Методическое пособие

Занятие 1. Вводное занятие

Занятие 2. Знакомство со средой программирования TRIK Studio

Теория:

Визуальное программирование – технология программирования, которая предусматривает создание программ с помощью визуальных блоков кода.

Одним из вариантов программирования Пионера является программа TRIK Studio (рис.1). (ССЫЛКА!)

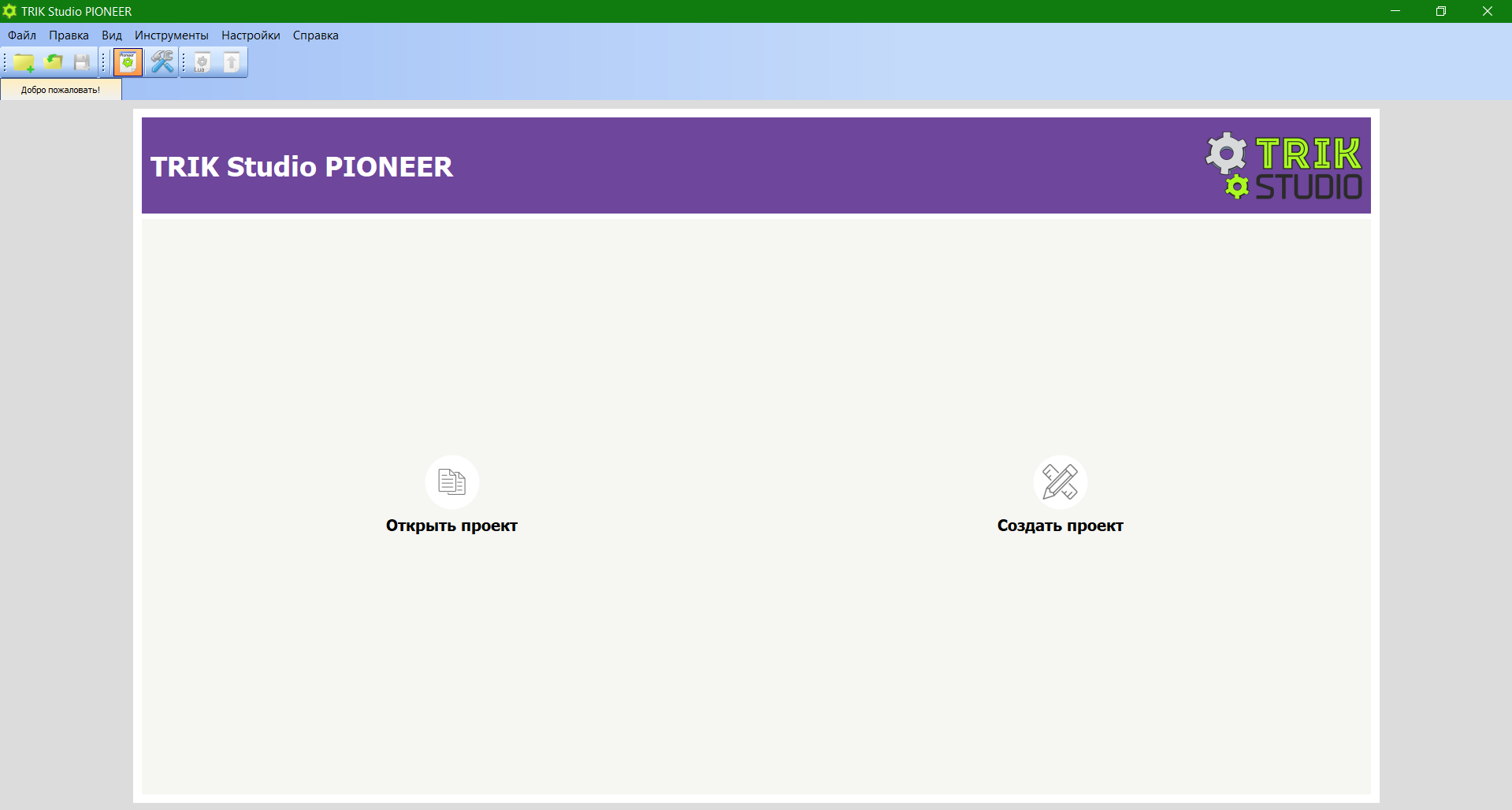


Рис.1 Начальное окно программы

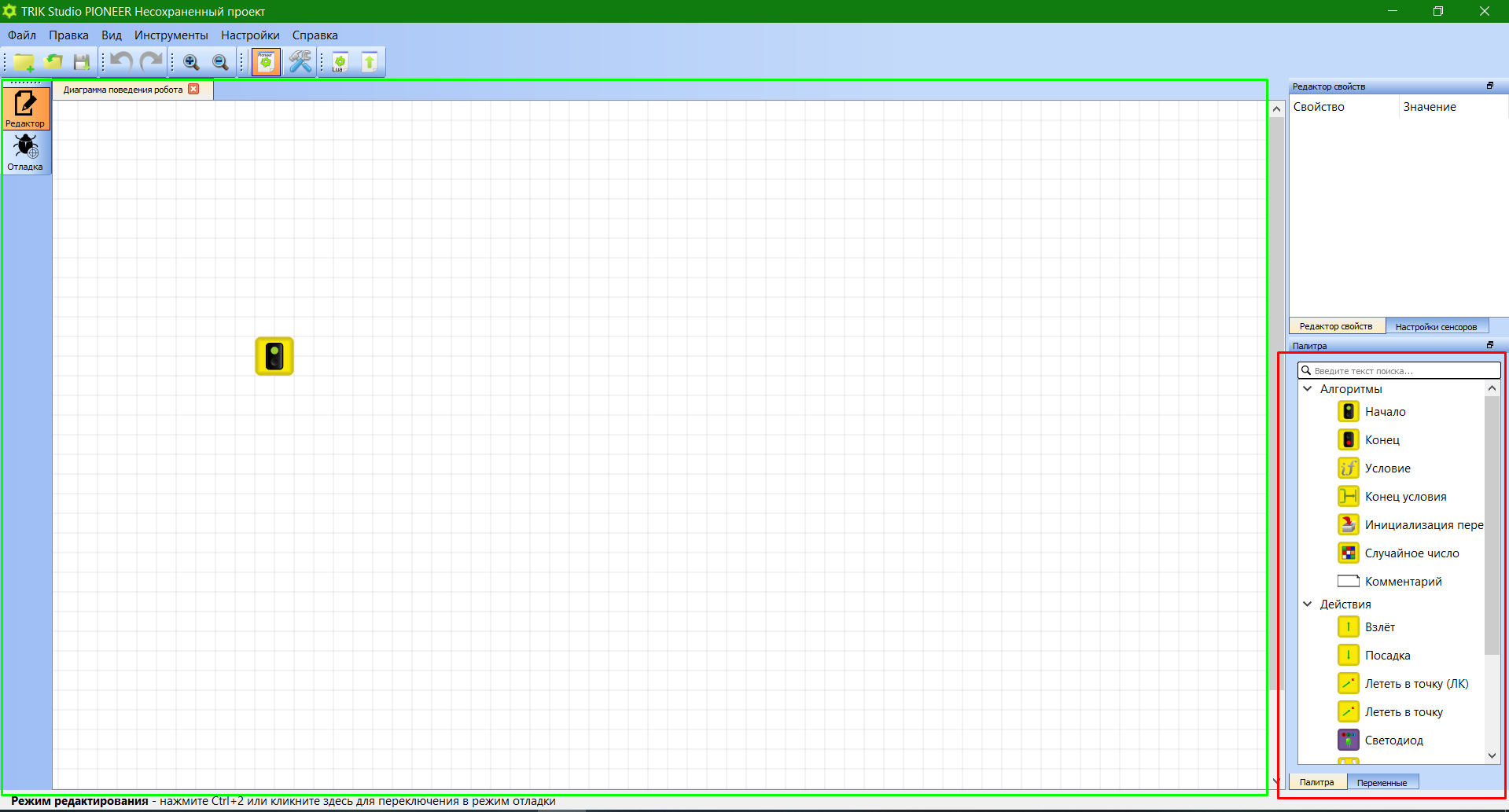


Рис.2 Проект в TRIK Studio

После нажатия «Создать проект» видим два основых блока (рис.2). Красным справа указан список блоков, которые можем использовать при работе. Зеленым, соответственно, поле для размещения этих блоков.

Все связи между блоками создаются ПКМ, каждый блок редактируется нажатием на него.

Обязательными блоками являются «Начало» и «Конец».

Практика:

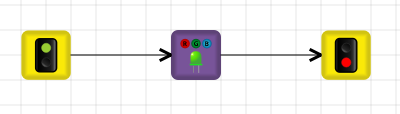
1. Первая программа в TRIK Studio, работа со светодиодом.

На главной плате Пионера находятся 4 светодиода, пронумерованные от 0 до 3. (фото платы?)

Каждый светодиод может светить в любом цвете (о том, как настроить цвета и яркость светодиодов мы поговорим на следующих занятиях).

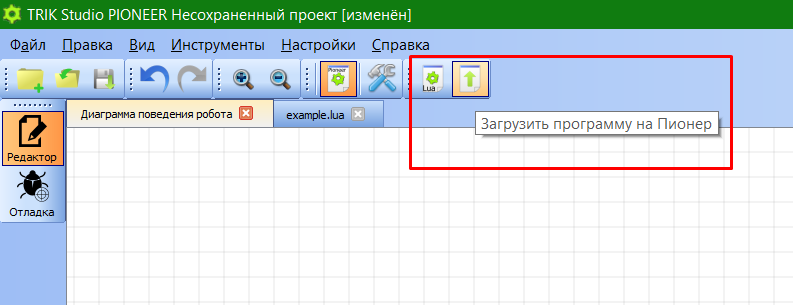
Возьмём с панели справа блок «Светодиод» и поставим его между блоками «Начало» и «Конец».

Правой кнопкой мыши соединим блоки.



При нажатии на блок светодиода выставим «красный» равным 1, остальные значения оставим 0. Таким образом, после загрузки программы светодиод под номером 0 будет гореть красным цветом.

После этого нажимаем на «загрузить программу на Пионер»



После нажатия автоматически откроется Pioneer Station (ссылка!)

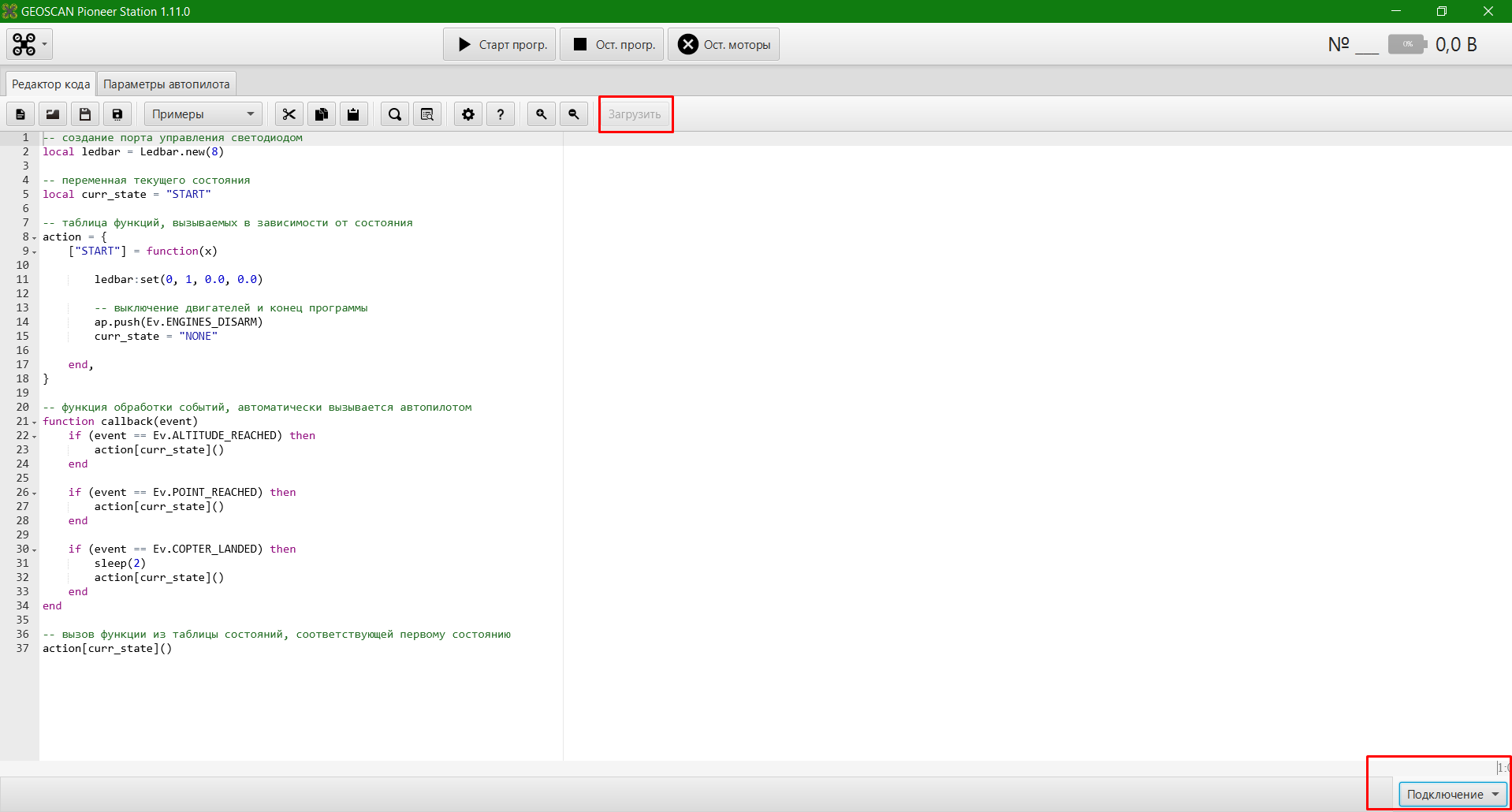


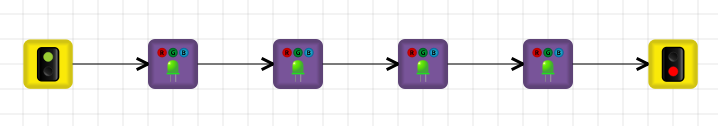
Рис. 3 Рабочее окно Pioneer Station

Этот код мы пока что не трогаем, подключаем Пионер по USB, снизу справа нажимаем «Подключение по кабелю USB», и загрузить (выделено красным).

После этого, при нажатии на кнопку «Старт» светодиод под номером 0 загорится красным. (фото?)

1. Самостоятельная работа

Попробуйте самостоятельно узнать нумерацию светодиодов, подставляя в код разные номера светодиодов и цвета (любые значения от 0 до 1).



1. Работа с блоком «Таймер».

Найдем на боковой панели блок «Таймер» и поставим их на рабочее поле, как на рисунке 4.

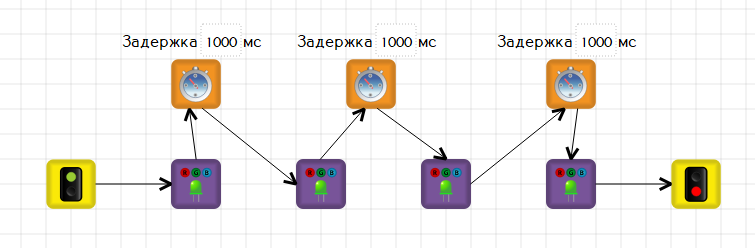


Рис. 4 Схема блоков «Таймер»

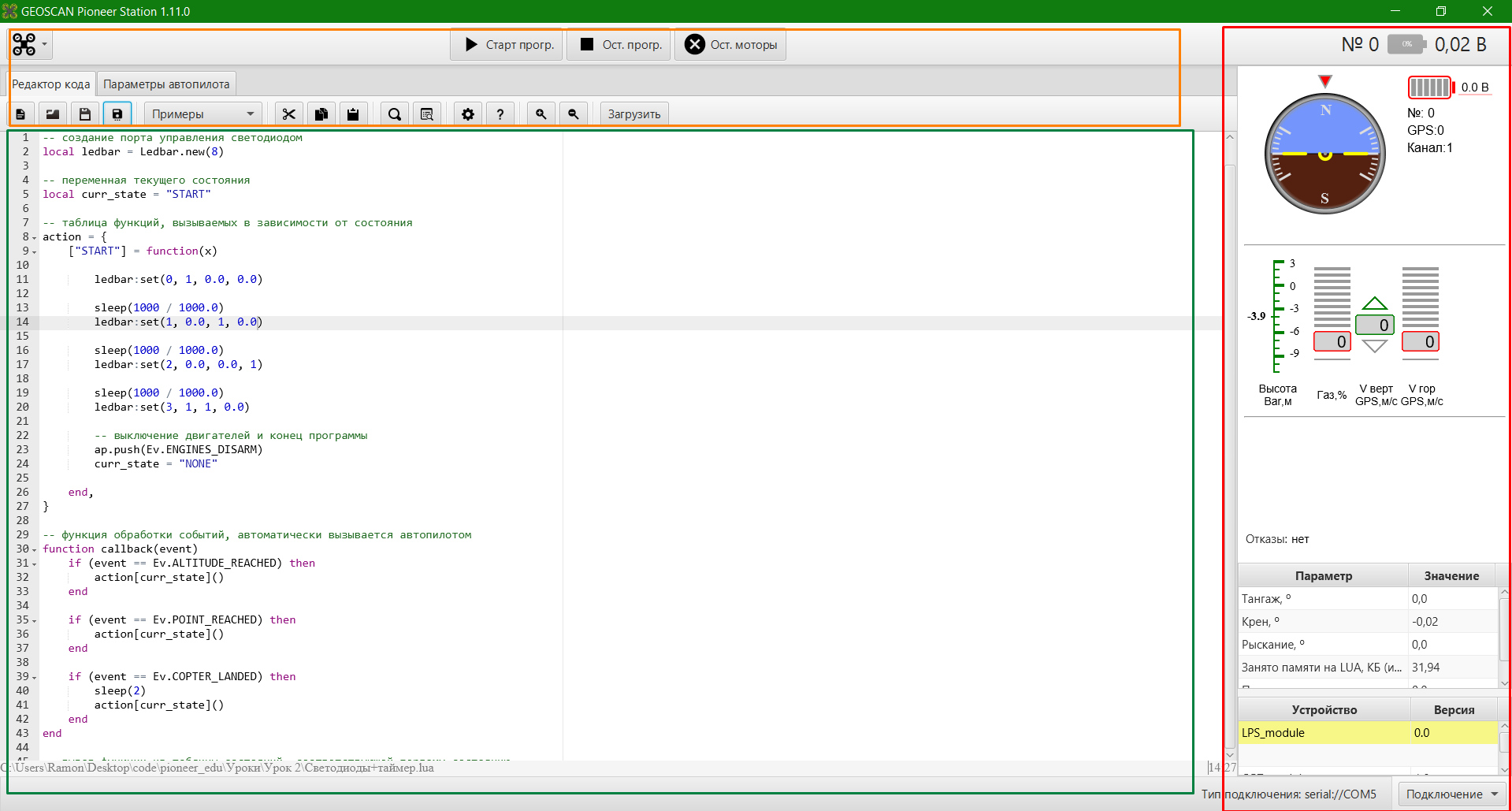
Данный блок позволяет сделать паузу в выполнении программы. Величину паузы указываем самостоятельно.

1. Самостоятельная работа

Урок 3. Знакомство со средой Pioneer Station

Теория:

Блочное программирование просто для понимания, однако TRIK Studio не позволяет использовать функционал дрона полностью. Основной средой для программирования Пионера является Pioneer Station.



Красный блок справа появляется при подключении Пионера, в нем можно увидеть информацию с датчиков Пионера.

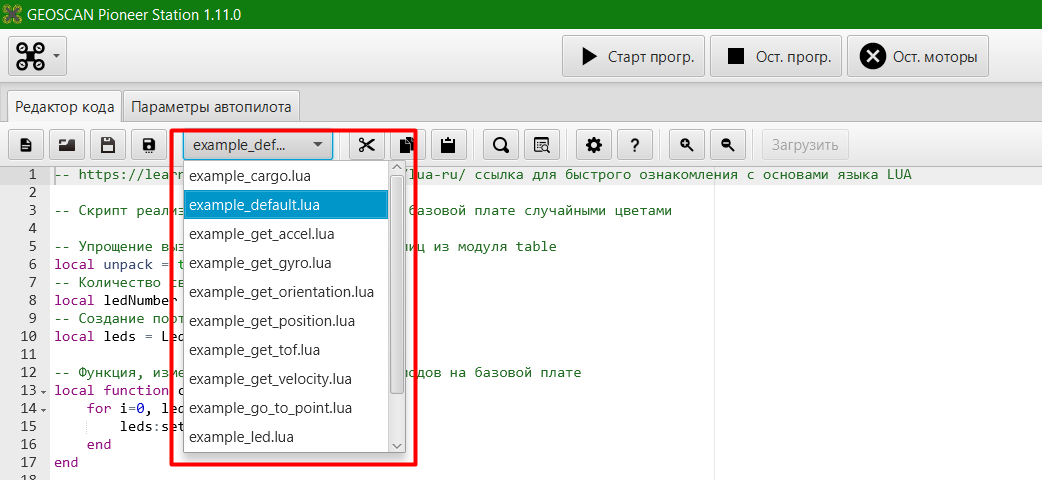
Зеленой рамкой выделено рабочее пространство студии.

Пионер написан на языке Lua, краткую информацию о нем вы можете почитать по ссылке <https://learnxinyminutes.com/docs/ru-ru/lua-ru/>

Практика:

1. Примеры программ.

В Пионере прописаны несколько примеров. Запустим некоторые из них. Найти их можете на скриншоте.



Каждый пример хорошо документирован, попробуйте запустить некоторые из них, допустим, example\_default.lua, который выполняет мигание светодиодов на плате.

Вот список всех примеров:

1. Example\_cargo – включение/выключение магнита каждую секунду
2. Example\_default – мигание светодиодами на плате
3. Example\_get\_\*\*\* - получение различной информации о полете и вывод ее на светодиоды
4. Example\_go\_to\_point – автономное перемещение дрона по заданным точкам
5. Example\_led – вывод чисел от 0 до 10 разными цветами.
6. Example\_path – усложненный вариант автономного полета
7. Learnlua – файл, показывающий все возможности языка Lua

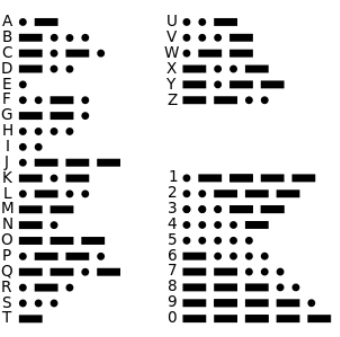
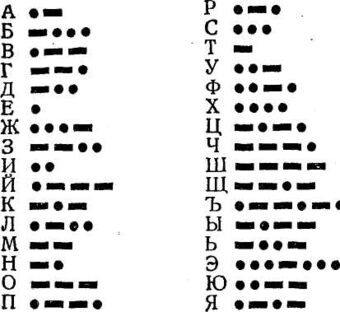
Для выполнения некоторых примеров необходимы дополнительные модули, такие как модуль подъема груза или led панель

Большую часть примеров мы напишем самостоятельно на будущих уроках, последовательно улучшая навыки программирования Пионера.

Занятие 4. Программирование в TRIK Studio – управление светодиодами

Теория:

Азбука Морзе – способ кодирования алфавита, цифр и знаков препинания с помощью двух знаков: точки и тире.

На рисунках показаны варианты английской и русской азбук Морзе.

С изобретением электрического телеграфа в 1774 году, требовалось придумать некий код, позволяющий только по длительности электрического сигнала шифровать и передавать различные символы. На протяжении 70 лет этот код модернизировался, пока в 1838 году Морзе не предложил свой вариант.

Справедливости ради, данный вариант также дорабатывался на протяжении десяти лет. Наибольший вклад внес Фридрих Герке, и этим вариантом мы пользуемся до сих пор.

Вслед за английской морзянкой стали развиваться азбуки Морзе на других языках, большинство из них были придуманы в течение пары лет после английской. Наибольший вклад в развитие телеграфа в России сделал Борис Якоби, русский физик-изобретатель.

Практика:

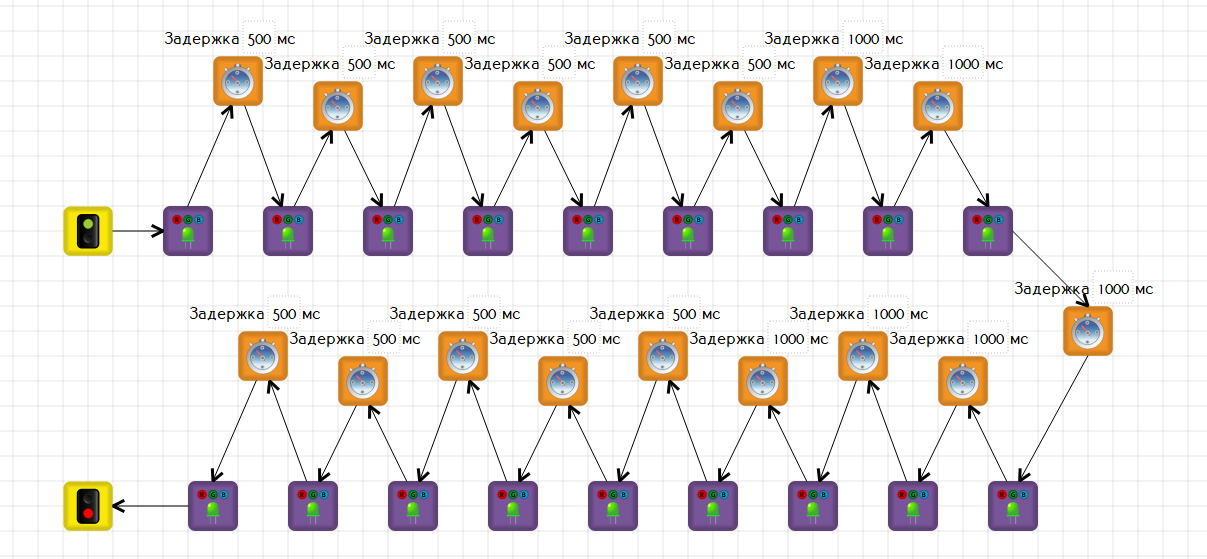
1. «Попавшие в беду»

Одной из самых известных и важных последовательностей на азбуке Морзе является международный сигнал бедствия – SOS.

Он применяется повсеместно приблизительно с начала 20 века и является последовательностью из 9 знаков: 3 точки, 3 тире, 3 точки.

Попробуем сделать так, чтобы коптер подавал сигнал бедствия своими светодиодами.

В TRIK Studio создадим цепочку из 18 светодиодов и 16 таймеров между ними, так как для каждого включения светодиода нам необходим блок, выключающий его.



В данном примере цвет и задержку выбирайте сами. Однако, так как точки должны мигать быстрее, чем тире, между точками проставим таймер по 500 мс, а между тире 1000 мс.

Занятие 5. Программирование на языке Lua – управление светодиодами.

Теория:

RGB-светодиод – это три одноцветных кристалла совмещенные в одном корпусе. Название RGB расшифровывается, как Red – красный, Green – зеленый, Blue – синий соответственно цветам, которые излучает каждый из кристаллов.



Как вы знаете, путём смешивания трёх основных цветов можно получить любой другой цвет.

Упрощённо говоря, яркость света – это отношение силы света, которую испускает поверхность к площади этой поверхности.

По тем же механизмам работает и наш глаз. В нем есть палочки, ответственные за восприятие яркости, и колбочки, которые воспринимают цвет. Этих колбочек в глазу три типа, по основным цветам – красные, синие и зеленые, каждая из которых реагирует на свой.

Практика:

1. Самостоятельное составление таблицы цветов.

В Пионере любой цвет на светодиоде кодируется тремя параметрами от 0 до 1, где 0 это отсутствие свечения в данном цвете, а 1 — это самый яркий цвет.

Попробуйте сами составить таблицу цветов радуги, подставляя различные цифры в блок светодиода.

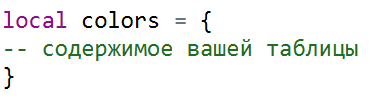
Таблицы в Lua создаются по одинаковой схеме:

*local <имя вашей таблицы> = {*

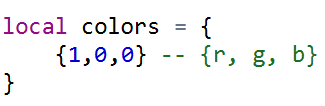
*-- содержимое таблицы*

*}*

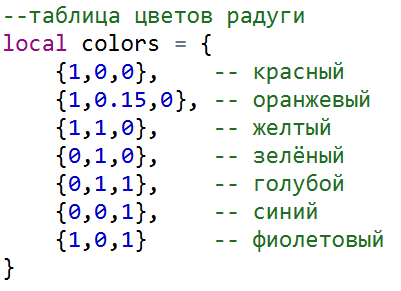
, где вместо «имя таблицы» указывайте любое, я поставлю colors.



Содержимое ячеек таблицы может быть, как именованное, так и неименованное. Сами ячейки могут содержать в себе один параметр, другие таблицы или целые массивы информации.

Теперь, каждый цвет закодируем в формате {красный, зеленый, синий}, допустим, для красного цвета: 

Создадим таблицу всех цветов радуги



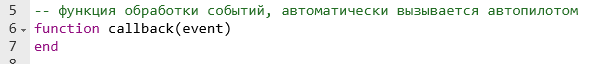
1. Код Lua для управления светодиодами.

Откроем Pioneer Station и создадим новый файл.

Для работы со светодиодами необходимо их инициализировать. Для этого введем переменную, в которую пропишем число светодиодов и команду Ledbar.new, которая их инициализирует.

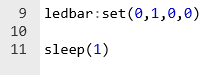


Далее необходимо прописать функцию callback, в которую должны приходит различные состояния при полёте, но, так как в этом задании не будет полета, оставим функцию пустой.



После этого мы можем прописать код, который будет управлять светодиодами.

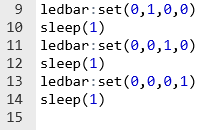
Каждый блок прописывается одной командой – Светодиод – ledbar:set(), Таймер – sleep().



В ledbar:set() в скобки пропишем 4 параметра: номер светодиода и значения красного, зеленого и синих цветов соответственно. На скриншоте загорится светодиод номер 0 красным цветом.

Команда sleep() требует только один параметр – время задержки в секундах.

Теперь мы можем сделать так, чтобы светодиод загорался всеми цветами радуги, используя таблицу цветов.



Данный пример переключает цвета: красный, зеленый и синий с задержкой в 1 секунду.

Также попробуйте поставить меньшую яркость, вместо 1 ставя 0.5.

Теперь воспользуемся таблицей, сделанной ранее.

Для начала напишем небольшую команду, которую будем использовать для удобства (она нужна для переноса данных из таблицы цветов в ledbar:set).



*local unpack = table.unpack*

Воспользуемся ею при включении светодиода



Первый параметр это номер светодиода, а цифра в квадратных скобках это порядковый номер цвета в нашей таблице, начиная с 1. Т.е. в данном примере светодиод загорится красным цветом.

Занятие 11. Объединение пилотируемого полета и автономной программы

Теория:

Настройка пульта для работы с квадрокоптером хорошо прописана в инструкции по сборке Пионера, начиная со стр.24.

Каждый тумблер на пульте имеет свой канал связи с Пионером. Всего их 8: 1-4 отвечают за стики управления (по осям x и y соответственно), 5-8 за тумблеры SwA, SwB, SwC и SwD в соответствии, указанном в инструкции (Настройки -> “Aux. channels”), а именно – 5 – SwC, 6 – SwD, 7 – SwB, 8 – SwA.

Каналы используются для смены каких-либо состояний в полёте. Например, тумблер SwB ставится в крайнее нижнее положение для перехода в режим автономного пилотирования, среднее положение тумблера SwC используется для зависания дрона на одной высоте.

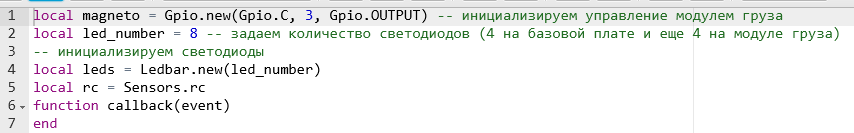
Практика:

1. Модуль магнитного захвата груза.

Магнит крепится снизу на коптер в соответствии с длиной паза, так как один паз длиннее, модуль всегда встает в правильном положении. При необходимости можно поставить стойки и вкрутить модуль винтами.

На пульте необходимо настроить 8 канал на канал SwA (Настройки -> “Aux. channels” -> “SwX” -> “SwA”)

После этого приступим к написанию программы. Для начала инициализируем светодиоды (теперь их стало 8, 4 на плате и 4 на магните), магнит и пропишем функцию callback.



Далее напишем функцию, которая будет менять цвет всех светодиодов.

Синтаксис функции такой:

local function <имя функции>()

<Блок тела функции>

end

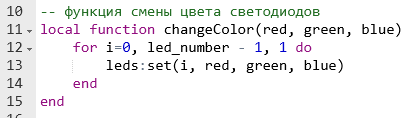
Внутри функции введем цикл for. Этот цикл с каким-то шагом выполняет то, что у него внутри (по умолчанию шаг = 1)

for <начало>, <конец>, [<шаг>] do

<Блок тела цикла>

end

Квадратные скобки у <шаг> означают то, что этот параметр необязателен.



Данная функция changeColor требует при вызове в скобках три параметра – цвет светодиода, а затем выполняет leds:set для каждого i от 0 до led\_number – 1, так как нумерация светодиодов начинается с 0.

Далее напишем таймер. Таймер выполняет какую-либо функцию с заданным интервалом. В нашем случае таймер должен каждые N секунд выполнять проверку положения тумблера, где N достаточно малое время, чтобы не было задержки.



На строке 18 описано получение данных именно с 8 канала.

После этого сделаем разветвление программы, условием которого будет положение тумблера.

Для этого используется блок if/else.

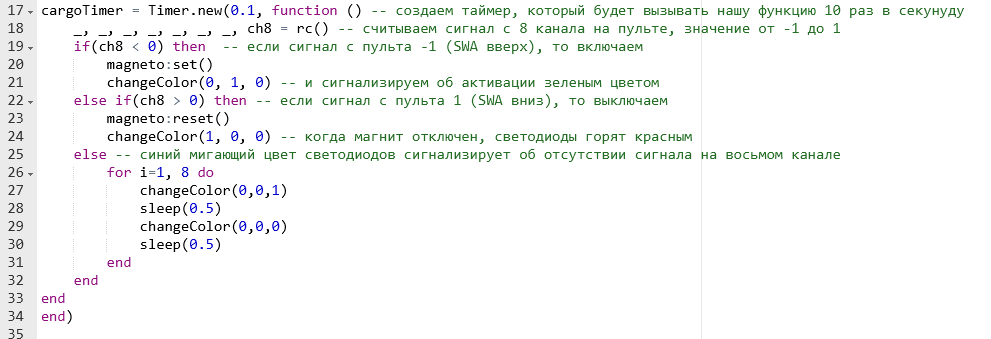
if <условие> then

<блок кода, при выполнении условия>

else

<блок кода, при не выполнении условия>

end



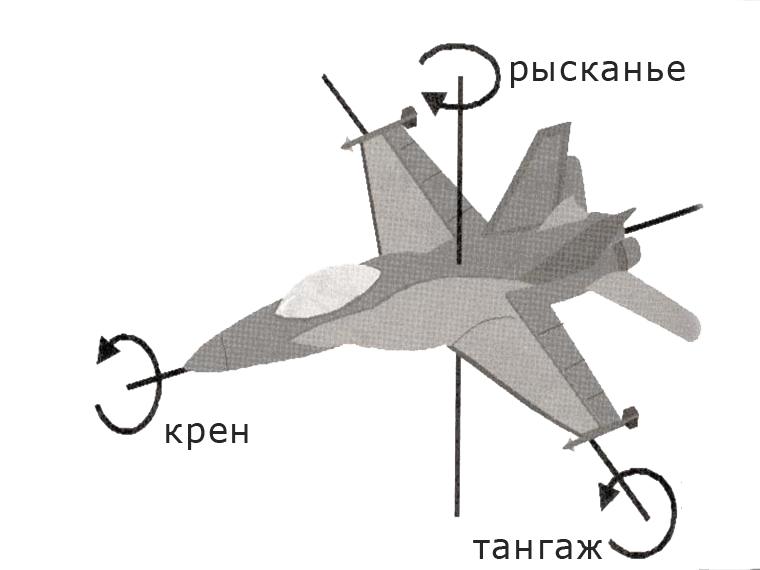
Не запутайтесь в числе end в конце блока.

Остается последняя строка кода, которая включит таймер - 

Попробуйте также добавить sleep перед выключением магнита.

Занятие 13. Программирование в TRIK Studio – Рыскание.

Теория:

Любой летательный аппарат может поворачиваться в 3 плоскостях. В зависимости от плоскости каждый поворот называется по разному. 

Тангаж – поворот по поперечной оси, т.е. нос летательного аппарата вверх/вниз.

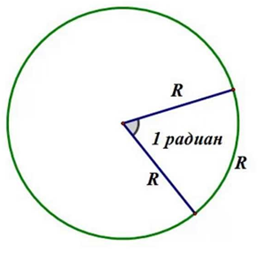
Крен – поворот по продольной оси, при нём крылья будут подниматься вверх/вниз в зависимости от направления.

Рыскание – поворот по вертикальной оси, нос летательного аппарата будет двигаться влево/вправо.

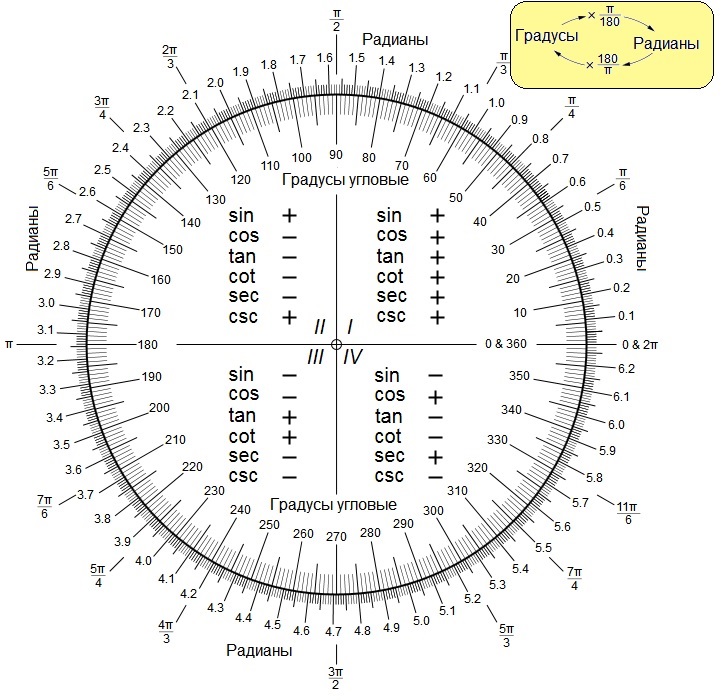
Из всех видов поворота нам важно именно рыскание.

Рыскание проводится на угол, однако в программировании важны радианы.

Радиан – это угол, который образуется, при наложении радиуса на окружность.

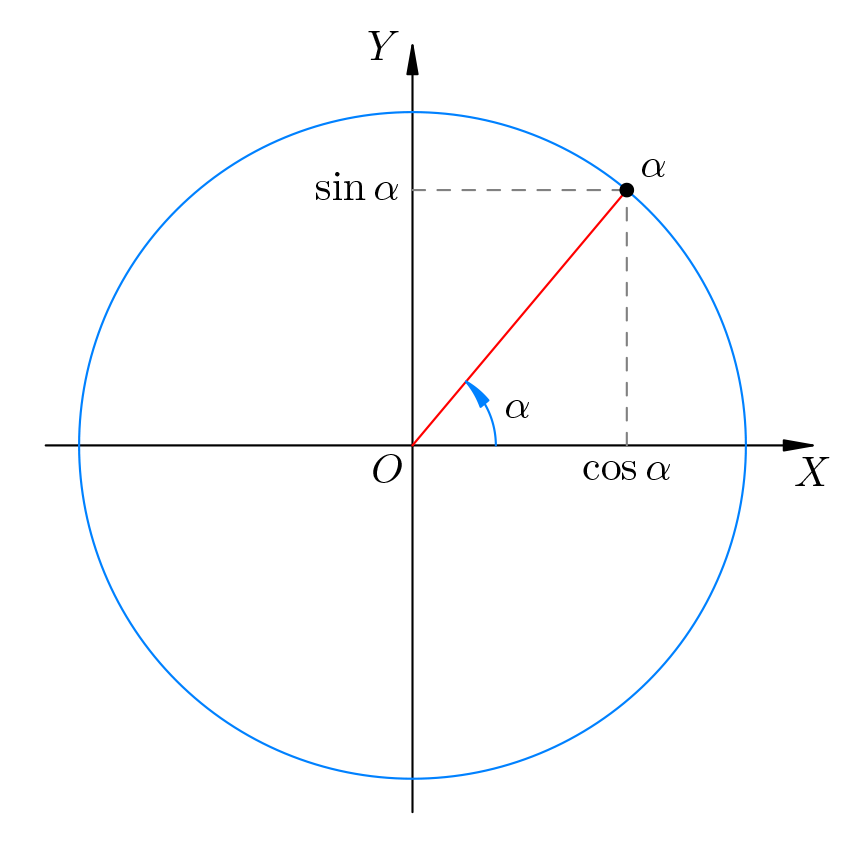


Радианы нам важны для понимания тригонометрического круга.



Так как часто мы будем запускать дрона по окружности, нам нужно рассчитать точки, к которым он будет лететь с помощью ap.goToLocalPoint(x,y,z).

Для вычисления координаты любой точки на окружности, зная лишь угол, нам необходимы синус и косинус (sin и cos).



Все значения синуса и косинуса любого угла есть в тригонометрических таблицах (как пример, таблица Брадиса) или в калькуляторе/интернете.

Практика:

1. Полет по окружности с ориентацией по курсу.

Для полёта по любой траектории нужно получить координату и угол поворота.

Для примера мы рассчитаем 8 точек (каждые 45 градусов).

Для расчета координаты воспользуемся формулой x = R \* cosA, y = R \* sinA.

R примем за 1 метр.

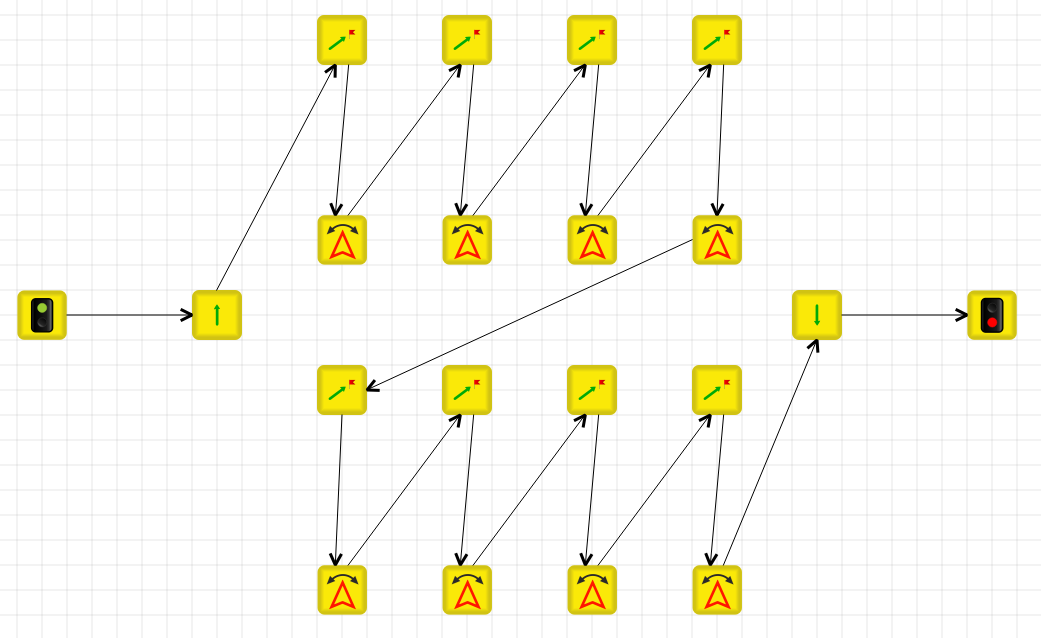
Вот полученные координаты в виде таблицы.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| x | 1 | 0,7 | 0 | -0,7 | -1 | -0,7 | 0 | 0,7 |
| y | 0 | 0,7 | 1 | 0,7 | 0 | -0,7 | -1 | -0,7 |

Угол тангажа будет -45, так как точки поставлены против часовой стрелки.

Высоту Z поставим 1 метр.

В TRIK Studio получается такая схема:



Занятие 14. Программирование на языке Lua – таблица

Теория:

Если вы имеете опыт программирования, то таблицы в Lua это аналог массива в любом другом языке.

Если же нет, таблица – это некое хранилище разных данных. В таблицах можно хранить любой тип данных, даже другие таблицы.

С таблицами мы уже сталкивались при создании таблицы цветов, но вспомним синтаксис.

local <название таблицы> = {

<Данные внутри таблицы>

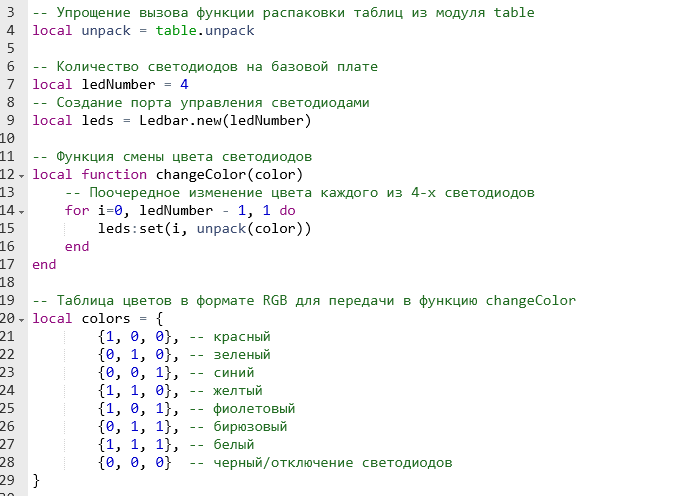
}

Повторите основы тригонометрии из предыдущего занятия.

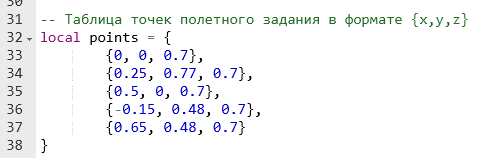
Практика:

1. Полет по массиву точек.

Сперва по аналогии с предыдущими занятими пропишем код для инициализации светодиодов, функцию смены цвета на всех светодиодах и таблицу цветов.

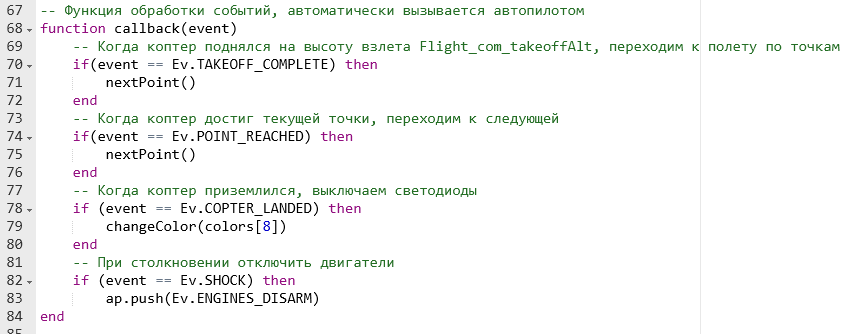


Далее создадим таблицу с точками полёта, для примера это будет правильная звезда. Координаты точек указаны на скриншоте (попробуйте их высчитать самостоятельно).



В функции callback необходимо прописать, что будет делать коптер при различных событиях. Мы будем отлавливать четыре из них:

1. Коптер успешно взлетел
2. Коптер добрался до нужной точки
3. Коптер ударился о препятствие
4. Коптер приземлился



Далее создадим таблицу с действиями, которые должен выполнить Пионер. Назовем ее “actions”. Пример этой таблицы вы можете посмотреть в коде Lua предыдущего занятия.

В этой таблице необходимо прописать различные события, которые будут вызываться. В конце каждого события нужно прописать, какое действие запустить.

Как пример приведем список дел каждое утро:

actions = {

if <проснулся> then

иди умываться

след.действие = “умывание”

end,

if <умывание> then

…

И так далее.

Список действий для полета по точкам у дрона следующий:

1. Включить двигатели, после небольшой паузы взлететь

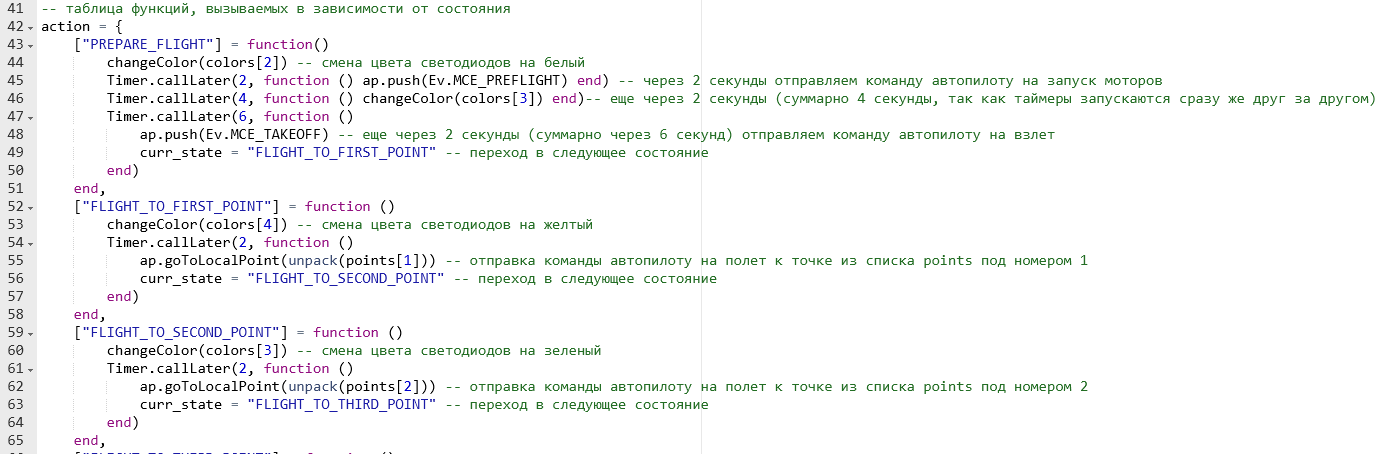
2. Добраться до точек 1-5

3. Приземлиться

Так как у Пионера нет индикации, кроме светодиодов, на каждой из точек будем зажигать их новым цветом, что будет говорить, на каком пункте списка actions сейчас дрон.

После запуска двигателей необходимо подождать какое-то время, чтобы они включились на полную мощность, поэтому на взлете будем использовать таймеры.

NB! Таймеры на Пионере выполняются одновременно, что позволяет прописывать точное время запуска различных команд. В один момент может работать не более 15 таймеров. Команду sleep нежелательно использовать, так как она приостанавливает действие всего скрипта, включая таймеры.



Каждое состояние в таблице имеет общий вид:

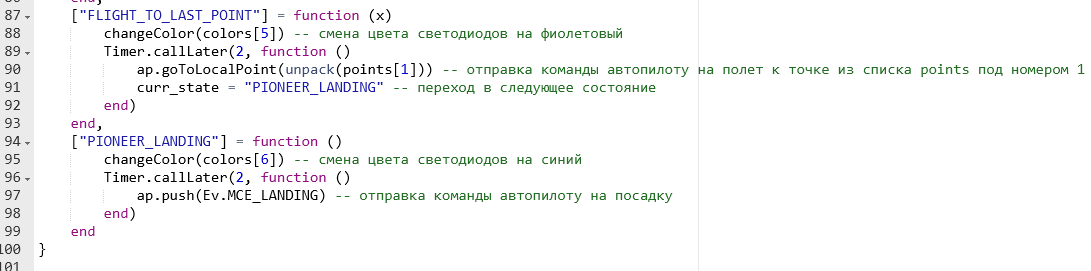
[<Название>] = function()

<блок действий функции>

end,

При взлете запускаются сразу три таймера: первый запускает двигатели, второй меняет цвет светодиодов и третий поднимает Пионер. Каждое действие выполняется спустя 2 секунды. Далее программа переходит на следующее состояние.

Полет по точкам – это 5 одинаковых блоков, которые отличаются только цветов светодиодов и точкой полета (по аналогии со светодиодами, прописываем unpack(<имя списка>[<номер точки>])).



Последняя точка – это возврат в начало (точка 1), затем переход к посадке.

Остается в самом конце программы прописать два небольших блока: название первого состояния для Пионера и запуск всей программы.





1. Движение по окружности с помощью цикла.

Попробуем повторить движение по окружности из предыдущего занятия, но проставим большее количество точек. Суть программирования в облегчении работы, поэтому напишем сначала цикл, который создаст n точек, чем больше, тем лучше.

Для начала создадим пустой список для точек.

local points = {}

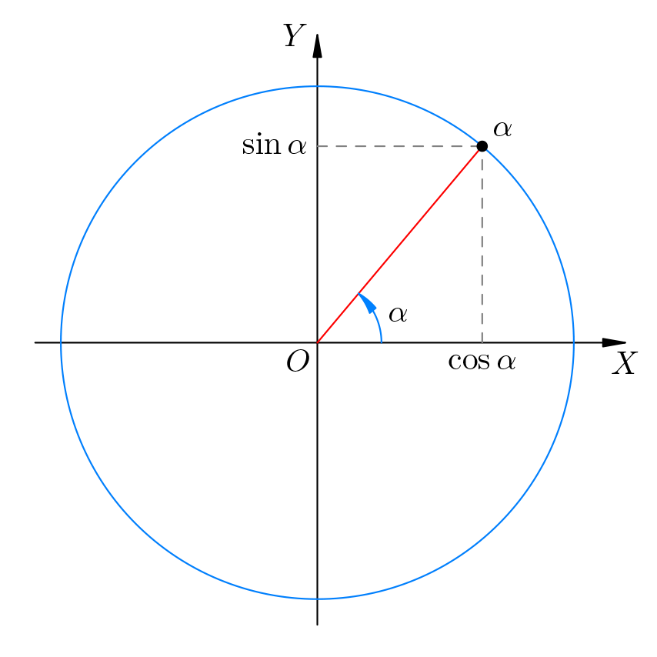
После этого используем цикл for.

for i=1, <количество точек>, do

points[i] = {<координаты точки>}

end

Для координат на окружности необходимо опять вернуться к тригонометрическому кругу.



Координата по x будет равна R окружности \* cosA; по y, соответственно, R \* sinA.

Получается, вид координаты в цикле будет {r\*cosA, r\*sinA, z}, где z – высота полета.

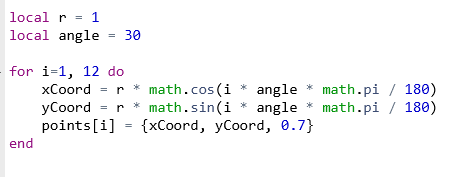
Радиус окружности пропишем в любом месте до цикла (в метрах)

local r = <значение>

Так как i в цикле увеличивается на 1, значение угла для каждой точки можно получить, как i \* angle, заранее прописав значение этой переменной по аналогии с радиусом.

Функции синуса и косинуса в языке Lua прописываются из библиотеки math, и получаются вида math.sin() и math.cos(). Но эта библиотека в параметр требует значение угла в радианах, так что необходимо перевести угловое значение в дуговое – по формуле angle \* math.pi / 180.

Таким образом, получается следующий цикл:



Также можно дописать внутри цикла код, который будет прекращать выполнение цикла после какого-либо условия, к примеру, когда угол уже больше 360 градусов.

для этого после points[i] пропишем:

if i \* angle > 360 then

break

end

Как летать по этим точкам, не прописывая десятки действий рассмотрим на следующем занятии.

Занятие 15. Программирование на языке Lua – Циклы.

Теория:

С циклами мы уже немного знакомы. На предыдущих занятиях использовался цикл for, это «цикл с счетчиком». Он выполняется определенное количество раз.

Кроме этого, мы будем пользоваться «безусловным» циклом. Это цикл, который выполняется при выполнении его условия, что может привести к образованию бесконечного цикла (если параметры в условии не меняются в цикле).

Синтаксис этого цикла такой:

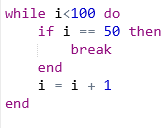
while <условие цикла> do

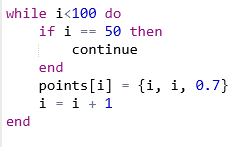
<блок команд>

end

Чаще всего именно в блоке команд меняют переменную в условии.

Все циклы имеют несколько полезных операторов, break и continue. Первый досрочно завершает цикл, а второй начинает следующий проход, пропуская весь код внутри тела цикла.

 Данный цикл прервется, когда счетчик дойдет до 50.

Данный цикл не создаст точку при i = 50, но все остальные создаст

Практика:

1. Бесконечный полёт по окружности.

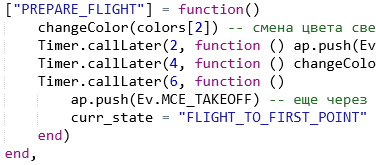
Для начала в Pioneer Station пропишем все команды, необходимые для инициализации светодиодов и упрощения работы.

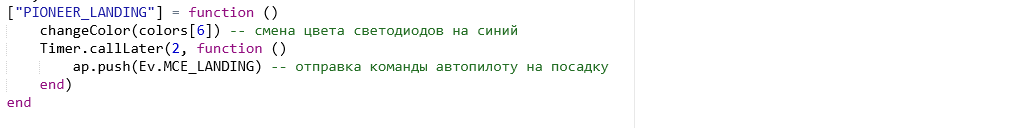


Также возьмем цикл, который создали на предыдущем занятии.

Теперь добавим список actions, в котором на этот раз будут только 3 состояния: взлет, полет к следующей точке и посадка.

Блоки взлета и посадки остаются прежними.



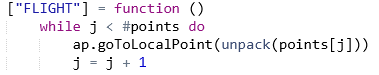


Все остальные блоки заменим на один, который назовем “FLIGHT” (не забудьте заменить название состояния в блоке взлета)

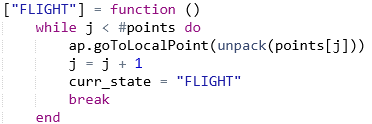
В теле состояния Flight пропишем безусловный цикл while, который будет выполняться, пока счетчик не превышает длину списка точек. Для этого в начале программы создадим переменную-счетчик j (local j = 1). Длину списка можно получить, написав #<название списка>.

Таким образом, инициализацией цикла будет while j < #points do

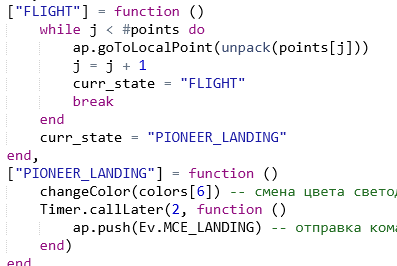
Внутри цикла пропишем перемещение к необходимой точке и увеличение счетчика.



Однако, в Пионере можно поставить только одно перемещение в одном состоянии, иначе дрон полетит сразу к последней точке (что в нашем случае будет той же самой точкой, что и начало полета). Поэтому нам необходимо выйти из цикла, предварительно поставить выход из цикла с помощью команды break, предварительно зациклив состояние, поставив переход к нему самому (curr\_state = “FLIGHT”)

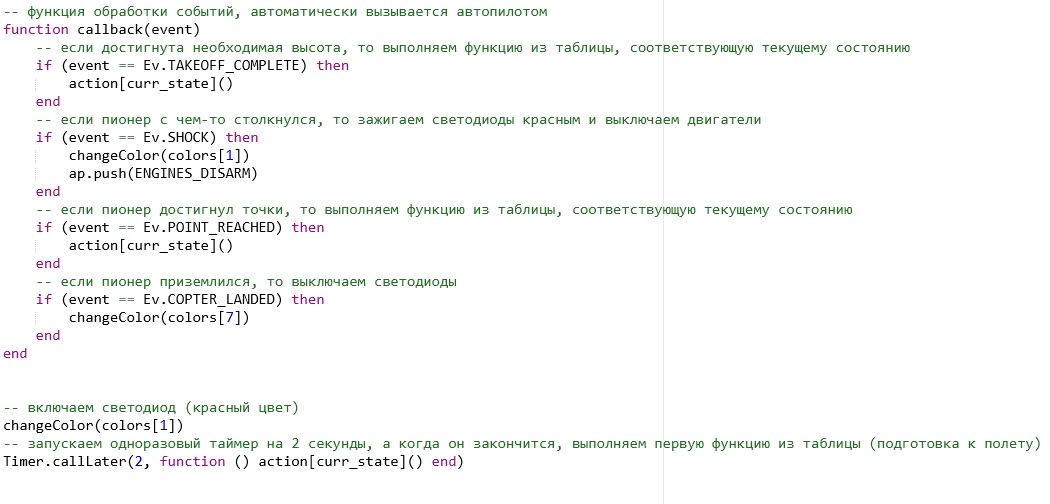


И, необходимо как то обработать выход из цикла, в нашем случае Пионер должен перейти в состояние посадки.



Таким образом, цикл будет выполняться до тех пор, пока не пройдет все точки, а после этого пойдет на посадку.

Осталось прописать функцию callback и начало программы, этот код возьмем из предыдущего занятия.

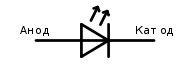


Попробуйте написать код для большего числа точек, чтобы получить более плавный полет.

Занятие 16. Программирование на языке Lua – управление светодиодной панелью

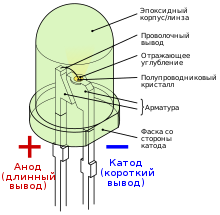
Теория:

Светодиод – полупроводниковый прибор, который излучает практически монохромный свет в прямом направлении (монохромный – очень узкий диапазон света, т.е. только волны определенного цвета).

В электрических схемах светодиод обозначается следующим образом  Как вы видите, у светодиода есть полярность, это все из-за его полупроводниковой структуры.

Полупроводник – это общее название группы веществ, которые имеют разную электрическую проводимость в зависимости от температуры (очень холодно – почти не проводят ток, тепло – хорошо проводят). Нахождение полупроводников значительно повлияло на развитие электроники, так как был изобретен полупроводниковый диод – прибор, пропускающий ток только в одну сторону (например, преобразует переменный ток в постоянный).

У стандартного светодиода две ноги – плюс (анод) и минус (катод).

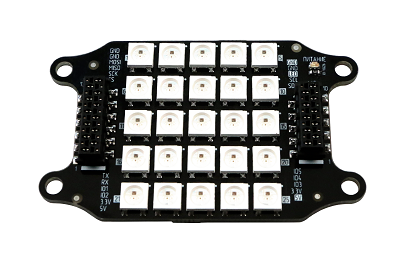


Похожая картина у RGB светодиода

У него 4 ноги, 1 катод и 3 анода, для каждого из цветов. Соответственно, подавая разный ток на каждую из ног светодиода можно получить любой цвет видимого спектра.

На предыдущих занятиях мы уже работали со светодиодами, но на этом будем использовать дополнительный LED модуль.

Он представляет из себя 25 светодиодов, пронумерованные с верхнего левого угла платы.



Для инициализации светодиодов используется класс Ledbar, который имеет 2 основные функции – инициализацию (Ledbar:new(<Count>), где Count – число светодиодов) и (Ledbar:set(x, r, g, b), где x – номер светодиода)

Практика:

1. Вывод цифр от 0 до 9