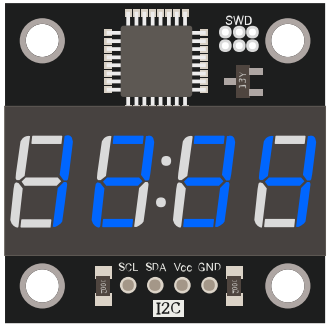
Четырехразрядный индикатор LED, FLASH-I2C, подключаем к Raspberry Pi



Общие сведения:

Trema модуль - Четырехразрядный индикатор LED, FLASH-I2C - является устройством вывода информации получаемой по шине I2C.

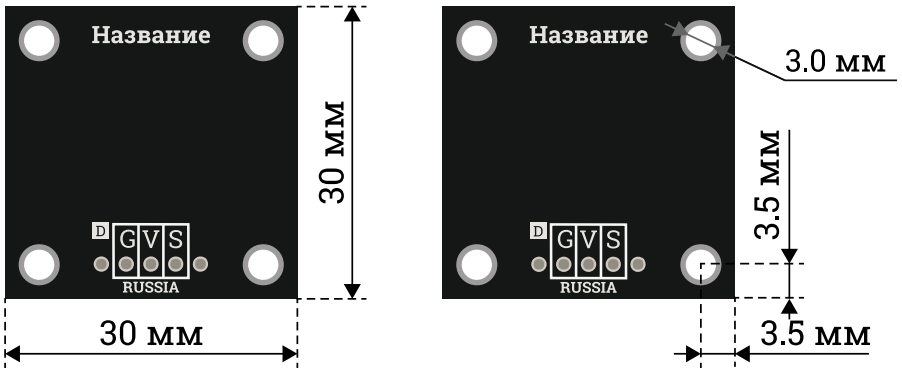
Модуль относится к серии «Flash», а значит к одной шине I2C можно подключить более 100 модулей, так как их адрес на шине I2C (по умолчанию 0x09), хранящийся в энергонезависимой памяти, можно менять программно.

Модуль является устройством вывода информации, его можно использовать для отображения чисел (в том числе и отрицательных), времени, температуры и некоторых символов.

Спецификация:

- Напряжение питания: 3,3 В или 5 В, поддерживаются оба напряжения.
- Ток потребляемый модулем: до 10 мА.
- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: Vcc.
- Количество разрядов индикатора: 4.
- Частота обновления выводимых данных: от 1 до 255 Гц.
- Рабочая температура: от -20 до +70 °C.

- Габариты: 30 х 30 мм.
- Вес: 15 г.



Подключение:

На плате модуля расположен разъем из 4 выводов для подключения к шине I2C.

- **SCL** - вход/выход линии тактирования шины I2C.
- **SDA** - вход/выход линии данных шины I2C.
- **Vcc** - вход питания 3,3 или 5 В.
- **GND** - общий вывод питания

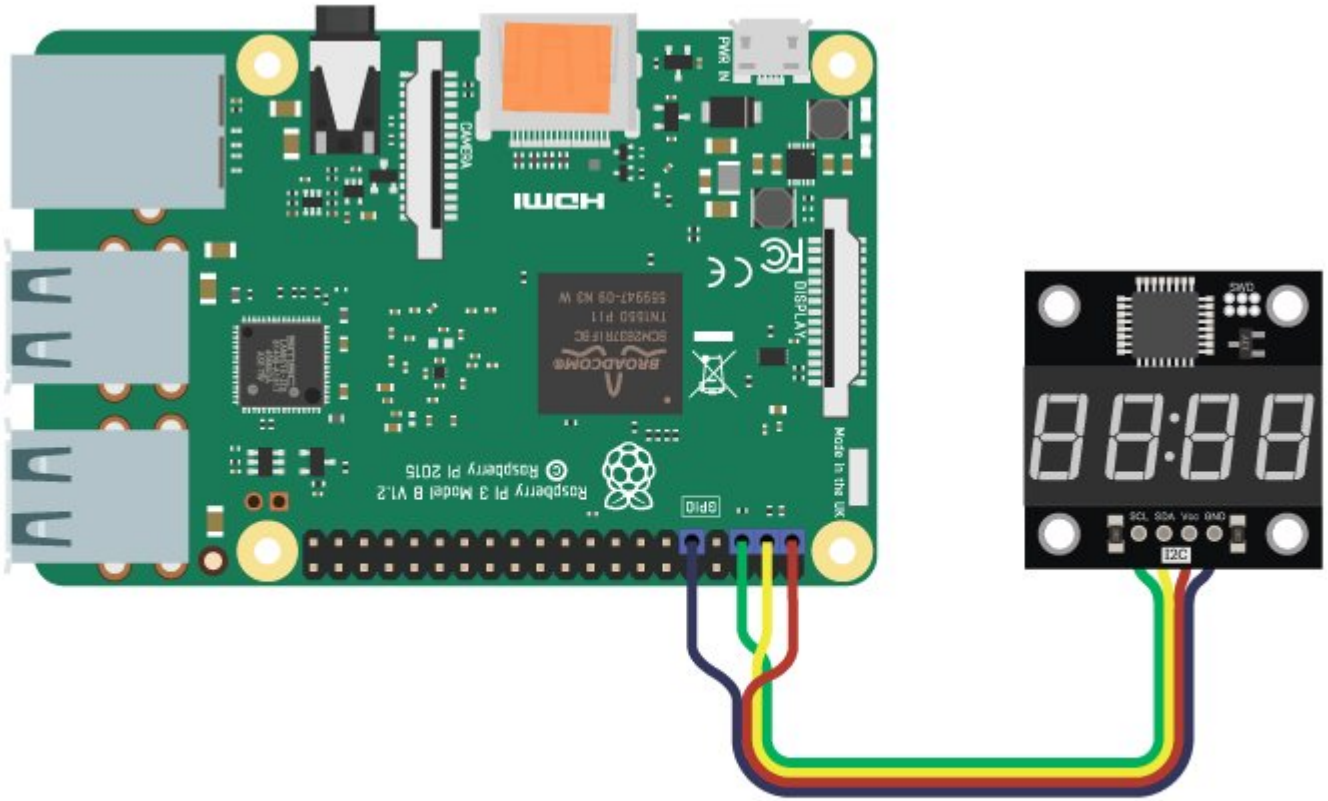
Для удобства подключения, предлагаем воспользоваться Trema+Expander Hat.

Модуль удобно подключать 2 способами, в зависимости от ситуации:

Способ - 1: Используя провода и Raspberry Pi

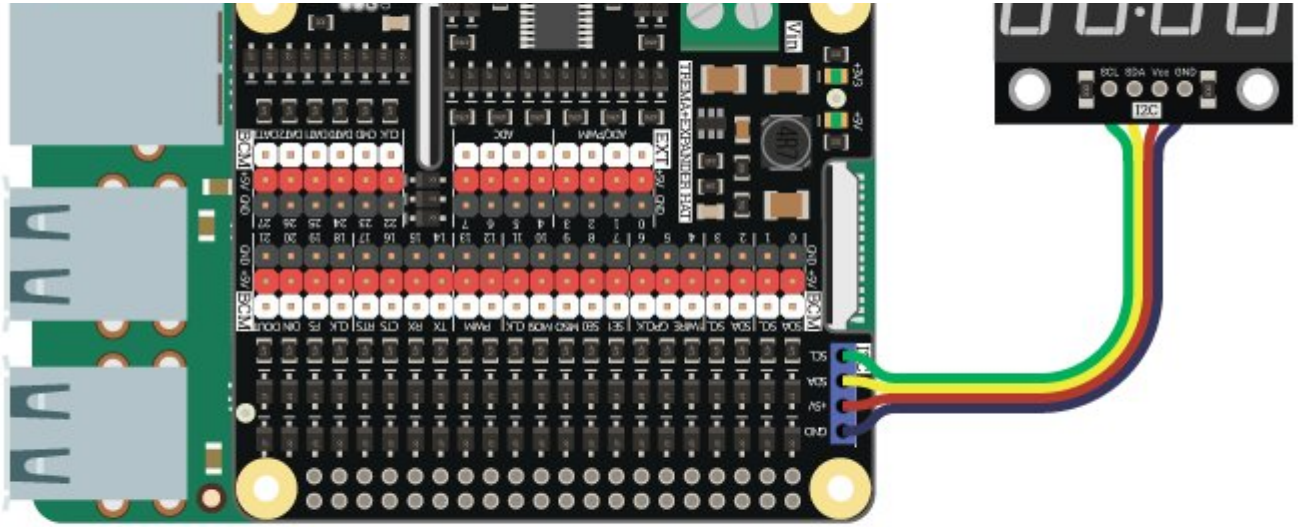
Используя провода «Мама — Мама», подключаем напрямую к Raspberry Pi

В этом случае необходимо питать логическую часть модуля от 3,3 В Raspberry



Способ - 2: Используя Trema+Expander Hat

Используя 4-х проводной шлейф, подключаем к Trema+Expander Hat.



Питание:

Входное напряжение питания модуля 3,3В или 5В постоянного тока (поддерживаются оба напряжения питания), подаётся на выводы Vcc и GND.

Подробнее о модуле:

Модуль построен на базе четырёхразрядного светодиодного индикатора, микроконтроллера STM32F030K6 и снабжён собственным стабилизатором напряжения. Модуль способен выводить числа, время, температуру и некоторые символы.

Модуль позволяет:

- Менять свой адрес на шине I2C.
- Управлять внутренней подтяжкой линий шины I2C (по умолчанию включена).
- Выводить числа, время, температуру и некоторые символы.

- Выравнивать выводимые числа по любой из 4 позиций.
- Определять направление сдвига числа от позиции по которой выполнено выравнивание.
- Задавать частоту обновления информации дисплея, от 1 до 255 Гц.
- Указывать яркость свечения всех сегментов дисплея, от 0 до 7.
- Поворачивать изображение индикатора на 180°.
- Указывать любому разряду индикатора мигать включёнными сегментами с заданной частотой.

Специально для работы с Trema модулем - Четырехразрядный индикатор LED, FLASH-I2C, нами разработана библиотека pyiArduinoI2Cled которая позволяет реализовать все функции модуля.

Для работы с модулем необходимо включить шину I2C.
Ссылка на подробное описание как это сделать.

Внимание! Для корректной работы модулей FLASH-I2C на Raspberry Pi под управлением Raspberry OS "Buster" необходимо выключить динамическое тактирование ядра (опция **core_freq_min** должна быть равна **core_freq** в `/boot/config.txt`) Ссылка на подробное описание.

Для подключения библиотеки необходимо сначала её установить. Сделать это можно в менеджере модулей в Thonny Python IDE (тогда библиотека установится локально для этого IDE), в виртуальной среде командой `pip3 install pyiArduinoI2Cled` или в терминале Raspberry (тогда библиотека будет системной) командой:

```
sudo pip3 install pyiArduinoI2Cled
```

Подробнее об установке библиотек можно узнать в этой статье.

Примеры:

Смена адреса модуля на шине I2C:

Пример позволяет указать адрес модулю, даже если его текущий адрес Вам неизвестен.

```
# Подключаем библиотеку для работы с модулем
from pyiArduinoI2Cled import *
import sys

# Объявляем объект module для работы с функциями и методами библиотеки pyiArduinoI2Cled.
# Если при объявлении объекта указать адрес, например, module(0x0B),
# то пример будет работать с тем модулем, адрес которого был указан.
module = pyiArduinoI2Cled(None, NO_BEGIN)

# Если сценарию не были переданы аргументы
if len(sys.argv) < 2:
    # Назначаем модулю адрес (0x07 < адрес < 0x7F).
    newAddress = 0x09

# Иначе
else:
    # Новый адрес - первый аргумент
    newAddress = int(sys.argv[1])

# Если модуль найден
if module.begin():
    print("Найден модуль %#.2x" % module.getAddress())

    # Если адрес удалось изменить
    if module.changeAddress(newAddress):
            print("Адрес изменён на %#.2x" % module.getAddress())

    else:
            print("Адрес не изменён!")
```

```
else:
    print("Модуль не найден!")
```

Для работы данного примера, на шине I2C должен быть только один модуль Четырехразрядный индикатор LED.

Данный скетч демонстрирует не только возможность смены адреса на указанный в переменной `newAddress`, но и обнаружение, и вывод текущего адреса модуля на шине I2C.

Вывод текущего времени

```
from datetime import datetime
from time import sleep

# Подключаем библиотеку для работы с индикатором I2C-flash.
from pyiArduinoI2Cled import *

# Объявляем объект библиотеки pyiArduinoI2Cled, указывая адрес модуля на шине I2C.
disp = pyiArduinoI2Cled(0x09)

# Если объявить объект без указания адреса, то адрес будет найден автоматически.
#disp = pyiArduinoI2Cled()

# Указываем двоеточию мигать (если оно выводится на индикатор).
disp.blink(0, True)

try:

    while True:

        hour = datetime.now().hour
        minute = datetime.now().minute
```

```
        # Выводим значения i и j как время (через двоеточие).
        disp.print(hour, minute, TIME)

except:

    # Очищаем дисплей если сценарий был прерван из-за исключения
    disp.clear()
```

Описание функций библиотеки:

В данном разделе описаны функции библиотеки pyiArduinoI2Cled для работы с Trema модулем - Четырехразрядный индикатор LED, FLASH-I2C на Raspberry Pi.

Подключение библиотеки:

- Если адрес модуля известен (в примере используется адрес 0x09):

```
# Подключаем библиотеку для работы с модулем.
from pyiArduinoI2Cled import *
# Создаём объект module для работы с функциями и методами библиотеки iarduino_I2C_4LED, указав адрес модуля на шине I2C (0x09)
module = pyiArduinoI2Cled(0x09)
```

- Если адрес модуля неизвестен (адрес будет найден автоматически):

```
# Подключаем библиотеку для работы с модулем.
from pyiArduinoI2Cled import *
# Создаём объект module для работы с функциями и методами библиотеки iarduino_I2C_4LED, без указания адреса.
module = pyiArduinoI2Cled()
```

- При создании объекта без указания адреса, на шине должен находиться только один модуль.

Функция begin();

- Назначение: Инициализация работы с модулем.
- Синтаксис: begin();
- Параметры: Нет.
- Возвращаемое значение: результат инициализации (True или False).
- Примечание: По результату инициализации можно определить наличие модуля на шине.
- Пример:

```
if dispLED.begin():
    print( "Модуль найден и инициирован!" )
```

Функция reset();

- Назначение: Перезагрузка модуля.
- Синтаксис: reset();
- Параметры: Нет.
- Возвращаемое значение: результат перезагрузки (True или False).
- Пример:

```
if dispLED.reset():
    print( "Модуль перезагружен" )
else:
    print( "Модуль не перезагружен" )
```

Функция changeAddress();

- Назначение: Смена адреса модуля на шине I2C.
- Синтаксис: changeAddress(АДРЕС);
- Параметр:
  - АДРЕС - новый адрес модуля на шине I2C (целое число от 0x08 до 0x7E)

- Возвращаемое значение: результат смены адреса (True или False).
- Примечание:
  - Адрес модуля сохраняется в энергонезависимую память, а значит будет действовать и после отключения питания.
  - Текущий адрес модуля можно узнать функцией getAddress().
- Пример:

```
if( dispLED.changeAddress(0x12):
    print( "Адрес модуля изменён на 0x12" )
else:
    print( "Не удалось изменить адрес"    )
```

Функция getAddress()

- Назначение: Запрос текущего адреса модуля на шине I2C.
- Синтаксис: getAddress()
- Параметры: Нет.
- Возвращаемое значение: АДРЕС - текущий адрес модуля на шине I2C (от 0x08 до 0x7E)
- Примечание: Функция может понадобиться если адрес модуля не указан при создании объекта, а обнаружен библиотекой.
- Пример:

```
print( "Адрес модуля на шине I2C = ", end='' )
print( dispLED.getAddress() )
```

Функция getVersion()

- Назначение: Запрос версии прошивки модуля.
- Синтаксис: getVersion()
- Параметры: Нет
- Возвращаемое значение: ВЕРСИЯ - номер версии прошивки от 0 до 255.
- Пример:

```
print( "Версия прошивки модуля:" )
print( dispLED.getVersion() )
```

Функция setPullI2C()

- Назначение: Управление внутрисхемной подтяжкой линий шины I2C.
- Синтаксис: setPullI2C([ФЛАГ])
- Параметр:
  - ФЛАГ требующий установить внутрисхемную подтяжку линий шины I2C (True или False).
- Возвращаемое значение:
  - результат включения / отключения внутрисхемной подтяжки (True или False).
- Примечание:
  - Вызов функции без параметра равносилен вызову функции с параметром True - установить.
  - Флаг установки внутрисхемной подтяжки сохраняется в энергонезависимую память модуля, а значит будет действовать и после отключения питания.
  - Внутрисхемная подтяжка линий шины I2C осуществляется до уровня 3,3 В, но допускает устанавливать внешние подтягивающие резисторы и иные модули с подтяжкой до уровня 3,3 В или 5 В, вне зависимости от состояния внутрисхемной подтяжки модуля.

Пример:

```
if dispLED.setPullI2C(True):
    print( "Внутрисхемная подтяжка установлена." )
if dispLED.setPullI2C(false):
    print( "Внутрисхемная подтяжка отключена."   )
```

Функция getPullI2C()

- Назначение: Запрос состояния внутрисхемной подтяжки линий шины I2C.
- Синтаксис: getPullI2C()
- Параметры: Нет.

- Возвращаемое значение: ФЛАГ включения внутрисхемной подтяжки (True или False).
- Пример:

```
if dispLED.getPullI2C():
    print( "Внутрисхемная подтяжка включена."  )
else:
    print( "Внутрисхемная подтяжка отключена." )
```

Функция clear()

- Назначение: очистка индикатора
- Синтаксис: clear()
- Параметры: Нет.
- Возвращаемые значения: Нет.
- Пример:

```
# Чистим LED индикатор (все диоды выключатся).
dispLED.clear()
```

Функция light()

- Назначение: Установка яркости свечения индикатора.
- Синтаксис: light(ЧИСЛО)
- Параметры:
  - ЧИСЛО - целое число, определяющее яркость, от 0 до 7 (максимальная яркость).
- Возвращаемые значения: Нет.
- Примечание: Яркость по умолчанию = 6.
- Пример:

```
# Устанавливаем максимальную яркость свечения LED индикатора.
```

```
dispLED.light(7)
```

Функция blink()

- Назначение: Управление миганием цифр и символов в указанных позициях.
- Синтаксис: blink(ПОЗИЦИЯ , СОСТОЯНИЕ);
- Параметры:
  - ПОЗИЦИЯ - целое число, указывающее позицию цифры: 1, 2, 3, 4 или 0 для двоеточия.
  - СОСТОЯНИЕ - флаг может принимать значение 0 (не мигать) или 1 (мигать).
- Возвращаемые значения: Нет.
- Примечание:
  - Если первый параметр ПОЗИЦИЯ больше 4, то СОСТОЯНИЕ применяется ко всему индикатору.
  - Функция управляет только миганием и не влияет на ранее установленные цифры, символы и точки.
  - Частота миганий задаётся функцией `frequ()` .
- Пример:

```
# Заставляем мигать двоеточие (если двоеточие включено).
dispLED.blink(0, true)

# Заставляем мигать первую цифру или символ (теперь будет мигать и двоеточие, и символ находящийся в первой позиции).
dispLED.blink(1, true)

# Отключаем мигание двоеточия (теперь будет мигать только цифра или символ находящаяся в первой позиции).
dispLED.blink(0, false)
```

Функция frequ()

- Назначение: Установка частоты миганий цифр и символов указанных функцией `blink()` .
- Синтаксис: frequ(ЧАСТОТА)
- Параметр:
  - ЧАСТОТА - целое число, указывающее частоту в Гц, от 1 до 4.

- Возвращаемые значения: Нет.
- Примечание:
  - Частота миганий цифр и символов позиция которых указана функцией `blink()` не зависит от частоты обновления дисплея указанной функцией `fps()` .
  - Частота по умолчанию = 1 Гц.
- Пример:

```
# Цифры и символы индикатора определённые функцией blink() будут мигать с частотой 1 Гц.
dispLED.frequ(1)
```

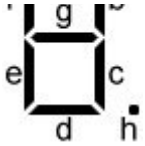
Функция turn();

- Назначение: Разворот изображения индикатора.
- Синтаксис: turn(ФЛАГ);
- Параметр:
  - ФЛАГ - может принимать значение 0 (без разворота) или 1 (повернуть на 180°).
- Возвращаемые значения: Нет.
- Примечание:
  - Если на индикаторе есть точки и они физически расположены только снизу цифр, то при развороте изображения, точки окажутся вверху и перед цифрами.
- Пример:

```
# Развернуть изображение индикатора на 180°.
dispLED.turn(1)
```

Функция setLED();

- Назначение: Установка светодиодов (сегментов) индикатора по битам.
- Синтаксис: setLED ([[[[[БАЙТ_№1] , БАЙТ_№2] , БАЙТ_№3] , БАЙТ_№4] , ФЛАГ]);

- Параметры: 

  - БАЙТ_№1 - каждый бит этого байта включает свой светодиод (сегмент) в 1 позиции.
  - БАЙТ_№2 - каждый бит этого байта включает свой светодиод (сегмент) в 2 позиции.
  - БАЙТ_№3 - каждый бит этого байта включает свой светодиод (сегмент) в 3 позиции.
  - БАЙТ_№4 - каждый бит этого байта включает свой светодиод (сегмент) в 4 позиции.
  - ФЛАГ - включает двоеточие.
- Возвращаемые значения: Нет.
- Примечание:
  - Каждый бит байта соответствует одному светодиоду (сегменту) в следующем порядке: **hgfedcba** (старший бит управляет сегментом «h», а младший бит управляет сегментом «a»).
  - Все параметры являются необязательными, отсутствие параметра означает что все светодиоды (сегменты) в данной позиции будут выключены.
- Пример:

```
# В 1 позиции включены сегменты «c» и «b» (как у цифры 1), а во 2 разряде включены сегменты «g»«f»«b»«a» (как у символа градус
dispLED.setLED(0B00000110, 0B01100011)
```

Функция print();

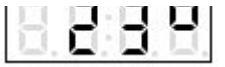
- Назначение: Вывод числа, массива чисел, текста, времени или температуры.
- Синтаксис: print (ЗНАЧЕНИЕ , ПАРАМЕТРЫ_ВЫВОДА_ЧИСЛА)
- Параметры:
  - ЗНАЧЕНИЕ - то, что требуется вывести на индикатор, это может быть:
    - Целое число (как положительное, так и отрицательное).
    - Дробное число (как положительное, так и отрицательное).
    - Массив (из 4 положительных целых чисел, от 0 до 9)
    - Текст (из следующих символов "0123456789 abcdefghijlnopstu .,:;*-_")

    - ПАРАМЕТРЫ_ВЫВОДА_ЧИСЛА - допускается указывать от 0 до 5 параметров:
        - LEN1, LEN2, LEN3 или LEN4 - количество символов в выводимом числе (включая знак минус).
        - POS1, POS2, POS3 или POS4 - позиция от левого края, к которой привязывается выводимое число.
        - LEFT или RIGHT - направление вывода относительно указанной позиции.
        - DEC или HEX - система счисления для выводимого числа.
        - TIME или TEMP - отображение чисел в виде времени или числа в виде температуры.
        - символ типа char - для заполнения им недостающих разрядов
- Возвращаемые значения: Нет.
- Примечание:
    - Параметры указываются только если значением является число (целое или с плавающей точкой).
    - Если первым параметром (после значения) указать число, то оно будет означать количество выводимых разрядов после запятой.
    - Примеры:

```
# Число 12345.6789 будет выведено как 5.67
dispLED.print( 12345.6789 , 2 , POS1 , LEN3 , RIGHT )
```

Вывод без форматирования:

dispLED.print(1 **)** # Вывод целого числа 1.	
dispLED.print(3.45 **)** # Вывод дробного числа 3.45 (по умолчанию выводится 1 знак после запятой)	
dispLED.print(-12 **)** # Вывод отрицательного целого числа -12.	
dispLED.print(-3.45 **)** # Вывод отрицательного дробного числа -3.45	
dispLED.print(250 , HEX **)** # Вывод числа 250 в шестнадцатиричной системе счисления	

dispLED.print(23 , TEMP **)** # Вывод температуры	
dispLED.print(23.6 , TEMP **)** # Вывод температуры	
dispLED.print(4 , 5 , TIME **)** # Вывод времени	
dispLED.print("67" **)** # Вывод текста	
dispLED.print("OFF" **)** # Вывод текста из букв латинского алфавита	
dispLED.print("8:9" **)** # Вывод текста с двоеточием	

Вывод чисел с указанием количества знаков после запятой:

dispLED.print(1 , 2 **)** # Вывод целого числа 1 с 2 знаками после запятой	
dispLED.print(-2 , 1 **)** # Вывод отрицательного целого числа -2 с 1 знаком после запятой	
dispLED.print(3.45 , 2 **)** # Вывод дробного числа 3.45 с 2 знаками после запятой	
dispLED.print(-3.45 , 0 **)** # Вывод отрицательного дробного числа -3.45 без знаков после запятой	

Вывод чисел с указанием их размерности:

dispLED.print(1 , LEN2 **)** # Вывод целого числа 1 в 2 разрядах	
dispLED.print(-1 , LEN3 **)** # Вывод отрицательного целого числа -1 в 3 разрядах	

dispLED.print(2.3 , LEN3 **)** # Вывод дробного числа 2.3 в 3 разрядах	
dispLED.print(-4.567 , 1 , LEN4 **)** # Вывод дробного числа -4.567 в 4 разрядах , с 1 знаком после запятой	
dispLED.print(8901 , LEN1 **)** # Вывод целого числа 8901 в 1 разряде	
dispLED.print(7 , LEN3 , '-' **)** # Вывод числа 7 в 3 разрядах, с заполнением пустых разрядов символом '-'	

Указание направления сдвига чисел:

dispLED.print(1 , RIGHT **)** # Вывод целого числа 1 со сдвигом вправо от старшего разряда	
dispLED.print(12 , RIGHT **)** # Вывод целого числа 12 со сдвигом вправо от старшего разряда	
dispLED.print(12.3 , RIGHT **)** # Вывод дробного числа 12.3 со сдвигом вправо от старшего разряда	
dispLED.print(1 , LEFT **)** # Вывод целого числа 1 со сдвигом влево (по умолчанию) от младшего разряда	
dispLED.print(12 , LEFT **)** # Вывод целого числа 12 со сдвигом влево (по умолчанию) от младшего разряда	
dispLED.print(12.3 , LEFT **)** # Вывод дробного числа 12.3 со сдвигом влево (по умолчанию) от младшего разряда	

Вывод чисел с привязкой к определённой позиции от левого края:

dispLED.print(74 , POS4 **)** # Вывод целого числа 74 со сдвигом влево (по умолчанию) от 4 позиции	

dispLED.print(74 , POS3 **)** # Вывод целого числа 74 со сдвигом влево (по умолчанию) от 3 позиции	
dispLED.print(74 , POS2 **)** # Вывод целого числа 74 со сдвигом влево (по умолчанию) от 2 позиции	
dispLED.print(74 , POS1 **)** # Вывод целого числа 74 со сдвигом влево (по умолчанию) от 1 позиции	
dispLED.print(74 , POS4 , RIGHT **)** # Вывод целого числа 74 со сдвигом вправо от 4 позиции	
dispLED.print(74 , POS3 , RIGHT **)** # Вывод целого числа 74 со сдвигом вправо от 3 позиции	
dispLED.print(74 , POS2 , RIGHT **)** # Вывод целого числа 74 со сдвигом вправо от 2 позиции	
dispLED.print(74 , POS1 , RIGHT **)** # Вывод целого числа 74 со сдвигом вправо от 1 позиции	

Параметры вывода числа можно комбинировать в любой последовательности:

за исключением числа, указывающего количество знаков после запятой, оно должно быть первым после выводимого значения

dispLED.print(12345.6789 , 2 , POS1 , LEN3 , RIGHT **);** # Вывод дробного числа 12345.6789 # выводить 2 знака после запятой # POS1 - число вывести на дисплей начиная с 1 позиции от левого края # LEN3 - размерность выводимого числа 3 разряда # RIGHT - число сдвигать вправо от указанной позиции.	

Дополнительные функции библиотеки:

В библиотеке реализованы две дополнительные функции, применение которых может быть интересно в познавательных целях.

Функция fps()

- Назначение: Установка частоты обновления всего изображения индикатора.
- Синтаксис: fps(ЧАСТОТА)
- Параметр:
  - ЧАСТОТА - значение от 1 до 255 Гц.
- Возвращаемые значения: Нет.
- Примечание:
  - Светодиоды индикатора светятся не одновременно, а поочерёдно включаются и выключаются так, что в любой момент времени может светиться только один светодиод.
  - Частота заданная данной функцией аналогична частоте кадровой развёртки дисплея, она определяет сколько раз в секунду обновится всё изображение индикатора.
  - На частотах ниже 30 Гц заметно мерцание, а на частотах ниже 10 Гц можно отследить в каком порядке включаются светодиоды.
- Пример:

```
# Обновлять изображение индикатора всего 2 раза в секунду.
dispLED.fps(2)
```

Функция scheme()

- Назначение: Установка схемы включения светодиодов.
- Синтаксис: scheme(СХЕМА)
- Параметр:
  - СХЕМА - может принимать одно из двух значений:
    - LED_CA - светодиоды индикатора включены по схеме с общим анодом.
    - LED_CC - светодиоды индикатора включены по схеме с общим катодом.
- Возвращаемые значения: Нет.
- Примечание:
  - Изменение схемы включения светодиодов может быть полезно только при замене индикатора на плате модуля.
- Пример:

```
# Указываем модулю управлять индикатором собранном по схеме с общими анодами.
dispLED.scheme(LED_CA)
# Указываем модулю управлять индикатором собранном по схеме с общими катодом.
dispLED.scheme(LED_CC)
```

Ссылки:

- Библиотека pyiArduinoI2Cled.
- Расширенные возможности библиотек iarduino для шины I2C.
- Wiki - Установка библиотек Python