

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет: Информатика и вычислительная техника

Кафедра: Кибербезопасность информационных систем

**Отчёт по лабораторным работам**

**по дисциплине “Аппаратные средства вычислительной техники”**

Выполнил обучающийся гр.ВКБ21

Котелевец Кирилл Александрович

(Фамилия, Имя, Отчество)

Проверила:

Доцент Енгибарян И.А.

(должность, Фамилия, Имя, Отчество)

Ростов-на-Дону

2024

Содержание

[Лабораторная работа №1 3](#_Toc164504324)

[Лабораторная работа №2 5](#_Toc164504325)

[Лабораторная работа №3 9](#_Toc164504326)

[Лабораторная работа №4 18](#_Toc164504327)

[Лабораторная работа №5 23](#_Toc164504328)

[Лабораторная работа №6 28](#_Toc164504329)

[Лабораторная работа №7 38](#_Toc164504330)

# Лабораторная работа №1

**Тема**: Знакомство с Tinkercad

**Цель:** ознакомиться с сервисом Tinkercad и зарегистрироваться на нём.

**Ход работы**

Tinkercad – это онлайн сервис, который сейчас принадлежит мастодонту мира CAD-систем — компании Autodesk. Tinkercad уже давно известен многим как простая и бесплатная среда для обучения 3D-моделированию. С ее помощью можно достаточно легко создавать свои модели и отправлять их на 3D-печать. Единственным ограничением для русскоязычного сегмента интернета долгое время являлось отсутствие русскоязычного интерфейса, сейчас эта ситуация исправляется. Совсем недавно Tinkercad получил возможность создания электронных схем и подключения их к симулятору виртуальной платы Arduino. Эти крайне важные и мощные инструменты способны существенно облегчить начинающим разработчикам Arduino процессы обучения, проектирования и программирования новых схем.

Прежде чем работать с данным с данным сервисом, на нем необходимо зарегистрироваться. На рисунке 1.1 представлен зарегистрированный профиль на Tinkercad.

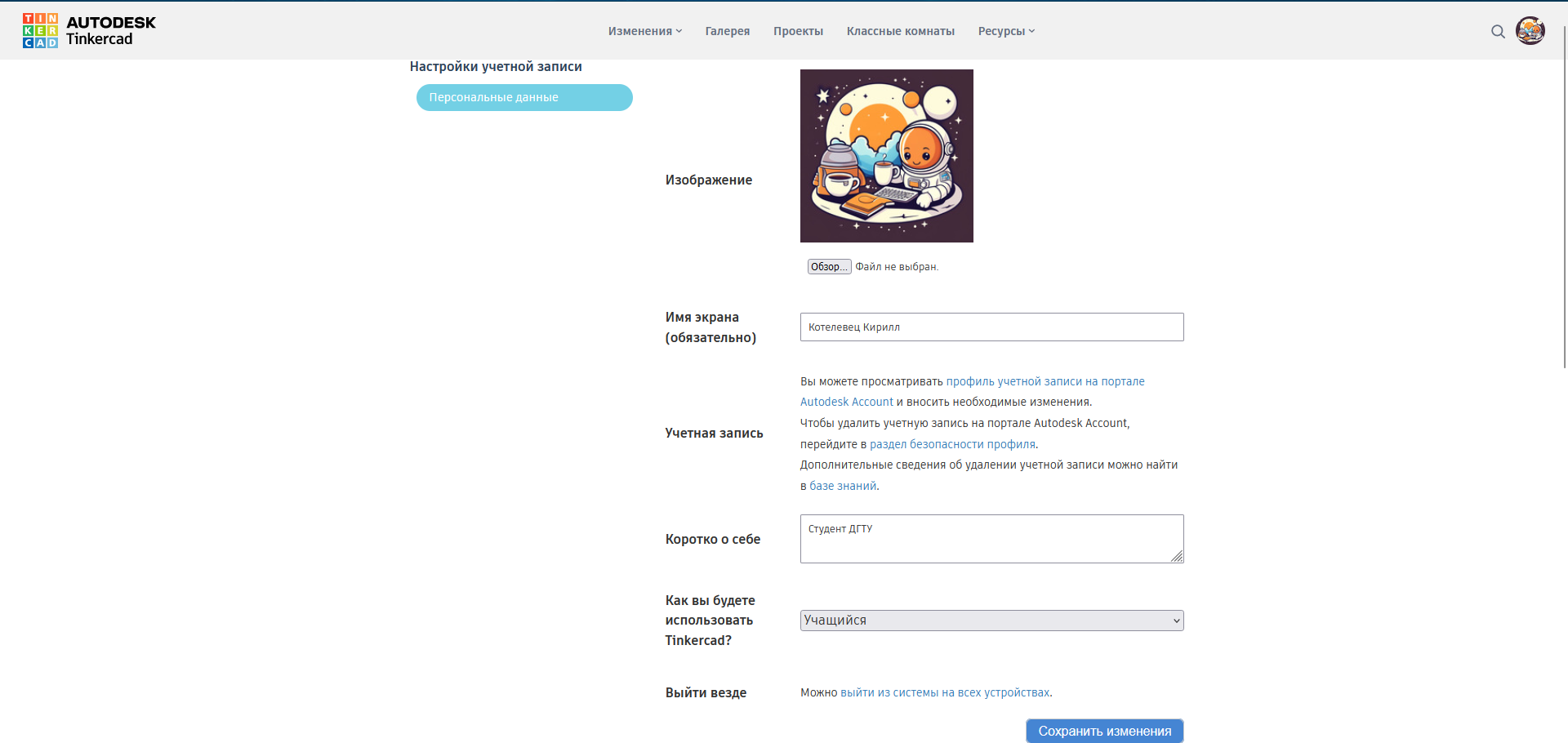


Рисунок 1.1 – профиль на сервисе Tinkercad

На рисунке 1.2 представлено окно, на котором можно увидеть мои проекты, а также выбор создания нового проекта. На рисунке 1.3 представлен сам интерфейс работы со схемами Arduino.

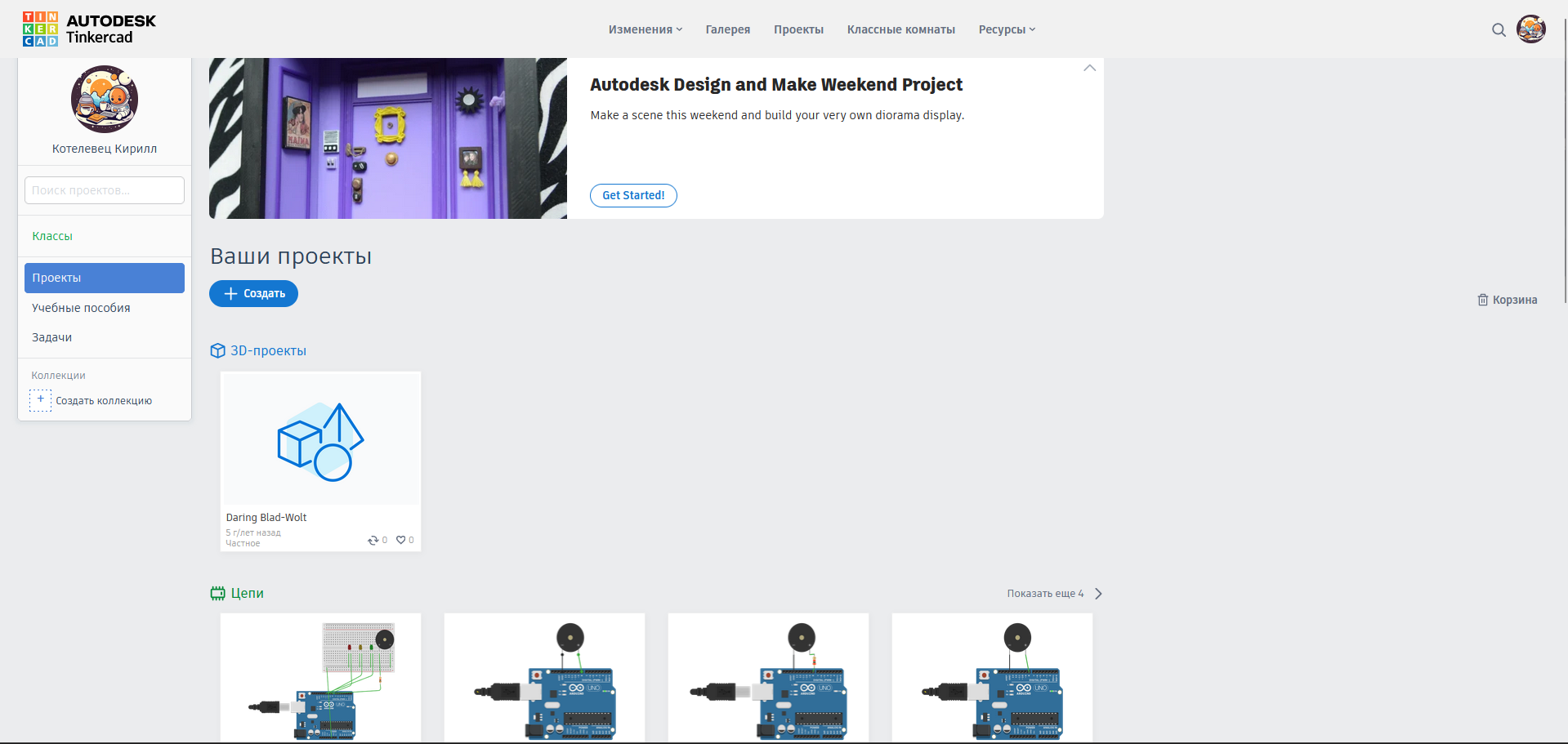


Рисунок 1.2 – проекты в Tinkercad

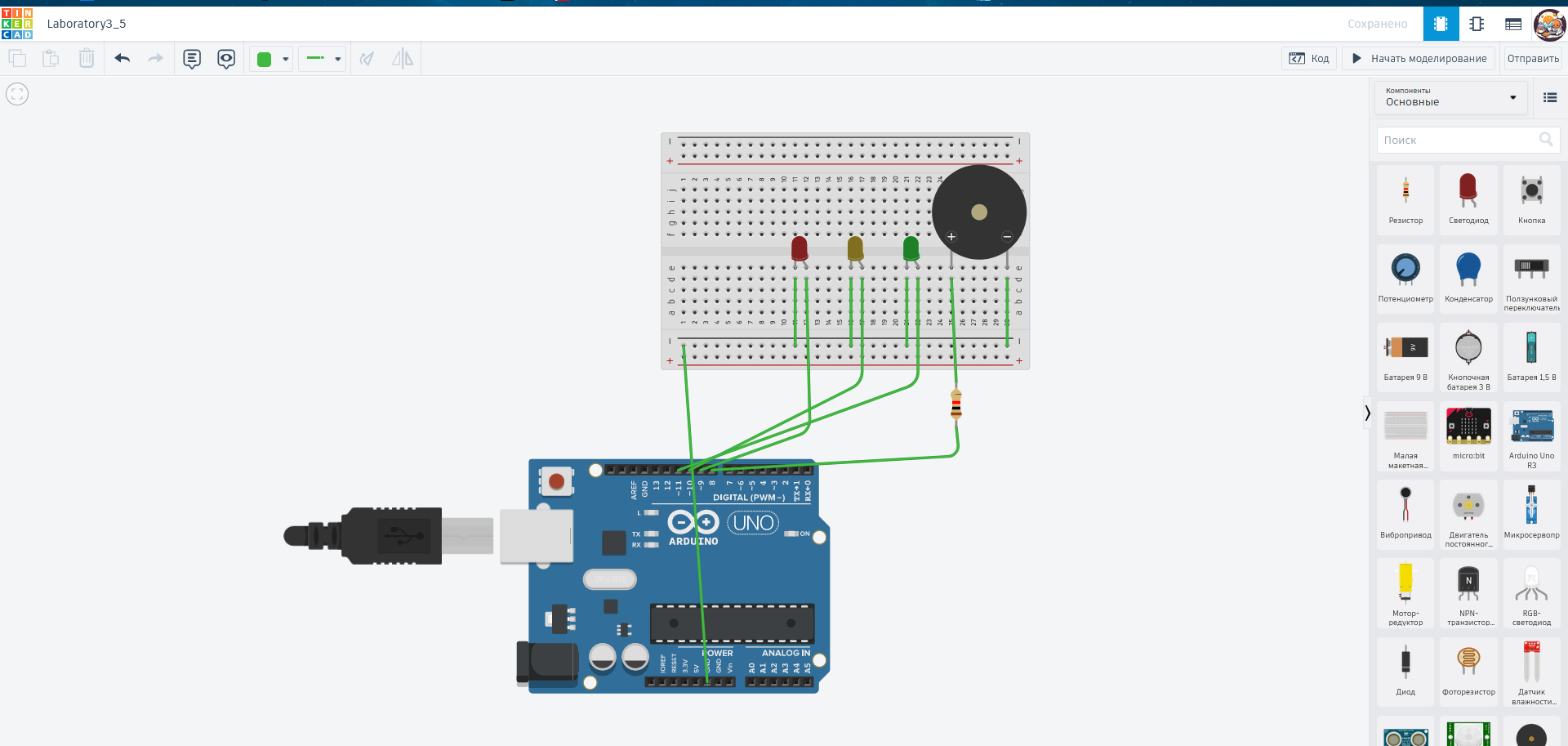


Рисунок 1.3 – интерфейс работы со схемами Arduino в Tinkercad

# Лабораторная работа №2

**Тема**: Мигание светодиодом.

**Цель:** научиться работать с светодиодами и применять их в схемах Arduino, изучить их особенности.

**Ход работы**

Задание 1. К стандартной плате Arduino UNO, которая есть в панели инструментов, подсоединим светодиод, находящийся на той же панели. Расположим светодиод рядом с платой и проведём провода от платы к нему. (Рисунок 2.1)

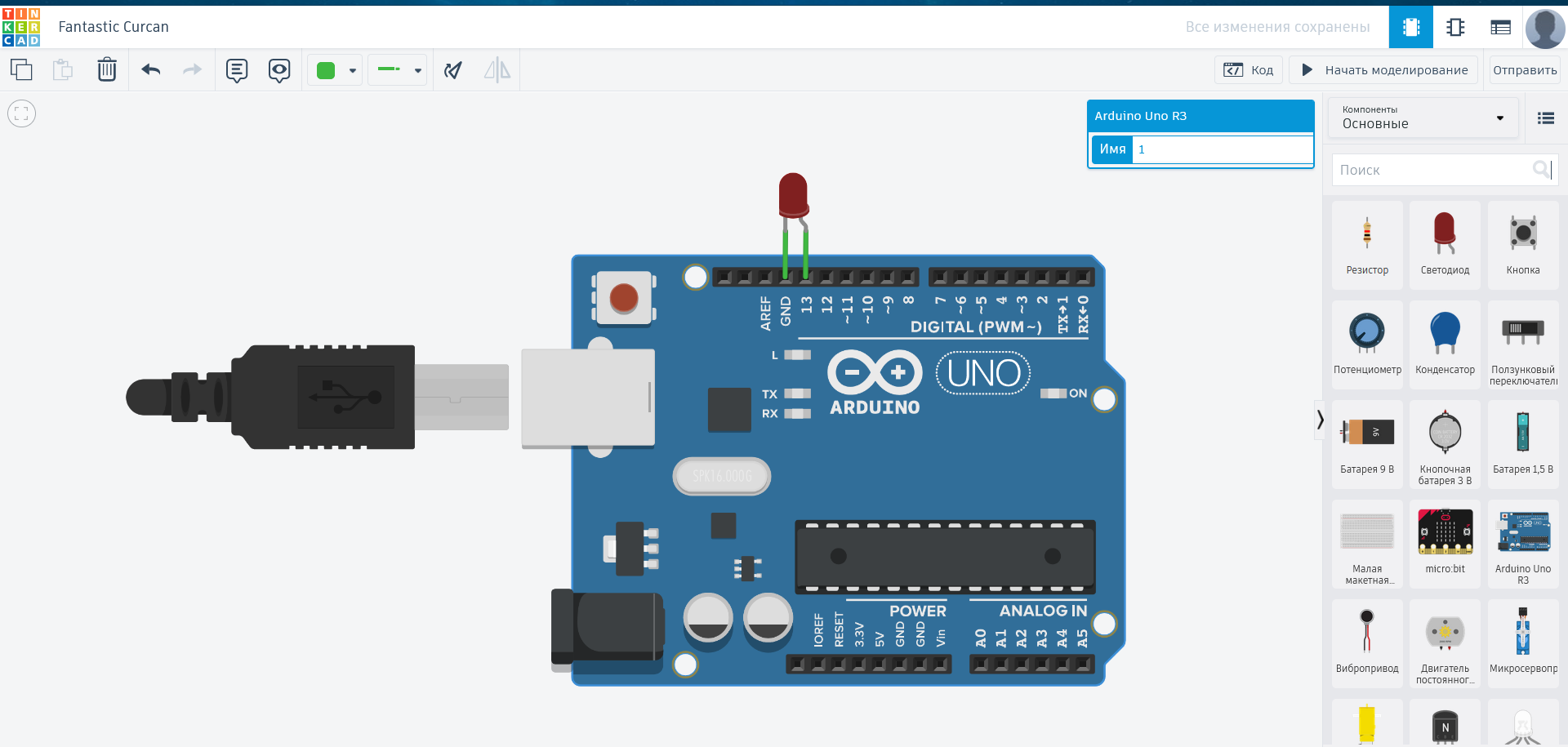


Рисунок 2.1 – Подсоединение светодиода к плате Arduino UNO

Затем, вызывая вкладку «Code» на верхней панели, переключаемся в ней на представление алгоритма работы платы «Blocks+text», чтобы видеть те команды, что мы пропечатываем в алгоритме, и их представление в виде кода. Данные действия показаны на рисунке 2.2.

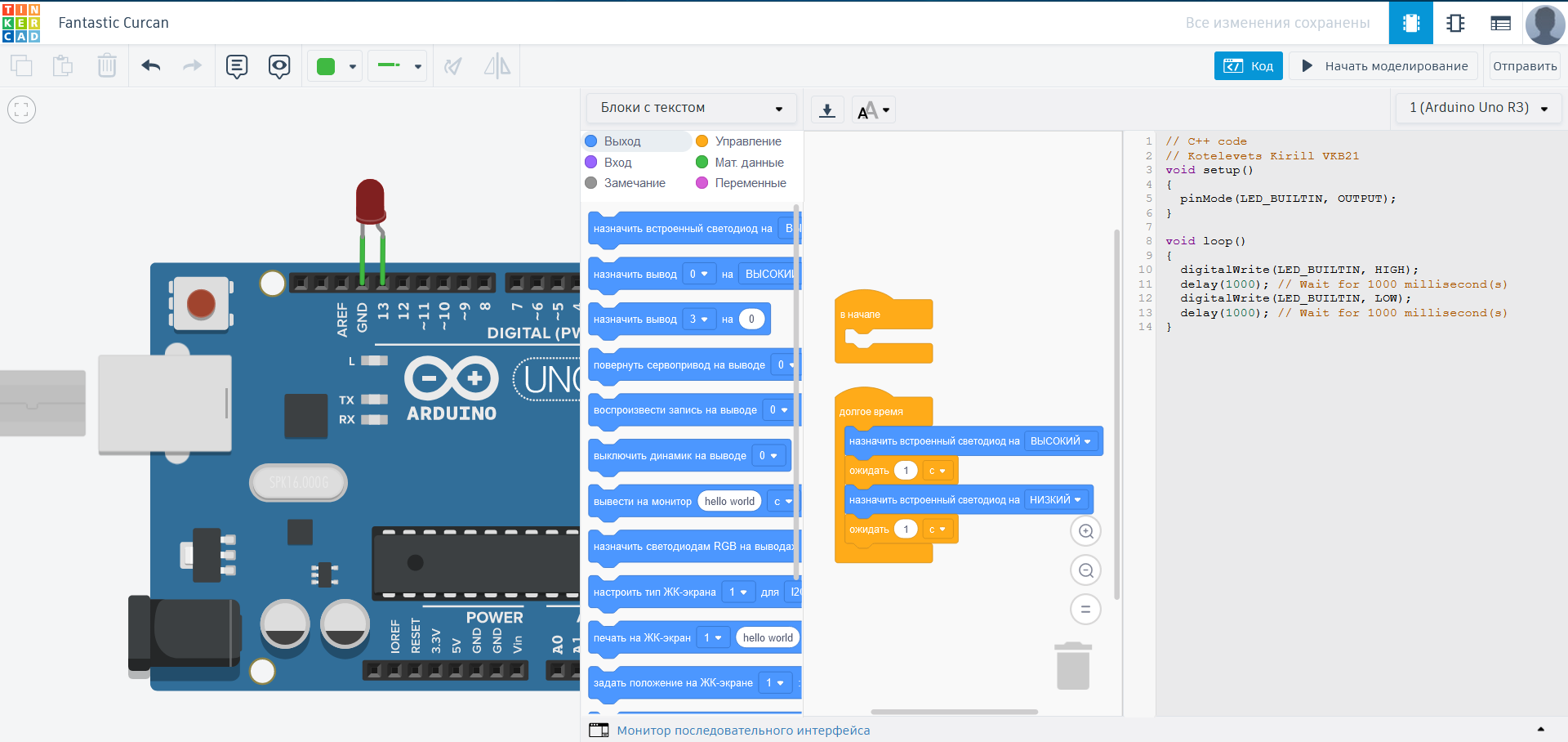


Рисунок 2.2 – Представление алгоритма работы платы в виде “Блоки и тексты”

Запустив программу, можно заметить, что светодиод будет мигать каждую секунду. Запрограммируем светодиод так, чтобы он выдавал знак SOS, т.е. в таком порядке мигания: пауза, 3 коротких, 3 длинных, 3 коротких (Рисунок 2.3)

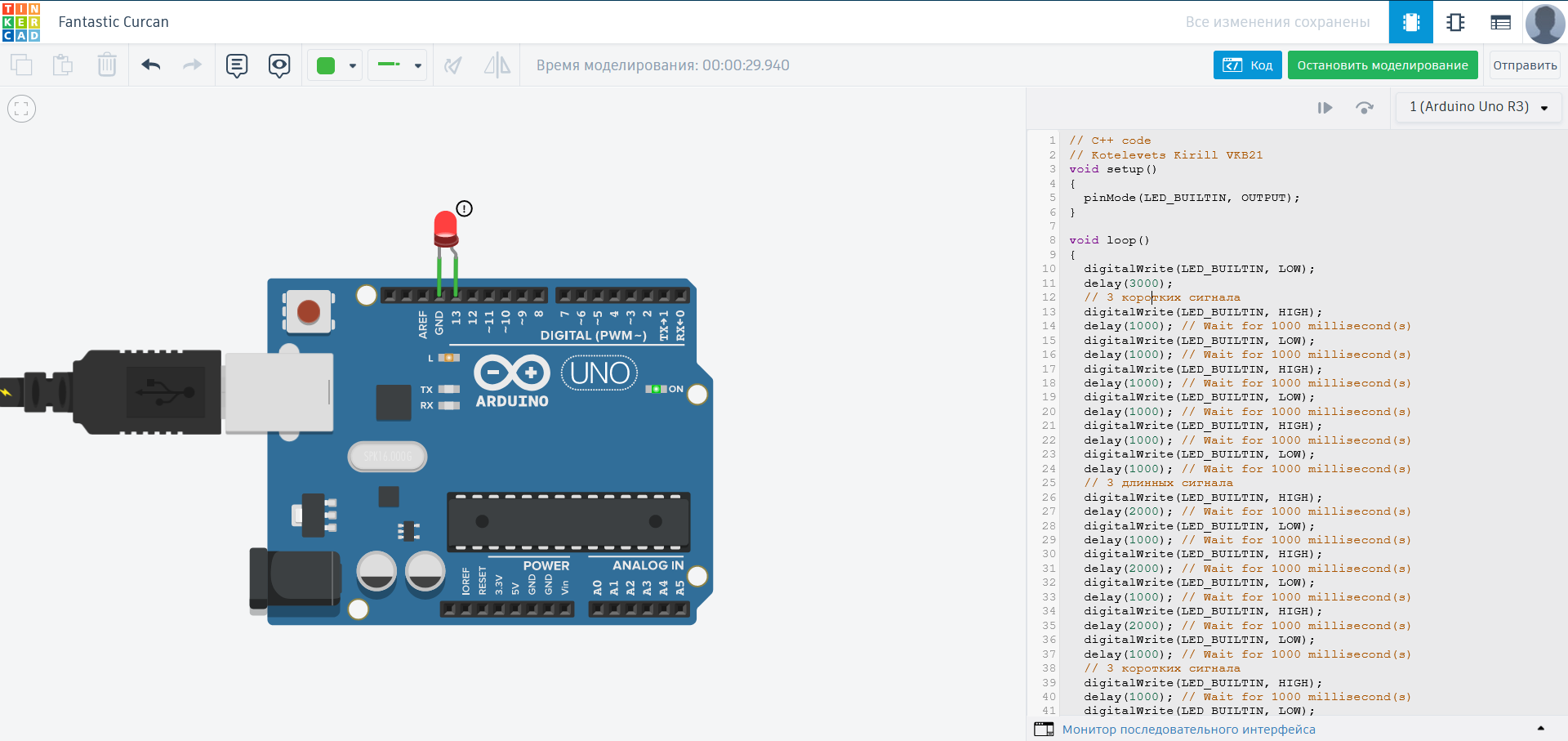


Рисунок 2.3 – Алгоритм работы сигнала SOS

Листинг программы:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | // C++ code  // Kotelevets Kirill VKB21  **void** **setup**()  {  pinMode(LED\_BUILTIN, OUTPUT);  }  **void** **loop**()  {  digitalWrite(LED\_BUILTIN, LOW);  delay(**3000**);  // 3 коротких сигнала  digitalWrite(LED\_BUILTIN, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  // 3 длинных сигнала  digitalWrite(LED\_BUILTIN, HIGH);  delay(**2000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, HIGH);  delay(**2000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, HIGH);  delay(**2000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  // 3 коротких сигнала  digitalWrite(LED\_BUILTIN, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(LED\_BUILTIN, LOW);  delay(**1000**); // Wait for 1000 millisecond(s)  } | |

Задание 2. Необходимо составить схему и код для трех цветных светодиода типа “Светофор”. На рисунке 2.4 представлена схема и ее блок.

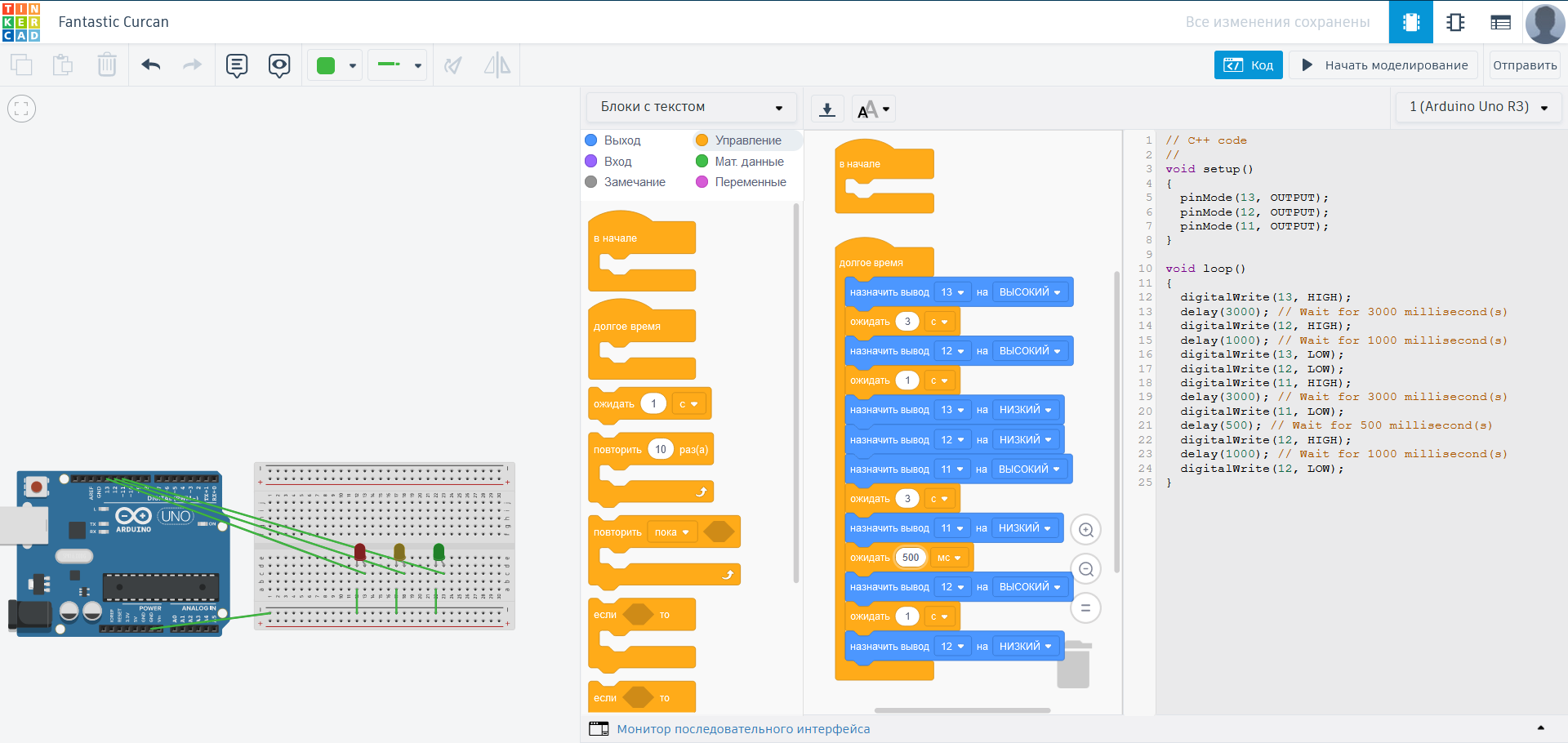


Рисунок 2.4 – Алгоритм работы Светофора

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | // C++ code  // Kotelevets Kirill VKB21  **void** **setup**()  {  pinMode(**13**, OUTPUT);  pinMode(**12**, OUTPUT);  pinMode(**11**, OUTPUT);  }  **void** **loop**()  {  digitalWrite(**13**, HIGH);  delay(**3000**); // Wait for 3000 millisecond(s)  digitalWrite(**12**, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(**13**, LOW);  digitalWrite(**12**, LOW);  digitalWrite(**11**, HIGH);  delay(**3000**); // Wait for 3000 millisecond(s)  digitalWrite(**11**, LOW);  delay(**500**); // Wait for 500 millisecond(s)  digitalWrite(**12**, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(**12**, LOW);  } | |

# Лабораторная работа №3

**Тема**: Звуковые сигналы

**Цель:** научиться работать с пьезоэлементами и применять их в схемах Arduino, изучить особенности работы со звуковыми сигналами и нотами.

**Ход работы**

Задание 1. Используя два пьезоэлемента, запрограммировать плату на воспроизведение сигнала SOS в нескольких разных тональностях. На рисунке 3.1 приставлен код и схема.

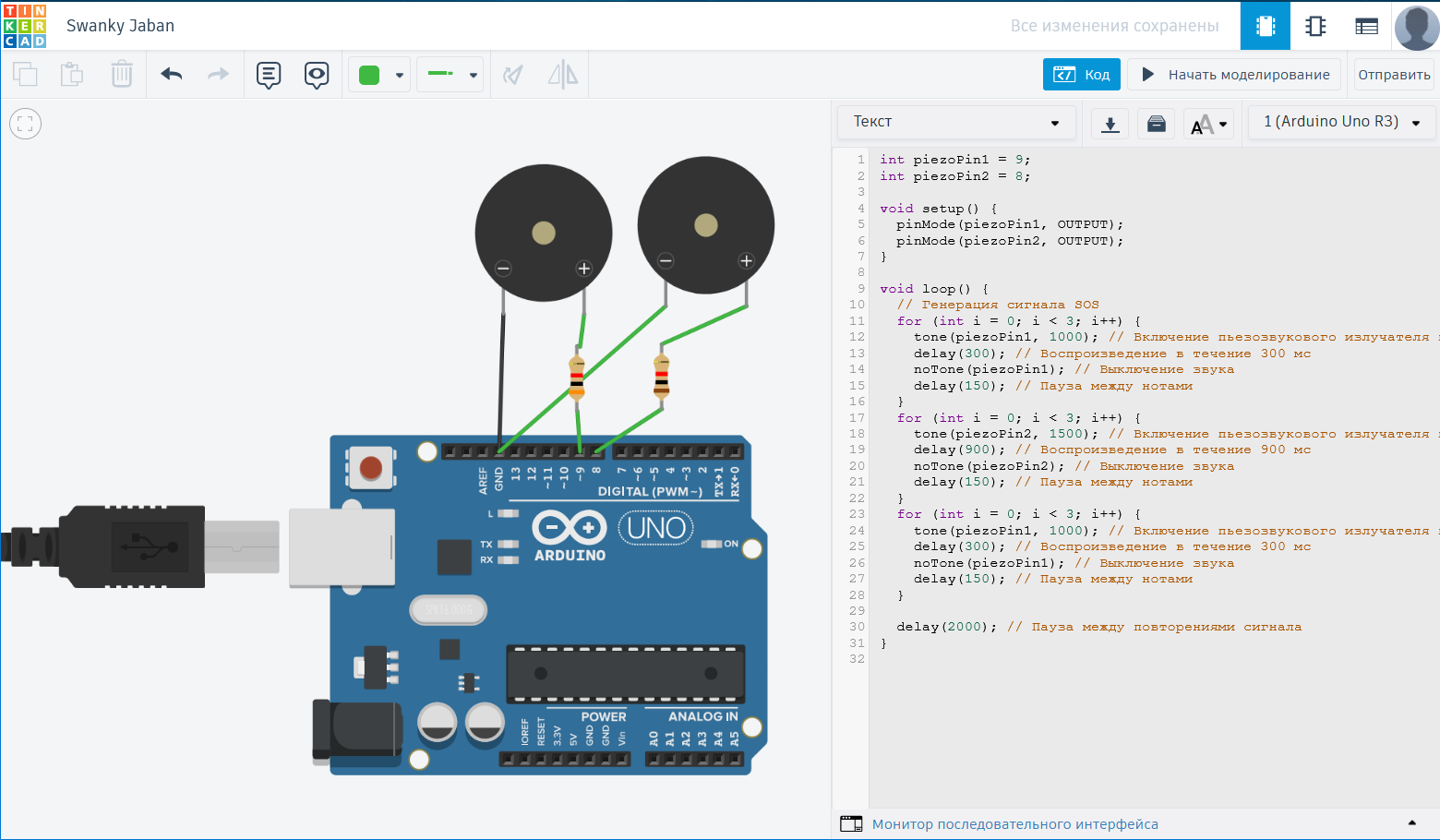


Рисунок 3.1 – схема и код программы для задания 1.

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | **int** piezoPin1 = **9**;  **int** piezoPin2 = **8**;  **void** **setup**() {  pinMode(piezoPin1, OUTPUT);  pinMode(piezoPin2, OUTPUT);  }  **void** **loop**() {  // Генерация сигнала SOS  **for** (**int** i = **0**; i < **3**; i++) {  tone(piezoPin1, **1000**); // Включение пьезозвукового излучателя на частоте 1000 Гц (нота A5)  delay(**300**); // Воспроизведение в течение 300 мс  noTone(piezoPin1); // Выключение звука  delay(**150**); // Пауза между нотами  }  **for** (**int** i = **0**; i < **3**; i++) {  tone(piezoPin2, **1500**); // Включение пьезозвукового излучателя на частоте 1500 Гц (нота C6)  delay(**900**); // Воспроизведение в течение 900 мс  noTone(piezoPin2); // Выключение звука  delay(**150**); // Пауза между нотами  }  **for** (**int** i = **0**; i < **3**; i++) {  tone(piezoPin1, **1000**); // Включение пьезозвукового излучателя на частоте 1000 Гц (нота A5)  delay(**300**); // Воспроизведение в течение 300 мс  noTone(piezoPin1); // Выключение звука  delay(**150**); // Пауза между нотами  }  delay(**2000**); // Пауза между повторениями сигнала  } | |

Задание 2. Необходимо использовать один пьезоэлемент при программировании платы на воспроизведение всех восьми нот. Схема и код представлены на рисунке 3.2.

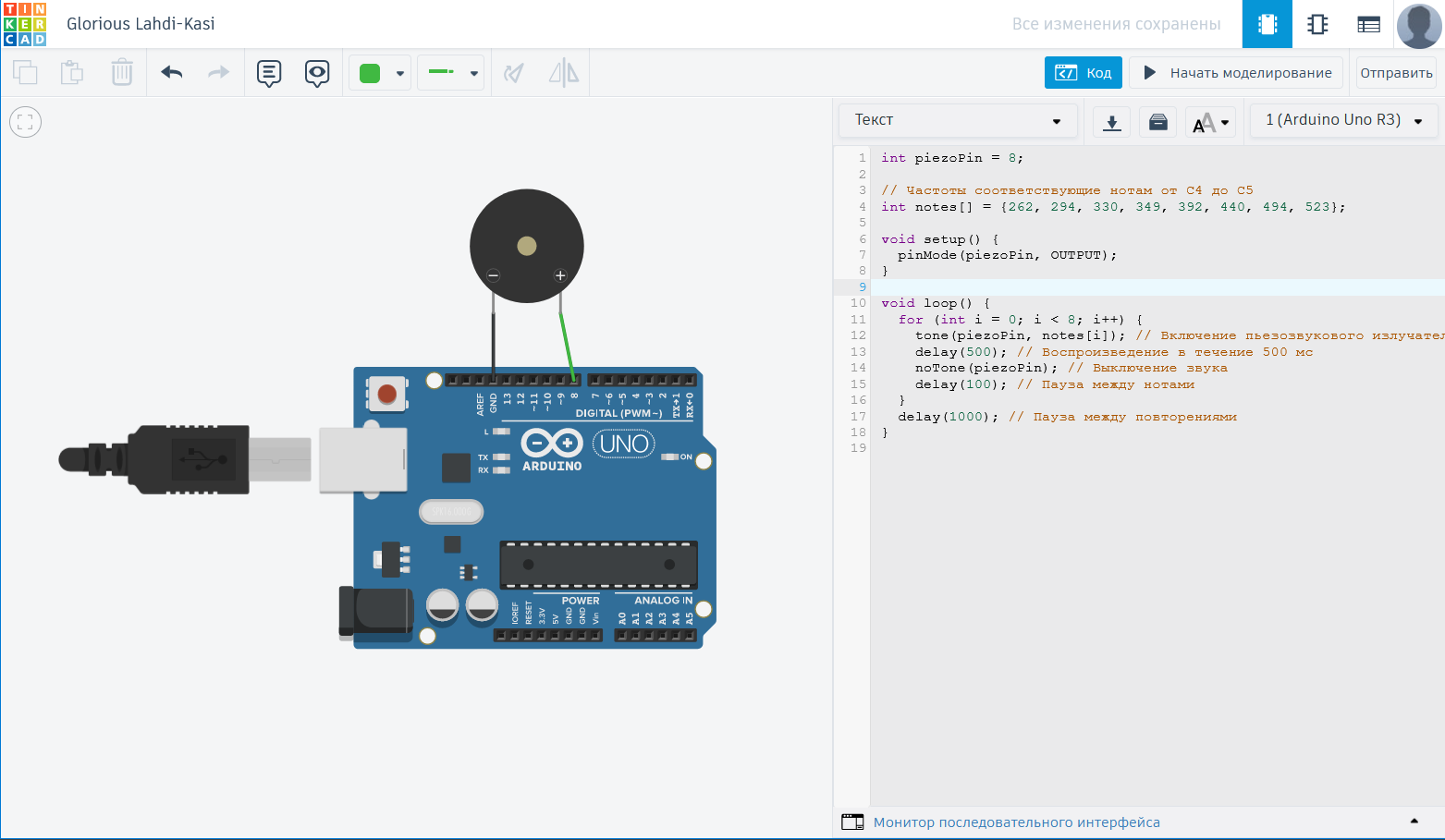


Рисунок 3.2 – схема и код программы для задания 2.

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **int** piezoPin = **8**;  // Частоты соответствующие нотам от C4 до C5  **int** notes[] = {**262**, **294**, **330**, **349**, **392**, **440**, **494**, **523**};  **void** **setup**() {  pinMode(piezoPin, OUTPUT);  }  **void** **loop**() {  **for** (**int** i = **0**; i < **8**; i++) {  tone(piezoPin, notes[i]); // Включение пьезозвукового излучателя на частоте, соответствующей текущей ноте  delay(**500**); // Воспроизведение в течение 500 мс  noTone(piezoPin); // Выключение звука  delay(**100**); // Пауза между нотами  }  delay(**1000**); // Пауза между повторениями  } | |

Задание 3. Необходимо запрограммировать плату на воспроизведение короткого фрагмента, выбранной мною музыкальной композиции. На рисунке 3.3 представлена схема и код программы. Композиция “Twinkle, Twinkle, Little Star”.

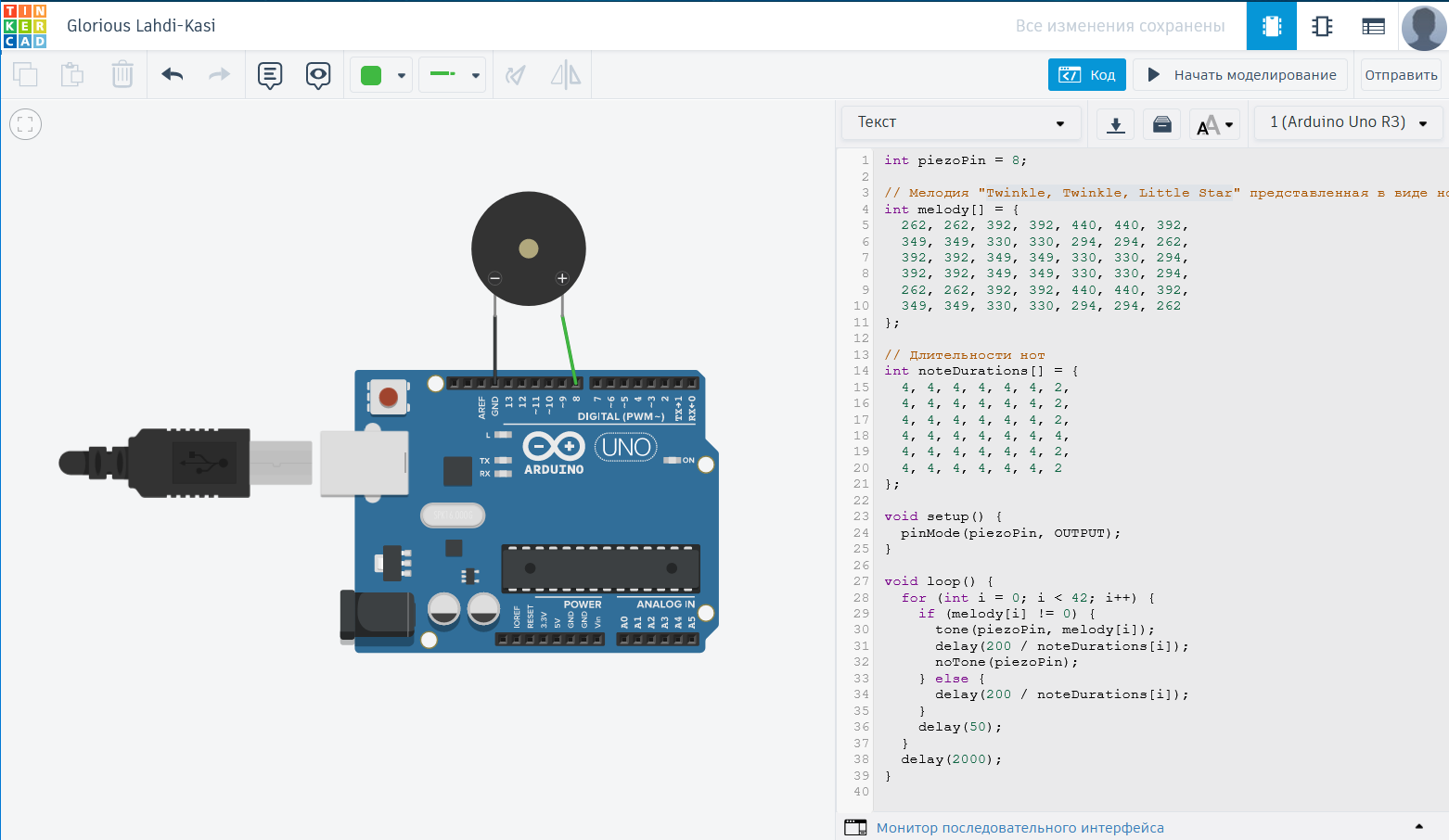


Рисунок 3.3 – схема и код программы для задания 3.

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | **int** piezoPin = **8**;  // Мелодия "Twinkle, Twinkle, Little Star"  **int** melody[] = {  **262**, **262**, **392**, **392**, **440**, **440**, **392**,  **349**, **349**, **330**, **330**, **294**, **294**, **262**,  **392**, **392**, **349**, **349**, **330**, **330**, **294**,  **392**, **392**, **349**, **349**, **330**, **330**, **294**,  **262**, **262**, **392**, **392**, **440**, **440**, **392**,  **349**, **349**, **330**, **330**, **294**, **294**, **262**  };  // Длительности нот  **int** noteDurations[] = {  **4**, **4**, **4**, **4**, **4**, **4**, **2**,  **4**, **4**, **4**, **4**, **4**, **4**, **2**,  **4**, **4**, **4**, **4**, **4**, **4**, **2**,  **4**, **4**, **4**, **4**, **4**, **4**, **4**,  **4**, **4**, **4**, **4**, **4**, **4**, **2**,  **4**, **4**, **4**, **4**, **4**, **4**, **2**  };  **void** **setup**() {  pinMode(piezoPin, OUTPUT);  }  **void** **loop**() {  **for** (**int** i = **0**; i < **42**; i++) {  **if** (melody[i] != **0**) {  tone(piezoPin, melody[i]);  delay(**200** / noteDurations[i]);  noTone(piezoPin);  } **else** {  delay(**200** / noteDurations[i]);  }  delay(**50**);  }  delay(**2000**);  } | |

Задание 4. Повторим прошлое задание, но с другой мелодией из “Звездных воин”. На рисунке 3.4 представлена схема и код.

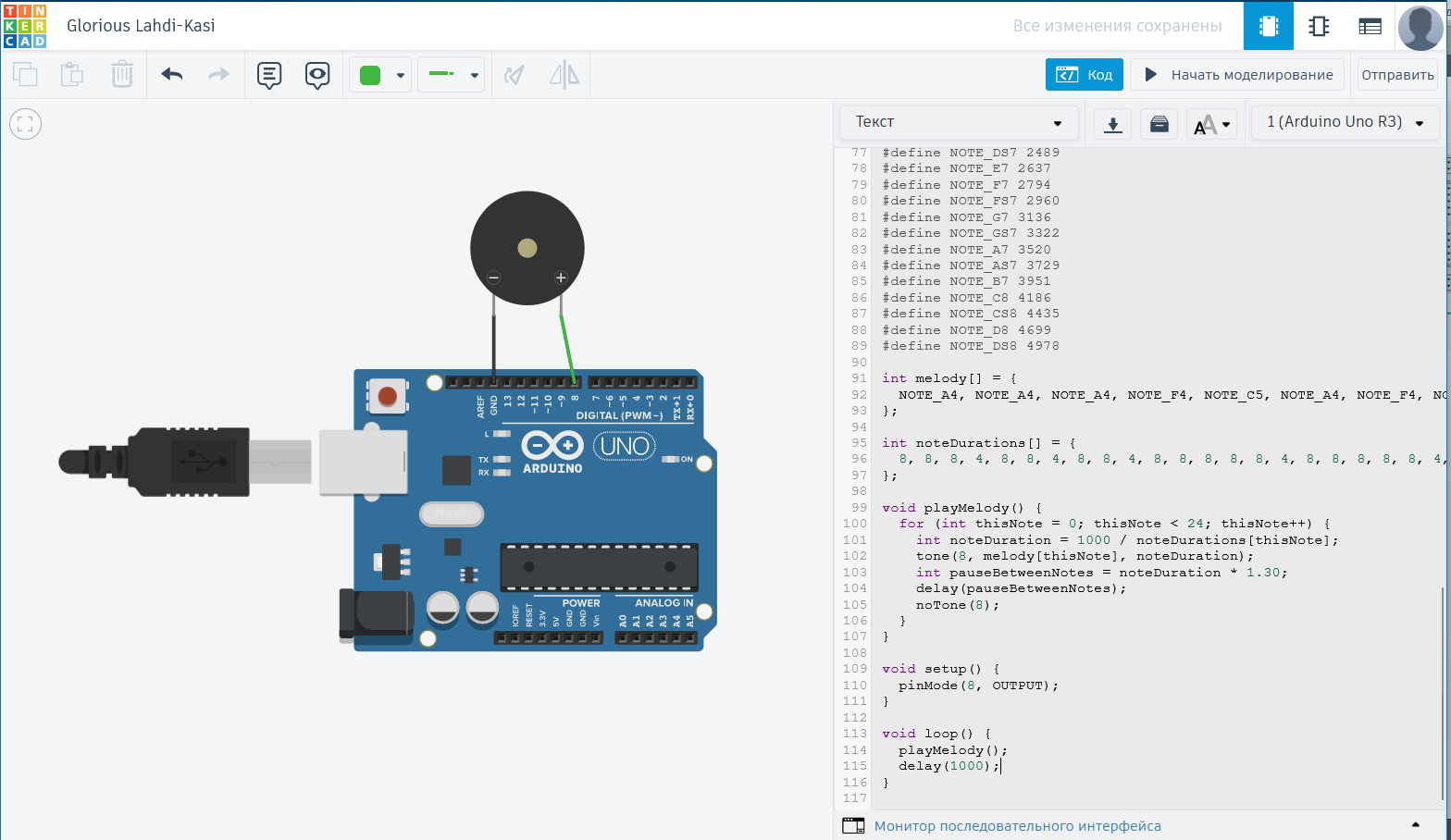


Рисунок 3.4 – схема и код программы для задания 4

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116 | #define NOTE\_B0 31  #define NOTE\_C1 33  #define NOTE\_CS1 35  #define NOTE\_D1 37  #define NOTE\_DS1 39  #define NOTE\_E1 41  #define NOTE\_F1 44  #define NOTE\_FS1 46  #define NOTE\_G1 49  #define NOTE\_GS1 52  #define NOTE\_A1 55  #define NOTE\_AS1 58  #define NOTE\_B1 62  #define NOTE\_C2 65  #define NOTE\_CS2 69  #define NOTE\_D2 73  #define NOTE\_DS2 78  #define NOTE\_E2 82  #define NOTE\_F2 87  #define NOTE\_FS2 93  #define NOTE\_G2 98  #define NOTE\_GS2 104  #define NOTE\_A2 110  #define NOTE\_AS2 117  #define NOTE\_B2 123  #define NOTE\_C3 131  #define NOTE\_CS3 139  #define NOTE\_D3 147  #define NOTE\_DS3 156  #define NOTE\_E3 165  #define NOTE\_F3 175  #define NOTE\_FS3 185  #define NOTE\_G3 196  #define NOTE\_GS3 208  #define NOTE\_A3 220  #define NOTE\_AS3 233  #define NOTE\_B3 247  #define NOTE\_C4 262  #define NOTE\_CS4 277  #define NOTE\_D4 294  #define NOTE\_DS4 311  #define NOTE\_E4 330  #define NOTE\_F4 349  #define NOTE\_FS4 370  #define NOTE\_G4 392  #define NOTE\_GS4 415  #define NOTE\_A4 440  #define NOTE\_AS4 466  #define NOTE\_B4 494  #define NOTE\_C5 523  #define NOTE\_CS5 554  #define NOTE\_D5 587  #define NOTE\_DS5 622  #define NOTE\_E5 659  #define NOTE\_F5 698  #define NOTE\_FS5 740  #define NOTE\_G5 784  #define NOTE\_GS5 831  #define NOTE\_A5 880  #define NOTE\_AS5 932  #define NOTE\_B5 988  #define NOTE\_C6 1047  #define NOTE\_CS6 1109  #define NOTE\_D6 1175  #define NOTE\_DS6 1245  #define NOTE\_E6 1319  #define NOTE\_F6 1397  #define NOTE\_FS6 1480  #define NOTE\_G6 1568  #define NOTE\_GS6 1661  #define NOTE\_A6 1760  #define NOTE\_AS6 1865  #define NOTE\_B6 1976  #define NOTE\_C7 2093  #define NOTE\_CS7 2217  #define NOTE\_D7 2349  #define NOTE\_DS7 2489  #define NOTE\_E7 2637  #define NOTE\_F7 2794  #define NOTE\_FS7 2960  #define NOTE\_G7 3136  #define NOTE\_GS7 3322  #define NOTE\_A7 3520  #define NOTE\_AS7 3729  #define NOTE\_B7 3951  #define NOTE\_C8 4186  #define NOTE\_CS8 4435  #define NOTE\_D8 4699  #define NOTE\_DS8 4978  **int** melody[] = {  NOTE\_A4, NOTE\_A4, NOTE\_A4, NOTE\_F4, NOTE\_C5, NOTE\_A4, NOTE\_F4, NOTE\_C5, NOTE\_A4, NOTE\_E5, NOTE\_E5, NOTE\_E5, NOTE\_F5, NOTE\_C5, NOTE\_G4, NOTE\_F4, NOTE\_C5, NOTE\_A4, NOTE\_A4, NOTE\_A4, NOTE\_A4, NOTE\_G4, NOTE\_F4, NOTE\_G4  };  **int** noteDurations[] = {  **8**, **8**, **8**, **4**, **8**, **8**, **4**, **8**, **8**, **4**, **8**, **8**, **8**, **8**, **8**, **4**, **8**, **8**, **8**, **8**, **8**, **4**, **4**, **4**  };  **void** **playMelody**() {  **for** (**int** thisNote = **0**; thisNote < **24**; thisNote++) {  **int** noteDuration = **1000** / noteDurations[thisNote];  tone(**8**, melody[thisNote], noteDuration);  **int** pauseBetweenNotes = noteDuration \* **1.30**;  delay(pauseBetweenNotes);  noTone(**8**);  }  }  **void** **setup**() {  pinMode(**8**, OUTPUT);  }  **void** **loop**() {  playMelody();  delay(**1000**);  } | |

Задание 5. Сделать светофор для незрячих. Запрограммировать плату на звучание определённой тональности, когда загорается один из трёх цветов светофора. На рисунке 3.5 представлена схема и код программы.

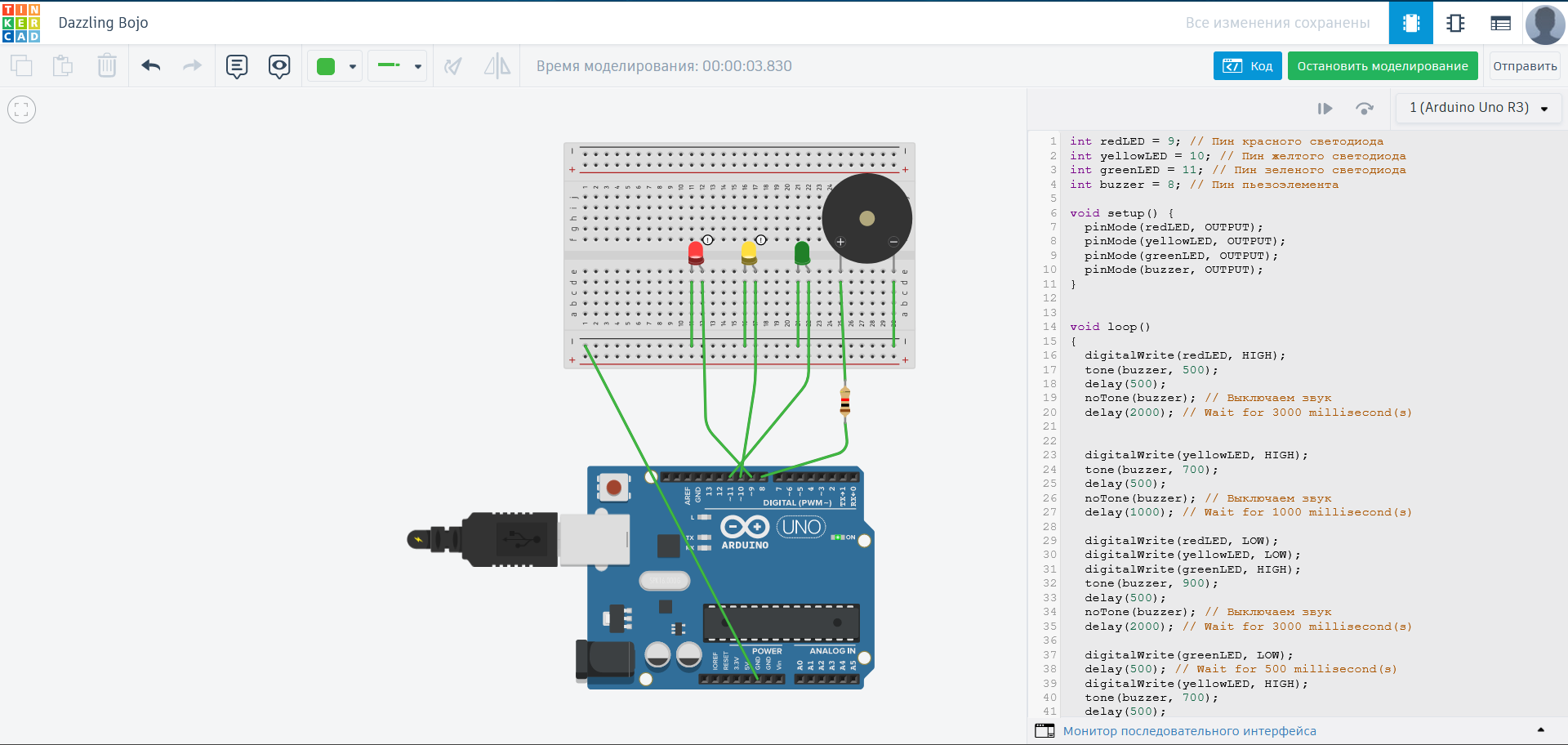


Рисунок 3.5 – схема и код программы для задания 5

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | **int** redLED = **9**; // Пин красного светодиода  **int** yellowLED = **10**; // Пин желтого светодиода  **int** greenLED = **11**; // Пин зеленого светодиода  **int** buzzer = **8**; // Пин пьезоэлемента  **void** **setup**() {  pinMode(redLED, OUTPUT);  pinMode(yellowLED, OUTPUT);  pinMode(greenLED, OUTPUT);  pinMode(buzzer, OUTPUT);  }  **void** **loop**()  {  digitalWrite(redLED, HIGH);  tone(buzzer, **500**);  delay(**500**);  noTone(buzzer); // Выключаем звук  delay(**2000**); // Wait for 3000 millisecond(s)  digitalWrite(yellowLED, HIGH);  tone(buzzer, **700**);  delay(**500**);  noTone(buzzer); // Выключаем звук  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(redLED, LOW);  digitalWrite(yellowLED, LOW);  digitalWrite(greenLED, HIGH);  tone(buzzer, **900**);  delay(**500**);  noTone(buzzer); // Выключаем звук  delay(**2000**); // Wait for 3000 millisecond(s)  digitalWrite(greenLED, LOW);  delay(**500**); // Wait for 500 millisecond(s)  digitalWrite(yellowLED, HIGH);  tone(buzzer, **700**);  delay(**500**);  noTone(buzzer); // Выключаем звук  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(yellowLED, LOW);  } | |

# Лабораторная работа №4

**Тема**: Работа с кнопками

**Цель:** ознакомиться с работой и применением кнопок на платах Arduino, научиться их использовать.

**Ход работы**

Задание 1. При нажатии кнопки запустить работу светофора. Для этого использую уже работающий проект светофора, просто немного его модернизирую с помощью кнопки для ее запуска. На рисунке 4.1 представлена схема и код программы. При нажатии кнопки срабатывает блок программы, которая 3 раза запускает алгоритм работы светофора.

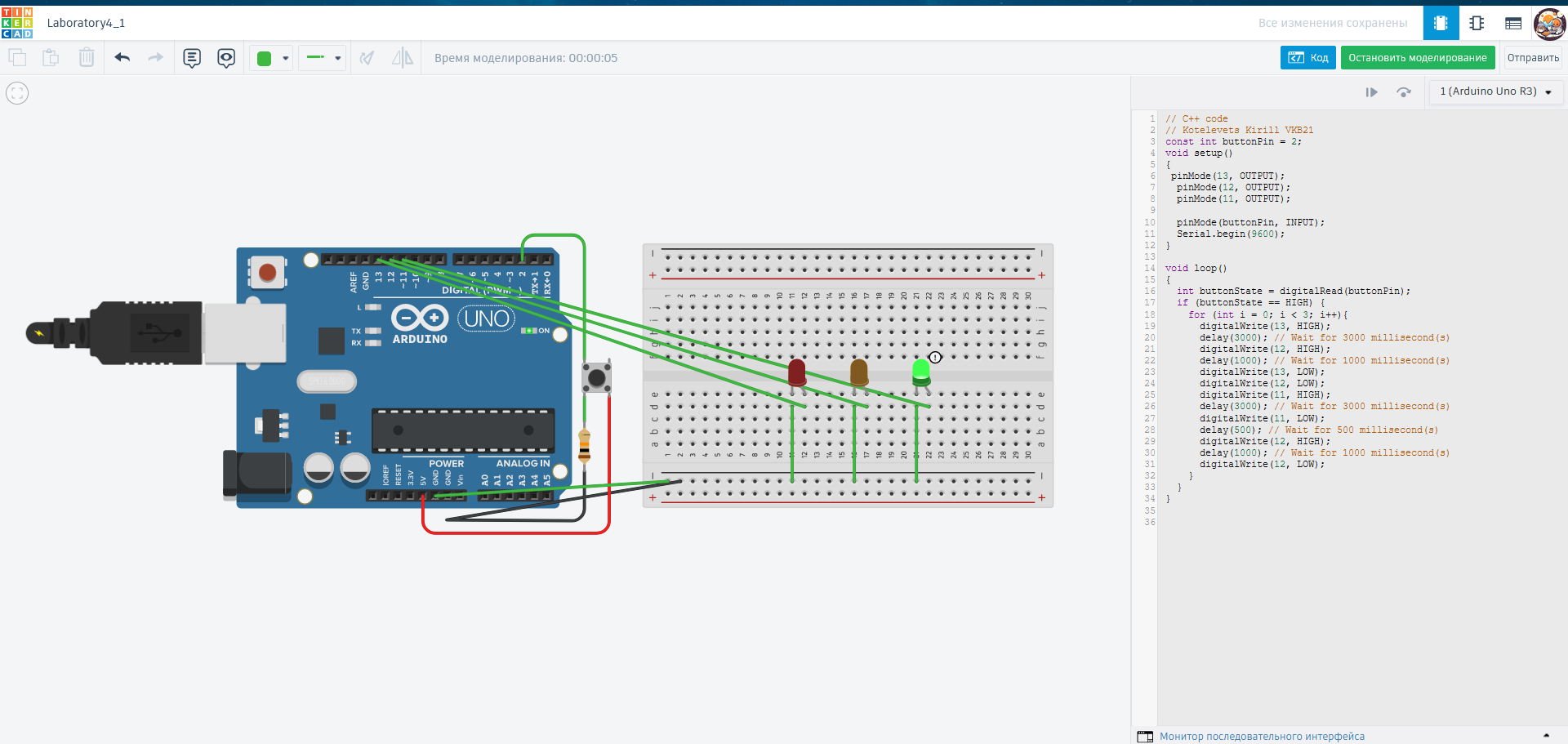


Рисунок 4.1 – схема и код программы для задания 1

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | // C++ code  // Kotelevets Kirill VKB21  **const** **int** buttonPin = **2**;  **void** **setup**()  {  pinMode(**13**, OUTPUT);  pinMode(**12**, OUTPUT);  pinMode(**11**, OUTPUT);  pinMode(buttonPin, INPUT);  Serial.begin(**9600**);  }  **void** **loop**()  {  **int** buttonState = digitalRead(buttonPin);  **if** (buttonState == HIGH) {  **for** (**int** i = **0**; i < **3**; i++){  digitalWrite(**13**, HIGH);  delay(**3000**); // Wait for 3000 millisecond(s)  digitalWrite(**12**, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(**13**, LOW);  digitalWrite(**12**, LOW);  digitalWrite(**11**, HIGH);  delay(**3000**); // Wait for 3000 millisecond(s)  digitalWrite(**11**, LOW);  delay(**500**); // Wait for 500 millisecond(s)  digitalWrite(**12**, HIGH);  delay(**1000**); // Wait for 1000 millisecond(s)  digitalWrite(**12**, LOW);  }  }  } | |

Задание 2. При нажатии кнопки выполнить одно из заданий из лаб. работы №3. В моем случае это будет задание 2. На рисунке 4.2 представлены схемы и блок программы.

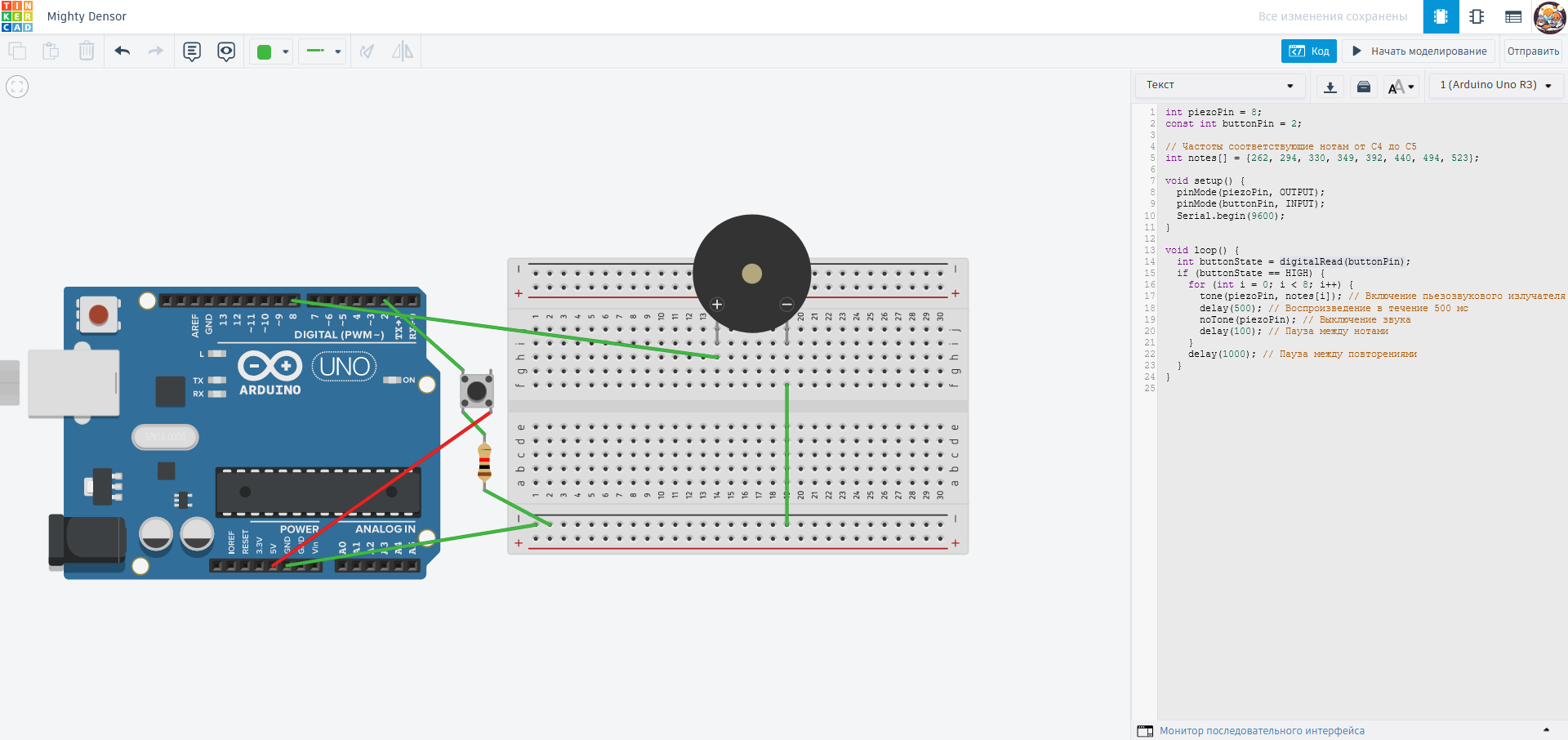


Рисунок 4.2 – схема и код программы для задания 2

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | **int** piezoPin = **8**;  **const** **int** buttonPin = **2**;  // Частоты соответствующие нотам от C4 до C5  **int** notes[] = {**262**, **294**, **330**, **349**, **392**, **440**, **494**, **523**};  **void** **setup**() {  pinMode(piezoPin, OUTPUT);  pinMode(buttonPin, INPUT);  Serial.begin(**9600**);  }  **void** **loop**() {  **int** buttonState = digitalRead(buttonPin);  **if** (buttonState == HIGH) {  **for** (**int** i = **0**; i < **8**; i++) {  tone(piezoPin, notes[i]);  delay(**500**); // Воспроизведение в течение 500 мс  noTone(piezoPin); // Выключение звука  delay(**100**); // Пауза между нотами  }  delay(**1000**); // Пауза между повторениями  }  } | |

Задание 3. Используя несколько кнопок, необходимо запрограммировать определённое преподавателем действие при нажатии одной из кнопок. В моем случае

…

Задание 4. Необходимо подключить 8 кнопок и один пьезоэлемент, используя бредборд больший. Эти кнопки должны символизировать ноты. Задача – симулировать игру на пианино. На рисунке 4.4 представлена схема и код программы.

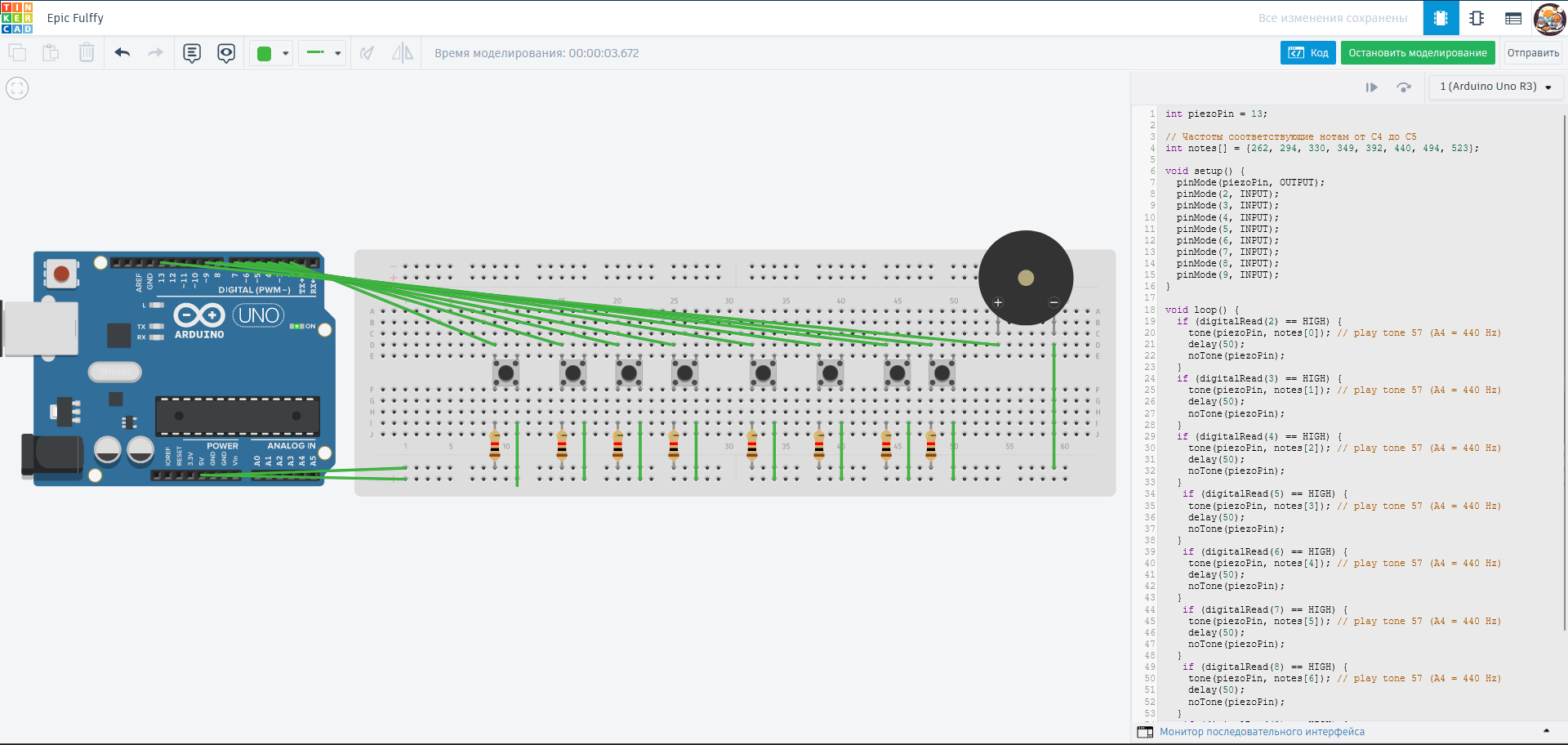


Рисунок 4.4 – схема и код программы для задания 4

Листинг программы

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | **int** piezoPin = **13**;  // Частоты соответствующие нотам от C4 до C5  **int** notes[] = {**262**, **294**, **330**, **349**, **392**, **440**, **494**, **523**};  **void** **setup**() {  pinMode(piezoPin, OUTPUT);  pinMode(**2**, INPUT);  pinMode(**3**, INPUT);  pinMode(**4**, INPUT);  pinMode(**5**, INPUT);  pinMode(**6**, INPUT);  pinMode(**7**, INPUT);  pinMode(**8**, INPUT);  pinMode(**9**, INPUT);  }  **void** **loop**() {  **if** (digitalRead(**2**) == HIGH) {  tone(piezoPin, notes[**0**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**3**) == HIGH) {  tone(piezoPin, notes[**1**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**4**) == HIGH) {  tone(piezoPin, notes[**2**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**5**) == HIGH) {  tone(piezoPin, notes[**3**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**6**) == HIGH) {  tone(piezoPin, notes[**4**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**7**) == HIGH) {  tone(piezoPin, notes[**5**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**8**) == HIGH) {  tone(piezoPin, notes[**6**]);  delay(**50**);  noTone(piezoPin);  }  **if** (digitalRead(**9**) == HIGH) {  tone(piezoPin, notes[**7**]);  delay(**50**);  noTone(piezoPin);  }  } | |

# Лабораторная работа №5

**Тема**: работа с ЖК дисплеем

**Цель:** ознакомиться с работой и применением ЖК дисплея на платах Arduino, научиться его использовать.

**Ход работы:**

Задание 1. Нужно использовать несколько кнопок и при нажатии одной из них высвечивать на дисплее номер кнопки. На рисунке 5.1 представлена схема и код программы.

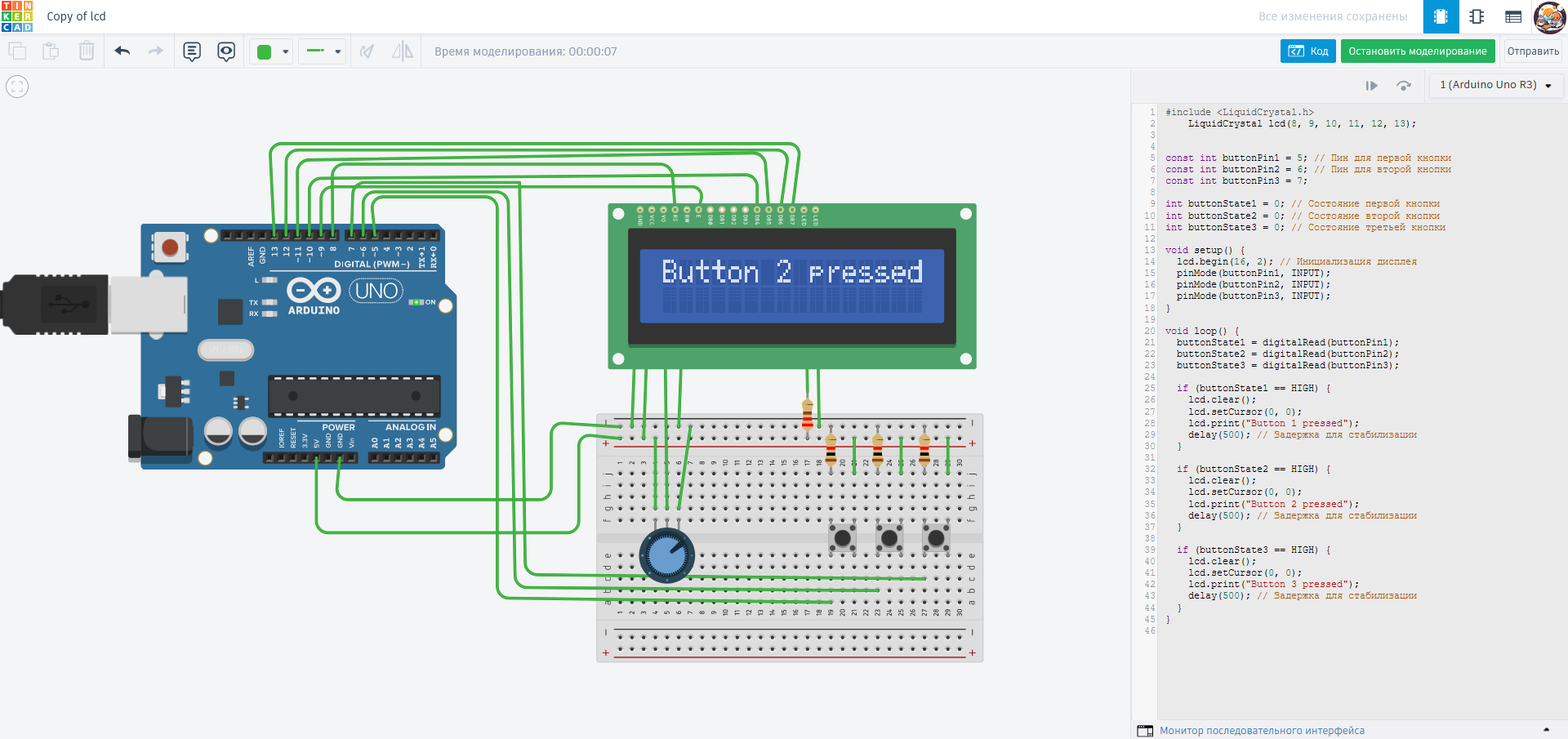


Рисунок 5.1 – Схема и код для первого задания

Листинг программы

|  |
| --- |
| #include <LiquidCrystal.h>  LiquidCrystal **lcd**(**8**, **9**, **10**, **11**, **12**, **13**);  **const** **int** buttonPin1 = **5**; // Пин для первой кнопки  **const** **int** buttonPin2 = **6**; // Пин для второй кнопки  **const** **int** buttonPin3 = **7**;  **int** buttonState1 = **0**; // Состояние первой кнопки  **int** buttonState2 = **0**; // Состояние второй кнопки  **int** buttonState3 = **0**; // Состояние третьей кнопки  **void** **setup**() {  lcd.begin(**16**, **2**); // Инициализация дисплея  pinMode(buttonPin1, INPUT);  pinMode(buttonPin2, INPUT);  pinMode(buttonPin3, INPUT);  }  **void** **loop**() {  buttonState1 = digitalRead(buttonPin1);  buttonState2 = digitalRead(buttonPin2);  buttonState3 = digitalRead(buttonPin3);  **if** (buttonState1 == HIGH) {  lcd.clear();  lcd.setCursor(**0**, **0**);  lcd.print("Button 1 pressed");  delay(**500**); // Задержка для стабилизации  }  **if** (buttonState2 == HIGH) {  lcd.clear();  lcd.setCursor(**0**, **0**);  lcd.print("Button 2 pressed");  delay(**500**); // Задержка для стабилизации  }  **if** (buttonState3 == HIGH) {  lcd.clear();  lcd.setCursor(**0**, **0**);  lcd.print("Button 3 pressed");  delay(**500**); // Задержка для стабилизации  }  } |

Задание 2. Использовать пьезоэлементы, звучащие в разных тональностях. На экране высвечивать номер пьезоэлемента и его тональность. На рисунке 5.2 представлена схема и код программы. Программа выдает три тональности: C, E, G.

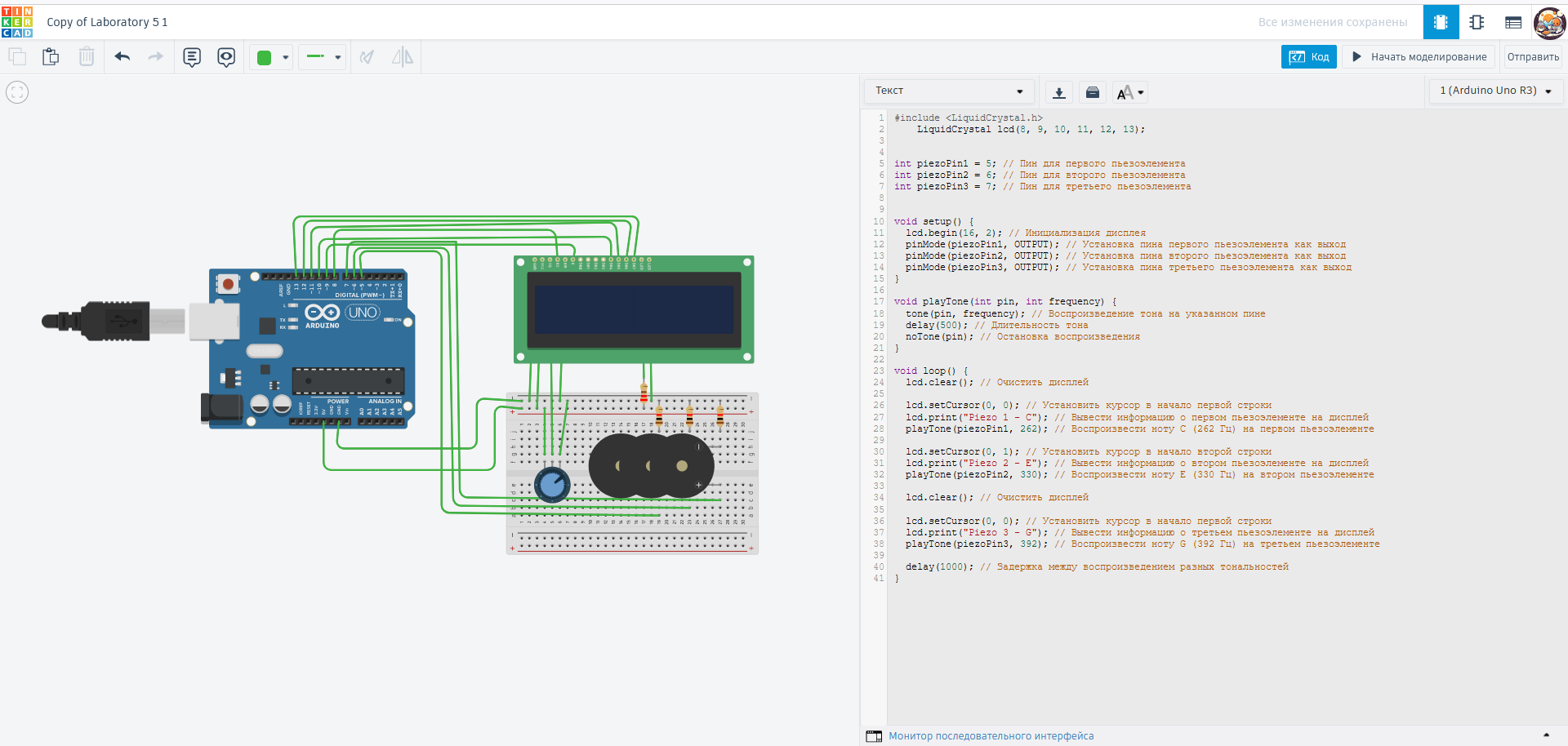


Рисунок 5.2 – Схема и код для второго задания

Листинг программы

|  |
| --- |
| #include <LiquidCrystal.h>  LiquidCrystal **lcd**(**8**, **9**, **10**, **11**, **12**, **13**);  **int** piezoPin1 = **5**; // Пин для первого пьезоэлемента  **int** piezoPin2 = **6**; // Пин для второго пьезоэлемента  **int** piezoPin3 = **7**; // Пин для третьего пьезоэлемента  **void** **setup**() {  lcd.begin(**16**, **2**); // Инициализация дисплея  pinMode(piezoPin1, OUTPUT); // Установка пина первого пьезоэлемента как выход  pinMode(piezoPin2, OUTPUT); // Установка пина второго пьезоэлемента как выход  pinMode(piezoPin3, OUTPUT); // Установка пина третьего пьезоэлемента как выход  }  **void** **playTone**(**int** pin, **int** frequency) {  tone(pin, frequency); // Воспроизведение тона на указанном пине  delay(**500**); // Длительность тона  noTone(pin); // Остановка воспроизведения  }  **void** **loop**() {  lcd.clear(); // Очистить дисплей  lcd.setCursor(**0**, **0**); // Установить курсор в начало первой строки  lcd.print("Piezo 1 - C"); // Вывести информацию о первом пьезоэлементе на дисплей  playTone(piezoPin1, **262**); // Воспроизвести ноту C (262 Гц) на первом пьезоэлементе  lcd.setCursor(**0**, **1**); // Установить курсор в начало второй строки  lcd.print("Piezo 2 - E"); // Вывести информацию о втором пьезоэлементе на дисплей  playTone(piezoPin2, **330**); // Воспроизвести ноту E (330 Гц) на втором пьезоэлементе  lcd.clear(); // Очистить дисплей  lcd.setCursor(**0**, **0**); // Установить курсор в начало первой строки  lcd.print("Piezo 3 - G"); // Вывести информацию о третьем пьезоэлементе на дисплей  playTone(piezoPin3, **392**); // Воспроизвести ноту G (392 Гц) на третьем пьезоэлементе  delay(**1000**); // Задержка между воспроизведением разных тональностей  } |

Задание 3. Необходимо подсоединить к плате несколько светодиодов, на экран выводить номер и цвет светодиода. На рисунке 5.3 показана схема и код программы.

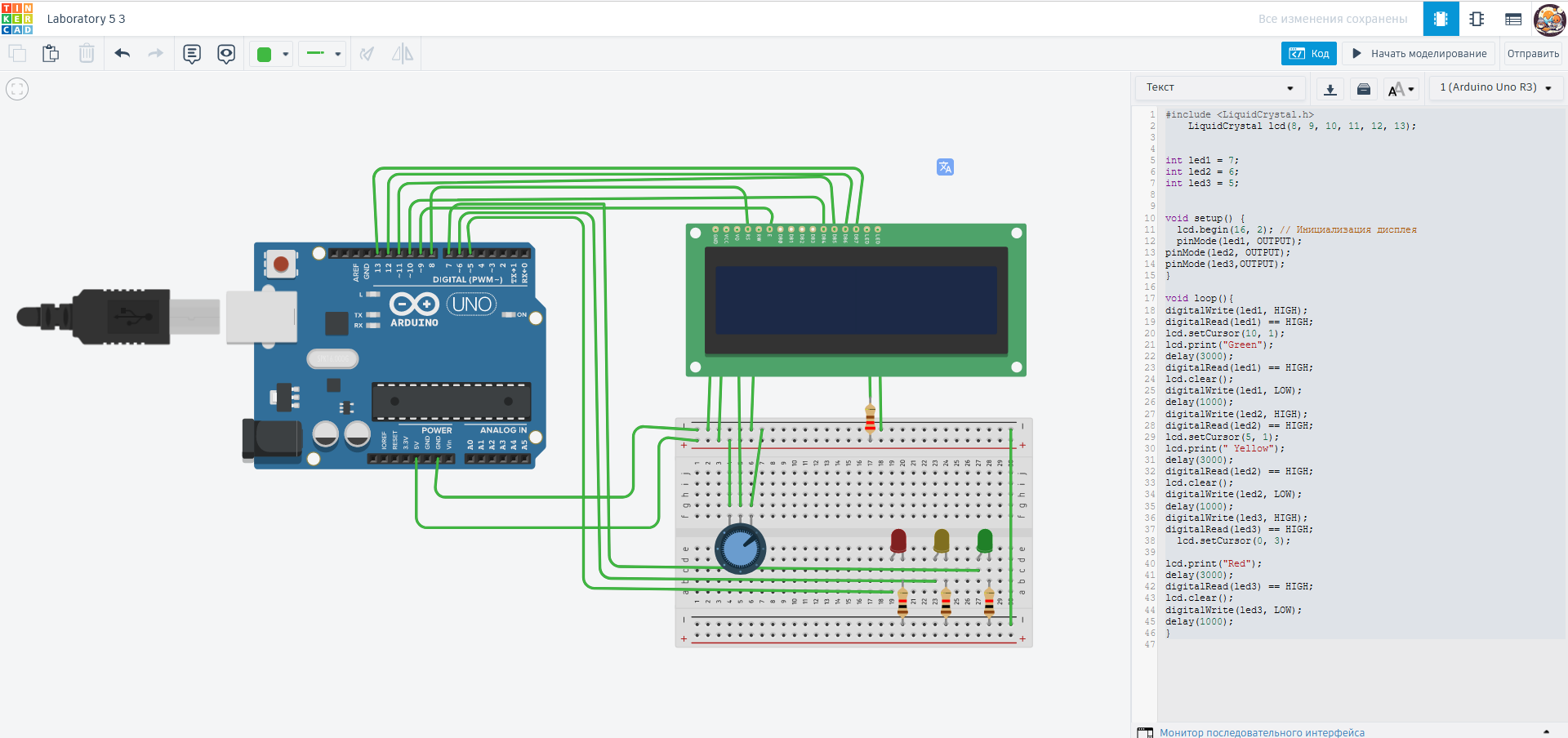


Рисунок 5.3 – Схема и код для третьего задания

Листинг программы

|  |
| --- |
| #include <LiquidCrystal.h>  LiquidCrystal **lcd**(**8**, **9**, **10**, **11**, **12**, **13**);  **int** led1 = **7**;  **int** led2 = **6**;  **int** led3 = **5**;  **void** **setup**() {  lcd.begin(**16**, **2**); // Инициализация дисплея  pinMode(led1, OUTPUT);  pinMode(led2, OUTPUT);  pinMode(led3,OUTPUT);  }  **void** **loop**(){  digitalWrite(led1, HIGH);  digitalRead(led1) == HIGH;  lcd.setCursor(**10**, **1**);  lcd.print("Green");  delay(**3000**);  digitalRead(led1) == HIGH;  lcd.clear();  digitalWrite(led1, LOW);  delay(**1000**);  digitalWrite(led2, HIGH);  digitalRead(led2) == HIGH;  lcd.setCursor(**5**, **1**);  lcd.print(" Yellow");  delay(**3000**);  digitalRead(led2) == HIGH;  lcd.clear();  digitalWrite(led2, LOW);  delay(**1000**);  digitalWrite(led3, HIGH);  digitalRead(led3) == HIGH;  lcd.setCursor(**0**, **3**);  lcd.print("Red");  delay(**3000**);  digitalRead(led3) == HIGH;  lcd.clear();  digitalWrite(led3, LOW);  delay(**1000**);  } |

# Лабораторная работа №6

**Тема**: изучение кода

**Цель:** ознакомиться с работой и применением кода на платах Arduino, научиться его использовать.

**Ход работы:**

Задание 1. Написать код, задающий работу железнодорожного светофора. На рисунке 6.1 представлена схема и код программы светофора, в котором присутствует дополнительный светофор – для пешеходов.

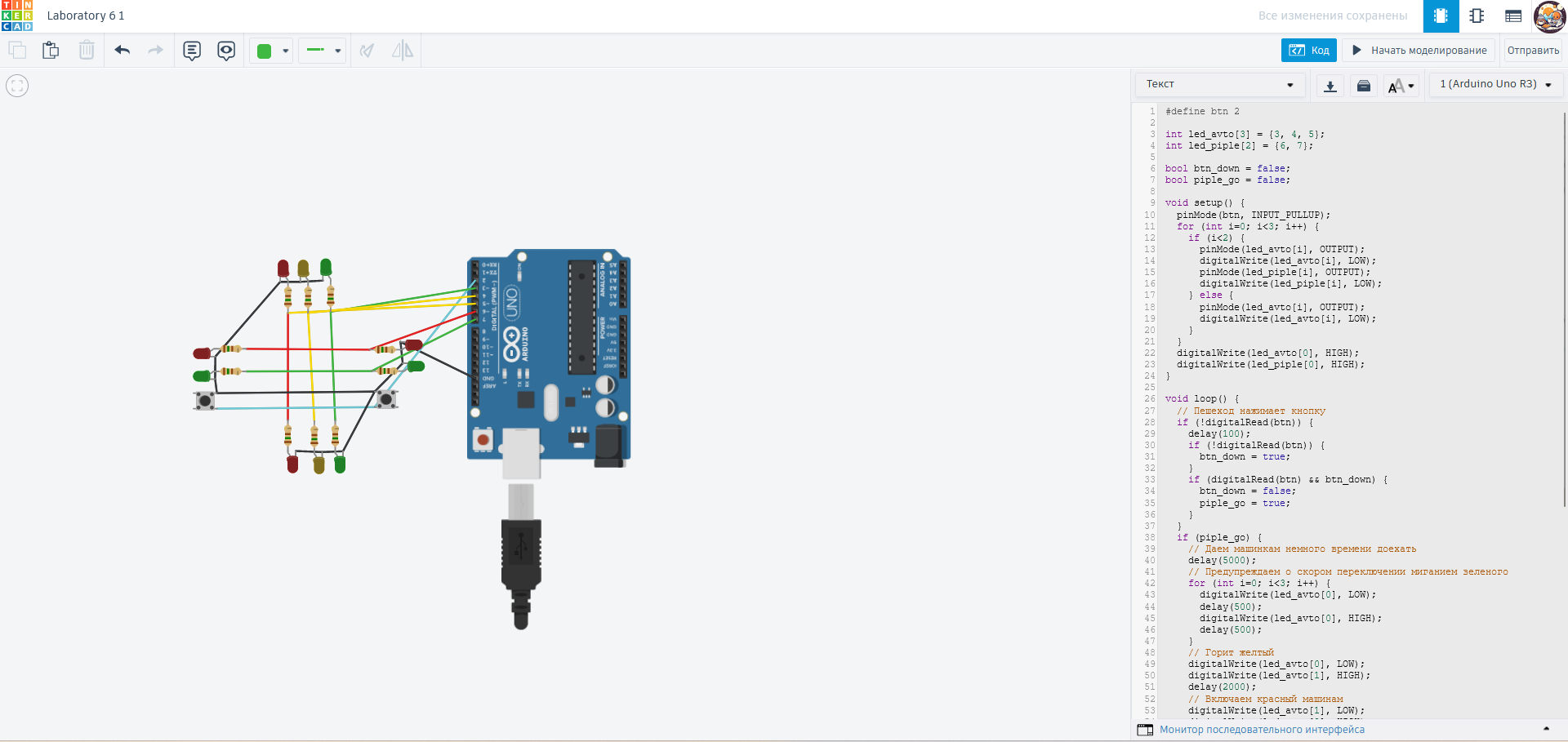


Рисунок 6.1 – Схема и код для первого задания

Листинг программы

|  |
| --- |
| #define btn 2  **int** led\_avto[**3**] = {**3**, **4**, **5**};  **int** led\_piple[**2**] = {**6**, **7**};  **bool** btn\_down = false;  **bool** piple\_go = false;  **void** **setup**() {  pinMode(btn, INPUT\_PULLUP);  **for** (**int** i=**0**; i<**3**; i++) {  **if** (i<**2**) {  pinMode(led\_avto[i], OUTPUT);  digitalWrite(led\_avto[i], LOW);  pinMode(led\_piple[i], OUTPUT);  digitalWrite(led\_piple[i], LOW);  } **else** {  pinMode(led\_avto[i], OUTPUT);  digitalWrite(led\_avto[i], LOW);  }  }  digitalWrite(led\_avto[**0**], HIGH);  digitalWrite(led\_piple[**0**], HIGH);  }  **void** **loop**() {  // Пешеход нажимает кнопку  **if** (!digitalRead(btn)) {  delay(**100**);  **if** (!digitalRead(btn)) {  btn\_down = true;  }  **if** (digitalRead(btn) && btn\_down) {  btn\_down = false;  piple\_go = true;  }  }  **if** (piple\_go) {  // Даем машинкам немного времени доехать  delay(**5000**);  // Предупреждаем о скором переключении миганием зеленого  **for** (**int** i=**0**; i<**3**; i++) {  digitalWrite(led\_avto[**0**], LOW);  delay(**500**);  digitalWrite(led\_avto[**0**], HIGH);  delay(**500**);  }  // Горит желтый  digitalWrite(led\_avto[**0**], LOW);  digitalWrite(led\_avto[**1**], HIGH);  delay(**2000**);  // Включаем красный машинам  digitalWrite(led\_avto[**1**], LOW);  digitalWrite(led\_avto[**2**], HIGH);  // Включаем зеленый людям  digitalWrite(led\_piple[**0**], LOW);  digitalWrite(led\_piple[**1**], HIGH);  delay(**4000**);  // Предупреждаем людей, что переход заканчивается  **for** (**int** i=**0**; i<**3**; i++) {  digitalWrite(led\_piple[**1**], LOW);  delay(**500**);  digitalWrite(led\_piple[**1**], HIGH);  delay(**500**);  }  digitalWrite(led\_piple[**1**], LOW);  digitalWrite(led\_piple[**0**], HIGH);  // Машины готовятся к старту  digitalWrite(led\_avto[**1**], HIGH);  delay(**1000**);  digitalWrite(led\_avto[**2**], LOW);  digitalWrite(led\_avto[**1**], LOW);  digitalWrite(led\_avto[**0**], HIGH);  piple\_go = false;  }  } |

Задание 2. Написать код для звукового воспроизведения восьми нот. (Тональности для пьезоэлемента(ов) подобрать по возможности близкими к оригиналу). На рисунке 6.2 представлена схема и код программы.

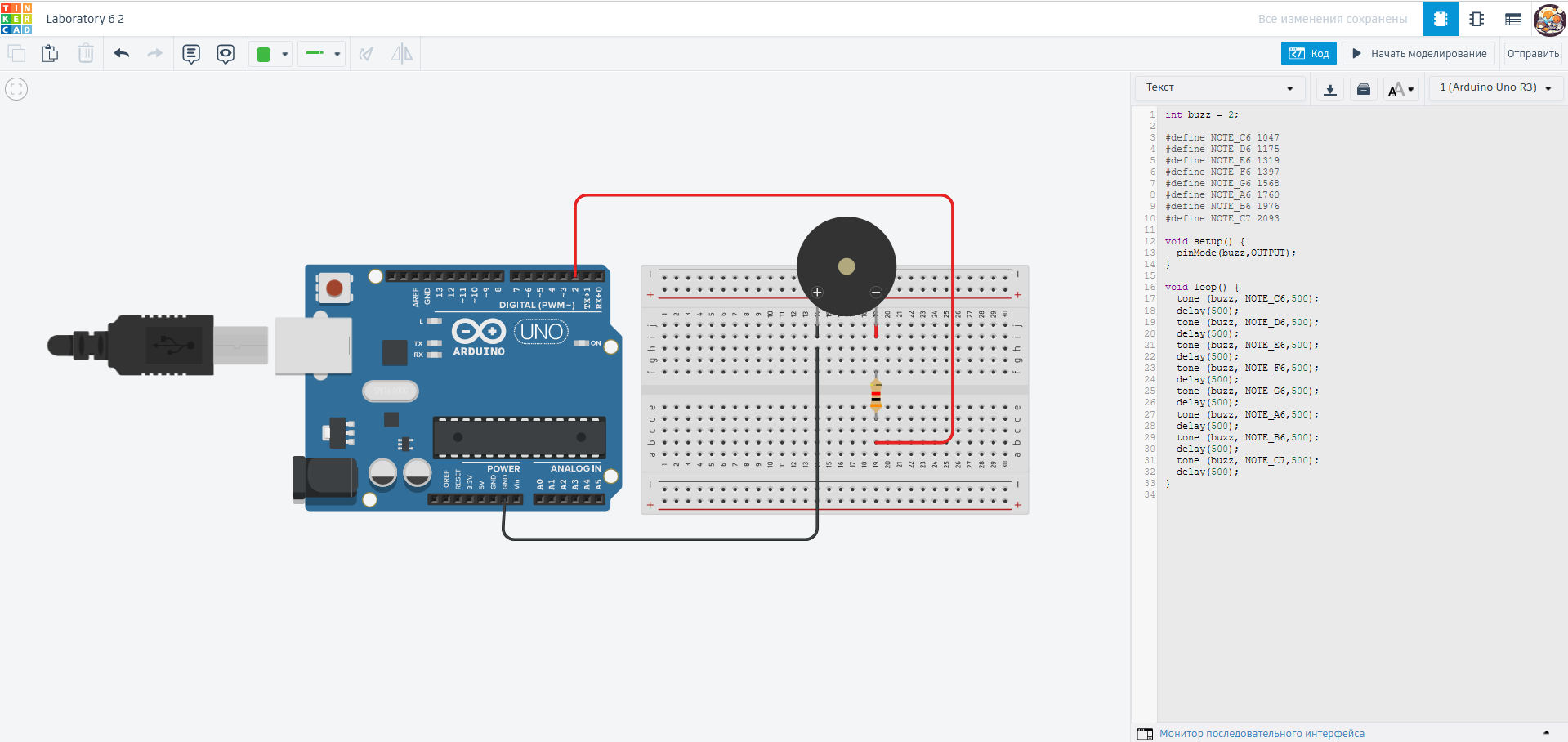


Рисунок 6.2 – Схема и код для второго задания

Листинг программы

|  |
| --- |
| **int** buzz = **2**;  #define NOTE\_C6 1047  #define NOTE\_D6 1175  #define NOTE\_E6 1319  #define NOTE\_F6 1397  #define NOTE\_G6 1568  #define NOTE\_A6 1760  #define NOTE\_B6 1976  #define NOTE\_C7 2093  **void** **setup**() {  pinMode(buzz,OUTPUT);  }  **void** **loop**() {  tone (buzz, NOTE\_C6,**500**);  delay(**500**);  tone (buzz, NOTE\_D6,**500**);  delay(**500**);  tone (buzz, NOTE\_E6,**500**);  delay(**500**);  tone (buzz, NOTE\_F6,**500**);  delay(**500**);  tone (buzz, NOTE\_G6,**500**);  delay(**500**);  tone (buzz, NOTE\_A6,**500**);  delay(**500**);  tone (buzz, NOTE\_B6,**500**);  delay(**500**);  tone (buzz, NOTE\_C7,**500**);  delay(**500**);  } |

Задание 3. Написать код для мигания светодиода «Led RGB» всеми цветами радуги подряд. На рисунке 6.3 представлена схема и блок программы.

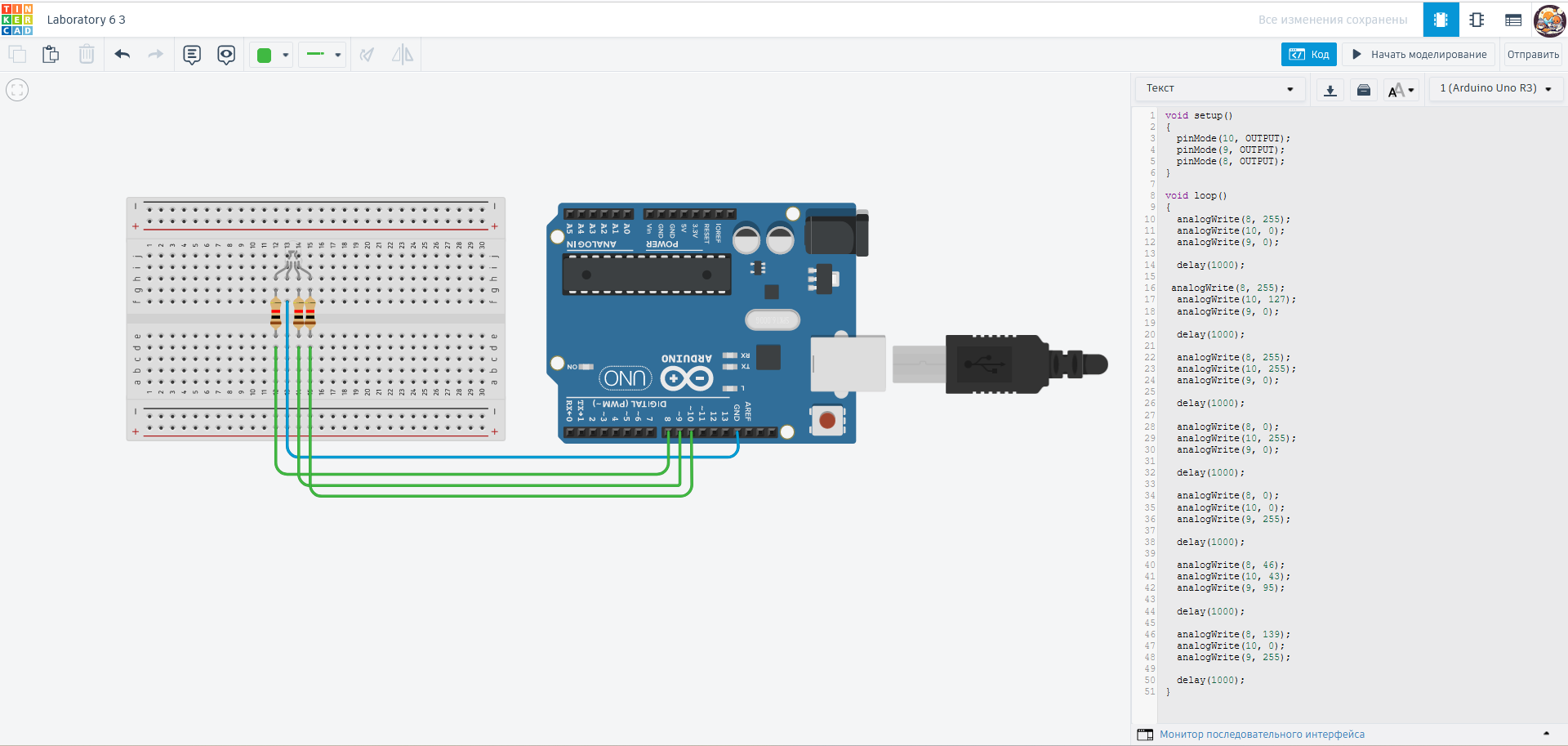


Рисунок 6.3 – Схема и код для третьего задания

Задание 4. Необходимо написать любой код, реализующий не элементарный алгоритм работы платы. На рисунке 6.4 представлена схема и код программы.

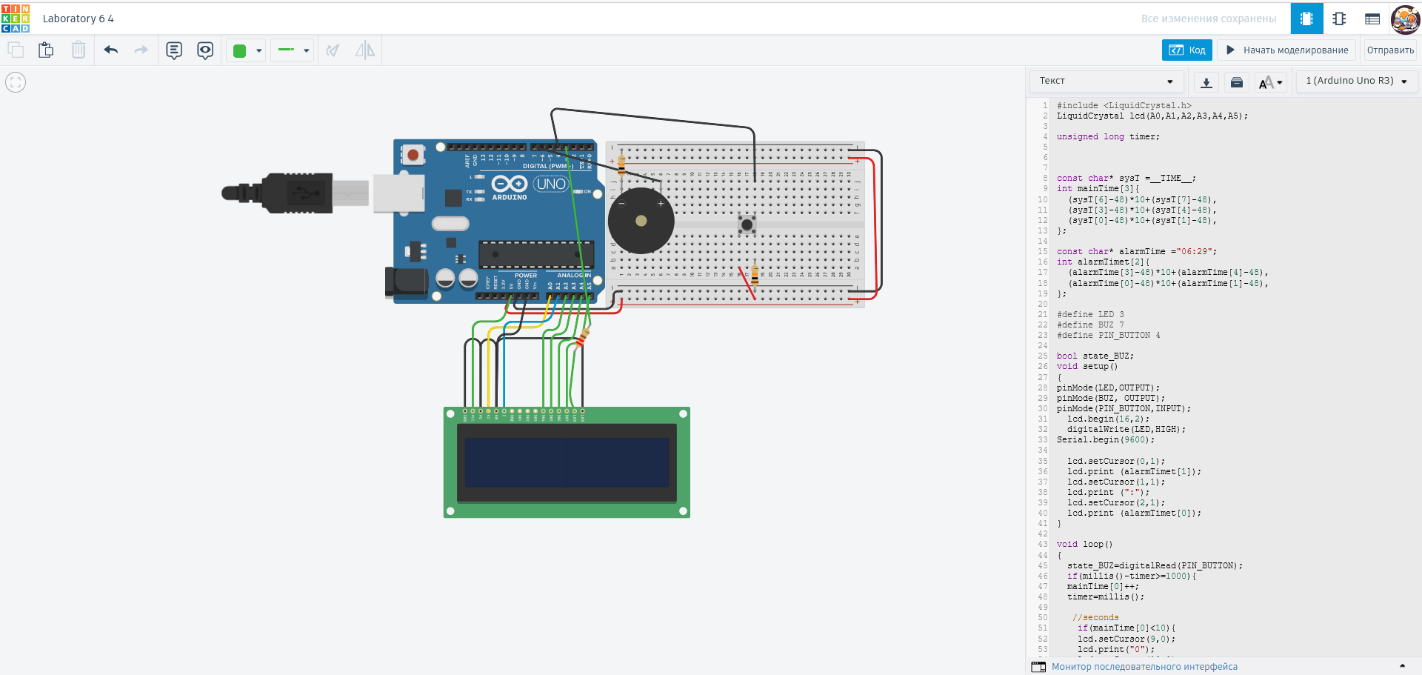


Рисунок 6.4 – Схема и код для четвертого задания

Листинг программы

|  |
| --- |
| #include <LiquidCrystal.h>  LiquidCrystal **lcd**(A0,A1,A2,A3,A4,A5);  **unsigned** **long** timer;  **const** **char**\* sysT =\_\_TIME\_\_;  **int** mainTime[**3**]{  (sysT[**6**]-**48**)\***10**+(sysT[**7**]-**48**),  (sysT[**3**]-**48**)\***10**+(sysT[**4**]-**48**),  (sysT[**0**]-**48**)\***10**+(sysT[**1**]-**48**),  };  **const** **char**\* alarmTime ="06:29";  **int** alarmTimet[**2**]{  (alarmTime[**3**]-**48**)\***10**+(alarmTime[**4**]-**48**),  (alarmTime[**0**]-**48**)\***10**+(alarmTime[**1**]-**48**),  };  #define LED 3  #define BUZ 7  #define PIN\_BUTTON 4  **bool** state\_BUZ;  **void** **setup**()  {  pinMode(LED,OUTPUT);  pinMode(BUZ, OUTPUT);  pinMode(PIN\_BUTTON,INPUT);  lcd.begin(**16**,**2**);  digitalWrite(LED,HIGH);  Serial.begin(**9600**);  lcd.setCursor(**0**,**1**);  lcd.print (alarmTimet[**1**]);  lcd.setCursor(**1**,**1**);  lcd.print (":");  lcd.setCursor(**2**,**1**);  lcd.print (alarmTimet[**0**]);  }  **void** **loop**()  {  state\_BUZ=digitalRead(PIN\_BUTTON);  **if**(millis()-timer>=**1000**){  mainTime[**0**]++;  timer=millis();  //seconds  **if**(mainTime[**0**]<**10**){  lcd.setCursor(**9**,**0**);  lcd.print("0");  lcd.setCursor(**10**,**0**);  lcd.print (mainTime[**0**]);  }  **if** (mainTime[**0**]>=**10**){  lcd.setCursor(**9**,**0**);  lcd.print (mainTime[**0**]);  }  **if** (mainTime[**0**]>=**60**){  mainTime[**0**]=**0**;  lcd.setCursor(**9**,**0**);  lcd.print ("00");  mainTime[**1**]++;  }  //minuts  **if** (mainTime[**1**]<**10**){  lcd.setCursor(**6**,**0**);  lcd.print("0");  lcd.setCursor(**7**,**0**);  lcd.print( mainTime[**1**]);  }  **if** (mainTime[**1**]>=**10**){  lcd.setCursor(**6**,**0**);  lcd.print(mainTime[**1**]);  }  **if** (mainTime[**1**]>=**60**){  mainTime[**1**]=**0**;  lcd.setCursor(**6**,**0**);  lcd.print ("00");  mainTime[**2**]++;  }  //hours  **if** (mainTime[**2**]<**10**){  lcd.setCursor(**3**,**0**);  lcd.print("0");  lcd.setCursor(**4**,**0**);  lcd.print( mainTime[**2**]);  }  **if** (mainTime[**2**]>**10**){  lcd.setCursor(**3**,**0**);  lcd.print (mainTime[**2**]);  }  **if**(mainTime[**2**]>=**24**){  mainTime[**2**]=**0**;  lcd.setCursor(**3**,**0**);  lcd.print ("00");  }  //dots  lcd.setCursor(**5**,**0**);  lcd.print(":");  lcd.setCursor(**8**,**0**);  lcd.print(":");  **if**(alarmTime== alarmTime && mainTime[**0**]==**0**){  tone(BUZ,**1000**,**500**);  }  **if**(state\_BUZ==**1**){  noTone(BUZ);  }  }  delay(**1**);  } |

Задание 5. *Написать код, задающий мерцание новогодней гирлянды с помощью кнопок. Рекомендуется использовать столько кнопок, сколько режимов мерцания подразумевается в Вашем алгоритме.* На рисунке 6.5 представлены схема и код программы

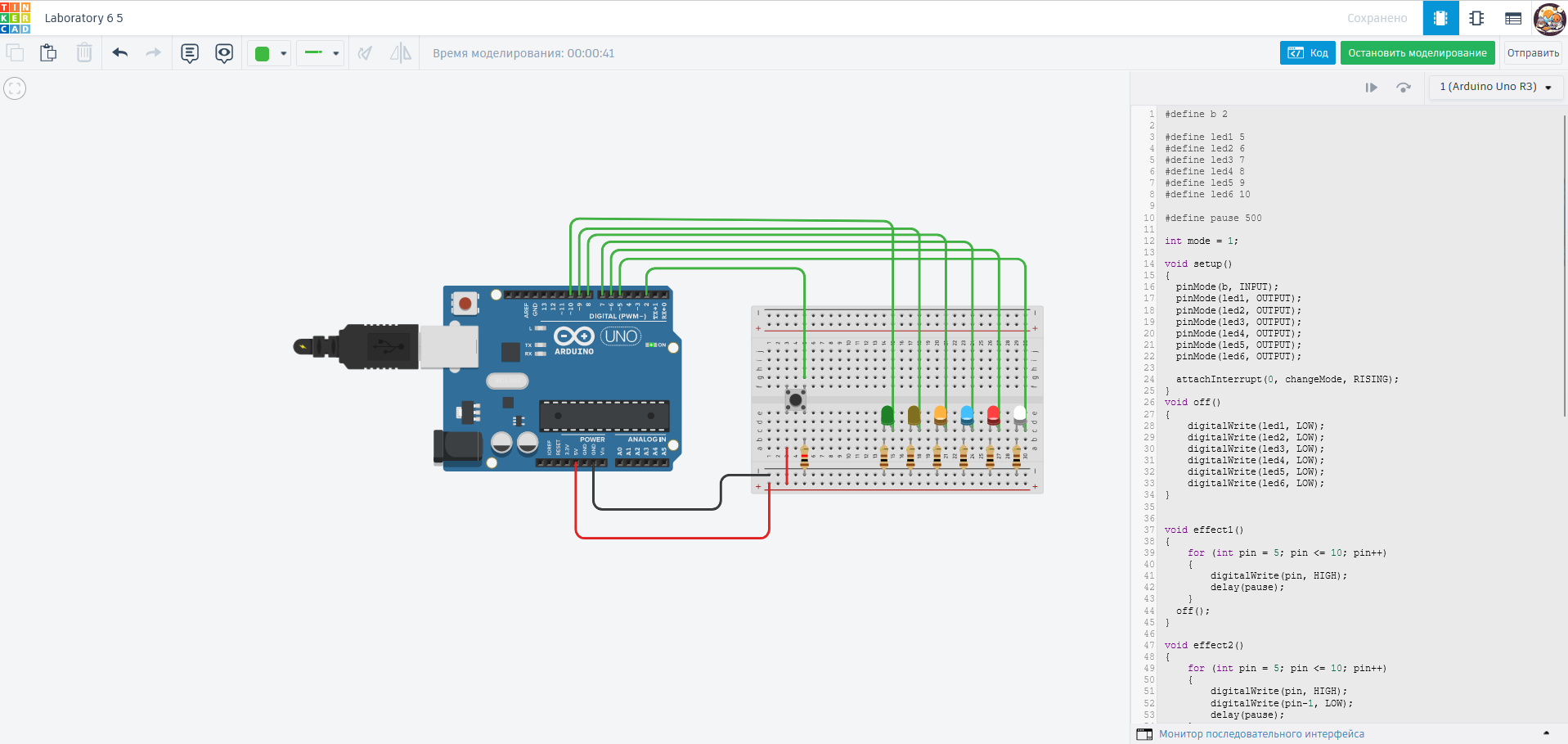


Рисунок 6.5 – Схема и код для пятого задания

Листинг программы

|  |
| --- |
| #define b 2  #define led1 5  #define led2 6  #define led3 7  #define led4 8  #define led5 9  #define led6 10  #define pause 500  **int** mode = **1**;  **void** **setup**()  {  pinMode(b, INPUT);  pinMode(led1, OUTPUT);  pinMode(led2, OUTPUT);  pinMode(led3, OUTPUT);  pinMode(led4, OUTPUT);  pinMode(led5, OUTPUT);  pinMode(led6, OUTPUT);  attachInterrupt(**0**, changeMode, RISING);  }  **void** **off**()  {  digitalWrite(led1, LOW);  digitalWrite(led2, LOW);  digitalWrite(led3, LOW);  digitalWrite(led4, LOW);  digitalWrite(led5, LOW);  digitalWrite(led6, LOW);  }  **void** **effect1**()  {  **for** (**int** pin = **5**; pin <= **10**; pin++)  {  digitalWrite(pin, HIGH);  delay(pause);  }  off();  }  **void** **effect2**()  {  **for** (**int** pin = **5**; pin <= **10**; pin++)  {  digitalWrite(pin, HIGH);  digitalWrite(pin-**1**, LOW);  delay(pause);  }  off();  }  **void** **effect3**()  {  **for** (**int** pin = **5**; pin <= **10**; pin++)  {  digitalWrite(pin, HIGH);  }  delay(pause);  off();  delay(pause);  }  **void** **effect4**()  {  **for** (**int** pin = **5**; pin <= **10**; pin++)  {  digitalWrite(pin, LOW);  }  delay(pause);  off();  delay(pause);  }  **void** **loop**()  {  **if** (mode == **1**)  {  effect1();  }  **else** **if** (mode == **2**)  {  effect2();  }  **else** **if** (mode == **3**)  {  effect3();  }  **else** **if** (mode == **4**)  {  effect4();  }  }  **void** **changeMode**()  {  mode++;  **if** (mode > **4**) mode = **1**;  } |

# Лабораторная работа №7

**Тема**: самостоятельное кодирование платы

**Цель:** ознакомиться с работой и применением кода на платах Arduino, научиться его использовать, а также изучить температурный датчик.

**Ход работы:**

Задание 1. запрограммировать плату так, чтобы начальная температура была 40 градусов (при этом не горит ни один светодиод), потом при поднятии температуры на каждые 10 градусов зажигался один светодиод.

(50 град.- один, 60 град.- два, 70 град.- все три светодиода). На рисунке 7.1 представлена схема и код программы.

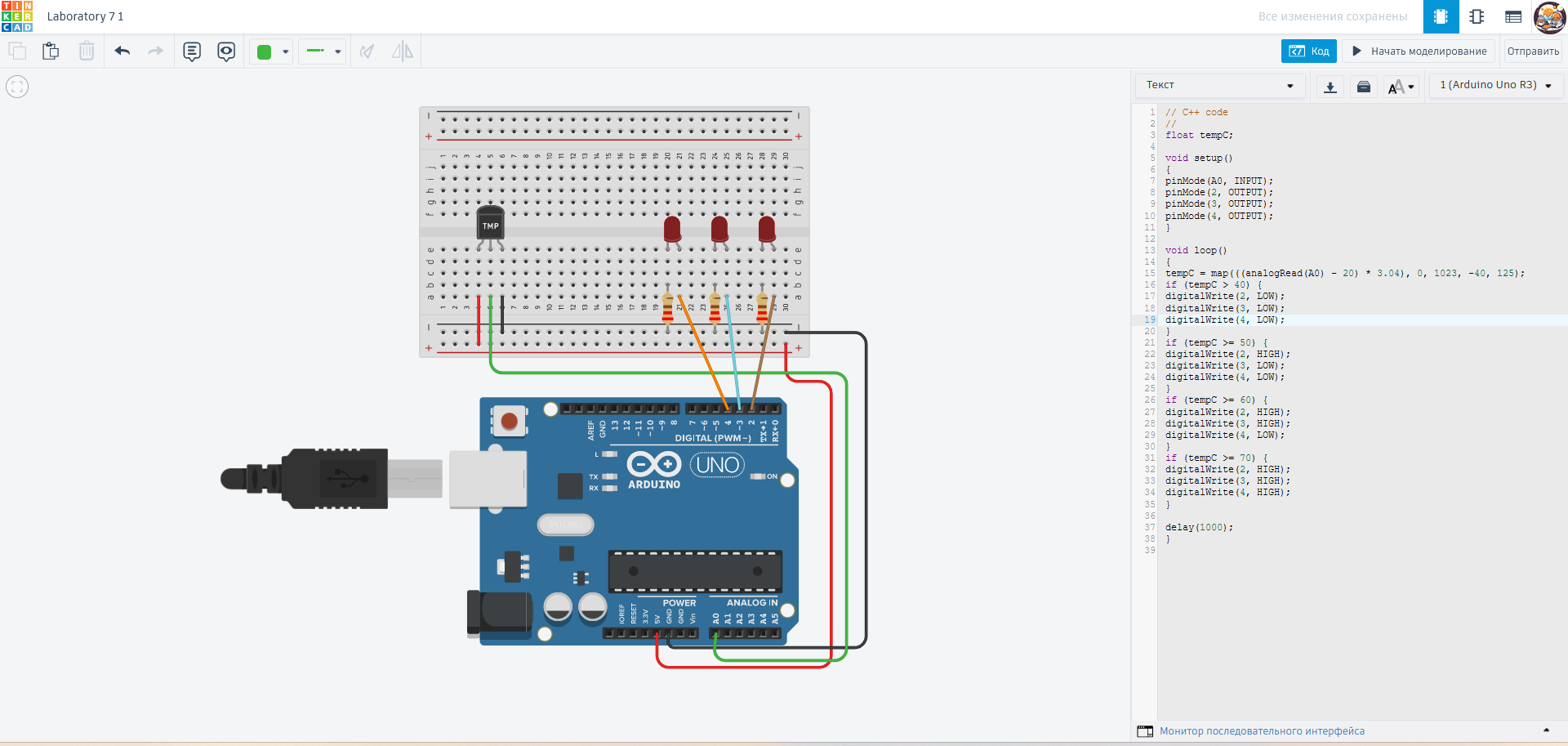


Рисунок 7.1 – Схема и код для задания

Листинг программы

|  |
| --- |
| // C++ code  //  **float** tempC;  **void** **setup**()  {  pinMode(A0, INPUT);  pinMode(**2**, OUTPUT);  pinMode(**3**, OUTPUT);  pinMode(**4**, OUTPUT);  }  **void** **loop**()  {  tempC = map(((analogRead(A0) - **20**) \* **3.04**), **0**, **1023**, -**40**, **125**);  **if** (tempC > **40**) {  digitalWrite(**2**, LOW);  digitalWrite(**3**, LOW);  digitalWrite(**4**, LOW);  }  **if** (tempC >= **50**) {  digitalWrite(**2**, HIGH);  digitalWrite(**3**, LOW);  digitalWrite(**4**, LOW);  }  **if** (tempC >= **60**) {  digitalWrite(**2**, HIGH);  digitalWrite(**3**, HIGH);  digitalWrite(**4**, LOW);  }  **if** (tempC >= **70**) {  digitalWrite(**2**, HIGH);  digitalWrite(**3**, HIGH);  digitalWrite(**4**, HIGH);  }  delay(**1000**);  } |

ASSEMBLER

Лабораторная 1

|  |
| --- |
| ; 11. X = (2\*A + B) / 4 - C / 2 + 168  data segment  a dw 10  b dw 20  c dw 8  x dw ?  data ends  code segment  assume cs: code, ds: data  start:  mov ax, data  mov ds, ax ; load addresses  mov ax, a ; ax = a  sal ax, 1 ; ax = a \* 2  add ax, b ; ax = a \* 2 + b  sar ax, 2 ; ax = (a \* 2 + b) / 4  mov bx, c ; bx = c  sar bx, 1 ; bx = c / 2  sub ax, bx ; ax = (a \* 2 + b) / 4 - c / 2  add ax, 168 ; ax = (a \* 2 + b) / 4 - c / 2 + 168  mov x, ax ; record result into x  quit:  mov ax, 4c00h ; end code 0  int 21h ; exit into dos  end start  code ends |

Лабораторная 2

|  |
| --- |
| data segment  max dw ?  mass dw 1,2,3,4,-10,6,-1,7,1,9  data ends  code segment  assume cs: code, ds: data  start:  mov ax, data ; загружаем адрес сегмента данных в регистр ax  mov ds, ax ; установка сегмента данных равного адресу из ax  lea bx, mass ; загрузка адреса начала массива чисел в регистр bx  mov cx, 10 ; инициализация счетчика цикла количеством элементов в массиве 10  mov ax, [bx] ; загрузка первого числа из массива в регистр  xor dl,dl ; обнуление регистра dl, который будет использоваться для подсчета отрицательных чисел  beg:  cmp byte [bx], 0 ; сравнение текущего элемента массива с нулем  jnl no ; если текущий элемент не отрицателньый переходим к метке no  inc dl ; увеличение счетчика отрицательных чисел  cmp [bx], ax ; сравнение текущего элемента с максимально найднем отриц числов (хранится в ax)  jge no ; если текущий элемент больше или равен максимальному найденом, то идем к метке no  mov ax, [bx] ; перенос значения в ax их bx  no:  add bx, 2 ; переходим к следующему элементу массива  loop beg ; выполнение цикла снова  cmp dl, 0 ; сравнение счетчика отрицательных чисел с нулем  jz quit ; если отриц. чисел не было найдено - завершение программы  mov max, ax ; сохранение значения в max  quit:  mov ax,4C00h  int 21h  code ends  end start |