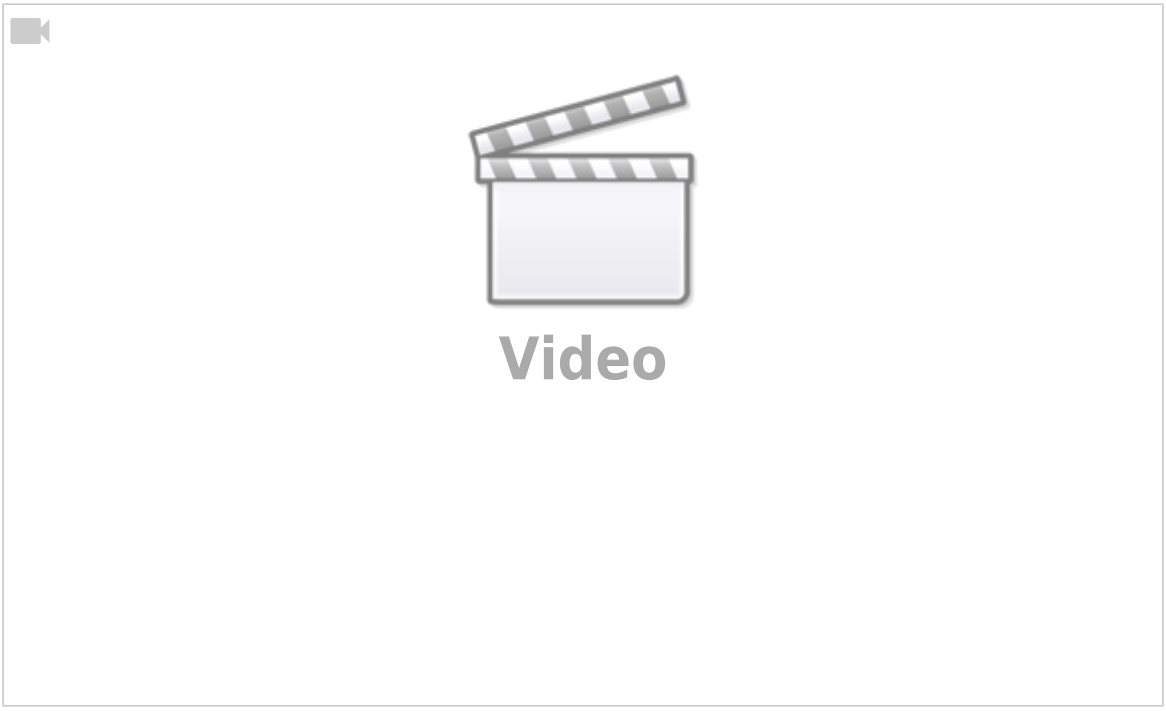
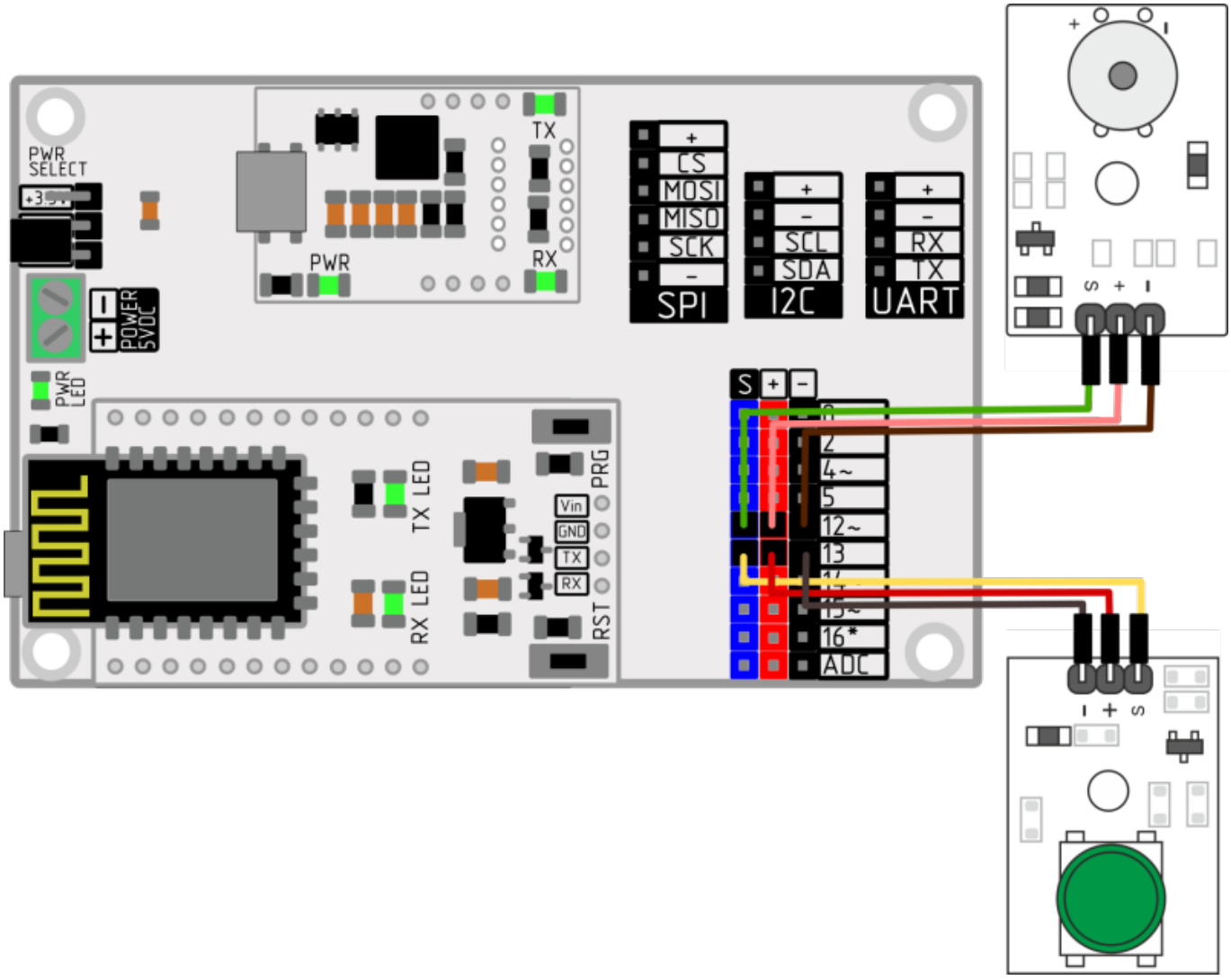
Урок 14. Диапазон



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



В цикле for в прошлом уроке мы указывали коллекцию значений (список) из которого брали значения. И при каждом проходе цикла очередное значение бралось из списка. Теперь же мы хотим задать мелодию. У нас есть не только ноты, но и длительность их звучания и пауза, которую нужно делать после каждой ноты. И все они разные.

Давайте создадим переменные для этой мелодии. Саму мелодию Вы легко угадаете, когда воспроизведёте её. А пока создадим переменную `song`, это будет список нот. Затем переменную `dur` - это будет список длительности воспроизведения каждой ноты, и, наконец, список `pauses`, хранящий паузы, которые будут делаться после каждой ноты.

Мы специально убрали пробелы при перечислении значений в списках, чтобы запись выглядела короче. Но для более удобного для прочтения вида, рекомендуется после каждой запятой в списке поставить пробел.

```
# список нот
song = ["sol","mi","mi","sol","mi","mi","sol","fa","mi","re","do","la",
"si","la","sol","mi","mi","sol","fa","mi","re","do"]

# список длительности в миллисекундах
dur = [400,200,200,400,200,200,200,200,200,200,400,400,200,200,400,200,
200,200,200,200,200,400]

# список пауз после ноты
```

```
pauses = [500,250,250,500,250,250,250,250,250,250,800,500,250,250,500,250,
250,250,250,250,250,800]
```

Длительность и паузы у нас будут измеряться миллисекундами, поэтому мы будем импортировать в начале скрипта не функцию `sleep`, а аналогичную ей `sleep_ms`, которая задерживает выполнение кода не в секундах, а в миллисекундах.

```
from time import sleep_ms
```

Ну а теперь нам надо понять как использовать цикл for чтобы пройти по каждой ноте из всех трёх списков. Списки у нас одинакового размера. Для того чтобы узнать размер списка можно воспользоваться функцией `len()` (определяет размер, *от английского lenght - длина*), в скобках указывается размер чего мы хотим узнать. Например, первого списка. Если вызвать:

```
print(len(song))
```

То мы увидим в терминале значение 22. Так и есть в нашем списке 22 ноты. В следующем списке 22 длительности. И в третьем списке 22 паузы. Таким образом нам нужно повторить цикл 22 раза. Для этого удобно использовать диапазоны - арифметическую прогрессию. Функция `range()` как раз возвращает диапазон чисел. Диапазон, по-умолчанию, начинается с нуля.

Функция `range()` принимает три аргумента:

- начиная с какого числа начать
- каким числом закончить (не включая его)
- какой интервал между числами должен быть

Первый и последний параметры не обязательны и если указан только один, то будет подразумеваться второй аргумент. То есть `range(22)` вернёт нам диапазон чисел от 0 до 22 (исключая само число 22): 0, 1, 2, 3, 4 ..., 20, 21. Мы можем использовать эту функцию как раз в цикле `for`.

Напишем функцию для воспроизведения каждой ноты, используя цикл `for` для диапазона от нуля до конца нашего списка:

```
# функция для воспроизведения всех звуков
def sing_song():
    for i in range(len(song)):
        note = song[i]
        sound(notes[note], dur[i], pauses[i])
```

Вы видите что мы использовали здесь не прямое указание размера нашего списка (22 ноты), а вычисление его длины, чтобы если мы захотим использовать больше или меньше нот, наш код продолжал работать.

Итого, целиком скрипт будет выглядеть следующим образом:

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

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

# словарь нот
notes = {}
notes["do"] = 262
notes["re"] = 294
notes["mi"] = 330
notes["fa"] = 349
notes["sol"] = 392
notes["la"] = 440
notes["si"] = 494

# список нот
song = ["sol","mi","mi","sol","mi","mi","sol","fa","mi","re","do",
"la","si","la","sol","mi","mi","sol","fa","mi","re","do"]

# список длительности в миллисекундах
dur = [400,200,200,400,200,200,200,200,200,200,400,400,200,200,400,200,
200,200,200,200,200,400]

# список пауз после ноты
pauses = [500,250,250,500,250,250,250,250,250,250,800,500,250,250,500,250,
250,250,250,250,250,800]

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

# функция для воспроизведения звука определённой частоты
def sound(freq, dur, pause):
    buz.freq(freq)   # установка частоты из параметра
    buz.duty(512)   # установить заполнение в 512
    sleep_ms(dur)   # задержка
    buz.duty(0)   # установить заполнение в 0
    sleep_ms(pause)   # пауза

# функция для воспроизведения всех звуков
def sing_song():
    for i in range(len(song)):
        note = song[i]
        sound(notes[note], dur[i], pauses[i])

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

Загрузите этот скрипт на контроллер и попробуйте угадать мелодию, которая звучит.

Запомнить:

- Функция range() возвращает диапазон чисел
- У этой функции может быть три параметра, но для её использования достаточно только одного
- Функцию очень удобно использовать в цикле for если нужно выполнять цикл ограниченное число раз

Предыдущий урок

Следующий урок