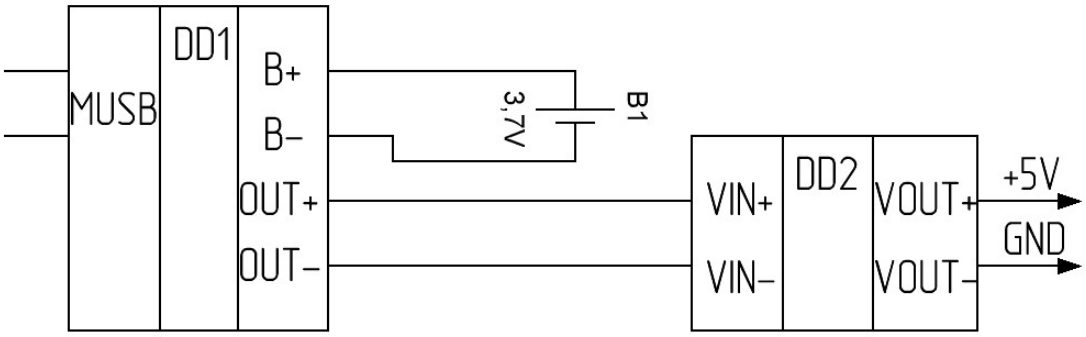


Рисунок 8.4 – Блок питания для шашек

Итоговая схема прибора показана на рисунке 8.5.

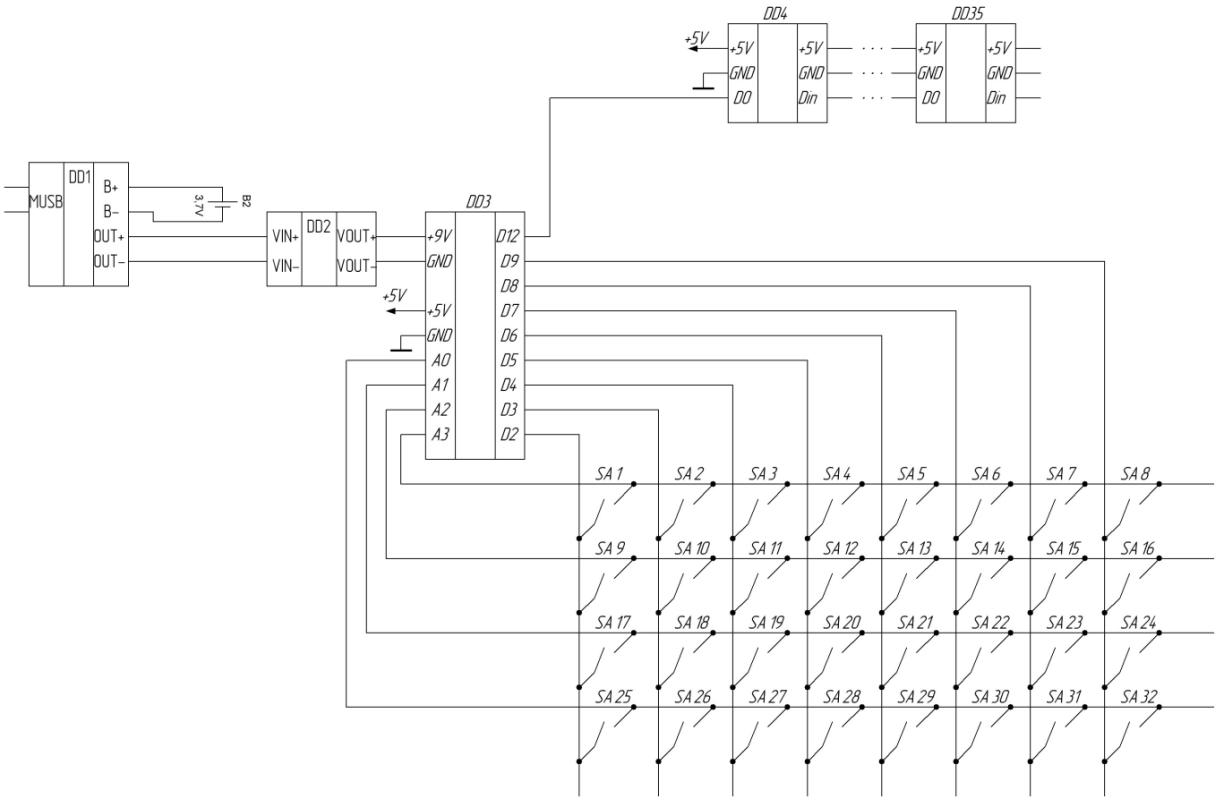


Рисунок 8.5 – Электронная игра в шашки

9 Разработка конструкции

Были закуплены: 32 кнопки, светодиодная лента с индивидуально адресуемыми диодами, текстолитовая плата [25], плата «Ардуино Uno».

В первую очередь, на плату припаяли 32 светодиода, и соединили их в нужном порядке медной проволокой (Рисунок 9.1, 9.2).

Рисунок 9.1 – Плата, вид сверху

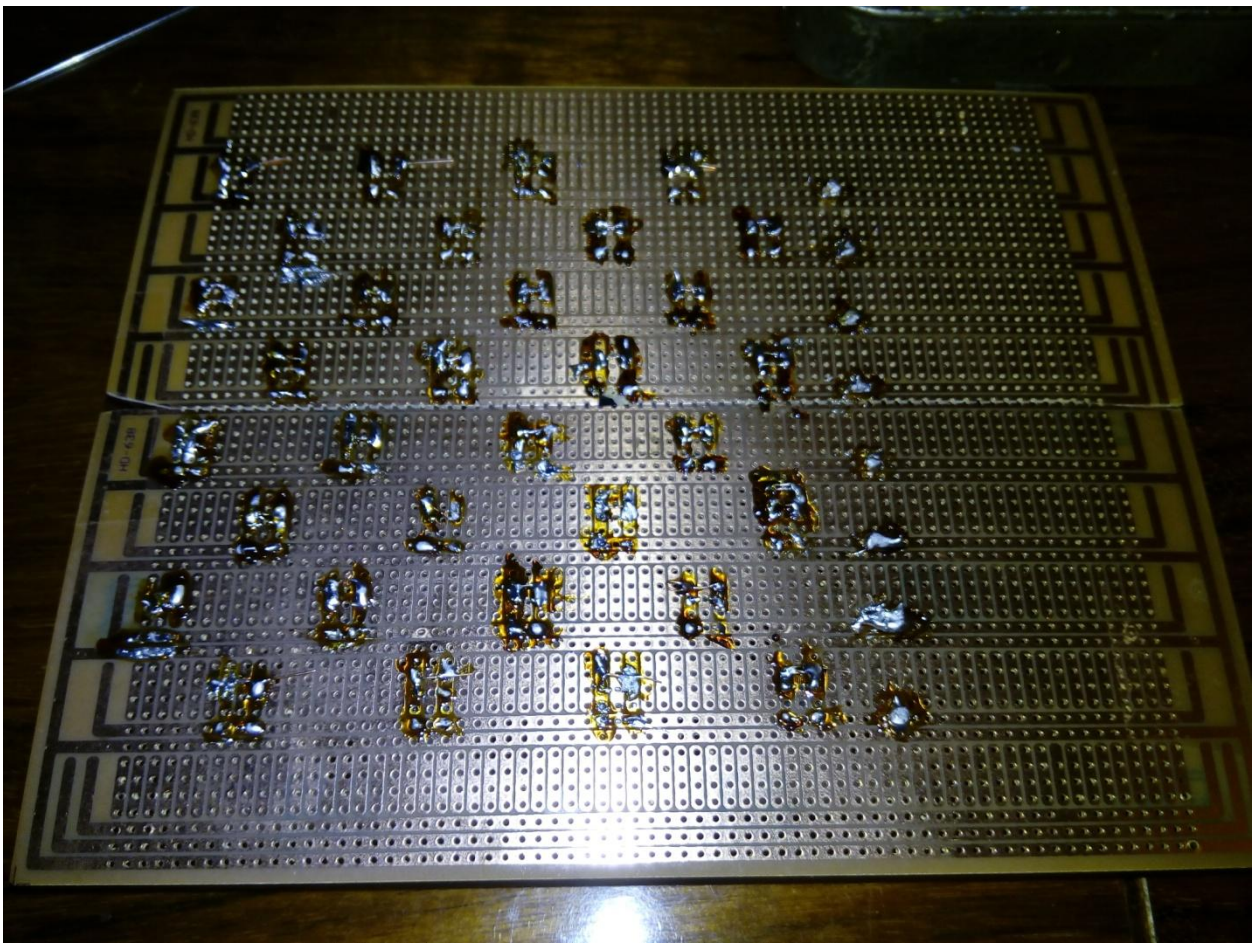


Рисунок 9.2 – Плата, вид снизу

Далее разрезали светодиодную ленту на кусочки, включающие в себя по 4 диода. Разместили их сверху, на пенопластовой опоре, с помощью двухстороннего скотча. После чего спаяли их последовательно друг за другом (Рисунок 9.3).

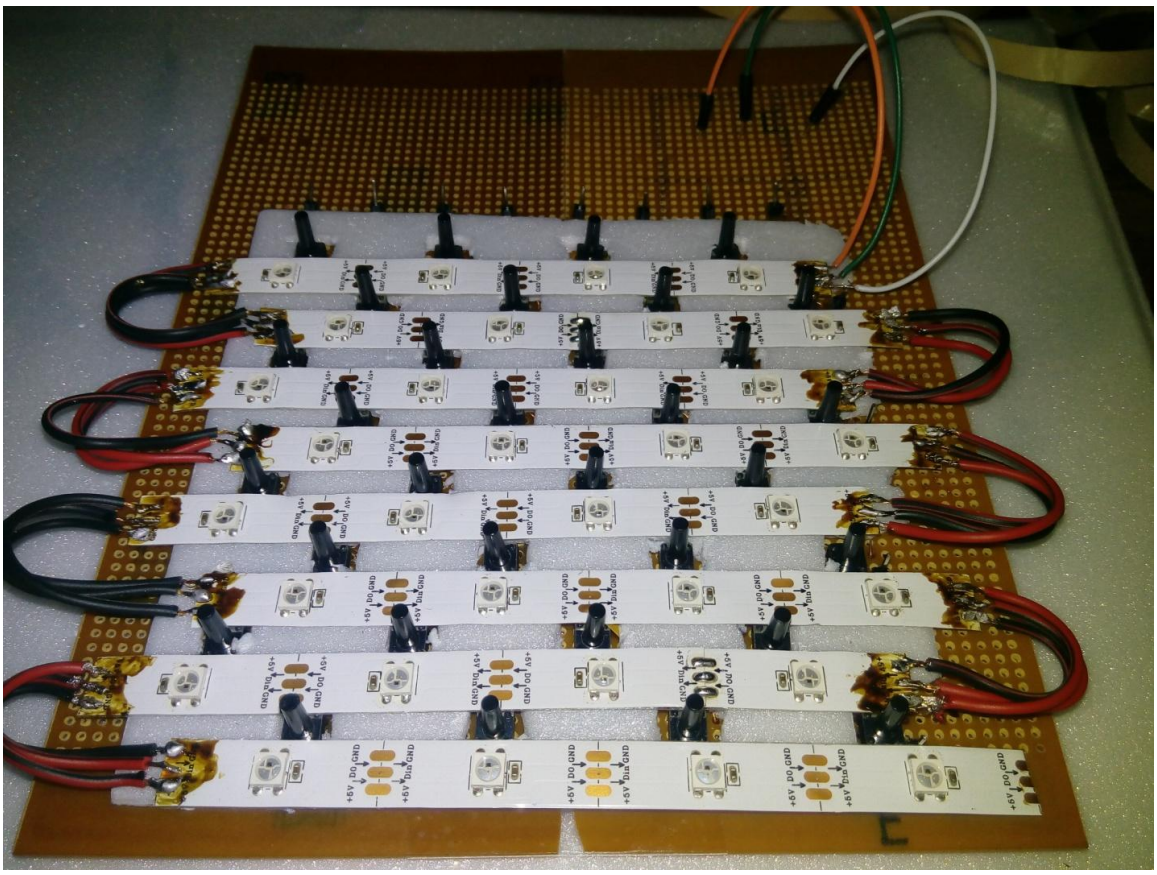


Рисунок 9.3 – Плата с диодами

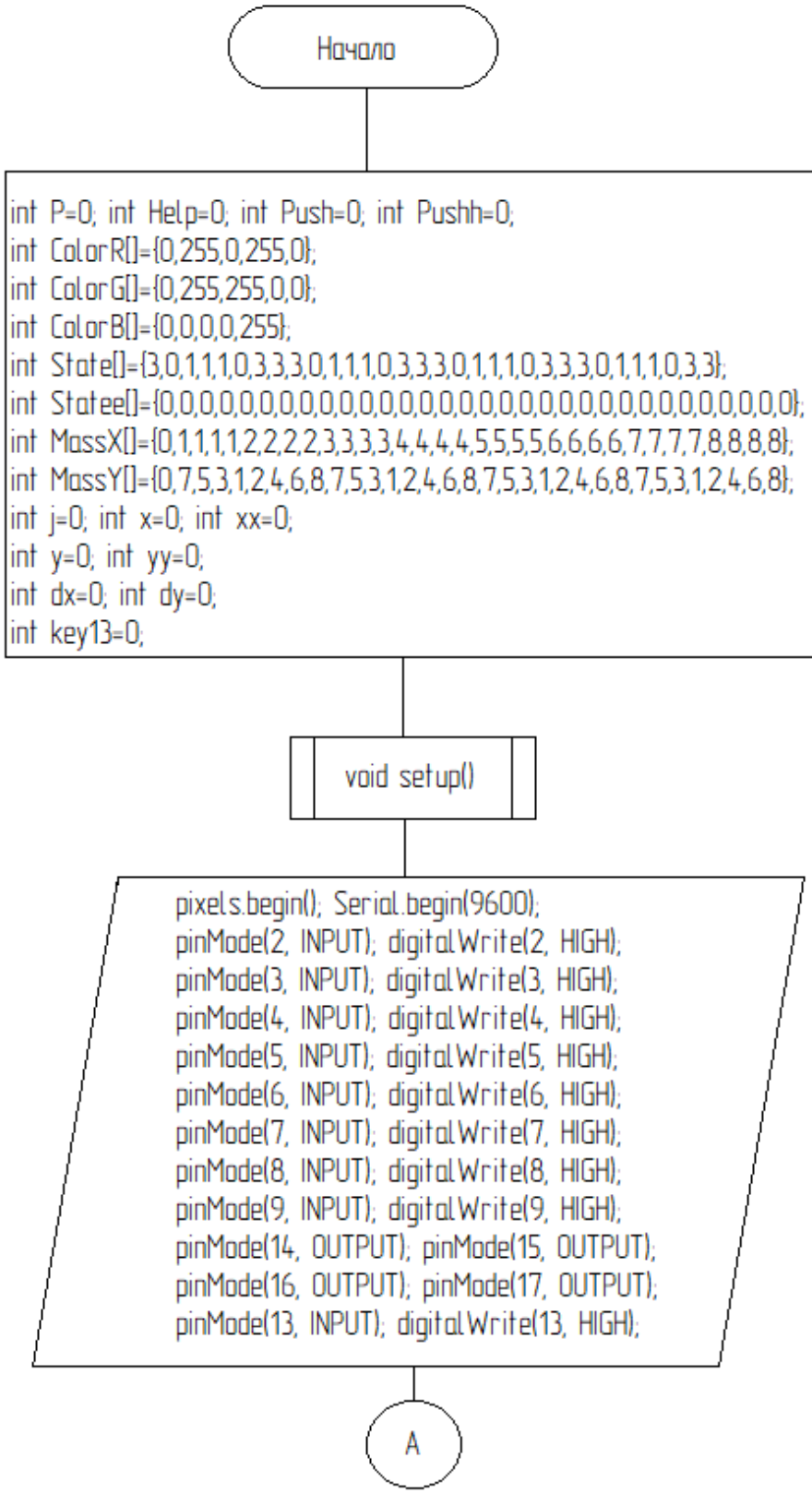
На свободном месте располагается плата «Ардуино». В качестве рассеивателя выбран белый пенопласт. Его вырезали по форме поля. И проделали отверстия для кнопок. Сверху накрыли светонепроницаемым картоном, с отверстиями для кнопок и света в нужных местах.

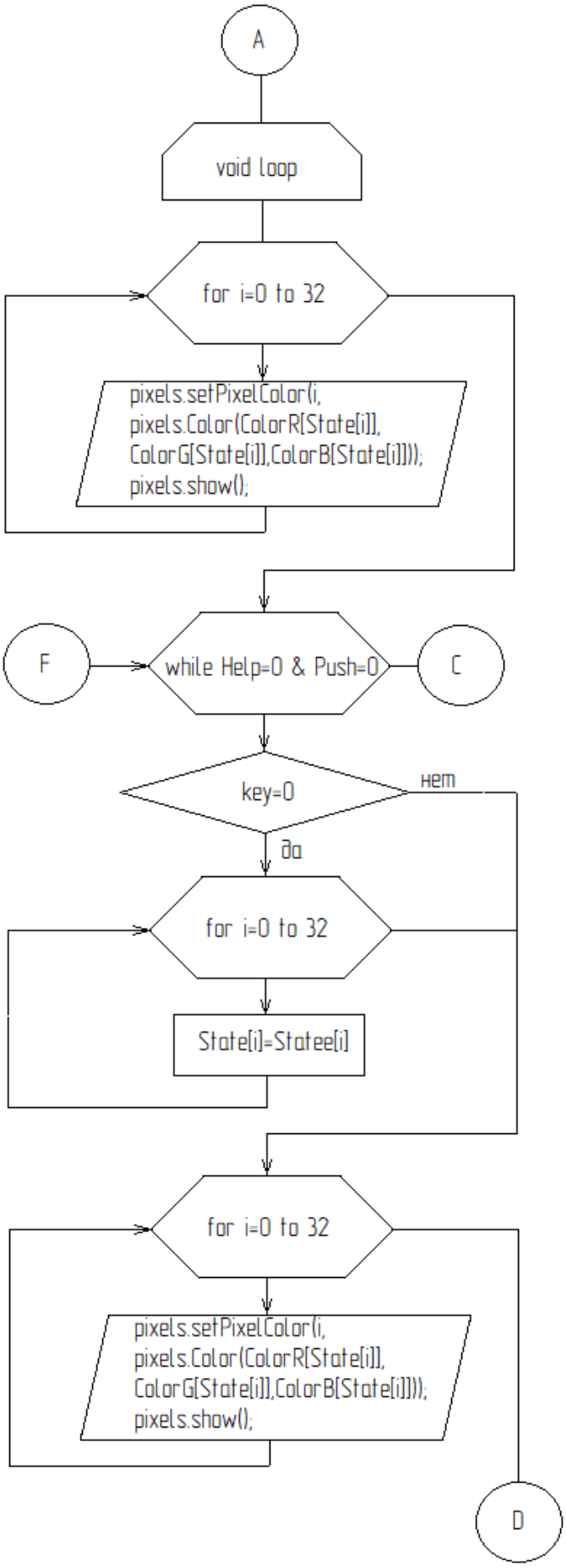
Дно и коробка сделаны из пенопласта, т.к. этот материал просто вырезать под нужную форму. Итоговая конструкция приведена на рисунке 9.4.

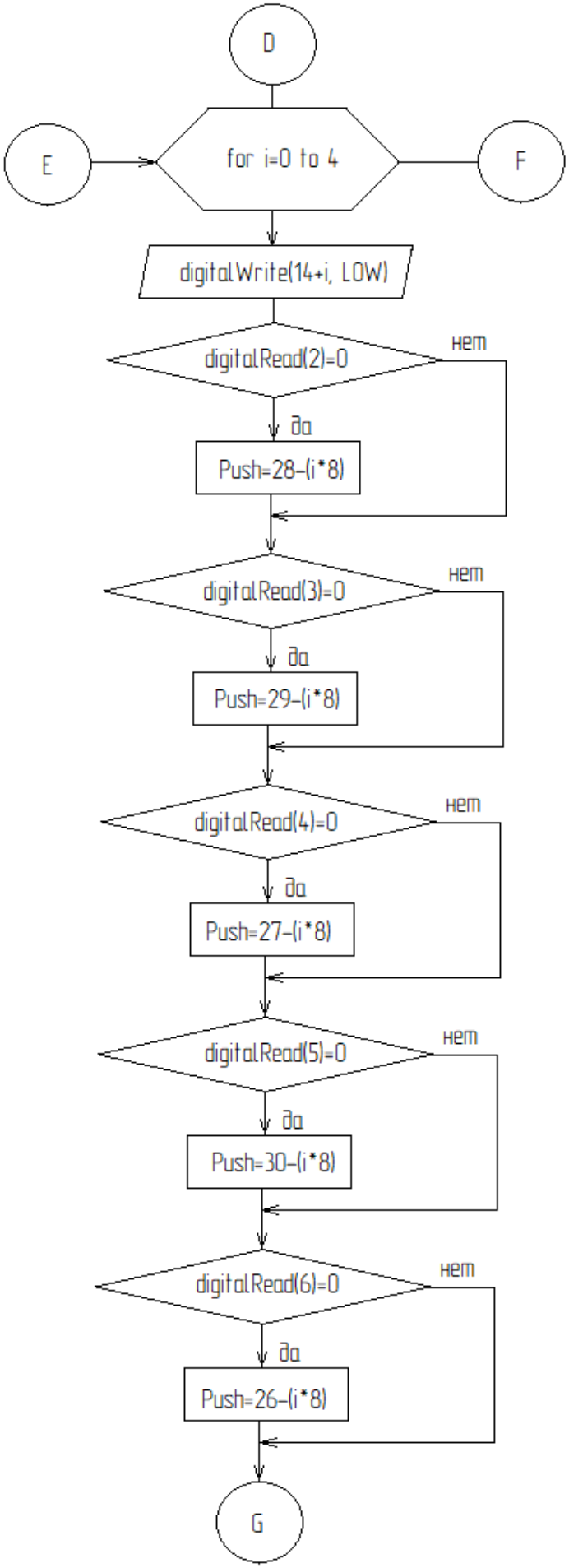
Рисунок 9.4 – Конструкция шашек

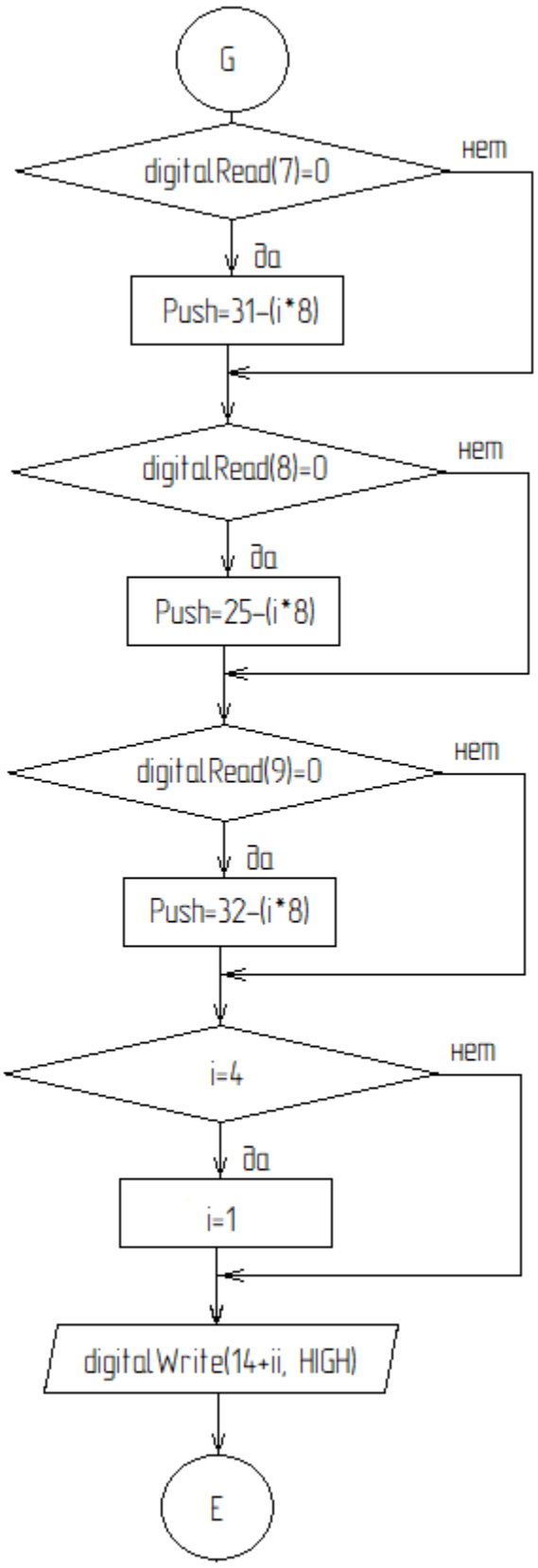
10 Разработка программы

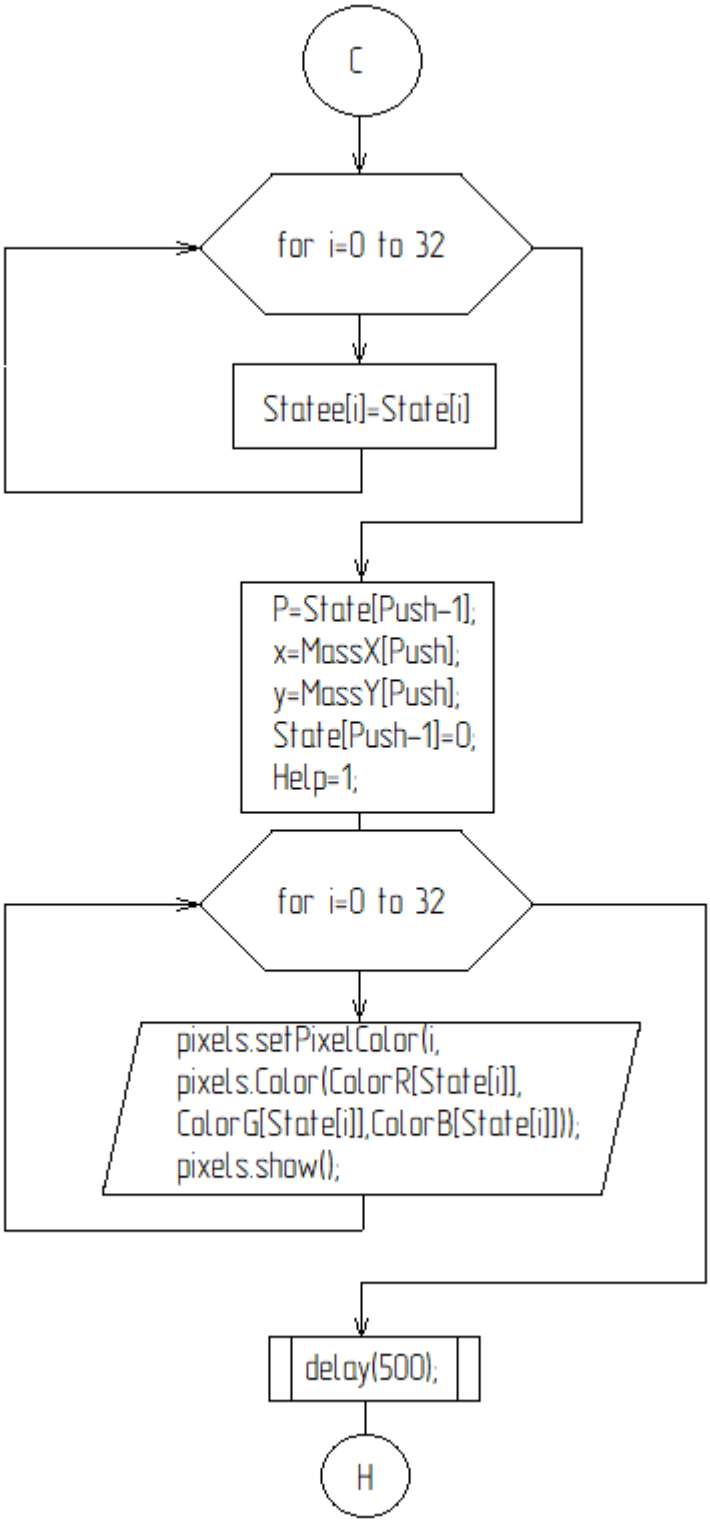
На рисунке 10.1 представлена блок-схема разработанной программы.

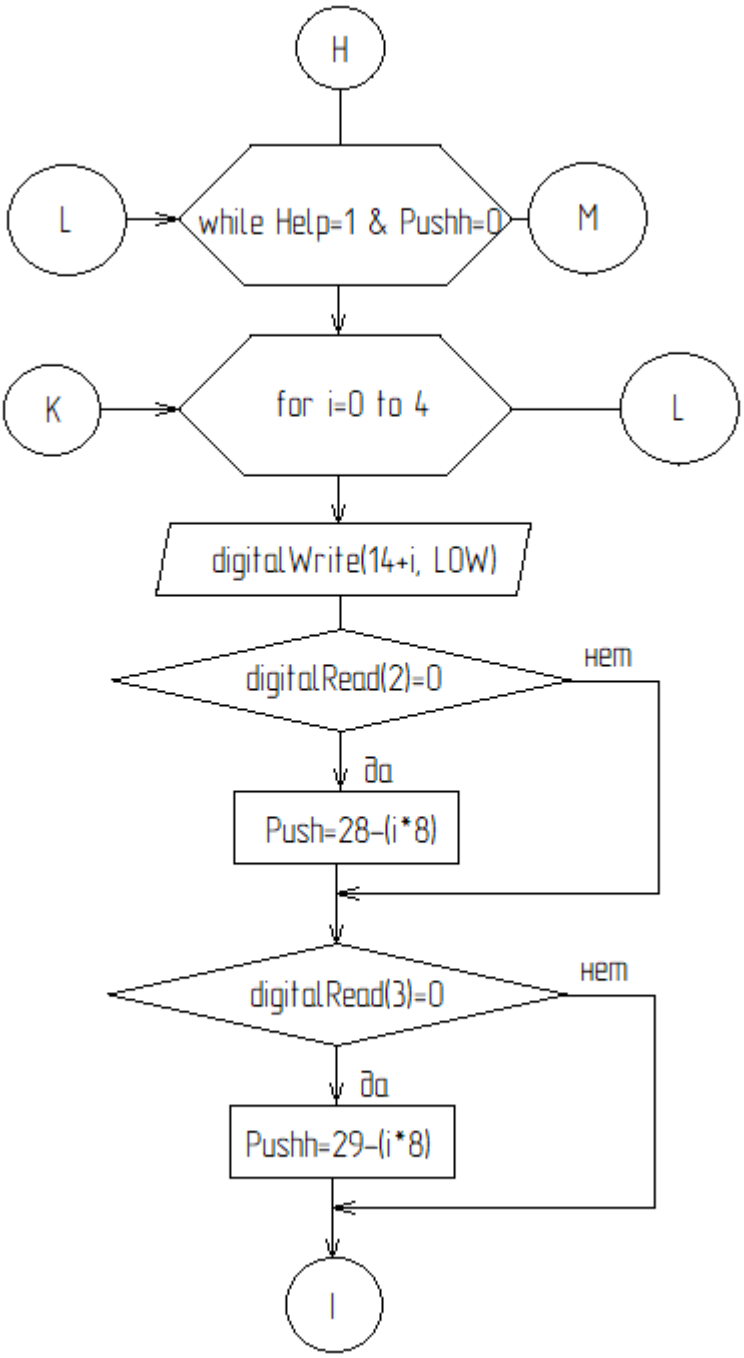


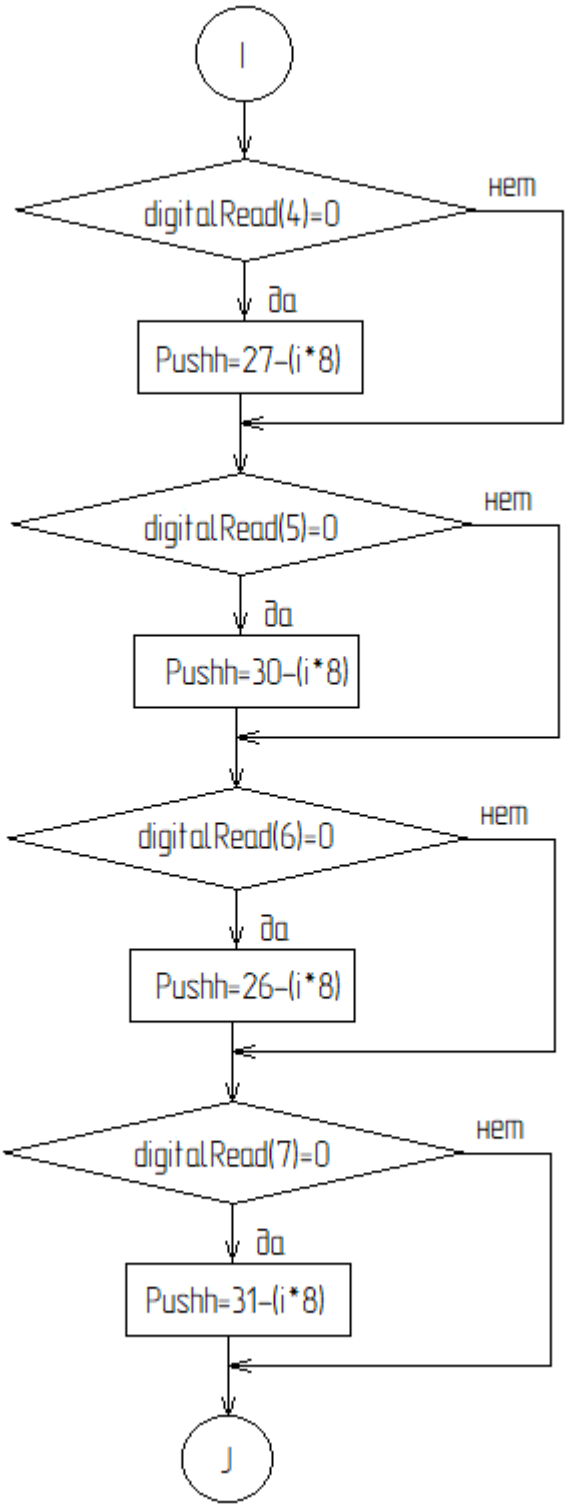

A
void loop
for i=0 to 32
pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],
ColorG[State[i]],ColorB[State[i]]));
pixels.show();
F
while Help=0 & Push=0
C
key=0
нет
да
for i=0 to 32
State[i]=Statee[i]
for i=0 to 32
pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],
ColorG[State[i]],ColorB[State[i]]));
pixels.show();
D

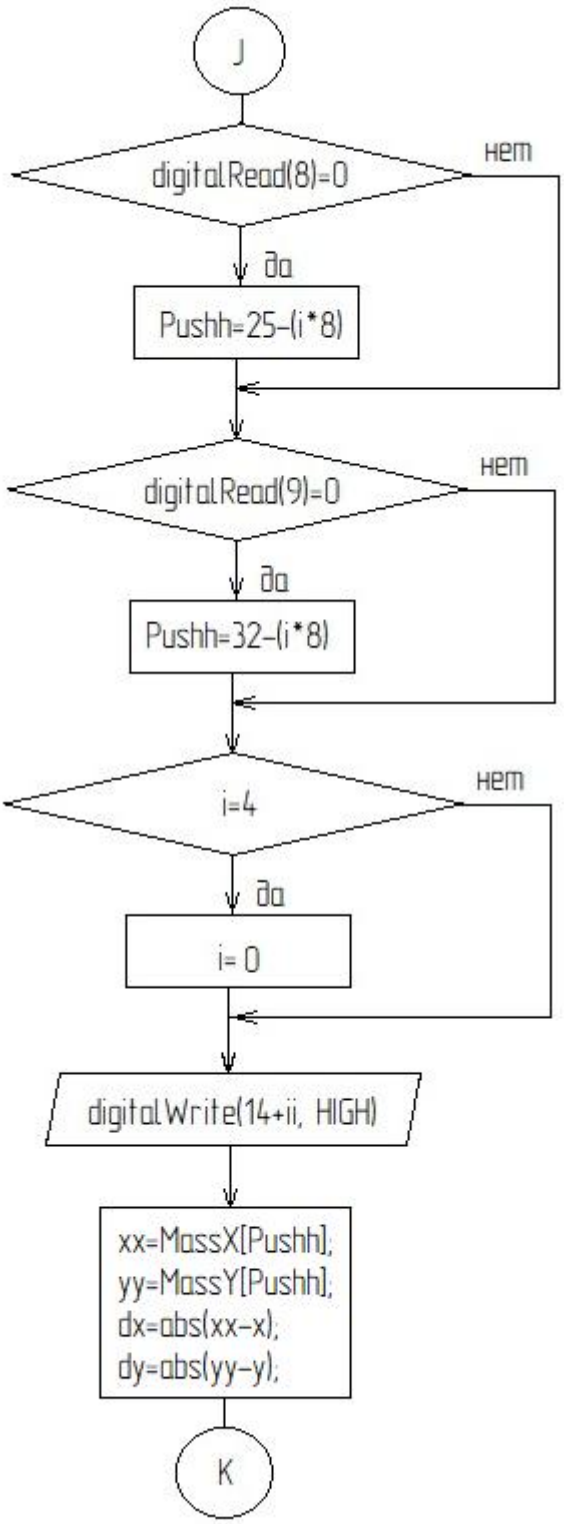

D
E
for i=0 to 4
F
digitalWrite(14+i, LOW)
digitalRead(2)=0
нет
да
Push=28-(i*8)
digitalRead(3)=0
нет
да
Push=29-(i*8)
digitalRead(4)=0
нет
да
Push=27-(i*8)
digitalRead(5)=0
нет
да
Push=30-(i*8)
digitalRead(6)=0
нет
да
Push=26-(i*8)
G

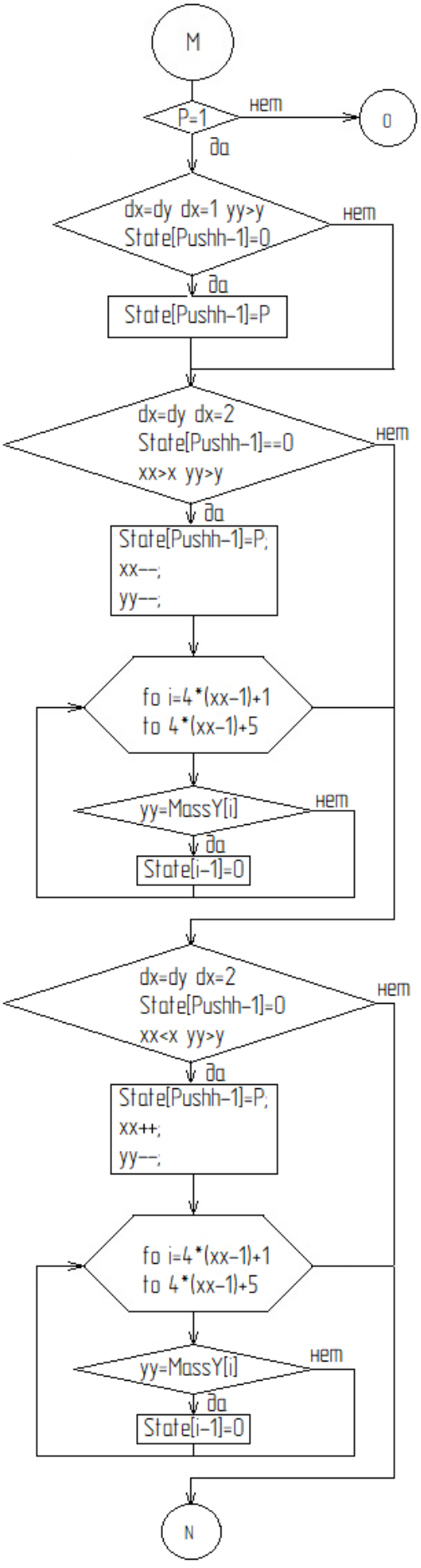

G
digitalRead(7)=0
нет
да
Push=31-(i*8)
digitalRead(8)=0
нет
да
Push=25-(i*8)
digitalRead(9)=0
нет
да
Push=32-(i*8)
i=4
нет
да
i=1
digitalWrite(14+ii, HIGH)
E

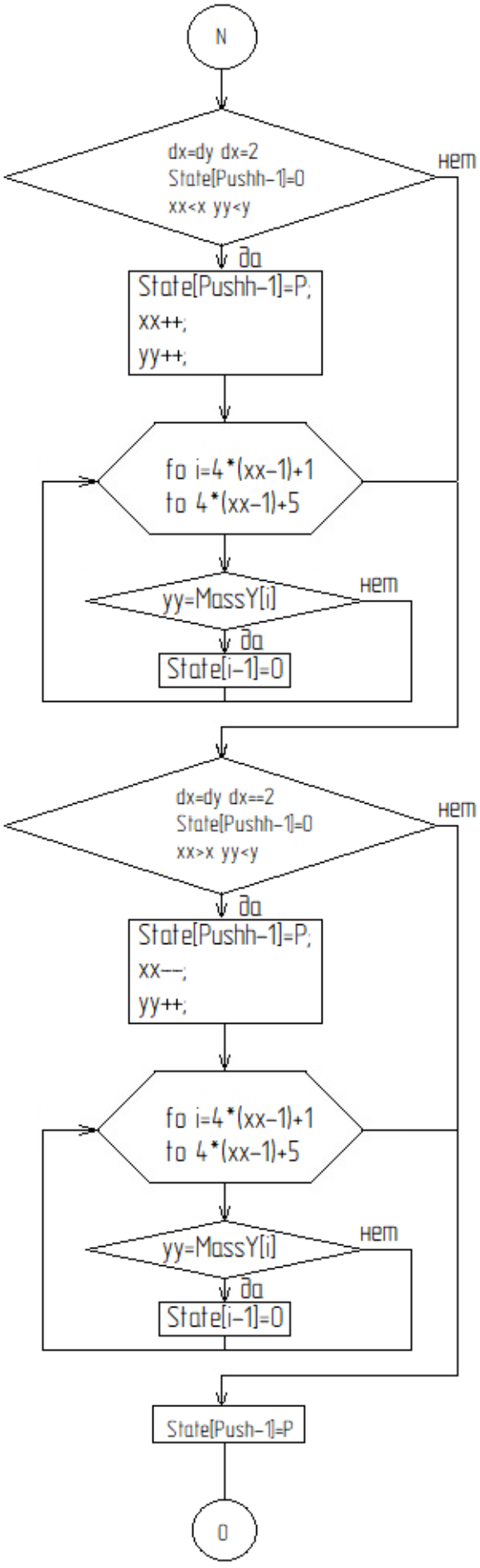

C
for i=0 to 32
Statee[i]=State[i]
P=State[Push-1];
x=MassX[Push];
y=MassY[Push];
State[Push-1]=0;
Help=1;
for i=0 to 32
pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],
ColorG[State[i]],ColorB[State[i]]));
pixels.show();
delay(500);
H

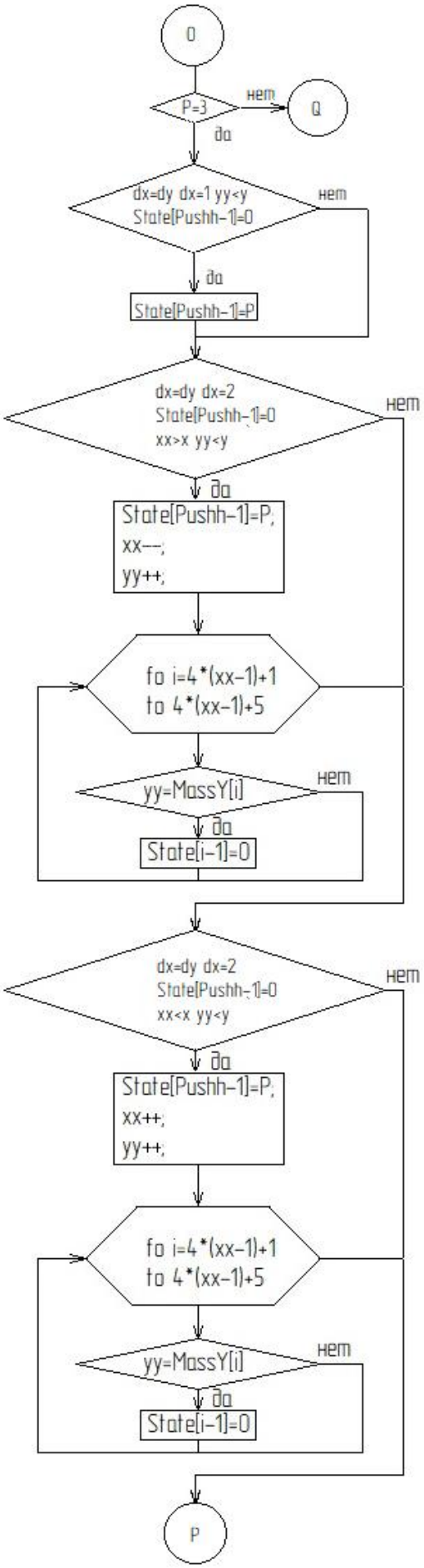

H
L
while Help=1 & Pushh=0
M
K
for i=0 to 4
L
digitalWrite(14+i, LOW)
digitalRead(2)=0
нет
да
Push=28-(i*8)
digitalRead(3)=0
нет
да
Pushh=29-(i*8)
I

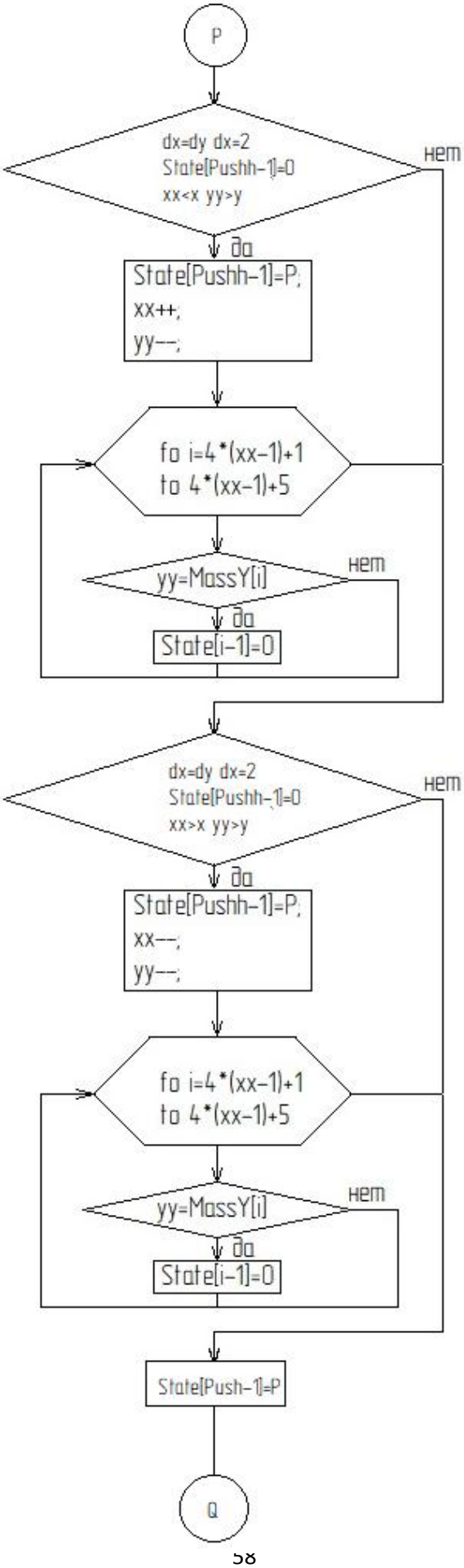

I
digitalRead(4)=0
нет
да
Pushh=27-(i*8)
digitalRead(5)=0
нет
да
Pushh=30-(i*8)
digitalRead(6)=0
нет
да
Pushh=26-(i*8)
digitalRead(7)=0
нет
да
Pushh=31-(i*8)
J

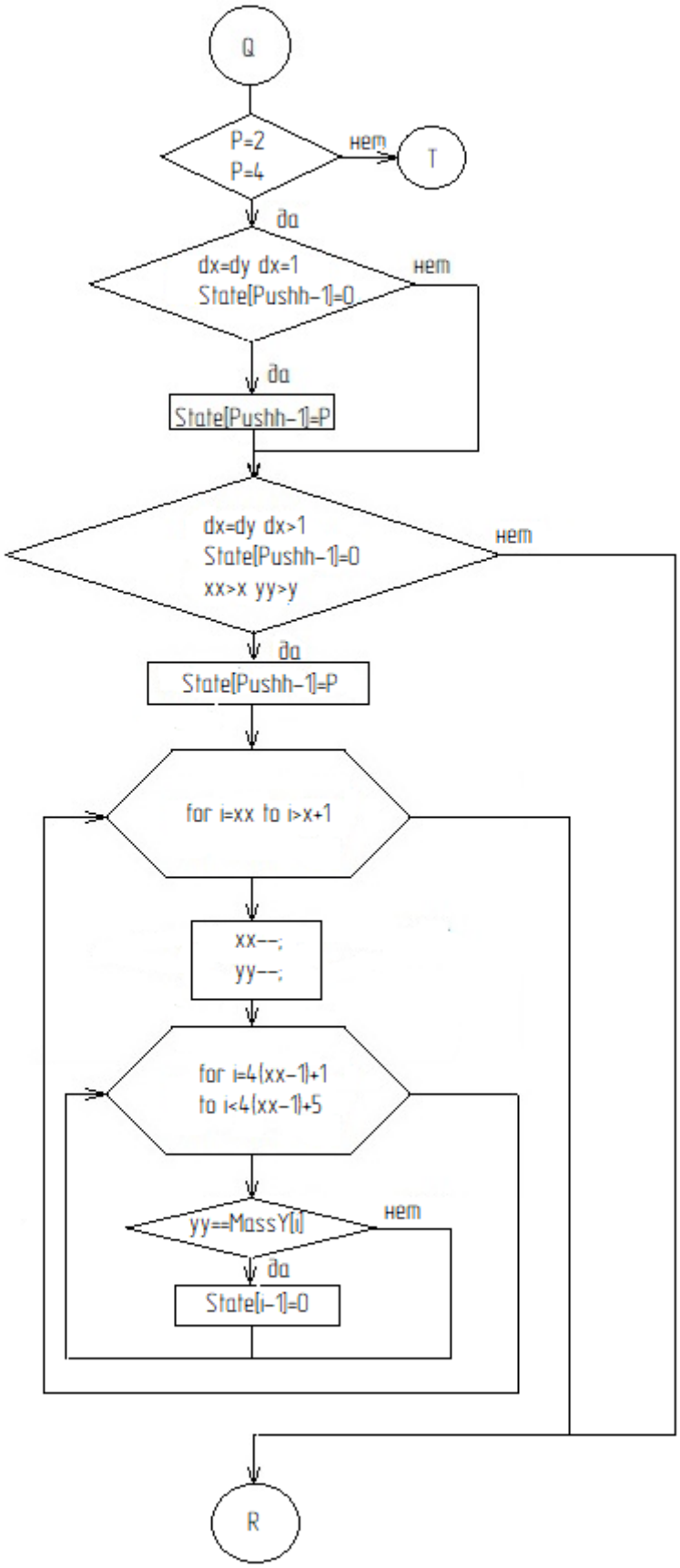

J
digitalRead(8)=0
нет
да
Pushh=25-(i*8)
digitalRead(9)=0
нет
да
Pushh=32-(i*8)
i=4
нет
да
i= 0
digitalWrite(14+ii, HIGH)
xx=MassX[Pushh];
yy=MassY[Pushh];
dx=abs(xx-x);
dy=abs(yy-y);
K

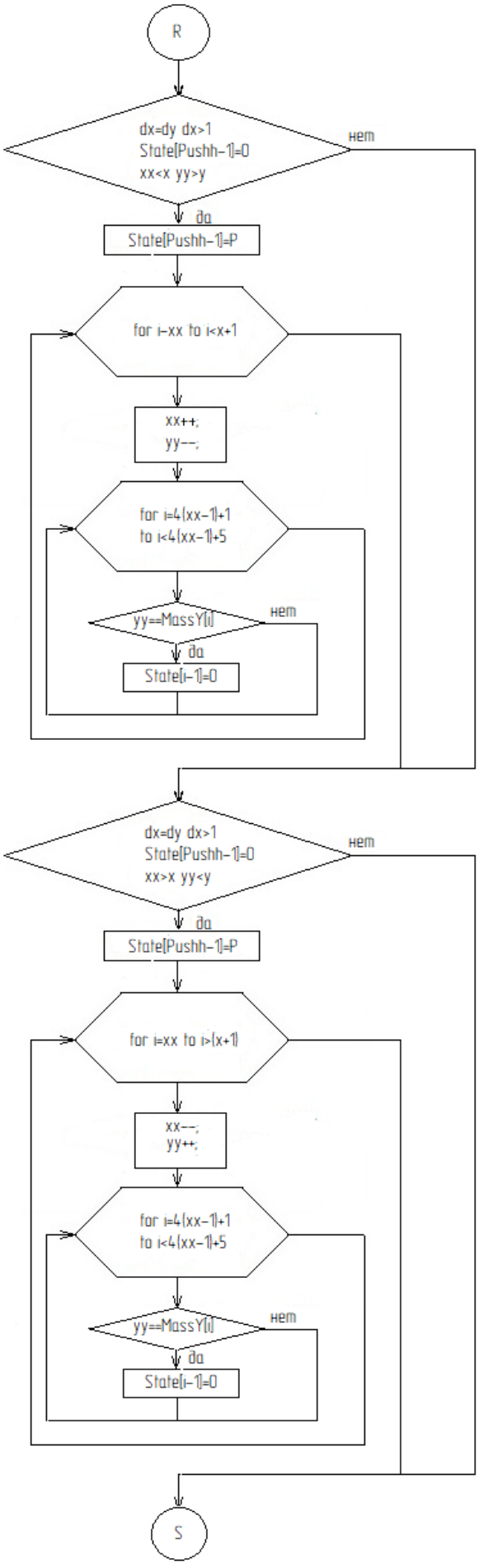

M
P=1
нет
0
да
dx=dy dx=1 yy>y
State[Pushh-1]=0
нет
да
State[Pushh-1]=P
dx=dy dx=2
State[Pushh-1]==0
xx>x yy>y
нет
да
State[Pushh-1]=P;
xx--;
yy--;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
dx=dy dx=2
State[Pushh-1]=0
xx<x yy>y
нет
да
State[Pushh-1]=P;
xx++;
yy--;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
N


N
dx=dy dx=2
State[Pushh-1]=0
xx<x yy<y
нет
да
State[Pushh-1]=P;
xx++;
yy++;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
dx=dy dx==2
State[Pushh-1]=0
xx>x yy<y
нет
да
State[Pushh-1]=P;
xx--;
yy++;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
State[Push-1]=P
0


0
P=3
нет
Q
да
dx=dy dx=1 yy<y
State[Pushh-1]=0
нет
да
State[Pushh-1]=P
dx=dy dx=2
State[Pushh-1]=0
xx>x yy<y
нет
да
State[Pushh-1]=P;
xx--;
yy++;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
dx=dy dx=2
State[Pushh-1]=0
xx<x yy<y
нет
да
State[Pushh-1]=P;
xx++;
yy++;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
P


P
dx=dy dx=2
State[Pushh-1]=0
xx<x yy>y
нет
да
State[Pushh-1]=P;
xx++;
yy--;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
dx=dy dx=2
State[Pushh-1]=0
xx>x yy>y
нет
да
State[Pushh-1]=P;
xx--;
yy--;
fo i=4*(xx-1)+1
to 4*(xx-1)+5
yy=MassY[i]
нет
да
State[i-1]=0
State[Push-1]=P
Q


Q
P=2
P=4
нет
T
да
dx=dy dx=1
State[Pushh-1]=0
нет
да
State[Pushh-1]=P
dx=dy dx>1
State[Pushh-1]=0
xx>x yy>y
нет
да
State[Pushh-1]=P
for i=xx to i>x+1
xx--;
yy--;
for i=4(xx-1)+1
to i<4(xx-1)+5
yy==MassY[i]
нет
да
State[i-1]=0
R


R
dx=dy dx>1
State[Pushh-1]=0
xx<x yy>y
нет
да
State[Pushh-1]=P
for i=xx to i<x+1
xx++;
yy--;
for i=4(xx-1)+1
to i<4(xx-1)+5
yy==MassY[i]
нет
да
State[i-1]=0
dx=dy dx>1
State[Pushh-1]=0
xx>x yy<y
нет
да
State[Pushh-1]=P
for i=xx to i>(x+1)
xx--;
yy++;
for i=4(xx-1)+1
to i<4(xx-1)+5
yy==MassY[i]
нет
да
State[i-1]=0
S

S
dx=dy dx>1
State[Pushh-1]=0
xx<x yy<y
нет
да
State[Pushh-1]=P
for i=xx to i<x+1
xx++;
yy++;
for i=4(xx-1)+1
to i<4(xx-1)+5
yy==MassY[i]
нет
да
State[i-1]=0
Statr[Push-1]=P;
T
State[7]=1
нет
да
State[7]=2
State[15]=1
нет
да
State[15]=2
State[23]=1
нет
да
State[23]=2
State[31]=1
нет
да
State[31]=2
State[3]=3
нет
да
State[3]=
State[11]=3
нет
да
State[11]=4
U

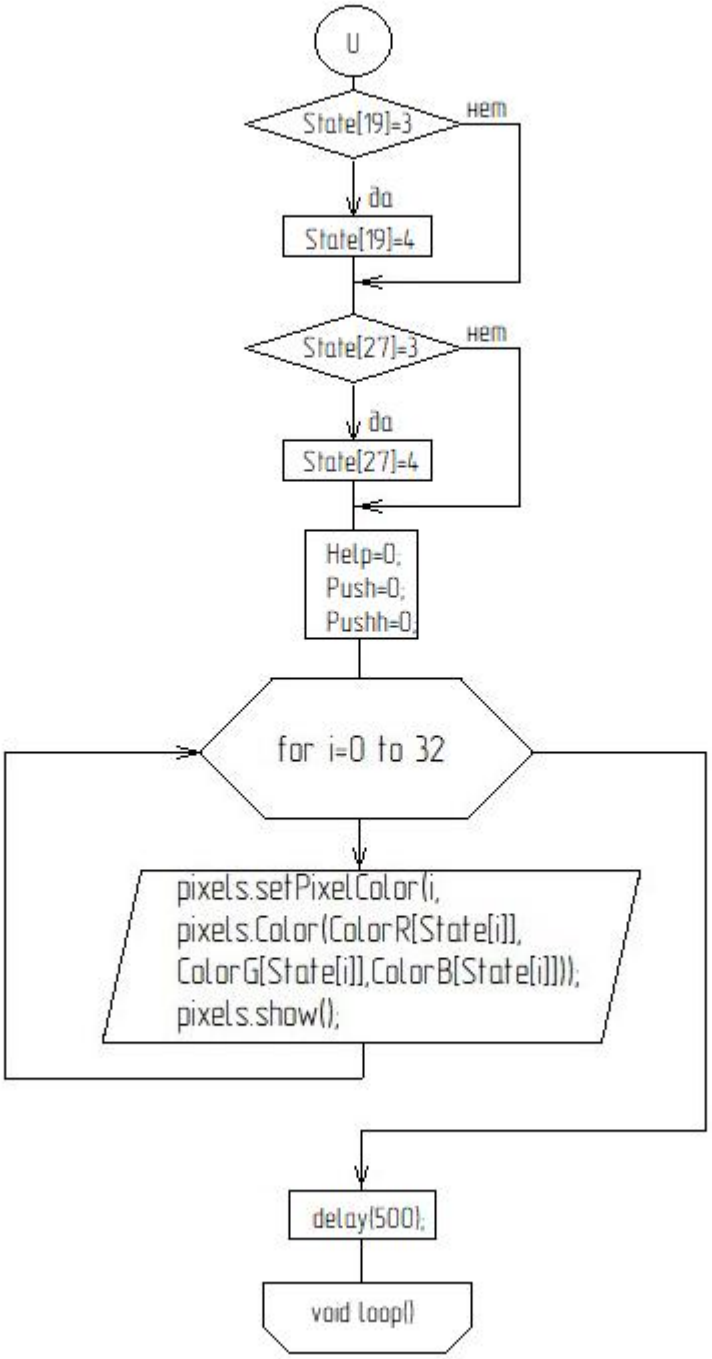


Рисунок 10.1 – Блок-схема программы

Текст программы приведен в конце данного пункта. Разработанная программа должна выполнять следующие задачи:

- Определять, какая кнопка нажата.

- Определять направление хода.

- Проверять допустимость хода.

- Отменять ход автоматически, в случае, если ход недопустим.

- Отменять ход вручную, если он совершен по ошибке игрока.

- Удалять съеденные шашки.

- Зажигать нужные диоды нужным цветом.

В начале программы прикрепляются необходимые библиотеки: Adafruit_NeoPixel.h – для управления светодиодной лентой. Инициализируются переменные, массивы.

В блоке void setup() устанавливаются режимы работы цифровых и аналоговых выводов «Ардуино» [26]. Инициализируется светодиодная лента [27].

В начале цикла void loop() используется цикл для зажигания светодиодов, согласно значениям из первого массива состояний [28]. В начале программы они нужны для стартовой расстановки шашек. Второй массив будет использоваться для отмены хода.

Затем идет блок, выполняющийся до тех пор, пока одна из кнопок не будет нажата. В самом начале он считывает, не нажата ли кнопка отмены. Если нажата, то в первый массив состояний записывается значения из второго, оперативного массива [29]. Пока не был совершен ни один ход, это действие не приведет к результату.

После этого выводится предыдущий ход. Далее идет цикл, последовательно меняющий состояния четырех выходов, с единицы на ноль и наоборот, позволяющий опрашивать ряды кнопок. Если в опрашиваемом ряду будет нажата кнопка, то цифровой вход, подключенный к ней даст значение нуля.

В этом случае данный блок заканчивает свою работу. Состояние до того, как ход сделан, запоминается во второй массив состояний, что бы потом можно было обратиться нему с целью отмены хода.

Запоминаются нажатая кнопка, координаты кнопки по оси x, y, с помощью вспомогательных массивов. Далее делается задержка, для того, что бы не было перескоков в ненужные блоки программы.

Далее идет снова блок опроса кнопок, аналогичный предыдущему, но в этот раз в нем не используются блоки отмены хода, а нажатие запоминается как второе.

Аналогично запоминаются ординаты второго нажатия. Вычисляется дальность хода в клетках по осям.

Далее идут блоки условий, позволяющие программе понять, какой тип хода совершен [30]. Всего есть три больших блока, позволяющие понять, ходила ли желтая шашка, или красная, или дамка. Внутри этих блоков множество условий, определяющие допустимость хода: по своей ли диагонали ходила шашка, не занята ли клетка, бьет ли она, на сколько клеток протяженность хода, направление.

Так же эти блоки позволяют убирать побитые шашки, а так же поставить шашку в то место, куда она сходила.

В случае, если не одно из условий не выполняется, то ход считается недопустимым, и шашка возвращается на то место, с которого начинала ход.

После этого идет блок, определяющий, что пора рядовую шашку перевести в дамку. Он срабатывает, когда шашка занимает клетку вдоль противоположной стенки доски. Эта шашка изменит цвет. Теперь программа ее будет распознавать как дамку. Она сможет ходить и бить на несколько клеток и в любом направлении.

Текст программы «Шашки»

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif
#define PIN          12   //пин управления
#define NUMPIXELS    32   //количество диодов
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN,
NEO_GRB + NEO_KHZ800);

int P=0;
int Help=0;
int Push=0;
int Pushh=0;
int ColorR[]={0,255,0,255,0};
int ColorG[]={0,255,255,0,0};
int ColorB[]={0,0,0,0,255};
int State[]={3,0,1,1,1,0,3,3,3,0,1,1,1,0,3,3,3,0,1,1,1,0,3,3,3,0,1,1,1,0,3,3};
int Statee[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int
MassX[]={0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6,7,7,7,7,8,8,8,8};
int
MassY[]={0,7,5,3,1,2,4,6,8,7,5,3,1,2,4,6,8,7,5,3,1,2,4,6,8,7,5,3,1,2,4,6,8};
int j=0;
int x=0;
int xx=0;
int y=0;
int yy=0;
int dx=0;
```

```
int dy=0;
int key13=0;

void setup() {
  pixels.begin(); // This initializes the NeoPixel library.
  Serial.begin(9600);
  pinMode(2, INPUT);  //Настраиваем пин Д2 как вход,
  digitalWrite(2, HIGH); //Подключаем подтягивающий резистор
  pinMode(3, INPUT);
  digitalWrite(3, HIGH);
  pinMode(4, INPUT);
  digitalWrite(4, HIGH);
  pinMode(5, INPUT);
  digitalWrite(5, HIGH);
  pinMode(6, INPUT);
  digitalWrite(6, HIGH);
  pinMode(7, INPUT);
  digitalWrite(7, HIGH);
  pinMode(8, INPUT);
  digitalWrite(8, HIGH);
  pinMode(9, INPUT);
  digitalWrite(9, HIGH);
  pinMode(14, OUTPUT); //задаем пин A0 как цифровой выход 14
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(17, OUTPUT);
  pinMode(13, INPUT);  //Вход кнопки перезагрузки
  digitalWrite(13, HIGH); //Подключаем подтягивающий резистор
}
```

```
void loop() {

 for(int i=0;i<32;i++){  //для всех диодов
  pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],ColorG[State[i]],ColorB[State[i]])); //присваивает
пикселю цвет
  pixels.show(); // This sends the updated pixel color to the hardware.
 }

 while (Help==0&Push==0) {  //пока кнопка не нажата

  key13=digitalRead(13); //Опрашиваем кнопку отмены хода
  if (key13==0){        //Если она нажата
   for (int iiiii=0; iiiii<33; iiiii++){  //алгоритм для того, что бы записать
предыдущее значение в массив выводимый на диоды
    State[iiiii]=Statee[iiiii];
   }
  }
  for(int i=0;i<32;i++){  //выведем немедленно
   pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],ColorG[State[i]],ColorB[State[i]])); //присваивает
пикселю цвет
   pixels.show(); // This sends the updated pixel color to the hardware.
  }

  for(int ii=0;ii<4;ii++){  //задаем цикл для переключения четырех
аналоговых входов
   digitalWrite(14+ii, LOW);  //подаем ноль на один из аналоговых
входов (для начала на D14==A0)
   if (digitalRead(2)==0)  {  //если цифровой вход считывает ноль
```

```
        Push=28-(ii*8);             //то нажата эта кнопка ПЕРВОЙ,
запомнили ее в переменной
       }
       else if (digitalRead(3)==0) {
        Push=29-(ii*8);
       }
       else if (digitalRead(4)==0)   {
        Push=27-(ii*8);
       }
       else if (digitalRead(5)==0)  {
        Push=30-(ii*8);
       }
       else if (digitalRead(6)==0) {
        Push=26-(ii*8);
       }
       else if (digitalRead(7)==0)   {
        Push=31-(ii*8);
       }
       else if (digitalRead(8)==0)   {
        Push=25-(ii*8);
       }
       else if (digitalRead(9)==0)  {
        Push=32-(ii*8);
       }
       if (ii==4) {    // если цикл из 4 переключений закончился
        ii=0;  // то начинаем заново
       }
       digitalWrite(14+ii, HIGH);  //подаем единицу на тот выход, на которй
сначала подали ноль
      }
```

```
        }

         for (int iiii=0; iiii<33; iiii++) { //Записываем расстановку шашек, пока ход не совершен
           Statee[iiii]=State[iiii];
         }

        P=State[Push-1];  //Запомнили каким цветом горит нажатая кнопка
        x=MassX[Push];  //Запомнили ее координату по оси x
        y=MassY[Push];
        State[Push-1]=0;  //погасили ее
        Help=1;  //переходим на следующее условие, что бы определить, какая кнопка будет второй
        for(int i=0;i<32;i++){  //для всех диодов
          pixels.setPixelColor(i, pixels.Color(ColorR[State[i]],ColorG[State[i]],ColorB[State[i]])); //присваивает пикселю цвет
          pixels.show(); // This sends the updated pixel color to the hardware.
        }
        delay(500);  // Задержка, что бы не перескочить случайно на следующий цикл

        while (Help==1&Pushh==0) {  //Если кнопка еще не нажата
          for(int iii=0;iii<4;iii++){
            digitalWrite(14+iii, LOW);
            if (digitalRead(2)==0)  {
            Pushh=28-(iii*8);              ////нажата эта кнопка ВТОРОЙ, запомнили ее в переменной
            }
            else if (digitalRead(3)==0) {
```

```
    Pushh=29-(iii*8);
   }
   else if (digitalRead(4)==0)   {
    Pushh=27-(iii*8);
   }
   else if (digitalRead(5)==0)  {
    Pushh=30-(iii*8);
   }
   else if (digitalRead(6)==0) {
    Pushh=26-(iii*8);
   }
   else if (digitalRead(7)==0)   {
    Pushh=31-(iii*8);
   }
   else if (digitalRead(8)==0)   {
    Pushh=25-(iii*8);
   }
   else if (digitalRead(9)==0)  {
    Pushh=32-(iii*8);
   }
   if (iii==4) {
    iii=0;
   }
   digitalWrite(14+iii, HIGH);
  }
}

xx=MassX[Pushh]; //Запомнили координату второго нажатия
yy=MassY[Pushh];
```

```
dx=abs(xx-x);  //вычислили разницу между первым и вторым нажатием по оси x
dy=abs(yy-y);
if (P==1) {  //Если ходила желтая шашка (не дамка)
 if (dx==dy&dx==1&yy>y&State[Pushh-1]==0){  //если она ходила по своей диагонали, и если на одну клетку, и если вперед, и если та клетка свободна
  State[Pushh-1]=P;  //то та клетка становится тем цветом, который запомнили раньше
 }
 else if (dx==dy&dx==2&State[Pushh-1]==0&xx>x&yy>y) {  //если она ходит по своей диагонали, и если на две клетки, и если вправо
  State[Pushh-1]=P;  //пустую клетку заполнили цетом
  xx--;  //вычислили кординату клетки, через которую шашка переступила
  yy--;
  for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) { //xx определяет на сколько номер клетки в общей матрице большой, перебираем 4 значения, например, 21, 22, 23, 24
   if (yy==MassY[i]) {  //если 21 значение в матрице У совпадает с координатой yy нужной нам клетки
    State[i-1]=0;  //то значение из общей матрицы под тем же индексом будет съеденой шашкой (-1, т.к. она начинается с нуля)
   }
  }
 }
 else if (dx==dy&dx==2&State[Pushh-1]==0&xx<x&yy>y) {  //тоже самое, только ходит влево
  State[Pushh-1]=P;
  xx++;
```

```
            yy--;
            for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
              if (yy==MassY[i]) {
                State[i-1]=0;
              }
            }
          }
          else if (dx==dy&dx==2&State[Pushh-1]==0&xx<x&yy<y) {  //тоже самое, только ходит влево
            State[Pushh-1]=P;
            xx++;
            yy++;
            for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
              if (yy==MassY[i]) {
                State[i-1]=0;
              }
            }
          }
          else if (dx==dy&dx==2&State[Pushh-1]==0&xx>x&yy<y) {  //тоже самое, только ходит влево
            State[Pushh-1]=P;
            xx--;
            yy++;
            for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
              if (yy==MassY[i]) {
                State[i-1]=0;
              }
            }
          }
          else {
```

```
          State[Push-1]=P;  // если ни одно условие не выполняется, то шашка возвращается на первоначальную клетку
         }
        }
        else if (P==3) {
         if (dx==dy&dx==1&yy<y&State[Pushh-1]==0){
          State[Pushh-1]=P;
         }
         else if (dx==dy&dx==2&State[Pushh-1]==0&xx>x&yy<y) {
          State[Pushh-1]=P;
          xx--;
          yy++;
           for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
            if (yy==MassY[i]) {
              State[i-1]=0;
            }
           }
         }
         else if (dx==dy&dx==2&State[Pushh-1]==0&xx<x&yy<y) {
          State[Pushh-1]=P;
          xx++;
          yy++;
          for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
           if (yy==MassY[i]) {
             State[i-1]=0;
           }
          }
         }
         else if (dx==dy&dx==2&State[Pushh-1]==0&xx<x&yy>y) {
          State[Pushh-1]=P;
```

```
    xx++;
    yy--;
    for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
      if (yy==MassY[i]) {
        State[i-1]=0;
      }
    }
  }
  else if (dx==dy&dx==2&State[Pushh-1]==0&xx>x&yy>y) {
    State[Pushh-1]=P;
    xx--;
    yy--;
    for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
      if (yy==MassY[i]) {
        State[i-1]=0;
      }
    }
  }
  else {
    State[Push-1]=P;
  }
}


else if (P==2||P==4){
  if (dx==dy&dx==1&State[Pushh-1]==0){
    State[Pushh-1]=P;
  }
  else if (dx==dy&dx>1&State[Pushh-1]==0&xx>x&yy>y) {
    State[Pushh-1]=P;
```

```
  for (int i=xx; i>(x+1); i--) {
    xx--;
    yy--;
    for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
      if (yy==MassY[i]) {
        State[i-1]=0;
      }
    }
  }
}
else if (dx==dy&dx>1&State[Pushh-1]==0&xx<x&yy>y) {
  State[Pushh-1]=P;
  for (int i=xx; i<(x+1); i++) {
    xx++;
    yy--;
    for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
      if (yy==MassY[i]) {
        State[i-1]=0;
      }
    }
  }
}
else if (dx==dy&dx>1&State[Pushh-1]==0&xx>x&yy<y) {
  State[Pushh-1]=P;
  for (int i=xx; i>(x+1); i--) {
    xx--;
    yy++;
    for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
      if (yy==MassY[i]) {
        State[i-1]=0;
```

```
        }
       }
     }
   }
   else if (dx==dy&dx>1&State[Pushh-1]==0&xx<x&yy<y) {
     State[Pushh-1]=P;
     for (int i=xx; i<(x+1); i++) {
       xx++;
       yy++;
       for(int i=4*(xx-1)+1; i<4*(xx-1)+5; i++) {
         if (yy==MassY[i]) {
           State[i-1]=0;
         }
       }
     }
   }
   else {
     State[Push-1]=P;
   }
}

if (State[7]==1) { //перевод в дамки
   State[7]=2;
}
if (State[15]==1) {
   State[15]=2;
}
if (State[23]==1) {
   State[23]=2;
}
```

```
        if (State[31]==1) {
          State[31]=2;
        }
        if (State[3]==3) {
          State[3]=4;
        }
        if (State[11]==3) {
          State[11]=4;
        }
        if (State[19]==3) {
          State[19]=4;
        }
        if (State[27]==3) {
          State[27]=4;
        }

        Help=0;
        Push=0;
        Pushh=0;

        for(int i=0;i<32;i++){  //для всех диодов
          pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],ColorG[State[i]],ColorB[State[i]])); //присваивает
пикселю цвет
          pixels.show(); // This sends the updated pixel color to the hardware.
        }
        delay(500);

        for(int i=0;i<32;i++){  //для всех диодов
```

```
            pixels.setPixelColor(i,
pixels.Color(ColorR[State[i]],ColorG[State[i]],ColorB[State[i]])); //присваивает
пикселю цвет
            pixels.show(); // This sends the updated pixel color to the hardware.
          }
        }
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы, были достигнуты две цели.

Первая: разработана и собрана установка для практического исследования восприятия глазом динамической индикации и ШИМ – регулирования яркости.

Разработаны структурная схема, электрическая схема, конструкция, программа прибора и физически реализован сам прибор. Проведены исследования ШИМ – регулирования и динамической индикации с помощью собранного прибора.

Вторая цель: создана электронная игра в шашки. Разработаны концепция прибора, структурная и электричесая схемы, конструкция, программа. Собран работающий прибор.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1) Обзор шашки/шахматы электронные 4Tune-G860 [Электронный ресурс] URL: http://gamma-gift.ru/statji/2014/01/31/statji1_76.html (Дата обращения: 11.12.2016).

2) Электронные пластиковые шашки [Электронный ресурс] URL: https://ru.aliexpress.com/item/Beginner-travel-taking-chess-Magnetic-Electronic-Checkers-total-8-games-LCD-display/32616842795.html?spm=2114.03020208.3.102.KKArUs&ws_ab_test=searchweb0_0,searchweb201602_2_10065_10068_10000009_10084_10083_10080_10082_10081_10060_10061_10062_10056_10055_10054_10059_10099_10078_10079_10093_427_10103_10073_10102_10096_10052_10050_10051,searchweb201603_3,afswitch_2&btsid=32cad572-d5c8-471c-9ffa-7d5832e82825 (Дата обращения: 11.12.2016).

3) Джойстик для IBM-PC [Электронный ресурс] URL: http://exos.org.ru/download/joystick.pdf (Дата обращения: 23.01.2018).

4) Arduino [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Arduino (Дата обращения: 10.01.2018).

5) Arduino [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Arduino (Дата обращения: 10.01.2018).

6) Arduino – описание выводов на примере Arduino UNO [Электронный ресурс] URL: http://www.joyta.ru/10674-arduino-opisanie-vyvodov-na-primere-arduino-uno/ (Дата обращения: 26.02.2018).

7) A Power Balance Aware Wireless Charger Deployment Method for Complete Coverage in Wireless Rechargeable Sensor Networks [Электронный ресурс] – URL: http://www.mdpi.com/1996-1073/9/9/695/htm (дата обращения: 21.11.2016)

8) Photovoltaic Power System with an Interleaving Boost Converter for Battery Charger Applications [Электронный ресурс] – URL:

https://www.hindawi.com/journals/ijp/2012/936843/ (дата обращения: 21.11.2016)

9) A Experiment Method of Wireless Power Transfer for Charging Devices [Электронный ресурс] – URL: http://www.ajer.org/papers/v5(11)/P05110110123.pdf (дата обращения: 21.11.2016)

10) Fractal properties of LED avalanche breakdown [Электронный ресурс] – URL: http://www.sciencedirect.com/science/article/pii/S2405722316301566 (дата обращения: 16.04.2017)

11) Theoretical and experimental analysis of AlGaInP micro-LED array with square-circle anode [Электронный ресурс] – URL: http://aip.scitation.org/doi/10.1063/1.4904217 (дата обращения: 16.04.2017)

12) Color Temperature Tunable White-Light LED Cluster with Extrahigh Color Rendering Index [Электронный ресурс] – URL: https://www.hindawi.com/journals/tswj/2014/897960/ (дата обращения: 16.04.2017)

13) Generating Selected Color using RGB, Auxiliary Lights, and Simplex Search [Электронный ресурс] – URL: http://www.matec-conferences.org/articles/matecconf/pdf/2015/13/matecconf_isot2015_05007.pdf (дата обращения: 16.04.2017)

14) Design of Programmable LED Controller with a Variable Current Source for 3D Image Display

[Электронный ресурс] – URL:
http://www.mdpi.com/2078-2489/5/4/652/htm (дата обращения: 16.04.2017)

15) LED Current Balance Using a Variable Voltage Regulator with Low Dropout vDS Control
[Электронный ресурс] – URL:
http://www.mdpi.com/2076-3417/7/2/206/htm (дата обращения: 16.04.2017)

16) Implementation and Test of a LED-Based Lamp for a Lighthouse
[Электронный ресурс] – URL:
https://www.hindawi.com/journals/ijp/2014/574270/ (дата обращения: 16.04.2017)

17) Download the Arduino Software [Электронный ресурс] URL: https://www.arduino.cc/en/Main/Software (Дата обращения: 24.12.2016).

18) Arduino’s LiquidCrystal Library [Электронный ресурс] URL: http://playground.arduino.cc/Main/LiquidCrystal (Дата обращения: 24.12.2016).

19) AVR&Arduino [Электронный ресурс] URL: http://atmel.ucoz.ru/publ/variabledeclaration/1-1-0-141 (Дата обращения: 29.12.2016).

20) ASCII [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ASCII (Дата обращения: 29.12.2016).

21) Светодиодне ленты WS2811 и другие[Электронный ресурс] URL: http://forums.balancer.ru/tech/forum/2016/02/t92411--svetodiodnye-lenty-ws2811-i-drugie.html#p4072950 (Дата обращения: 10.01.2018).

22) Подключение матрицы кнопок к Arduino [Электронный ресурс] URL: http://mypractic.ru/urok-18-podklyuchenie-matricy-knopok-k-arduino-funkciya-tone.html (Дата обращения: 10.01.2018).

23) Зарядное устройство [Электронный ресурс] URL: https://ru.aliexpress.com/item/1PCS-5V-1A-Micro-USB-18650-Lithium-Battery-Charging-Board-Charger-Module-Protection-Dual-

Functions/32467578996.html?spm=a2g0s.9042311.0.0.R0CQEo (Дата обращения: 09.11.2017).

24) Регулятор напряжения [Электронный ресурс] URL: https://ru.aliexpress.com/item/1pcs-MT3608-2A-Max-DC-DC-Step-Up-Power-Module-Booster-Power-Module-For-Arduino/32581504487.html?spm=a2g0s.9042311.0.0.R0CQEo (Дата обращения: 09.11.2017).

25) Плата [Электронный ресурс] URL: https://www.aliexpress.com/item/High-Quality-1pc-DIY-Prototype-PCB-Universal-Matrix-Circuit-Board-Breadboard-8-5x20cm-free-shipping/32610503095.html?spm=a2g0s.9042311.0.0.HZx2Y3 (Дата обращения: 09.11.2017).

26) Программирование Ардуино [Электронный ресурс] URL: https://all-arduino.ru/programmirovanie-arduino/ (Дата обращения: 24.05.2018).

27) Подключение и управление светодиодной лентой Ардуино [Электронный ресурс] URL: https://lampaexpert.ru/svetodiodnaya-lenta/podklyuchenie-i-upravlenie-arduino (Дата обращения: 24.05.2018).

28) Циклы в Arduino [Электронный ресурс] URL: http://studrobots.ru/циклы-в-arduino/ (Дата обращения: 24.05.2018)

29) Массивы [Электронный ресурс] URL: https://arduinoplus.ru/coding-arduino/data/data-types/arrays/ (Дата обращения: 24.05.2018)

30) Условия if else в Arduino [Электронный ресурс] URL: https://arduinomaster.ru/program/arduino-if-else-uslovie/ (Дата обращения: 24.05.2018)

Поз. об.	Наименование	Кол.	Примечание
	Резисторы		
R1	МЛТ-0,5-141 Ом ±10%	1	
R2	МЛТ-0,5-570 Ом ±10%	1	
R3, R5	МЛТ-0,5-100 Ом ±10%	2	
R4	МЛТ-0,5-470 Ом ±10%	1	
R6	МЛТ-0,5-2 кОм ±10%	1	
R7	МЛТ-0,5-1 кОм ±10%	1	
	Светодиоды		
HL1	АЛ107	1	Светодиод
HL2	АЛ107	1	Светодиод
HL3	АЛ107	1	Светодиод
	Схемы		
DD1	Ардуино Уно	1	
DD2	Джойстик KY-023	1	
DD3	Дисплей LCD	1	

Согласовано

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Колуч.	Лист	№док.	Подп.	Дата
Разраб.		Николаев А.В.			
Утв.		Шевцов А.А.			
Пров.		Прядилов А.В.			

Прибор для изучения динамической индикации

Стад	Лист	Листов

Перечень элементов

Группа ЭЛм-1601

Поз. об.	Наименование	Кол.	Примечание
	Ключи		
SA1-SA32	Кнопка ПКН-6х6х12	32	
	Микросхемы		
DD1	Зарядное устройство TP4056	1	
DD2	Трансформатор напряжения MT3608	1	
DD3	Ардуино уно	1	
DD4-DD35	Адресные светодиоды WS2812	32	
	Элементы питания		
B1	Батарея 3,7В	1	

Согласовано

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Колуч.	Лист	№док.	Подп.	Дата			
Разраб.		Николаев А.В.				Электронные шашки	Стад	Лист	Листов
Утв.		Шевцов А.А.							
Пров.		Прядилов А.В,							
						Перечень элементов	Группа ЭЛм-1601		
