Эксперимент 4. Кнопка и подтягивающий резистор

Познакомимся с кнопкой— важным электронным компонентом. Кнопка нужна везде, где требуется взаимодействие электроники с человеком. Для начала, рассмотрим, как устроена кнопка. Как можно догадаться, внутри корпуса кнопки находится пара металлических контактов. Когда кнопка находится в нормальном состоянии, ее контакты разомкнуты, а при нажатии— замыкаются.

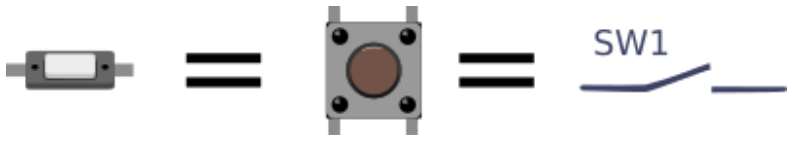


Рисунок 1. Внешний вид тактовых кнопок и обозначение на электрических схемах

Подключение кнопки

Как мы узнали ранее, микроконтроллер оперирует с понятиями *логического нуля* и *логической единицы*, но как получить их с помощью кнопки? Логично подключить кнопку между выводом микроконтроллера и напряжением питания. Когда кнопка нажимается, напряжение питания подается на ножку, что является *логической единицей*. Такая схема изображена на рисунке 2.

Рисунок 2. Неправильная схема подключения кнопки

Почему же эта схема неправильная? Дело в том, что когда кнопка нажата, на ножку микроконтроллера, действительно, подается напряжение питания, что является уровнем логической единицы, однако, когда кнопка не нажата, то на ножку не подается ничего. Говорят, что она «висит в воздухе»— то есть никуда не подключена. Такое состояние не является состоянием логического нуля. Состояние такого вывода может самопроизвольно меняться, например из-за статического электричества, или даже радиоволны. Чтобы получить состояние логического нуля, нужно подключить ножку к *земле*. Так называют минус питания, «общий» провод, относительно которого и отсчитывается напряжение во всей схеме.

Но как подключить кнопку, чтобы в одном состоянии она соединяла ножку с питанием, а в другом— с землей, ведь у нее только один контакт, который либо замкнут, либо разомкнут? В таком случае используют резистор, с помощью которого ножку микроконтроллера *подтягивают* к противоположному уровню.

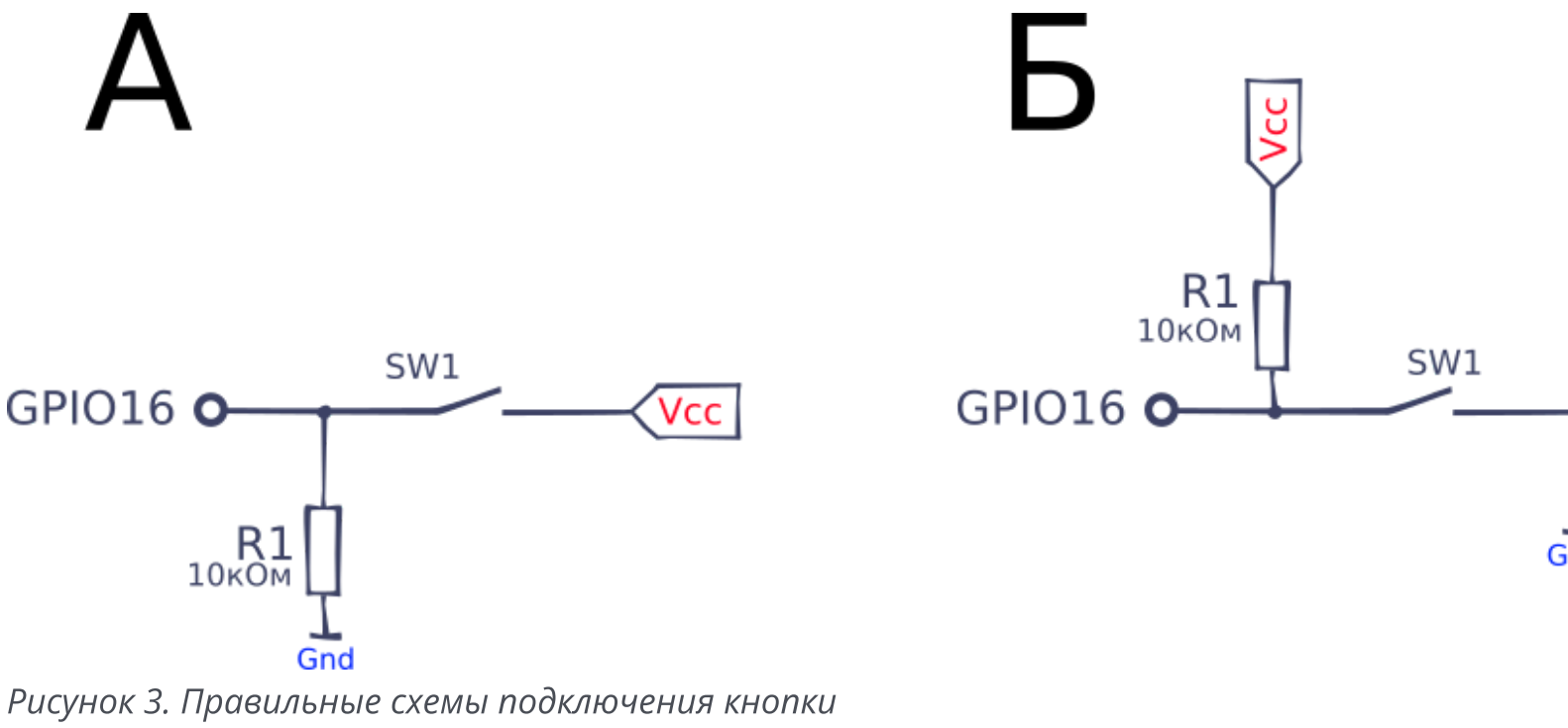


Рисунок 3. Правильные схемы подключения кнопки

На рисунке 3 изображено два варианта правильного подключения кнопки.

Вариант А: Когда кнопка не нажата, ножка микроконтроллера подключена к земле через резистор, что является состоянием логического нуля. Сопротивление подтягивающего резистора выбирают достаточно большим, чтобы уменьшить протекающий через него ток, например, 10 килоом. Когда кнопка нажата, вывод микроконтроллера подключается к плюсу питания, что является логической единицей. Хотя, в это же время, соединение с землей через резистор также сохраняется, но ток, протекающий через него, слишком мал, чтобы «соперничать» с прямым подключением к питанию, поэтому он не может помешать установлению логической единицы на ножке микроконтроллера.

Вариант Б: Когда кнопка не нажата, ножка микроконтроллера подключена к питанию через резистор, что является состоянием логической единицы. А когда нажата— к земле, что является логическим нулем.

Теперь мы знаем как правильно подключать кнопку, а из предыдущего эксперимента возьмем схему подключения светодиода. Объединим это в одну схему.

Схема эксперимента

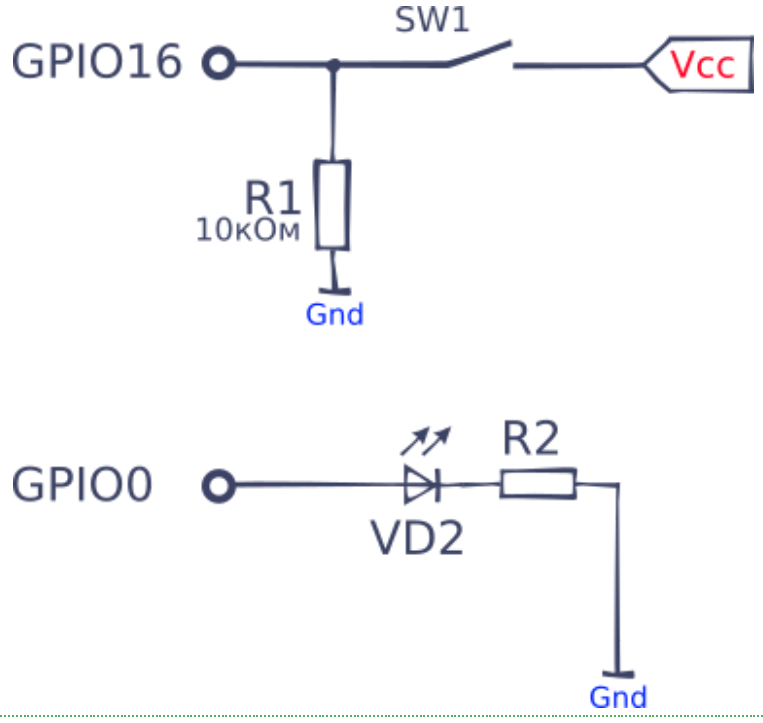


Рисунок 4. Электрическая принципиальная схема эксперимента

На рисунке изображен токоограничительный резистор последовательно со светодиодом. При сборке схемы мы не будем устанавливать его сами так как он уже установлен на плате конструктора.

Соберем эту схему:

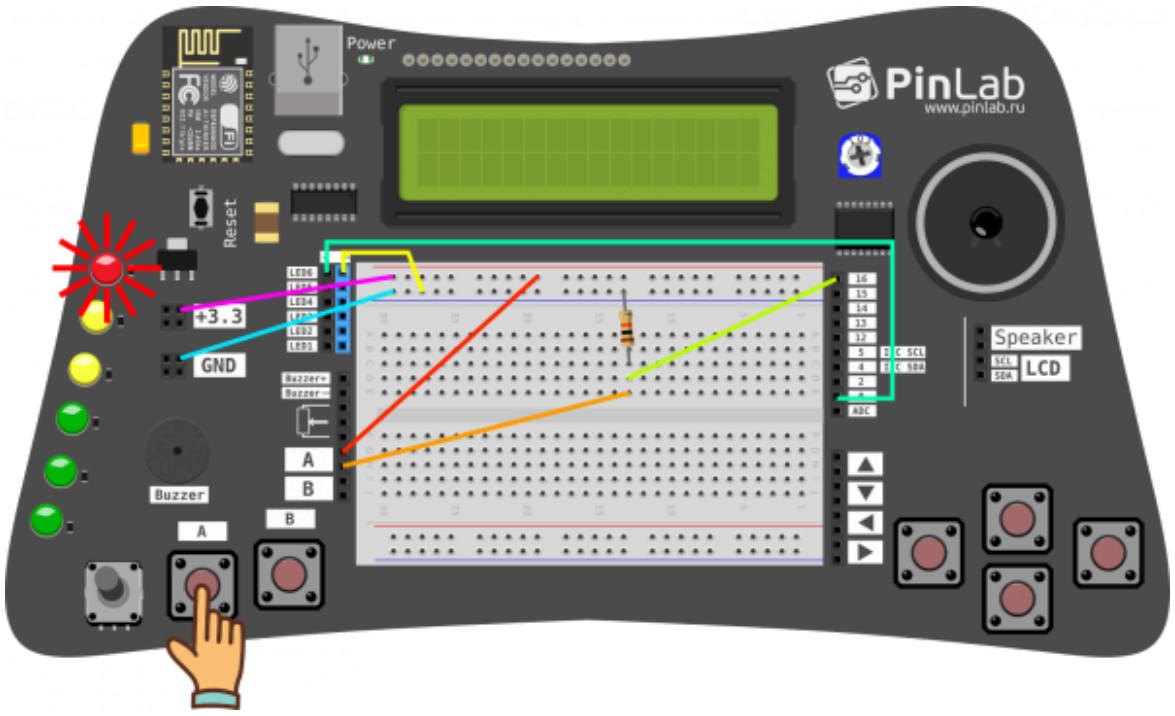


Рисунок 5. Монтажная схема эксперимента

Программный код эксперимента

Теперь напишем программный код. Пусть, когда кнопка не нажата, светодиод не горит. А при нажатии кнопки — горит.

Exp4.py

```
1.  from machine import Pin
2.  _init()
3.
4.  ButtonPin = 16
5.  LedPin = 0
6.
7.  Button = Pin(ButtonPin, Pin.IN)
8.  Led = Pin(LedPin, Pin.OUT)
9.
10. while True:
11.     button_value = Button.value()
12.     if button_value == 1:
13.         Led.on()
14.     else:
15.         Led.off()
```

Как и в прошлом эксперименте, сначала подключаем необходимые библиотеки, а именно `Pin` из системной библиотеки `machine` . Вызываем функцию инициализации платы.

```
4.  ButtonPin = 16
5.  LedPin = 0
```

Создаем переменные, в которые записываем номера выводов микроконтроллера для подключения кнопки и светодиода. Кнопку подключаем к 16 выводу, светодиод к 0.

```
4.  Button = Pin(ButtonPin, Pin.IN)
5.  Led = Pin(LedPin, Pin.OUT)
```

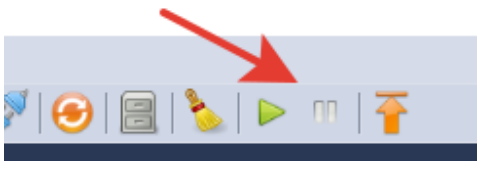
Создаем объекты выводов микроконтроллера и настраиваем их. Ножку для кнопки настраиваем как вход (`Pin.IN`), а для светодиода— как выход (`Pin.OUT`). Теперь можно работать с этими выводами.

```
10. while True:
```

Начинаем бесконечный цикл

```
11.     button_value = Button.value()
12.     if button_value == 1:
13.         Led.on()
14.     else:
15.         Led.off()
```

Тело цикла. Этот код повторяется бесконечное количество раз, пока программа не будет прервана пользователем с помощью кнопки



```
12.     button_value = Button.value()
```

Сначала мы считываем состояние вывода, к которому подключена кнопка, с помощью вызова функции `Button.value()` и записываем это состояние в переменную `button_value` . Если кнопка нажата, то (в соответствии со схемой на рисунке 4) на выводе микроконтроллера состояние логической единицы, значит в переменную `button_value` запишется `1` . А если кнопка не нажата, на ножке логический ноль, в переменную запишется 0.

```
12.     if button_value == 1:
```

Здесь мы впервые сталкиваемся с *условным оператором*. Оператор if проверяет выполнение условия и исполняет код в зависимости от того выполнено условие или нет. Если условие выполняется, то выполняется код в блоке прямо под ним. Если условие ложно, то исполняется код в блоке `else` .

В нашем случае условием является `button_value == 1` . Символ двойного равенства `==` это оператор сравнения. Он сравнивает значение выражения слева от себя со значением выражения справа от себя. Если эти значения равны, то равенство выполняется, иначе — нет.

Подробнее об операторе if [https://pythonworld.ru/osnovy/instrukciya-if-elif-else-proverka-istinnosti-trexmestnoe-vyrazhenie-ifelse.html]

Ранее мы уже сталкивались с оператором присвоения = . Этот оператор предназначен для присвоения значения переменной. Например

```
a = 2
```

присваивает значение 2 переменной `a` . Теперь же мы использовали оператор сравнения == . Он сравнивает два значения. По виду они похожи, но это совершенно разные операторы и путать их нельзя.

Итак, в строке 12 с помощью оператора `if` мы проверяем выполнение условия равенства переменной button_value цифре 1. По сути это проверка нажата ли кнопка. Если да, если условие выполняется, то выполняется код

```
13.             Led.on()
```

который зажигает светодиод. А если условие не верно, а значит кнопка не нажата, то выполняется код в блоке `else`

```
15.             Led.off()
```

который гасит светодиод.

Теперь, когда мы разобрали каждую строку программы, взглянем опять на алгоритм целиком: программа постоянно, бесконечное количество раз, проверяет нажата ли кнопка. Если кнопка нажата, то на светодиод зажигается, если же кнопка не нажата то светодиод выключается.

Дополнительное задание

1. Как изменить программу, чтобы наоборот, когда кнопка не нажата — светодиод светился, а если нажата — гас?

2. Как изменить электрическую схему, чтобы добиться такого же результата, но без изменения

программы?