МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

ОТЧЕТ

по лабораторной работе №2

«Программирование микроконтроллера Ардуино»

по дисциплине

«Информационно-управляющие вычислительные системы»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Мартынов Д. С. \_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТЫ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В. А\_\_\_

(подпись) (фамилия, и., о)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мосташов В. С\_\_\_

(подпись) (фамилия, и., о)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Рябов Д. А\_\_\_\_

(подпись) (фамилия, и., о)

\_\_\_\_\_\_\_\_\_\_\_\_\_ Расторопов Д. С\_\_\_

(подпись) (фамилия, и., о)

\_\_\_\_\_\_19-В-2\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2022

Оглавление

[Ход работы 3](#_Toc103862585)

[Эксперимент 9. Миксер 3](#_Toc103862586)

[Эксперимент 10. Кнопочный переключатель 5](#_Toc103862587)

[Эксперимент 11. Светильник с кнопочным управлением 6](#_Toc103862588)

[Эксперимент 12. Кнопочные ковбои 8](#_Toc103862589)

[Эксперимент 13. Секундомер 10](#_Toc103862590)

[Эксперимент 18. Тестер батареек 12](#_Toc103862591)

[Эксперимент 20. Перетягивание каната 14](#_Toc103862592)

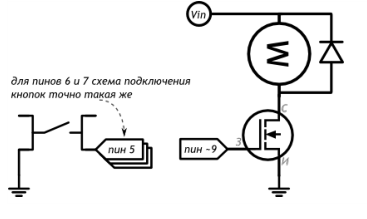
# Ход работы

## Эксперимент 9. Миксер

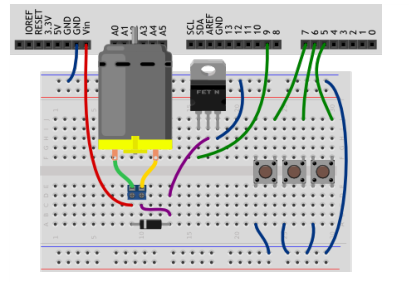
### Список деталей для эксперимента

* 1 плата Arduino Uno
* беспаечная макетная плата
* 3 тактовых кнопки
* 1 коллекторный двигатель
* 1 выпрямительный диод
* 1 полевой MOSFET-транзистор
* 15 проводов «папа-папа»

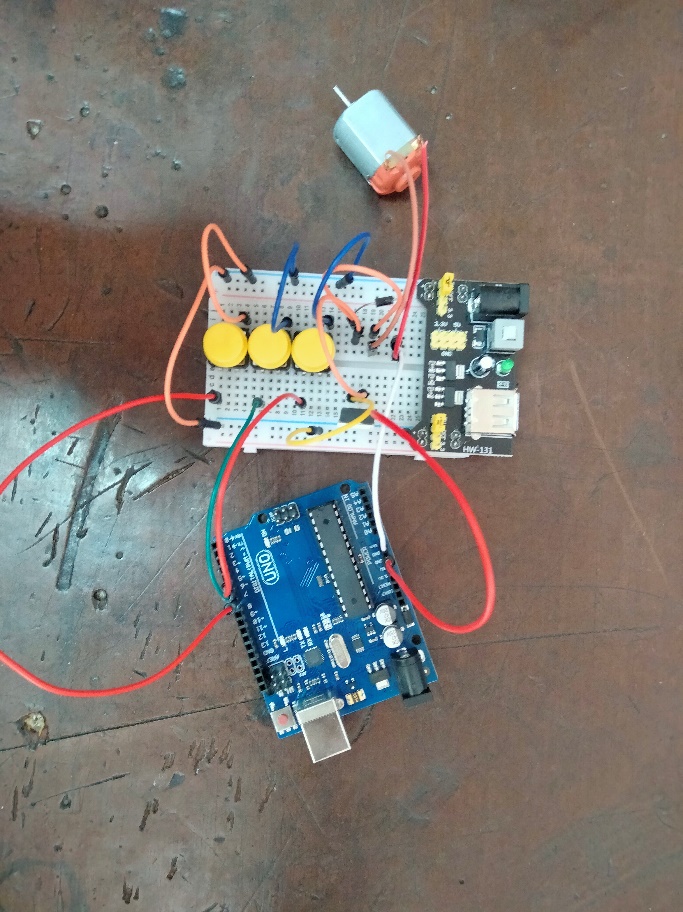
### Принципиальная схема



### Схема на макете



### Фото собранной схемы



### Код программы

#define MOTOR\_PIN 9

#define FIRST\_BUTTON\_PIN 5

#define BUTTON\_COUNT 3

// имена можно давать не только числам, но и целым выражениям.

// Мы определяем с каким шагом (англ. step) нужно менять

// скорость (англ. speed) мотора при нажатии очередной кнопки

#define SPEED\_STEP (255 / (BUTTON\_COUNT - 1))

void setup(){

pinMode(MOTOR\_PIN, OUTPUT);

// на самом деле, в каждом пине уже есть подтягивающий

// резистор. Для его включения необходимо явно настроить пин

// как вход с подтяжкой (англ. input with pull up)

for (int i = 0; i < BUTTON\_COUNT; ++i)

pinMode(i + FIRST\_BUTTON\_PIN, INPUT\_PULLUP);

}

void loop(){

for (int i = 0; i < BUTTON\_COUNT; ++i) {

//если кнопка отпущена, нам она не интересна. //Пропускаем оставшуюся часть цикла for, продолжая

//(англ. continue)его дальше, для следующего значения i

if (digitalRead(i + FIRST\_BUTTON\_PIN))

continue;

//кнопка нажата — выставляем соответствующую ей //скорость мотора. Нулевая кнопка остановит вращение, //первая заставит крутиться в пол силы, вторая — на //полную

int speed = i \* SPEED\_STEP;

// подача ШИМ-сигнала на мотор заставит его крутиться с

// указанной скоростью: 0 — стоп машина, 127 — полсилы,

// 255 — полный вперёд!

analogWrite(MOTOR\_PIN, speed);

}

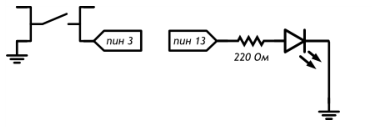
}

## Эксперимент 10. Кнопочный переключатель

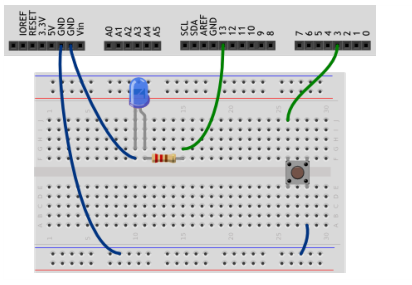
### Список деталей для эксперимента

* 1 плата Arduino Uno
* 1 беспаечная макетная плата
* 1 тактовая кнопка
* 1 резистор номиналом 220 Ом
* 1 светодиод
* 5 проводов «папа-папа»

### Принципиальная схема



### Схема на макете



### Код программы

#define BUTTON\_PIN 3

#define LED\_PIN 13

boolean buttonWasUp = true; // была ли кнопка отпущена?

boolean ledEnabled = false; // включен ли свет?

void setup(){

pinMode(LED\_PIN, OUTPUT);

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

}

void loop(){

// определить момент «клика» несколько сложнее, чем факт //того, что кнопка сейчас просто нажата. Для определения //клика мы сначала понимаем, отпущена ли кнопка прямо //сейчас...

boolean buttonIsUp = digitalRead(BUTTON\_PIN);

// если «кнопка была отпущена и (&&) не отпущена сейчас»

if (buttonWasUp && !buttonIsUp) {

// ...может это «клик», а может и ложный сигнал (дребезг),

// возникающий в момент замыкания/размыкания пластин кнопки,

// поэтому даём кнопке полностью «успокоиться»...

delay(10);

// ...и считываем сигнал снова

buttonIsUp = digitalRead(BUTTON\_PIN);

if (!buttonIsUp) {

// если она всё ещё нажата...

// ...это клик! Переворачиваем сигнал светодиода

ledEnabled = !ledEnabled;

digitalWrite(LED\_PIN, ledEnabled);

}

}

// запоминаем последнее состояние кнопки для новой итерации

buttonWasUp = buttonIsUp;

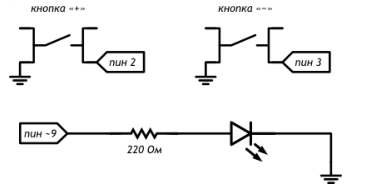
}

## Эксперимент 11. Светильник с кнопочным управлением

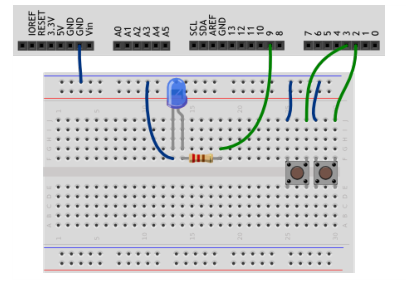
### Список деталей для эксперимента

* 1 плата Arduino Uno
* 1 беспаечная макетная плата
* 2 тактовых кнопки
* 1 резистор номиналом 220 Ом
* 1 светодиод
* 7 проводов «папа-папа»

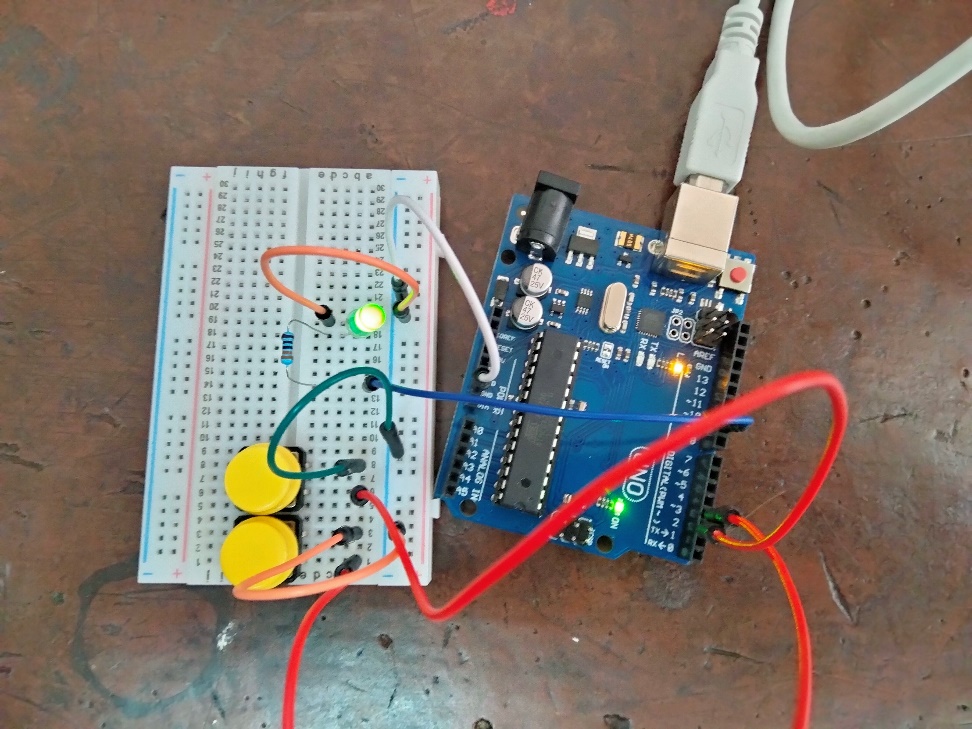
### Принципиальная схема



### Схема на макете



### Фото собранной схемы



### Код программы

#define PLUS\_BUTTON\_PIN 2

#define MINUS\_BUTTON\_PIN 3

#define LED\_PIN 9

int brightness = 100;

boolean plusUp = true;

boolean minusUp = true;

void setup(){

pinMode(LED\_PIN, OUTPUT);

pinMode(PLUS\_BUTTON\_PIN, INPUT\_PULLUP);

pinMode(MINUS\_BUTTON\_PIN, INPUT\_PULLUP);

}

void loop(){

analogWrite(LED\_PIN, brightness);

// реагируем на нажатия с помощью функции, написанной нами

plusUp = handleClick(PLUS\_BUTTON\_PIN, plusUp, +35);

minusUp = handleClick(MINUS\_BUTTON\_PIN, minusUp, -35);

}

// Собственная функция с 3 параметрами: номером пина с кнопкой

// (buttonPin), состоянием до проверки (wasUp) и градацией

// яркости при клике на кнопку (delta). Функция возвращает

// (англ. return) обратно новое, текущее состояние кнопки

boolean handleClick(int buttonPin, boolean wasUp, int delta){

boolean isUp = digitalRead(buttonPin);

if (wasUp && !isUp) {

delay(10);

isUp = digitalRead(buttonPin);

// если был клик, меняем яркость

//в пределах от 0 до 255

if (!isUp)

brightness = constrain(brightness + delta, 0, 255);

}

// возвращаем значение обратно, в вызывающий код

return isUp;

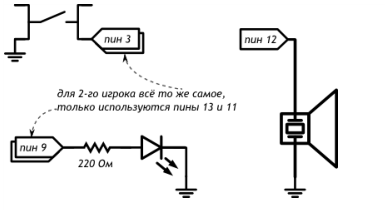
}

## Эксперимент 12. Кнопочные ковбои

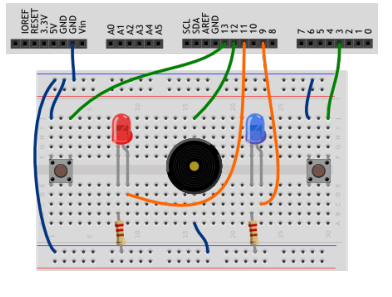
### Список деталей для эксперимента

* 1 плата Arduino Uno
* 1 беспаечная макетная плата
* 2 тактовых кнопки
* 2 резистора номиналом 220 Ом
* 2 светодиода
* 1 пьезопищалка
* 10 проводов «папа-папа»

### Принципиальная схема



### Схема на макете



### Код программы

#define BUZZER\_PIN 12 // пин с пищалкой

#define PLAYER\_COUNT 2 // количество игроков-ковбоев

// вместо перечисления всех пинов по-одному, мы объявляем пару

// списков: один с номерами пинов с кнопками, другой — со

// светодиодами.

int buttonPins[PLAYER\_COUNT] = {3, 13};

int ledPins[PLAYER\_COUNT] = {9, 11};

void setup(){

pinMode(BUZZER\_PIN, OUTPUT);

for (int player = 0; player < PLAYER\_COUNT; ++player) {

// при помощи квадратных скобок получают значение в //массиве под указанным в них номером. pinMode(ledPins[player], OUTPUT);

pinMode(buttonPins[player], INPUT\_PULLUP);

}

}

void loop(){

// даём сигнал «пли!», выждав случайное время от 2 до 7 сек

delay(random(2000, 7000));

// 3 килогерца, 250 миллисекунд

tone(BUZZER\_PIN, 3000, 250);

for (int player = 0; ; player = (player+1) % PLAYER\_COUNT) {

// если игрок номер «player» нажал кнопку

if (!digitalRead(buttonPins[player])) {

// включаем его светодиод и сигнал победы на 1 сек

digitalWrite(ledPins[player], HIGH);

tone(BUZZER\_PIN, 4000, 1000);

delay(1000);

digitalWrite(ledPins[player], LOW);

// Есть победитель! Выходим (англ. break) из цикла

break;

}

}

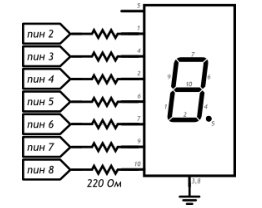
}

## Эксперимент 13. Секундомер

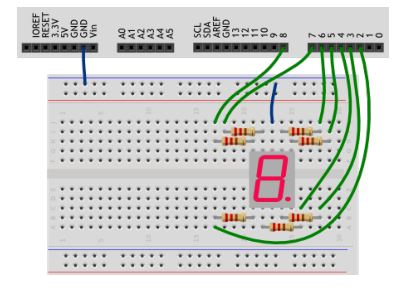
### Список деталей для эксперимента

* 1 плата Arduino Uno
* 1 беспаечная макетная плата
* 1 семи сегментный индикатор
* 7 резисторов номиналом 220 Ом
* 9 проводов «папа-папа»

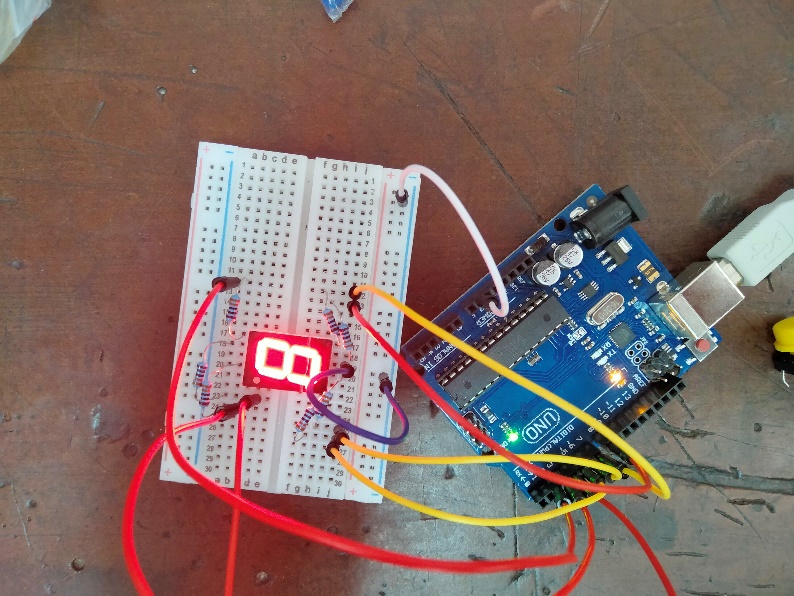
### Принципиальная схема



### Схема на макете



### Фото собранной схемы



### Код программы

#define FIRST\_SEGMENT\_PIN 2

#define SEGMENT\_COUNT 7

// префикс «0b» означает, что целое число за ним записано в

// в двоичном коде. Единицами мы обозначим номера сегментов

// индикатора, которые должны быть включены для отображения

// арабской цифры. Всего цифр 10, поэтому в массиве 10 чисел.

// Нам достаточно всего байта (англ. byte, 8 бит) для хранения

// комбинации сегментов для каждой из цифр.

byte numberSegments[10] = {

0b00111111, 0b00001010, 0b01011101, 0b01011110, 0b01101010,

0b01110110, 0b01110111, 0b00011010, 0b01111111, 0b01111110,

};

void setup(){

for (int i = 0; i < SEGMENT\_COUNT; ++i)

pinMode(i + FIRST\_SEGMENT\_PIN, OUTPUT);

}

void loop(){

// определяем число, которое собираемся отображать. Пусть им

// будет номер текущей секунды, зацикленный на десятке

int number = (millis() / 1000) % 10;

// получаем код, в котором зашифрована арабская цифра

int mask = numberSegments[number];

// для каждого из 7 сегментов индикатора

for (int i = 0; i < SEGMENT\_COUNT; ++i) {

//определяем: должен ли он быть включён. Для этого

//считываем бит , соответствующий текущему сегменту «i». //Истина — он установлен (1), ложь — нет (0)

boolean enableSegment = bitRead(mask, i);

// включаем/выключаем сегмент

//на основе полученного значения

digitalWrite(i + FIRST\_SEGMENT\_PIN, enableSegment);

}

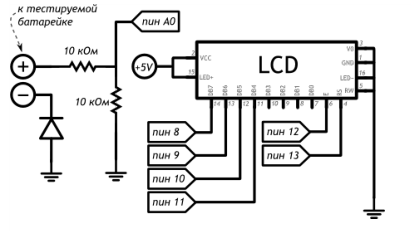
}

## Эксперимент 18. Тестер батареек

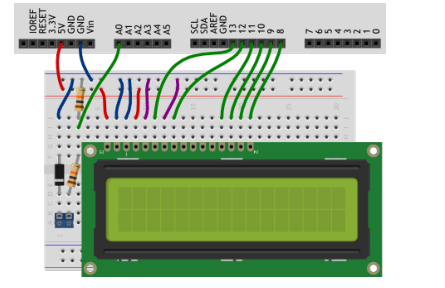
### Список деталей для эксперимента

* 1 плата Arduino Uno
* 1 беспаечная макетная плата
* 2 резистора номиналом 10 кОм
* 1 выпрямительный диод
* 1 текстовый экран
* 16 проводов «папа-папа»

### Принципиальная схема



### Схема на макете



### Код программы

// Подключаем библиотеку для работы с жидкокристаллическим

// экраном

#include <LiquidCrystal.h>

// на диоде, защищающем от неверной полярности, падает доля

// напряжения. Необходимо это учитывать

#define DIODE\_DROP 0.7

// Объявляем объект, для управления дисплеем. Для его создания

// необходимо указать номера пинов, к которым он подключен в

// порядке: RS E DB5 DB6 DB7 DB8

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

void setup(){

// начинаем работу с экраном. Сообщаем объекту количество

// строк и столбцов.

lcd.begin(16, 2);

// печатаем сообщение на первой строке

lcd.print("Battery voltage:");

}

void loop(){

// высчитываем напряжение подключенной батарейки

float voltage = analogRead(A0) / 1024.0 \* 10.0;

// если напряжение на делителе напряжения было зафиксировано,

// нужно прибавить напряжение на диоде, т.к. оно было съедено

if (voltage > 0.1)

voltage += DIODE\_DROP;

// устанавливаем курсор, колонку 0, строку 1. На деле — это

// левый квадрат 2-й строки, т.к. нумерация начинается с нуля

lcd.setCursor(0, 1);

// печатаем напряжение в батарейке с точностью до сотых долей

lcd.print(voltage, 2);

// следом печатаем единицы измерения

lcd.print(" Volts");

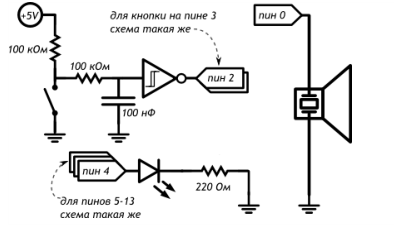
}

## Эксперимент 20. Перетягивание каната

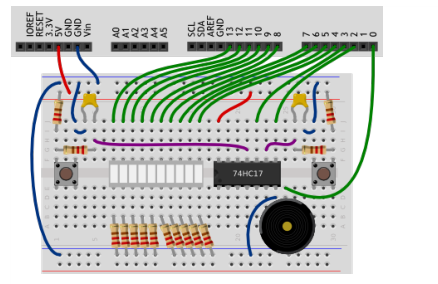
### Список деталей для эксперимента

* 1 плата Arduino Uno
* 1 беспаечная макетная плата
* 1 светодиодная шкала
* 10 резисторов номиналом 220 Ом
* 4 резисторов номиналом 100 кОм
* 2 тактовых кнопки
* 2 керамических конденсатора номиналом 100 нФ
* 1 пьезопищалка
* 1 инвертирующий триггер Шмитта
* 24 провода «папа-папа»

### Принципиальная схема



### Схема на макете



### Код программы

#define BUZZER\_PIN 0

#define FIRST\_BAR\_PIN 4

#define BAR\_COUNT 10

#define MAX\_SCORE 20

// глобальные переменные, используемые в прерываниях

// должны быть отмечены как нестабильные

volatile int score = 0;

void setup(){

for (int i = 0; i < BAR\_COUNT; ++i)

pinMode(i + FIRST\_BAR\_PIN, OUTPUT);

pinMode(BUZZER\_PIN, OUTPUT);

// Прерывание приостанавливает основную программу, выполняет //заданную функцию, а затем возобновляет основную программу. //Нам нужно прерывание на нажатие кнопки, т.е. при смене //сигнала с высокого на низкий, т.е. на нисходящем фронте

attachInterrupt(INT1, pushP1, FALLING); // INT1 — это 3-й пин

attachInterrupt(INT0, pushP2, FALLING); // INT0 — это 2-й пин

}

void pushP1() { ++score; } // функция-прерывание 1-го игрока

void pushP2() { --score; } // функция-прерывание 2-го игрока

void loop(){

tone(BUZZER\_PIN, 2000, 1000); // даём сигнал к старту.

// пока никто из игроков не выиграл, обновляем «канат»

while (abs(score) < MAX\_SCORE) {

int bound = map(score, -MAX\_SCORE, MAX\_SCORE, 0, BAR\_COUNT);

int left = min(bound, BAR\_COUNT / 2 - 1);

int right = max(bound, BAR\_COUNT / 2);

for (int i = 0; i < BAR\_COUNT; ++i)

digitalWrite(i + FIRST\_BAR\_PIN, i >= left && i <= right);

}

tone(BUZZER\_PIN, 4000, 1000); // даём сигнал победы

while (true) {} // «подвешиваем» плату до перезагрузки

}