Эксперимент 5. Эмуляция кнопки с фиксацией

Попробуем имитировать кнопку с фиксацией. После первого нажатия на кнопку светодиод зажигается, а гаснет после второго нажатия. Для этой цели можно применять кнопку с механической фиксацией положения включено-выключено, но мы реализуем ее программными средствами. Просто при обработке нажатия на кнопку будем учитывать ее предыдущее состояние и запоминать его.

Схема эксперимента

Схема эксперимента не изменилась по сравнению с прошлым, изменения будут только в программной части

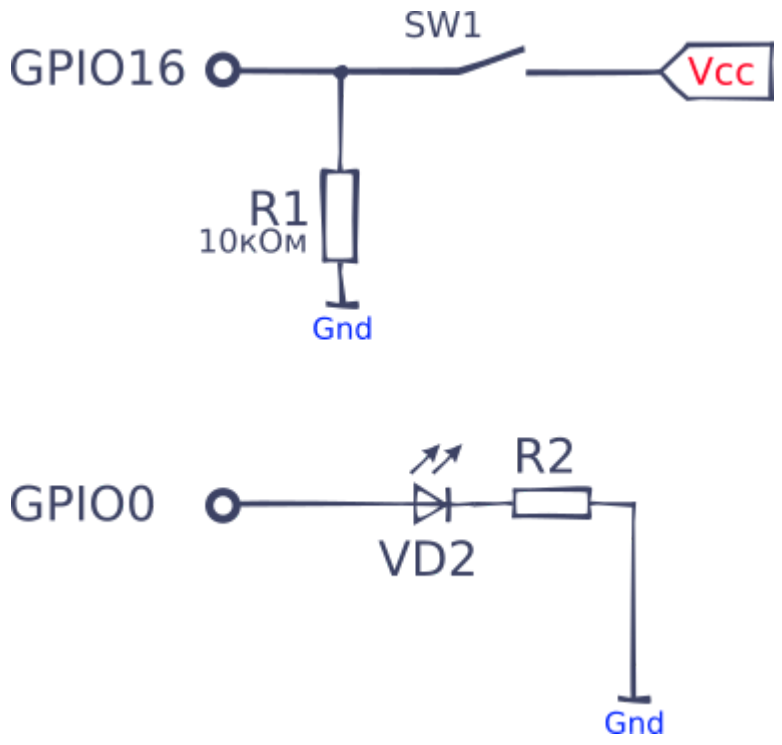

Рисунок 1. Электрическая принципиальная схема эксперимента

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:

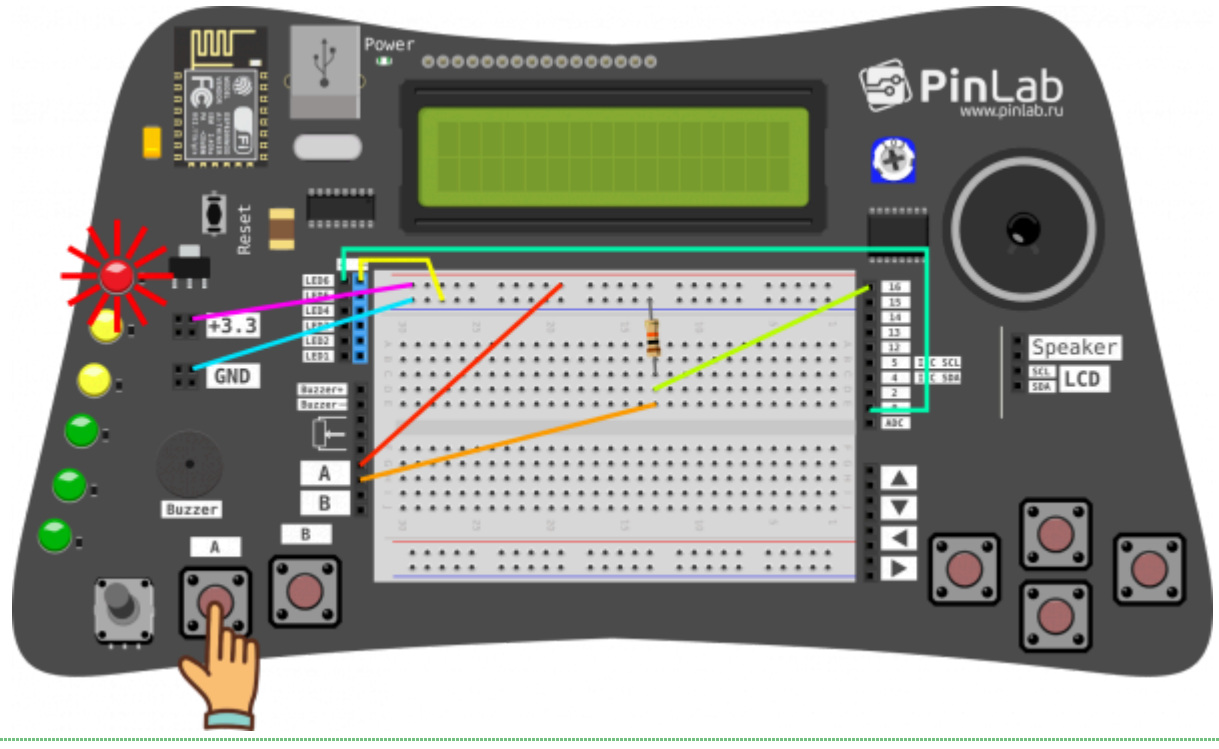

Рисунок 2. Монтажная схема эксперимента

Программный код эксперимента

Exp5.py

```
1.  from machine import Pin
2.  _init()
3.
4.  ButtonPin = 16
5.  LedPin = 0
6.  old_button_value = 0
7.  LedState = 0
8.
9.  Button = Pin(ButtonPin, Pin.IN)
10. Led = Pin(LedPin, Pin.OUT)
11.
12. while True:
13.     button_value = Button.value()
14.     if old_button_value != button_value and button_value == 1:
15.         if LedState:
16.             Led.off()
17.             LedState = 0
18.         else:
19.             Led.on()
20.             LedState = 1
21.
22.     old_button_value = button_value
```

Создадим две переменные:

- `old_button_value` будем использовать для хранения результата прошлой проверки состояния кнопки. Это нам потребуется для выявления факта нажатия на кнопку. Если при прошлой проверке кнопка была не нажата, а при текущей проверке нажата — значит только что произошло нажатие.
- `LedState` будем использовать для хранения текущего состояния светодиода — включен он или выключен. Для того, чтобы менять это состояние на противоположное после фиксации факта нажатия на кнопку.

Считываем состояние кнопки и проверяем факт нажатия:

```
13.     button_value = Button.value()
14.     if old_button_value != button_value and button_value == 1:
```

Выражение `old_button_value != button_value` проверяет факт того, что новое состояние кнопки не равно предыдущему. Оператор `!=` — оператор неравенства. Выражение `button_value == 1` проверяет текущее состояние кнопки. Это выражение истинно, когда кнопка нажата. А между этими двумя выражениями мы применили логический оператор `and` . Он переводится как И, возвращает истину когда оба выражения (слева и справа от него) истины. Если хотя бы одно из них не выполняется, то и все выражение целиком ложно.

Иными словами проверяем условие того, что состояние кнопки изменилось И, что новым состоянием кнопки является нажатое состояние. Это выражение позволяет выявить факт нажатия на кнопку.

После регистрации факта нажатии на кнопку нам нужно изменить состояние светодиода на противоположное. Если он был включен — выключаем. Если выключен — включаем.

Напоследок обновляем старое состояние состояние кнопки.

```
22.  old_button_value = button_value
```

Дополнительные задания

Модифицируй программу так, чтобы светодиод зажигался или выключался только после двух нажатий. Подсказка: потребуется переменная для хранения количества нажатий.