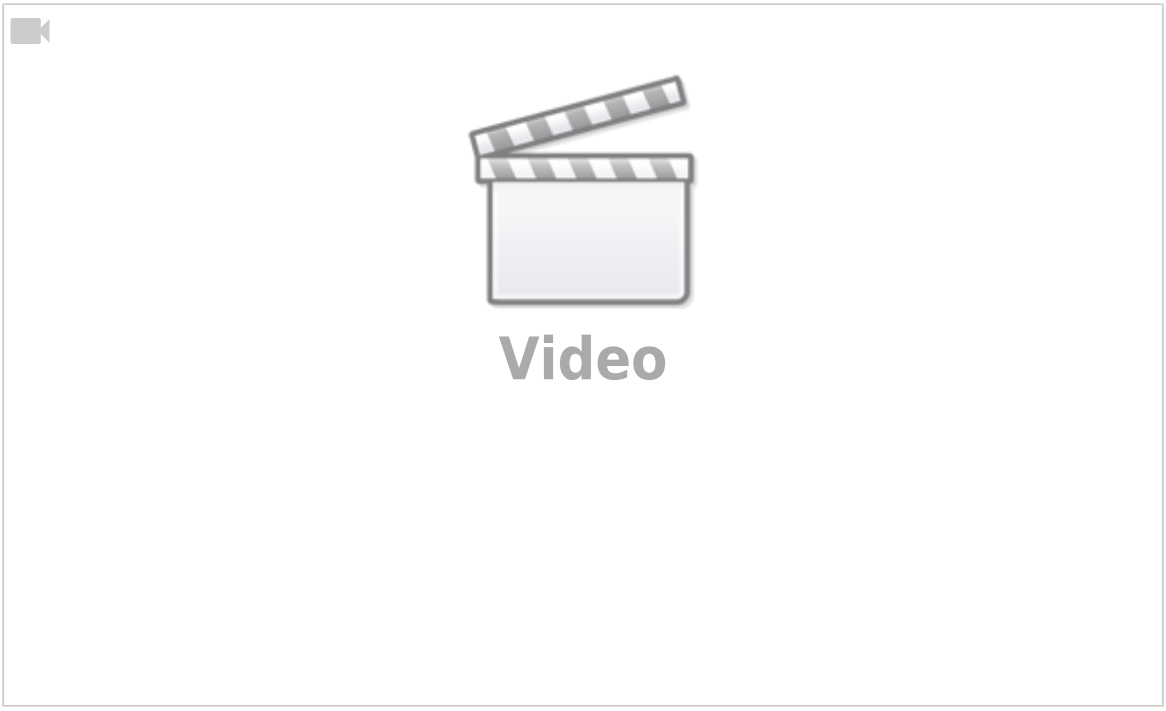
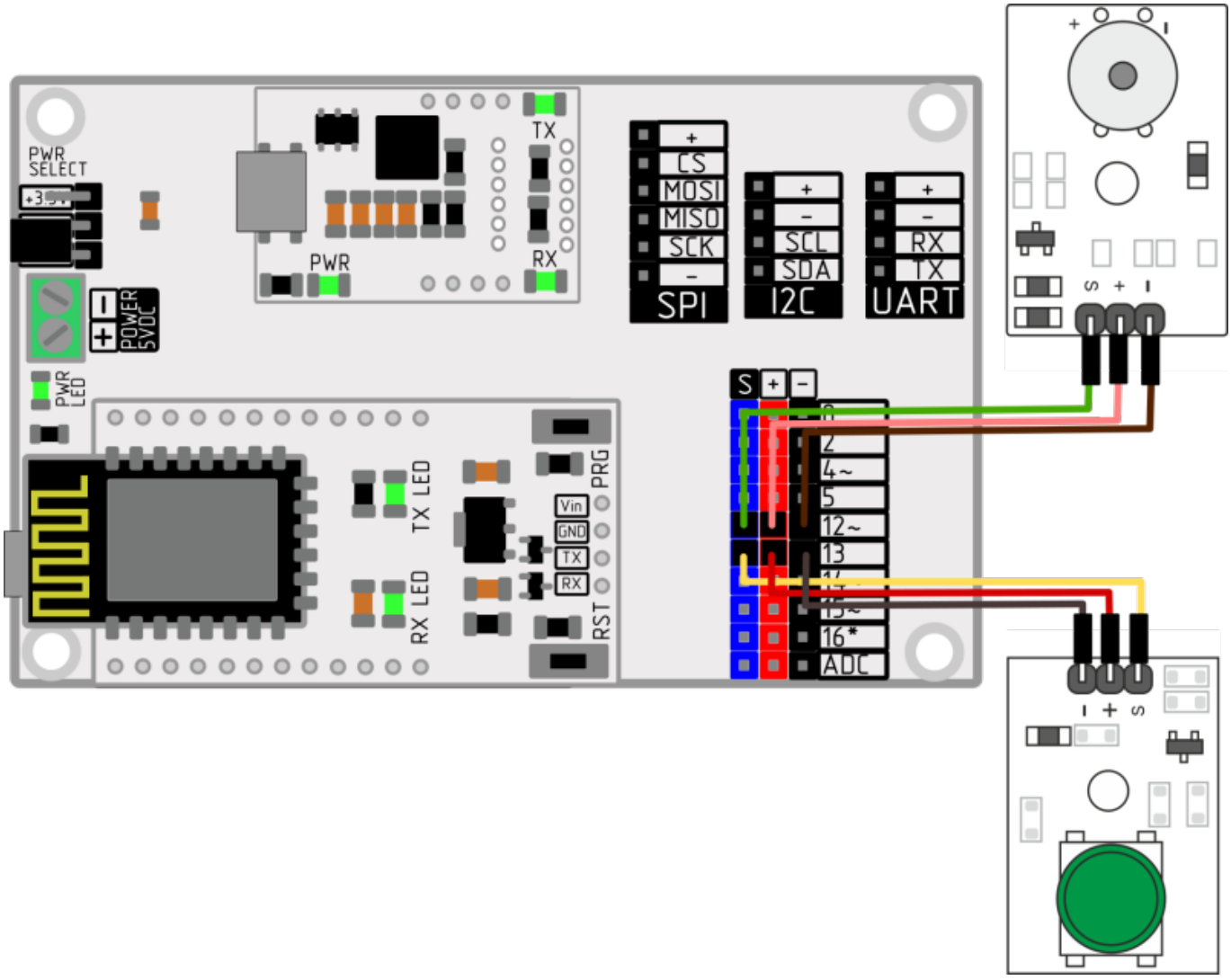
Урок 11. Списки



В этом уроке мы будем использовать ту же схему подключения что и в прошлом уроке:



Поэкспериментируем со звучанием зуммера. Задавая разную частоту звучания, звук становится выше или ниже. Попробуем установить разную частоту звучания и сравнить звук. Зададим несколько частот (например, 100, затем 200 и так далее) и воспроизведём их на зуммере. Для того чтобы задать несколько частот, можно для каждой создать свою переменную. Примерно так:

```
freq100 = 100
freq200 = 200
freq300 = 300
```

Но на самом деле это не очень удобно, ведь мы можем начать подбирать звучание частот на слух и увидеть что понадобятся ещё промежуточные значения, или пару убрать и заменить. Логичнее было бы хранить значения частот в одной переменной и туда их добавлять или убирать.

Именно для этого существуют **контейнеры** (или **коллекции значений**). То есть это переменная, которая хранит несколько значений. Одним из таких простых контейнеров является **список**. **Список значений** - это переменная, которая хранит в себе несколько значений. Указывать что данная переменная хранит список значений нужно с помощью квадратных скобок. например, объявим переменную `freqs` и укажем что это список значений, но он пустой.

```
freqs = []
```

А сейчас, объявим переменную и сразу добавим в список три значения: 100, 200 и 300. Для этого в квадратных скобках нужно перечислить значения через запятую.

```
freqs = [100, 200, 300]
```

Сейчас переменная freqs хранит в себе три значения. Это можно представить себе как таблицу с одной колонкой:

100
200
300

Добавить значение в список можно с помощью функции append.

```
freqs.append(400)
```

Таким образом список будет выглядеть так:

100
200
300
400

Для того чтобы удалить значение из списка, воспользуйтесь функцией `remove()`:

```
freqs.remove(300)
```

Теперь список будет выглядеть так:

100
200
400

К каждому элементу списка можно обращаться по номеру. То есть можно представить что система автоматически нумерует каждое значение в списке (**нумерация начинается с 0**). То есть список тогда будет выглядеть как таблица из двух колонок. Первая колонка - вспомогательная, хранит порядковый номер значения.

0	100
1	200
2	400

Если нам нужен второй элемент (а так как нумерация начинается с нуля, то второй элемент имеет индекс 1), то обратиться к нему по индексу нужно так:

```
freqs[1]
```

Итак, для сравнения зададим список из трёх частот и воспроизведём их на зуммере с паузой в

секунду. Для воспроизведения одной частоты напишем функцию под названием sound, которая будет принимать значение частоты и воспроизводить звук данной частоты в течении трёх секунд:

```
# функция для воспроизведения звука определённой частоты
def sound(new_freq):
    buz.freq(new_freq)   # установка частоты из параметра
    buz.duty(512)   # установить заполнение в 512
    sleep(3)   # задержка в 3 секунды
    buz.duty(0)   # установить заполнение в 0
    sleep(1)   # задержка в 1 секунду
```

Для того чтобы воспроизвести все три частоты из списка напишем функцию show_all_sounds, которая поочерёдно для каждого элемента списка будет вызывать написанную нами ранее функцию sound:

```
# функция для воспроизведения всех звуков
def show_all_sounds():
    sound(freqs[0])   # вызов функции  sound  с первым значением из списка  freqs
    sound(freqs[1])   # вызов функции  sound  со вторым значением из списка  freqs
    sound(freqs[2])   # вызов функции  sound  с третьим значением из списка  freqs
```

А в основном цикле программы укажем, что при нажатии на кнопку, нужно вызвать функцию show_all_sounds, которая и воспроизведёт все звуки:

```
# основной цикл программы
while True:
    if but.value():
        show_all_sounds()   # если нажали на кнопку, воспроизведём все звуки
```

Полностью скрипт будет выглядеть следующим образом:

```
# импорт модулей
from machine import Pin, PWM
from time import sleep

# создадим ШИМ вывод  buz  на 12 выводе
buz = PWM(Pin(12, Pin.OUT))

freqs = [100, 500, 1000]

# кнопка подключена к выводу 13
but = Pin(13, Pin.IN)

# функция для воспроизведения звука определённой частоты
def sound(new_freq):
    buz.freq(new_freq)   # установка частоты из параметра
    buz.duty(512)   # установить заполнение в 512
    sleep(3)   # задержка в 3 секунды
    buz.duty(0)   # установить заполнение в 0
```

```
    sleep(1)  # задержка в 1 секунду

# функция для воспроизведения всех звуков
def show_all_sounds():
    sound(freqs[0])  # вызов функции sound с первым значением из списка freqs
    sound(freqs[1])  # вызов функции sound со вторым значением из списка freqs
    sound(freqs[2])  # вызов функции sound с третьим значением из списка freqs

# основной цикл программы
while True:
    if but.value():
        show_all_sounds()  # если нажали на кнопку, воспроизведём все звуки
```

Сохраните его и запустите на исполнение. Прослушайте три варианта частоты звучания зуммера.

Запомнить:

- Список - это контейнер (коллекция значений), который может хранить несколько значений одновременно.
- Список обозначается квадратными скобками.
- В список можно добавлять значения, удалять и производить другие действия.
- К элементу списка можно обратиться по индексу.

Предыдущий урок

Следующий урок