Лабораторная работа 3
Тема: «Гашение дребезга контактов программным способом»

Цель работы: приобрести практические навыки по устранению дребезга контактов при подключении кнопок к микроконтроллерам.

Последовательность выполнения работы:

- Изучить теоретические сведения приведенные в лабораторной работе.
- Собрать схемы на макетной плате для приведенных примеров.
- Запрограммировать микроконтроллер согласно тексту программы указанному в примере.
- Выполнить задание для самостоятельной работы.

СОДЕРЖАНИЕ ОТЧЕТА

- Название лабораторной работы, ее цель.
- Задание на лабораторную работу (по варианту).
- Схемы подключения к микроконтроллеру.
- Программный код для скетчей.
- Вывод о проделанной работе.

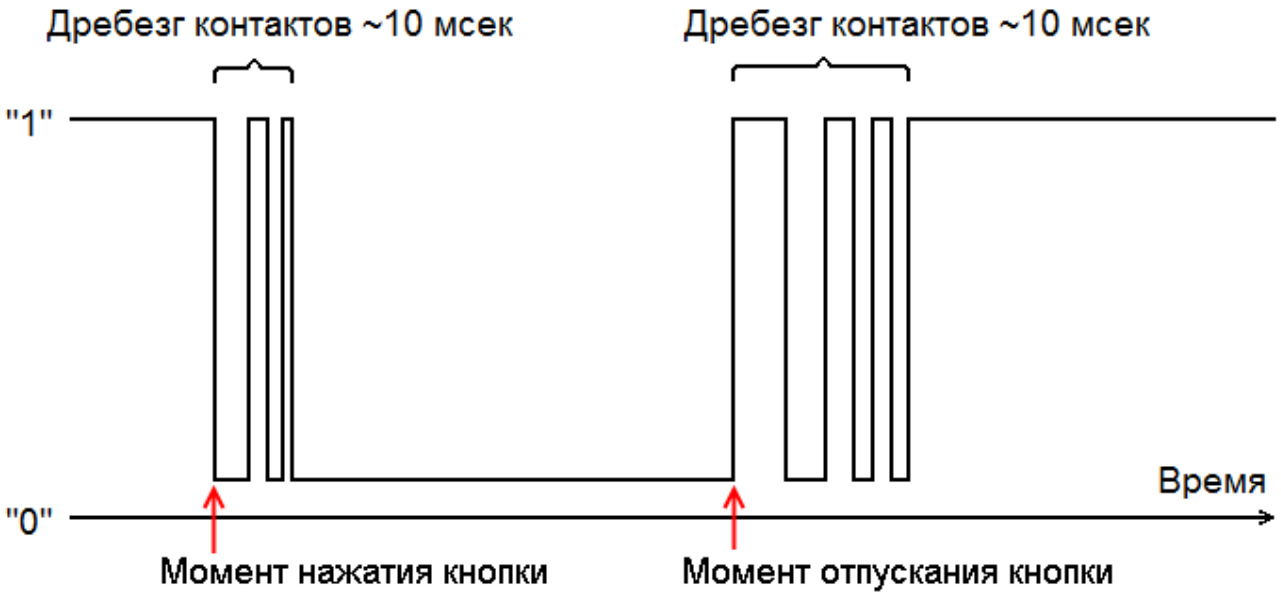
Теоретические сведения

Инструкция по подключению кнопок к микроконтроллерам.

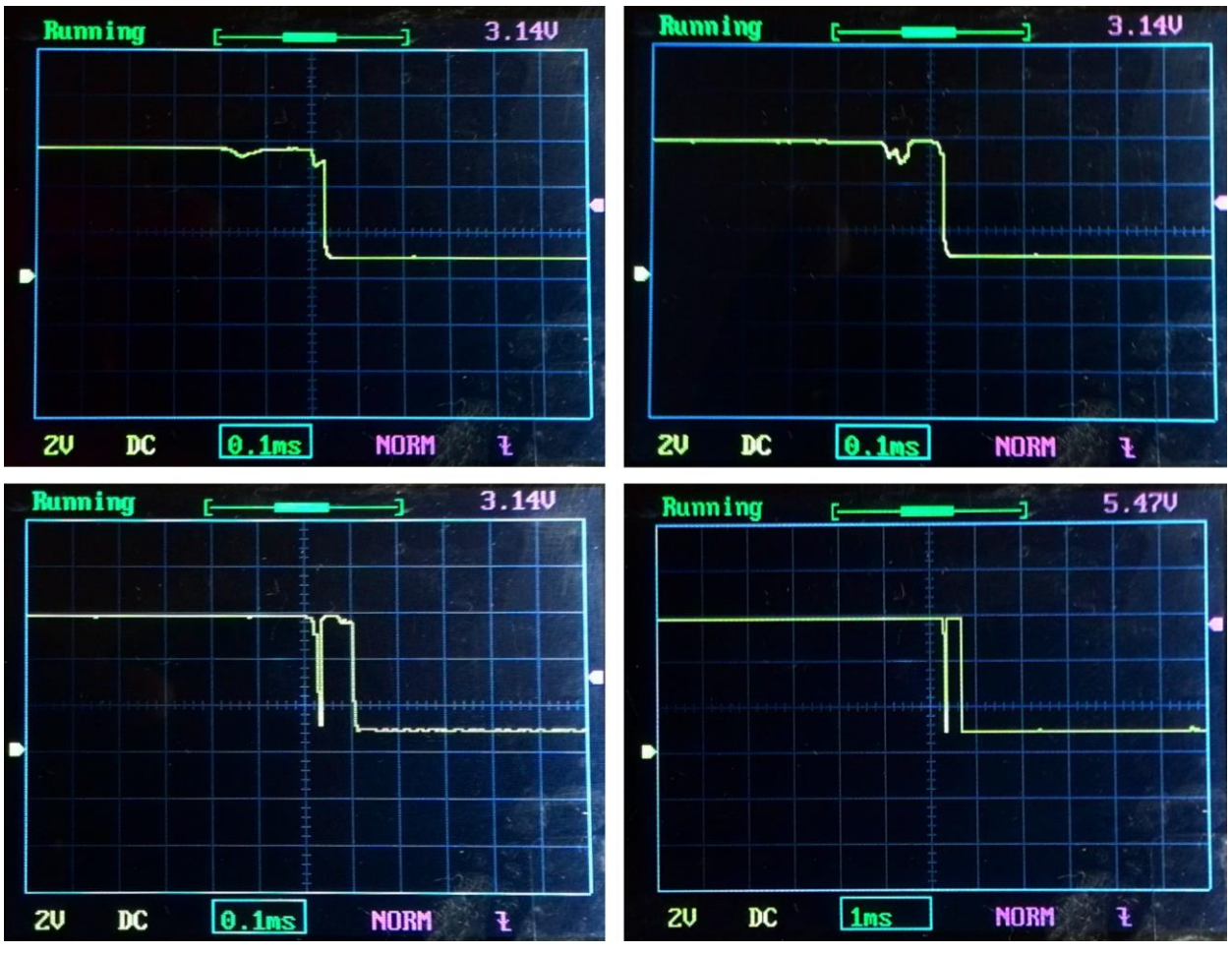
- Вам понадобится
- Arduino;
- тактовая кнопка;
- резистор номиналом 10 кОм;
- светодиод;
- соединительные провода.

«Дребезг» контактов

Кнопка – очень простое и полезное изобретение, служащее для лучшего взаимодействия человека и техники. Но, как и всё в природе, она не идеальна. Проявляется это в том, что при нажатии на кнопку и при её отпускании возникает т.н. «дребезг» ("bounce" по-английски). Это многократное переключение состояния кнопки за короткий промежуток времени (порядка нескольких миллисекунд), прежде чем она примет установившееся состояние. Это нежелательное явление возникает в момент переключения кнопки из-за упругости материалов кнопки или из-за возникающих при электрическом контакте микроискр.



«Дребезг» контактов – это явление, свойственное механическим переключателям, кнопкам, тумблерам и реле. Из-за того, что контакты обычно делают из металлов и сплавов, которые обладают упругостью, при физическом замыкании они не сразу устанавливают надёжное соединение. В течение короткого промежутка времени контакты несколько раз смыкаются и отталкиваются друг от друга. В результате этого электрический ток принимает установившееся значение не моментально, а после череды нарастаний и спадов. Длительность этого переходного эффекта зависит от материала контактов, от их размера и конструкции. На иллюстрации показана типичная осциллограмма при замыкании контактов тактовой кнопки. Видно, что время от момента переключения до установившегося состояния составляет несколько миллисекунд. Это и называется «дребезгом».



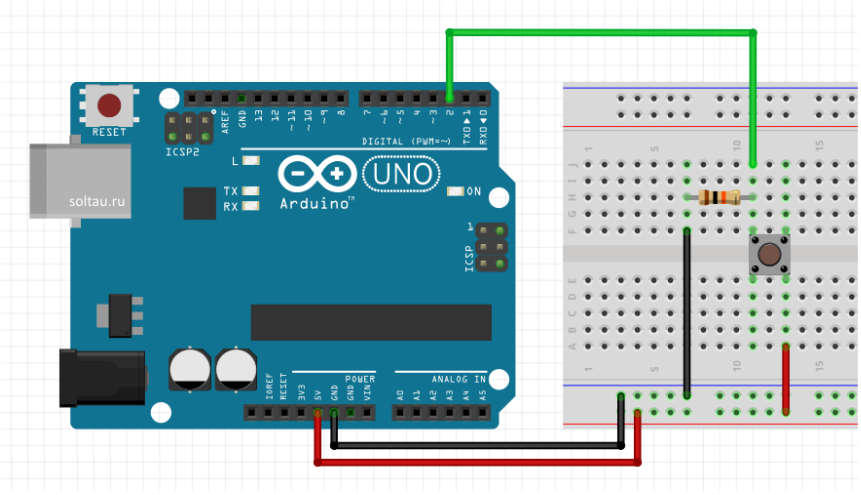
Так выглядит эффект дребезга контактов на осциллограммах

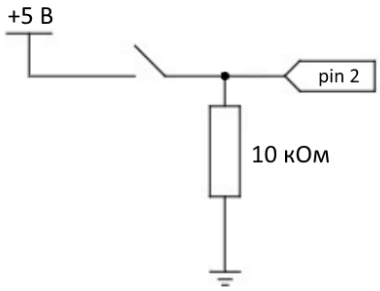
> Этот эффект не заметен в электрических цепях управления освещением, двигателями или другими **инерционными** датчиками и приборами.

Но в цепях, где идёт быстрое считывание и обработка информации (где частоты того же порядка, что и импульсы «дребезга», или выше), это является проблемой. В частности, Arduino UNO, который работает на частоте 16 МГц, отлично ловит «дребезг» контактов, принимая последовательность единиц и нулей вместо единичного переключения от 0 к 1.

Подключение кнопки к Arduino для демонстрации подавления «дребезга»

Давайте посмотрим, как дребезг контактов влияет на правильную работу схемы. Подключим к Arduino тактовую кнопку по схеме со стягивающим резистором. Будем по нажатию кнопки зажигать светодиод и оставлять включённым до повторного нажатия кнопки. Для наглядности подключим к цифровому выводу 13 внешний светодиод, хотя можно обойтись и встроенным.





Алгоритм подавления «дребезга» контактов

Чтобы реализовать задачу подавления дребезга контактов, первое, что приходит в голову:

запоминать предыдущее состояние кнопки;
сравнивать с текущим состоянием;
если состояние изменилось, то меняем состояние светодиода.

Напишем такой скетч и загрузим в память Arduino.

```
int switchPin = 2; // вывод считывания кнопки
int ledPin = 13; // вывод светодиода
boolean lastButton = false; // предыдущее состояние
кнопки
boolean ledOn = false; // включён или выключен светодиод

void setup() {
  pinMode(switchPin, INPUT); // состояние кнопки
считываем (in)
  pinMode(ledPin, OUTPUT); // светодиод запитываем (out)
}

void loop() {
  int pressed = digitalRead(switchPin); /* состояние
кнопки:
  HIGH, true - нажата, LOW, false - нет */
  if (pressed == true && lastButton == false) { /* если
кнопка сейчас нажата, а до этого была не нажата */
    ledOn = !ledOn; // меняем состояние светодиода
    lastButton = true; // запоминаем новое состояние
кнопки
  }
  else {
    lastButton = digitalRead(switchPin); // считываем
состояние кнопки
  }
  digitalWrite(ledPin, ledOn); // зажигаем или гасим
светодиод
}
```

Подавление дребезга контактов с помощью задержки

Постараемся исправить ситуацию. Мы знаем, что дребезг контактов проявляет себя в течение нескольких миллисекунд после замыкания контактов. Давайте после изменения состояния кнопки выжидать, скажем, 5 мсек. Это время для человека является практически мгновением, и нажатие кнопки человеком обычно происходит значительно дольше – несколько десятков миллисекунд. А Arduino прекрасно работает с такими короткими промежутками времени, и эти 5 мсек позволят ему отсечь дребезг контактов от нажатия кнопки.

```
int switchPin = 2; // пин кнопки
int ledPin = 13; // пин светодиода
boolean lastButton = false; // предыдущее состояние кнопки
boolean currentButton = false; // текущее состояние кнопки
boolean ledOn = false; // состояние светодиода

void setup() {
  pinMode (switchPin, INPUT);
  pinMode (ledPin, OUTPUT);
}

void loop() {
  currentButton = debounce (lastButton); // получаем состояние
кнопки без дребезга
  if (lastButton == false && currentButton == true) { // если
кнопка была нажата дольше 5 мсек,
    ledOn = !ledOn; // то меняем состояние светодиода
  }
  lastButton = currentButton; // обнуляем состояние нажатия
кнопки
  digitalWrite (ledPin, ledOn); // зажигаем/гасим светодиод
}

// Процедура определения нажатия кнопки без дребезга:
boolean debounce(boolean last) {
  boolean current = digitalRead(switchPin); // считываем
текущее состояние кнопки
  if (last != current) { // если состояние изменилось
    delay(5); // делаем задержку на 5 мсек, пока уляжется
дребезг
    current = digitalRead(switchPin); // и считываем снова
  }
  return current; // возвращаем текущее состояние кнопки
}
```

В данном скетче мы объявим процедуру debounce() ("bounce" по-английски – это как раз «дребезг», приставка "de" означает обратный процесс), на вход которой мы подаём предыдущее состояние кнопки. Если нажатие кнопки длится более 5 мсек, значит это действительно нажатие. Определив нажатие, мы меняем состояние светодиода.

Загрузим скетч в плату Arduino. Теперь всё гораздо лучше! Кнопка срабатывает без сбоев, при нажатии светодиод меняет состояние, как мы и хотели.

Библиотеки для подавления дребезга контактов

Аналогичная функциональность обеспечивается специальными библиотеками, например, библиотекой Bounce2. Для установки библиотеки помещаем её в директорию /libraries/ среды разработки Arduino и перезапускаем IDE.

Библиотека **Bounce2** содержит следующие методы:

Название	Назначение
Bounce()	инициализация объекта "Bounce";
void interval (мсек)	устанавливает время задержки в миллисекундах;
void attach (номерПина)	задаёт вывод, к которому подключена кнопка;
int update()	обновляет объект и возвращает true, если состояние пина изменилось, и false в противном случае;
int read()	считывает новое состояние пина.

Перепишем наш скетч с использованием библиотеки. Можно также запоминать и сравнивать прошлое состояние кнопки с текущим, но давайте упростим алгоритм.

```
#include <Bounce2.h> // подключаем библиотеку

const int switchPin = 2; // пин кнопки
const int ledPin = 13; // пин светодиода
int cnt = 0; // счётчик нажатий

Bounce b = Bounce(); // инстанциируем объект Bounce

void setup() {
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH); // включаем подтягивающий
резистор
  pinMode(ledPin, OUTPUT);
  b.attach(switchPin); // объект Bounce будет слушать кнопку
на пине switchPin
  b.interval(5); // устанавливаем время задержки в [мс]
}

void loop() {
  if (b.update() && b.read() == 0) { // если зарегистрировано
событие и кнопка нажата,
    cnt += 1; // инкрементируем счётчик нажатий
    if (cnt %2 == 0) digitalWrite(ledPin, LOW); // если
нажатий чётное число, гасим светодиод
    else digitalWrite(ledPin, HIGH); // иначе - зажигаем
светодиод
  }
}
```

При нажатии кнопки будем считать нажатия, и каждое нечётное нажатие будем включать светодиод, каждое чётное – выключать. Такой скетч смотрится лаконично, его легко прочитать и легко применить.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Написать программу опрашивающую две кнопки и зажигающую два светодиода без использования специальной библиотеки
2. Написать программу опрашивающую две кнопки и зажигающую два светодиода с использованием специальной библиотеки