Практическое занятие №
Тема: «Программирование LED-эффектов»

Цель работы: приобрести практические навыки по программированию диамических световых эффектов на платформе Arduino.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно заданию в примере.

– Выполнить задание для самостоятельной работы.

Содержание отчета:

– Название практического занятия, его цель.

– Фото или скриншоты собранной схемы.

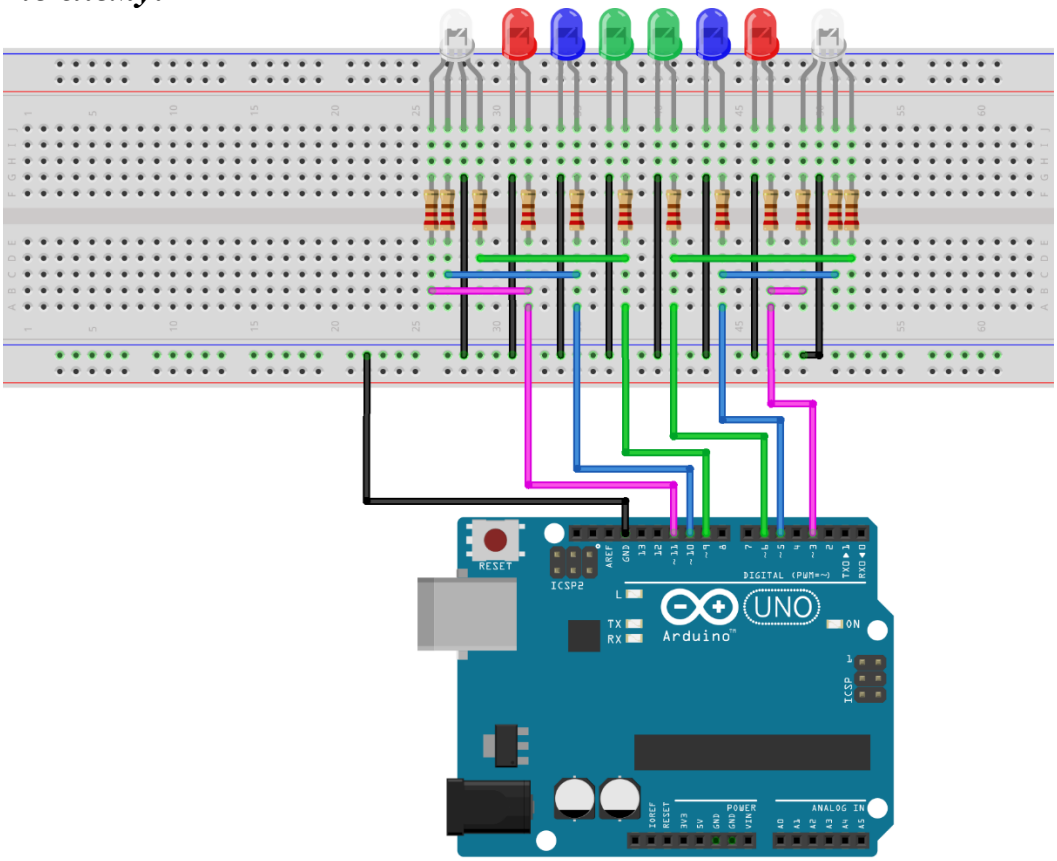
– Написанный программный код вставить текстом, Courier New, 12 кегль, одинарный отступ без абзацев.

– Вывод о проделанной работе.

– Файл Fritzing с принципиальной и монтажной схемой.

ЗАДАНИЯ

Собрать схему:



Написать программный код:

Описать принцип работы программы (loop), а также функции эффектов.

```
// Определяем пины для светодиодов
const byte ledPins[] = {3, 5, 6, 9, 10, 11};
const int numLeds = 6;

// Переменные для эффектов
unsigned long previousMillis = 0;
int effectDelay = 50;
int currentEffect = 0;
int brightness[numLeds] = {0};
int fadeDirection = 1;

void setup() {
  // Настраиваем пины как выходы
  for (int i = 0; i < numLeds; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
  Serial.begin(9600);
  Serial.println("Проигрываниие 10 LED-эффектов начаты");
}

void loop() {
  // Переключаем эффекты каждые 8 секунд
  if (millis() - previousMillis > 8000) {
    previousMillis = millis();
    currentEffect = (currentEffect + 1) % 11; // 0-10
эффектов

    // Выключаем все светодиоды при смене эффекта
    for (int i = 0; i < numLeds; i++) {
      analogWrite(ledPins[i], 0);
    }

    Serial.print("Эффект ");
    Serial.println(currentEffect + 1);
  }

  // Выполняем текущий эффект
  switch (currentEffect) {
    case 0:
      effect1_SequentialBlink();
```

```
      break;
    case 1:
      effect2_Breathing();
      break;
    case 2:
      effect3_RunningDot();
      break;
    case 3:
      effect4_PingPong();
      break;
    case 4:
      effect5_Accordion();
      break;
    case 5:
      effect6_SimultaneousFade();
      break;
    case 6:
      effect7_RainbowWave();
      break;
    case 7:
      effect8_BinaryCount();
      break;
    case 8:
      effect9_Chaos();
      break;
    case 9:
      effect10_Fire();
      break;
  }
}

// Эффект 1: Последовательное мигание
void effect1_SequentialBlink() {
  static int currentLed = 0;
  static unsigned long lastChange = 0;

  if (millis() - lastChange > 200) {
    // Выключаем все
    for (int i = 0; i < numLeds; i++) {
      analogWrite(ledPins[i], 0);
    }

    // Включаем текущий
    analogWrite(ledPins[currentLed], 255);
```

```
    currentLed = (currentLed + 1) % numLeds;
    lastChange = millis();
  }
}

// Эффект 2: Дыхание всех светодиодов
void effect2_Breathing() {
  static int breathValue = 0;
  static int breathDirection = 5;

  breathValue += breathDirection;

  if (breathValue >= 255 || breathValue <= 0) {
    breathDirection = -breathDirection;
  }

  for (int i = 0; i < numLeds; i++) {
    analogWrite(ledPins[i], breathValue);
  }

  delay(20);
}

// Эффект 3: Бегущая точка
void effect3_RunningDot() {
  static int pos = 0;
  static unsigned long lastMove = 0;

  if (millis() - lastMove > 100) {
    // Гасим все
    for (int i = 0; i < numLeds; i++) {
      analogWrite(ledPins[i], 0);
    }

    // Зажигаем текущую и соседние
    analogWrite(ledPins[pos], 255);
    if (pos > 0) analogWrite(ledPins[pos-1], 100);
    if (pos < numLeds-1) analogWrite(ledPins[pos+1], 100);

    pos = (pos + 1) % numLeds;
    lastMove = millis();
  }
}
```

```
// Эффект 4: Пинг-понг
void effect4_PingPong() {
  static int pos = 0;
  static int dir = 1;
  static unsigned long lastMove = 0;

  if (millis() - lastMove > 150) {
    // Гасим все
    for (int i = 0; i < numLeds; i++) {
      analogWrite(ledPins[i], 0);
    }

    // Зажигаем текущий
    analogWrite(ledPins[pos], 255);

    pos += dir;
    if (pos >= numLeds-1 || pos <= 0) {
      dir = -dir;
    }

    lastMove = millis();
  }
}

// Эффект 5: Гармошка
void effect5_Accordion() {
  static int center = 0;
  static int dir = 1;
  static unsigned long lastMove = 0;

  if (millis() - lastMove > 200) {
    // Гасим все
    for (int i = 0; i < numLeds; i++) {
      analogWrite(ledPins[i], 0);
    }

    // Зажигаем от центра
    for (int offset = 0; offset < numLeds/2; offset++) {
      int left = center - offset;
      int right = center + offset;

      if (left >= 0) analogWrite(ledPins[left], 255 -
offset*50);
```

```
      if (right < numLeds) analogWrite(ledPins[right], 255 -
offset*50);
    }

    center += dir;
    if (center >= numLeds-1 || center <= 0) {
      dir = -dir;
    }

    lastMove = millis();
  }
}

// Эффект 6: Одновременное плавное включение/выключение
void effect6_SimultaneousFade() {
  static int fadeValue = 0;
  static int fadeDir = 3;

  fadeValue += fadeDir;

  if (fadeValue >= 255 || fadeValue <= 0) {
    fadeDir = -fadeDir;
  }

  for (int i = 0; i < numLeds; i++) {
    analogWrite(ledPins[i], fadeValue);
  }

  delay(15);
}

// Эффект 7: Радужная волна
void effect7_RainbowWave() {
  static int hue = 0;

  for (int i = 0; i < numLeds; i++) {
    // Преобразуем HSV в RGB
    int phase = (hue + i * 40) % 360;
    int brightness = 0;

    if (phase < 120) {
      brightness = map(phase, 0, 120, 0, 255);
    } else if (phase < 240) {
      brightness = map(phase, 120, 240, 255, 0);
```

```
    }

    analogWrite(ledPins[i], brightness);
  }

  hue = (hue + 5) % 360;
  delay(30);
}

// Эффект 8: Двоичный счет
void effect8_BinaryCount() {
  static int counter = 0;
  static unsigned long lastIncrement = 0;

  if (millis() - lastIncrement > 500) {
    // Устанавливаем яркость в соответствии с битами
    for (int i = 0; i < numLeds; i++) {
      if (counter & (1 << i)) {
        analogWrite(ledPins[i], 255);
      } else {
        analogWrite(ledPins[i], 50); // Низкая яркость для 0
      }
    }

    counter = (counter + 1) % (1 << numLeds);
    lastIncrement = millis();
  }
}

// Эффект 9: Хаотичное мерцание
void effect9_Chaos() {
  for (int i = 0; i < numLeds; i++) {
    // Случайная яркость со сглаживанием
    brightness[i] = (brightness[i] + random(-30, 30)) / 2;
    brightness[i] = constrain(brightness[i], 0, 255);

    analogWrite(ledPins[i], brightness[i]);
  }

  delay(50);
}

// Эффект 10: Эффект огня
void effect10_Fire() {
```

```
  for (int i = 0; i < numLeds; i++) {
    // Имитация пламени
    int flicker = random(0, 150);
    int heat = map(i, 0, numLeds, 100, 255); // Ближе к
центру - горячее

    int fireValue = heat - flicker;
    fireValue = constrain(fireValue, 0, 255);

    analogWrite(ledPins[i], fireValue);
  }

  delay(random(30, 100));
}
```