

## Начало работы: установка программного обеспечения

Для начала работы с набором “Умная теплица” требуется выполнить установку программного обеспечения на контроллер и на компьютер. Важно отметить, что подойдет компьютер под управлением операционной системы семейства Windows.

Перейдите по [ссылке](#), загрузите и распакуйте архив в удобном для Вас месте, либо скачайте его с флеш накопителя, идущим в комплекте.

Архив имеет следующую структуру:

- ЙоТик - все, относящееся к контроллеру ЙоТик
  - Прошивка - ПО, необходимое для установки на контроллер
- Компьютер - все, относящееся к компьютеру
  - IoTik Studio - установщик среды разработки + примеры
  - Драйверы - драйверы для USB

### Установка Python

Чтобы успешно загрузить прошивку на контроллер потребуется интерпретатор Python 3.8.0. Загрузить его можно с официального сайта [www.python.org](http://www.python.org), либо по ссылке [Python 3.8.0](#). При установке не забудьте поставить галочку в чекбоксе “Add Python 3.8 to PATH”.

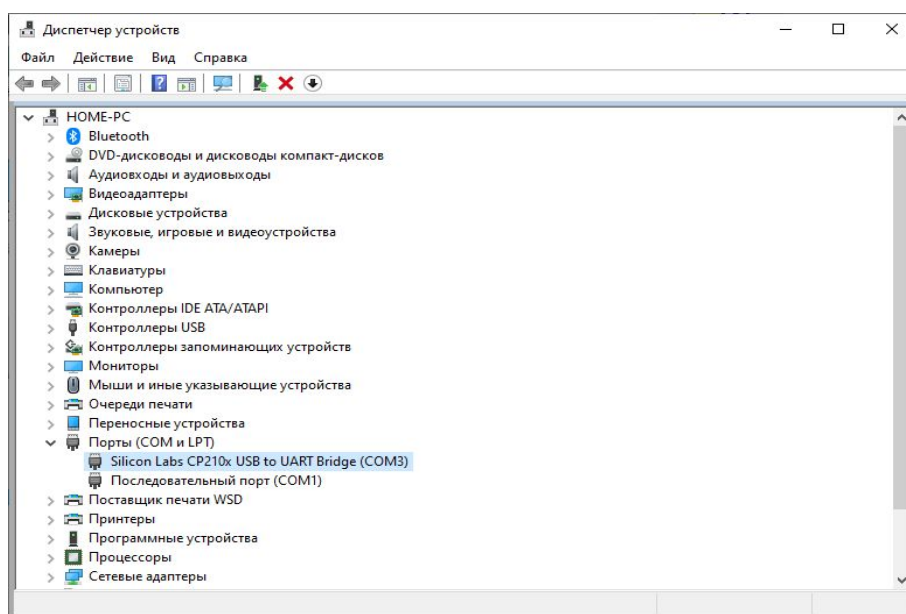


## Установка драйвера

Подключите контроллер к компьютеру с помощью кабеля USB. Откройте диспетчер устройств и разверните раздел “Порты (COM и LPT)”. Откройте папку “Драйвера”, затем распакуйте архив с названием “CP210x Universal Driver.zip” в удобном для Вас месте, перейдите в папку “CP210x\_Universal\_Windows\_Driver”. Если операционная система на Вашем компьютере имеет разрядность 64 бита, то запустите исполняемый файл “CP210xVCPInstaller\_x64.exe”, иначе -- “CP210xVCPInstaller\_x86.exe”. Следуйте дальнейшим указаниям мастера установки.

После окончания установки драйвера в диспетчере устройств в разделе “Порты (COM и LPT)” должно появиться новое устройство.

Запомните номер порта данного устройства (в данном примере COM3).

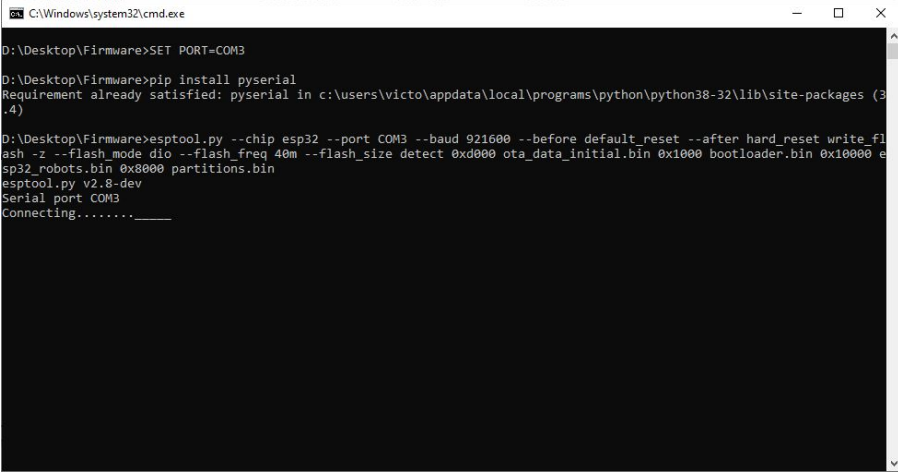


Если устройство не появилось, то установите второй драйвер из папки “Pololu -- CP2102”, аналогично выбирая установочные файлы в зависимости от разрядности операционной системы.

## **Установка IoTik Studio**

Откройте исполняемый файл “IoTik Studio Installer.exe” и следуйте указаниям мастера установки.

## **Загрузка прошивки на контроллер**



```
C:\Windows\system32\cmd.exe
D:\Desktop\Firmware>SET PORT=COM3
D:\Desktop\Firmware>pip install pyserial
Requirement already satisfied: pyserial in c:\users\victo\appdata\local\programs\python\python38-32\lib\site-packages (3.4)
D:\Desktop\Firmware>esptool.py --chip esp32 --port COM3 --baud 921600 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_size detect 0xd000 ota_data_initial.bin 0x1000 bootloader.bin 0x10000 esp32_robots.bin 0x8000 partitions.bin
esptool.py v2.8-dev
Serial port COM3
connecting.....
```

Распакуйте архив с прошивкой “Firmware.zip” в удобном для Вас месте, нажмите правой кнопкой мыши по файлу “flash.bat”, затем выберите “изменить”. Если появилось предупреждение защитника Windows, то нажмите “Подробнее”, а затем “Выполнить в любом случае”. Поменяйте значение порта на значение, которое отображается у нового устройства в диспетчере устройств (в данном примере первая строка в файле будет выглядеть так: SET PORT=COM3). Сохраните и закройте скрипт. Запустите его двойным щелчком. Откроется терминал и начнется загрузка прошивки.

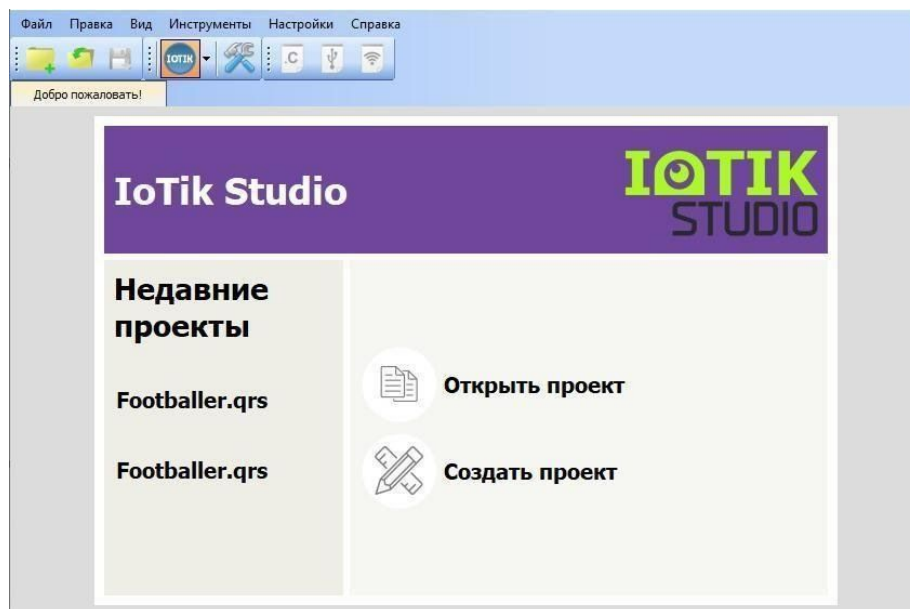
Для подтверждения загрузки нажмите кнопку flash на контроллере (ближайшую к USB порту).

Таким образом, настройка программного окружения завершена, контроллер и компьютер готовы к дальнейшей работе.

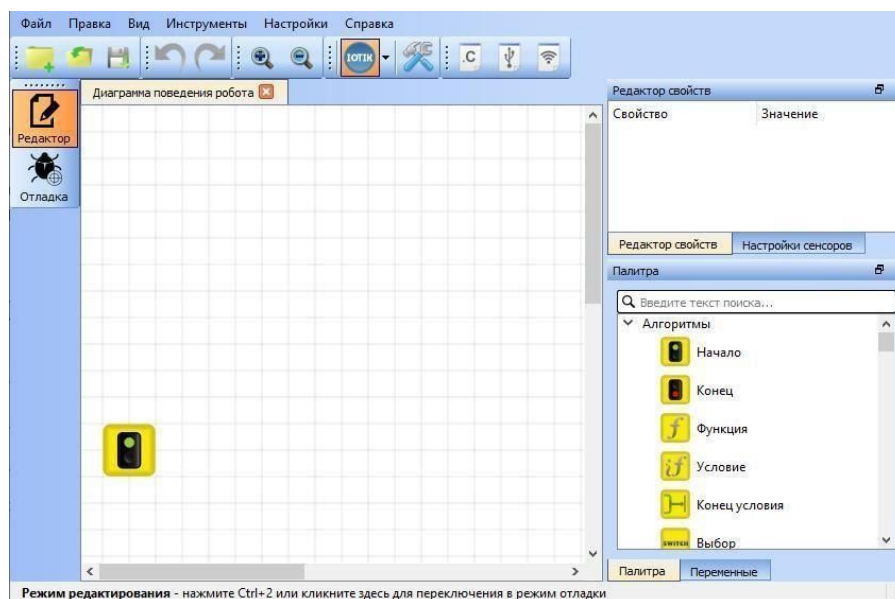
## Начало работы: IoTik Studio + IoTik

IoTik Studio -- среда программирования роботов на базе контроллеров IoTik. Ее основная особенность -- возможность быстро и просто создавать программы с помощью визуальных блоков. Разберемся, как работать в данной среде.

При запуске IoTik Studio открывается главное окно, где предлагается открыть недавно созданный проект, либо создать новый.



В любом из этих случаев откроется окно с рабочим пространством.



## **Основные принципы составления диаграмм**

Чтобы поместить визуальный блок на поле, нужно навести курсор мыши на необходимый блок, нажать левую кнопку мыши, и, удерживая кнопку, переместить курсор на поле, а затем отпустить кнопку (такой способ перемещения объектов называется «drag-n-drop»).

Для соединения двух визуальных блоков необходимо выделить блок (поместить курсор мыши на блок и нажать левую кнопку мыши), откуда будет выходить стрелка, а затем потянуть за голубой круг справа от блока (зажать левую кнопку мыши и вести курсор) и отпустить левую кнопку мыши на том блоке, в который будет входит стрелка.

Для редактирования свойств визуального элемента требуется выделить блок и заменить необходимое свойство в редакторе свойств. Например, таким образом можно поменять свойство стрелок на “ложь”, “истина”, “тело цикла”, либо поменять порт устройства.

Для удаления визуального элемента с поля необходимо навести курсор мыши на блок, нажать правую кнопку мыши и выбрать «удалить».

### **Загрузка программы на контроллер: USB**

Подключите контроллер к компьютеру с помощью кабеля USB, узнайте COM-порт, соответствующий контроллеру. В главном меню IoTik Studio выберите пункт «Настройки», в выпадающем списке – «Настройки», в списке слева – «Роботы». В настройках USB укажите COM-порт, соответствующий контроллеру.

Нажмите “Применить”. С этого момента появляется возможность использовать кнопку “Загрузка по USB”.

**Примечание:** ввиду технических особенностей загрузка программы в контроллер с помощью USB не является надежной, поэтому, если контроллер не работает ожидаемым образом, то его нужно перезагрузить и попытаться отправить программу заново.

### **Загрузка программы на контроллер: Wi-Fi**

Если Ваш компьютер оснащен сетевой картой, которая поддерживает беспроводную передачу данных, то для отправки программы на контроллер необходимо подключиться к точке доступа с именем “iotik-32” и паролем “0123456789”, которую создает контроллер. В главном меню IoTik Studio выберите пункт «Настройки», в выпадающем списке – «Настройки», в списке слева – «Роботы». Убедитесь, что в настройках TCP стоит IP-адрес 192.168.4.1. Нажмите “Применить”. С этого момента появляется возможность использовать кнопку “Загрузка по Wi-Fi”.

## Встроенный светодиод

**Цель задания:** научиться управлять включением/выключением встроенного светодиода, использовать таймер задержки, циклы, переменные.

**Краткие сведения:** на контроллере ЙоТик32 расположены четыре светодиода:

- LED1 (синий) – активен при наличии питания по USB;
- LED2 (красный) – активен при наличии внешнего питания;
- LED3 (розовый) – активируется *программно*, то есть можно составить такую *программу*, которая управляет состоянием данного светодиода;
- LED4 (зеленый) должен быть включен всегда и означает, что к «мозгу» контроллера ЙоТик, модулю ESP32, поступает

питание. (тут картинка ЙоТика с подписанными светодиодами)

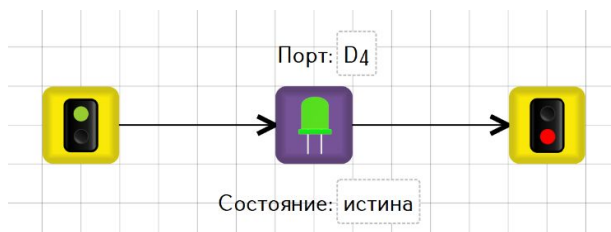
**Таймер** в общепринятом понимании – это устройство, которое подает сигнал через заданный промежуток времени. При составлении программ как правило используется *таймер задержки* – таймер, который по прошествии заданного промежутка времени подает сигнал, разрешающий продолжить выполнение программы.

**Цикл** в программировании – конструкция, которая позволяет *многократно* выполнять указанные инструкции.

**Переменная** – как правило символьный атрибут, обозначающий какое-либо число, которое может *меняться* в ходе исполнения программы.

### Часть 1

Предлагается построить диаграмму, которая включает светодиод. Составьте следующую диаграмму и загрузите программу на контроллер.

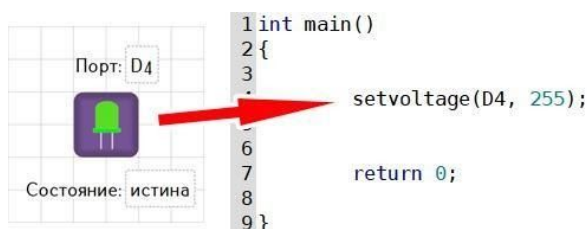


**Примечание:** на контроллере IoTik 32A светодиод ассоциирован с портом D4, если Вы используете контроллер IoTik 32B, то укажите порт D18.

**Ожидаемый результат:** светодиод LED3 (розовый) загорится.

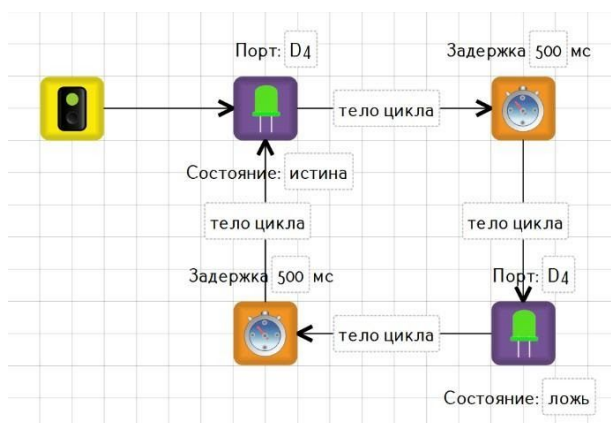
### Код на текстовом языке

Любая корректная диаграмма на графическом языке может быть транслирована в текстовый язык PyСи. Блок «Начало» можно интерпретировать как «цел главная()» и открывающаяся фигурная скобка, а блок «Конец» -- как «возврат 0» и закрывающуюся фигурную скобку. Все, что находится между этими двумя блоками будет заключено и между соответствующим текстовым представлением в текстовом языке. Код на языке PyСи, сгенерированный по диаграмме будет выглядеть следующим образом.



### Часть 2

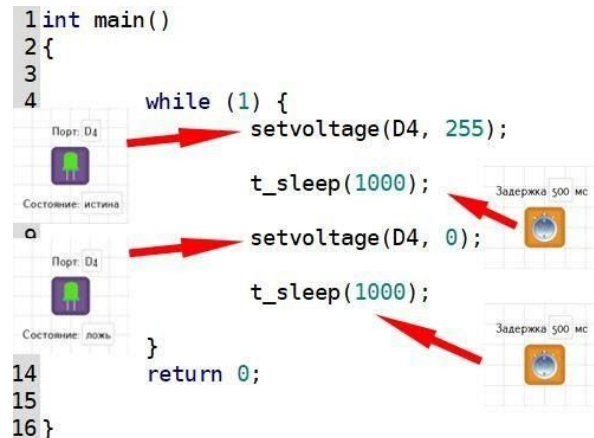
Сейчас последует более сложная задача – бесконечное количество раз попеременно включать/выключать светодиод через заданный промежуток времени (пусть 500 мс.). Расположите и соедините пиктограммы, как показано на рисунке далее, а затем загрузите программу в память контроллера.



**Примечание:** на диаграмме нет блока «конец», так как действия выполняются в так называемом *вечном цикле*.

**Ожидаемый результат:** светодиод LED3 (розовый) будет бесконечно включаться и выключаться на пол секунды.

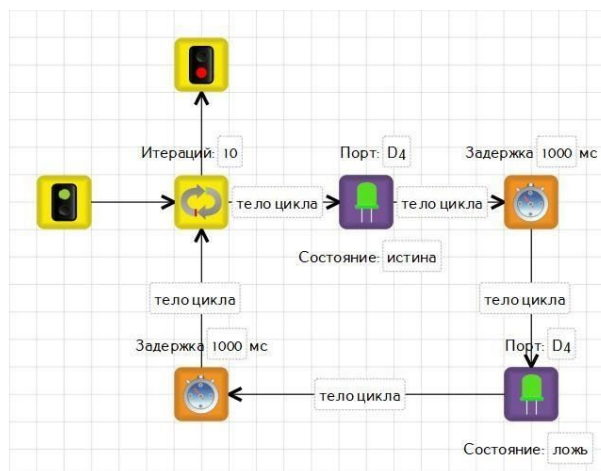
## Код на текстовом языке



На изображении показано соответствие визуальных блоков коду на текстовом языке. Цикл из стрелок на диаграмме транслируется в бесконечный цикл «while», а его содержимое заключается между фигурных скобок. В данной диаграмме нет блока «Конец», так как не предполагается завершение программы, однако в текстовом языке всегда будет «return 0» и закрывающаяся скобка, так как иначе код будет считаться некорректным.

## Часть 3

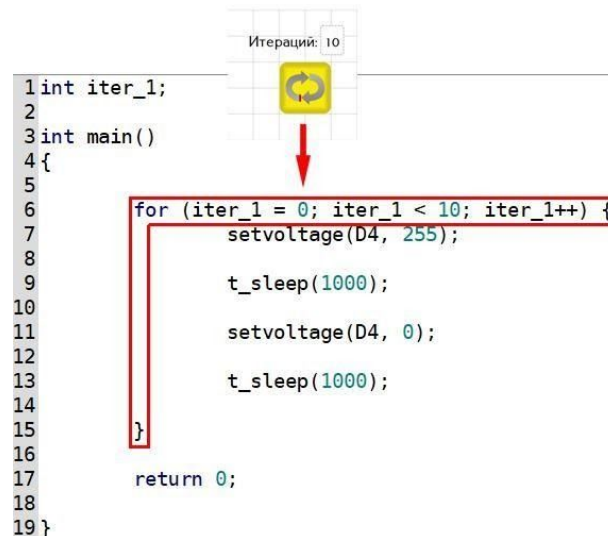
Как правило инструкции есть смысл повторять какое-то определенное количество раз, бесконечный цикл в таком случае не подходит. Предлагается научиться попеременно включать/выключать светодиод 10 раз. Расположите и соедините пиктограммы, как на рисунке далее.



**Примечание:** так как количество повторений конечно, то у программы есть конец, а значит должен присутствовать блок «конец».

**Ожидаемый результат:** светодиод LED3 (розовый) активируется и погаснет 10 раз с интервалом пол секунды.

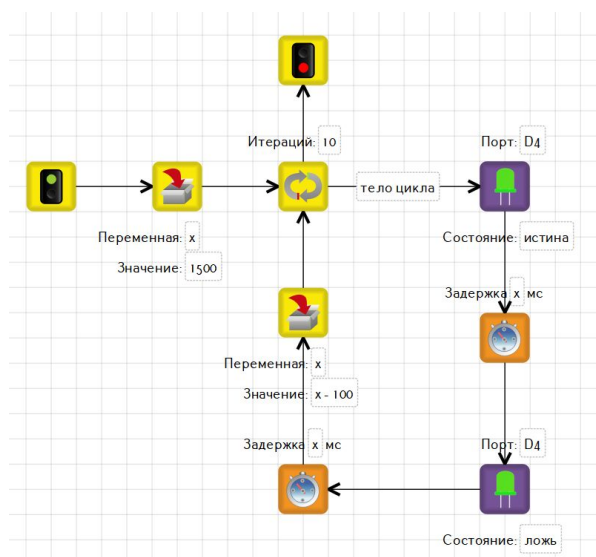
## Код на текстовом языке



Данный код отличается от предыдущего тем, что используется не бесконечный цикл «while», а цикл «for», который отработает 10 итераций. Данная конструкция получается из визуального блока «Цикл», если из этого блока выходит и входит цикл из стрелок.

## Часть 4

В процессе составления программы довольно часто появляется необходимость запомнить какое-либо значение для того, чтобы использовать его потом. Пример: когда мы измеряем площадь прямоугольника, мы запоминаем его длину, ширину, и только потом производим операцию умножения. Для этих целей в программировании используется переменная. Рассмотрим следующую задачу: требуется попеременно включать/выключать светодиод, при этом уменьшая время задержки. Расположите и соедините пиктограммы, как на рисункне далее.



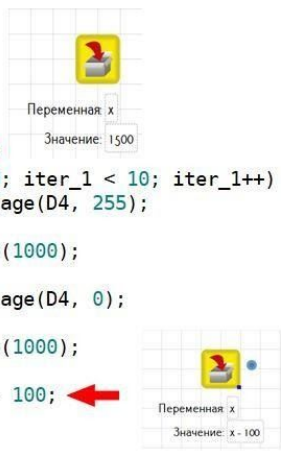
Данная диаграмма требует некоторых пояснений. В самом начале инициализируется переменная «х» значением равным 1500, которая отвечает за задержку. Далее в теле цикла происходит следующее: включается светодиод, задержка х миллисекунд, выключается светодиод, задержка х миллисекунд, значение переменной х уменьшается на 100. Таким образом, в следующей итерации цикла задержка будет на 100 миллисекунд меньше, чем на текущей итерации, следовательно включение/выключение светодиода происходит чаще. Таких итераций десять.

**Примечание:** предполагается, что задержка – положительное числовое значение, то есть переменная «х», которая передается в таймер, не должна принимать значения 0, (-100), (-50) и тд. В противном случае знак числа игнорируется и используется абсолютное значение, что не всегда соответствует ожиданиям составителя программы. В теле цикла значение переменной «х» каждую итерацию уменьшается на 100, то есть через 15 итераций значение переменной «х» будет равно нулю, а через 16 – (-100). Такие моменты нужно принимать во внимание.

**Ожидаемый результат:** светодиод LED3 (розовый) активируется и погаснет 10 раз, при этом с течением времени интервал будет уменьшаться.

### Код на текстовом языке

```
1 int iter_1;
2 int x;
3
4 int main()
5 {
6
7     x = 1500;
8     for (iter_1 = 0; iter_1 < 10; iter_1++) {
9         setvoltage(D4, 255);
10
11         t_sleep(1000);
12
13         setvoltage(D4, 0);
14
15         t_sleep(1000);
16         x = x - 100;
17     }
18
19     return 0;
20
21
22 }
```



The diagram illustrates the state of the variable `x` at two points in the program. The first block shows the initialization of `x` to 1500. The second block shows the value of `x` after the first iteration of the loop, where it has been decremented to 1400. Red arrows point from the code lines `x = 1500;` and `x = x - 100;` to their respective variable monitors.

## **Задания для самостоятельного выполнения**

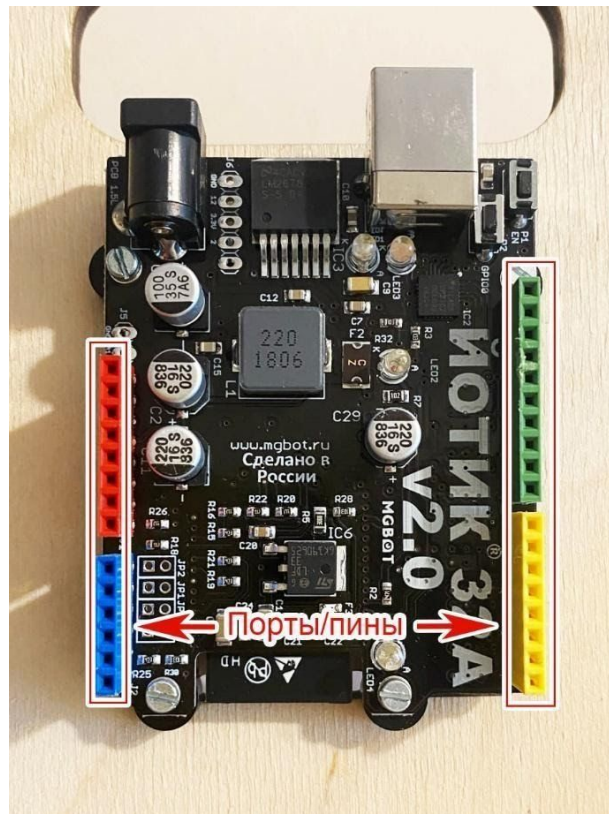
- 1. Сигнал SOS:** составьте такую программу, чтобы контроллер с помощью светодиода в вечном цикле передавал международный сигнал бедствия -- три коротких сигнала, три длинных, а затем снова три коротких. Рекомендуемые задержки: короткий интервал/короткая продолжительность - 200мс, длинный интервал/длинная продолжительность -- 800мс.
- 2. Номер итерации:** составьте такую программу, чтобы контроллер мигал светодиодом количество раз, равное номеру итерации цикла. Отделите сигналы о количестве итераций большим интервалом, например 1000мс.

**Внешние устройства: помпа, вентилятор, светодиодная матрица, угловой сервомотор**

**Цель задания:** научиться подключать внешние устройства к контроллеру IoTik и управлять ими программно.

### **Необходимые сведения**

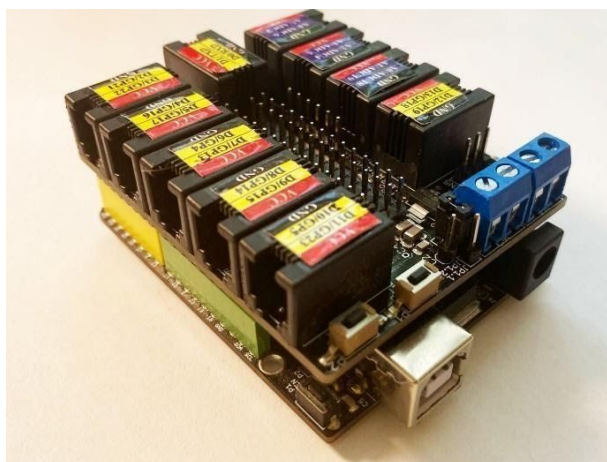
Контроллер ЙоТик32 на борту имеет *порты* (или *пины*), специальные разъемы, предназначенные для подключения внешних устройств.



Для удобной организации подключения внешних устройств используется *шилд*, специальная плата, которая позволяет подключать внешние устройства с помощью коннекторов RJ-9.



Такая плата вставляется ножками в порты контроллера до упора. Контроллер с установленным шилдом представлен на изображении.

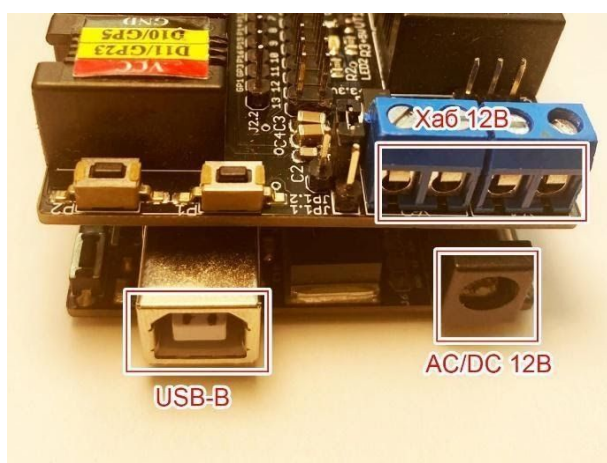


Подключаемые к контроллеру внешние устройства делятся на три основные категории:

- аналоговые;
- цифровые;
- I2C.

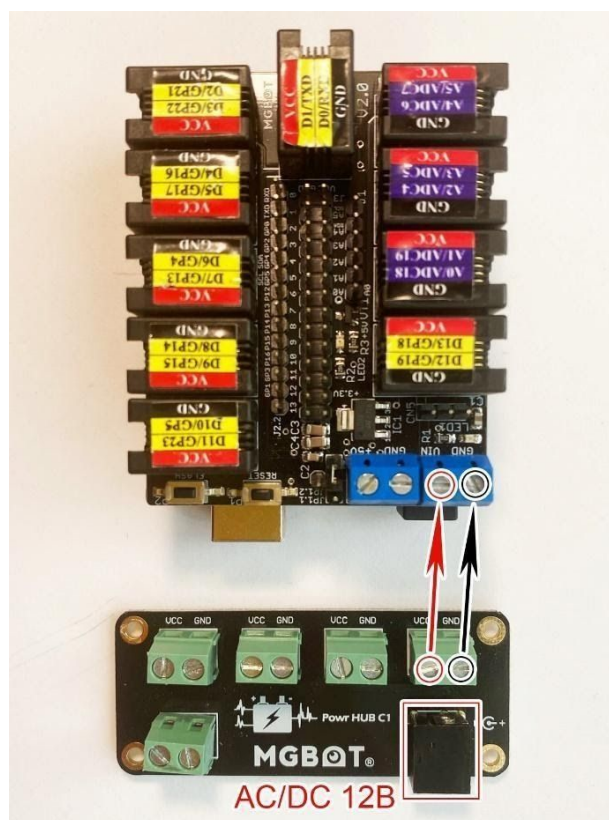
Для корректной работы аналоговые устройства рекомендуется подсоединять к синим разъемам на шилде. Цифровые и I2C устройства корректно работают, будучи подключенными в любой разъем.

Питание контроллера может быть осуществлено через USB-B или с помощью блока питания AC/DC 12В.



Контроллер имеет возможность подавать устройствам напряжение 3В или 5В. Бывают ситуации, когда нужно внешнее устройство требует большее напряжение. В таких случаях используется плата Power HUB C1. К ней подключается блок питания AC/DC 12В, а данная плата в свою очередь подключается проводами к шилду, таким образом питая контроллер. Для питания контроллера клеммы GND и VCC на одной из пяти колодок Power HUB C1 нужно соединить с клеммой GND и VIN на

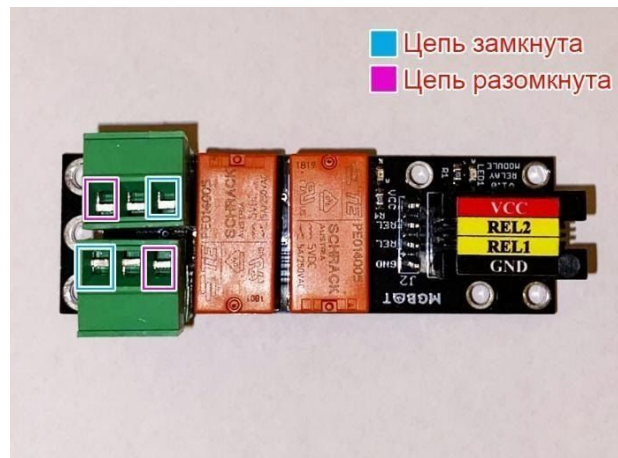
одной из двух колодок шилда соответственно. Для соединения клемм с маркировкой GND принято использовать провод темного цвета. Такое правило позволяет легче ориентироваться в проводах, когда их много.



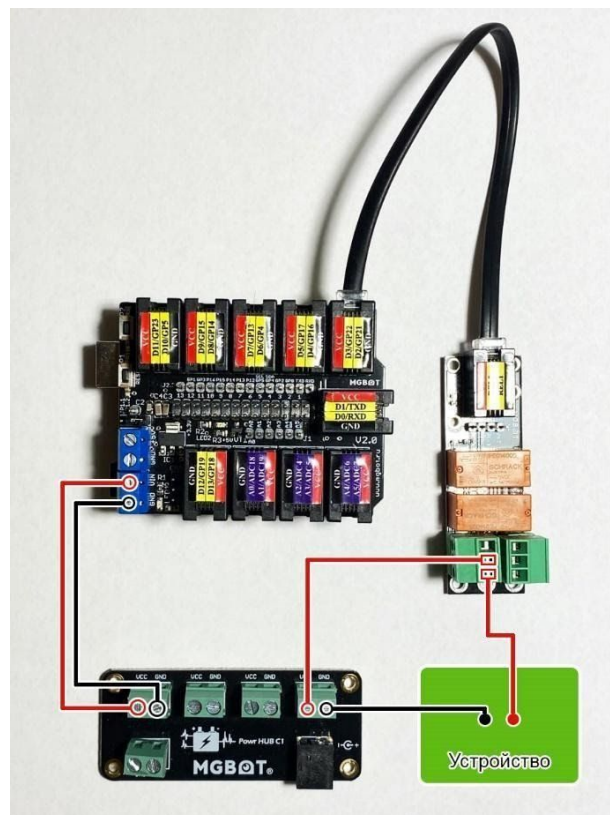
В наборе «Умная теплица» присутствуют помпа и вентилятор. Эти устройства требуют напряжения 12В для работы. Их можно подключить к одной из пар клемм платы Power HUB и устройства заработают, но их нельзя будет остановить, не прекратив питание всей системы. Для управления внешними устройствами, которые требуют большее напряжение, чем способен давать контроллер, используются *реле*.

Реле – это механизм, который позволяет замкнуть электрическую цепь по команде контроллера. В наборе присутствует модуль с двумя реле. К колодке подключается участок электрической цепи, которым нужно управлять. В зависимости от того, в какие клеммы подсоединены провода, цепь будет разомкнута либо замкнута по умолчанию (то есть без подключения модуля реле к контроллеру). Один из концов цепи в обязательном порядке должен быть подсоединен в среднюю клемму.

На изображении показано, куда нужно подключить второй конец провода, чтобы замкнуть или разомкнуть цепь по умолчанию.

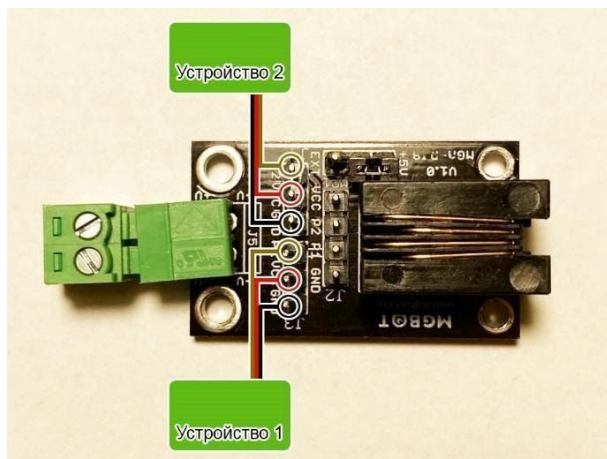


В конечном итоге схема подключения внешнего устройства, управляемого с помощью реле, должна выглядеть похожим образом.



Рекомендуется подключать цепь к реле таким образом, чтобы она была разомкнута по умолчанию, так как подачу сигнала на реле проще ассоциировать с включением устройства, чем с выключением.

В наборе есть светодиодная матрица и угловой сервомотор, которые не могут быть подключены напрямую в порт RJ-9 на шилде. По этой причине используется RJ-9 переходник, который позволяет подключить до двух устройств.

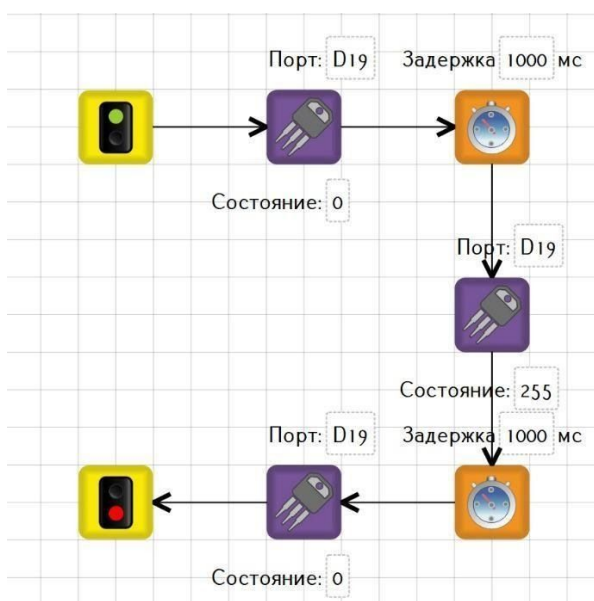


## Часть 1

Научимся управлять работой вентилятора. Подключите вентилятор к модулю с реле, а модуль с реле в одно из свободных гнезд RJ-9 на шилде, пусть в гнездо GP18/GP19. Подсоедините источник питания 12В.

**Примечание:** на некоторые порты контроллера по умолчанию подается напряжение, поэтому возможно непреднамеренное срабатывание реле, и, как следствие, подключенного к нему устройства при подсоединении модуля с реле к некоторым гнездам шилда. Это исправляется с помощью выключения задействованных портов контроллера в начале программы.

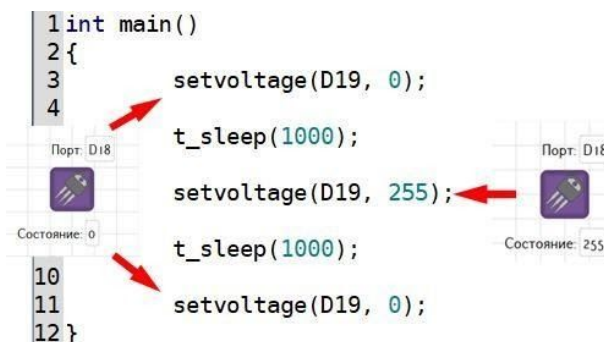
Составьте следующую диаграмму в IoTik Studio.



**Примечание:** на данной диаграмме используется порт D19. Он был определен опытным путем: если после загрузки программы что-то происходит с вентилятором, значит порт верный; если щелкает реле, а с вентилятором не происходит ничего, значит нужен «соседний» порт D18.

**Ожидаемый результат:** вентилятор включится, проработает 1 секунду, а затем выключится.

**Код на текстовом языке:**



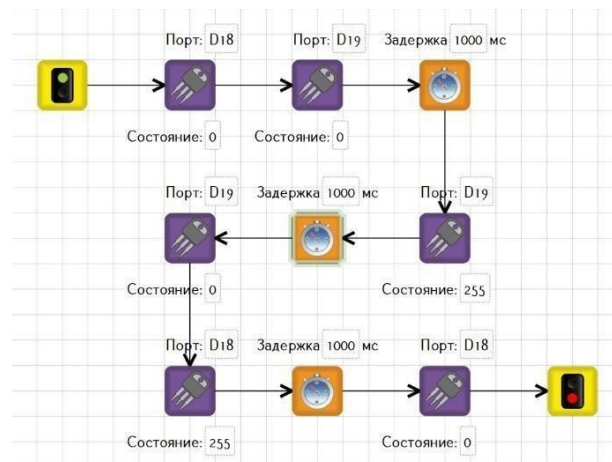
**Примечание:** визуальные блоки «Светодиод» и «Mosfet» транслируются в функцию «setvoltage(port, value)» на текстовом языке РуСи. Функциональное отличие данных блоков заключается в следующем: «Светодиод» позволяет подать в качестве аргумента value только два значения (0 и 255), а «Mosfet» -- диапазон от 0 до 255. Таким образом, эти блоки в каком-то смысле взаимозаменяемы в некоторых случаях, однако не стоит этим пользоваться, так как можно запутаться.

## Часть 2

Предлагается сделать что-то вроде теста системы: проверить работоспособность всех имеющихся устройств. Пусть сначала в течение секунды поработает помпа, а потом вентилятор.

**Примечание:** помпу рекомендуется включать только с опущенными в воду концами шланга. В противном случае помпа может выйти из строя.

Подключите помпу к модулю с реле и составьте следующую диаграмму в IoTik Studio и загрузите на контроллер.



**Ожидаемый результат:** вентилятор и помпа включатся на одну секунду по очереди.

**Код на текстовом языке**

```
1 int main()
2 {
3
4     setvoltage(D18, 0);
5
6     setvoltage(D19, 0);
7
8     t_sleep(1000);
9
10    setvoltage(D19, 255);
11
12    t_sleep(1000);
13
14    setvoltage(D19, 0);
15
16    setvoltage(D18, 255);
17
18    t_sleep(1000);
19
20    setvoltage(D18, 0);
21
22    return 0;
23
24 }
```

### Часть 3

Теперь будем использовать все устройства, работу которых можно наблюдать. Подключите светодиодную матрицу и угловой сервомотор к RJ-9 переходнику. Для однозначности пусть светодиодная матрица – устройство 1, а угловой сервомотор – устройство 2 (см. важные сведения). Подключите переходник к шилду в гнездо GP16/GP17. Тогда светодиодная матрицы будет соответствовать порту GP16, а угловой

Порт: D18      Порт: D19      Задержка 1000 мс

Состояние: 0      Состояние: 0

Порт: D19      Задержка 1000 мс      Порт: D19

Состояние: 0      Состояние: 255

Порт: D18      Задержка 1000 мс      Порт: D18

Состояние: 255      Состояние: 0

Порт: D16      Задержка 1000 мс      Порт: D16      Задержка 1000 мс      Порт: D16

Цвет: белый      Цвет: красный      Цвет: белый

Интенсивность: 0 %      Интенсивность: 100 %      Интенсивность: 100 %

Символ: 'A'      Символ: 'B'      Символ: 'A'

Порты: D17      Задержка 1000 мс      Порты: D17

Угол: 90 °      Угол: -90 °

**Примечание 2:** в свойствах углового сервомотора угол может принимать значение от -90 до 90 градусов.

**Ожидаемый результат:** сначала проработает вентилятор одну секунду, потом помпа, затем матрица выведет литеру «А» белым цветом, через секунду – «В» красным, а еще через секунду выключится, потом сервомотор откроет дверцу теплицы на секунду и закроет ее.

## Код на текстовом языке

```
1 int matrix_1[4];
2 int matrix_2[4];
3 int matrix_3[4];
4
5 int main()
6 {
7
8     setvoltage(D18, 0);
9     setvoltage(D19, 0);
10    t_sleep(1000);
11
12    setvoltage(D19, 255);
13    t_sleep(1000);
14
15    setvoltage(D19, 0);
16    setvoltage(D18, 255);
17    t_sleep(1000);
18
19    setvoltage(D18, 0);
20
21    matrix_1[0] = 'A';
22    matrix_1[1] = 255;
23    matrix_1[2] = 255;
24    matrix_1[3] = 255;
25    setsignal(MATRIX, { D16 }, matrix_1);
26    t_sleep(1000);
27
28    matrix_2[0] = 'B';
29    matrix_2[1] = 255;
30    matrix_2[2] = 0;
31    matrix_2[3] = 0;
32    setsignal(MATRIX, { D16 }, matrix_2);
33    t_sleep(1000);
34
35    matrix_3[0] = 'A';
36    matrix_3[1] = 255;
37    matrix_3[2] = 255;
38    matrix_3[3] = 255;
39    setsignal(MATRIX, { D16 }, matrix_3);
40    setmotor(D17, 90);
41    t_sleep(1000);
42
43    setmotor(D17, -90);
44
45
46    return 0;
47 }
48 }
```

Порт: D16  
Цвет: белый  
Интенсивность: 100 %  
Символ: 'A'

Порт: D16  
Цвет: красный  
Интенсивность: 100 %  
Символ: 'B'

На строках 1-3 объявляются четырехэлементные массивы: которые далее будут переданы в функцию «setsignal()». В нулевом элементе хранится выводимый на матрицу символ, а в первом: втором и третьем

– интенсивность красного, зеленого и синего цветов соответственно.

### Задания для самостоятельного выполнения:

1. Составьте такую диаграмму, чтобы контроллер сначала выводил на матрицу белую букву “F” и включал вентилятор на некоторое время, затем через несколько секунд выводил на матрицу синюю букву “P” и включал помпу на некоторое время, а еще через несколько секунд желтую букву “D” и открывал форточку на некоторое время.

2. Составьте такую диаграмму, чтобы контроллер выводил на матрицу букву “А” всеми доступными цветами через некоторый интервал времени в бесконечном цикле.
3. Составьте такую диаграмму, чтобы контроллер по Вашему расписанию проветривал теплицу, включал свет и поливал растение.

## Внешние устройства: датчики

**Цель задания:** научиться связывать работу датчиков с работой других устройств с помощью условного оператора.

### Необходимые сведения

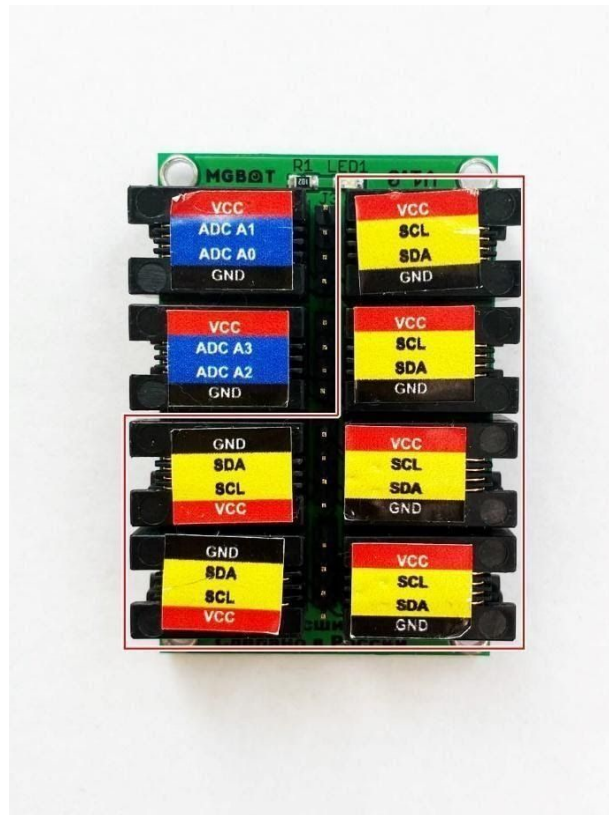
Датчики – это такие внешние устройства, которые выполняют невидимую работу – собирают данные или реагируют на что-либо и посылают определенные сигналы контроллеру. Например: датчик температуры отправляет контроллеру информацию о температуре вблизи датчика; датчик касания посылает сигнал контроллеру, если зафиксировал касание. Основная цель датчиков – предоставлять данные для того, чтобы на них *реагировать*. Так можно организовать автоматический полив, проветривание, освещение и многие другие полезные возможности.

С набором «Умная теплица» поставляются четыре устройства: датчик температуры, влажности воздуха и давления, датчик ультрафиолетового излучения, датчик освещенности и емкостный датчик температуры и влажности почвы.

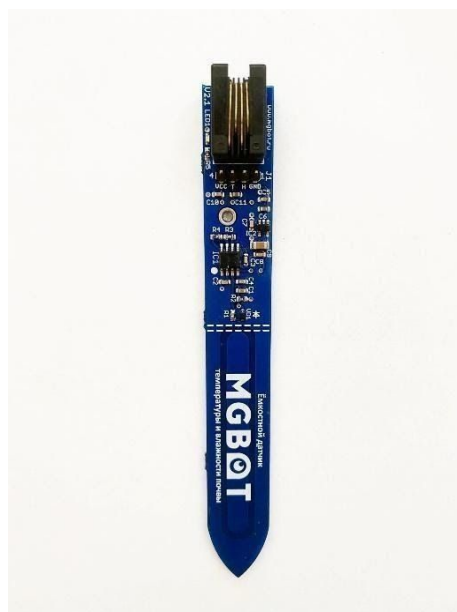
Первые три датчика являются I2C устройствами, то есть могут быть подключены вместе одновременно на два порта контроллера. Внешний вид и отличительные черты датчиков представлены на изображении.



Для удобной организации I2C устройств используется плата расширения I2C. На изображении показан внешний вид платы.



Для корректной работы рекомендуется подключать I2C устройства в разъемы с желтыми наклейками.



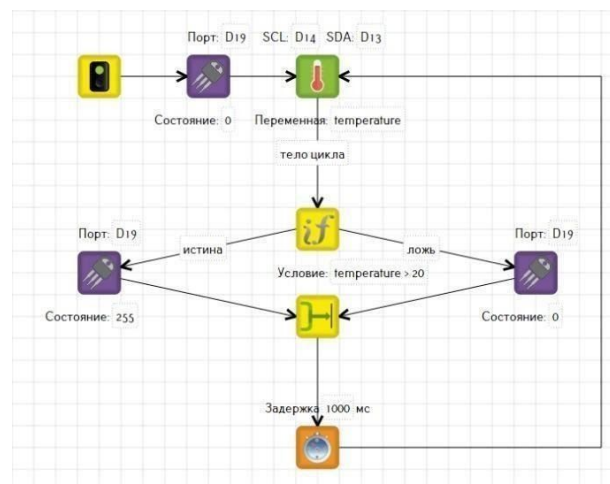
Емкостный датчик температуры и влажности почвы является аналоговым устройством и для корректной работы должен быть подключен в разъемы с синей наклейкой на шилде. Внешний вид емкостного датчика показан на следующем изображении.

## Часть 1

Температура является одним из основных параметров, который необходимо регулировать для комфортного пребывания растения в теплице. Предлагается создать программу для контроллера, которая запускает вентилятор при достижении температуры воздуха, пусть 20 градусов цельсия.

Подключите один провод и датчик температуры, влажности воздуха и давления к плате расширения I2C в любой разъем RJ-9 с желтой наклейкой, а плату расширения соедините свободным концом провода с любым свободным разъемом на шилде, пусть для однозначности с GP13/GP14.

Составьте следующую диаграмму и загрузите программу на контроллер.



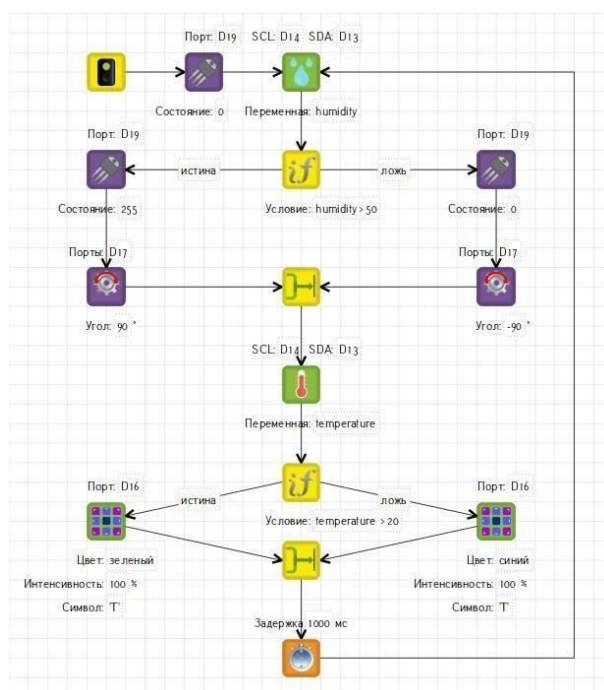
**Ожидаемый результат:** вентилятор включается при температуре строго выше 20 градусов цельсия, иначе выключается.

**Код на текстовом языке**

```
1 int temperature;
2
3 int main()
4 {
5
6     setvoltage(D19, 0);
7
8     while (1) {
9         temperature = getdigsensor(HEATING, { D14, D13 });
10
11         if (temperature > 20) {
12             истина setvoltage(D19, 255);
13         } else {
14             ложь setvoltage(D19, 0);
15         }
16         t_sleep(1000);
17     }
18
19     return 0;
20
21
22
23 }
```

## Часть 2

Работа вентилятора не только снижает температуру, но и иссушает воздух. Проветривание становится более эффективным, если открывать при этом форточку. Предлагается научиться включать вентилятор и открывать форточку, если влажность будет превышать какое-то значение, пусть 50%, а также выводить на светодиодной матрице данные о температуре в таком виде: если температура ниже порогового значения, пусть 20 градусов, то демонстрировать литеру «Т» синего цвета, иначе – зеленого. Составьте следующую диаграмму и загрузите программу на контроллер.




**Ожидаемый результат:** при достижении влажности воздуха 50% вентилятор будет включаться и форточка – открываться, а матрица будет выводить синюю литеру «Т», если температура опустится ниже 20 градусов, иначе – зеленую.

**Примечание:** светодиодную матрицу можно использовать для отображения состояния датчиков, однако численное значение с ее помощью увидеть нельзя.

## Код на текстовом языке

```
1 int humidity;
2 int matrix_1[4];
3 int matrix_2[4];
4 int temperature;
5
6 int main()
7 {
8
9     setvoltage(D19, 0);
10
11     while (1) {
12         humidity = getdigsensor(HUMIDITY, { D14, D13 });
13
14         if (humidity > 50) {
15             setvoltage(D19, 255);
16
17             setmotor(D17, 90);
18
19         } else {
20             setvoltage(D19, 0);
21
22             setmotor(D17, -90);
23
24         }
25         temperature = getdigsensor(HEATING, { D14, D13 });
26
27         if (temperature > 20) {
28             matrix_1[0] = 'T';
29             matrix_1[1] = 0;
30             matrix_1[2] = 255;
31             matrix_1[3] = 0;
32             setsignal(MATRIX, { D16 }, matrix_1);
33
34         } else {
35             matrix_2[0] = 'T';
36             matrix_2[1] = 0;
37             matrix_2[2] = 0;
38             matrix_2[3] = 255;
39             setsignal(MATRIX, { D16 }, matrix_2);
40
41         }
42         t_sleep(1000);
43     }
44     return 0;
45 }
46
47 }
```

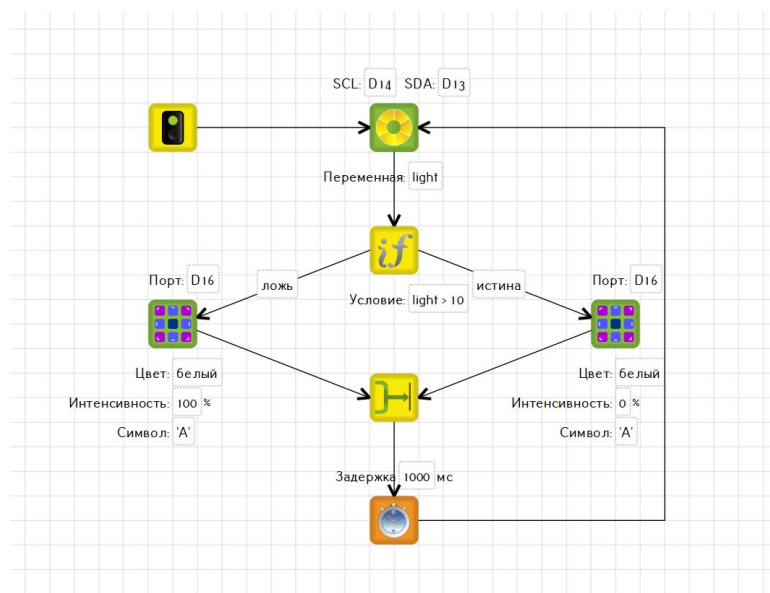


### Часть 3

Теперь научимся включать матрицу, если естественного освещения не хватает. С помощью блока “Матрица” можно выводить не только литеры, а также включать все светодиоды. Для этого в редакторе свойств нужно снять галочку с пункта “Печать”.

Свойство	Значение
Цвет	белый
Интенсивность (%)	0
Порт	D16
Печать	<input type="checkbox"/> ложь
Символ	'А'

Составьте следующую диаграмму и загрузите программу на контроллер.



### Ожидаемый результат:

При недостатке освещения будет включаться матрица. Проверьте это, поместив теплицу в условия недостаточного освещения (Например в шкаф).

### Код на текстовом языке:

```

1 int light;
2 int matrix_1[4];
3 int matrix_2[4];
4
5 int main()
6 {
7
8     while (1) {
9         light = getdigsensor(LIGHT, { D14, D13 });
10
11         if (light > 10) {
12             matrix_1[0] = '\0';
13             matrix_1[1] = 0;
14             matrix_1[2] = 0;
15             matrix_1[3] = 0;
16             setsignal(MATRIX, { D16 }, matrix_1);
17
18         } else {
19             matrix_2[0] = '\0';
20             matrix_2[1] = 255;
21             matrix_2[2] = 255;
22             matrix_2[3] = 255;
23             setsignal(MATRIX, { D16 }, matrix_2);
24
25         }
26         t_sleep(1000);
27     }
28     return 0;
29 }
30
31 }
  
```

### Задания для самостоятельного выполнения:

1. Составьте такую диаграмму, чтобы контроллер в бесконечном цикле с интервалом 1000мс выводил литеру "Т", окрашенную в синий, либо красный цвета, в зависимости от значения температуры воздуха.

2. Составьте такую диаграмму, чтоб контроллер помимо информации о значении температуры воздуха также выводил на матрицу и информацию о давлении и влажности аналогично предыдущему заданию.
3. Составьте такую диаграмму, чтобы матрица передавала информацию о значении любого из датчиков с помощью трех цветов (то есть используя два пороговых значения).

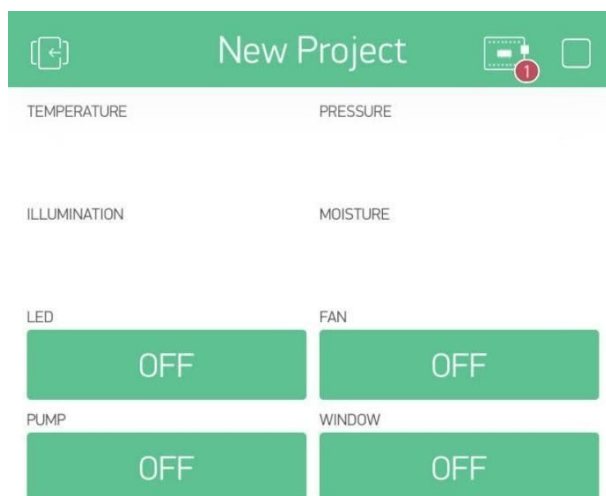
## Blynk: мониторинг и дистанционное управление

**Цель задания:** научиться получать численные значения с датчиков и управлять работой теплицы дистанционно.

### Необходимые сведения

Blynk – это облачный сервис, который позволяет создавать пульта управления из мобильных устройств Android или IOS для различных микроконтроллеров.

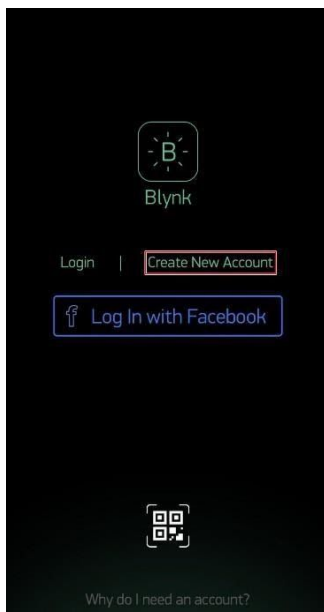
Пример такого пульта приведен на следующем изображении: значения с датчиков температуры, давления, освещенности и влажности отображаются в соответствующих полях, также есть возможность включать и выключать светодиодную матрицу, вентилятор, помпу, открывать форточку.



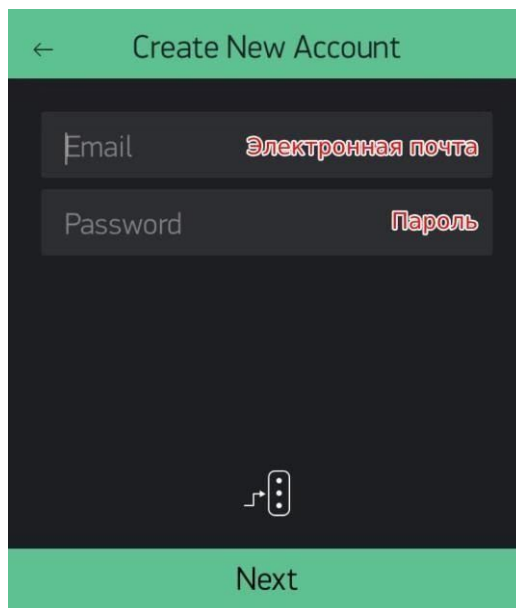
Установите приложение Blynk на свое мобильное устройство. Используйте данную ссылку <http://onelink.to/t4qjsy> или отсканируйте QR- код с помощью мобильного устройства.



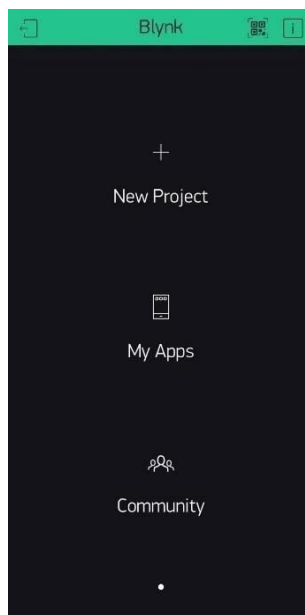
Откройте приложение. Вам будет предложено войти в существующий аккаунт, либо зарегистрировать новый. Далее будет разобрана процедура регистрации нового аккаунта. Выберите “Create New Account”.



Далее необходимо ввести данные будущей учетной записи, электронную почту и пароль, а затем нажать “Next”.

The image displays the 'Create New Account' form within the Blynk application. The form has a green header bar with a back arrow and the title 'Create New Account'. Below the header are two input fields: 'Email' with a red label 'Электронная почта' to its right, and 'Password' with a red label 'Пароль' to its right. At the bottom of the form is a white icon representing a mobile device with a checkmark, indicating successful registration. A green bar at the very bottom contains the word 'Next' in white text.

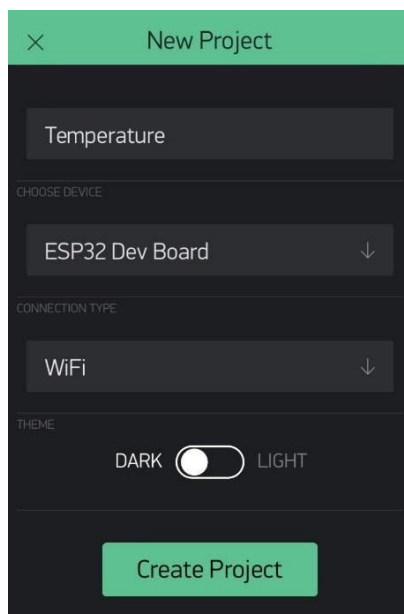
Приложение автоматически войдет в созданный аккаунт, отправив уведомление на электронную почту. Откроется главное меню.



В Blynk есть понятие «Энергия». Это своего рода валюта, за которую можно покупать виджеты – элементы пульта управления. При удалении виджетов с рабочего пространства, энергия возвращается. При создании аккаунта начисляется 2000 единиц энергии, однако, если этого мало, то можно купить QR-код на 20000 энергии по ссылке <https://mgbot.ru/catalog/raznoe/qr-kod-na-20000-energii-dla-prilozhenija-blynk/>. Для сканирования кода нажмите на пиктограмму QR кода в верхнем правом углу экрана.

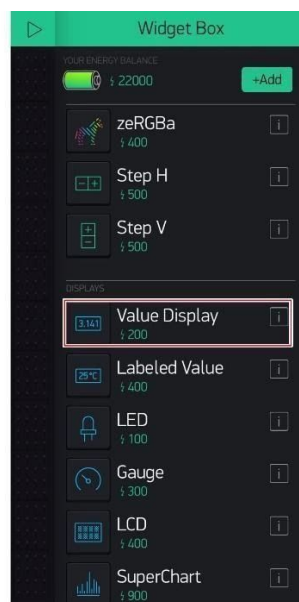
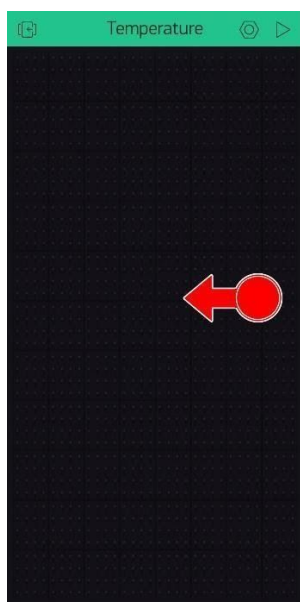
## Часть 1

Научимся считывать показания датчика температуры с помощью Blynk. Откройте приложение Blynk, создайте новый проект, выбрав “New Project”. Назовите проект, выберите в выпадающем списке “Choose Device” ESP32 Dev Board, в “Connection Type” -- WiFi. Затем выберите “Create Project”.

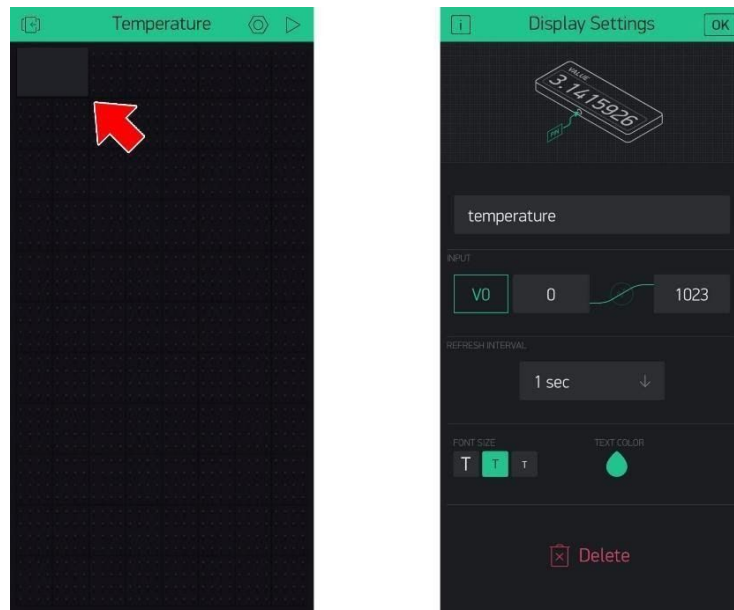


После этого откроется рабочее пространство, а на почту придет токен аутентификации, который связывает мобильное устройство с контроллером, которым оно управляет. Сохраните токен, так как он понадобится при составлении диаграммы в IoTik Studio.

Откройте панель элементов, смахнув экран справа налево и выберите “Value Display”.



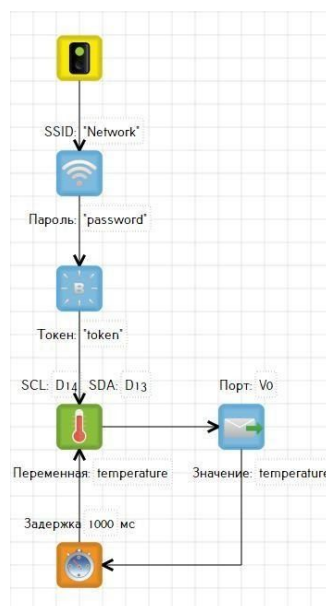
Этот элемент появится на рабочем пространстве. Нажмите на него, чтобы открыть его свойства. Введите желаемое отображаемое название элемента. В секции “Input” нажмите на кнопку “PIN” и выберите “Virtual port” и порт V0. В секции “Refresh Interval” выберите желаемый интервал обновления датчика.



На этом работа с мобильным устройством закончена.

Подключите плату расширения I2C в любой свободный порт на шилде, а к шилду – датчик температуры.

Составьте в IoTik Studio следующую диаграмму, заменяя SSID на имя используемой точки доступа, password – на пароль от точки доступа, а



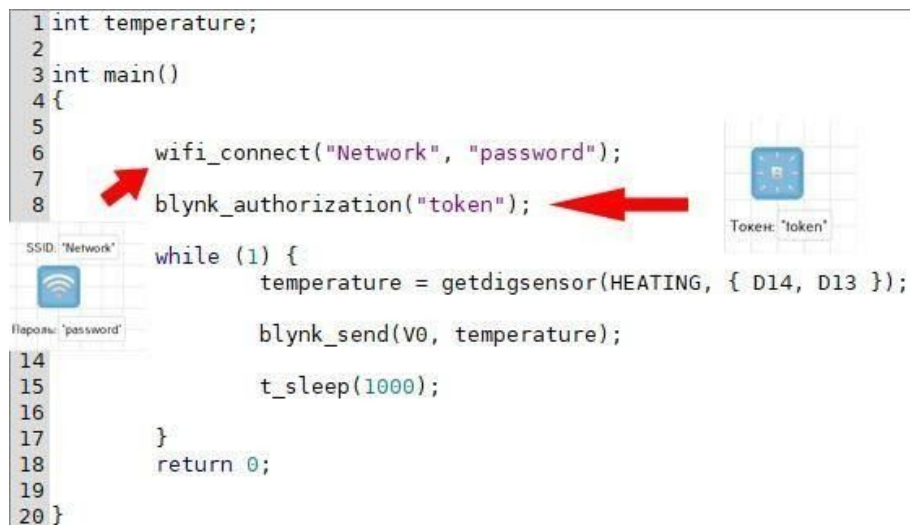
auth token – на токен, который был отослан на электронную почту. Загрузите программу на контроллер.

Подождите 10-20 секунд и нажмите на кнопку «Play» в мобильном приложении Blynk в верхнем правом углу.

**Ожидаемый результат:** в мобильном приложении Blynk отобразится температура воздуха.

### Код на текстовом языке

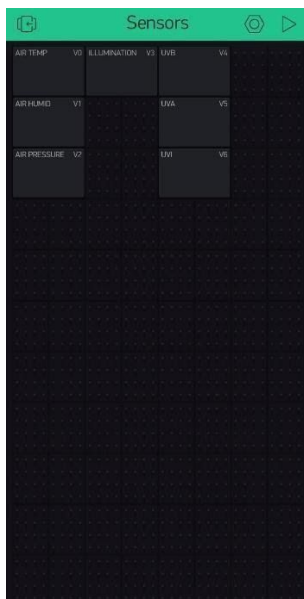
```
1 int temperature;
2
3 int main()
4 {
5
6     wifi_connect("Network", "password");
7     blynk_authorization("token");
8
9     while (1) {
10         temperature = getdigsensor(HEATING, { D14, D13 });
11         blynk_send(V0, temperature);
12         t_sleep(1000);
13     }
14     return 0;
15 }
16
17
18
19
20 }
```



## Часть 2

Теперь подключим все датчики и выведем значения с них на экран мобильного устройства.

Создайте в мобильном приложении Blynk проект, переместите на рабочее пространство элементы «Value Display» и сконфигурируйте их.



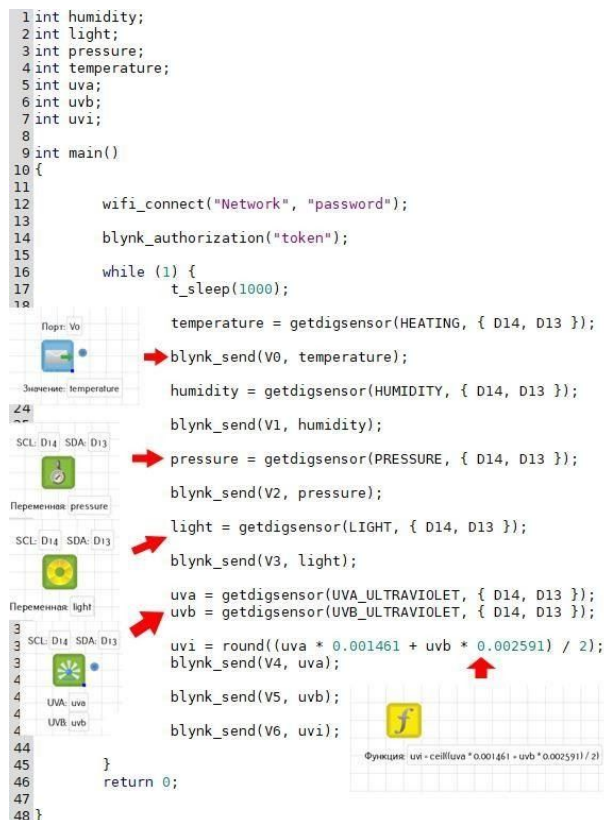
[illegible]

По этой причине используется блок «Функция», который считает коэффициент UVI.

**Ожидаемый результат:** в мобильном приложении Blynk отобразятся значения со всех датчиков.

## Код на текстовом языке

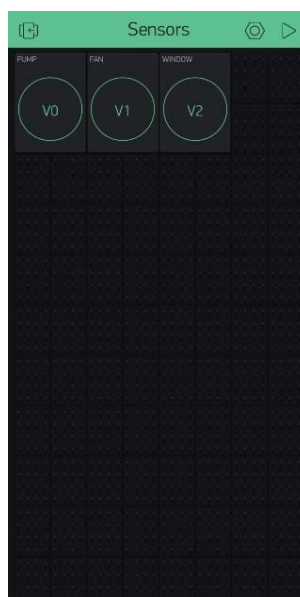
```
1 int humidity;
2 int light;
3 int pressure;
4 int temperature;
5 int uva;
6 int uvb;
7 int uvi;
8
9 int main()
10 {
11     wifi_connect("Network", "password");
12     blynk_authorization("token");
13
14     while (1) {
15         t_sleep(1000);
16
17         temperature = getdigsensor(HEATING, { D14, D13 });
18         blynk_send(V0, temperature);
19
20         humidity = getdigsensor(HUMIDITY, { D14, D13 });
21         blynk_send(V1, humidity);
22
23         pressure = getdigsensor(PRESSURE, { D14, D13 });
24         blynk_send(V2, pressure);
25
26         light = getdigsensor(LIGHT, { D14, D13 });
27         blynk_send(V3, light);
28
29         uva = getdigsensor(UVA_ULTRAVIOLET, { D14, D13 });
30         uvb = getdigsensor(UVB_ULTRAVIOLET, { D14, D13 });
31
32         uvi = round((uva * 0.001461 + uvb * 0.002591) / 2);
33         blynk_send(V4, uva);
34         blynk_send(V5, uvb);
35         blynk_send(V6, uvi);
36     }
37     return 0;
38 }
```



## Часть 3

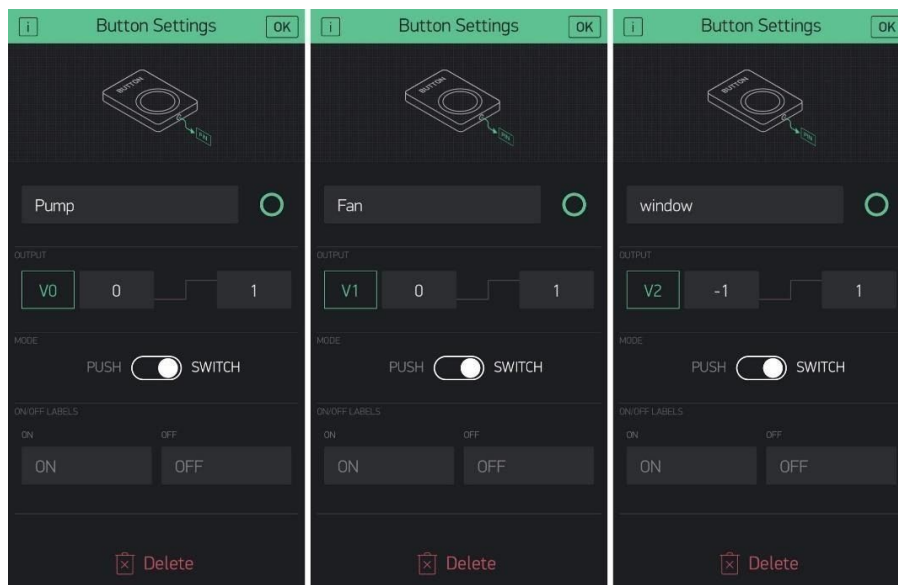
Теперь научимся управлять работой вентилятора, помпы и углового сервомотора с помощью мобильного приложения Blynk.

Создайте новый проект в мобильном приложении Blynk и переместите три элемента “Button” на рабочее пространство.

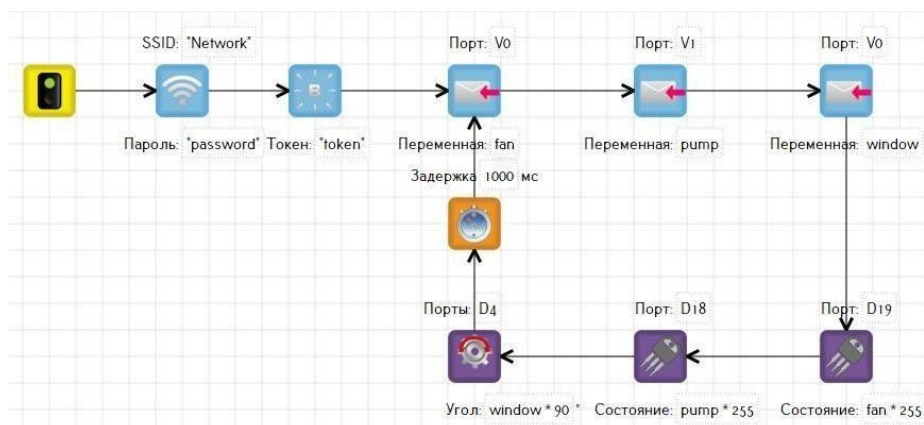


Дайте каждому элементу название, в зависимости от того, какую функциональность он реализует. Стоит помнить, что в IoTik Studio

вентилятор и помпа управляются путем подачи значений 0, либо 255 в визуальный элемент “Mosfet”, а угловой сервомотор – диапазоном значений от -90 до 90 в визуальный элемент “Угловой сервомотор”.



Составьте следующую диаграмму в IoTik Studio, заменяя “Network” и “password” на данные Вашей точки доступа, а “token” -- на токен аутентификации, который пришел на электронную почту при создании проекта в мобильном приложении Blynk.



Подождите 10-20 секунд и нажмите на кнопку «Play» в мобильном приложении Blynk в верхнем правом углу.


**Ожидаемый результат:** при нажатии кнопок в мобильном приложении будут включаться, либо отключаться вентилятор, помпа, а также открываться или закрываться форточка.

## Код на текстовом языке

```
1 int fan;
2 int pump;
3 int window;
4
5 int main()
6 {
7
8     wifi_connect("Network", "password");
9
10    blynk_authorization("token");
11
12    while (1) {
13        fan = blynk_receive(V0);
14        pump = blynk_receive(V1);
15        window = blynk_receive(V2);
16        setvoltage(D19, fan * 255);
17        setvoltage(D18, pump * 255);
18        setmotor(D17, window * 90);
19        t_sleep(1000);
20    }
21    return 0;
22 }
```

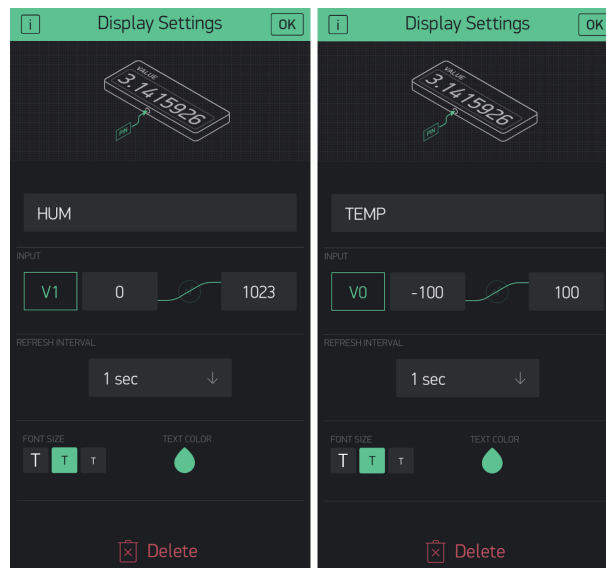
Порт: V0

Переменная: fan



## Часть 4

Теперь научимся измерять температуру и влажность почвы. Создайте проект в мобильном приложении Blynk, переместите туда два виджета “Valued display” и сконфигурируйте их так, чтобы один дисплей получал значения на виртуальный порт V0, а другой -- на V1.



Приготовьте стакан воды, подключите емкостный датчик к любому аналоговому порту, пусть к ADC4/ADC5. Составьте в IoTik Studio следующую диаграмму, заменяя SSID на имя используемой точки доступа, password – на пароль от точки доступа, а auth token – на токен, который был отослан на электронную почту.



Загрузите программу на контроллер.

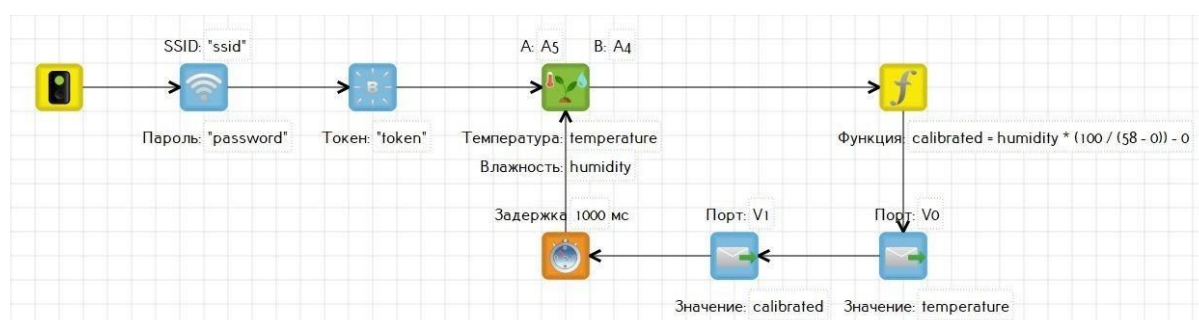
Подождите 10-20 секунд и нажмите на кнопку «Play» в мобильном приложении Blynk в верхнем правом углу.

Дисплеи покажут значения, получаемые с датчиков. На данном этапе нас интересуют значения с датчика влажности. Ввиду его технических особенностей два датчика влажности могут показывать *разные* значения, поэтому этот датчик нужно *калибровать*.

### **Калибровка датчика влажности**

Протрите конец датчика сухой тканью, и запомните значение, которое показывает виджет в Blynk (далее это значение будет называться min). Теперь опустите конец датчика в стакан с водой, подождите до тех пор, пока значение влажности перестанет изменяться и зафиксируйте значение (далее это значение будет называться max). Посчитайте уравнение переноса по формуле:

$$calibrated = 100 / (max - min) * humidity - min$$



Вставьте это уравнение в блок “Формула”, заменяя значения в примере на собственные.

### **Обратите внимание:**

В блоке “Отправить значение” нужно отправлять переменную calibrated.

### **Ожидаемый результат:**

Виджеты в мобильном приложении Blynk будут показывать значения температуры и влажности среды, в которой находится датчик.

### Код на текстовом языке

```
1 float calibrated;
2 int humidity;
3 int temperature;
4
5 int main()
6 {
7
8     wifi_connect("ssid", "password");
9
10    blynk_authorization("token");
11
12    while (1) {
13        temperature = getansensor(TEMPERATURE, A5);
14        humidity = getansensor(SOIL, A4);
15        calibrated = humidity * (100 / (58 - 0)) - 0;
16        blynk_send(V0, temperature);
17
18        blynk_send(V1, calibrated);
19
20        t_sleep(1000);
21    }
22    return 0;
23
24
25 }
```

**Задания для самостоятельного выполнения:**

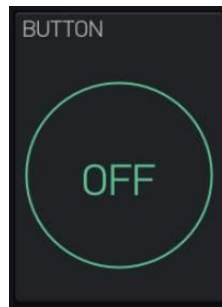
1. Попробуйте отредактировать свойства элемента “Button” в мобильном приложении Blynk и диаграмму в IoTik Studio таким образом, чтобы форточка открывалась на меньшее количество градусов.
2. Попробуйте отрегулировать угол открывания форточки с помощью элемента “Slider”.
3. Попробуйте регулировать яркость матрицы с помощью элемента “Slider”.
4. Попробуйте выводить на дисплей температуру в градусах Фаренгейта. Формула для перевода градусов Цельсия в градусы Фаренгейта:  $temp * 1.8 + 32$ , где  $temp$  - градусы Цельсия.

## Обзор виджетов Blynk

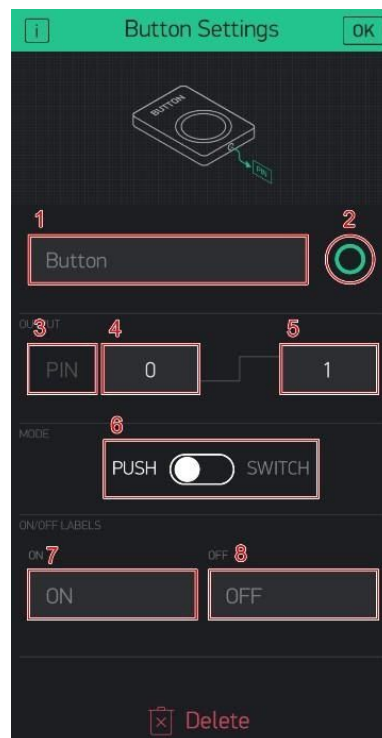
### Управляющие виджеты

#### Кнопка (Button)

С помощью кнопки, как правило, управляют сущностями, которые имеют два состояния. Например, светодиод может быть включен или выключен, форточка – открыта или закрыта, и для управления ими отлично подойдет кнопка.



Рассмотрим настройки кнопки.

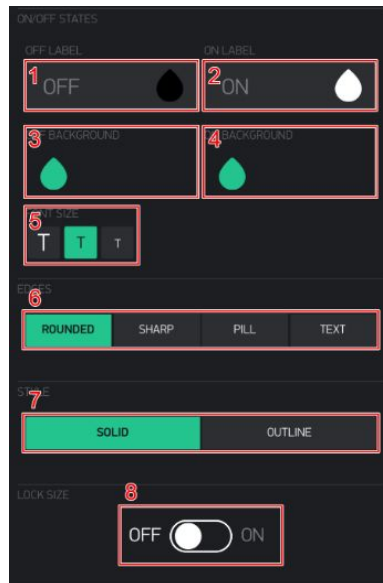


1. Отображаемое название: чтобы было понятно, чем управляет кнопка, имеет смысл ее назвать коротким и содержательным именем. Например, если кнопка включает вентилятор, то ее следует назвать «Вентилятор».
2. Отображаемый цвет: иногда в изобилии кнопок можно запутаться, а искать нужную по названию -- слишком долго. Для этого есть возможность окрашивать кнопки в разные цвета, чтобы быстрее и удобнее ориентироваться.

3. Пин: для того, чтобы связать конкретную кнопку с конкретным действием, нужно указать для этого пин, который будет принимать значения, в зависимости от положения кнопки. В связке с IoTik Studio есть возможность использовать *только* виртуальные пины (virtual pins).
4. Значение при отжатой кнопке: числовое значение, которое отправляется на пин, когда кнопка находится в отжатом положении.
5. Значение прижатой кнопке: числовое значение, которое отправляется на пин, когда кнопка зажата.
6. Позиционность кнопки: по умолчанию кнопка нажата тогда, когда палец касается ее, однако, если хочется, чтобы кнопка стала позиционной, то нужно переместить переключатель в положение «switch».
7. Надпись на кнопке при отжатом положении.
8. Надпись на кнопке прижатом положении.

## Стилизованная кнопка (Styled button)

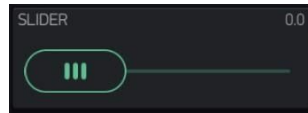
Реализует ту же функциональность, что и обыкновенная кнопка, только имеет больше вариантов оформлений. Рассмотрим список расширенных настроек оформления.



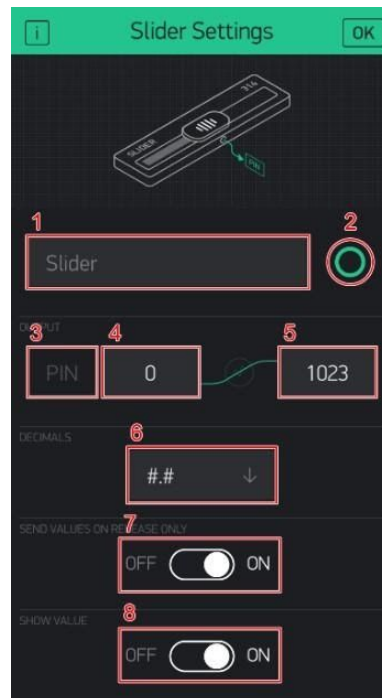
1. Отображаемая надпись и ее цвет при отжатой кнопке.
2. Отображаемая надпись и ее цвет при нажатой кнопке.
3. Цвет фона при отжатой кнопке.
4. Цвет фона при нажатой кнопке.
5. Размер текста.
6. Оформление краев кнопки: слегка скругленные, острые, существенно скругленные, сплошной текст.
7. Стиль кнопки: сплошное заполнение, контур.
8. Подгонка размера кнопки по тексту.

## ***Ползунок/вертикальный ползунок (Slider/vertical slider)***

Ползунок позволяет передавать значения в пределах заданного диапазона.



Рассмотрим настройки ползунка.



1. Отображаемое название.
2. Отображаемый цвет.
3. Пин.
4. Нижняя граница диапазона значений.
5. Верхняя граница диапазона значений.
6. Точность передаваемого значения: # -- целые числа, #.# -- числа с точностью до десятых, #.## -- сотых и тд.
7. Отправка значения при спуске пальца с ползунка/раз в некоторый интервал времени.
8. Показывать/скрывать передаваемое значение в верхнем правом углу ползунка.

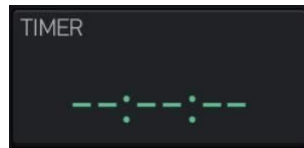
## Использование с IoTik Studio

Кнопка отправляет два значения на указанный порт (в IoTik Studio есть возможность использовать только *виртуальный порт*): по умолчанию 0 если на нее не нажимают и 1 – если нажимают. Со стороны контроллера должна быть возможность эти значения принимать. В IoTik Studio это реализуется с помощью блока «Получить значение».

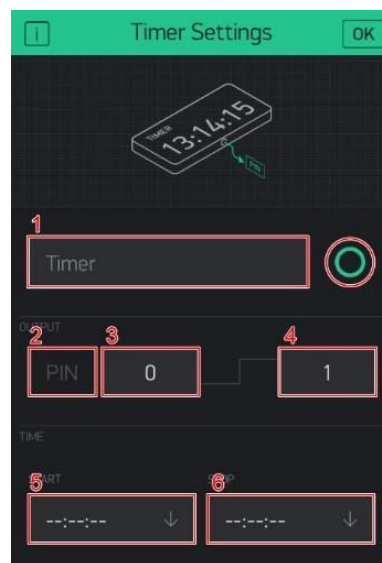


## Таймер (timer)

Таймер запускает и/или прекращает какое-либо действие по наступлению заданного времени, передавая определенное значение до наступления времени «start» и после «stop» (обычно 0), и после времени «start» и до «stop» (обычно 1).



Рассмотрим настройки таймера.



1. Отображаемое имя.
2. Отображаемый цвет.
3. Пин.
4. Передаваемое значение до наступления времени «start» и после «stop».
5. Передаваемое значение после наступления времени «start» и до «stop».
6. Время «start».
7. Время «stop».

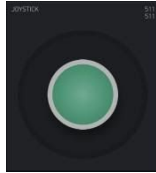
## Использование с IoTik Studio

Таймер отправляет два значения на указанный порт: по умолчанию 1 между временем «start» и «stop» и 0 в остальных случаях. Со стороны контроллера должна быть возможность эти значения принимать. В IoTik Studio это реализуется с помощью блока «Получить значение».

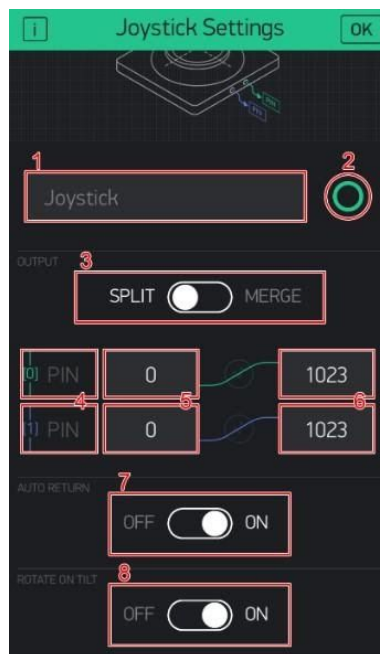


## Джойстик (Joystick)

Джойстик обычно используется для управления движением во всех направлениях. Передает на контроллер два значения: смещение по вертикали и горизонтали.



Рассмотрим настройки джойстика.



1. Отображаемое имя.
2. Отображаемый цвет.
3. Способ отправки данных: каждое значение на определенный пин/массив данных на один пин.
4. Пин.
5. Минимальное передаваемое значение.
6. Максимальное передаваемое значение.
7. Автоматический возврат джойстика в первоначальную позицию.
8. Автоматическая ориентация джойстика при повороте устройства.

## Использование с IoTik Studio

Джойстик отправляет значения положения рукоятки по вертикали и горизонтали в два указанных порта. Со стороны контроллера должна быть возможность эти значения принимать. В IoTik Studio это реализуется с помощью блока «Получить значение».

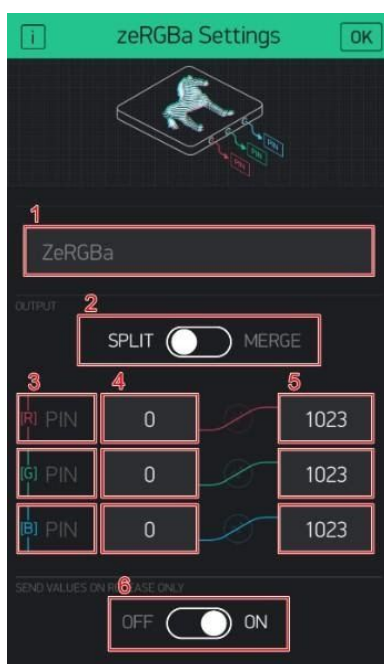


## ZeRGBa

Данный виджет является палитрой цветов. Передает на контроллер три значения: интенсивность красного, зеленого и синего цветов.



Рассмотрим настройки виджета ZeRGBa.



1. Отображаемое название.
2. Способ отправки данных: каждое значение на определенный пин/массив данных на один пин.
3. Пин.
4. Минимальное передаваемое значение.
5. Максимальное передаваемое значение.
6. Момент передачи данных: при отпускании/раз в некоторый интервал.

## Использование с IoTik Studio

ZeRGBa отправляет значения интенсивности красного зеленого и синего цветов в три указанных порта. Со стороны контроллера должна быть возможность эти значения принимать. В IoTik Studio это реализуется с помощью блока «Получить значение».



## **Горизонтальный/вертикальный шаговый управляющий элемент (step H/step V)**

Данный виджет позволяет передавать значения в пределах заданного диапазона пошагово.



Рассмотрим настройки шагового управляющего элемента.



1. Отображаемое название.
2. Отображаемый цвет.
3. Пин.
4. Минимальное передаваемое значение.
5. Максимальное передаваемое значение.
6. Шаг изменения значения.
7. Отправлять на устройство шаг/актуальное значение
8. Зацикливание значений, т.е после достижения максимума отсчет начинается с минимального значения.
9. Отображаемые иконки: стрелки/плюс-минус.

## Использование с IoTik Studio

Шаговый управляющий элемент отправляет значения в указанный порт. Со стороны контроллера должна быть возможность эти значения принимать. В IoTik Studio это реализуется с помощью блока «Получить значение».



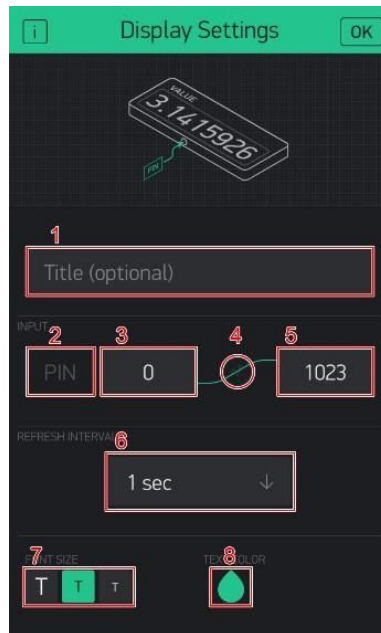
## Дисплеи

### *Дисплей значений (Value display)*

Дисплей значений нужен для отображения значений, передаваемых датчиками.



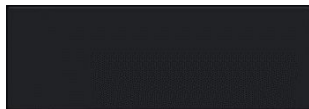
Рассмотрим настройки дисплея значений.



1. Отображаемое название.
2. Пин.
3. Минимальное отображаемое значение.
4. Режим масштабирования данных на диапазон от 0 до 100: пусть датчик передает значения от 0 до 1024, тогда при включении отображения данных дисплей будет переводить область значений датчика в интервал от 0 до 100.
5. Максимальное отображаемое значение.
6. Частота обновления дисплея.
7. Размер текста.
8. Отображаемый цвет.

## Маркированный дисплей значений (Labeled value display)

Данный дисплей аналогичен обычному дисплею значений и имеет дополнительные возможности отображения данных.



Рассмотрим настройки маркированного дисплея значений.



1. Отображаемое название.
2. Пин.
3. Минимальное отображаемое значение.
4. Режим масштабирования данных.
5. Максимальное отображаемое значение.
6. Выравнивание текста.
7. Формат отображения данных: пусть мы хотим, чтобы дисплей выводил температуру в таком формате: «Temp: x degrees». Тогда в данное поле следует написать «Temp /pin/ degrees».
8. Частота обновления дисплея
9. Размер текста.
10. Отображаемый цвет.

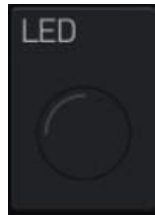
## Использование с IoTik Studio

Дисплей отображает значения, отправленные в указанный порт. Со стороны контроллера должна быть возможность эти значения отправлять. В IoTik Studio это реализуется с помощью блока «Отправить значение».

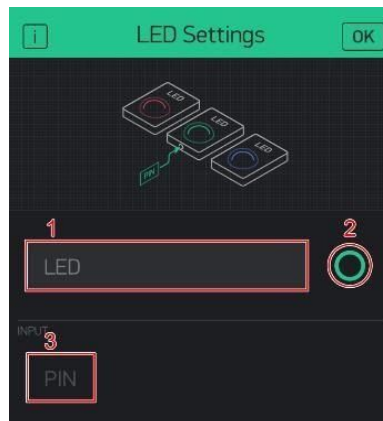


## Светодиод (LED)

Светодиод нужен для индикации наступления какого-либо события, связанного с датчиками. Например, если температура поднялась выше порогового значения.



Рассмотрим настройки светодиода.



1. Отображаемое значение.
2. Отображаемый цвет.
3. Пин.

## Использование с IoTik Studio

Светодиод включается, если на указанный порт отправить значение 255, а отключается – если отправить 0. Со стороны контроллера должна быть возможность отправить эти значения. В IoTik Studio это реализуется с помощью блока «Отправить значение».

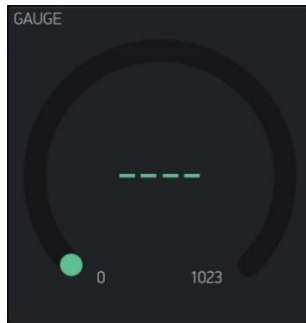


Также в IoTik Studio есть возможность включать/выключать светодиод и менять его цвет с помощью блока «Светодиод» из раздела «Blynk».



## Измерительный прибор (Gauge)

Выполняет ту же роль, что и дисплей, но за счет визуальной интерпретации данные представлены нагляднее.



Рассмотрим настройки измерительного прибора.

1. Отображаемое название.
2. Пин.
3. Минимальное отображаемое значение.
4. Режим масштабирования данных.
5. Максимальное отображаемое значение.
6. Формат отображения данных.
7. Размер текста.
8. Отображаемый цвет.
9. Частота обновления.

## Использование с IoTik Studio

Измерительный прибор отображает значения, отправленные в указанный порт. Со стороны контроллера должна быть возможность эти значения отправлять. В IoTik Studio это реализуется с помощью блока «Отправить значение».

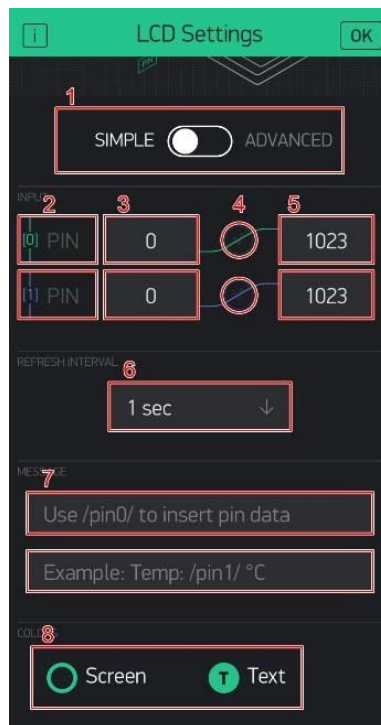


## LCD

Данный виджет позволяет выводить текстовую строку, либо значения с двух датчиков.



Рассмотрим настройки LCD.

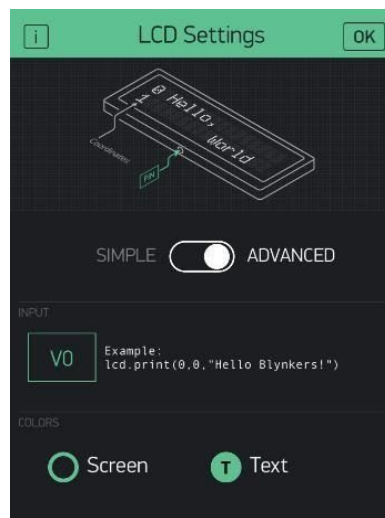


1. Переключение упрощенного и продвинутого режимов конфигурирования.

Далее будут рассмотрены возможности упрощенного режима.

2. Пины.
3. Минимальное отображаемое значение.
4. Режим масштабирования данных.
5. Максимальное отображаемое значение.
6. Частота обновления.
7. Формат отображения данных.
8. Отображаемый цвет экрана и текста.

Далее представлен интерфейс настройки продвинутого режима.



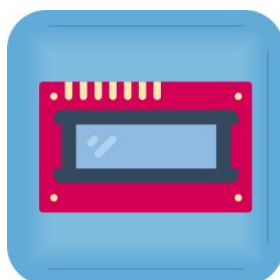
Режим называется продвинутым, так как позволяет гибко конфигурировать отображение текста на дисплее на стороне контроллера с помощью текстового языка C/C++. В нашем случае используется текстовый язык RuC, который такими возможностями не обладает, поэтому данный режим используется для отображения строки в стандартном виде.

### Использование с IoTik Studio

LCD отображает значения, отправленные в два указанных порта. Со стороны контроллера должна быть возможность отправлять эти значения. В IoTik Studio это реализуется с помощью блока «Отправить значение».

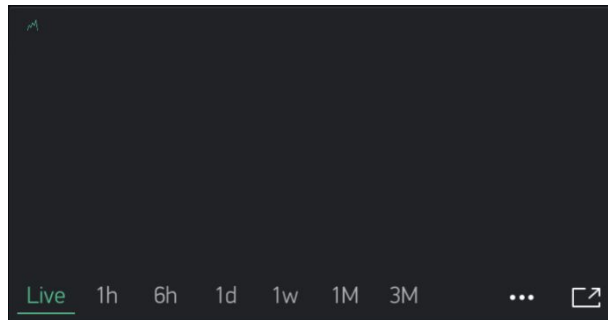


Также с помощью LCD можно отобразить текстовую строку. Для этого используется блок «Дисплей» из раздела Blynk.

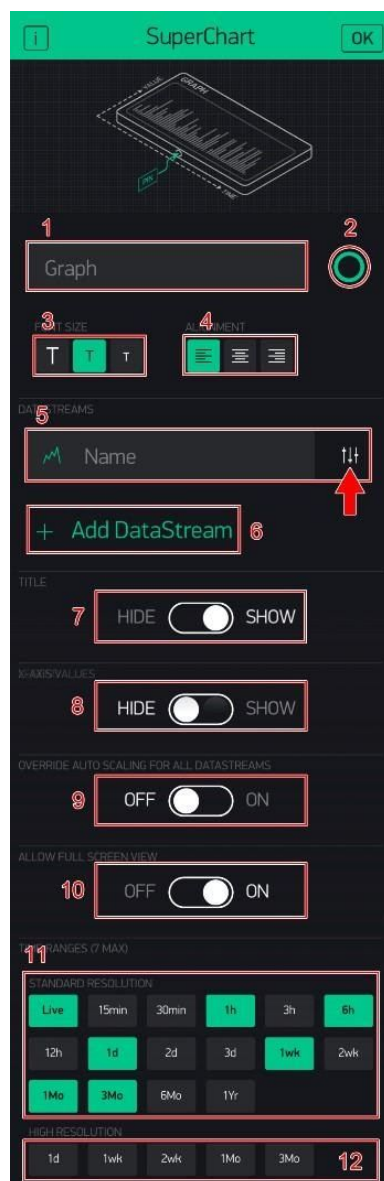


## Диаграмма (SuperChart)

Диаграмма нужна для графического представления значений датчиков в течение некоторого временного промежутка.



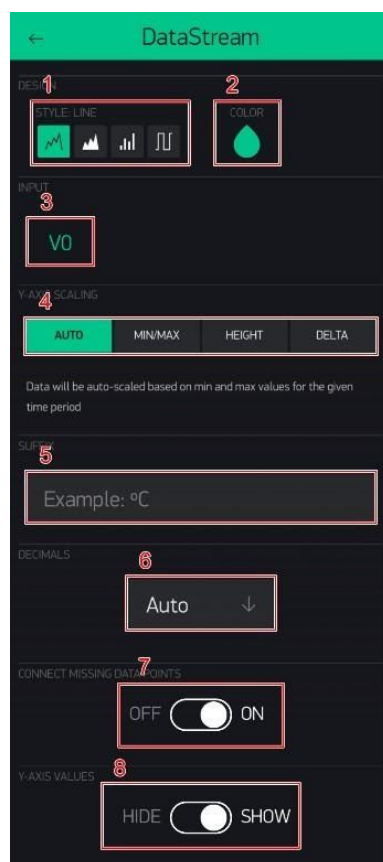
Рассмотрим настройки диаграммы.



1. Отображаемое название.
2. Отображаемый цвет.
3. Размер текста.

4. Выравнивание текста.
5. Отображаемое название диаграммы и переход к интерфейсу ее настройки (будет рассмотрен далее).
6. Добавить новый график
7. Спрятать или показать название диаграммы.
8. Отображение значений горизонтальной оси.
9. Запретить автомасштабирование для всех графиков.
10. Разрешить полноэкранный режим.
11. Выбор временного интервала, за который можно посмотреть статистику в обычном режиме.
12. Выбор временного интервала, за который можно посмотреть статистику в полноэкранном режиме.

Далее будут рассмотрены настройки непосредственно графиков.



1. Стиль графика.
2. Цвет графика.
3. Пин.
4. Способ масштабирования вертикальной оси.
5. Формат отображения подписи графика.
6. Точность числа.
7. Соединять точки до и после потери данных.
8. Отображения подписи вертикальной оси.

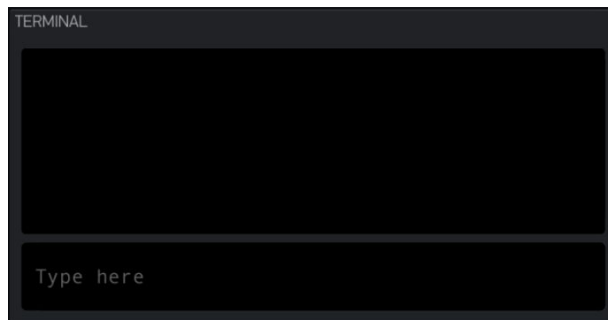
## Использование с IoTik Studio

Диаграмма отображает значения, отправленные в указанные порты (максимум 4 штуки). Со стороны контроллера должна быть возможность отправлять эти значения. В IoTik Studio это реализуется с помощью блока «Отправить значение».

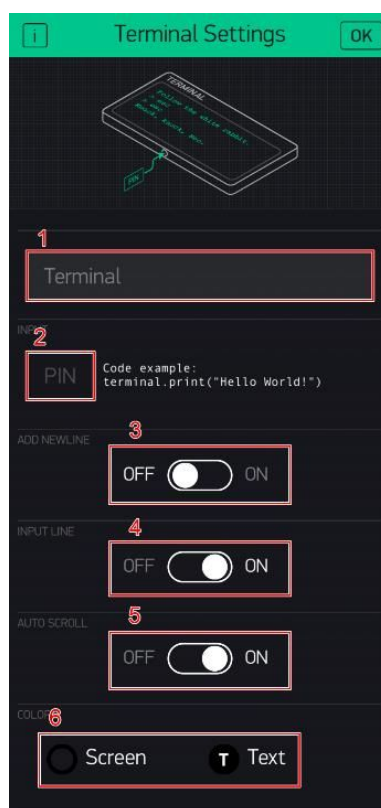


## Терминал (Terminal)

В рамках IoTik Studio терминал может только выводить текстовые сообщения.



Рассмотрим настройки терминала.



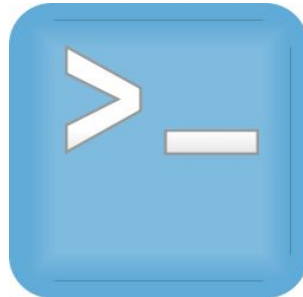
1. Отображаемое название.
2. Пин.
3. Строка ввода (рекомендуется отключить ее за невозможностью использования).
4. Автопрокрутка.
5. Цвет терминала и текста.

## Использование с IoTik Studio

Терминал отображает значения и строки, отправленные в указанный порт. Со стороны контроллера должна быть возможность отправлять эти значения. В IoTik Studio это реализуется с помощью блока «Отправить значение».



Чтобы отображать строки используется блок «Терминал» из раздела «Blynk».

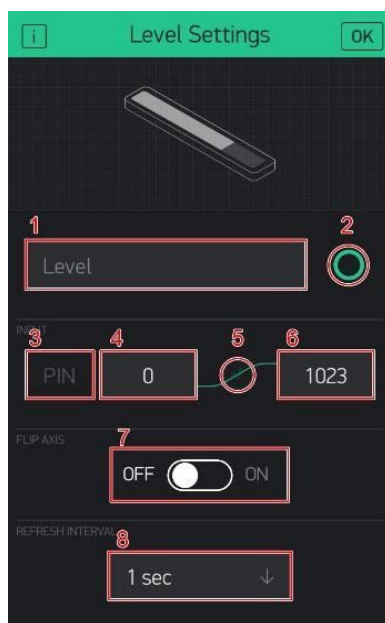


## Уровень горизонтальный/вертикальный (Level H/V)

Виджет для визуализации данных, похож в некотором роде на измерительный прибор.



Рассмотрим настройки уровня.



1. Отображаемое название.
2. Отображаемый цвет.
3. Пины.
4. Минимальное отображаемое значение.
5. Режим масштабирования данных.
6. Максимальное отображаемое значение.
7. Инверсия графика.
8. Частота обновления.

## Использование с IoTik Studio

Уровень отображает значения, отправленные в указанный порт. Со стороны контроллера должна быть возможность отправлять эти значения. В IoTik Studio это реализуется с помощью блока «Отправить значение».



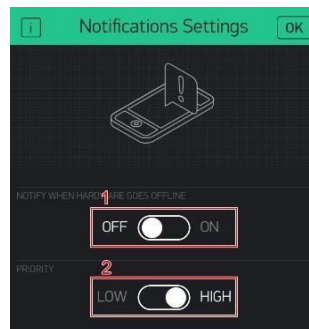
## Уведомления

### ***Push-уведомление (Notification)***

Виджет позволяет отправлять любое сообщение в push-уведомлении на устройстве. Также имеет возможность сигнализировать о разрыве



соединения контроллера с интернетом.



Рассмотрим настройки данного виджета.

1. Включение уведомлений о разрыве соединения контроллера с интернетом.
2. Включение высокого/низкого приоритета. При включенном высоком приоритете уведомления приходят быстрее.

### **Использование с IoTik Studio**

Для отправки push-уведомления с произвольной строкой в IoTik Studio используется блок «Уведомление» из раздела «Blynk».



## Дополнительные возможности IoTik Studio

Blynk позволяет модифицировать интерфейс пульта не только в мобильном приложении Blynk, но и на стороне контроллера. Для этого существует блок «Свойство» из раздела «Blynk».



В основном в виджетах можно менять отображаемые цвета и названия. В качестве примера возьмем измерительный прибор (Gauge). Чтобы изменить отображаемое название данного виджета, в блоке

«Свойства» нужно отредактировать поля следующим образом:

- Порт: порт, с которым работает измерительный прибор.
- Свойство: свойство, отвечающее за отображаемое название – «label».
- Значение: любая строка, например «My widget».

Чтобы изменить цвет, отредактируйте поля так:

- Порт: порт, с которым работает измерительный прибор.
- Свойство: «color».
- Значение: цвет в HEX формате, например «D3435C» (для нахождения HEX формата цвета рекомендуется использовать любой онлайн сервис, например <https://colorscheme.ru/color-converter.html>).

## **Словарь терминов**

I2C устройство -- цифровое устройство, работающее по протоколу I2C.

Аналоговое устройство -- устройство, работающее с аналоговым сигналом.

Аналоговый сигнал -- сигнал данных, который может быть представлен непрерывной функцией от времени.

Блок питания -- устройство, которое используется для создания напряжения, необходимого для работы контроллера и периферийных устройств.

Диаграмма -- совокупность визуальных блоков, соединенных друг с другом.

Драйвер -- программное обеспечение, необходимое для корректной работы с внешним устройством.

Клемма -- механический зажим для присоединения проводов к колодке.

Колодка -- изделие, предназначенное для присоединения проводов с помощью клемм.

Коннектор -- устройство для соединения электрических цепей, которое вставляется в порт, гнездо, разъем. RJ-9 -- коннектор.

Контроллер -- устройство управления. ЙоТик32 -- контроллер.

Переменная -- как правило символьный атрибут, обозначающий какое-либо число, которое может меняться в ходе исполнения программы.

Порт, разъем, гнездо -- устройство для соединения электрических цепей, куда вставляется коннектор.

Прошивка -- программное обеспечение для контроллера, необходимое для работы с ним.

Реле -- это механизм, который позволяет замкнуть электрическую

Светодиод -- прибор, создающий оптическое излучение при подаче тока через него.

Таймер в общепринятом понимании -- это устройство, которое подает сигнал через заданный промежуток времени.

Цикл в программировании -- конструкция, которая позволяет многократно выполнять указанные инструкции.

Цифровое устройство -- устройство, работающее с цифровым сигналом.

Цифровой сигнал -- сигнал данных, который можно представить в виде последовательности цифровых значений.

Шилд -- плата, расширяющая функциональность контроллера.

цепь по команде контроллера.