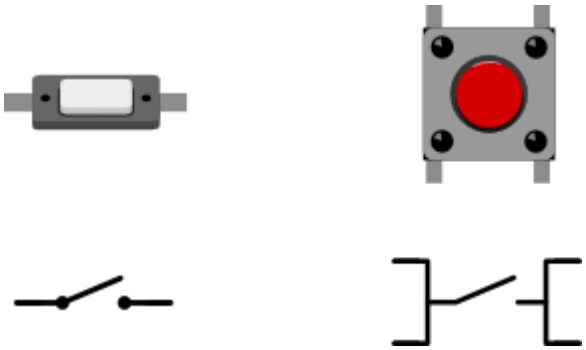
Практическая работа 8

Мерзкое пианино

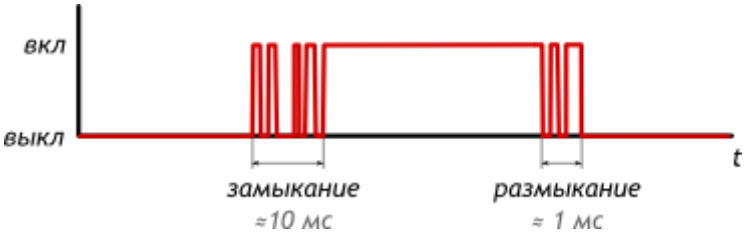
В этом эксперименте мы создаем маленькую клавиатуру, на которой можно сыграть несколько нот.

Тактовая кнопка — простой, всем известный механизм, замыкающий цепь пока есть давление на толкатель.



Кнопки с 4 контактами стоит рассматривать, как 2 пары рельс, которые соединяются при нажатии.

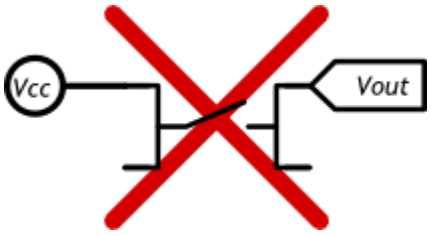
Эффект дребезга



При замыкании и размыкании между пластинами кнопки возникают микроискры, провоцирующие до десятка переключений за несколько миллисекунд. Явление называется *дребезгом* (англ. bounce). Это нужно учитывать, если необходимо фиксировать «клики».

Схема подключения

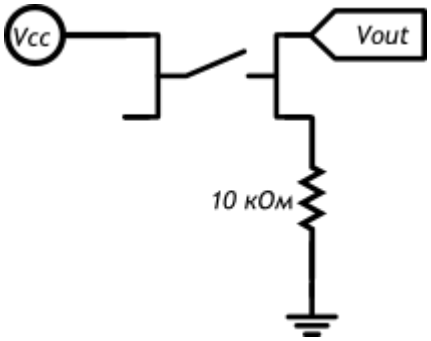
Напрашивается подключение напрямую. Но это наивный, неверный способ.



Пока кнопка нажата, выходное напряжение Vout = Vcc, но пока она отпущена, Vout ≠ 0. Кнопка и провода в этом случае работают как антенна, и Vout будет «шуметь», принимая случайные значения «из воздуха».

Пока соединения нет, необходимо дать резервный, слабый путь, делающий напряжение определённым. Для этого используют один из двух вариантов.

Схема со стягивающим резистором



- ✓ Есть нажатие: Vout = Vcc
- ✓ Нет нажатия: Vout = 0

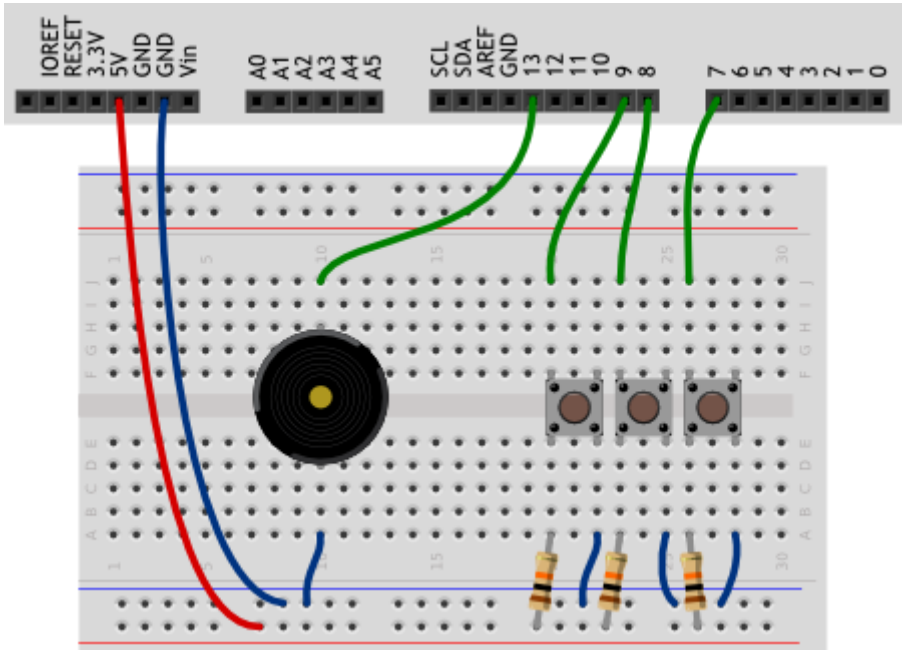
Задание 1. Ответить на вопросы

1. Что такое эффект дребезга?
2. Что произойдет, если кнопку подключить «на прямую»?

Задание 2. Список деталей для эксперимента

1. 1 плата Arduino Uno
2. 1 беспаечная макетная плата
3. 1 пьезопищалка
4. 3 тактовых кнопки
5. 3 резистора 10 КОм
6. 10 проводов «папа-папа»

Схема на макетной плате:



1. Зарисуйте принципиальную схему установки.

Обратите внимание

✓ Ножки тактовой кнопки, расположенные с одной стороны, разомкнуты, когда кнопка не нажата. Ножки, расположенные друг напротив друга на противоположных сторонах макетки находятся на одной «рельсе». Воспользовавшись этим, мы можем расположить резистор с одной стороны макетки, а провод, подключаемый к порту Arduino, с другой стороны.

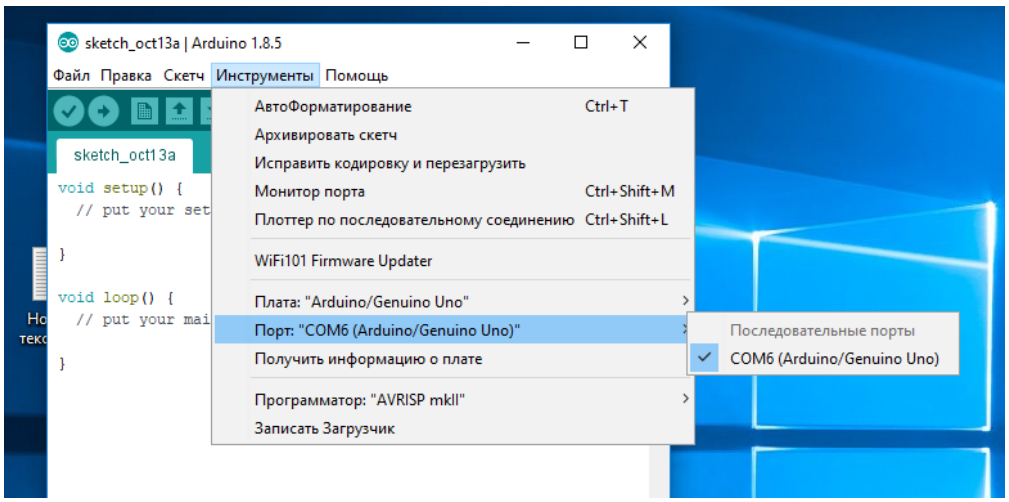
✓ В данном эксперименте мы подключаем кнопки по схеме с подтягивающим резистором.

– Для того, чтобы данный вариант программы работал, важно, чтобы кнопки были подключены к портам, находящимся рядом друг с другом, т.е. имеющим соседние номера.

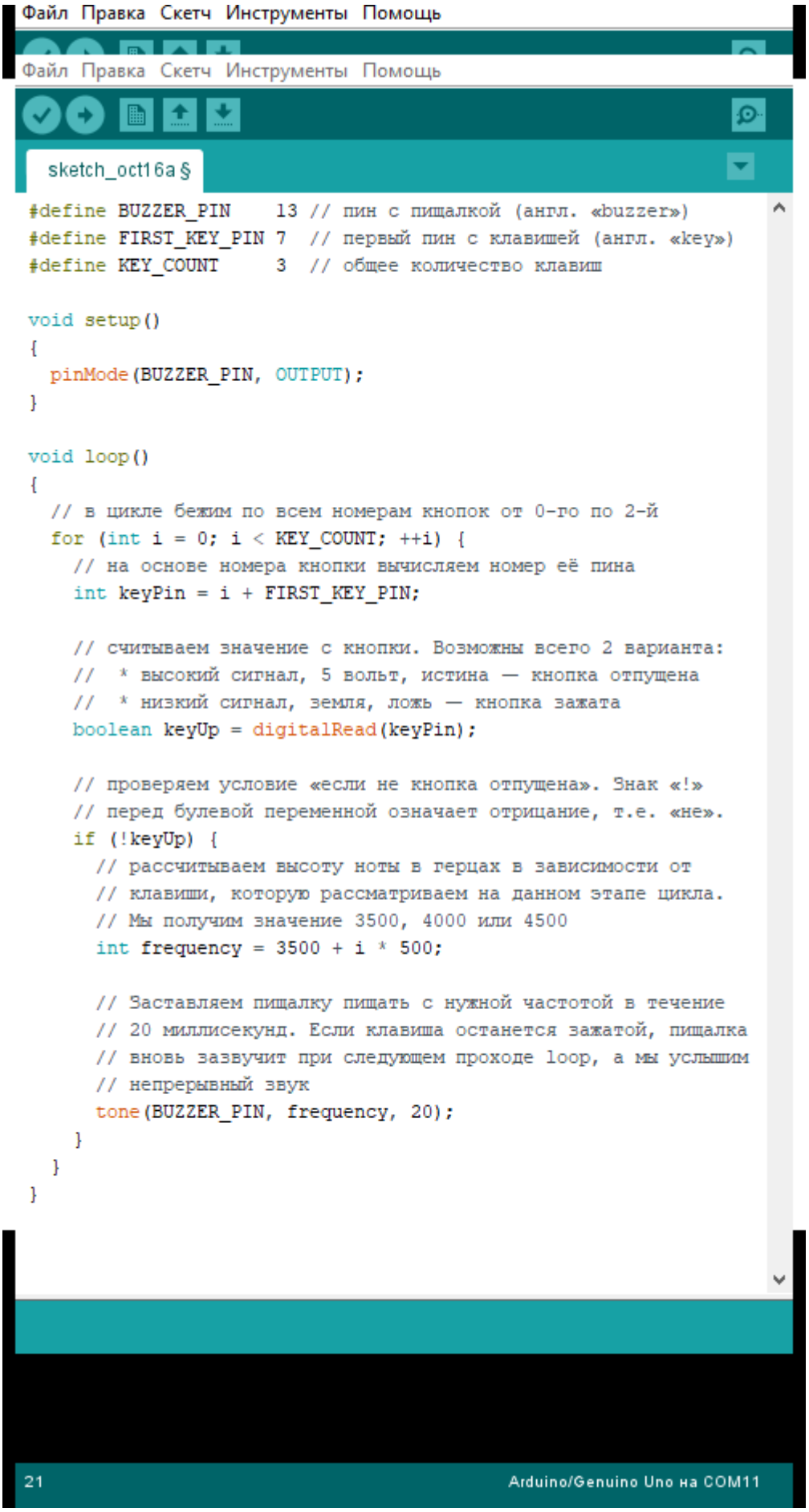
Задание 3. Программирование микроконтроллера

1. Запустите приложение

2. Убедитесь, что выбран нужный порт



3. Наберите в редакторе кода следующий код программы:


Файл Правка Скетч Инструменты Помощь
Файл Правка Скетч Инструменты Помощь
sketch_oct16a §
#define BUZZER_PIN 13 // пин с пищалкой (англ. «buzzer»)
#define FIRST_KEY_PIN 7 // первый пин с клавишей (англ. «key»)
#define KEY_COUNT 3 // общее количество клавиш
void setup()
{
pinMode(BUZZER_PIN, OUTPUT);
}
void loop()
{
// в цикле бежим по всем номерам кнопок от 0-го по 2-й
for (int i = 0; i < KEY_COUNT; ++i) {
// на основе номера кнопки вычисляем номер её пина
int keyPin = i + FIRST_KEY_PIN;
// считываем значение с кнопки. Возможны всего 2 варианта:
// * высокий сигнал, 5 вольт, истина — кнопка отпущена
// * низкий сигнал, земля, ложь — кнопка зажата
boolean keyUp = digitalRead(keyPin);
// проверяем условие «если не кнопка отпущена». Знак «!»
// перед булевой переменной означает отрицание, т.е. «не».
if (!keyUp) {
// рассчитываем высоту ноты в герцах в зависимости от
// клавиши, которую рассматриваем на данном этапе цикла.
// Мы получим значение 3500, 4000 или 4500
int frequency = 3500 + i * 500;
// Заставляем пищалку пищать с нужной частотой в течение
// 20 миллисекунд. Если клавиша останется зажатой, пищалка
// вновь зазвучит при следующем проходе loop, а мы услышим
// непрерывный звук
tone(BUZZER_PIN, frequency, 20);
}
}
}
21
Arduino/Genuino Uno на COM11

Пояснения к коду

✓ Благодаря тому, что в начале программы мы определили FIRST_KEY_PIN и KEY_COUNT, мы можем подключать произвольное количество кнопок к любым идущим друг за другом цифровым пинам, и для корректировки программы нам не придется менять параметры цикла for. Изменить понадобится лишь эти константы:

- цикл в любом случае пробегает от 0 до KEY_COUNT;
- перед считыванием порта мы задаем смещение на номер первого используемого порта — FIRST_KEY_PIN.

✓ Функция digitalRead(pin) возвращает состояние порта, номер которого передан ей параметром pin. Это может быть состояние HIGH или LOW. Или, выражаясь иначе: высокое напряжение или низкое, 1 или 0, true или false

✓ Поскольку мы получаем с порта одно из двух состояний, мы сохраняем его в переменную уже знакомого нам типа boolean, и можем работать с ней как с логическим значением.

✓ Мы используем логический оператор отрицания «не» !. Если keyUp имеет значение 0, выражение !keyUp будет иметь значение 1 и наоборот.

✓ Поскольку мы собрали схему с подтягивающим резистором, при нажатии кнопки мы будем получать на соответствующем порте 0.

✓ Действия, описанные в условном выражении if, выполняются, когда его условие имеет значение «истина» (единица). Поэтому для выполнения действия по нажатию, мы инвертируем сигнал с кнопки.

Задание 4. Ответьте на следующие вопросы

1. Почему мы не настраивали порты, к которым подключены кнопки, как `INPUT`, но устройство работает?
2. Каким образом мы избежали написания отдельного когда для чтения каждой кнопки?

3. Почему разные «ноты», издаваемые пищалкой, звучат с разной громкостью?
4. Для чего мы использовали оператор логического отрицания `!`?

Задание 5. Самостоятельно измените существующую программу и схему

1. Сделайте так, чтобы наше пианино звучало в диапазоне от 2 кГц до 5 кГц.

2. Добавьте еще 2 кнопки и измените программу так, чтобы можно было извлечь 5 различных нот.

3. Подключите кнопки по схеме со стягивающим резистором и измените программу так, чтобы она продолжала работать.