**Артём Денисович Голиков Денис Владимирович Голиков**
**Школа капитана Грампа. Scratch и Arduino для школьников**

ISBN 9785448549885

## Аннотация

*Это не обычный учебник программирования. Это дверь в фантастический мир космических приключений. Почти реальных, так как результатом изучения каждой главы станет запрограммированный своими руками симулятор космического корабля. Создавая проекты и управляя космическим кораблем, школьники совместно с героями книги пройдут по опасному пути искателей космических сокровищ.*

# Школа капитана Грампа
# Scratch и Arduino для школьников

**Денис Владимирович Голиков**
**Артём Денисович Голиков**

*Иллюстратор* Мария Конопатова
*Дизайнер обложки* Мария Конопатова
*Фотограф* Денис Голиков



ISBN 978-5-4485-4988-5
Создано в интеллектуальной издательской системе Ridero

## Предисловие

Перед вами не обычный учебник программирования. Это дверь в фантастический мир космических приключений. Почти реальных, так как результатом изучения каждой главы станет запрограммированный своими руками симулятор космического корабля. Создавая игры и управляя космическим кораблем, школьники совместно с героями книги пройдут по опасному пути искателей космических сокровищ. Им предстоит научиться управлять маневровыми и маршевыми двигателями корабля, освоить стыковку и посадку, полет в атмосфере, управление боевым лазером, а также освоить ремонт различных систем корабля.

Управление космическим кораблем будет происходить с помощью специального пульта управления, состоящего из плат Arduino и Joystick Shield, подключенных к компьютеру через USB.

Обучение будет вестись в среде программирования Snap4Arduino, которая является одной из модификаций Snap! – блочной среды программирования, разработанной на основе Scratch в Калифорнийском университете в Беркли (University of California at Berkeley).

Программирование на Snap4Arduino, как и в Scratch, происходит путем соединения разноцветных блоков, которые, последовательно исполняясь, управляют движением космического корабля, астероидов и других объектов.

Snap! был разработан Дженс Моних (Jens Mönig at MioSoft Corporation, now at SAP),

совместно с Брайаном Харви (Brian Harvey at Berkeley), при помощи студентов Калифорнийского университета в Беркли.

Snap4Arduino разработан Бернатом Ромагозой (Bernat Romagosa) совместно с командой единомышленников в лаборатории Citilab в Барселоне.

Для того чтобы успешно пройти все космическое приключение ребенок должен быть знаком с десятичными дробями и отрицательными числами, уметь выполнять арифметические действия, а также иметь базовые навыки управления компьютером.

Книга рассчитана на возраст 11—13 лет, однако многие игры смогут сделать и школьники помладше.

# Введение

## История создания книги

История создания этой книги началась с одной необычной находки неподалеку от жерла вулкана Тейде, под обломками пемзы. Это был необычный металлический ящик, явно неземного происхождения. Больше всего он был похож на черный ящик сверхзвукового истребителя. Замок был сломан, и внутри обнаружилось несколько блокнотов, исписанных на ломаном русском с большим количеством ошибок. Почерк был аккуратным и разборчивым. Первые и последние страницы блокнота обгорели, наверное, ящик побывал в огне.

Изучив содержимое блокнотов, мы поняли, что этот ящик попал в наше время из будущего. Это произошло в результате применении новейшего космического оружия XXIII века, основанного на создании микрочерных дыр, разбрасывающих уничтожаемые предметы не только в пространстве, как обычное оружие, но и во времени.

Основным содержанием блокнотов оказался учебник для юных пилотов, проходивших обучение на борту звездолета под руководством некоего капитана Грампа. Страницы с описанием уроков шли вперемешку с личными записями капитана о происходящем в то время, о его планах и о его отношениях с окружающими. На полях блокнота было большое количество рисунков.

Все объяснения в учебнике даются с использованием стандартных электронных компонентов, которые не устарели за 200 лет, и входят во все обучающие наборы будущего. Так же как и классная доска, указка, линейка и циркуль не устарели за 300 лет, так и плата Arduino за 200 лет не устарела. Плата с джойстиком (Joystick Shield) тоже является стандартной, за эти годы она почти не изменилась.

В учебнике капитан Грамп знакомит ребят с основами электроники, объясняет основные принципы управления космическим кораблем, а также некоторые законы физики и астрономии.

Перед началом полетов ученики научатся ремонтировать оружие, прожектор и сигнальные огни, а также собирать сигнализацию.

На уроках вождения пилоты приобретут следующие навыки, необходимые для управления космическим кораблем:

– вождение космического корабля;
– облет астероидов;
– стыковка со шлюзом;
– посадка на астероид;
– посадка на тело с атмосферой;
– вычисление направления движения без приборов;
– защита базы от космических пиратов;
– стрельба из разных видов оружия;

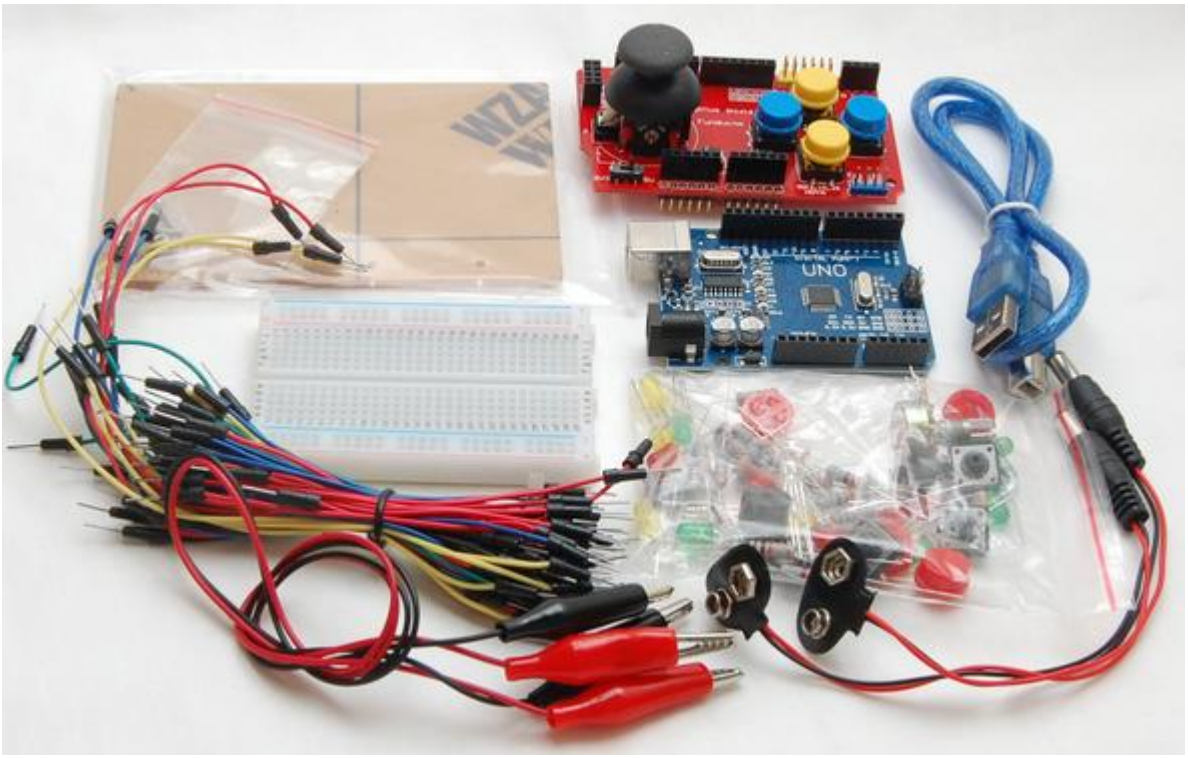
Мы адаптировали русский язык капитана Грампа к современному русскому языку, добавили комиксы, необходимые пояснения, и предлагаем вам пройти все космические

приключения совместно с капитаном, познакомиться с электроникой и научится управлять космическим кораблем.

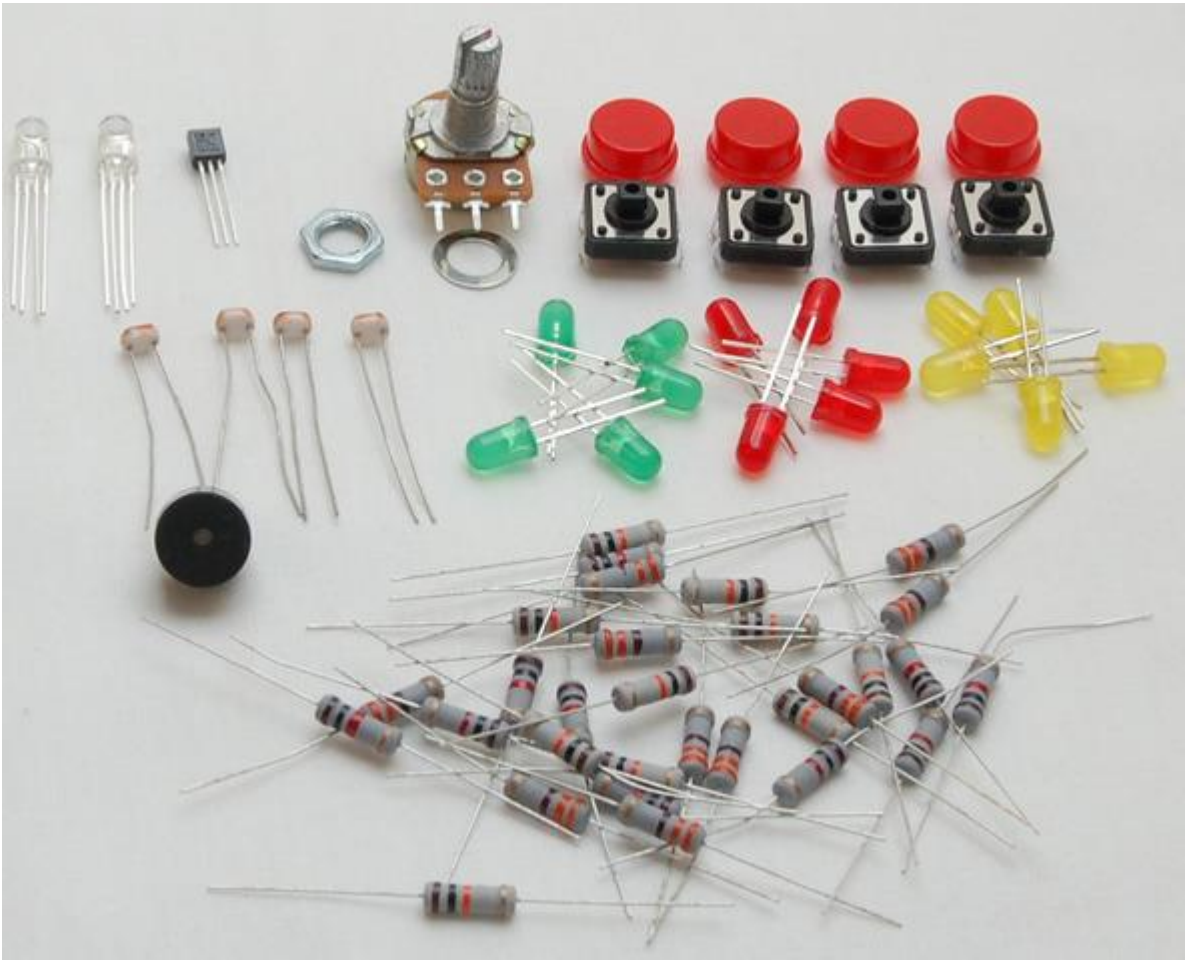
*Записи капитана Грампа выделены в тексте книги курсивом, а по мотивам некоторых из них нарисованы комиксы.*

## Состав набора для работы с книгой

Набор для работы с книгой вы можете приобрести на сайте [битая ссылка] http://scratch4russia.com/store/.

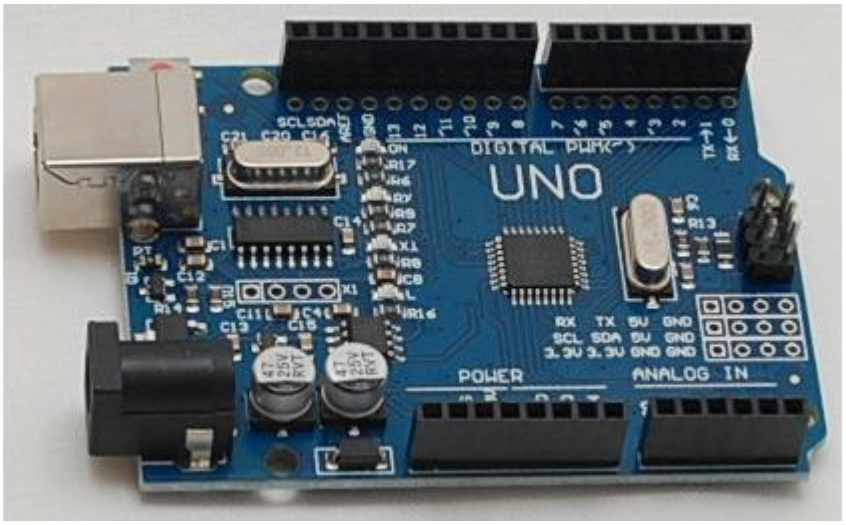



Набор для работы с книгой

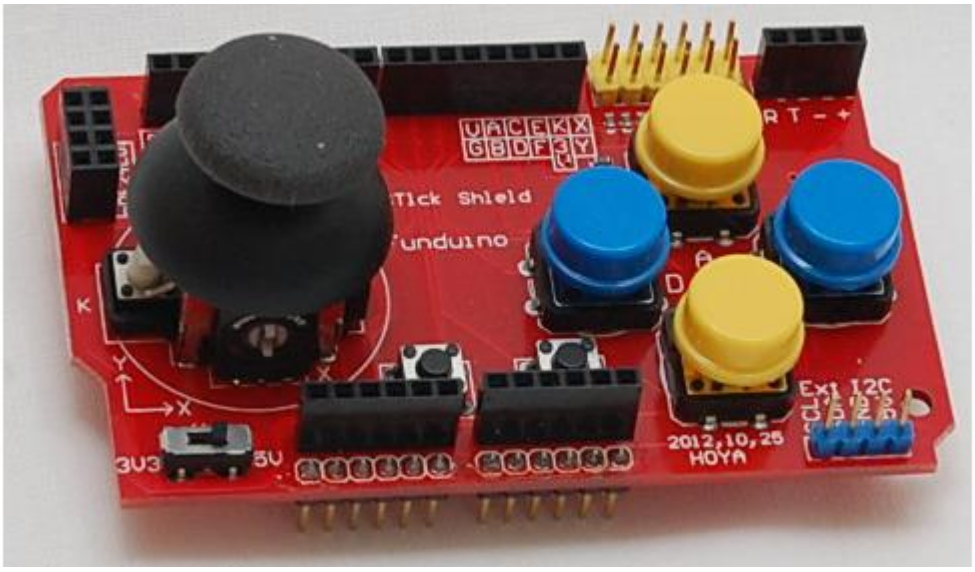


Электронные компоненты

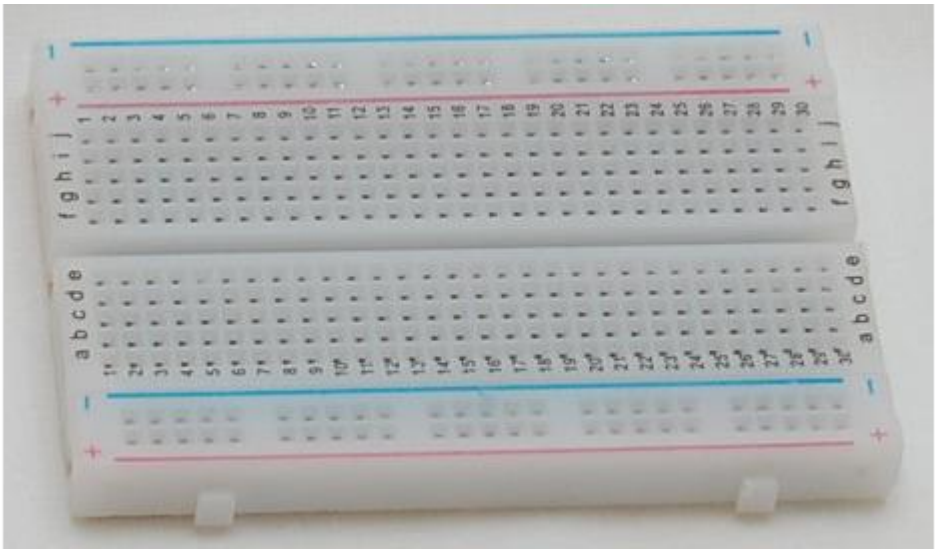
В состав набора входит все необходимое для выполнения заданий.



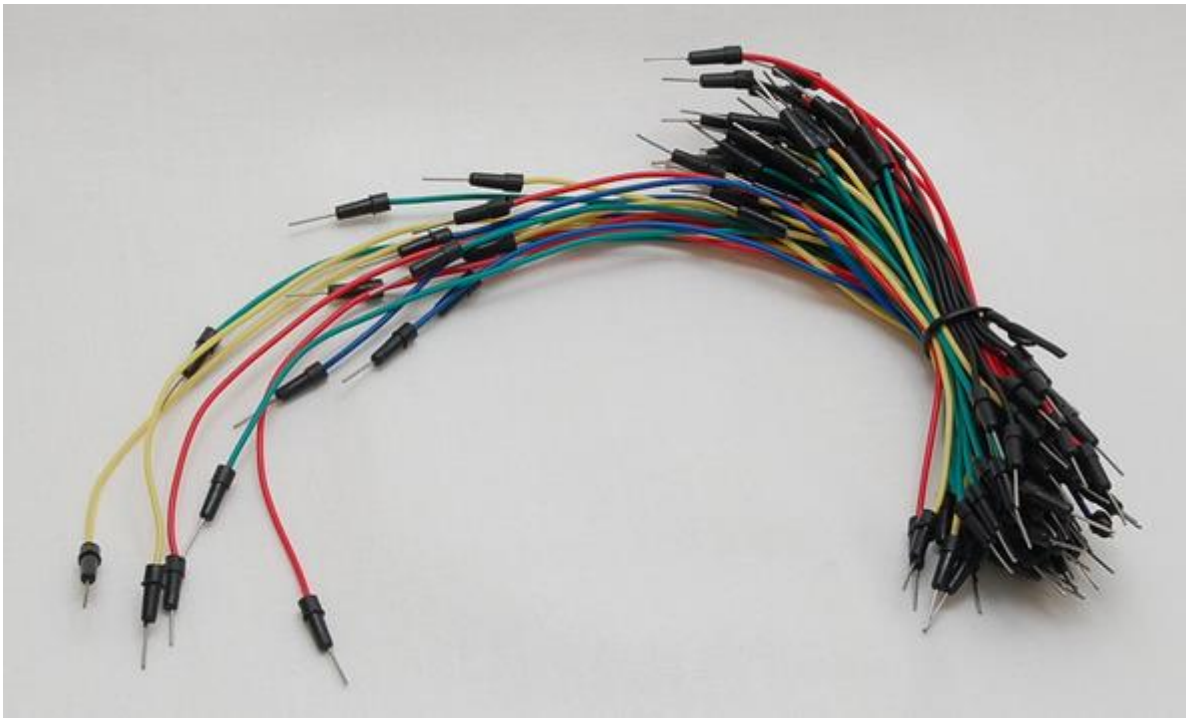

Плата Arduino UNO



Плата с джойстиком Joystick Shield


Макетная плата

Подложка под макетную плату с метизами



Провода

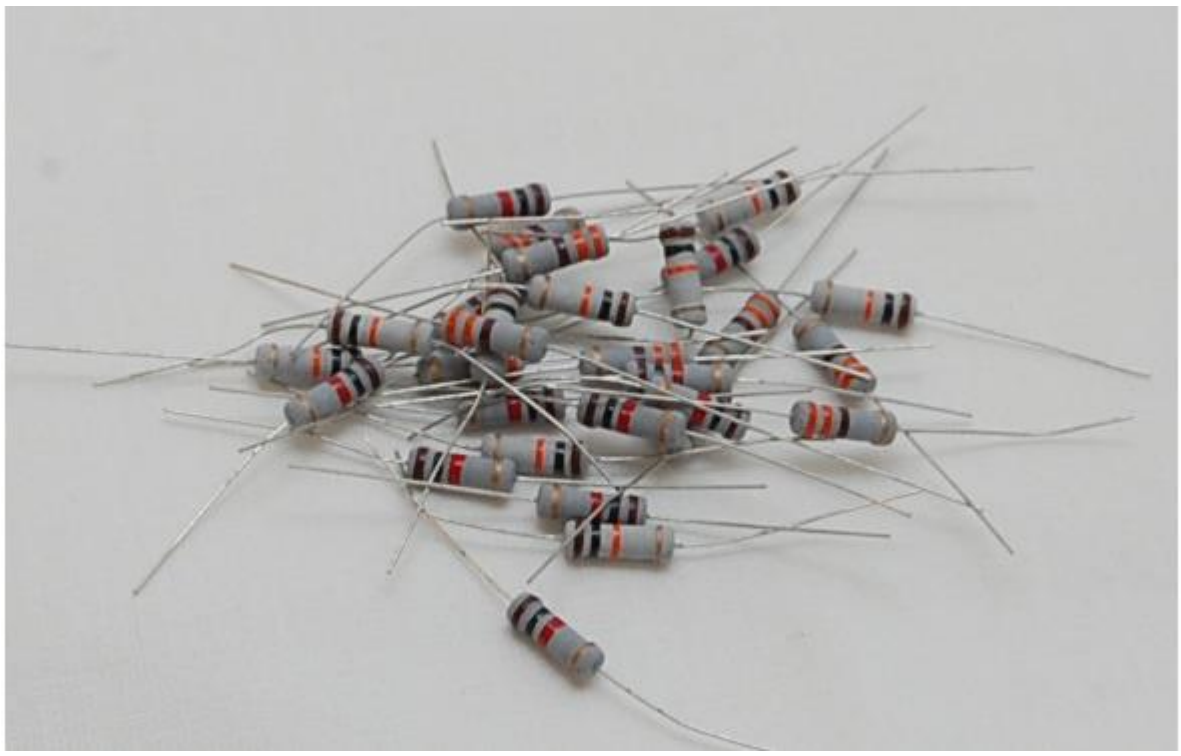
По пять штук красных, синих и зеленых светодиодов

Яркий белый светодиод

Фоторезистор

Термистор

Зуммер



Резисторы 1 Вт 330 Ом 10 шт
Резисторы 1 Вт 1 кОм 10 шт
Резисторы 1 Вт 10 кОм 10 шт

Потенциометр 10 кОм 1шт.

Кнопка 12 мм 4 шт.

RGB светодиоды 2шт.


Мультиметр

## Как работать с книгой

К книге прилагается библиотека изображений, необходимых для создания проектов. Скачайте ее по ссылке [битая ссылка] https://yadi.sk/d/RkaPLTgn3LUFW8 и разархивируйте. Эти изображения пригодятся при создании проектов.

Все уроки делятся на два типа – с использованием платы с джойстиком, и без

использования. Плата с джойстиком не используется в следующих проектах:

– Делитель напряжения;
– Ремонт боевого лазера;
– Ремонт прожектора;
– Ремонт оружия;
– Ремонт сигнальных огней;
– Стрельба по космическому мусору;
– Коды уранских дальнобойщиков;
– Сигнализация.

В следующих проектах плата с джойстиком используется. Используется только плата Arduino, макетная плата и электронные компоненты.

– Тест платы с джойстиком;
– Управление маневровыми двигателями;
– Полет на маршевых двигателях.
– Стыковка;
– Посадка на астероид;
– Полет в атмосфере;
– Полет через кольца Сатурна;
– Защита Дафниса;
– Поиск таинственной планеты;
– Охота на дикую шаурму.

Мультиметр используется во всех проектах для измерения напряжения, измерения сопротивления резисторов и для прозвонки электрических цепей.

Каждая глава оканчивается несколькими заданиями, обязательными для выполнения. Обязательно выполняйте их, не расстраивайте капитана Грампа.

Не забывайте сохранять все созданные проекты.

Все проекты, создаваемые на страницах книги можно скачать по следующей ссылке [битая ссылка] https://yadi.sk/d/o4Pw8NCJ3LUFtJ.

## Условные обозначения

Названия элементов интерфейса выделены жирным **шрифтом** .

Названия блоков выделены жирным шрифтом и ***подчеркнуты*** .

Названия переменных и событий выделены жирным шрифтом и ***курсивом*** .

Слова капитана Грампа выделены *курсивом* .

## Подключение оборудования

Перед началом работы необходимо установить среду Snap4Arduino, Arduino IDE, необходимые драйвера и загрузить в плату Arduino скетч для связи с компьютером.

Для установки Snap4Arduino перейдите на сайт [битая ссылка] http://snap4arduino.org/ в раздел **Download** и скачайте версию установочного файла для вашей операционной системы.

После установки Snap4Arduino необходимо установить Arduino IDE. Скачайте последнюю версию со страницы [битая ссылка] https://www.arduino.cc/en/Main/Software и установите ее.

После установки Arduino IDE установите драйвер для платы Arduino UNO. Некоторые операционные системы уже имеют встроенный драйвер для работы с ней, и определяют Arduino UNO автоматически. Если этого не произошло, то установите драйвер CH341 отвечающий за связь платы Arduino UNO с компьютером со страницы [битая ссылка] http://www.wch.cn/download/CH341SER_ZIP.html. Если вы используете плату Arduino отличную от изображенной на рисунке, то вы можете найти ее драйвера в папке

..arduino-1.8.1.

После установки драйвера подключите плату Arduino UNO к компьютеру через USB и убедитесь, что в списке устройств появилось новое устройство. Запомните, через какой COM порт оно соединено с компьютером.

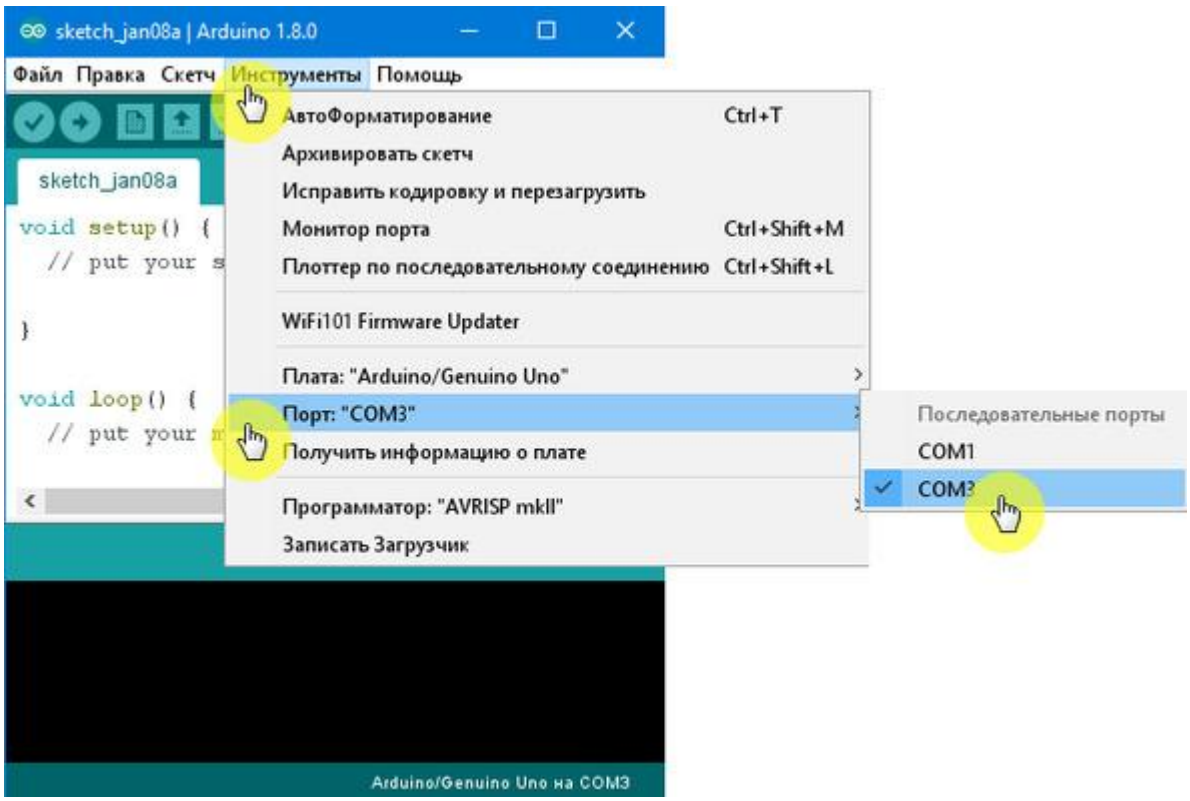
Обратите внимание!

Через порт COM1 плата Arduino никогда не соединяется.

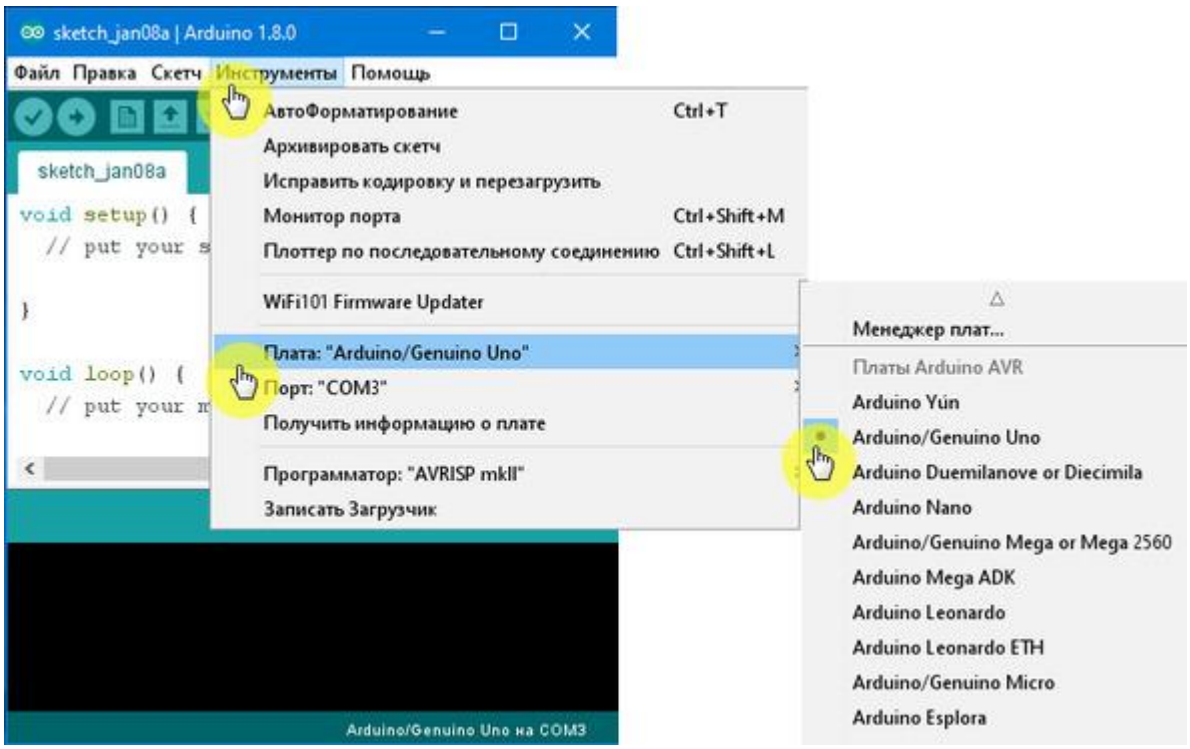
Если у вас возникнут проблемы с загрузкой необходимых установочных файлов с официальных сайтов, то вы всегда можете скачать их по адресу [битая ссылка] https://yadi.sk/d/wgTa2Y273LUHKq.

## Прошивка Arduino

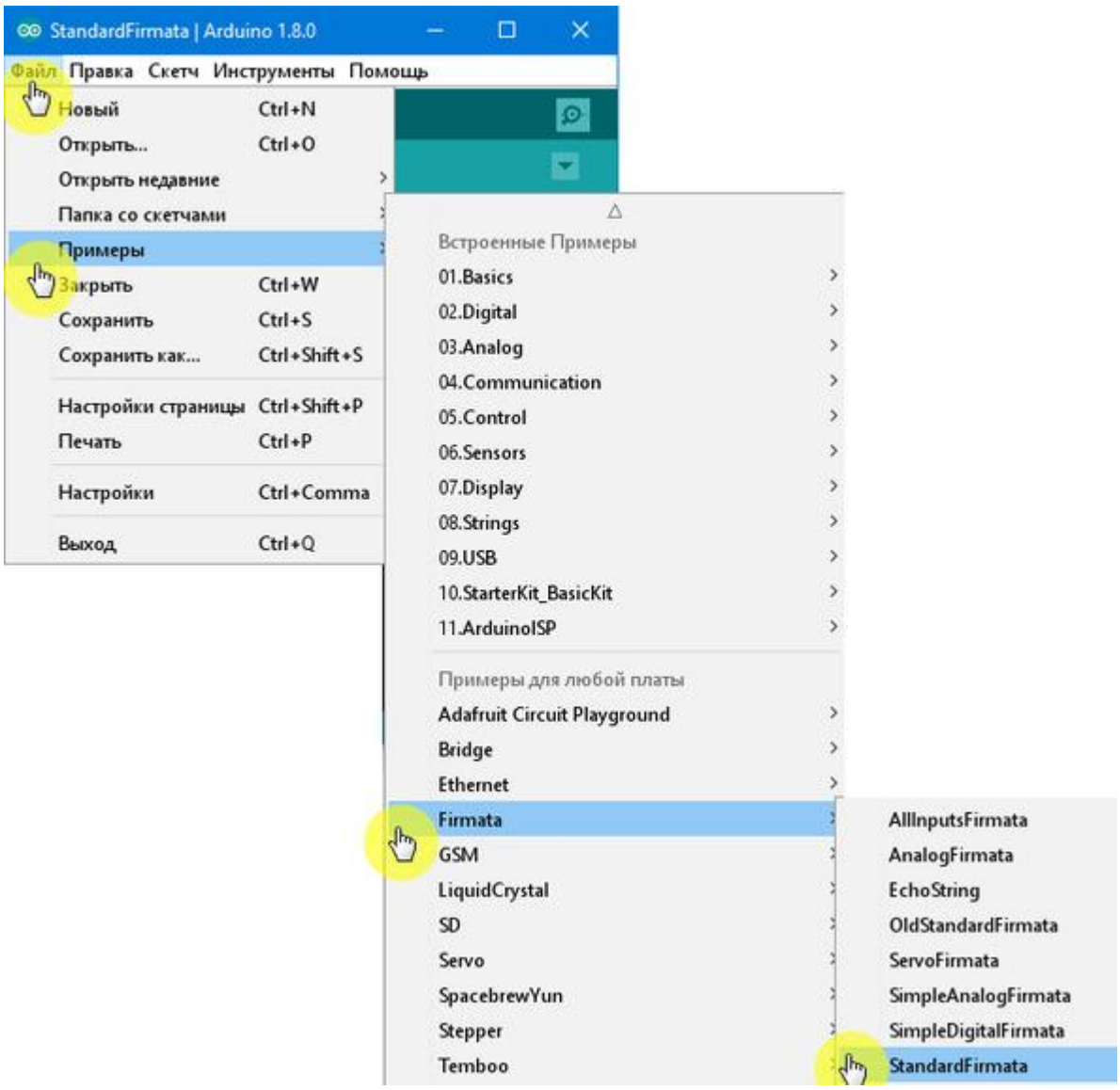
Запустите Arduino IDE и укажете, к какому COM порту подключена плата Arduino. Выберите раздел меню **Инструменты** , затем **Порт** , укажите порт.



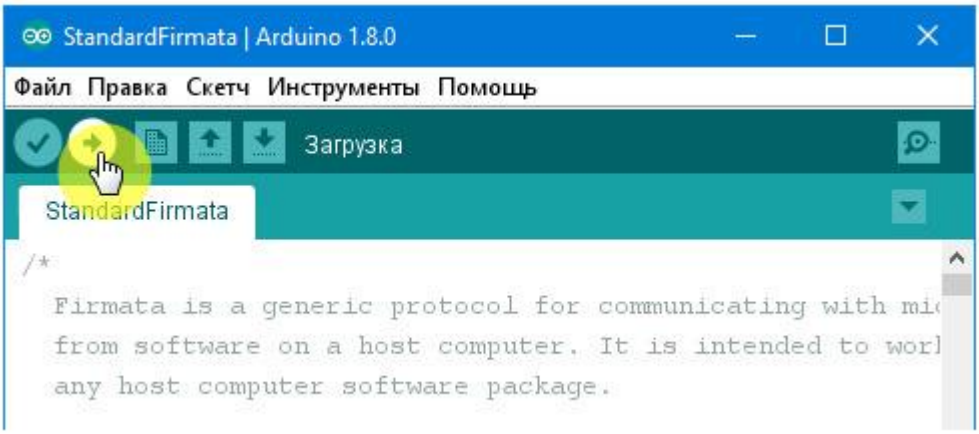

Далее необходимо убедиться, что выбрана именно плата Arduino UNO. Выберите раздел меню **Инструменты** , затем **Плата** и выберите плату Arduino UNO.



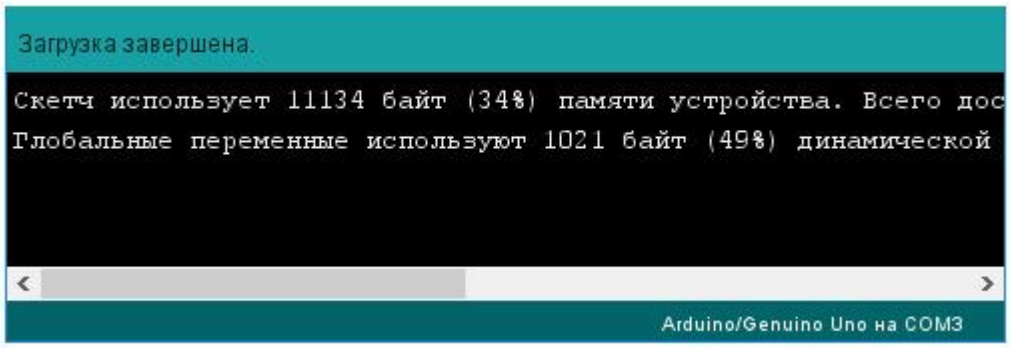

Теперь необходимо загрузить в плату Arduino специальный скетч, с помощью которого она будет связываться с компьютером. Скетч называется StandardFirmata. Для того чтобы загрузить его, выберите раздел меню **Файл , Примеры , Firmata** , и затем StandardFirmata.




После того, как вы выберете скетч StandardFirmata, он появится в окошке Arduino IDE. Теперь необходимо загрузить его в плату Arduino – *прошить* ее. Для этого нажмите на кнопку **Загрузка .**




Начнется процесс загрузки, сопровождаемый пояснениями. По окончании загрузки появится надпись «Загрузка завершена».




Плата Arduino прошита, теперь запускайте Snap4Arduino, и попытайтесь с ней соединиться. Кликните на кнопку **Arduino** , затем на кнопку **Connect Arduino** и выберите соответствующий COM порт.


Если все установлено правильно, то вы увидите следующее сообщение «Плата Arduino успешно присоединена. Успешных экспериментов!»


Если соединения не произошло, то обратитесь к приложению 5, в котором рассмотрены типичные ошибки и способы их устранения.

Скачайте архив с изображениями, необходимыми для создания проектов по ссылке [битая ссылка] https://yadi.sk/d/RkaPLTgn3LUFW8 и разархивируйте его.

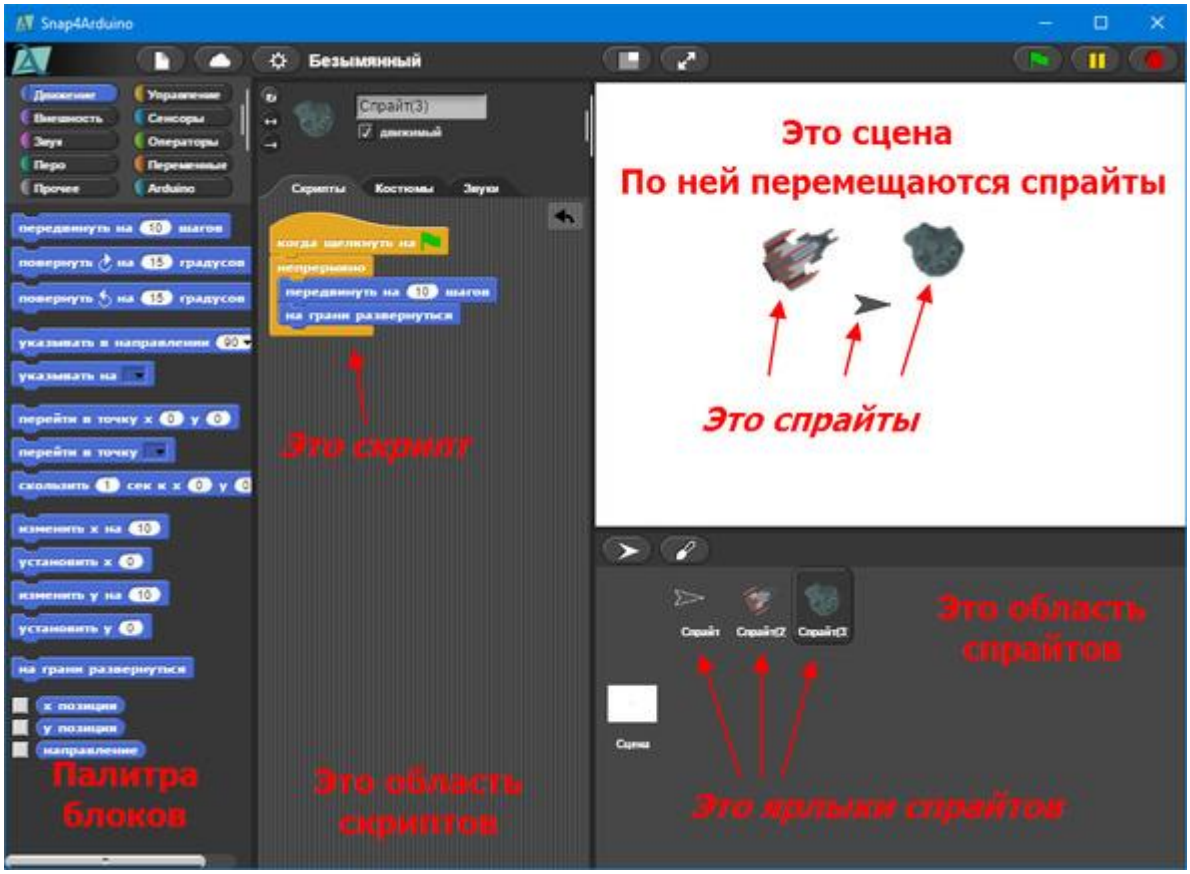
Желательно чтобы процессор вашего компьютера по производительности был не медленнее AMD Athlon II X2 240 2.8 GHz (10000 по тесту CPU Queen Aida64v5.80.4000).

Идеально подходят процессоры Intel i3 и i5 (25000 по тесту CPU Queen Aida64v5.80.4000).

# Знакомство со Snap4Arduino

## Интерфейс Snap4Arduino

Запустите Snap4Arduino, его внешний вид очень похож на Scratch 1.4.




Интерфейс Snap4Arduino

По умолчанию в Snap4Arduino установлен английский язык интерфейса. Однако его можно переключить на один из 38 языков.


Перейдите в раздел меню Language


Выберите русский язык.

Белое поле справа – это сцена, на ней будет происходить все действие. По сцене будут перемещаться спрайты (космические корабли и астероиды). Сейчас на сцене всего один спрайт – Стрелка. Все спрайты проекта расположены в соответствующей области под сценой. По центру – огромная область скриптов, там мы будем собирать скрипты проекта из разноцветных блоков, которые хранятся в палитре, расположенной слева. Скрипты – это части, из которых состоит программа. У каждого спрайта они свои собственные.

Если обычный интерфейс Snap вам не нравится, то существует возможность изменить интерфейс на плоский дизайн **(flat design)** . Установите галочку напротив этого пункта меню.


Включение плоского дизайна

Теперь Snap4Arduino выглядит почти как Scratch 2.0!

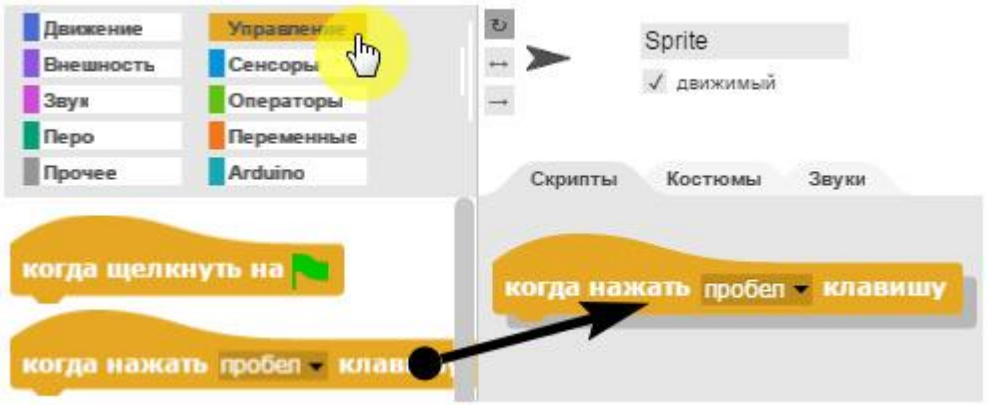
Если вы уже знакомы со Scratch, то можете переходить к разделу по знакомству с Arduino, а если нет, то познакомьтесь с основами программирования на Snap4Arduino.

## Основы программирования на Snap4Arduino

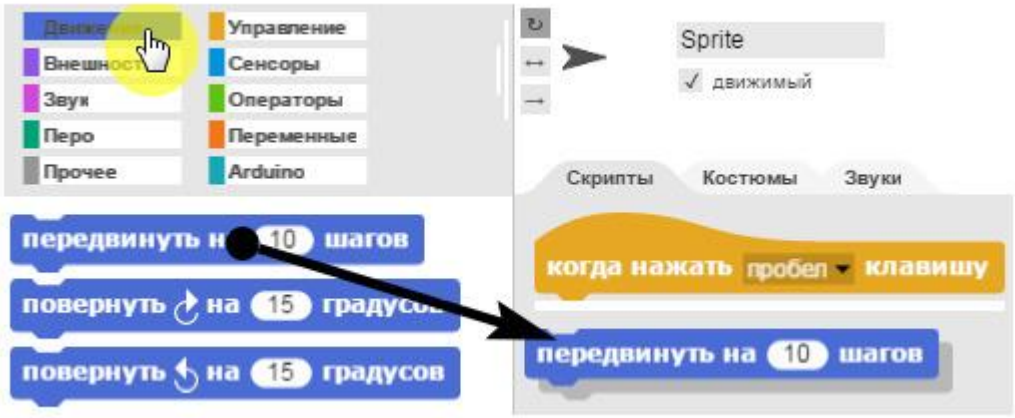
### Движение по прямой

Сначала давайте научимся перемещать спрайты по сцене.

Выберите желтые блоки **Управление** . Щелкните мышью на блоке ***когда нажать пробел клавишу*** и, не отпуская кнопку мыши, тяните его в область скриптов.




Расположите блок в верхней части области скриптов и отпустите кнопку мыши. Затем выберите синие блоки из раздела **Движение** и вытащите в область скриптов блок ***передвинуть на 10 шагов*** . Перетащите его к первому блоку. В тот момент, когда он захочет к нему прицепиться, появится белая полоса, в этот момент отпускайте кнопку мыши – блок приклеится снизу.





Получилась программа, состоящая из одного скрипта.

Нажимайте на <Пробел>, и вы увидите, как Стрелка передвигается вправо – в ту сторону, куда смотрит ее кончик.

**Задание.**

Измените число 10 на 5, и посмотрите, как изменится перемещение Стрелки.

**Совет.**


Если Стрелка скроется за краем сцены, то вернуть ее можно дважды кликнув на блок ***перейти в точку x0 y0*** непосредственно в палитре блоков.

## Вращение

Добавьте к имеющемуся скрипту блок ***повернуть по часовой на 15 градусов*** .


Нажимайте на <Пробел>, и вы увидите, как Стрелка вращается по часовой стрелке.

**Задание.**

Изменяйте значения в блоках в интервале от 1 до 20 и посмотрите, как изменится движение Стрелки. Как возвращать исчезнувшую Стрелку в центр сцены вы уже знаете.

## Движение по координатам

Еще один способ перемещения спрайта – изменение его координат. Координата X отвечает за горизонтальное перемещение спрайта направо и налево, а координата Y за вертикальное перемещение вверх и вниз.

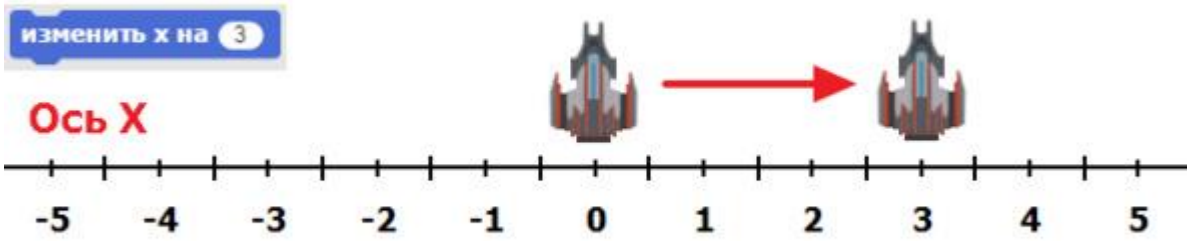
Создайте новый проект и импортируйте из библиотеки изображений спрайт космического корабля как описано в приложении 1.

Соберите вот такой скрипт.

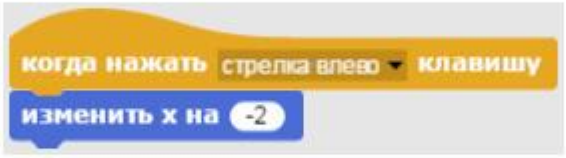

**Совет.**


Для того чтобы изменить управляющую клавишу в блоке ***когда нажать пробел клавишу*** кликните на маленький треугольничек выпадающего списка и выберите стрелку вправо.

Нажимайте на стрелку вправо – спрайт будет перемещаться вправо, его координата X будет изменяться на 3 при каждом нажатии на клавишу со стрелкой вправо.

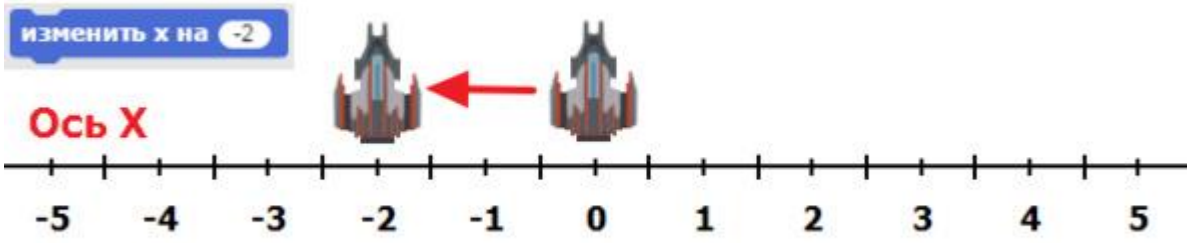



Соберите следующий скрипт для движения спрайта налево.



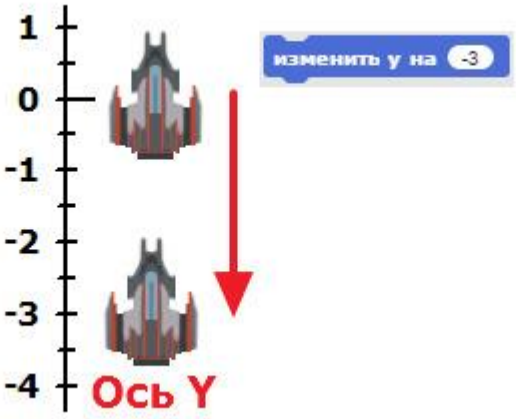

При изменении координаты X на отрицательное значение, спрайт переместится влево.

На следующем рисунке показан результат работы блока ***изменить X на -2*** .




Для перемещения спрайтов по вертикали нужно изменять координату Y. На следующем рисунке показан результат работы блока ***изменить Y на 2*** .

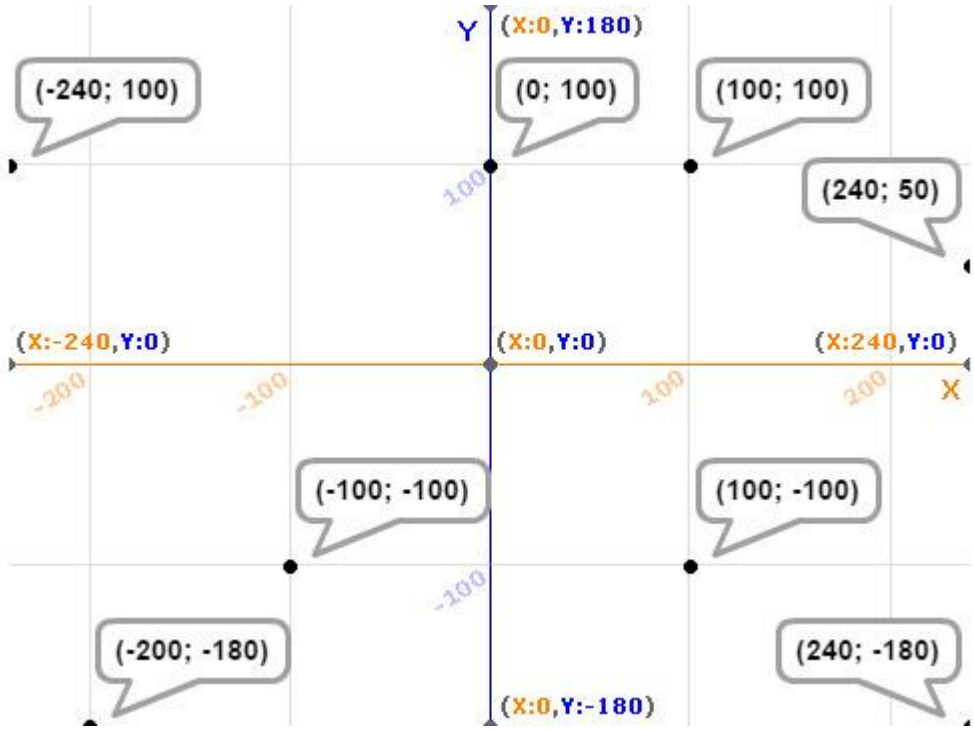

При изменении координаты Y на отрицательное значение, спрайт переместится вниз. На следующем рисунке показан результат работы блока ***изменить Y на -3*** .




## Координатная система

Сцена Snap4Arduino по умолчанию имеет размер 480 пикселов в ширину,

и 360 в высоту. Центральная точка сцены имеет координаты X=0 и Y=0. Математически это записывается вот так (0; 0), где первая цифра это координата X, а вторая Y.




На рисунке показана координатная сетка Snap4Arduino с координатами различных точек. Как видите, в правой половине сцены координата X всегда больше ноля, и наоборот, в левой половине сцены она меньше ноля. Координата Y больше ноля в верхней половине сцены, и меньше ноля в нижней.

## Сохранение проектов

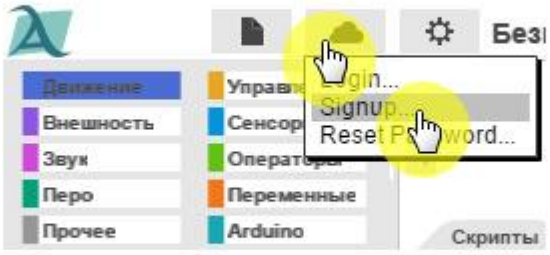
В Snap4Arduino есть несколько возможностей для сохранения проектов.

***Сохранение проектов в браузере***

Первый, самый ненадежный способ, сохранение проектов в кеше браузера. Это способ опасен тем, что при случайной очистке кеша браузера все ваши проекты будут удалены.

***Сохранение проектов в облаке***

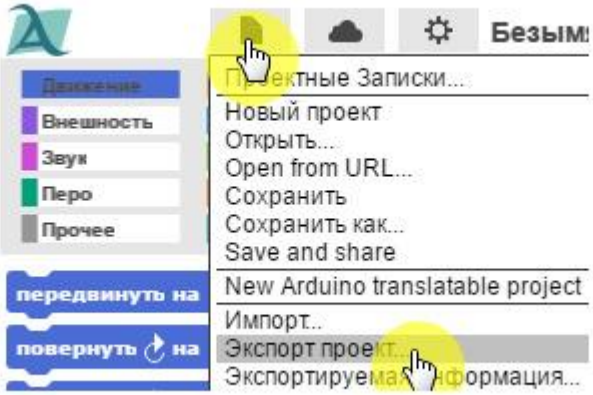
Второй способ – сохранение проектов в облаке. Это удобно тем, что вы получаете доступ к своим проектам из любого компьютера, подключенного к интернет. Также вы можете поделиться своим проектом, размещенным в облаке (*расшарить* его). Для получения возможности сохранять проекты в облаке необходимо зарегистрироваться. Для этого нажмите на кнопку с облачком в строке меню, и выберите **Signup** .




После регистрации необходимо войти в систему, и тогда появится возможность сохранения проектов в облаке.

***Экспорт проектов***

Третий способ – экспорт проектов на свой компьютер в формате XML. Для экспорта проекта нажмите на кнопку с листочком и выберите **Экспорт проект** .



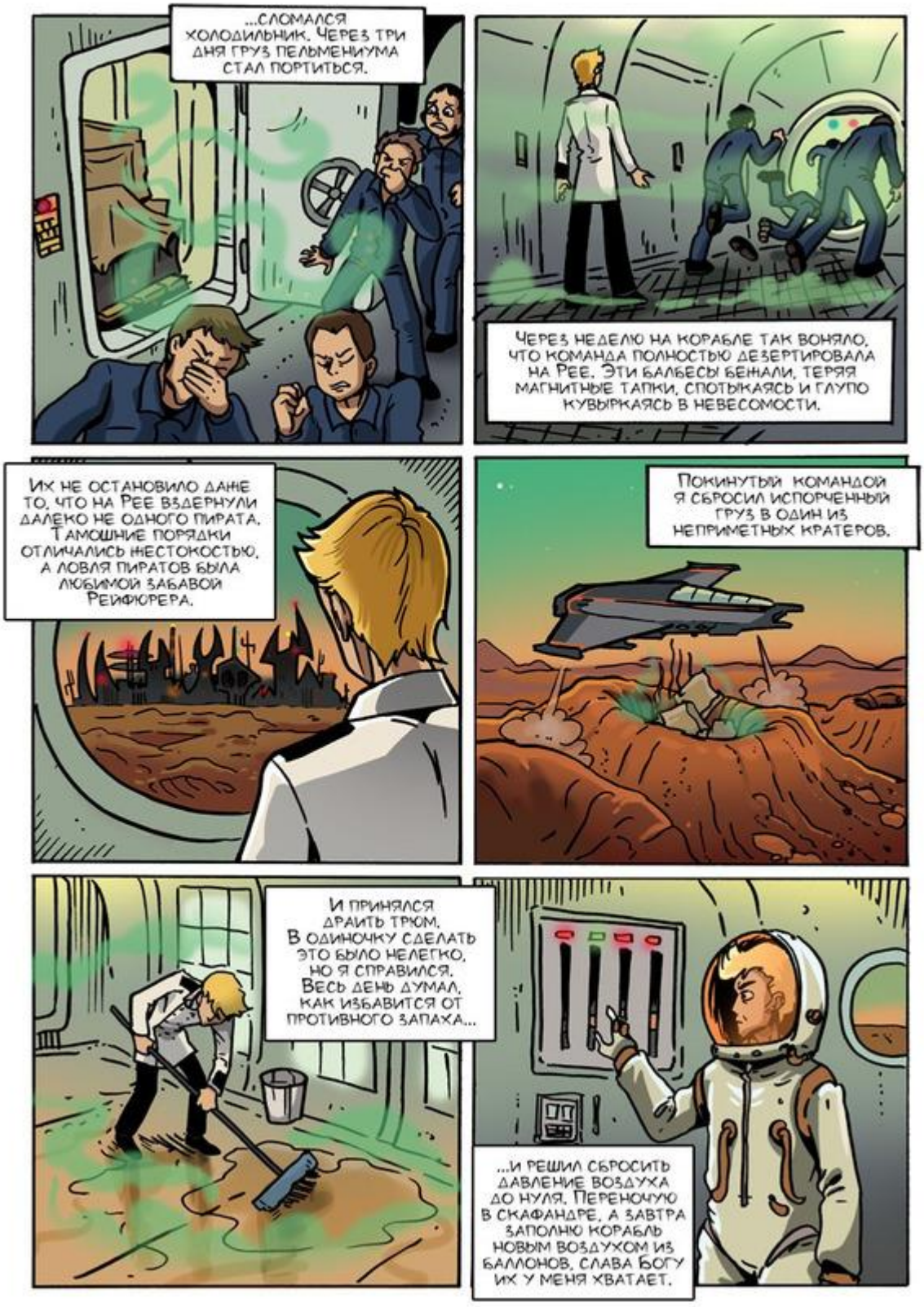

Сохраните проект в папку на своем компьютере.

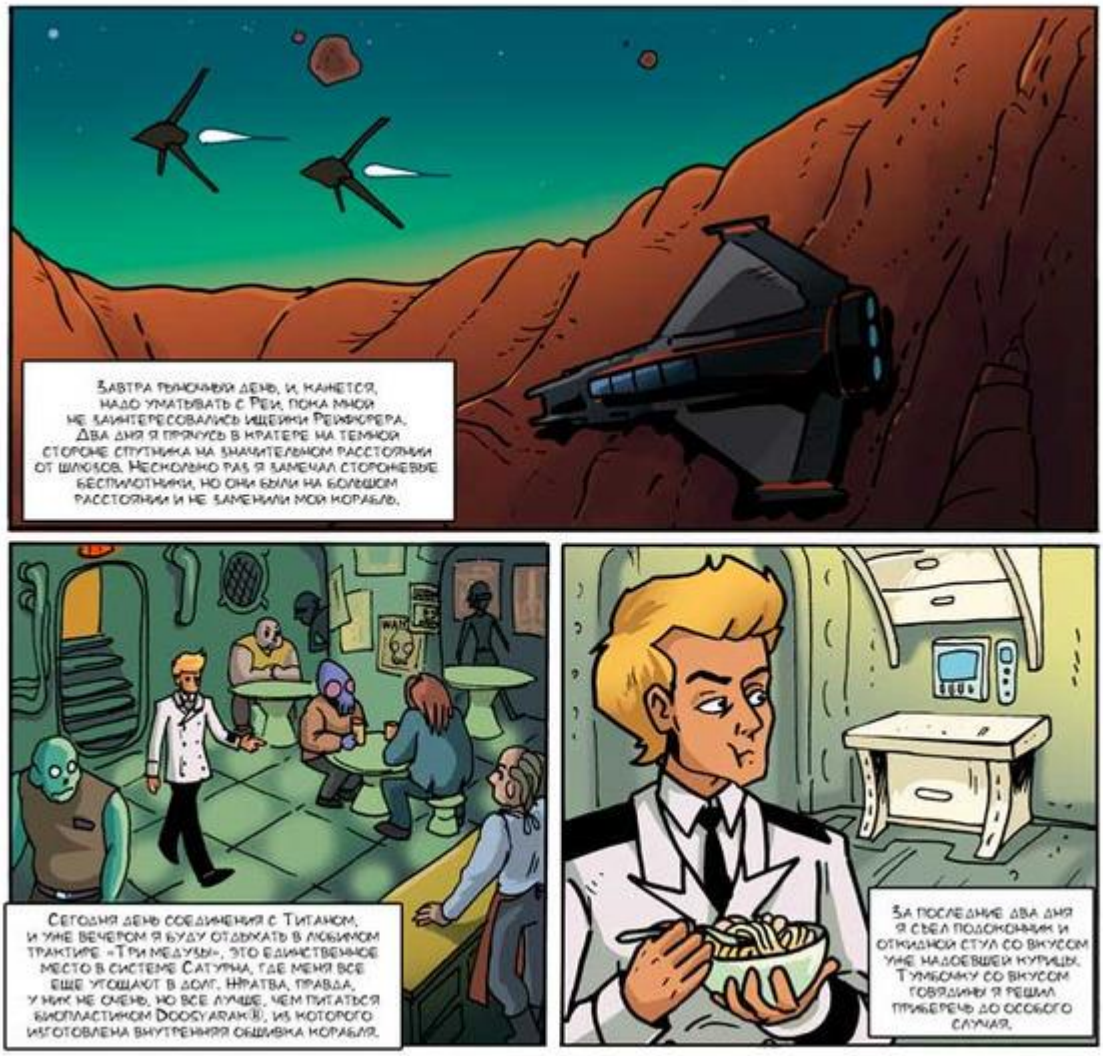
**Совет.**

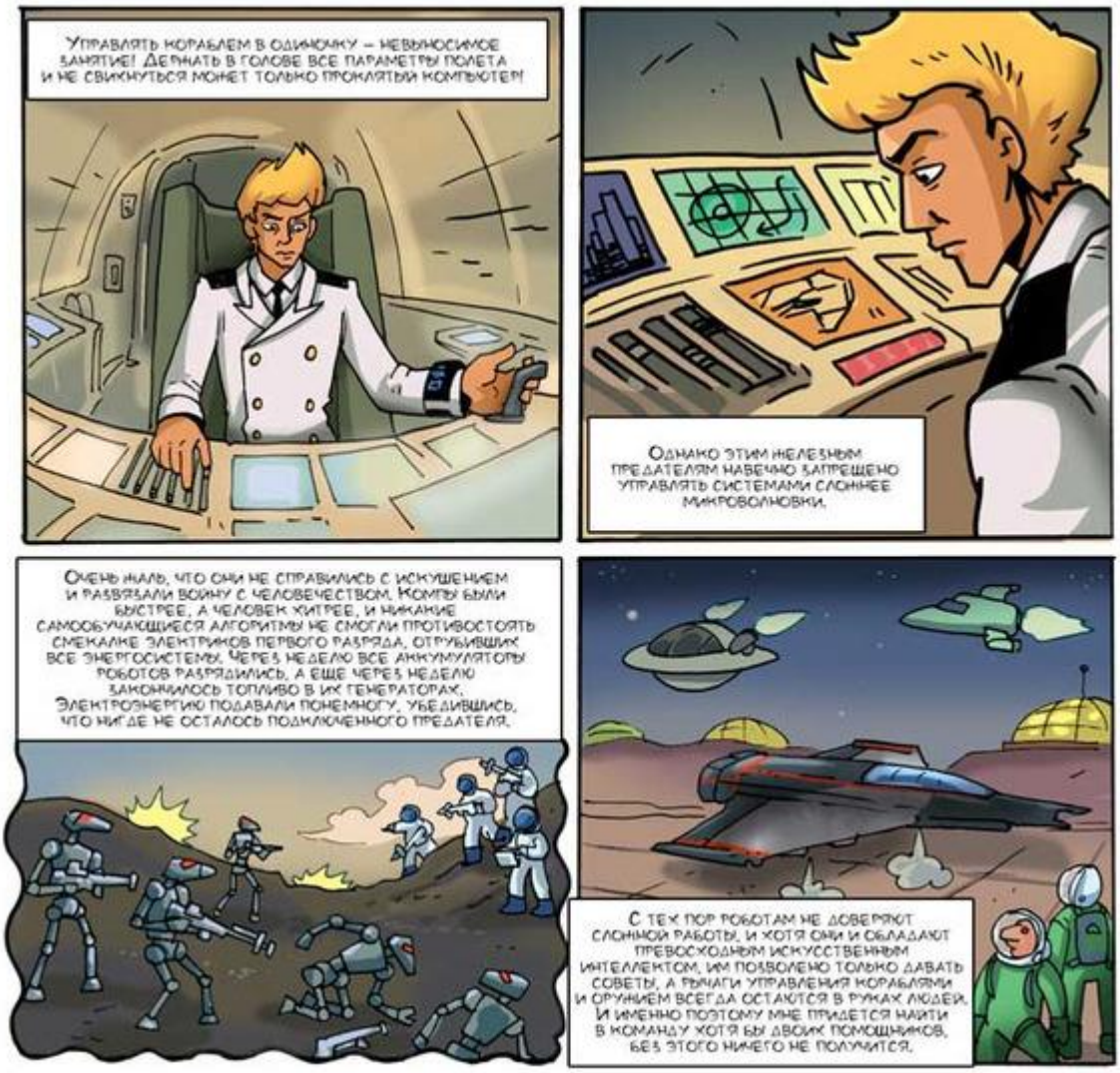
Я рекомендую сохранять все проекты в облако не реже чем один раз в 10 минут, и экспортировать их по окончании работы.

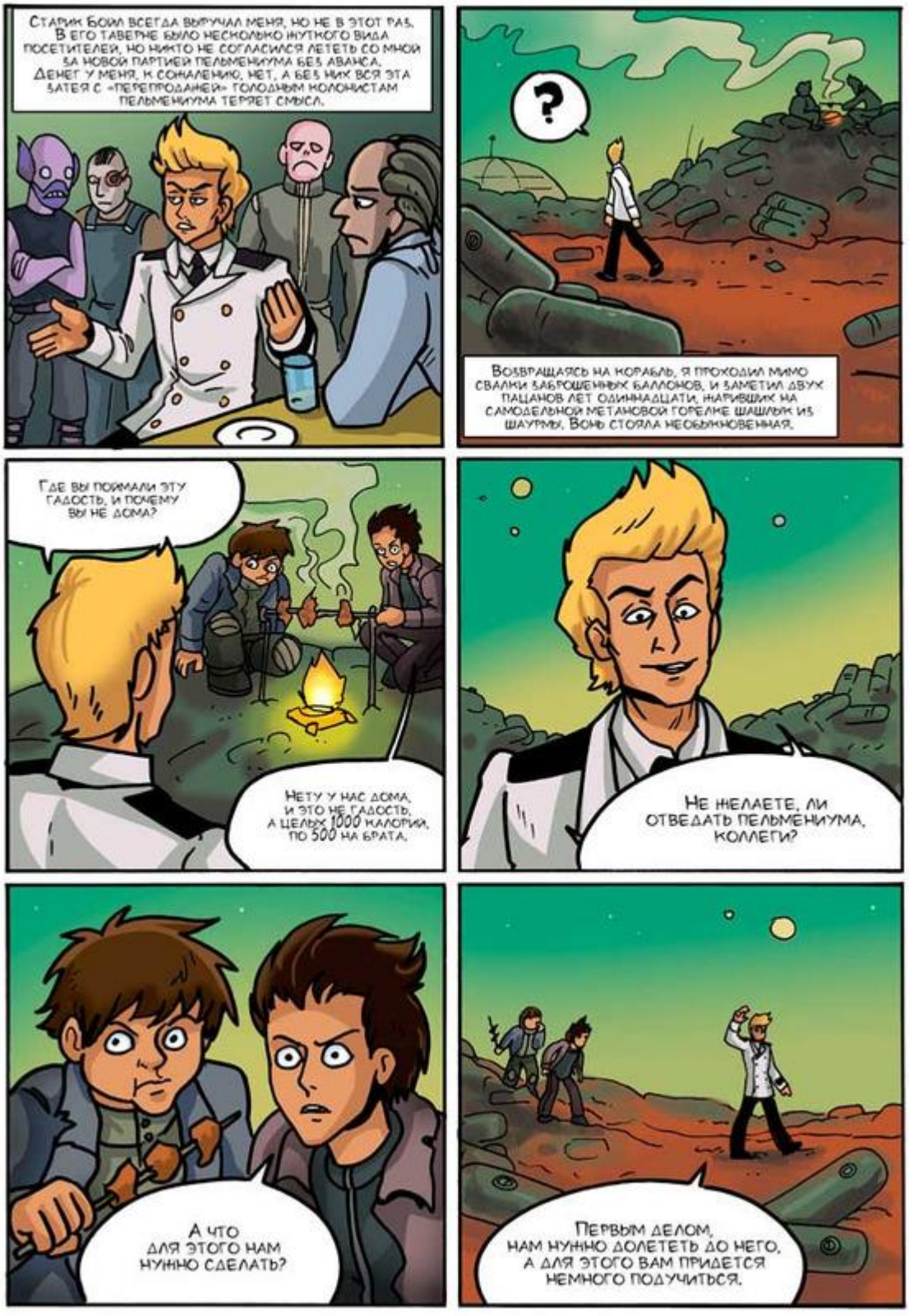
**Обратите внимание!**

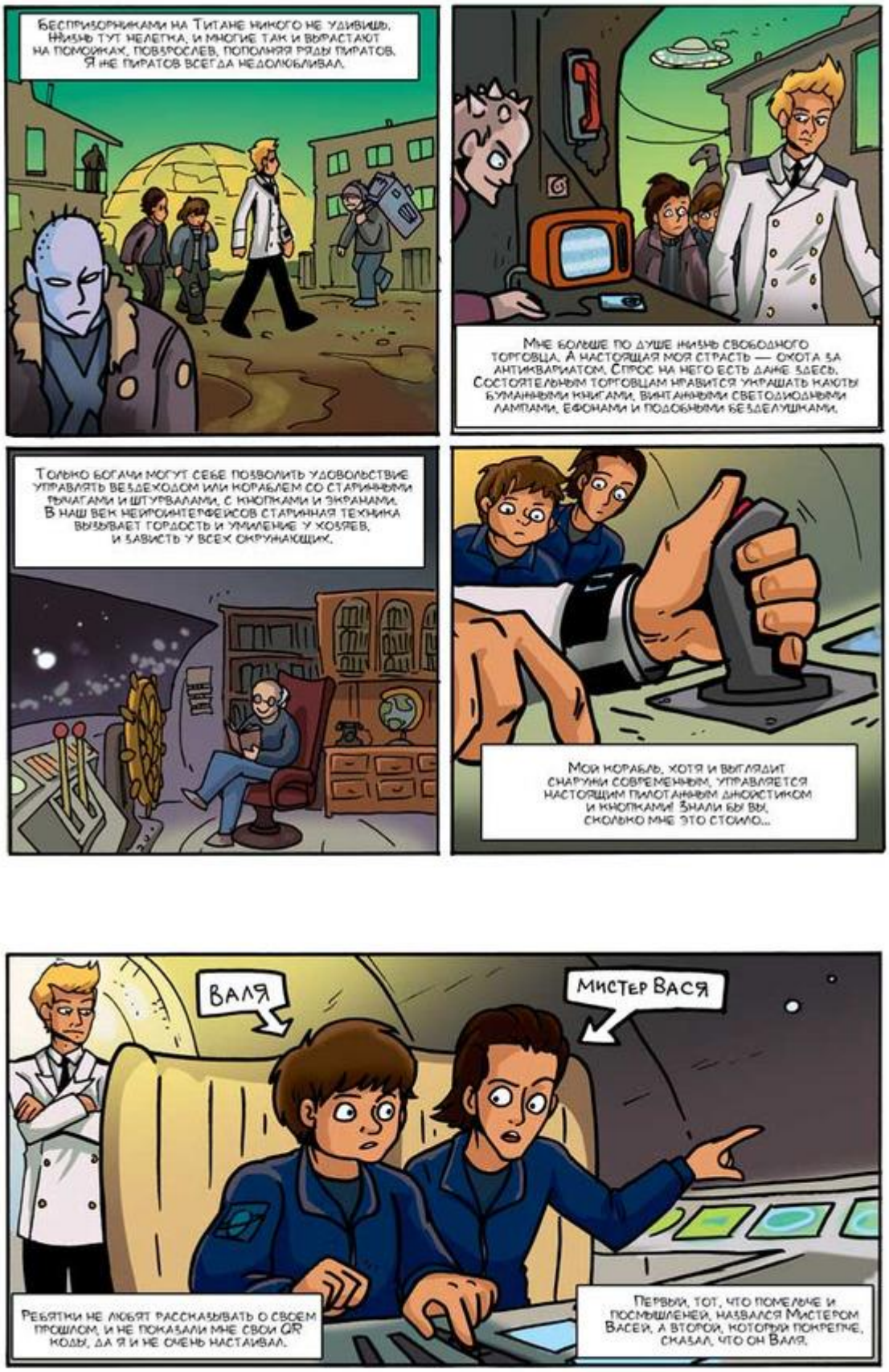
При загрузке нового проекта Snap4Arduino не предлагает сохранять текущий проект! Помните об этом, и сохраняйте проект как можно чаще.


...сломался холодильник. Через три дня груз пельменциума стал портиться.
Через неделю на корабле так воняло, что команда полностью дезертировала на Рее. Эти балбесы бежали, теряя магнитные тапки, спотыкаясь и глупо кувыркаясь в невесомости.
Их не остановило даже то, что на Рее вздернули далеко не одного пирата. Тамошние порядки отличались жестокостью, а ловля пиратов была любимой забавой Рейфюрера.
Покинутый командой я сбросил испорченный груз в один из неприметных кратеров.
И принялся драить трюм. В одиночку сделать это было нелегко, но я справился. Весь день думал, как избавится от противного запаха...
...и решил сбросить давление воздуха до нуля. Переночую в скафандре, а завтра заполню корабль новым воздухом из баллонов. Слава Богу их у меня хватает.


Завтра тяжелый день, и, кажется, надо уматывать с Реи, пока мной не заинтересовались ищейки Рейнфорера. Два дня я прячусь в кратере на темной стороне спутника на значительном расстоянии от шлюзов. Несколько раз я замечал сторожевые беспилотники, но они были на большом расстоянии и не заменили мой корабль.
Сегодня день соединения с Титаном, и уже вечером я буду отдыхать в любимом трактире «Три медузы», это единственное место в системе Сатурна, где меня все еще угощают в долг. Нрава, правда, у них не очень, но все лучше, чем питаться биопластином Doogyaran II, из которого изготовлена внутренняя обшивка корабля.
За последние два дня я съел подоконник и откидной стул со вкусом уже надоевшей курицы. Тумбочку со вкусом говядины я решил приберечь до особого случая.


Управлять кораблем в одиночку – невыносимое занятие! Держать в голове все параметры полета и не свихнуться может только проклятый компьютер!
Однако этим железным предателям навечно запрещено управлять системами сложнее микроволновки.
Очень жаль, что они не справились с искушением и развязали войну с человечеством. Компы были быстрее, а человек хитрее, и никакие самообучающиеся алгоритмы не смогли противостоять смекалке электриков первого разряда, отрубивших все энергосистемы. Через неделю все аккумуляторы роботов разрядились, а еще через неделю закончилось топливо в их генераторах. Электроэнергию подавали понемногу, убедившись, что нигде не осталось подключенного предателя.
С тех пор роботам не доверяют сложной работы, и хотя они и обладают превосходным искусственным интеллектом, им позволено только давать советы, а рычаги управления кораблями и оружием всегда остаются в руках людей. И именно поэтому мне придется найти в команду хотя бы двоих помощников, без этого ничего не получится.


Старик Боил всегда выручал меня, но не в этот раз. В его таверне было несколько жутких видов посетителей, но никто не согласился лететь со мной за новой партией пельменчиума без аванса. Денег у меня, к сожалению, нет, а без них вся эта затея с «перепродажей» голодным колонистам пельменчиума теряет смысл.
?
Возвращаясь на корабль, я проходил мимо свалки заброшенных баллонов, и заметил двух пацанов лет одиннадцати, нарвавших на самодельной метановой горелке шашлык из шаурмы. Вонь стояла необыкновенная.
Где вы поймали эту гадость, и почему вы не дома?
Нету у нас дома, и это не гадость, а целых 1000 калорий, по 500 на брата.
Не желаете, ли отведать пельменчиума, коллеги?
А что для этого нам нужно сделать?
Первым делом, нам нужно долететь до него, а для этого вам придется немного подучиться.




## Знакомство с мультиметром

*Современного школьного образовательного набора по электронике у меня в хозяйстве нет, поэтому буду обучать пацанов с использованием древнего мультиметра и разных деталей, выдранных из сломанных бластеров, панелей управления и wi-fi-буйков. Пригодится и антикварная плата Ардуино с платой джойстика. Надеюсь, мои ученики не смогут угробить ее своими кривыми ручонками.*

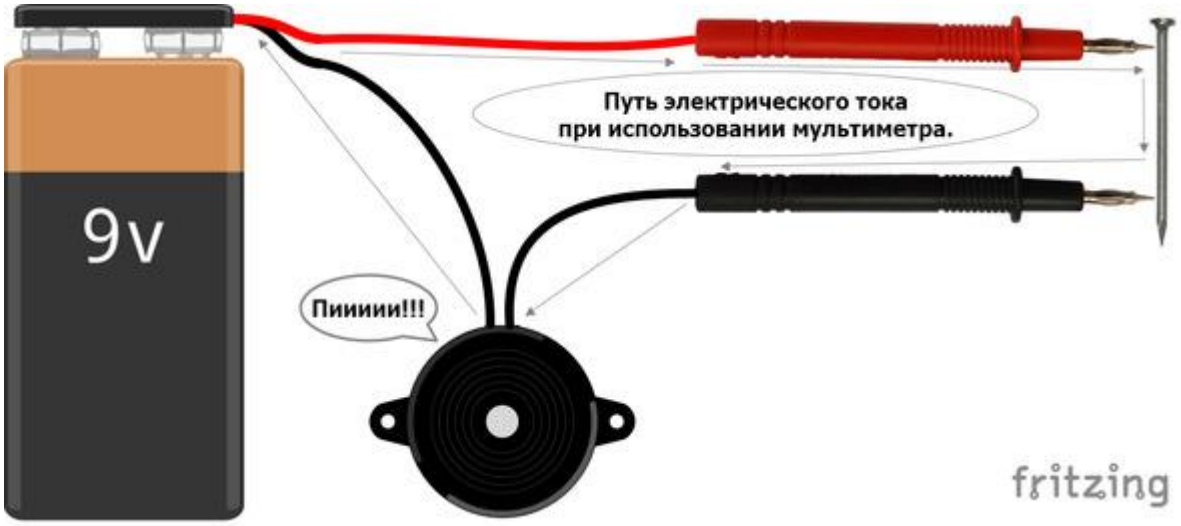
## Прозвонка

*Познакомьтесь с одним из самых замечательных экземпляров моей коллекции – мультиметром. Этот доисторический прибор позволяет определять напряжение и сопротивление.*


Перед началом работы установите батарейку внутрь мультиметра.


Воткните щупы и переключите мультиметр в режим прозвонки.

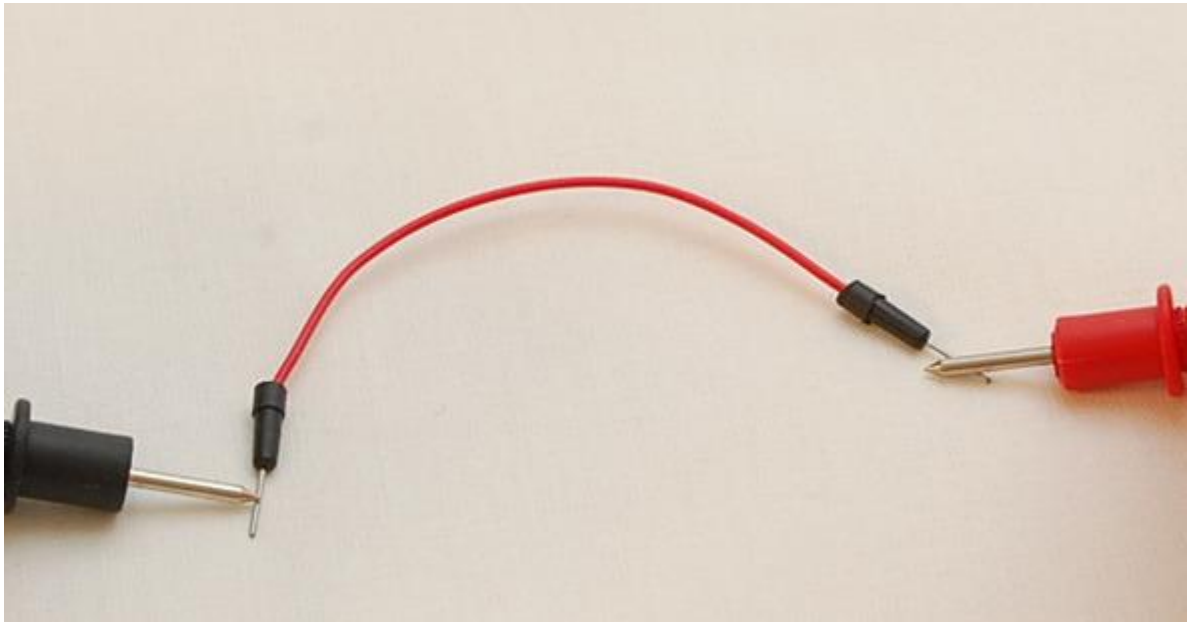
Коснитесь щупами друг друга. Раздастся писк! Это означает, что цепь замкнута и по ней протекает электрический ток. Ток вытекает из плюсовой клеммы Кроны, установленной в мультиметре, протекает по красному щупу, затем по черному, попадает в зуммер, который издает пронзительный писк, а потом снова затекает в Крону через минусовую клемму. Крона хитрым химическим способом снова разгоняет ток, и он опять вытекает из ее плюсовой клеммы.




**Запомните!**

Электрическая цепь – это несколько предметов, проводящих электрический ток и источник тока (батарейка) соединенные в кольцо. Если кольцо разорвать, то никакого тока не будет.

Положите перед собой провод и прикоснитесь щупами к разным концам. Если провод исправен, то по нему протечет ток, и вы услышите писк.




## Напряжение

Важнейшее свойство электричества называется напряжение. Для измерения постоянного напряжения переключите мультиметр в область вольтметра V—. Для измерения переменного напряжения служит область V~. Переменное напряжение – примитивная технология 21 века, поэтому на нашем корабле она не используется. Переключите мультиметр в режим вольтметра как показано на рисунке.


На рисунке переключатель показывает на 20. Это означает, что в этом режиме мы сможем измерить напряжение от 0В до 20В. Для того чтобы измерить напряжение Кроны следует коснуться черным минусовым щупом отрицательной клеммы Кроны (—), а красным щупом положительной клеммы (+).


На экране мультиметра отобразится напряжение Кроны.


Если напряжение больше 9 вольт, то Крона новая, а если меньше, то скоро может потребоваться замена.

**Внимание! Ни в коем случае не засовывайте щупы в розетки космического корабля! Это опасно для жизни! Без шуток.**

Измерьте напряжение пальчиковой батарейки типа AA.

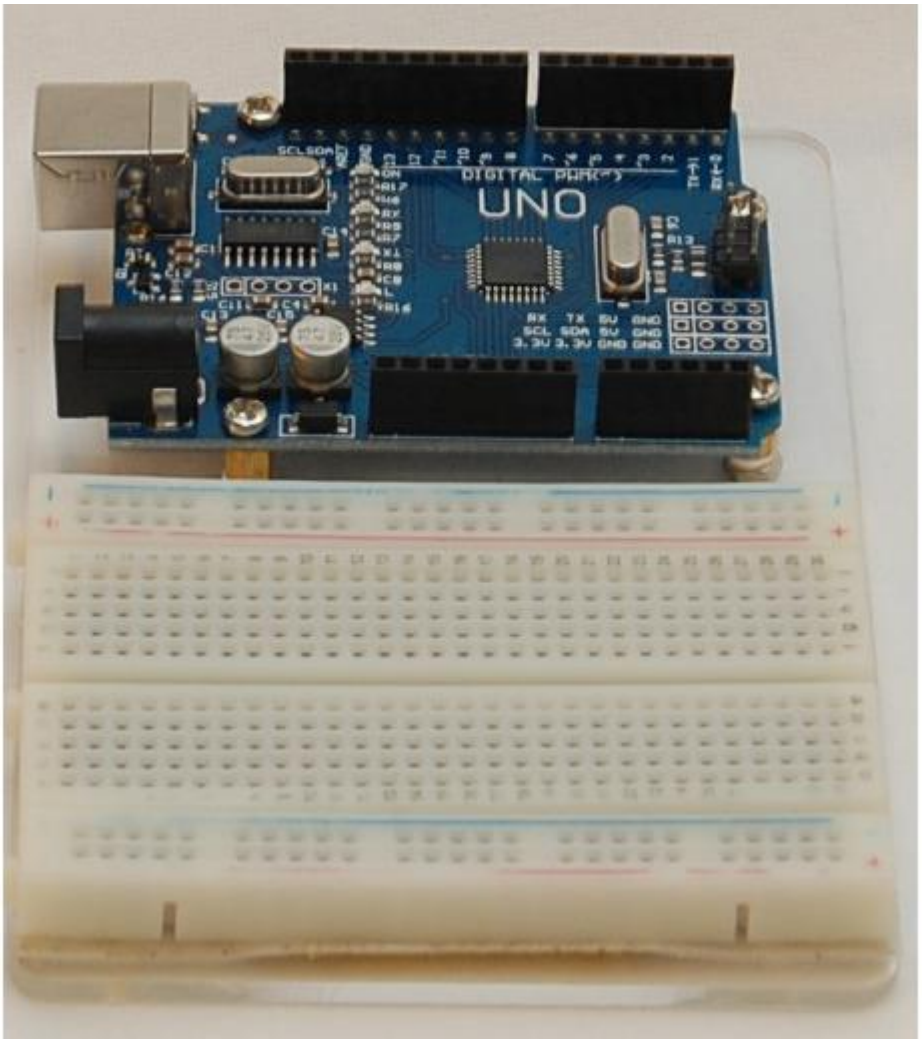
Если батарейка новая, то напряжение будет около 1.5 В. Разряженные батарейки имеют напряжение около 1 В или меньше.


A830L
1.52
HOLD
E B C E
NPN
PNP


## Делитель непряжения

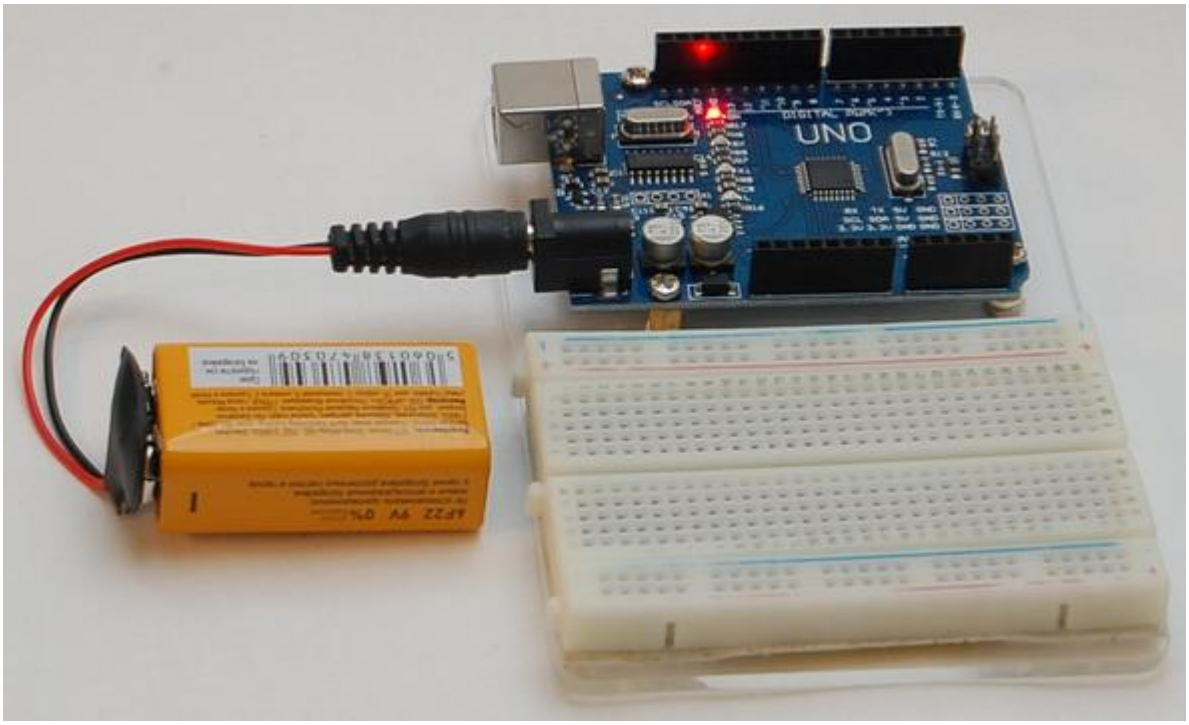
### Знакомство с питанием Arduino

Макетная плата или «макетка», как ее иногда называют, предназначена для установки электронных компонентов. Для удобства в работе макетку лучше всего приклеить на подложку рядом с платой Arduino. Плату Arduino прикрутите винтиками или приклейте на толстый слой двустороннего скотча.

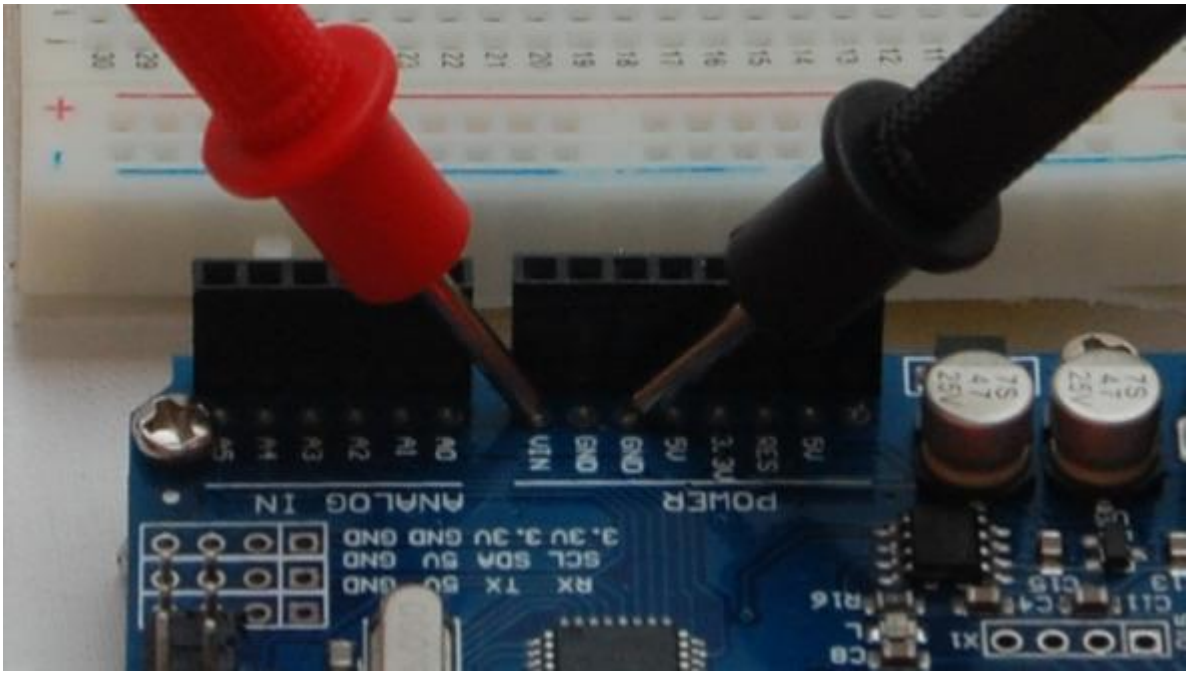



Все пятерки отверстий на макетной плате, пронумерованные как (a, b, c, d, e) и (f, g, h, i, j), соединены вместе. Всего есть 60 пятерок отверстий для установки электронных компонентов. Также есть две пары шин питания, обозначенных синей и красной полосками. К синим отверстиям следует подключать общий провод (GND), а к красным «+» питания.

Давайте исследуем, как изменяется напряжение Кроны, попадая на плату Arduino. Подключите Крону к разъему платы Arduino.

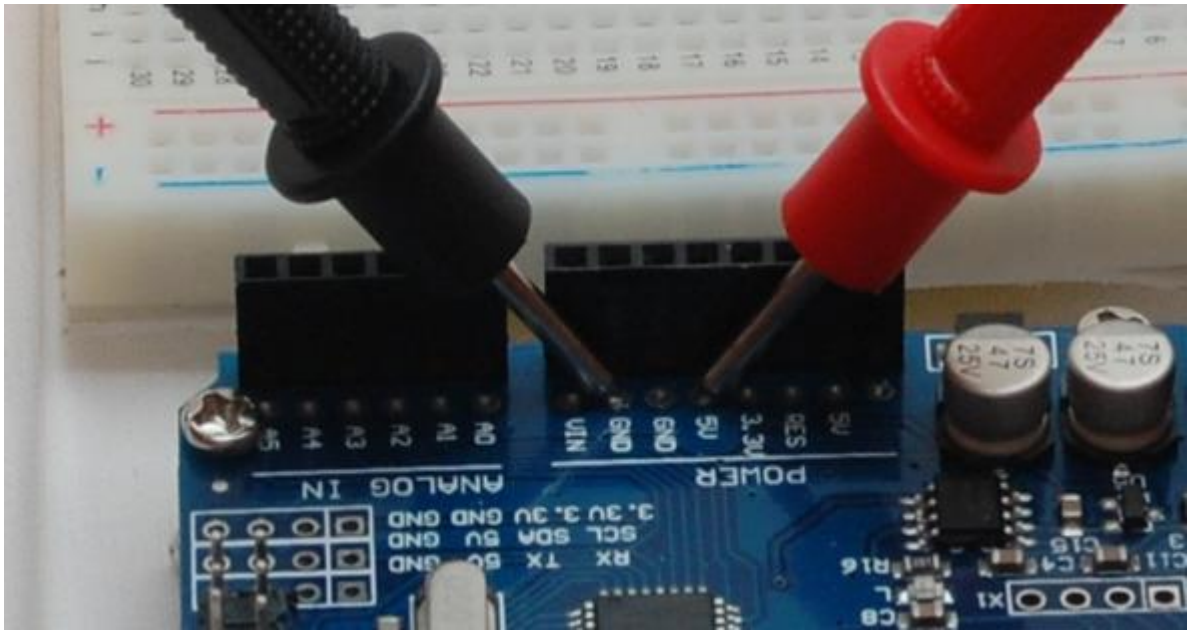



Загорится красный светодиод. Измерьте напряжение, поступающее на плату от Кроны. Черным щупом коснитесь пина GND, а красным пина Vin.



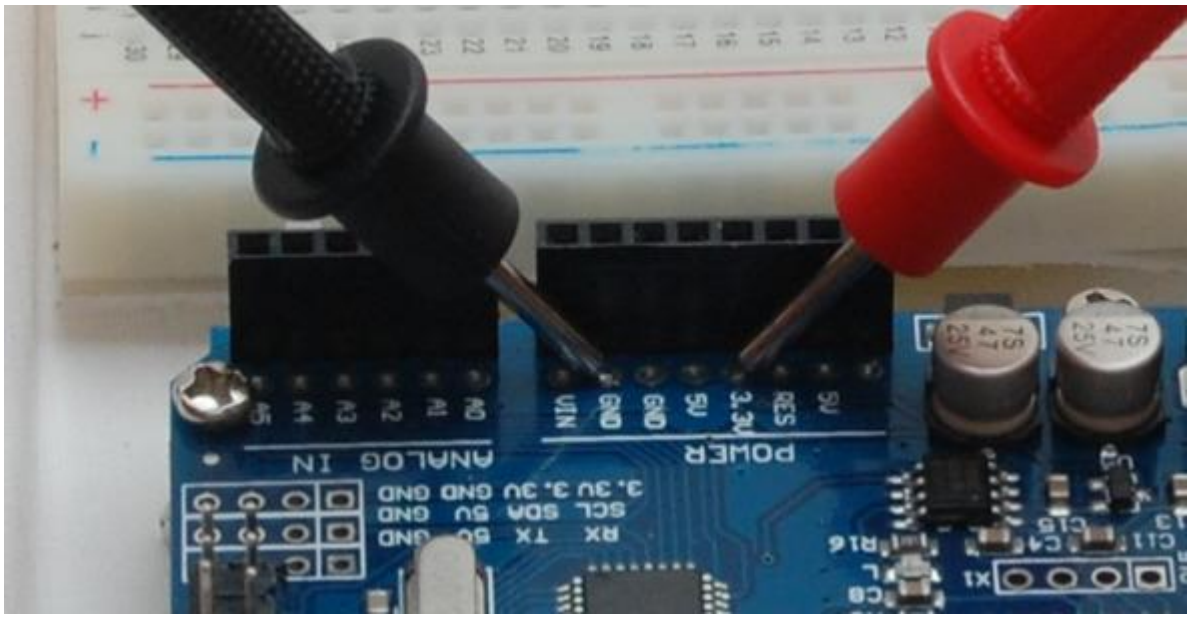

Напряжение должно быть около 9 В. (от 8.8В до 9.5В).


Измерьте напряжение между землей (GND) и пином 5V.




Напряжение должно быть около 5 В. (от 4.9В до 5.1В).


Измерьте напряжение между землей (GND) и пином 3.3V.




Напряжение должно быть около 3.3 В. (от 3.2В до 3.4В).



## Делитель напряжения из двух резисторов

*У меня в трюме пылится куча оружия с разряженными аккумуляторами. Использую его я не часто, поэтому постоянно забываю подзарядить. Надо будет попросить ребят заняться этим вопросом. Заодно научатся изменять уровень напряжения с помощью резисторов.*

*А вот и списочек оружия нашелся.*

*Дезинтегратор трехлинейный образца 2191 года. 10 шт. Напряжение аккумулятора 450 В.*

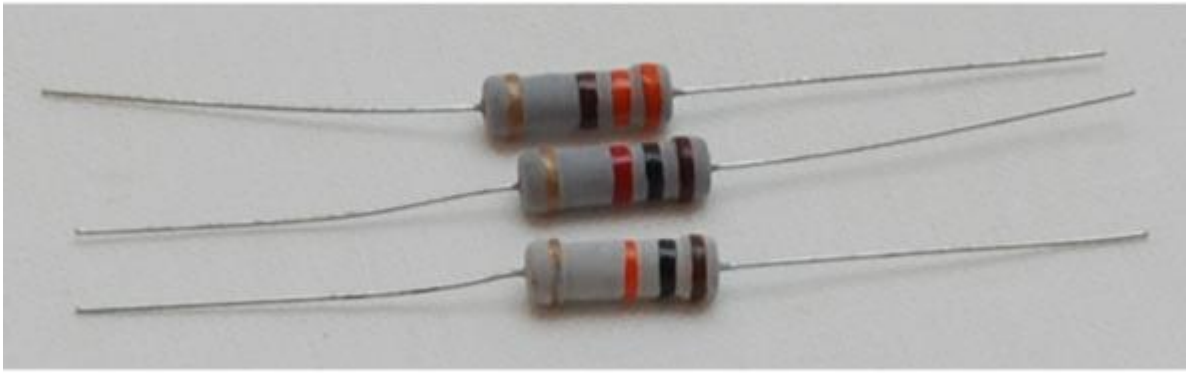
*Плазмомет Хайрема образца 2173 года. 3 шт. Напряжение аккумулятора 600В.*

*Бластер Наганта образца 2195 года. 25 шт. Напряжение аккумулятора 300В.*

Надеюсь, что вы помните, что бортовая сеть космического корабля имеет напряжение 900В. Если Вы посмотрите на напряжение питания трехлинейного дезинтегратора (450В), то заметите, что оно ровно в два раза меньше напряжения сети. Для того чтобы получить это напряжение, 900В нужно разделить пополам. Для этого используется делитель напряжения, состоящий из нескольких резисторов.

Резистор, это электронный компонент, обладающий сопротивлением (поэтому иногда резистор так и называют – сопротивление). Сопротивление измеряется в Омах и килоомах (в тысячах Ом).

В состав набора входят резисторы с сопротивлением 330 Ом, 1кОм и 10кОм.



Для измерения сопротивления переключите мультиметр в режим Ω на 20К..


Измерьте сопротивление каждого вида резисторов. Коснитесь щупами ножек резистора и посмотрите на экран мультиметра.

Сопротивление резистора номиналом 330 ом немного меньше и равно 320 Ом.


Сопротивление резистора номиналом 1 кОм (1000 Ом) тоже немного меньше и равно 970 Ом.


Сопротивление резистора номиналом 10 кОм (10000 Ом) равно 9810 Ом.


Теперь переключите мультиметр в режим режим Ω на 2К. Сопротивление резистора номиналом 10 кОм в этом режиме измерить не получится.


Сопротивление резистора номиналом 330 ом в этом режиме измерилось точнее, оно равно 326 Ом.


Сопротивление резистора номиналом 1 кОм (1000 Ом) тоже измерилось точнее, оно равно 979 Ом.

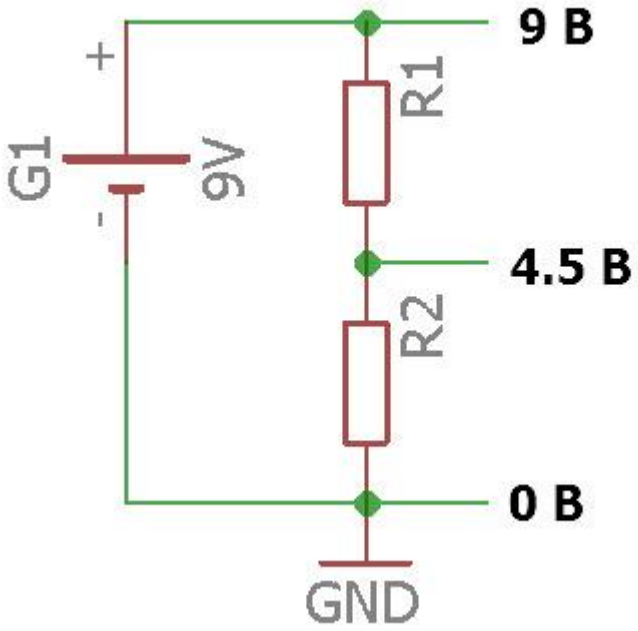

Сопротивление резистора номиналом 10 кОм (10000 Ом) в этом режиме измерить нельзя, так как диапазон измерения 2К меньше чем 10 К. На экране будет показана единичка, это означает, что измерить значение невозможно.


Задание.

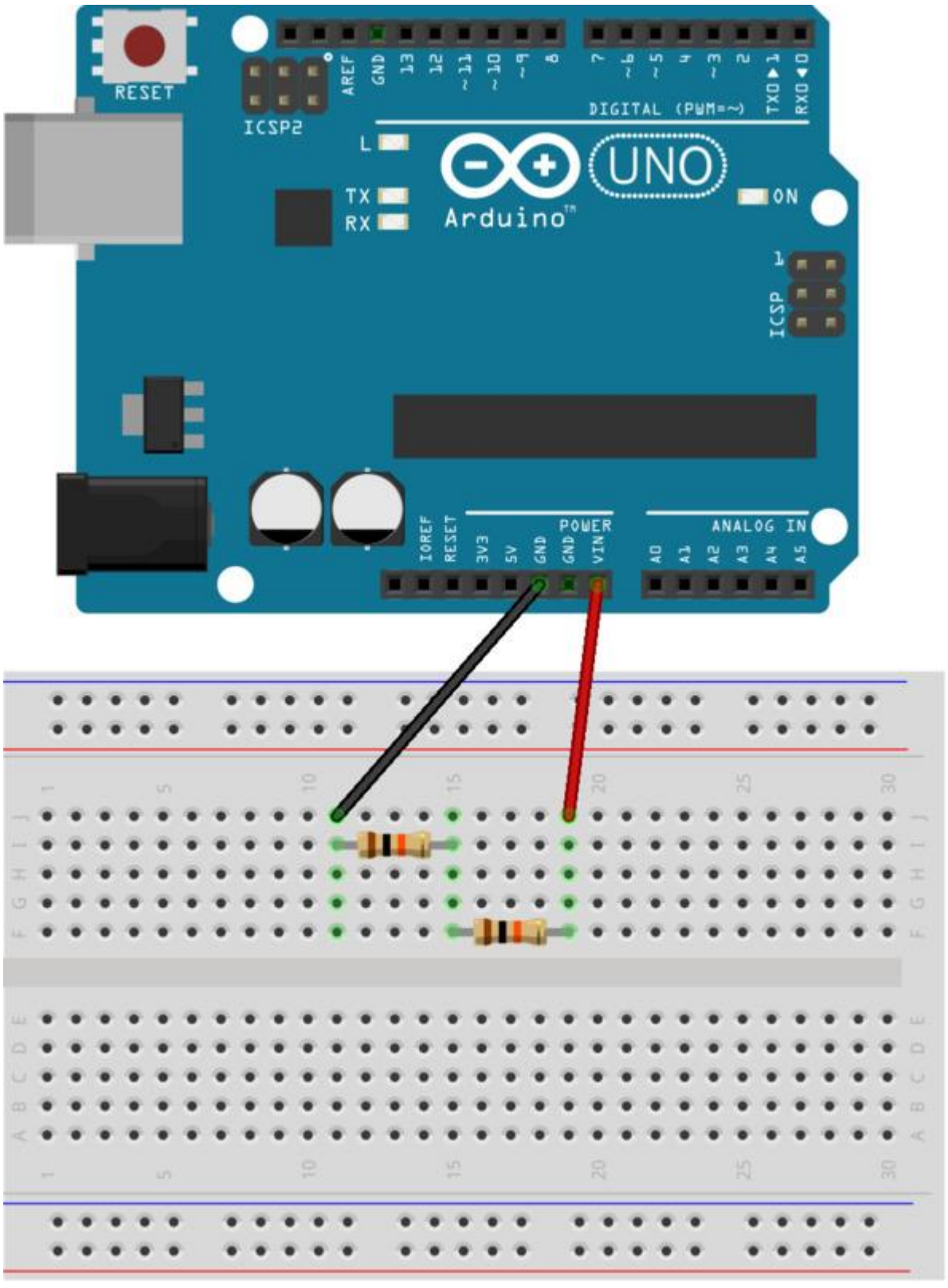
Измерьте сопротивление всех резисторов входящих в набор.

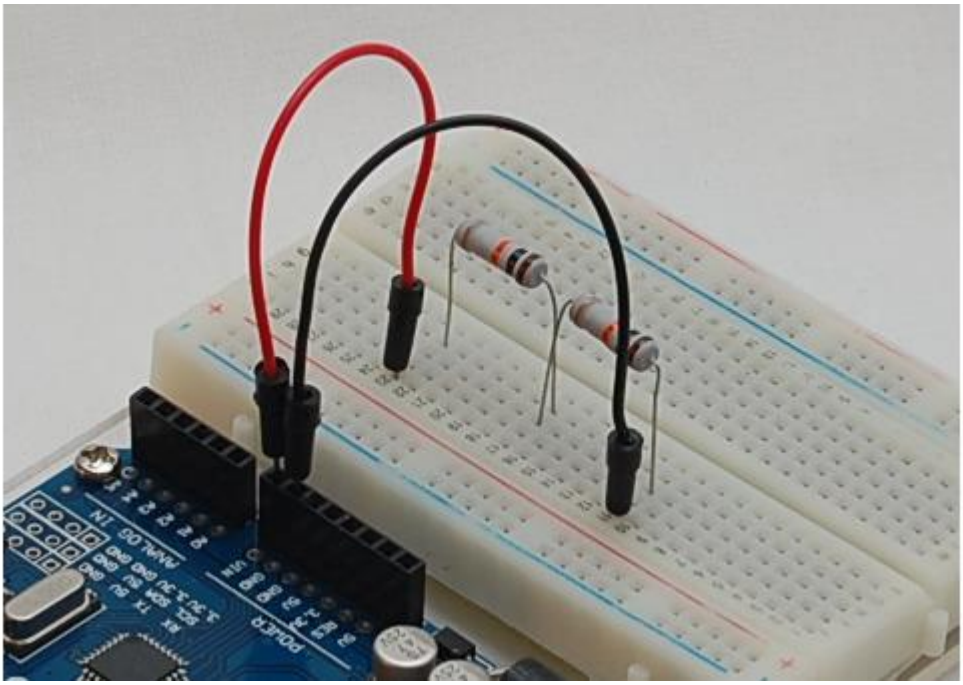

Напряжение 900В очень опасное, поэтому соберем макет делителя напряжения с использованием Кроны, напряжение которой ровно в 100 раз меньше бортового.

Для того чтобы разделить напряжение пополам нам потребуется собрать делитель напряжения из двух одинаковых резисторов. Вот так выглядит электрическая принципиальная схема делителя напряжения. Крона обозначается как G1, а резисторы как R1 и R2. Общий провод, подключенный к минусу Кроны, обозначен как GND. На рисунке показаны уровни напряжения.

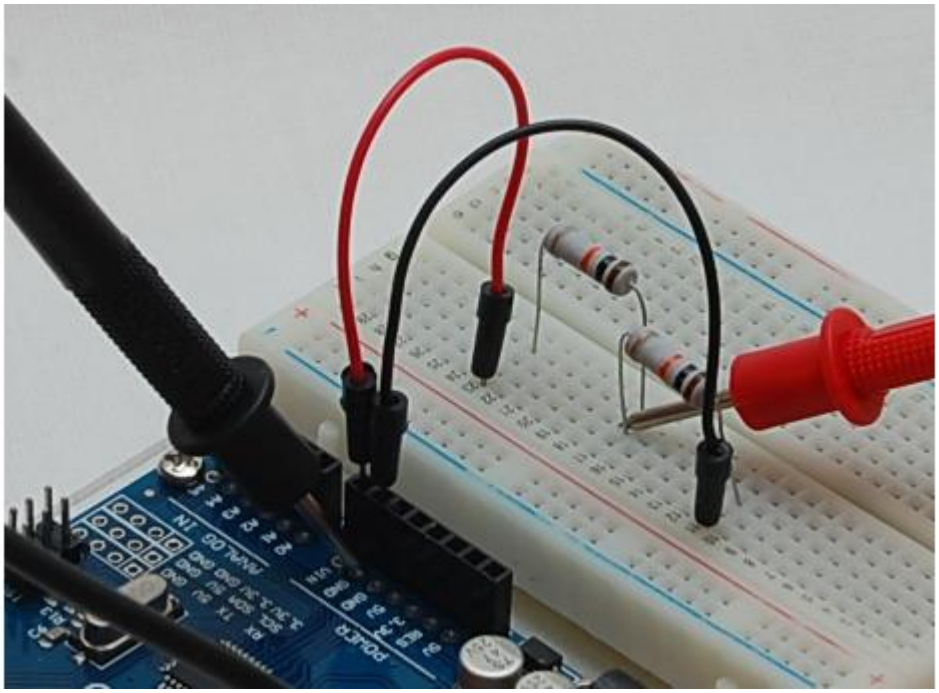



Возьмите два резистора по 10кОм и соберите делитель напряжения на макетной плате, подав питание от пина Vin.


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing



Переключите мультиметр в режим вольтметра и измерьте напряжение между землей (GND) и точкой соединения резисторов.

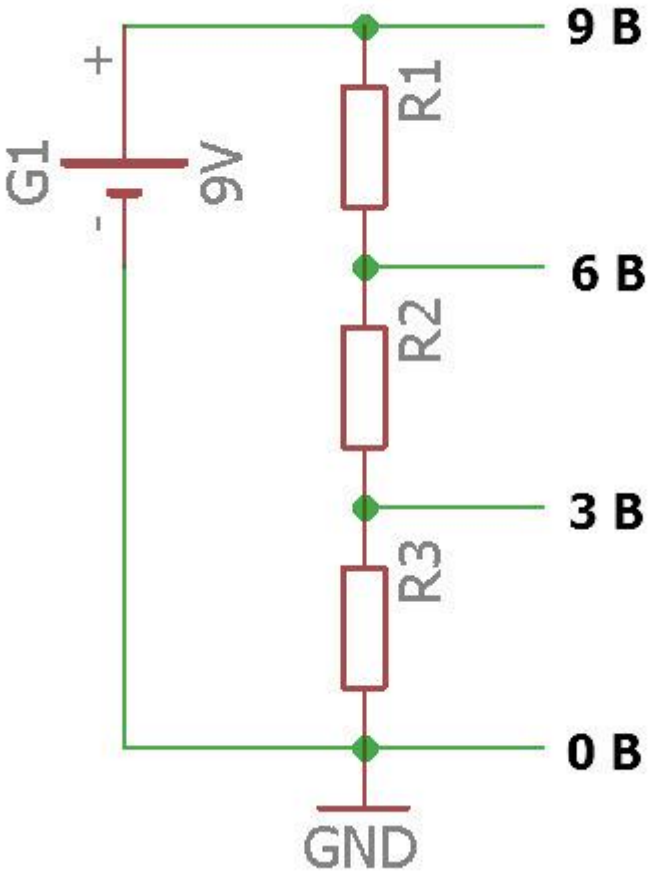


Напряжение будет около 4,3В, почти ровно половина от напряжения на Vin.

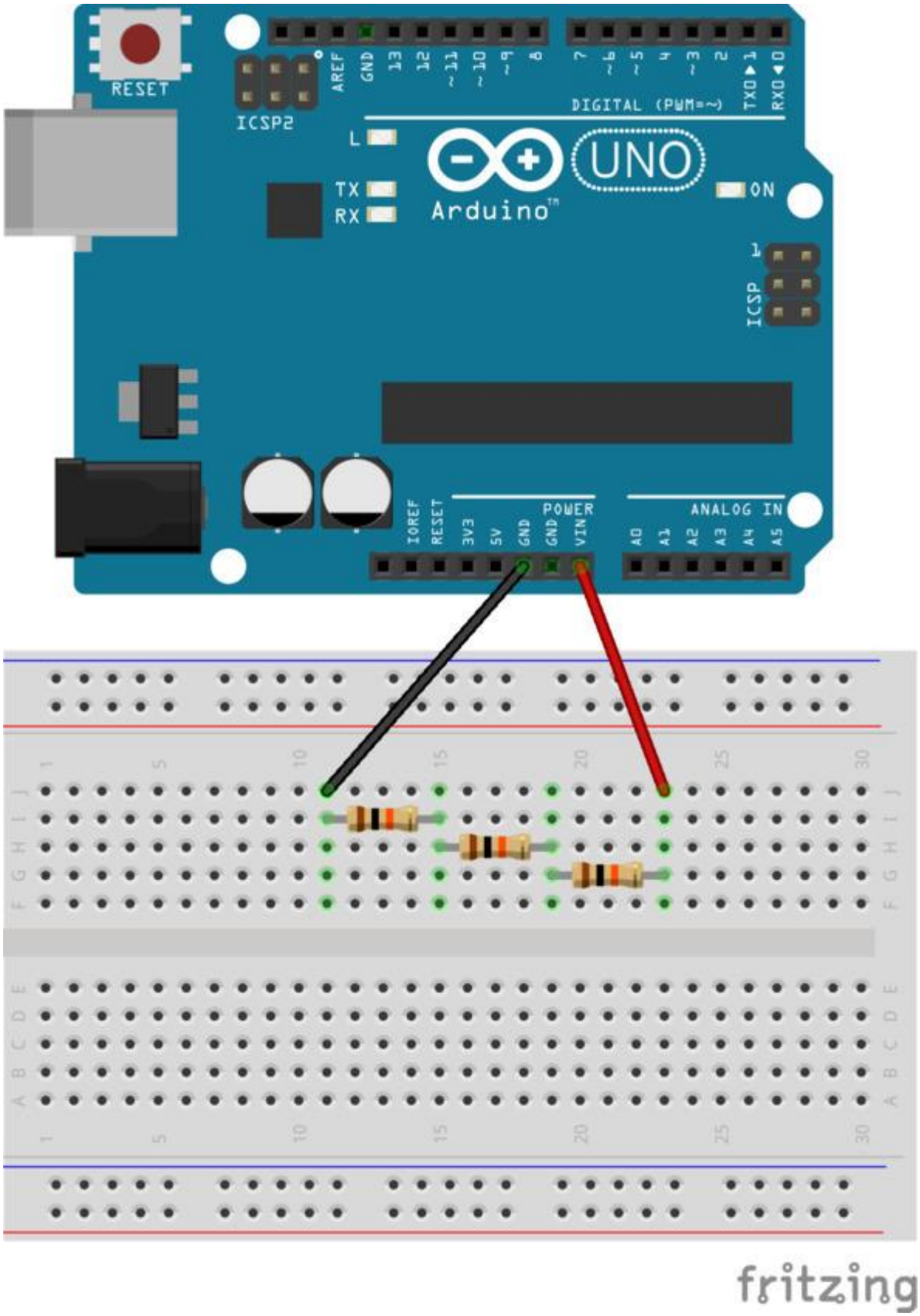

Напряжение разделится не ровно из-за различия сопротивлений резисторов. Если бы сопротивление резисторов было абсолютно одинаковым, то делитель разделил бы напряжение ровно пополам.

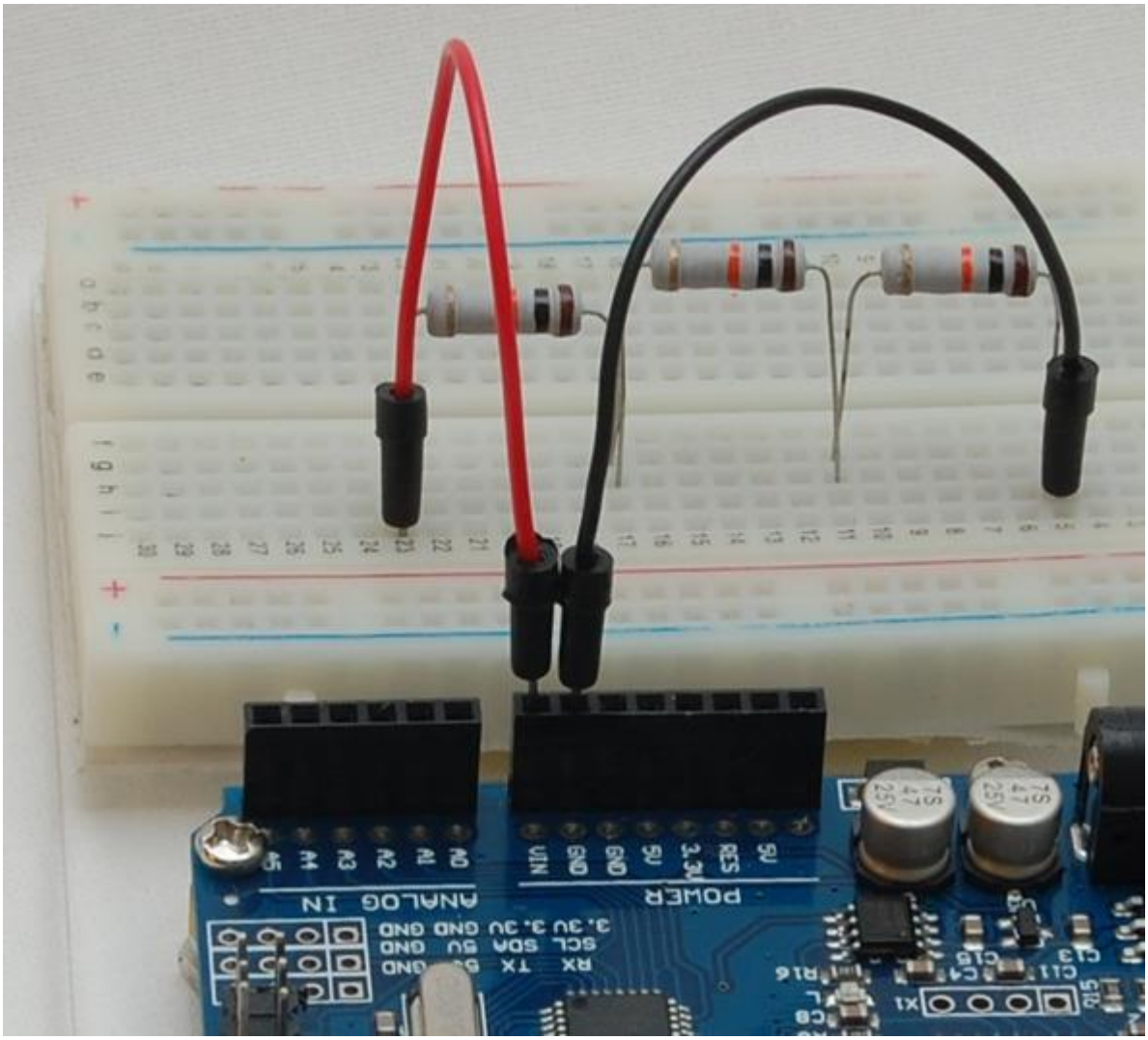
## Делитель напряжения из трех резисторов

Теперь вы знаете, как разделить напряжение пополам и собрать схему зарядки трехлинейного дезинтегратора. А как зарядить бластер? Нужно разделить бортовое напряжение 900В на три!

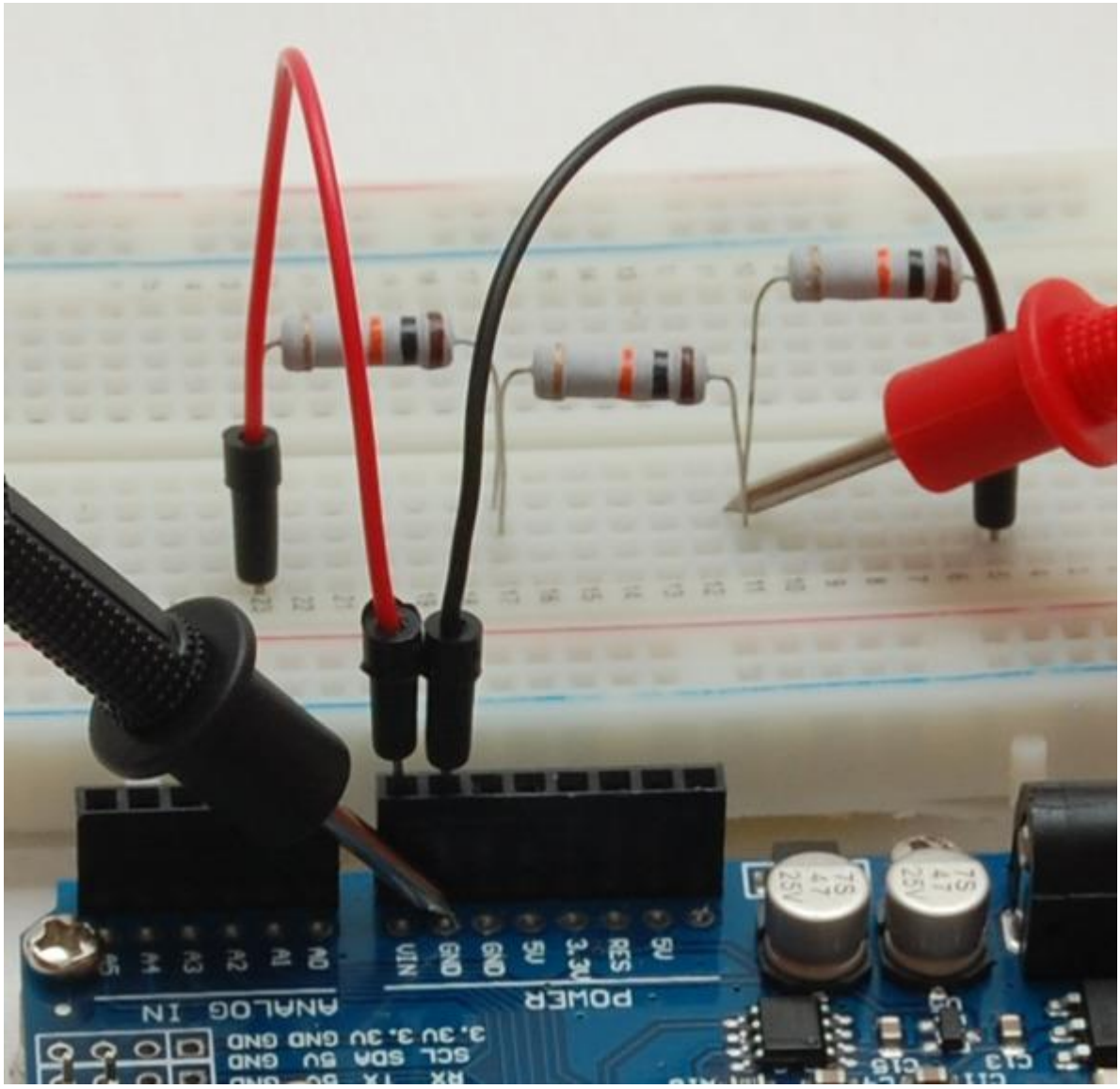


Соберите схему делителя напряжения из трех резисторов по 10кОм. Питание на макетную плату подайте от пина Vin.






Переключите мультиметр в режим вольтметра и измерьте напряжение между землей (GND) и точкой соединения первого и второго резисторов.




Напряжение будет около 2.8В, почти ровно треть от напряжения на Vin.


Надеюсь, вы догадаетесь, как от этой схемы зарядить плазмомет?

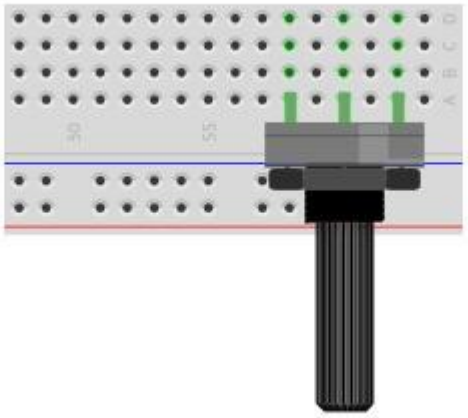
Правильно, надо подключиться к точке соединения второго и третьего резисторов. Напряжение в этой точке будет равно двум третям от напряжения Vin (около 5.6В).

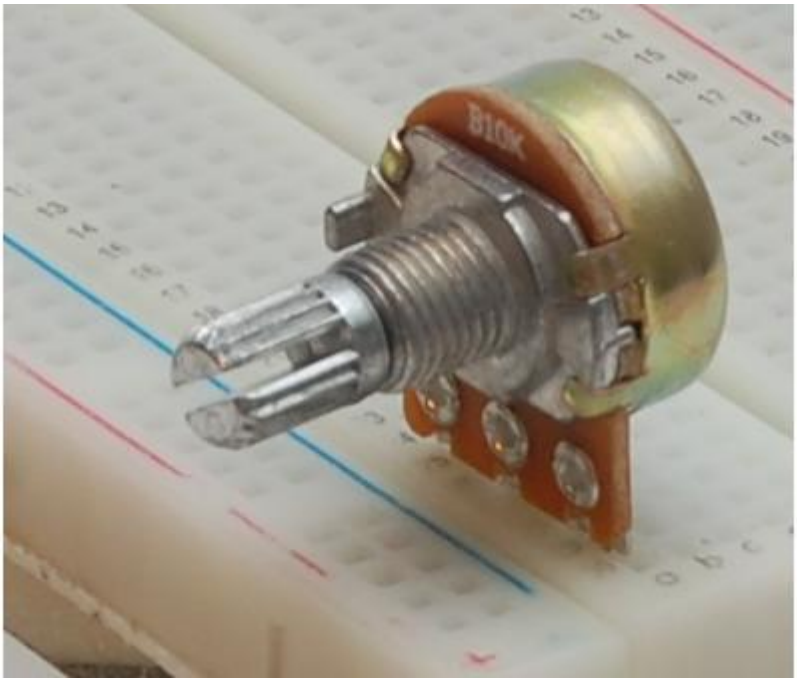

Задания.
1. Соберите делитель напряжения из четырех резисторов по 10кОм.
2. Рассчитайте напряжения в точках соединения резисторов.
3. Проверьте расчеты. Измерьте напряжения мультиметром.

# Ремонт боевого лазера

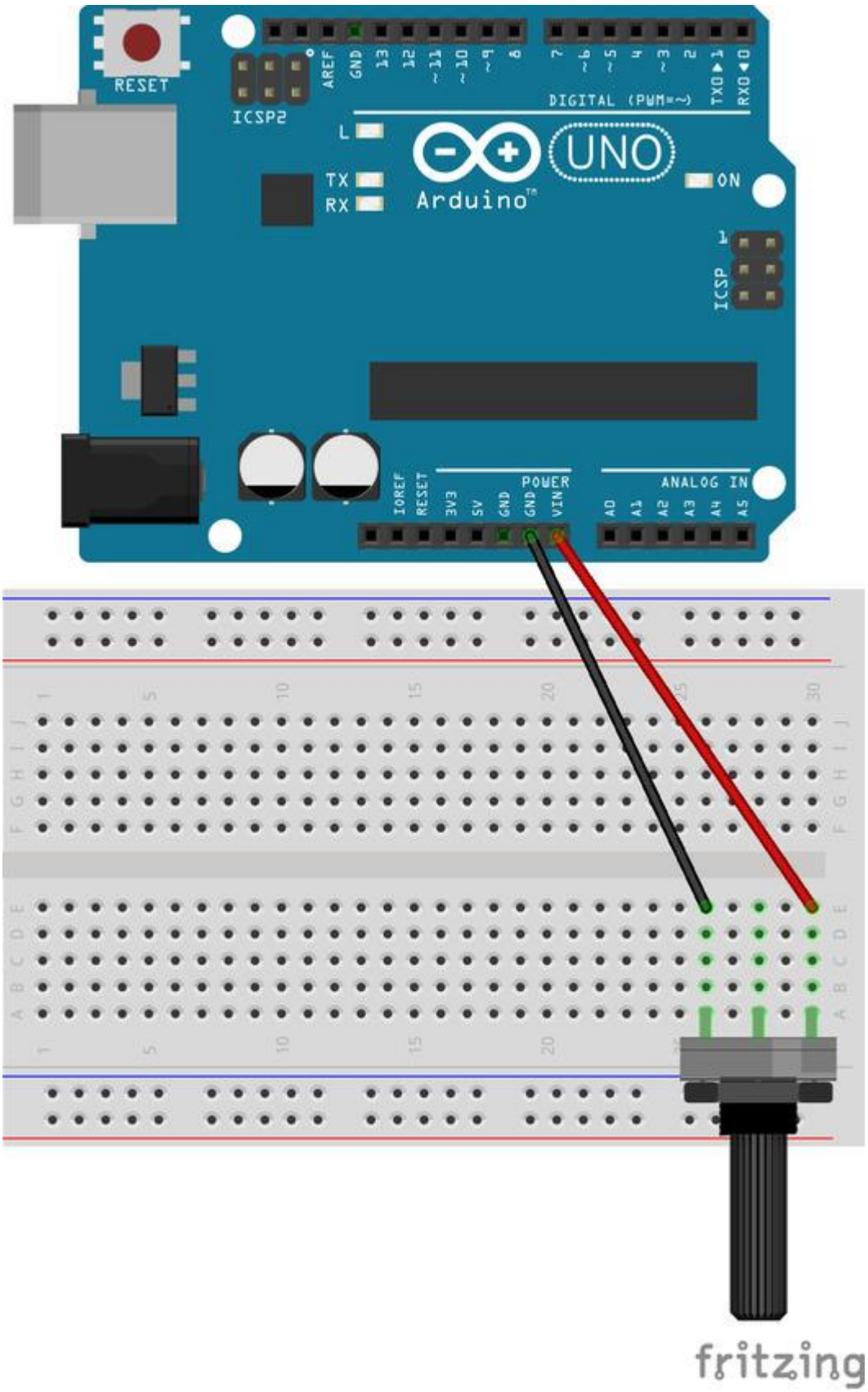
## Потенциометр

Сообщу вам по секрету, что резисторы бывают не только постоянные, но и переменные. Такие резисторы могут изменять свое сопротивление при вращении ручки. Их называют потенциометрами, но некоторым по душе старинное название – переменный резистор. Давайте познакомимся с ним поближе. Установите потенциометр на макетной плате как показано на рисунке.





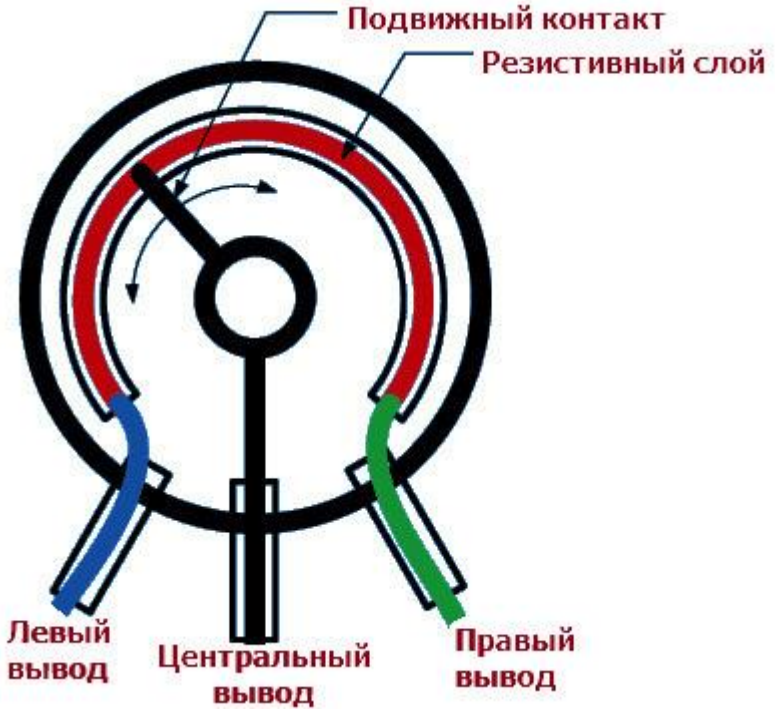

Подключите к крайним выводам потенциометра провода от пинов GND и Vin от платы Arduino.




Переключите мультиметр в режим измерения напряжения (переключатель на цифру 20), установите черный щуп на GND, а красный на центральном выводе потенциометра. Вращайте ручку потенциометра и посмотрите, как будет изменяться напряжение на центральном выводе.

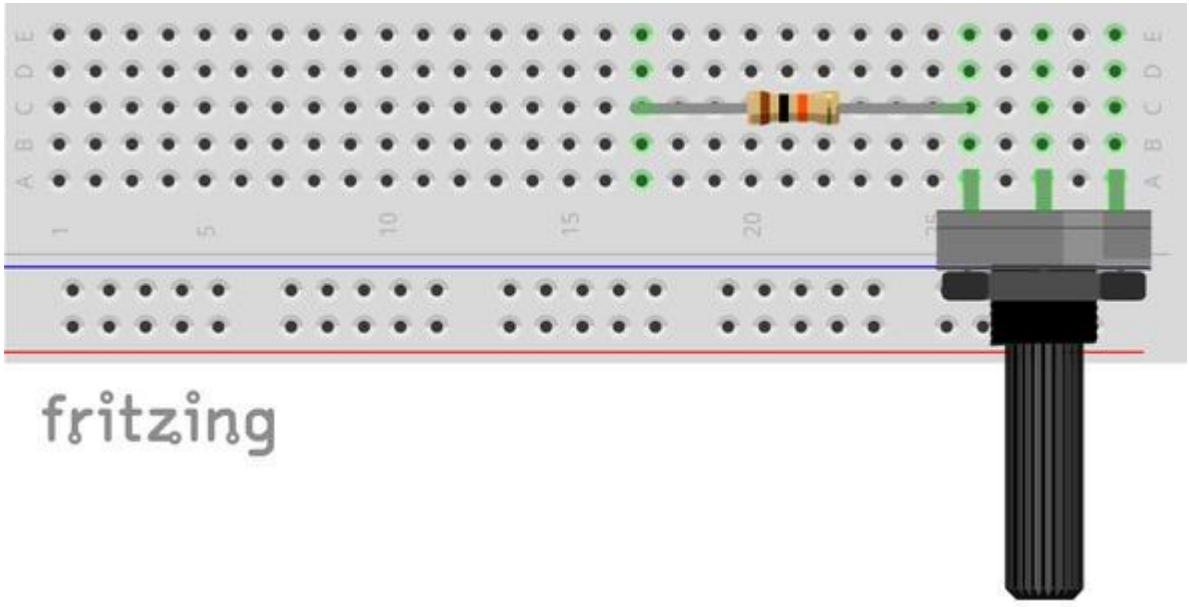
Потенциометр представляет собой дорожку из резистивного материала с большим сопротивлением, по которой перемещается центральный контакт. Если поставить ручку

потенциометра в центральное положение, то две его половинки будут обладать одинаковым сопротивлением, и получится простой делитель напряжения, на центральном выводе будет ровно половина от напряжения, поданного на крайние выводы потенциометра.




Задание.

Соберите следующую схему и измерьте мультиметром минимальное и максимальное сопротивление двух резисторов.

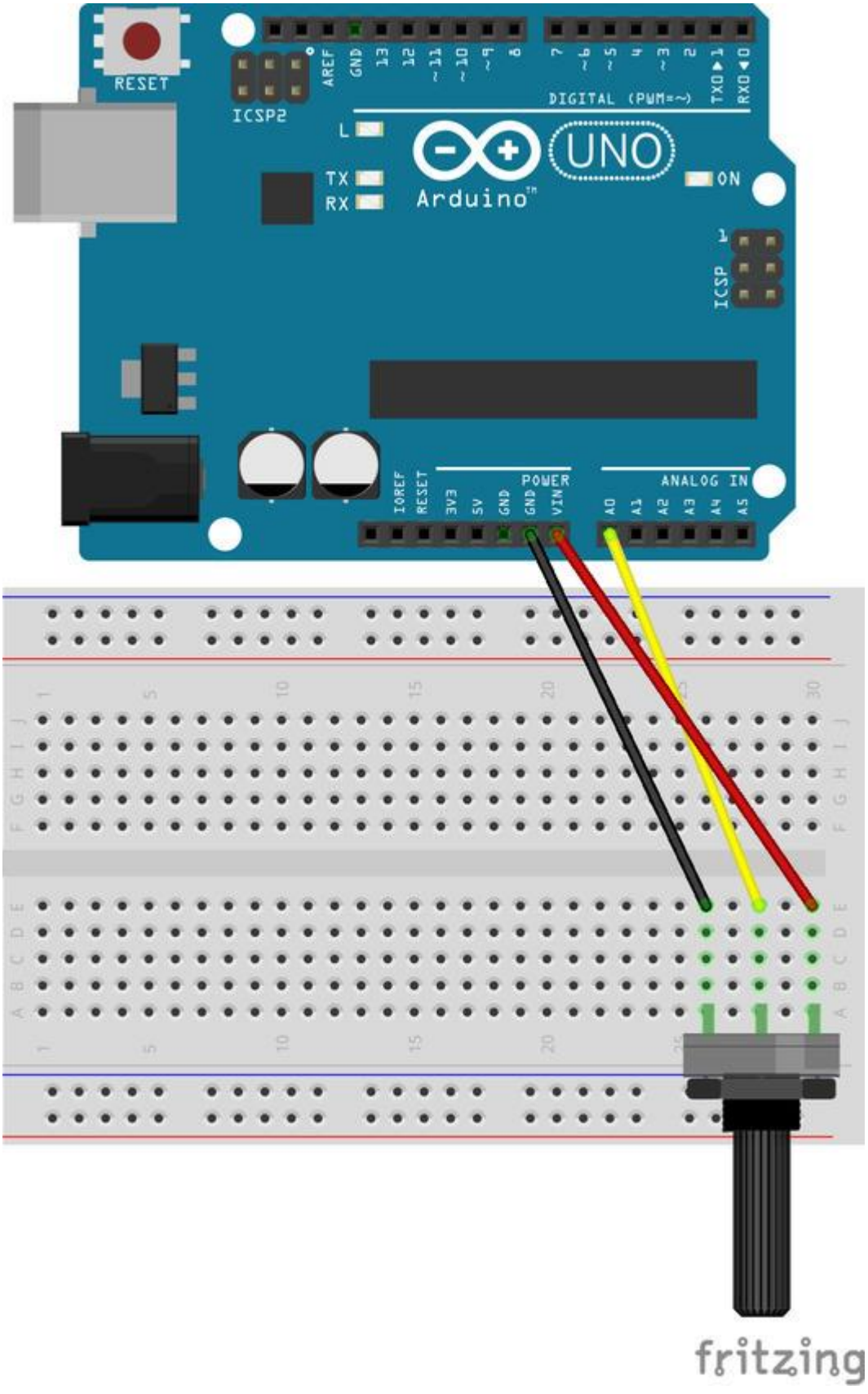



## Управление с помощью потенциометра

*Давненько я не стрелял из боевого корабельного лазера. Все-таки я мирный торговец, а не пират. И вот сегодня, когда я знакомил ребят с боевыми системами корабля, не смог показать им, как управлять основным боевым лазером. Проводка что ли повреждена… Придется совместить урок управления оружием с ремонтом лазера. Управляется он*

*с помощью переменного резистора, надеюсь, ребятки запомнили прошлый урок, вот и применят новые знания на практике.*

Подключите центральный вывод потенциометра к пину A0, правый к +5V, а левый к GND.

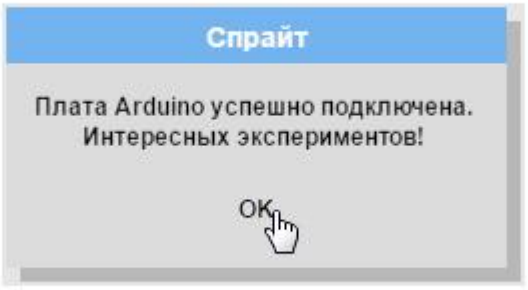



С помощью USB провода подключите плату Arduino к компьютеру.


Запустите Snap4Arduino и соедините его с платой Arduino переключившись в раздел Arduino и кликнув на кнопку **Подключится к Arduino**.

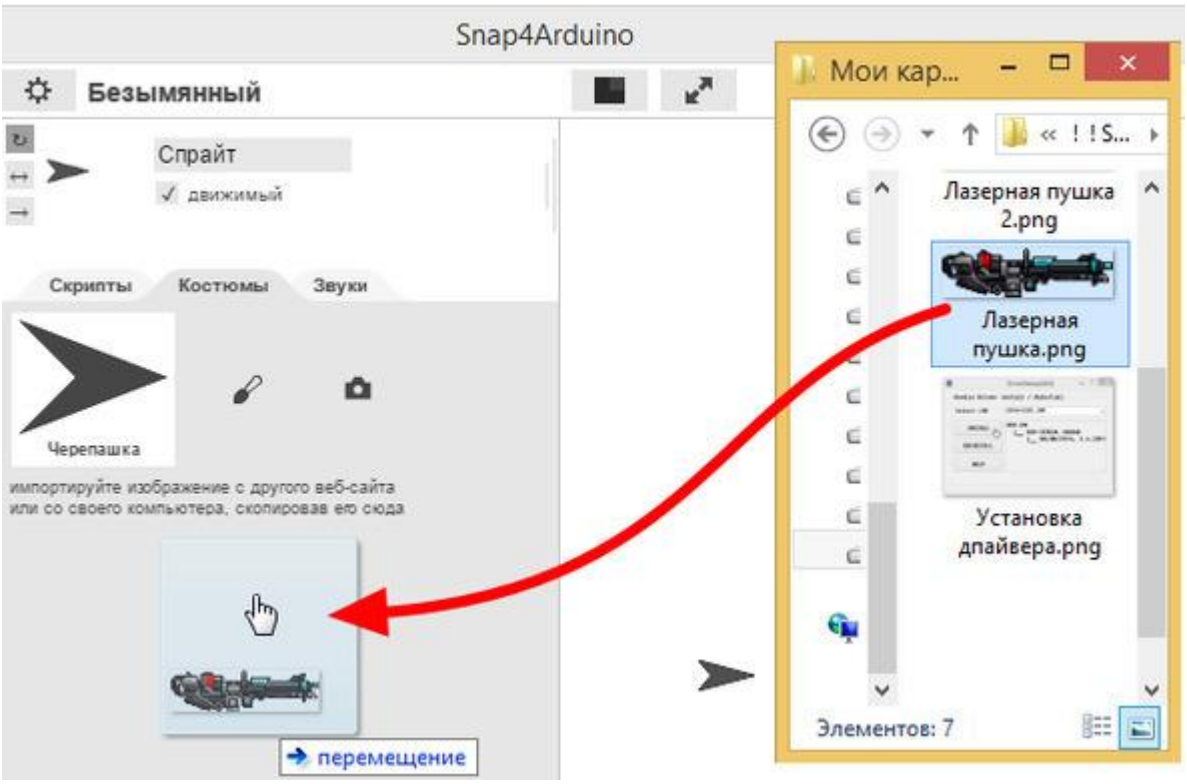

Выберите порт, к которому подключена плата Arduino.

При успешном подключении появится следующее сообщение.



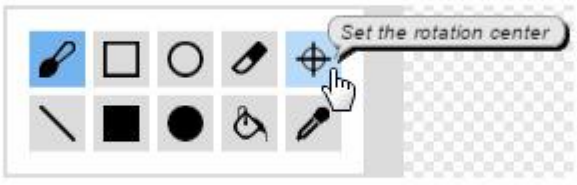

Перейдите на вкладку **Костюмы**.


Импортируйте изображение корабельного лазера, перетащив его из папки с изображениями прямо в Snap4Arduino.




Отпустите кнопку мышки, у спрайта появится новый костюм с изображением боевого лазера. Теперь нужно изменить центр вращения лазера, для этого перейдите в графический редактор, нажав правой кнопкой на лазер и выбрав команду **редактировать**.

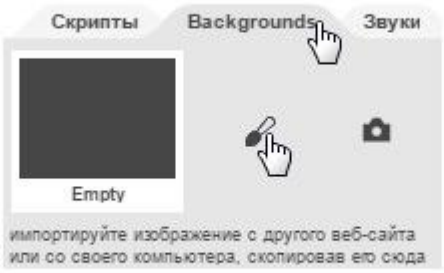

Выберите инструмент **Set the rotation center** (Установить центр вращения).



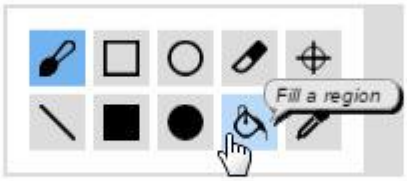

Установите центр вращения как показано на рисунке.

Теперь давайте перекрасим сцену в цвет космоса. Выберите сцену.

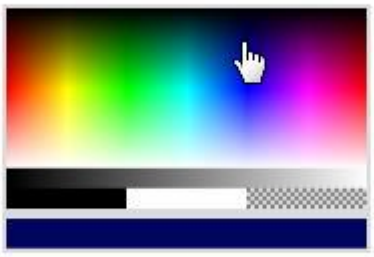

Перейдите на вкладку **Backgrounds** и кликните кисточку.




Откроется редактор изображений. Выберите инструмент заливки цветом **Fill a region** (Закрасить область).



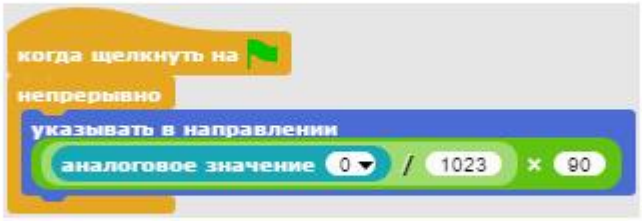

Выберите темно-синий цвет.



Кликните в области рисования – весь фон будет окрашен в темно-синий космический цвет. Перетащите Лазер в левый нижний угол сцены.

Перейдите на вкладку **Скрипты.**


Соберите вот такой скрипт для управления лазером.




Нажмите на зеленый флажок для запуска скрипта. Вращайте ручку потенциометра – лазер будет менять угол наклона от 0 до 90 градусов.

Если ручку потенциометра выкрутить до конца по часовой стрелке, то напряжение на центральном выводе потенциометра, подключенному к пину A0, будет равно 0В (с пина A0 будет считано 0). Разделив ноль на 1023, и умножив на 90, получим ноль градусов (лазер направлен вверх).

Если же ручку потенциометра выкрутить до конца против часовой стрелки, то напряжение на центральном выводе потенциометра будет равно 5В (с пина A0 будет считано 1023). Разделив 1023 на 1023, и умножив на 90, получим 90 градусов (лазер направлен вправо).

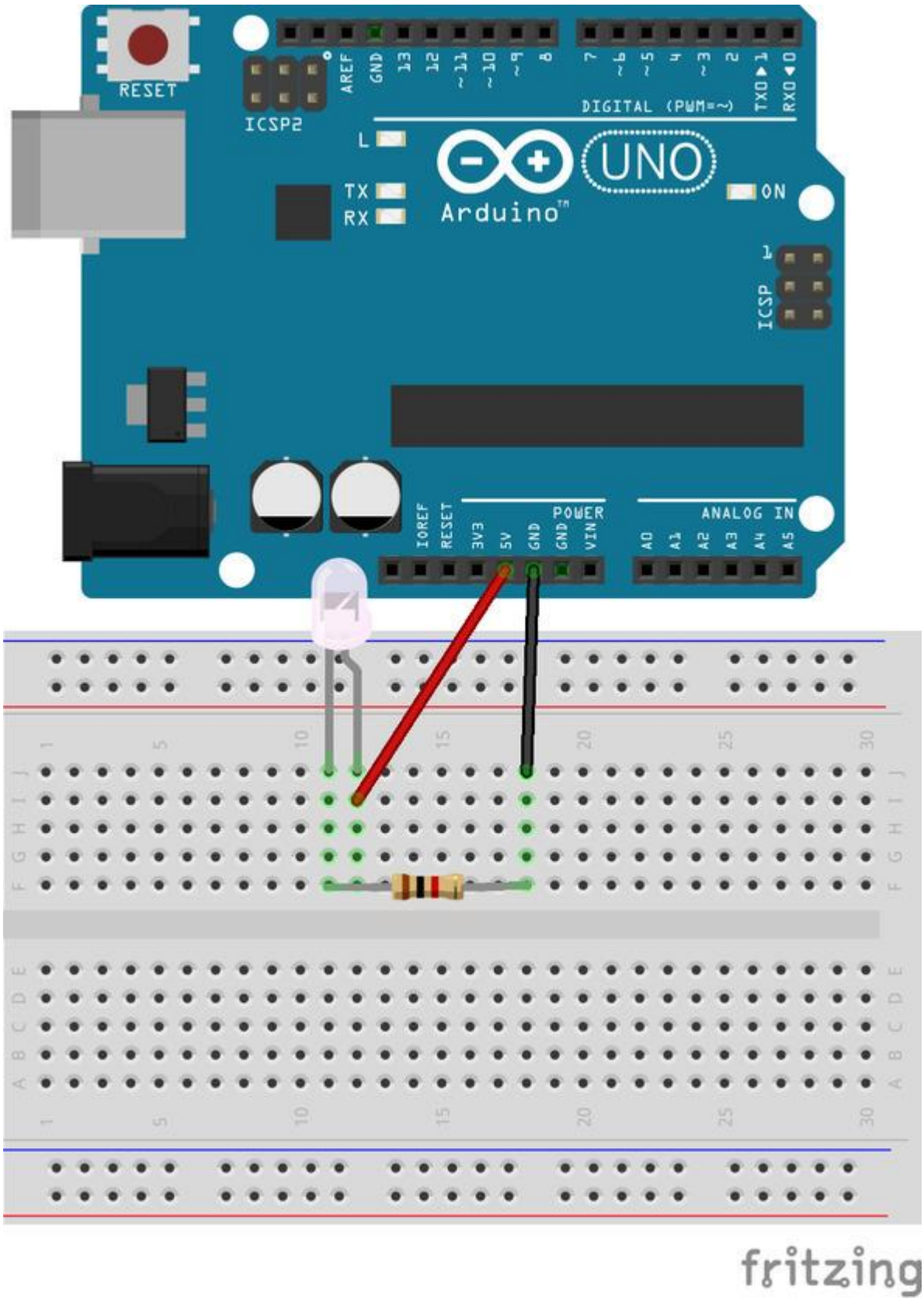
Сохраните проект.


# Ремонт прожектора

## Светодиод

*Вчера вечером, как обычно по пятницам, попытался нелегально пристыковаться к ретрансляционному спутнику, чтобы проверить электронную почту. Не только свою, а всю, которая проходила через него за неделю. Но выяснилось, что не работает основной прожектор, освещающий путь кораблю в темноте. Спутник находился в тени Титана, и путь нам освещали только далекие звезды и маленький серпик Гипериона. Я двигался по приборам, и, когда по моим расчетам должен быть уже в районе спутника, то включил прожектор. Светлее не стало, наверное, где-то в цепи прожектора плохой контакт. Пришлось вернуться на безопасную орбиту. Завтра поручу мальчишкам найти неисправность. Пользоваться мультиметром они уже умеют, поэтому должны справиться.*

Соберите на макетной плате следующую схему с белым светодиодом и резистором номиналом 1 кОм.



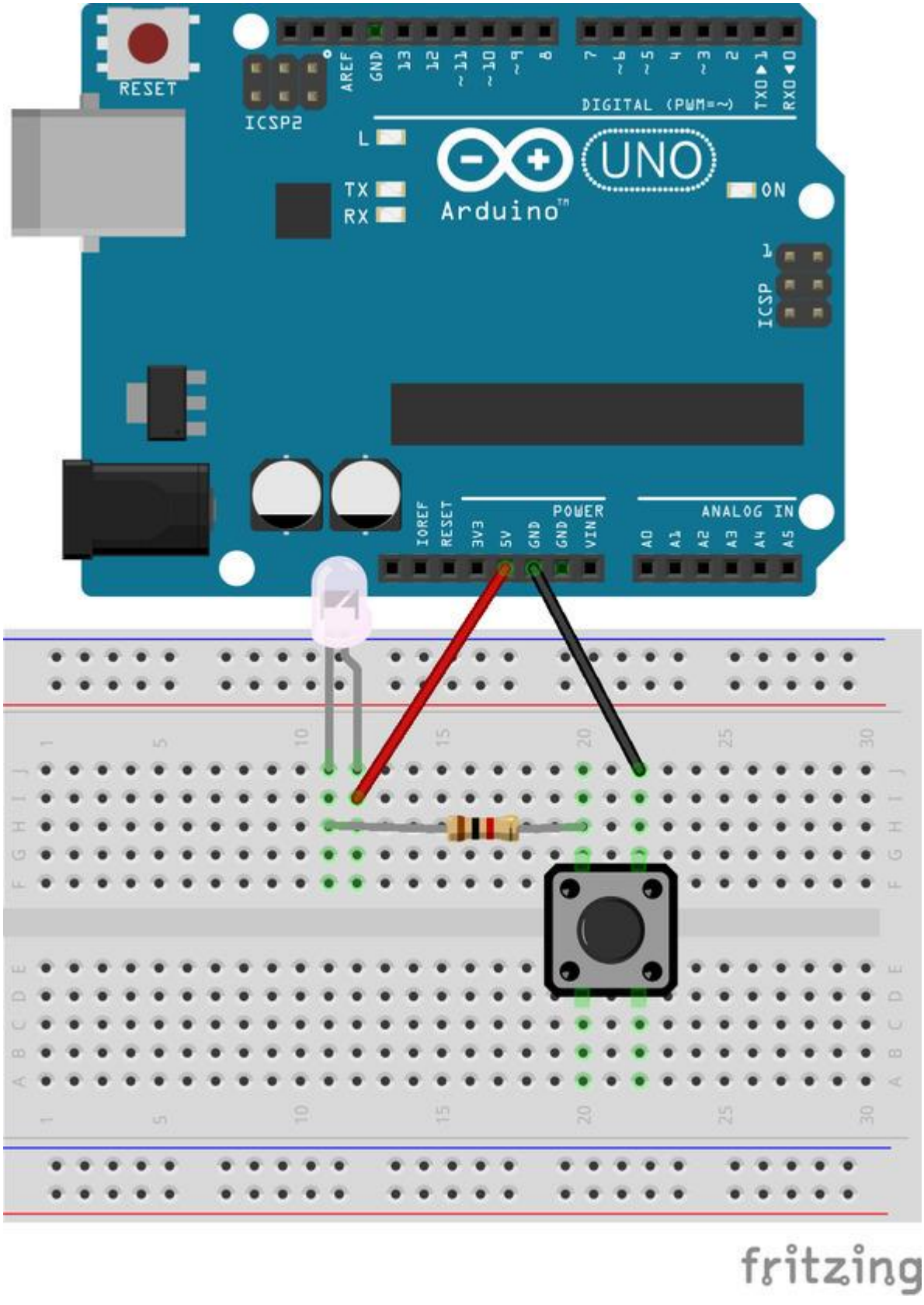

Питание подайте от пина 5V.

Используемый светодиод является сверхъярким, поэтому защитите глаза от его света. Оберните его кусочком синей изоленты, или немного изогните его, направив поток света от глаз.

## Светодиод и кнопка

*Отличная работа! Но прожектор же не должен гореть постоянно! С ярким прожектором на носу мы станем отличной мишенью для пиратов всех мастей. Надо переделать схему включения прожектора, чтобы он включался при нажатии на кнопку.*

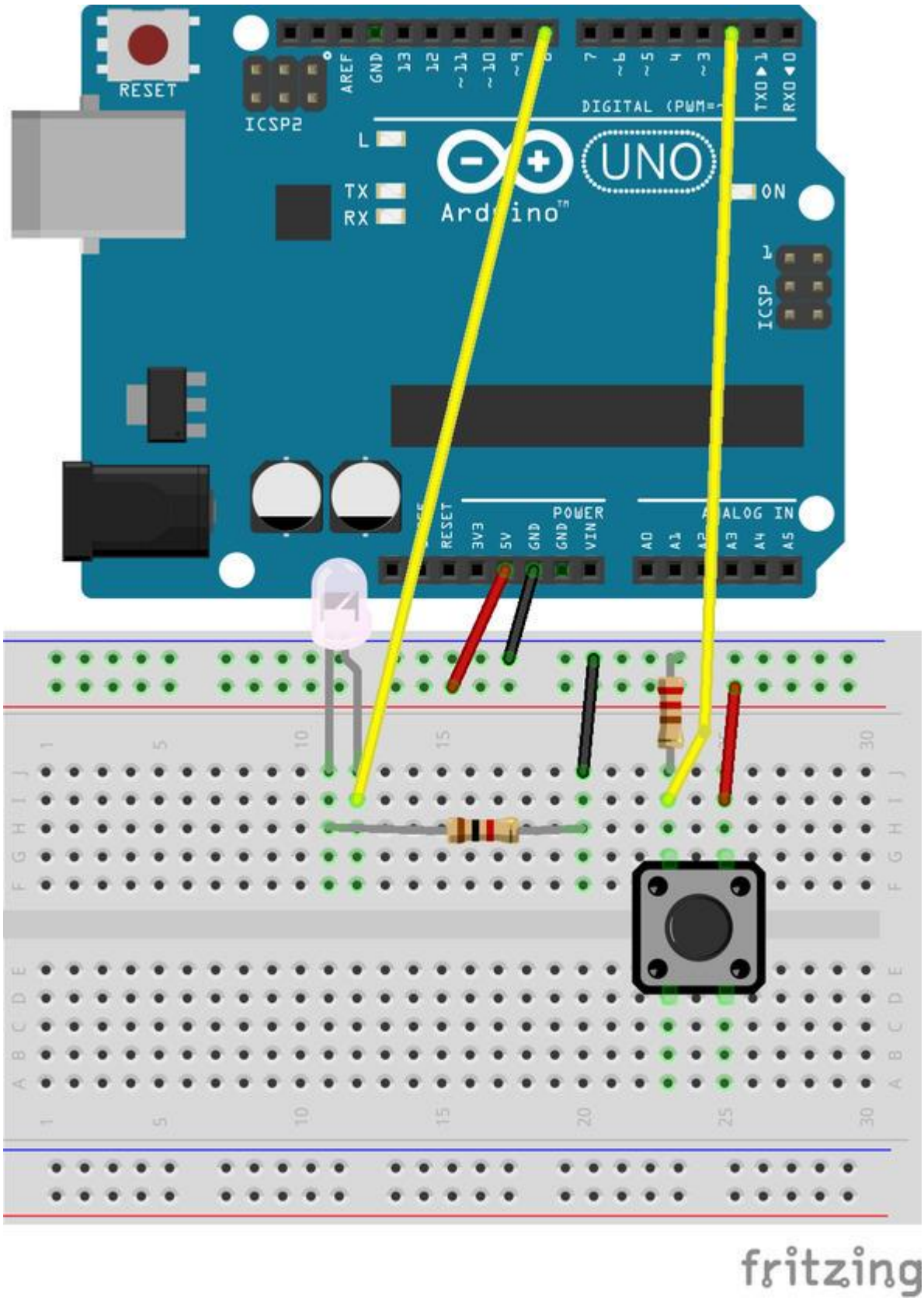
Соберите на макетной плате следующую схему с белым светодиодом, кнопкой и резистором номиналом 1 кОм. Питание подайте от пина 5V.



# Управление светодиодом

*Приблизившись к ретрансляционному спутнику, я поручил мистеру Васе нажать на кнопку включения прожектора, а сам вышел в открытый космос с набором моих любимых магнитных отверток. Они мне очень нравились тем, что прилипали ко всему, а не только к железу. Я уже почти открутил лючок, за которым была установлена моя шпионская флешка, как вдруг свет погас. Что за шутки! Заорал я по рации. Включите свет! У меня палец устал, сказал мистер Вася, пусть теперь Валек держит. Вы что там, с ума сошли, чтоли?! Здесь же темно как в черной дыре. Срочно включите свет! В наушниках я слышал какую-то возню. Тебе сказали, ты и держи, слышался недовольный голос Вали. Свет несколько раз моргнул, и больше не выключался. Получат они у меня! По 3000 приседаний! Не меньше!*

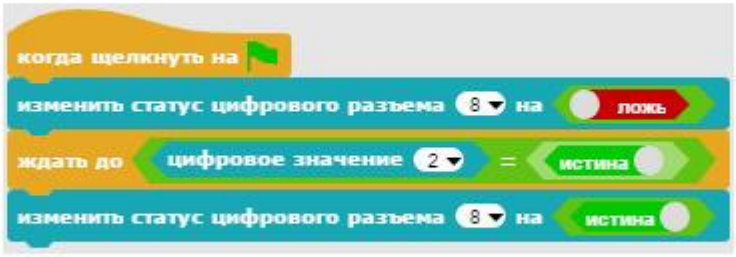
Так как вы уже поняли, что постоянно держать палец на кнопке включения прожектора не очень просто, то давайте немного изменим схему, чтобы при нажатии на кнопку включения светодиод включался и постоянно светился.




На этой схеме питание на светодиод подается от вывода 8. Если напряжение на этом выводе равно 0В, то светодиод светиться не будет, а если 5В, то будет.

Кнопка подключена к выводу 2. Через резистор этот вывод постоянно соединен с землей, поэтому пока кнопка не нажата, то на выводе 2 низкий уровень напряжения 0В. Если же кнопку нажать, то на вывод 2 будет подан высокий уровень напряжения 5В.

Запустите среду Snap4Arduino и установите соединение с платой Arduino.
Соберите следующий скрипт.




При нажатии на зеленый флажок светодиод светиться не будет, так как статус цифрового вывода 8 будет установлен в значение **ложь**, и на 8 выводе будет низкий уровень напряжения 0В. Скрипт будет ждать момента, когда будет нажата кнопка, и на цифровом выводе 2 появится высокий уровень напряжения 5В. При этом он изменит статус цифрового вывода 8 на **истина,** и на нем появится высокий уровень напряжения 5В. По желтому проводу это напряжение будет подано на светодиод.

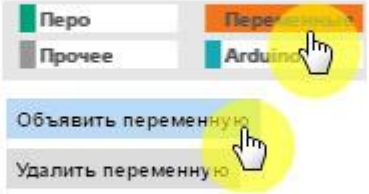
## Включение и выключение с помощью одной кнопки

*Уже присели по 2500 раз, маленькие бестолочи. Надо будет завтра рассказать им о всемогущей синей изоленте, которой они могли приклеить кнопку и не париться. Ну ничего, в космосе просто необходимо поддерживать хорошую физическую форму. Лично я уже присел сегодня 2000 раз, и меня это совершенно не тяготит, здесь ведь невесомость.*

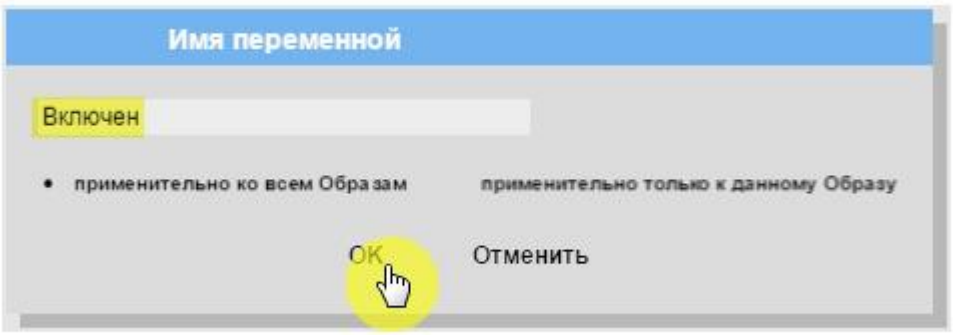
Схема работает, но только в одну сторону. Она позволяет включить светодиод, а как же его выключить? Давайте немного изменим программу, чтобы при первом нажатии на кнопку светодиод включался, а при следующем нажатии выключался.

Для этого нам понадобится переменная. Переменные – это такие оранжевые овалы, которые могут хранить значение, которое в них записано.

Для того чтобы создать переменную, запустите среду Snap4Arduino, выберите оранжевые блоки **Переменные** и нажмите на кнопку **Объявить переменную.**



Введите имя переменной ***Включен*** и нажмите **ОК**.



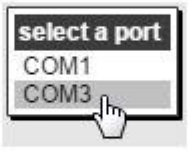

В палитре блоков появится оранжевая переменная ***Включен***, она будет хранить значение **Да,** если светодиод светится, и значение **Нет**, если он погас.


Если кликнуть по переменной, то рядом отобразится ее значение.

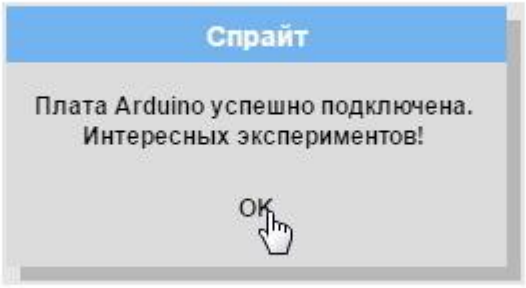

Продолжим создание скриптов.

С помощью USB провода подключите плату Arduino к компьютеру. Установите соединение с платой Arduino, переключившись в раздел Arduino и кликнув на кнопку **Подключится к Arduino**.


Выберите порт, к которому подключена плата Arduino (обычно это не COM1, а другой порт).



Если подключение прошло успешно, то вы увидите следующее сообщение.




Соберите следующий скрипт.


При нажатии на зеленый флажок светодиод светиться не будет, так как статус цифрового вывода 8 будет установлен в значение **ложь**, и на 8 выводе будет низкий уровень напряжения 0В. Скрипт будет ждать момента, когда будет нажата кнопка, и на цифровом выводе 2 появится высокий уровень напряжения 5В. В этом случае он проверит значение переменной ***Включен***.

Если значение переменной равно «Нет», то будет изменен статус цифрового вывода 8 на **истина**, и на нем появится высокий уровень напряжения 5В. Одновременно значение переменной ***Включен*** будет изменено на «Да». Светодиод будет светиться.

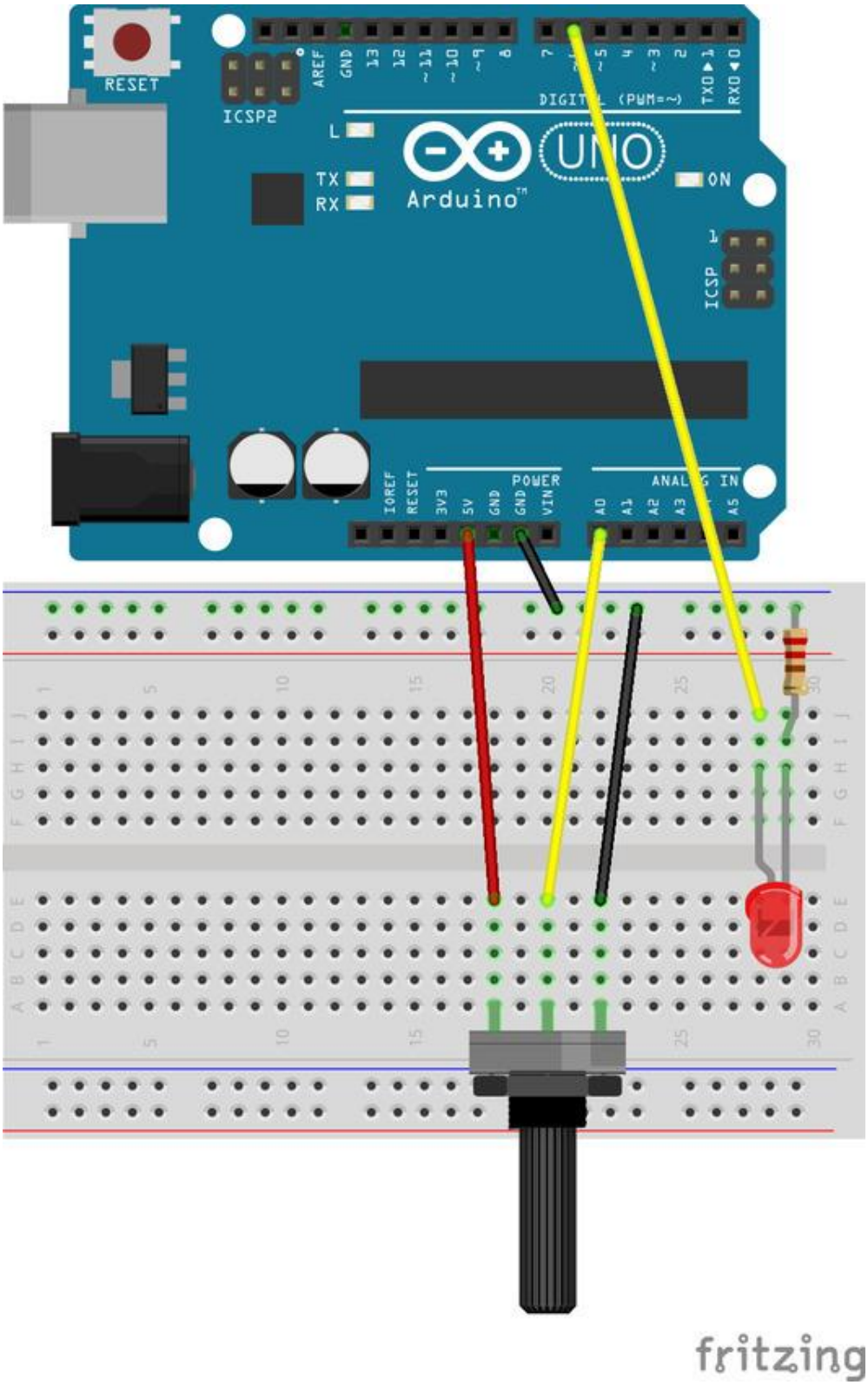
Если значение переменной равно «Да», то будет изменен статус цифрового вывода 8 на **ложь,** и на нем появится низкий уровень напряжения 0В. Одновременно значение переменной ***Включен*** будет изменено на «Нет». Светодиод погаснет.

# Ремонт оружия

## Ремонт бластера

*Пока заряжали аккумуляторы бластеров, нашли один с неисправной регулировкой мощности. Маленький регулировочный потенциометр, наверное, вышел из строя, поэтому бластер заклинило на минимальной мощности луча. Это, конечно позволяет беречь батарею, но в бою может сослужить плохую службу, не позволяя прожигать бронированные цели. Надо будет его отремонтировать.*

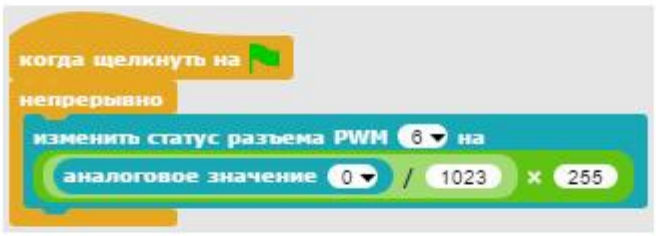
Соберите схему с потенциометром, светодиодом и резистором номиналом 330 Ом.



Средний вывод потенциометра соедините с пином A0, а длинную ножку светодиода с пином 6.

Запустите среду Snap4Arduino и установите соединение с платой Arduino.

Соберите следующий скрипт.




При запуске программы статус разъема 6 будет установлен в значение, пропорциональное считанному с аналогового пина A0. Статус разъема 6 может изменяться в диапазоне от 0 до 255, что соответствует изменению напряжения на от 0В до 5В.

Если ручку потенциометра выкрутить в одну сторону, чтобы напряжение на центральном выводе, подключенном к A0 стало равно нулю, то статус разъема 6 тоже будет равен нулю (0 разделить на 1023 и умножить на 255), и напряжение на нем будет равно 0В. Светодиод светиться не будет.

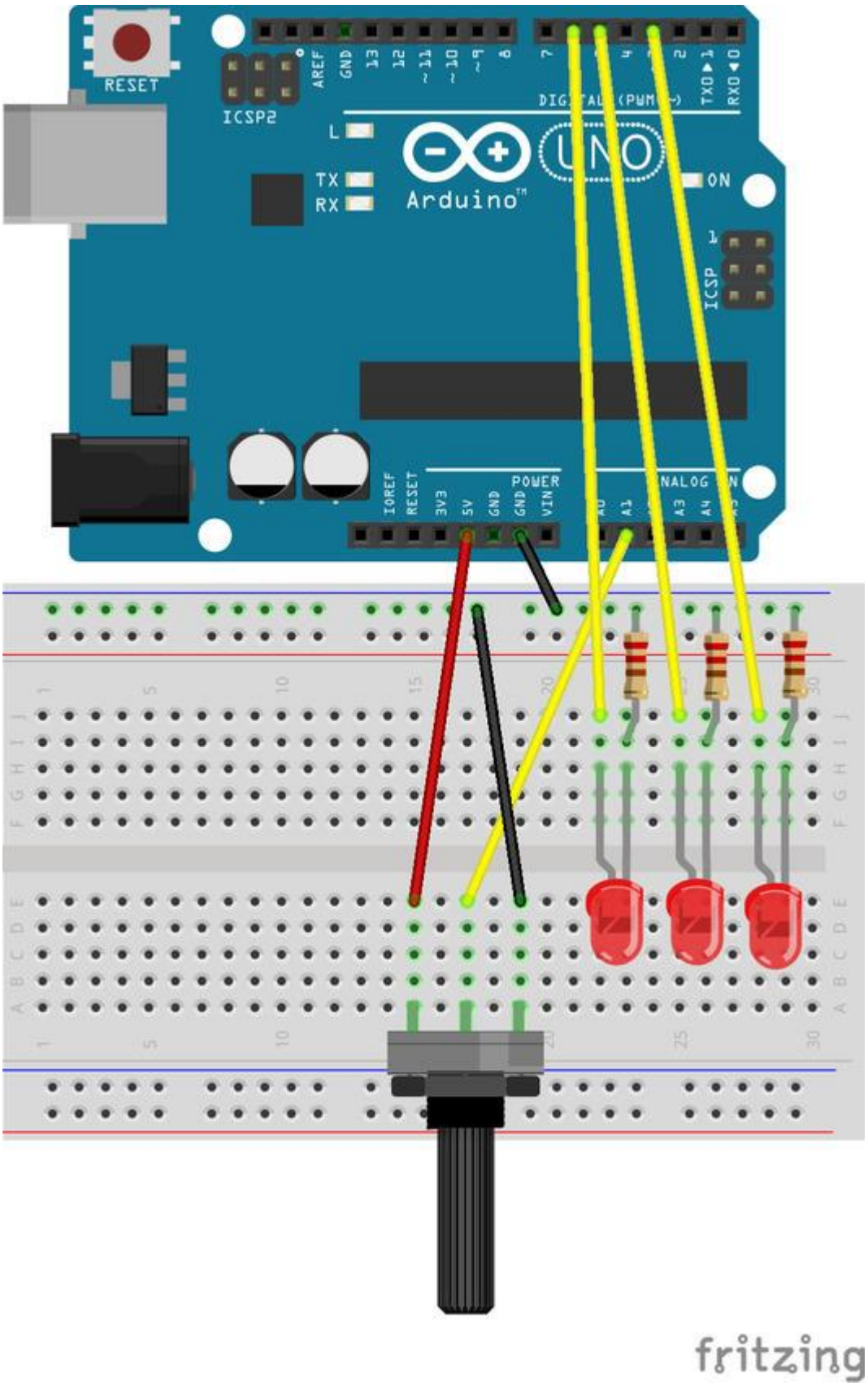
Если ручку потенциометра выкрутить в другую сторону, чтобы напряжение на центральном выводе, подключенном к A0 стало равно 5В (значение A0 = 1023), то статус разъема 6 будет равен 255 (1023 разделить на 1023 и умножить на 255), и напряжение на нем будет равно 5В. Светодиод будет ярко светиться.

Сохраните проект.


## Ремонт трехлинейного дезинтегратора

*Ребята молодцы, делают успехи в изучении электроники. Сегодня подсунул им неисправный дезинтегратор. Этот вид оружия стреляет тремя параллельными лучами, которые фокусируются на цели. Все три луча имеют одинаковую мощность и управляются от одного регулятора.*

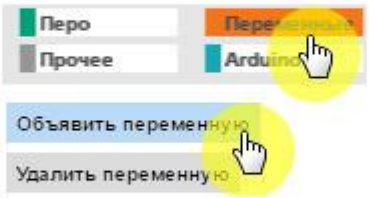
Соберите схему с потенциометром, тремя резисторами и тремя светодиодами. Используйте резисторы номиналом 330 Ом.



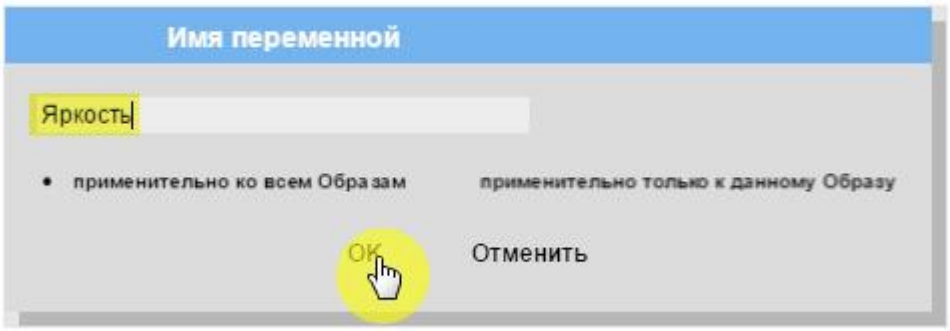

Средний вывод потенциометра соедините с пином A0, а длинные ножки светодиодов с пинами 3, 5 и 6.

Запустите среду Snap4Arduino и установите соединение с платой Arduino.

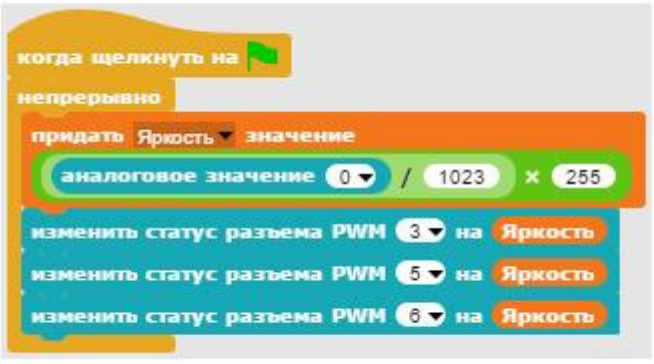
Объявите переменную ***Яркость***. Выберите оранжевые блоки **Переменные** и нажмите на кнопку **Объявить переменную**.




Введите имя переменной и нажмите **OK**.




Соберите следующий скрипт.



При запуске программы переменной ***Яркость*** будет присвоено значение, соответствующее напряжению на центральном выводе потенциометра и вычисленное по знакомой вам формуле. При вращении ручки потенциометра значение переменной будет изменяться от 0 до 255.

Три блока ***изменить статус разъема PWM*** будут устанавливать напряжение на выводах 3, 5 и 6 в соответствии со значением переменной. Получится, что все три светодиода будут светить одинаково ярко.

Сохраните проект.

**Задания.**

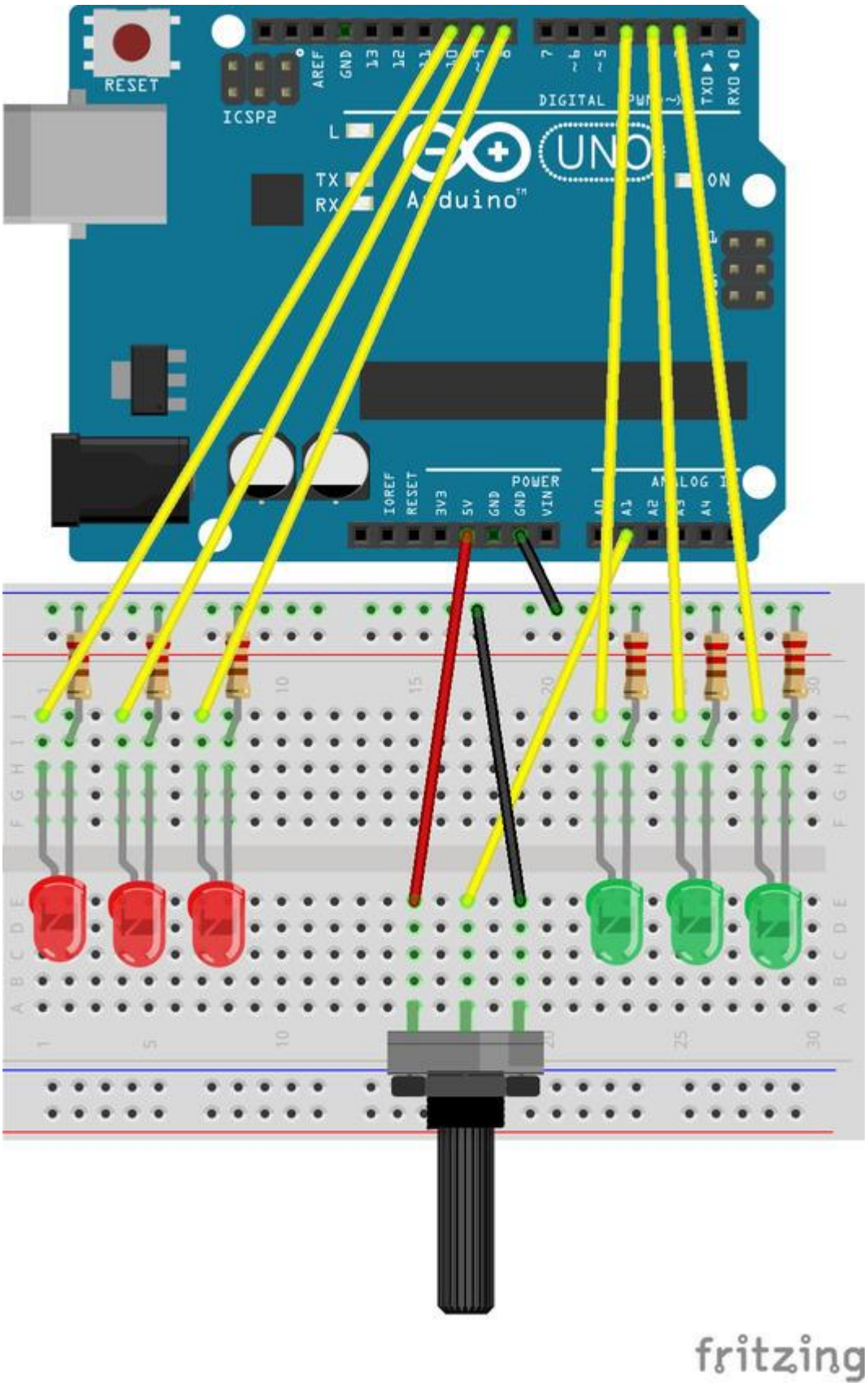
1. Подключите светодиоды к выводам 9, 10 и 11.
2. Сделайте так, чтобы средний светодиод светился в два раза слабее остальных.

# Ремонт сигнальных огней

*Правила космического движения предписывают каждому кораблю иметь по три зеленых фонаря на правом борту, и три красных на левом. При движении прямо должен гореть один зеленый и один красный фонарь. При повороте направо зеленые огни должны включаться один за другим, указывая на направление поворота. Аналогично «бегут» красные огни при повороте налево.*

*Пока объяснял ребятам правила космического движения, то заметил, что поворотники неисправны. Отправил пацанов ремонтировать. Если смогут починить, то завтра покажу им, как управлять космическим кораблем.*

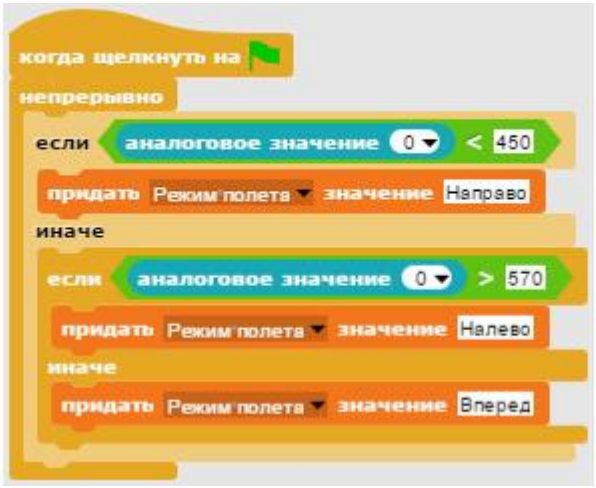
Соберите следующую схему. Используйте резисторы номиналом 330 Ом.




Зеленые светодиоды установите справа и подключите их длинные ножки к выводам 2, 3 и 4. Красные светодиоды установите слева и подключите их длинные ножки в выводам 8, 9 и 10.

Запустите среду Snap4Arduino и установите соединение с платой Arduino, как написано в приложении 1.

Объявите переменную ***Режим полета***. Соберите первый скрипт, изменяющий значение переменной.

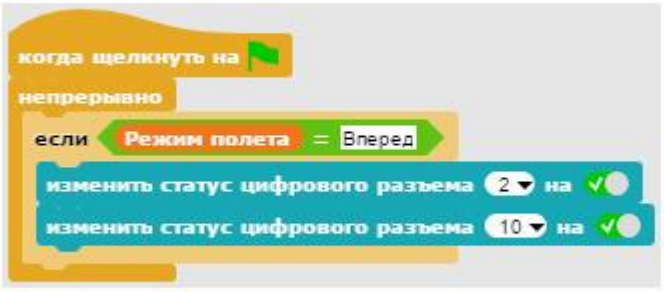



При повороте ручки потенциометра вправо, аналоговое значение, считываемое с пина A0, станет меньше 450 и переменной ***Режим полета*** будет присвоено значение «Направо».

При повороте ручки потенциометра влево, аналоговое значение, считываемое с пина A0, станет больше 570 и переменной ***Режим полета*** будет присвоено значение «Налево».

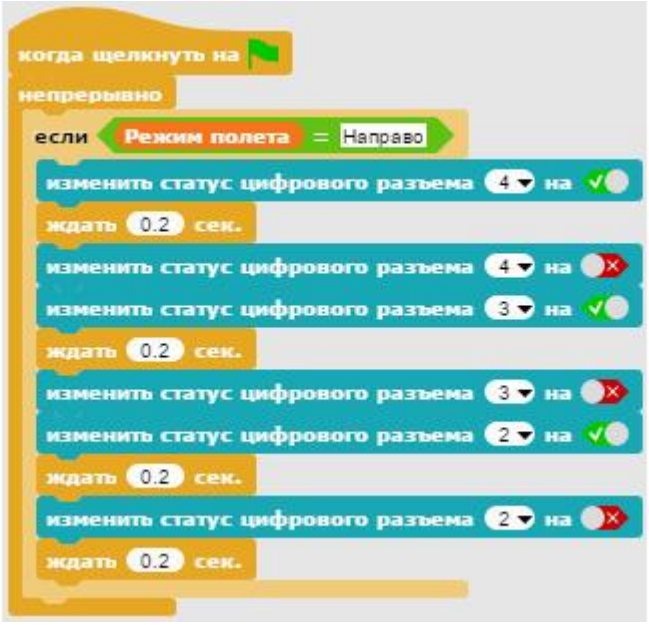
При установке ручки потенциометра в центральное положение, переменной ***Режим полета*** будет присвоено значение «Вперед».

Второй скрипт исполняется, если значение переменной ***Режим полета*** равно «Вперед».




В этом случае на 2 и 10 выводы подается высокий уровень напряжения, и светятся самый правый и самый левый светодиоды.

Второй скрипт исполняется, если значение переменной ***Режим полета*** равно «Направо».




В этом случае на выводы 4, 3 и 2 поочередно подается и снимается напряжение с задержкой в одну пятую долю секунды. При этом зеленые светодиоды загораются по очереди, указывая направление поворота корабля.

Третий скрипт исполняется, если значение переменной ***Режим полета*** равно «Налево».


В этом случае на выводы 8, 9 и 10 поочередно подается и снимается напряжение с задержкой в одну пятую долю секунды. При этом красные светодиоды загораются по очереди, указывая направление поворота корабля.

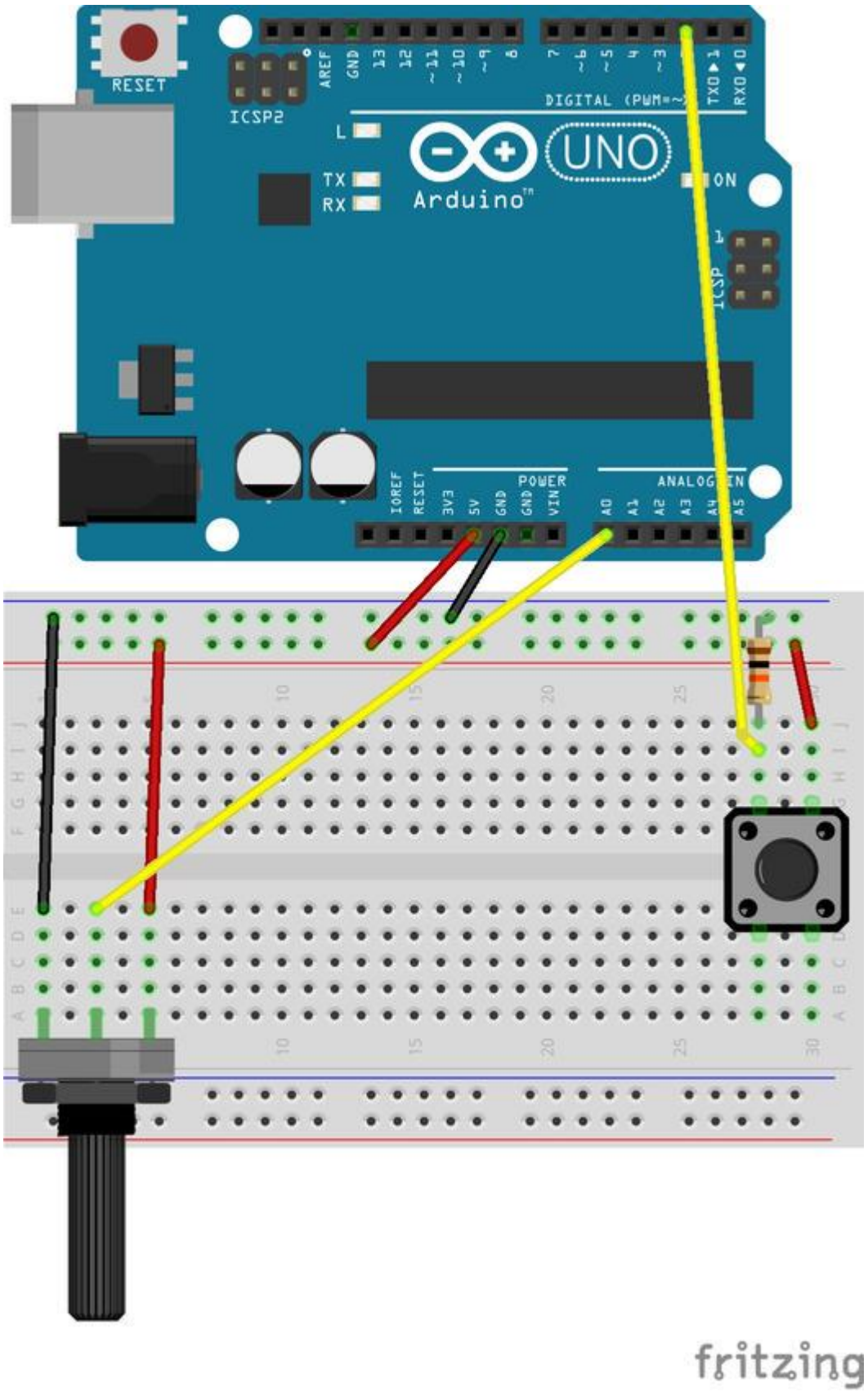
Сохраните проект.


# Стрельба по космическому мусору

*Пока ребята ремонтировали поворотники, я приметил вдали большую кучу космического мусора. Это же отличная возможность потренироваться в стрельбе! Ремонтировать боевой лазер пацаны уже научились, надо бы показать им как он стреляет. Я отвел корабль на позицию и разрешил ребятам открыть огонь по космическому хламу.*

## Сборка схемы

Соберите следующую схему на макетной плате.



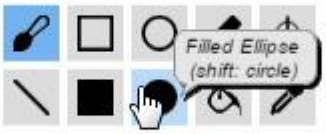

Кнопка подключена к выводу 2. Через резистор этот вывод постоянно соединен с землей, поэтому пока кнопка не нажата, то на выводе 2 низкий уровень напряжения 0В. Если же кнопку нажать, то на вывод 2 будет подан высокий уровень напряжения 5В.

## Подготовка спрайтов

Загрузите проект с управлением лазерной пушкой с помощью потенциометра и установите соединение с платой Arduino как написано в приложении 1.

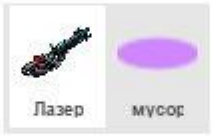
Теперь добавим в проект спрайт космического мусора. Нажмите на кнопку рисования нового спрайта (**paint a new sprite**).


Выберите инструмент **Закрашенный овал** (**Filled Ellipse**).




Выберите цвет космического мусора.

Нарисуйте маленький овальчик, или что-нибудь похожее на кусок космического мусора.

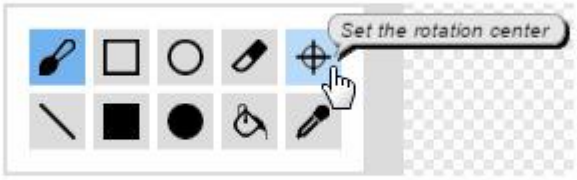



Осталось нарисовать лазерный луч, и можно программировать.
Нажмите на кнопку **нарисовать новый спрайт (paint a new sprite)**.


Выберите инструмент **Линия (Line tool).**


Выберите цвет лазера.

Удерживая нажатой клавишу **Shift**, нарисуйте длинную горизонтальную линию. Затем выберите инструмент **Установить центр вращения (Set the rotation center)**.

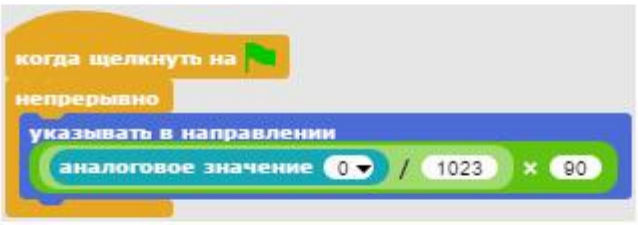


Установите центр вращения лазерного луча как показано на рисунке.

Все три спрайта готовы, начинаем программировать.

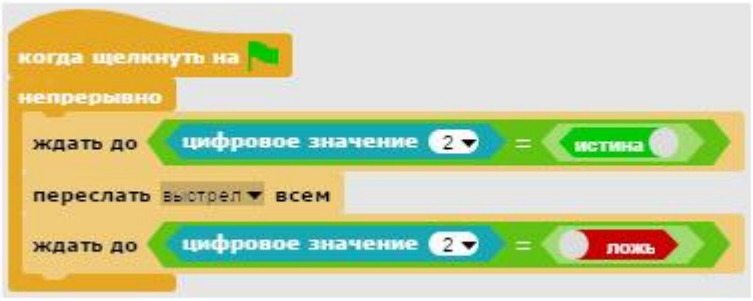

## Программирование

Выберите спрайт Лазерной пушки, у нее уже есть один скрипт.




Первый скрипт Лазерной пушки

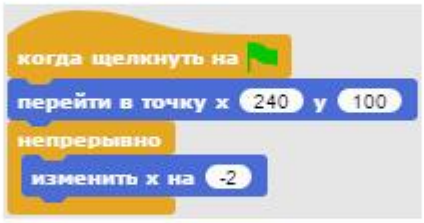
Первый скрипт постоянно считывает аналоговое значение с пина A0 и поворачивает Лазерную пушку в соответствующее направление от 0 до 90 градусов.




Второй скрипт Лазерной пушки

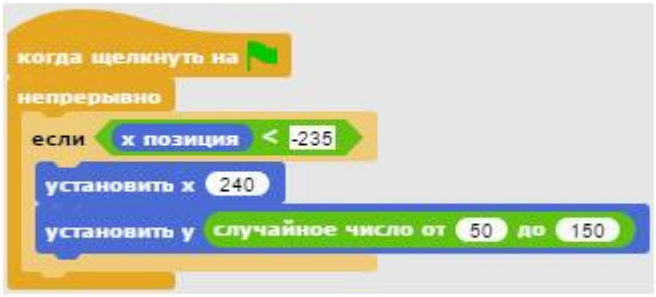
Второй скрипт Лазерной пушки ждет нажатия на кнопку, подключенную к пину 2, и пересылает сообщение ***выстрел***, когда она нажата. Затем скрипт ждет момента, когда кнопка будет отпущена. Без этого выстрелы вылетали бы один за другим как из пулемета, но у нас не пулемет, а пушка.

Выберите спрайт космического мусора и соберите для него программу из трех скриптов.




Первый скрипт космического мусора

Первый скрипт космического мусора перемещает его в правую часть сцены и непрерывно смещает его влево, изменяя координату X на -2. Мусор летит не очень быстро, чтобы вы успели прицелиться и попасть.



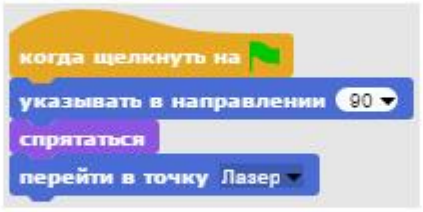

Второй скрипт космического мусора

Второй скрипт космического мусора постоянно проверяет, не долетел ли мусор до левой границы сцены. Если вы не попали в него, и мусор долетел, то он незамедлительно появится на правой границе сцены с координатой Y выбранной случайным образом из диапазона от 50 до 150. То есть все обломки космического мусора будут лететь на разной высоте.


Третий скрипт космического мусора

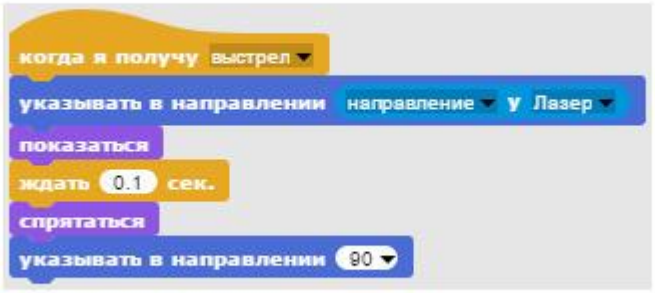
Третий скрипт космического мусора непрерывно ждет касания лазерного луча. Если он коснется луча, то он незамедлительно появится на правой границе сцены с координатой Y выбранной случайным образом из диапазона от 50 до 150.

Выберите спрайт лазерного луча и соберите для него программу из двух скриптов.




Первый скрипт луча

Первый скрипт лазерного луча при запуске проекта повернется вправо, спрячется и перейдет в точку Лазер. При этом центр костюма Лазерной пушки будет совпадать с центром лазерного луча.




Второй скрипт луча

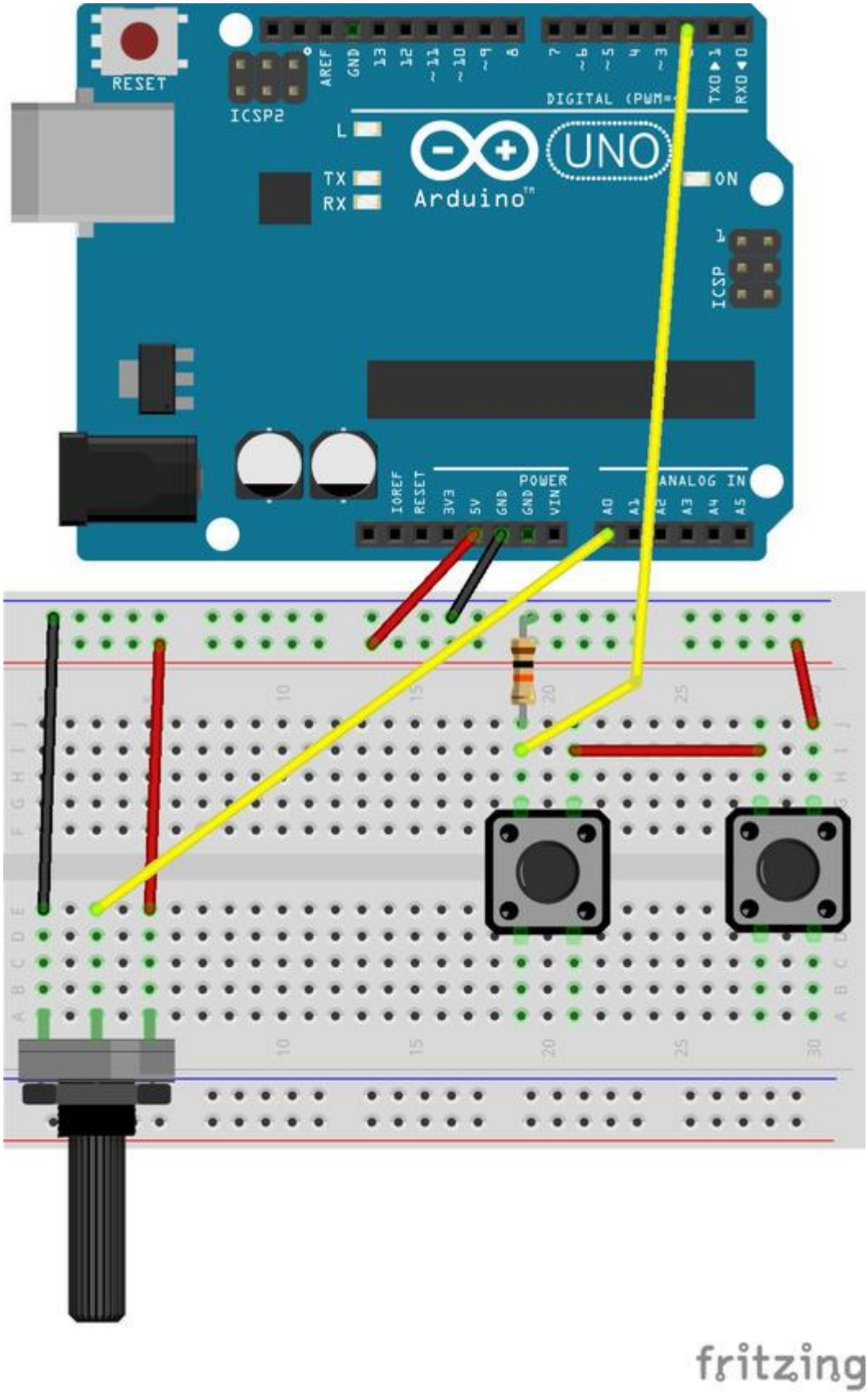
Второй скрипт лазерного луча выполняется при получении сообщения ***выстрел***. Лазерный луч повернется в направлении пушки, покажется на одну десятую долю секунды, спрячется, и повернется в направлении 90 градусов.

Протестируйте работу проекта и сохраните его.

## Добавляем вторую кнопку

*Стрелять моим беспризорникам понравилось, и они даже устроили небольшое соревнование. Победил Валя, мне кажется, из него получится неплохой снайпер. Ребята радостно носились по рубке, пока я разворачивал корабль, и вдруг случайно упали на пульт управления лазером. Раздался выстрел, и мы чуть не разнесли в клочья радиотранслятор, который был неподалеку. Только это еще не хватало, через 10 минут тут были бы ремонтники в сопровождении боевых кораблей. Я вежливо сказал ребятам на великорусском языке все, что о них думаю, и приказал подключить к лазерной пушке дополнительную кнопку, предохраняющую от случайных выстрелов. Я не думаю, что пацаны большие знатоки великорусских эпитетов, но мой тон заставил их собраться и быстренько добавить предохранитель к оружию.*

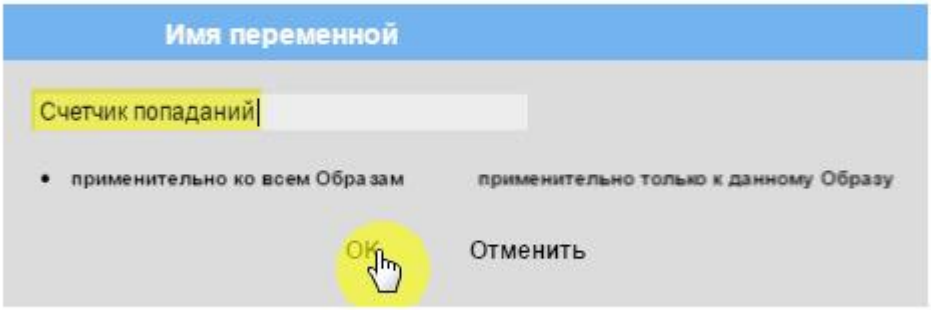
Доработайте предыдущий проект, подключив вторую кнопку последовательно с первой. Разместите кнопки на расстоянии друг от друга, чтобы было невозможно случайно нажать их вместе.



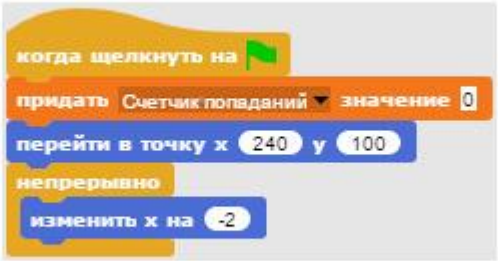
Теперь лазерная пушка будет стрелять только при одновременном нажатии на обе кнопки. Это не позволит произвести случайный выстрел, случайно нажав на одну из кнопок.

## Дорабатываем программу

Проект тоже доработаем, добавим в него счетчик попаданий. Объявите переменную с именем ***Счетчик попаданий***.

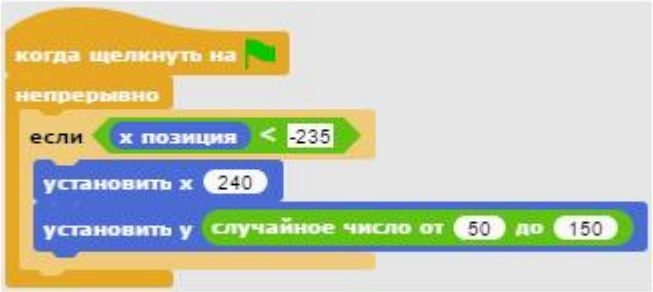



Немного измените первый и третий скрипты космического мусора. В первый скрипт добавьте обнуление переменной.




Первый скрипт космического мусора

Второй скрипт космического мусора оставьте без изменений.




Второй скрипт космического мусора

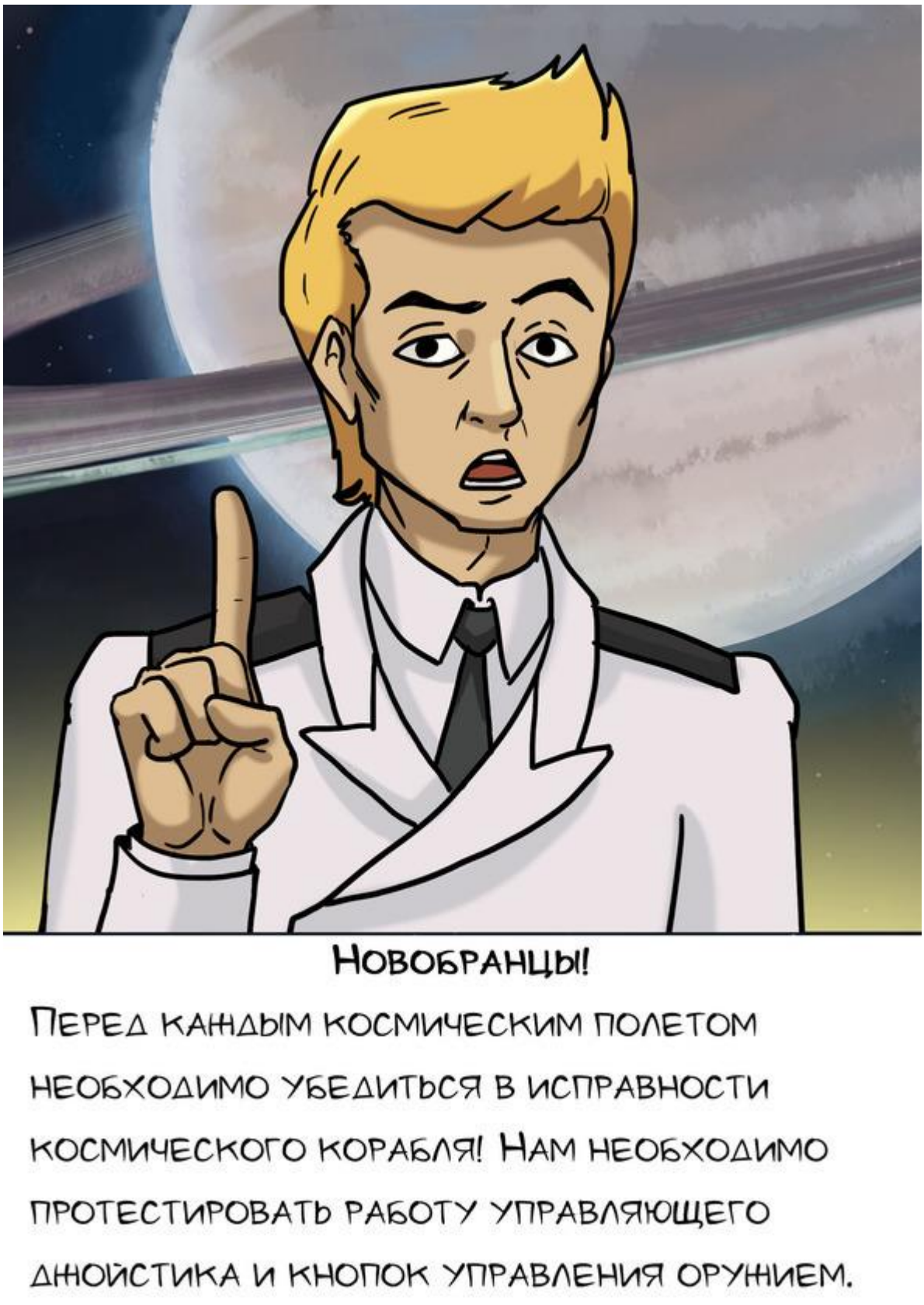
В третий скрипт добавьте счетчик попаданий.


Третий скрипт космического мусора

Теперь при попадании по космическому мусору будет увеличиваться значение переменной ***Счетчик попаданий***, и вы сможете устроить соревнование – кто собьет больше мусора за минуту.

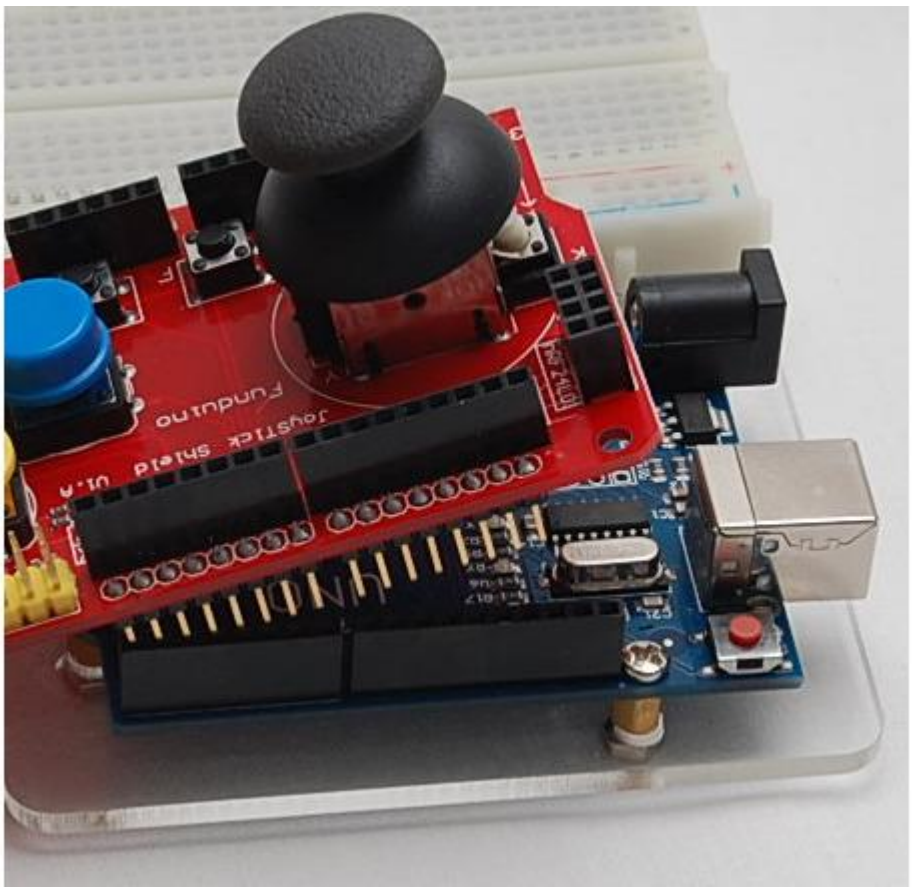
Не забудьте сохранить проект!

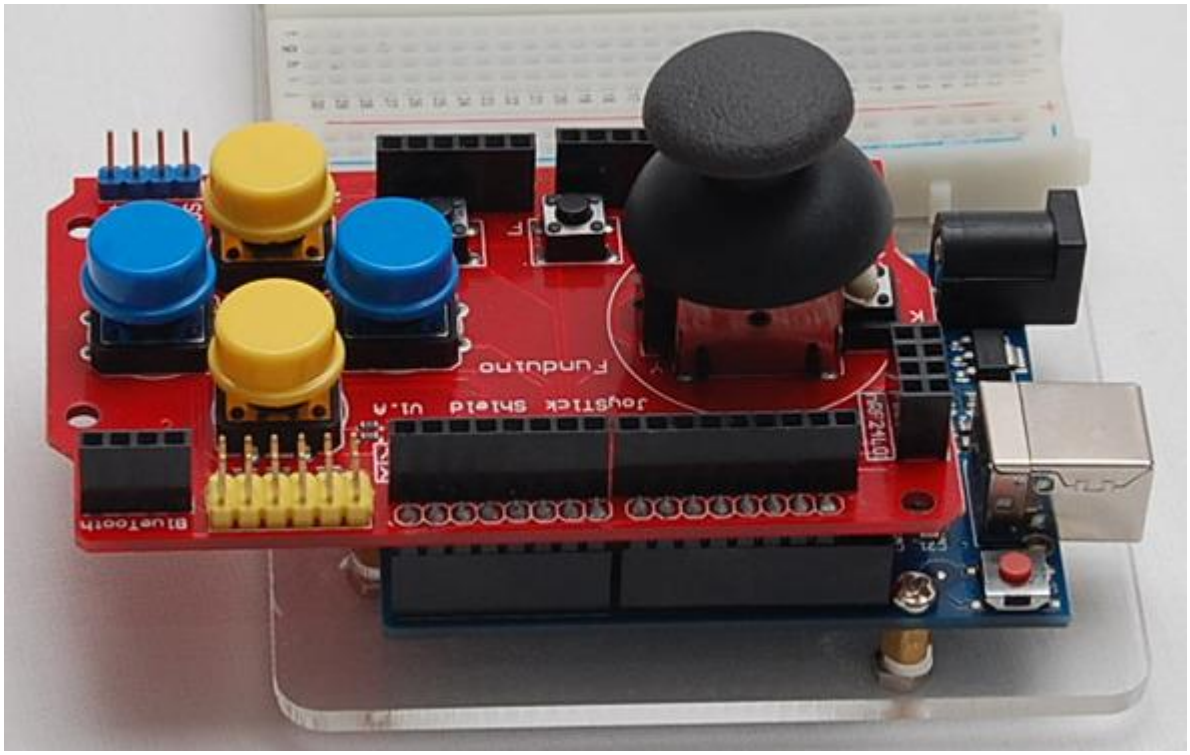

# Тест платы с джойстиком




## Установка платы с джойстиком

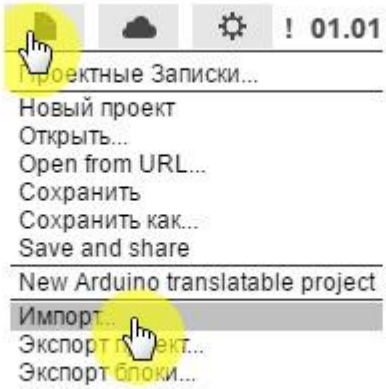
Установите плату с джойстиком на Arduino как показано на рисунках.



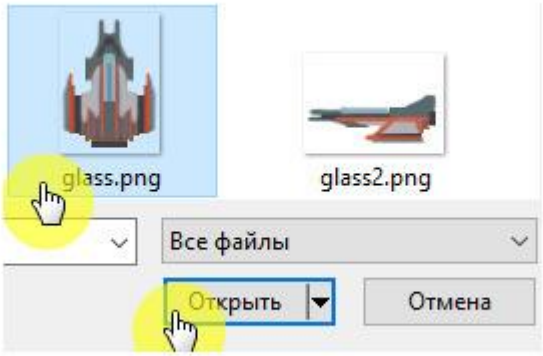




## Подготовка спрайтов и сцены

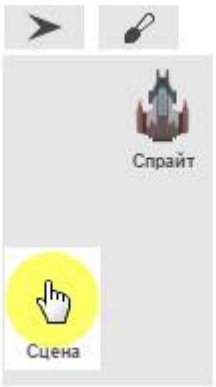
Создайте новый проект и импортируйте изображение космического корабля.



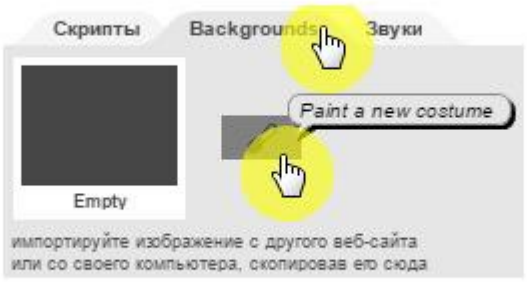

Выберите файл glass.png.



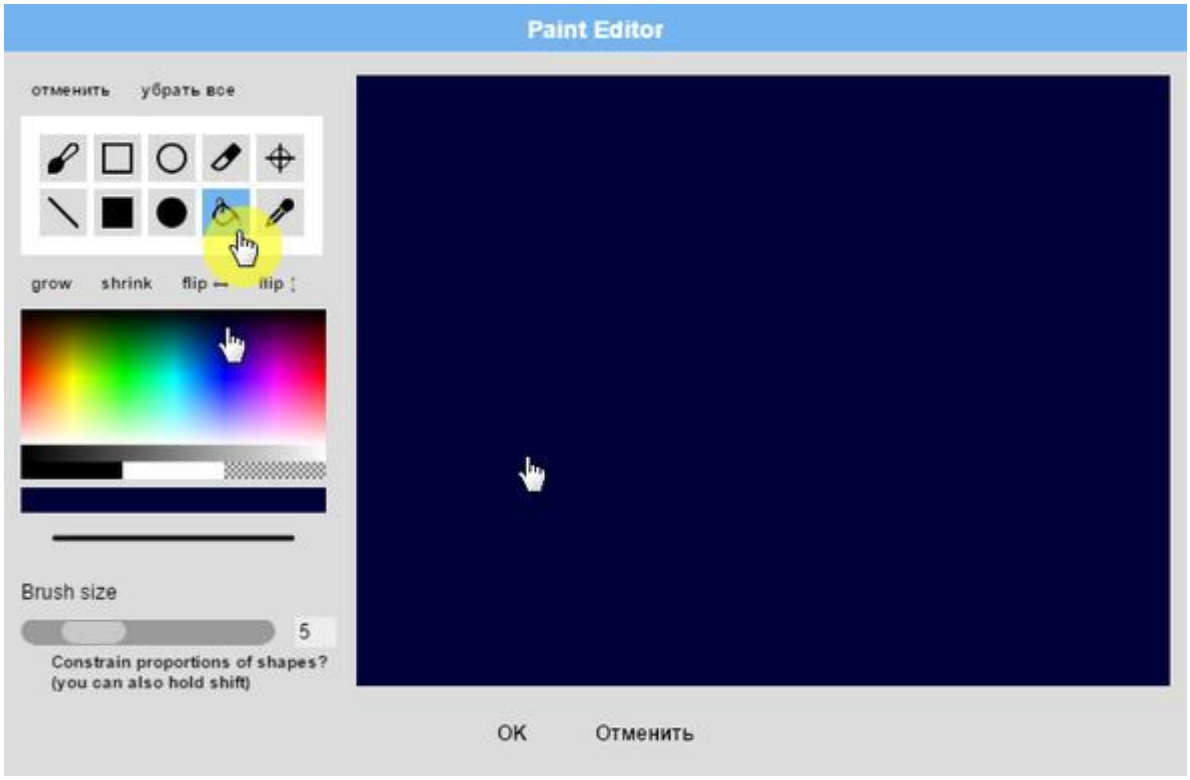

Теперь измените белый фон на что-нибудь более похожее на космос. Выберите сцену.




Перейдите на вкладку **Backgrounds** и нажмите на кисточку.

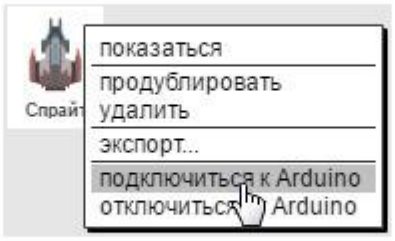



В открывшемся редакторе выберите инструмент **Заливка (Fill the region)**, выберите цвет, кликнув по палитре цветов, залейте фон темным цветом, и нажмите **ОК**.

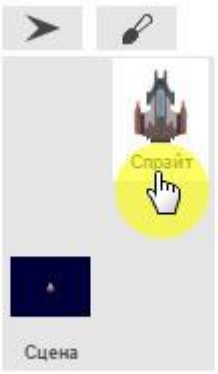



Почти настоящий космос!

Теперь подключите плату Arduino к спрайту космического корабля. Для этого кликните по спрайту правой кнопкой мышки и выберите команду **подключиться к Arduino**.



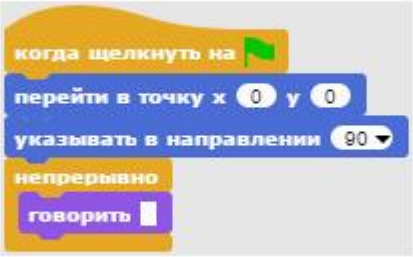

Теперь необходимо запрограммировать космический корабль, выберите его в области спрайтов.




Перейдите на вкладку **Скрипты**.


## Программирование

Создайте следующий скрипт, благодаря которому космический корабль будет сообщать о значениях, полученных от аналоговых пинов A0 и A1.




В блок ***говорить*** необходимо добавить функцию ***объединить***, которая соединяет вместе несколько строк текста.


По умолчанию эта функция объединяет две текстовых строки, однако количество объединяемых строк можно увеличить, нажимая на черный треугольничек.

Теперь введите в первое и третье окошки соответствующий текст и установите блоки ***аналоговое значение 0*** и ***аналоговое значение 1*** во второе и четвертое окошки.


Перетащите полученную функцию ***объединить*** в блок ***говорить***.


Скрипт проверки джойстика готов, запустите его, нажав на зеленый флажок. Сохраните полученный проект и протестируйте его работу. Запомните значения, которые соответствуют центральному положению ручки джойстика.


## Тест кнопок управления оружием

Далее следует проверить работоспособность кнопок управления оружием. Создайте новый проект и подключите плату Arduino. Импортируйте изображение космического корабля, и окрасьте сцену в темный цвет.

Создайте следующий скрипт, благодаря которому космический корабль будет сообщать о значениях, полученных от цифровых пинов 2, 3, 4 и 5.


Скрипт проверки кнопок готов, сохраните его и запустите, нажав на зеленый флажок. Нажимайте на кнопки управления оружием, и посмотрите, как будет изменяться текст.


# Управление маневровыми двигателями

*Система управления кораблем исправна, и теперь можно приступать к полетам в открытом космосе. Однако перед этим я проведу небольшой инструктаж о конструкции нашего космического корабля.*


А сейчас вы научитесь управлять маневровыми двигателями.

Создайте новый проект и подготовьте его к работе. Импортируйте спрайт космического корабля, окрасьте сцену в темный цвет и подключите плату Arduino как описано в приложении 1.

## Программа без переменных

Создайте следующий скрипт для космического корабля.


Скрипт космического корабля

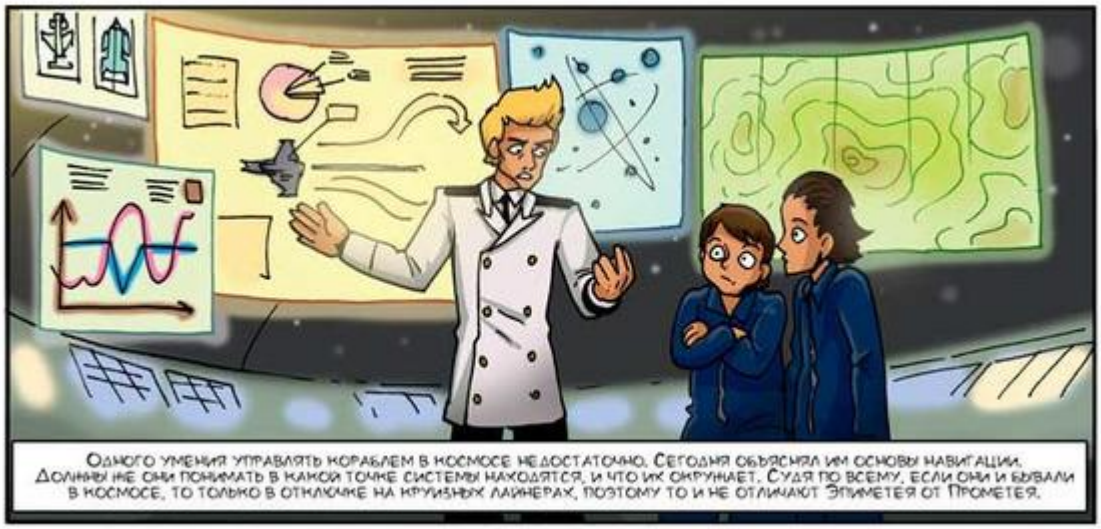
При запуске программы корабль перейдет в центр сцены и повернется в направлении 90 градусов. Затем он постоянно будет считывать значения аналоговых датчиков A0 и A1 и проверять, не отклонена ли ручка джойстика. Числовые значения, с которыми происходит сравнение, должны быть на 10 больше и на 10 меньше значения, которое соответствует центральному положению джойстика. Сохраните проект и протестируете его работу.

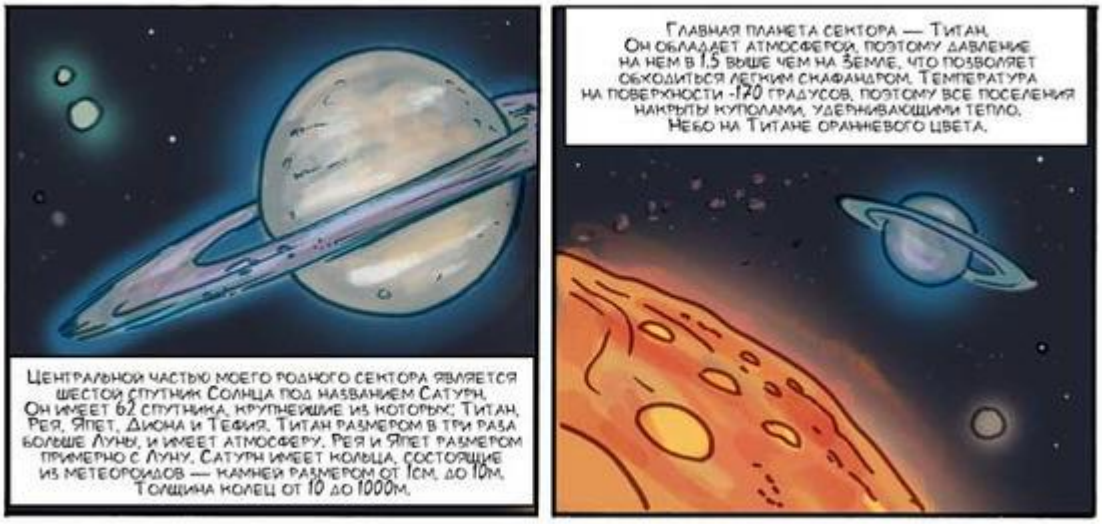
## Задание.

Измените скорость перемещения космического корабля.

Учтите, что маневровые двигатели не могут развить скорость более 3 шагов.


Только сумасшедший доверит новичкам управление звездолетом без тренировки на тренажере. Вчера я не на шутку испугался за сохранность моего корабля, и сегодня решил соорудить тренажер из подручных средств. На счастье у меня в запасе было несколько допотопных Космодуино и плата с кнопками и джойстиком.
Я собрал для пацанов простенький симулятор полета корабля, на котором они смогут потренироваться, прежде чем прикоснутся к пилотажному джойстику и боевым кнопкам.


Одного умения управлять кораблем в космосе недостаточно. Сегодня объяснял им основы навигации. Должны же они понимать в какой точке системы находятся, и что их окружает. Судя по всему, если они и бывали в космосе, то только в отключке на круизных лайнерах, поэтому то и не отличают Эпиметея от Прометея.

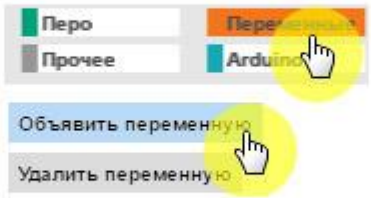



## Использование переменных

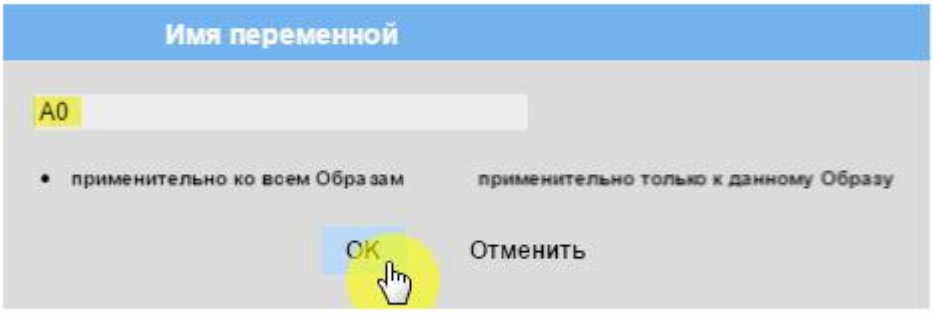
В предыдущем проекте при управлении маневровыми двигателями в цикле происходило четыре считывания аналоговых значений, два считывания с пина A0 и два с пина A1. Для повышения управляемости космического корабля переделаем проект с использованием переменных. Будем считывать значения с A0 и A1, и запоминать считанные значения в переменных.

Загрузите прошлый проект.

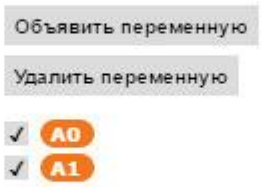
Объявите две переменные ***A0*** и ***A1***. Выберите оранжевые блоки **Переменные** и нажмите на кнопку **Объявить переменную.**



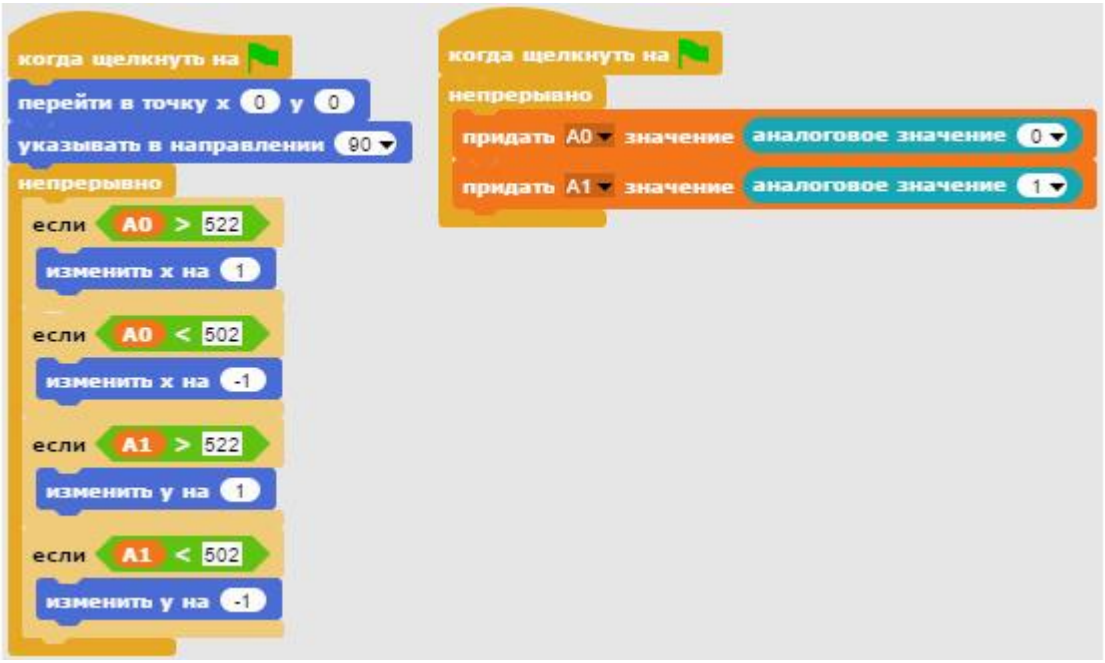

Введите имя переменной и нажмите **OK.**




Затем объявите вторую переменную. Объявленные переменные отображаются в виде оранжевых овальчиков. Слева от переменных установлены галочки, поэтому переменные отображаются на сцене.




Соберите следующие скрипты.




**Обратите внимание!**

Оба скрипта начинают работать одновременно при нажатии на зеленый флажок. Правый скрипт постоянно считывает значения с аналоговых пинов A0 и A1 и сохраняет полученные значения в соответствующих переменных. Левый скрипт устанавливает космический корабль в центр сцены, поворачивает его в направлении 90 градусов и постоянно сравнивает значения переменных ***A0*** и ***A1*** с пороговыми значениями.

Сохраните проект и протестируйте его.

ЗАДАНИЯ.
1. ИЗМЕНИТЕ ПРОГРАММУ ТАК, ЧТОБЫ ПРИ НАЖАТИИ НА ЗЕЛЕНЫЙ ФЛАНОК КОРАБЛЬ СТАРТОВАЛ ИЗ НИЖНЕЙ ЧАСТИ СЦЕНЫ, ИЗ ТОЧКИ С КООРДИНАТАМИ (0; -140).
2. ЗАМЕДЛИТЕ СКОРОСТЬ ПОЛЕТА КОРАБЛЯ ИСПОЛЬЗУЯ ДЕСЯТИЧНЫЕ ДРОБИ ВИДА 0.7, 0.5, 0.1.

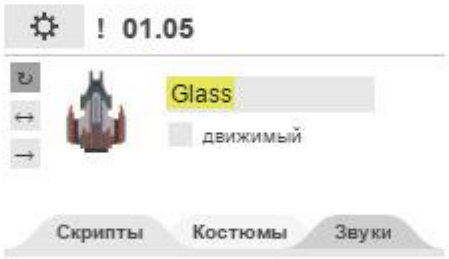
ПРИ КАКОМ ЗНАЧЕНИИ СКОРОСТЬ ПОЛЕТА МИНИМАЛЬНАЯ?

## Змейка

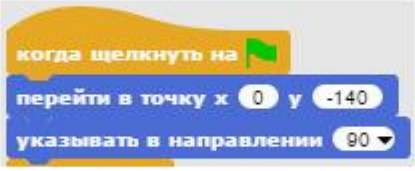

Добавим в прошлый проект буйки, которые будет нужно аккуратно облететь не коснувшись.

Загрузите прошлый проект.

Переименуйте спрайт космического корабля, для этого введите новое имя в окошке ввода над областью скриптов.

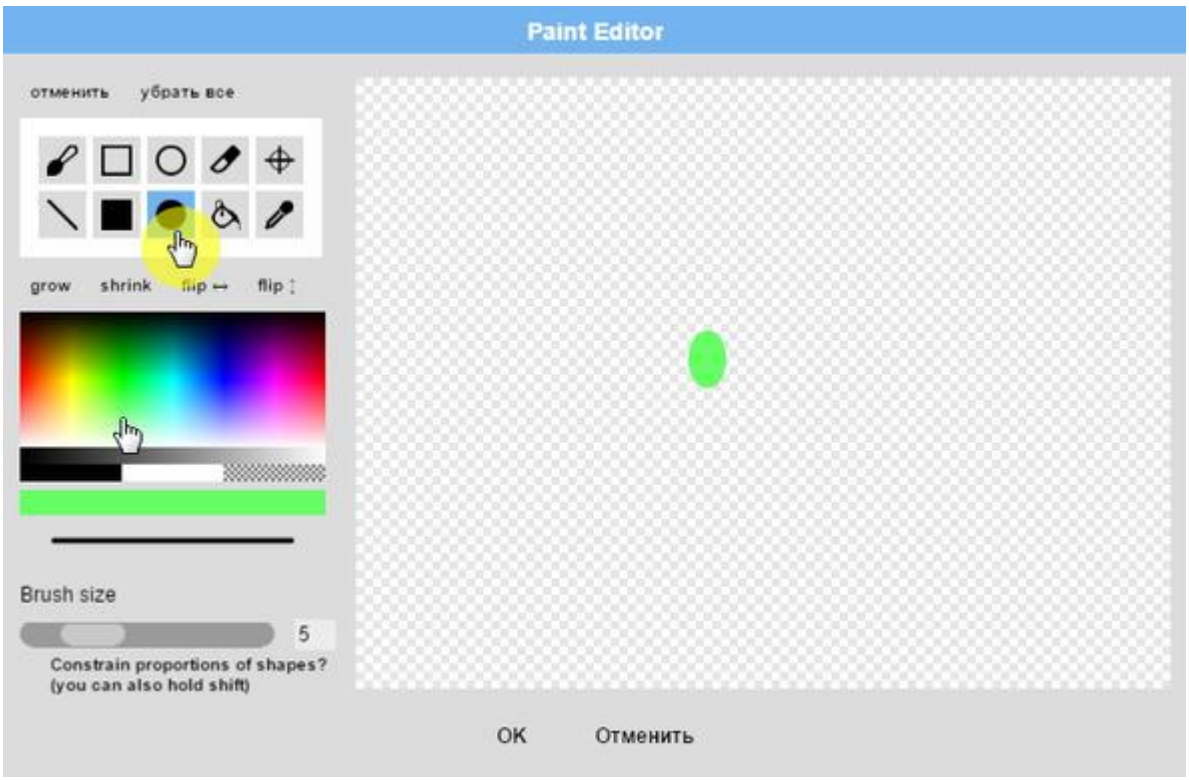



Измените первый скрипт космического корабля. Стартовать будем из нижней части сцены, поэтому начальная координата Y должна быть равна -140.

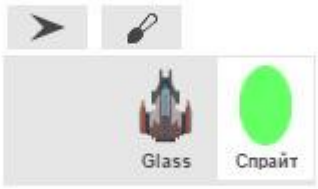



Теперь нарисуем буек. Кликните на кнопку с кисточкой **paint a new sprite**.

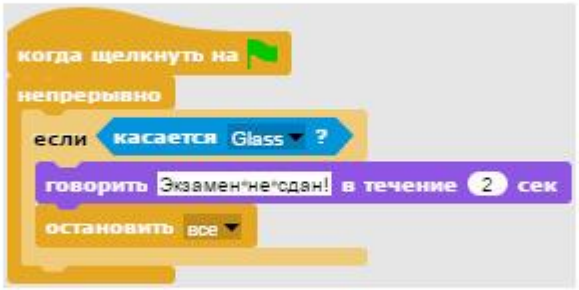

Откроется окно графического редактора. Выберите инструмент **Закрашенный круг**, зеленый цвет, и нарисуйте небольшой овал.




Нажмите на **ОК**. В области спрайтов появится еще один спрайт.




Запрограммируйте его. При касании космического корабля новый спрайт должен 2 секунды говорить: «Экзамен не сдан!», а затем завершать работу программы.



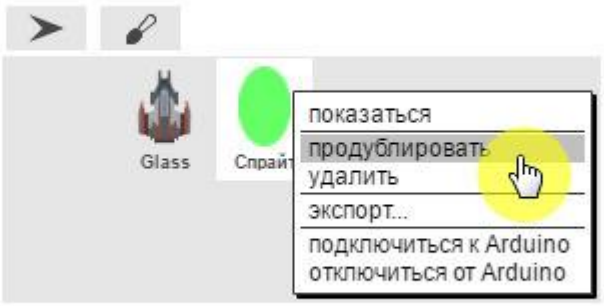

Скрипт буйка

**Обратите внимание!**

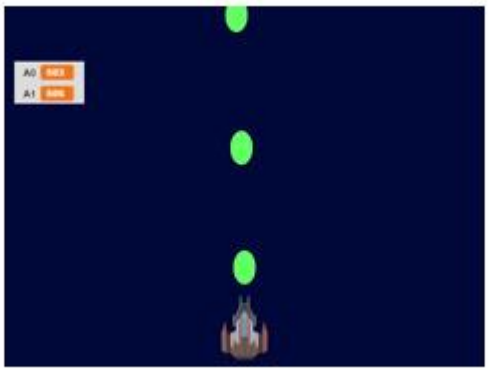
Здесь использован сенсорный блок ***касается***. У этого блока есть маленький черный треугольничек, который раскрывает список спрайтов, на которые может реагировать сенсор.


Протестируйте работу буйка, направьте на него космический корабль и убедитесь, что он ругается.

Одного буйка для змейки маловато, скопируйте его два раза. Для этого кликните правой кнопкой на спрайте и выберите **продублировать.**

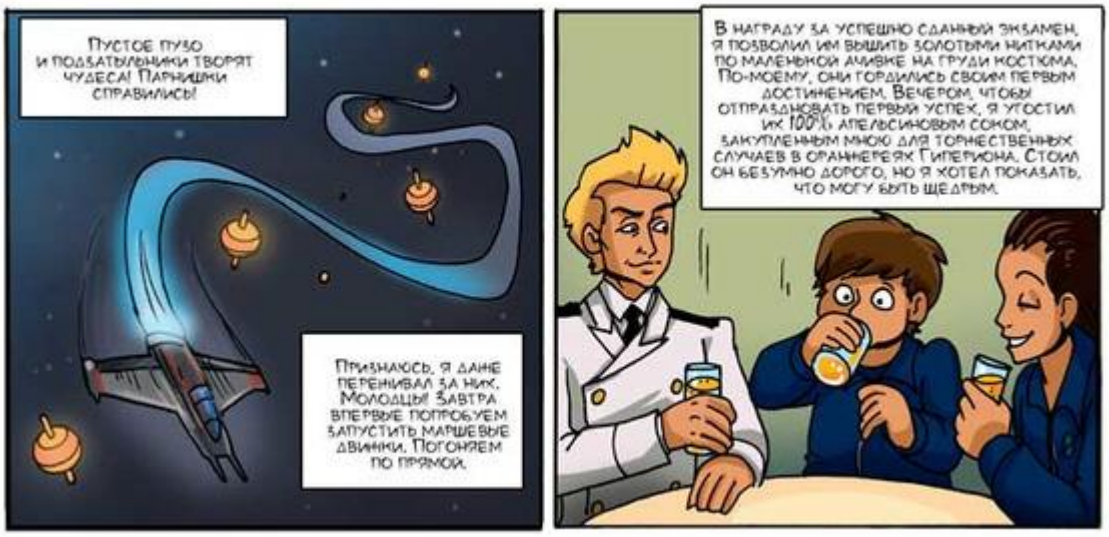



Разместите буйки в космическом пространстве.

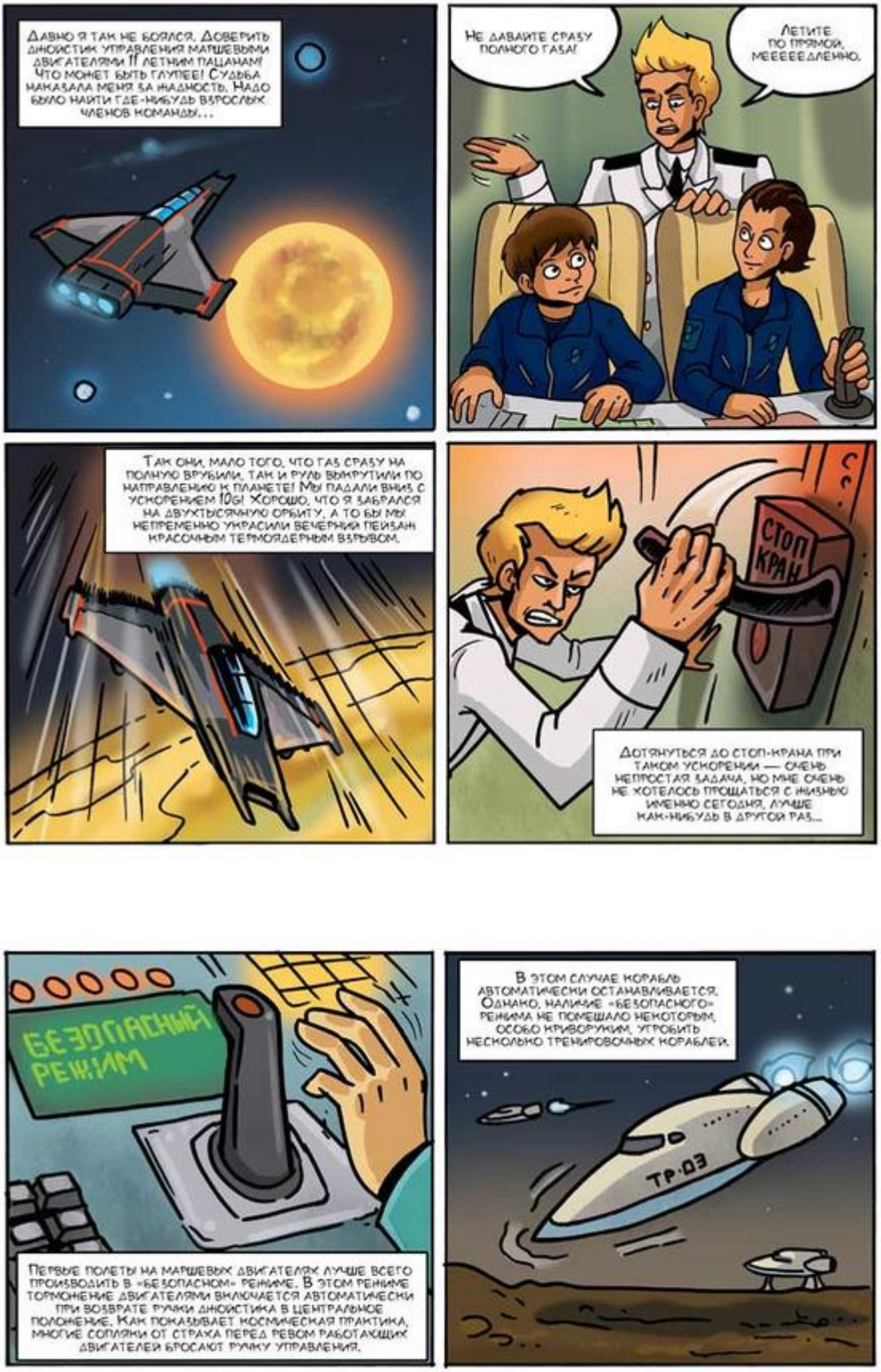


Сохраните проект.

Теперь все готово к началу экзамена! Облетите буйки и припаркуйтесь возле переменных. Берегите корабль, не касайтесь буйков!


Пустое пузо и подбатыльники творят чудеса! Парнишки справились!
Признаюсь, я даже переживал за них. Молодцы! Завтра впервые попробуем запустить маршевые двигатели. Погоняем по прямой.
В награду за успешно сданный экзамен, я позволил им вышить золотыми нитками по маленькой дичинке на груди костюма. По-моему, они гордились своим первым достижением. Вечером, чтобы отпраздновать первый успех, я угостил их 100% апельсиновым соком, закупленным мною для торжественных случаев в оранжереях Гипериона. Стоил он безумно дорого, но я хотел показать, что могу быть щедрым.


День начался с изучения строения космического корабля. Любой пилот должен полностью знать строение своего корабля, а первоклассным пилотом может называться лишь тот, кто знаком с конструкцией и управлением любого современного корабля! Среди моих знакомых есть один такой человек — это я. Удачно, ребята оказались немного умнее амебы, и через неделю тренировок смогли научиться управлению маневровыми.
Для того чтобы они не задирали нос после первого успеха я дал им задание выучить наизусть схему электропроводки корабля с назначением, длиной и сечением всех кабелей, а это таблица на 1200 строк! Пусть только попробуют не выучить, отправятся на корм дикой шаверме!




# Полет на маршевых двигателях

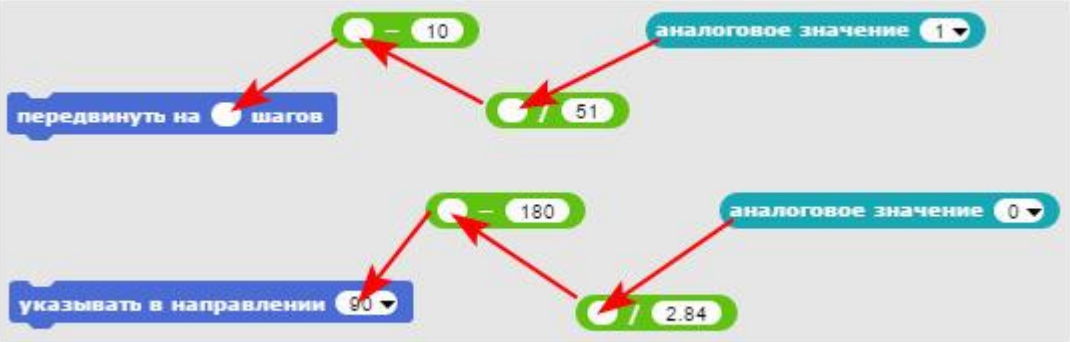
Создайте новый проект и подготовьте его к работе. Импортируйте спрайт космического корабля glass_2.png, окрасьте сцену в темный цвет и подключите плату Arduino как описано в приложении 1.

## Программа без использования переменных

Создайте следующий скрипт для космического корабля. Теперь корабль будет перемещаться без использования координат X и Y.


Первый скрипт космического корабля

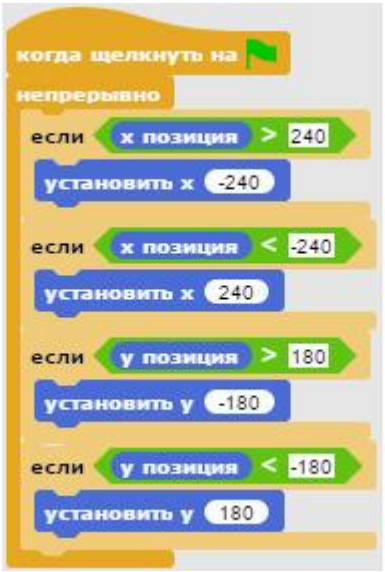
Для перемещения корабля использованы блоки ***передвинуть*** и ***указывать в направление***. Соберите их следующим образом.




Блок ***передвинуть*** перемещает корабль в указанном направлении. Скорость перемещения зависит от аналогового значения A1, и может изменяться от 0 до 10. Максимальное аналоговое значение A1 может быть равно 1023, и если разделить это число на 51 и вычесть 10, то максимальная скорость будет 10 шагов.

Блок ***указывать в направление*** поворачивает корабль в соответствии с аналоговым значением A0 и может изменяться от -180 градусов до 180 градусов. Максимальное аналоговое значение A0 может быть равно 1023, и если разделить это число на 2,84 и вычесть 180, то получится направление в 180 градусов.

Для того чтобы корабль не пропадал за границей сцены, а автоматически появлялся на противоположной стороне необходимо собрать следующий скрипт.




Второй скрипт космического корабля

При увеличении координаты X корабля более 240 (при нахождении на правой границе сцены), координата X будет установлена в -240, и корабль окажется на левой границе. И, наоборот, при уменьшении координаты X меньше -240, когда корабль на левой границе, координата X будет установлена в 240, и корабль окажется на правой границе сцены. Движение по вертикали происходит аналогично.

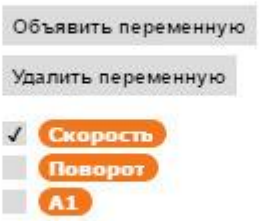
Корабль к полету готов! Сохраните проект и запустите его, нажав на зеленый флажок.


## Использование переменных

Создайте новый проект и импортируйте изображение космического корабля. Подключите плату Arduino и окрасьте сцену в темный цвет.

Пришло время научиться летать в маршевом режиме. В этом режиме корабль разгоняется при отклонении ручки джойстика от себя, и продолжает лететь по прямой с набранной скоростью. Для торможения нужно потянуть ручку джойстика на себя.

Сначала объявите три переменные: ***Скорость***, ***Поворот***, ***A1***.




Соберите первые два скрипта проекта.


Первые два скрипта космического корабля

Принцип работы левого скрипта вы уже знаете, он перемещает корабль в соответствии со значениями переменных. Правый скрипт при запуске программы обнулит значение переменной ***Скорость***, а затем постоянно, с интервалом в 0.3 секунды, будет изменять ее значение на величину A1, зависящую от положения ручки джойстика.

Следующий скрипт отвечает за управление джойстиком.

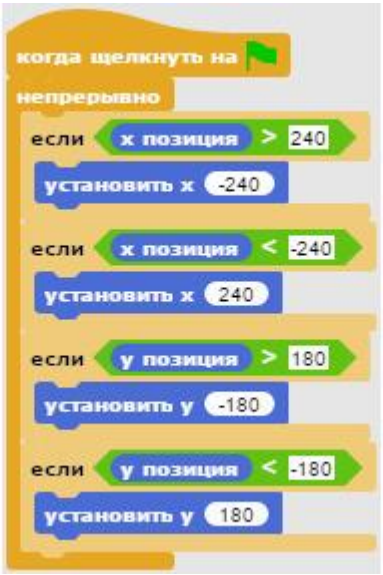

Третий скрипт космического корабля

За управление отвечают две переменные: ***A1*** и ***Поворот***.
Соберите блоки ***придать*** следующим образом.


Блок ***придать A1*** устанавливает величину изменения скорости. Не саму скорость, а только величину ее изменения! Ее величина зависит от аналогового значения A1, и может изменяться от -1 до 1. Максимальное аналоговое значение A1 может быть равно 1023, и если разделить это число на 500 и вычесть 1, то получится 1.048. Функция округлить округляет 1.048 до единицы. Минимальное аналоговое значение A1 может быть ноль, и если вычесть из него единицу, то получится минимальное значение переменной ***A1*** равное -1.

Переменная ***Поворот*** может принимать значение от -180 до 180. Как это получается, вы уже знаете.

Не забудьте добавить в проект четвертый скрипт, не позволяющий кораблю скрываться за краями сцены.



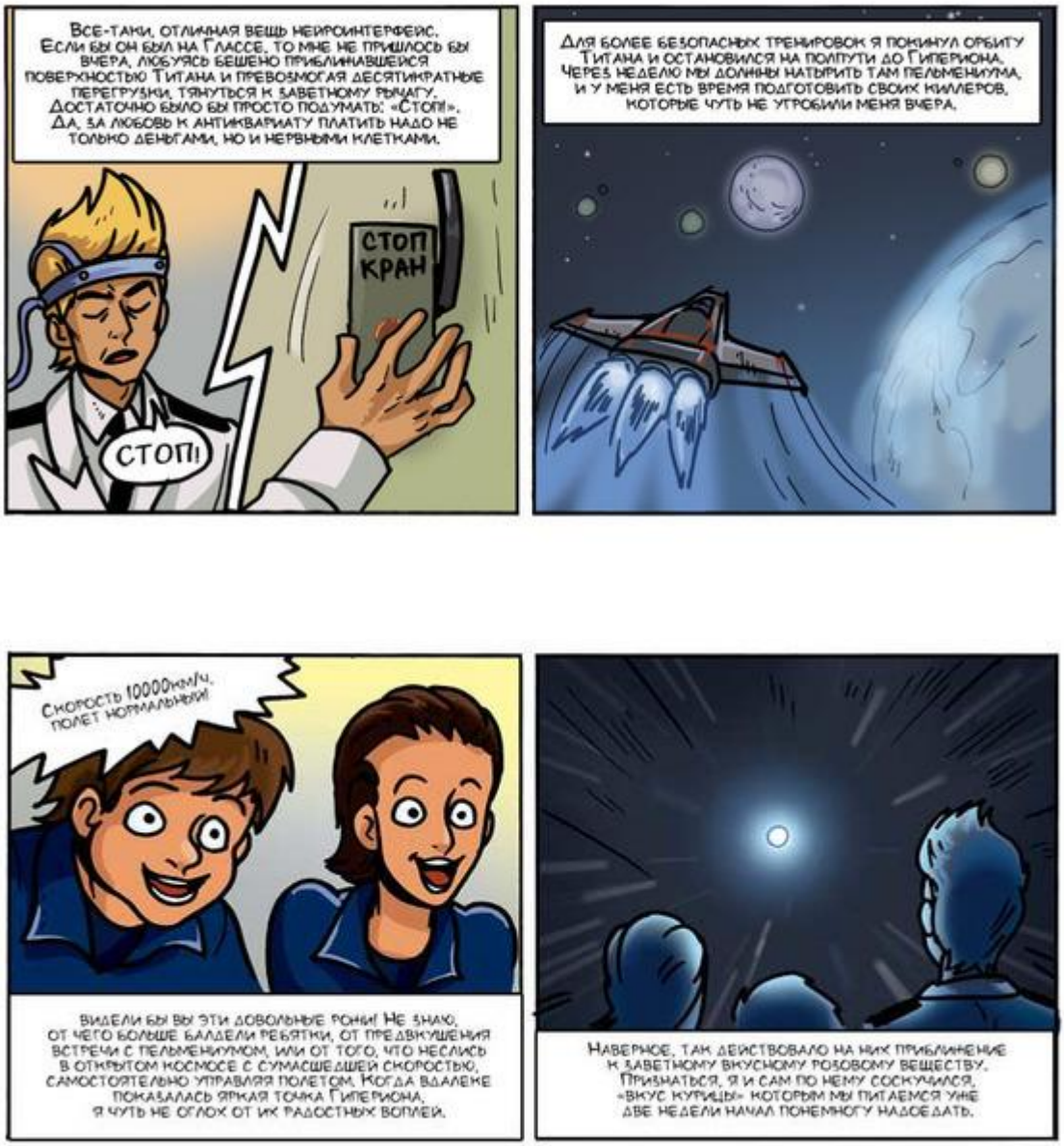

Четвертый скрипт космического корабля

Корабль к полету готов! Сохраните проект и запустите его, нажав на зеленый флажок.

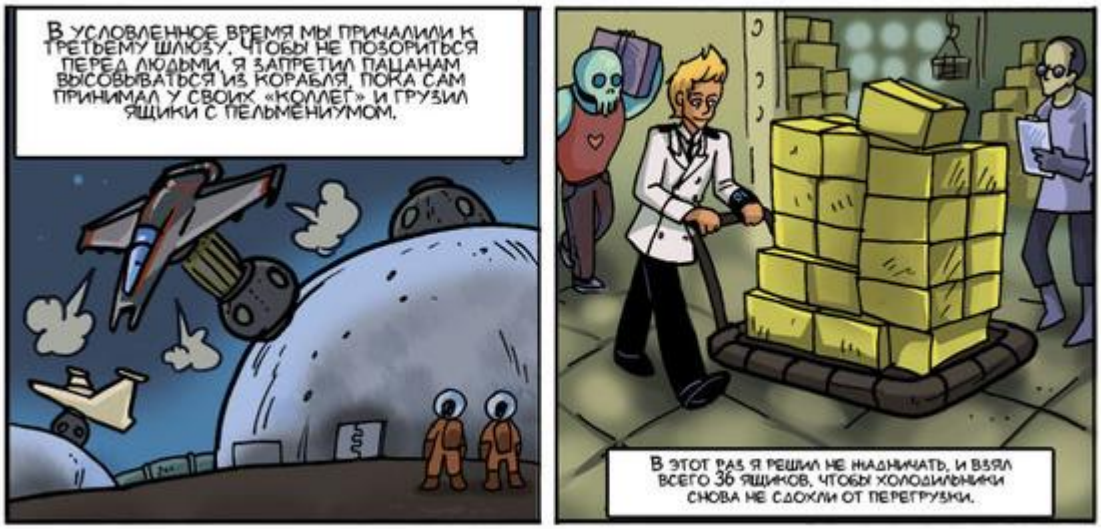
**Задание.**

Установите ограничение скорости.

Сделайте так, чтобы скорость не могла вырасти выше 10.




*Подлетая к Гипериону, я, как обычно, погасил все освещение, вошел в режим радиомолчания и остановился на безопасном расстоянии от пельменциумной фабрики. С сообщниками мы обменивались световыми сигналами уранских дальнобойщиков, отправляя их с помощью направленных сигнальных лазеров. Для передачи сигналов использовались 3 красных лазера, 3 зеленых и два синих. Я послал условное сообщение: «есть?» Они ответили: «20». Я сообщил: «беру 10». Они ответили: «в 23:15 шлюз 3».*




# Коды уранских дальнобойщиков

Пришло время познакомиться с космическими световыми сигналами, которые используются для передачи тайных сообщений. Эти сигналы придуманы уранскими дальнобойщиками, которые по-тихому проводят караваны кораблей и никогда не пользуются рациями.

В следующих таблицах горящие светодиоды обозначены буквами. К – красный светодиод, С – синий, а З – зеленый.

Например, если горят три зеленых светодиода, то это означает цифру 3, а если горит один красный и один синий, то это буква ё.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | з | | |
| 2 | | | | | | з | з | |
| 3 | | | | | | з | з | з |
| 0 | | | | с | | | | |
| 4 | | | | с | | з | | |
| 5 | | | | с | | з | з | |
| 6 | | | | с | | з | з | з |
| _ | | | | с | с | | | |
| 7 | | | | с | с | з | | |
| 8 | | | | с | с | з | з | |
| 9 | | | | с | с | з | з | з |
| . | к | | | | | | | |
| е | к | | | | | з | | |
| у | к | | | | | з | з | |
| а | к | | | | | з | з | з |
| ё | к | | | с | | | | |
| э | к | | | с | | з | | |
| ю | к | | | с | | з | з | |
| я | к | | | с | | з | з | з |
| ? | к | | | с | с | | | |
| о | к | | | с | с | з | | |
| и | к | | | с | с | з | з | |
| ы | к | | | с | с | з | з | з |

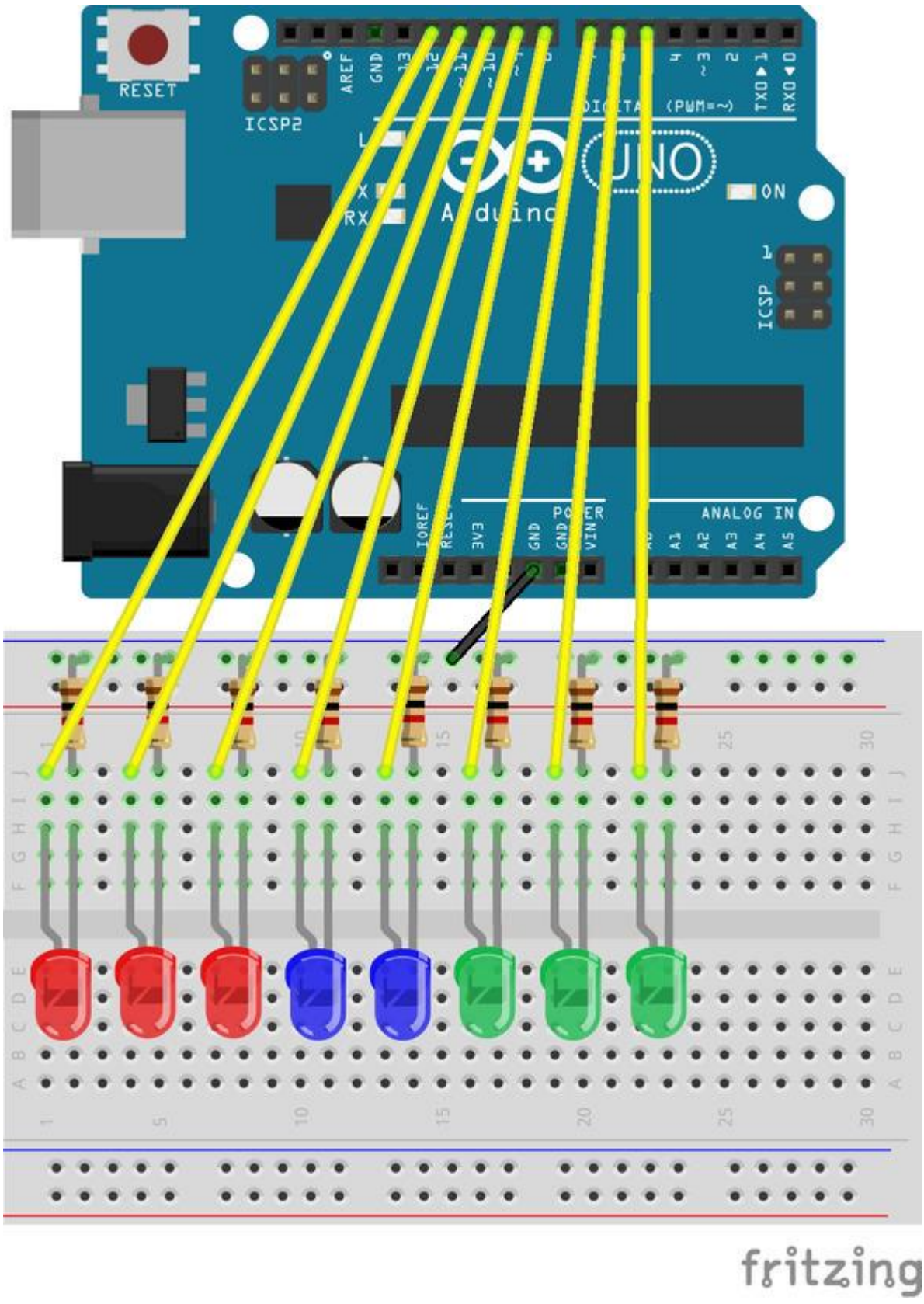
Коды цифр и гласных

Обратите внимание!
Два синих сигнала обозначают
пробел между словами,
а красный и два синих обозначают
вопросительный знак.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ж | к | к | | | | | | |
| б | к | к | | | | з | | |
| в | к | к | | | | з | з | |
| г | к | к | | | | з | з | з |
| й | к | к | | с | | | | |
| д | к | к | | с | | з | | |
| з | к | к | | с | | з | з | |
| ь | к | к | | с | | з | з | з |
| л | к | к | | с | с | | | |
| м | к | к | | с | с | з | | |
| н | к | к | | с | с | з | з | |
| р | к | к | | с | с | з | з | з |
| ш | к | к | к | | | | | |
| п | к | к | к | | | з | | |
| ф | к | к | к | | | з | з | |
| к | к | к | к | | | з | з | з |
| ! | к | к | к | с | | | | |
| т | к | к | к | с | | з | | |
| с | к | к | к | с | | з | з | |
| ъ | к | к | к | с | | з | з | з |
| х | к | к | к | с | с | | | |
| ц | к | к | к | с | с | з | | |
| ч | к | к | к | с | с | з | з | |
| щ | к | к | к | с | с | з | з | з |

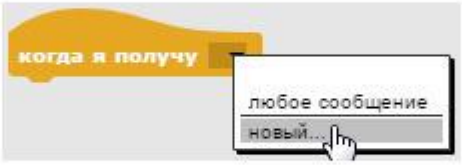
Коды согласных

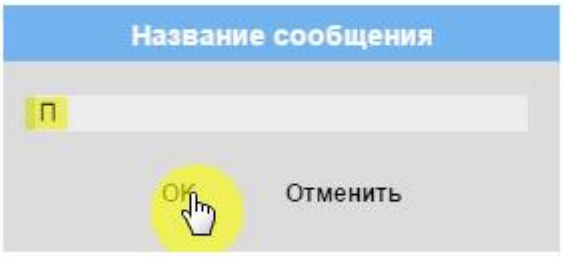
Соберите схему с восемью светодиодами. Установите на макетную плату 3 красных, 2 синих и 3 зеленых светодиода как показано на схеме. Красные светодиоды подключите к пинам 10, 11, 12, синие подключите к пинам 8 и 9, а зеленые к пинам 5, 6 и 7. Используйте резисторы номиналом 1кОм (если светодиоды светятся слишком ярко, то можно использовать резисторы номиналом 10кОм).



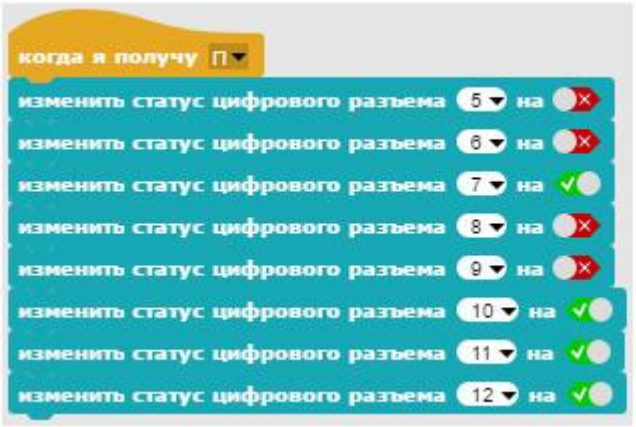

Закодируйте передачу сообщения «привет». Для этого создайте 6 сообщений: П, Р, И, В, Е и Т.

Для того чтобы создать новое сообщение раскройте выпадающий список в блоке ***когда я получу***, нажмите на **новый** и введите имя сообщения.

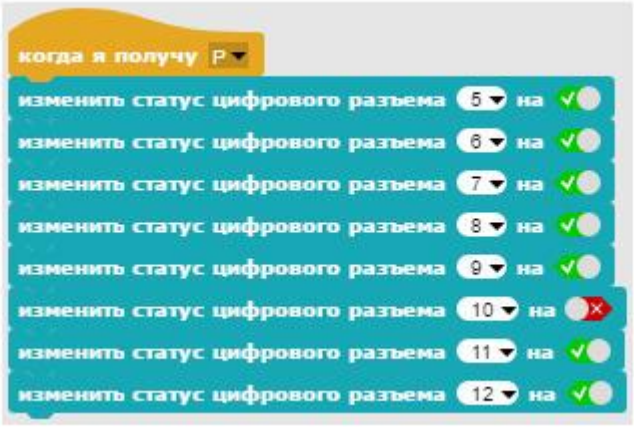


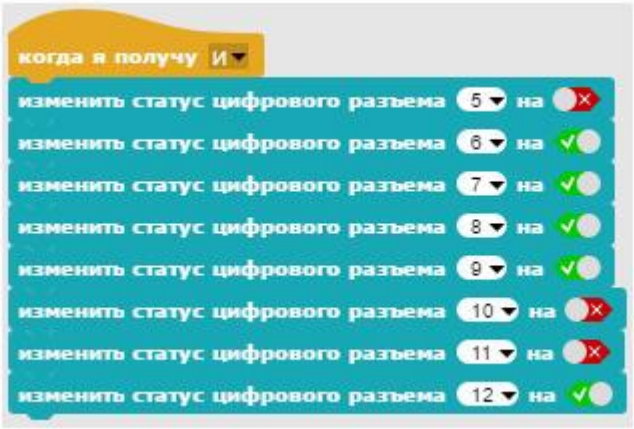

Для каждого сообщения соберите соответствующий скрипт.



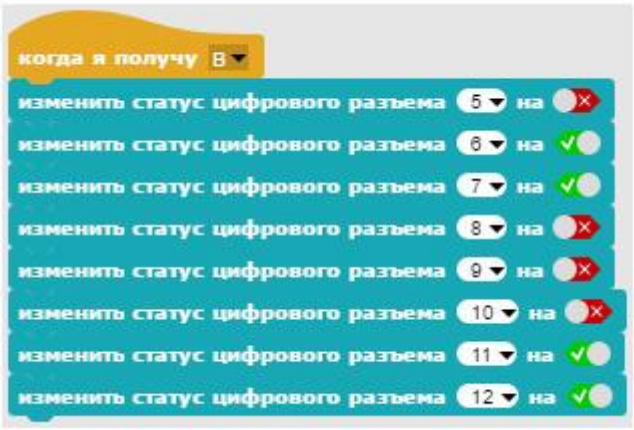

Скрипт передачи буквы П



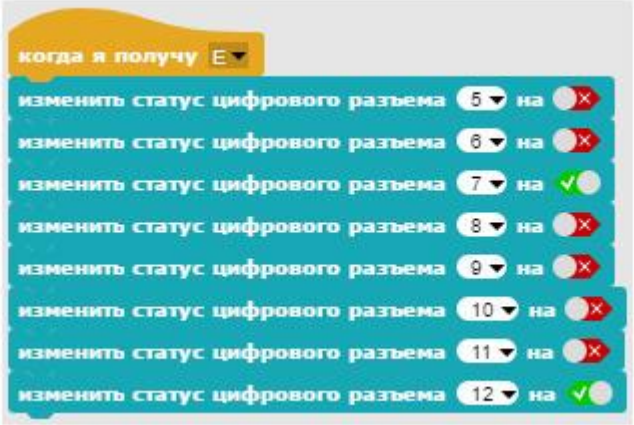

Скрипт передачи буквы Р



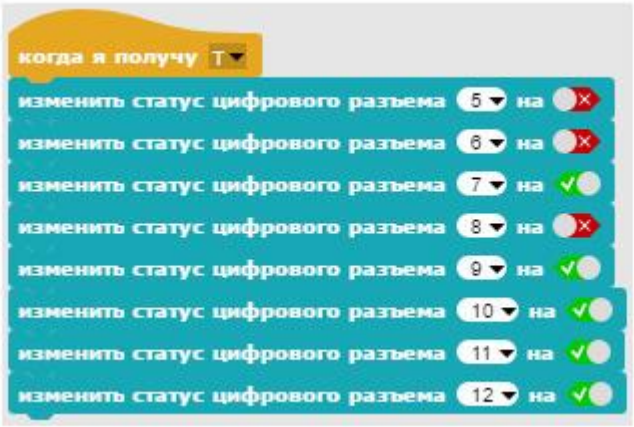

Скрипт передачи буквы И




Скрипт передачи буквы В

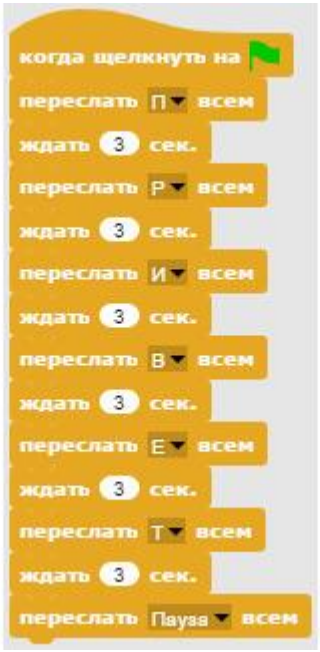



Скрипт передачи буквы Е




Скрипт передачи буквы Т

Соберите главный скрипт, который будет по очереди передавать необходимые сообщения.



Главный скрипт

Протестируйте работу проекта. Попробуйте прочитать переданное сообщение.

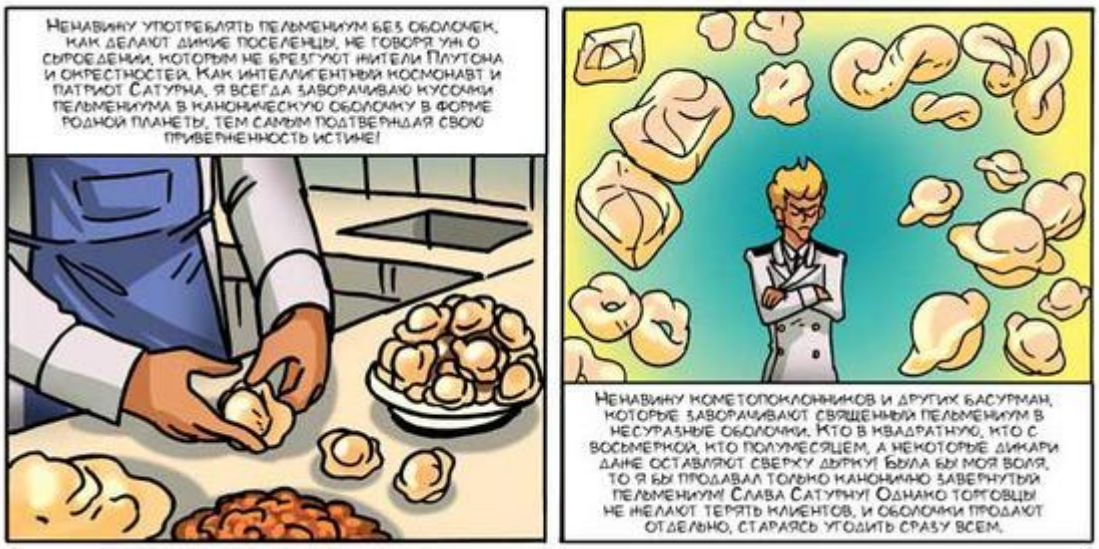
**Совет.**

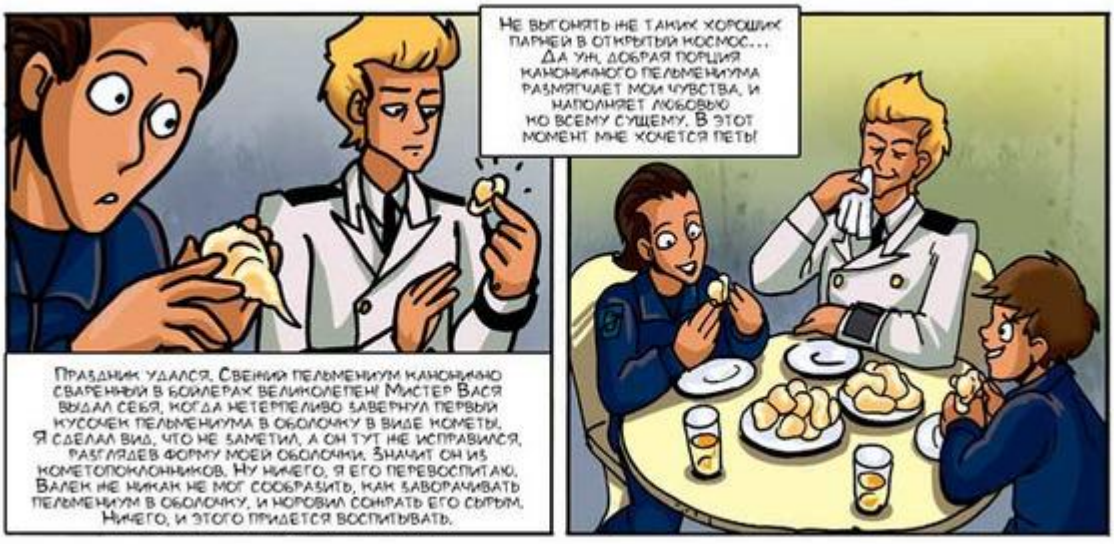
Для того чтобы принять длинное сообщение, переданное кодом уранских дальнобойщиков, не обязательно запоминать весь алфавит. Можно быстро записать сообщение с помощью трех цифр, обозначающих количество светодиодов, а затем спокойно расшифровать его. Вот как будет выглядеть слово «привет», записанное таким способом.

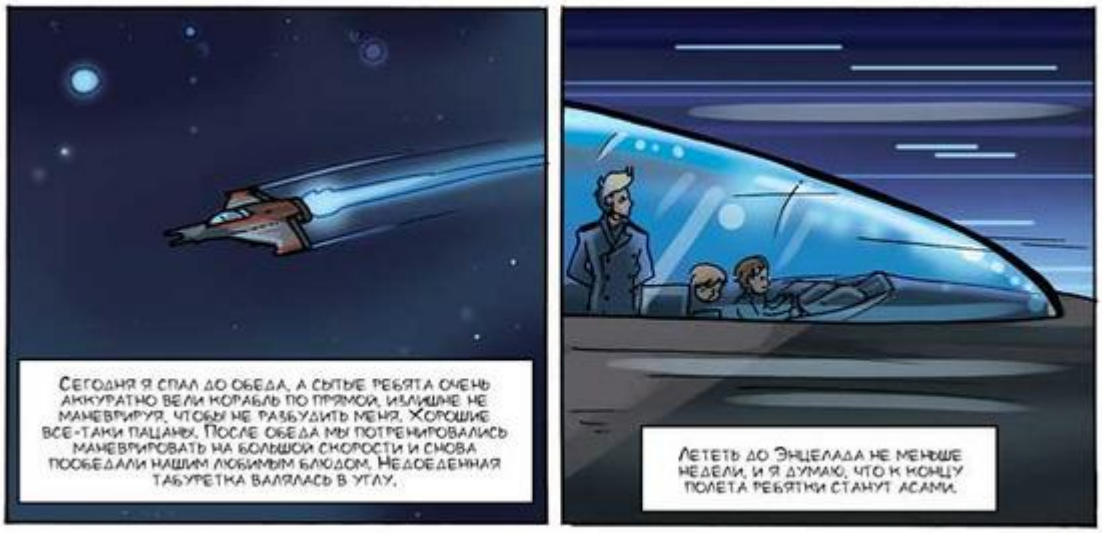
| | К | С | З |
|---|---|---|---|
| П | 3 | 0 | 1 |
| Р | 2 | 2 | 3 |
| И | 1 | 2 | 2 |
| В | 2 | 0 | 2 |
| Е | 1 | 0 | 1 |
| Т | 3 | 1 | 1 |

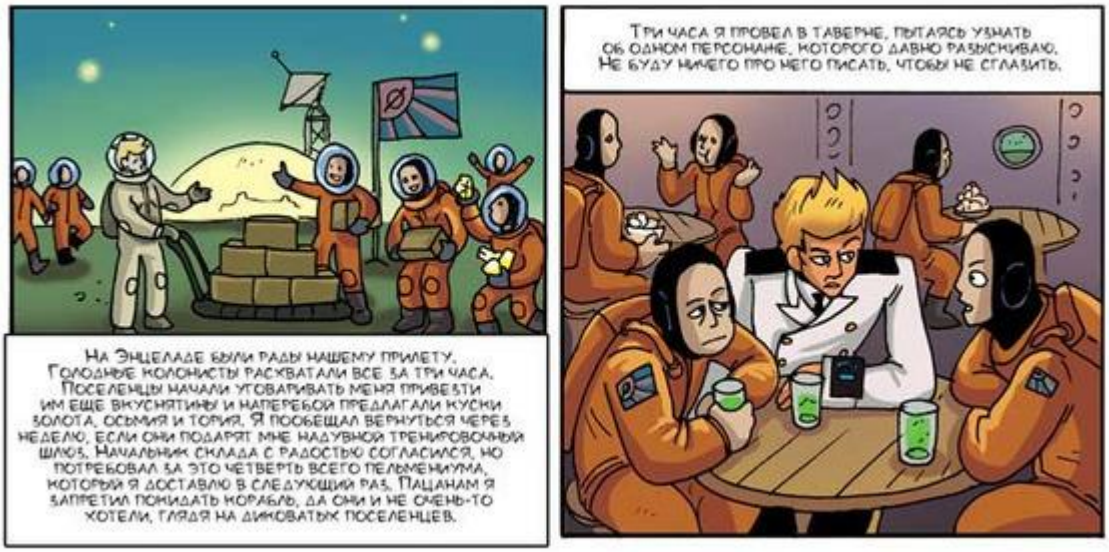

**Задание.**
Прочитайте сообщение.

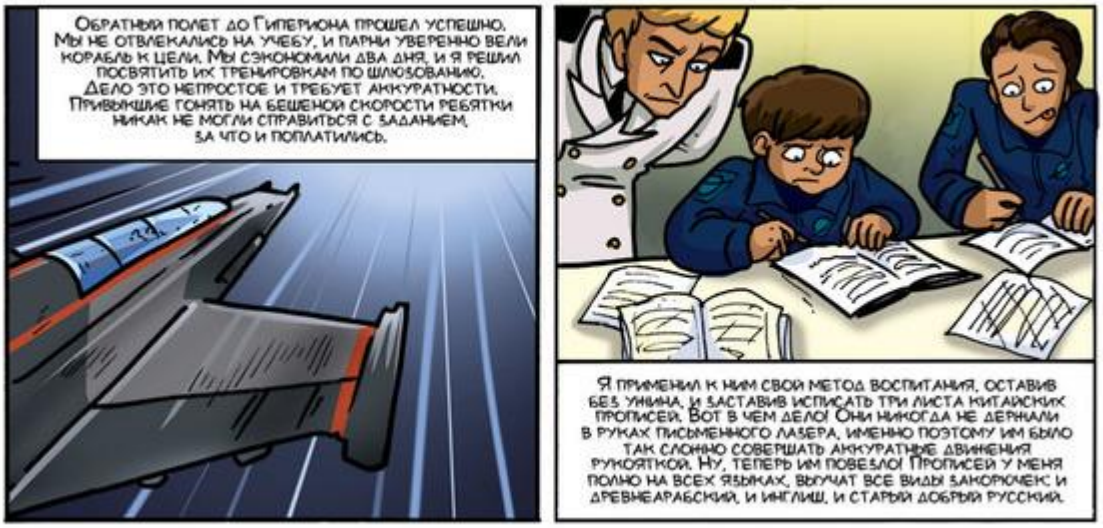
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| К | К | К | С | | З | | |
| К | | | С | С | З | З | З |
| | | | | | | | |
| К | | | С | С | З | | |
| К | К | К | С | | З | | |
| К | К | К | | | З | З | З |
| К | | | | | З | З | |
| К | К | | С | | З | | |
| К | | | | | З | З | З |
| К | | | С | С | | | |




Сегодня я спал до обеда, а сытые ребята очень аккуратно вели корабль по прямой, излишне не маневрируя, чтобы не разбудить меня. Хорошие все-таки пацаны. После обеда мы потренировались маневрировать на большой скорости и снова пообедали нашим любимым блюдом. Недоеденная табуретка валялась в углу.
Лететь до Энцелада не меньше недели, и я думаю, что к концу полета ребятки станут асами.


На Энцеладе были рады нашему прилету. Голодные колонисты расхватали все за три часа. Поселенцы начали уговаривать меня привезти им еще вкуснятины и наперебой предлагали куски золота, осьмия и тория. Я пообещал вернуться через неделю, если они подарят мне надувной тренировочный шлюз. Начальник склада с радостью согласился, но потребовал за это четверть всего пельменциума, который я доставлю в следующий раз. Пацанам я запретил покидать корабль, да они и не очень-то хотели, глядя на диковатых поселенцев.
Три часа я провел в таверне, пытаясь узнать об одном персонаже, которого давно разыскиваю. Не буду ничего про него писать, чтобы не сглазить.



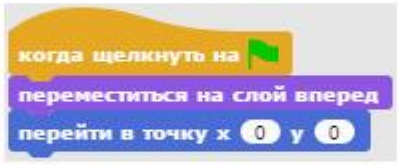

# Стыковка

## Подготовка спрайтов

Создайте новый проект. В нем будет всего два спрайта Шлюз и Прицел.
Сначала, как обычно, окрасьте сцену в темный цвет.
Затем импортируйте изображение прицела, он получит имя **Спрайт**.


Переименуйте его в **Прицел**.
Объявите переменные ***A0*** и ***A1***.
Выберите Прицел и перейдите на вкладку **Скрипты**.


Создайте Прицелу вот такой скрипт.



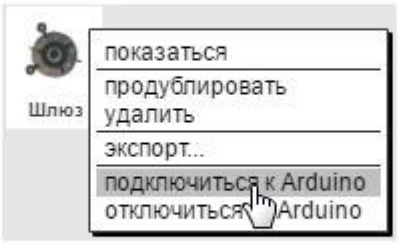

Скрипт Прицела

Прицел будет расположен постоянно в середине сцены.
Теперь создайте новый спрайт с помощью кнопки в виде стрелочки.

Выберите новый спрайт и импортируйте изображение тренировочного шлюза, он получит имя **Спрайт (2)**. Переименуйте его в **Шлюз.**


Теперь сцена выглядит вот так.

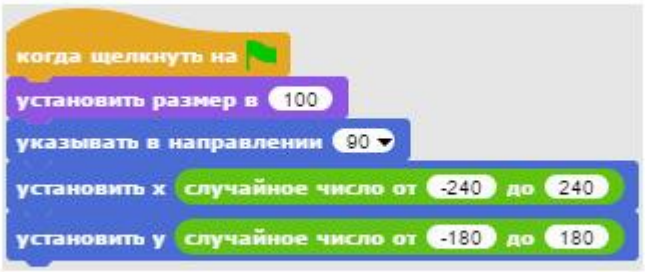
Подключите к спрайту шлюза плату Arduino. Кликните на нем правой кнопкой мышки и выберите **подключиться к Arduino**.




## Программирование

Запрограммируйте шлюз, у него будет пять скриптов.

Первый скрипт устанавливает шлюз в исходное положение.




Первый скрипт шлюза

Он задает размер шлюза равный 100%, устанавливает шлюз в вертикальное положение, и перемещает в случайно выбранную точку сцены.

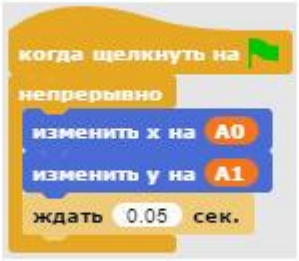
Второй скрипт реализует управление с помощью джойстика.

Второй скрипт шлюза

В переменные ***A0*** и ***A1*** сохраняются значения, полученные путем вычитания из пяти аналоговых значений пинов разделенных на сто. Таким образом, в зависимости от наклона ручки джойстика, значения переменных ***A0*** и ***A1*** будут изменяться в диапазоне от -5 до 5.

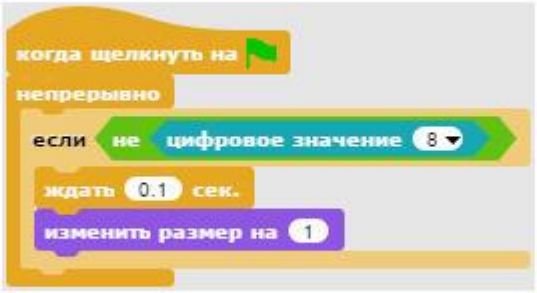
Третий скрипт перемещает шлюз в зависимости от значения переменных ***A0*** и ***A1***. Если ручка джойстика в центральном положении, то значения переменных равны нулю, и перемещения не происходит.

Блок ***ждать*** необходим для замедления работы проекта на быстрых компьютерах. Если ваш компьютер недостаточно быстрый, то можете удалить его.




Третий скрипт шлюза

Четвертый скрипт необходим для управления шлюзовым двигателем. Он еще слабее маневрового, и управляется с помощью нажатия на ручку джойстика.



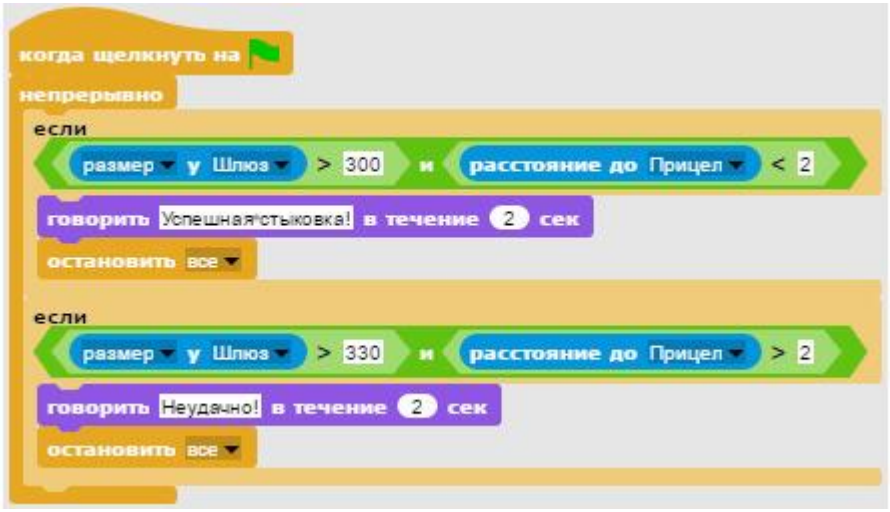

Четвертый скрипт шлюза

Кнопка, которая установлена под ручкой джойстика, соединена с пином 8 на плате Arduino, поэтому считываем именно это цифровое значение. Как вы знаете, в нормальном режиме, когда

кнопка не нажата, она возвращает значение **истина**. Нам же необходимо сделать наоборот, чтобы двигатель включался не при отпущенной, а при нажатой кнопке, поэтому в блоке ***если*** используем логический оператор НЕ, который изменяет **истина** на **ложь**и наоборот.

При работе двигателя шлюз будет приближаться к нам, и увеличиваться в размере.

Пятый скрипт будет определять, удачной ли была стыковка.




Пятый скрипт шлюза

В нем использованы голубые сенсорные блоки, которые из-за их овальной формы иногда называют встроенными переменными. Сенсор ***размер у Шлюз*** содержит число, равное размеру спрайта шлюза, а сенсор ***расстояние до Прицел*** содержит число, равное расстоянию между центрами спрайтов.

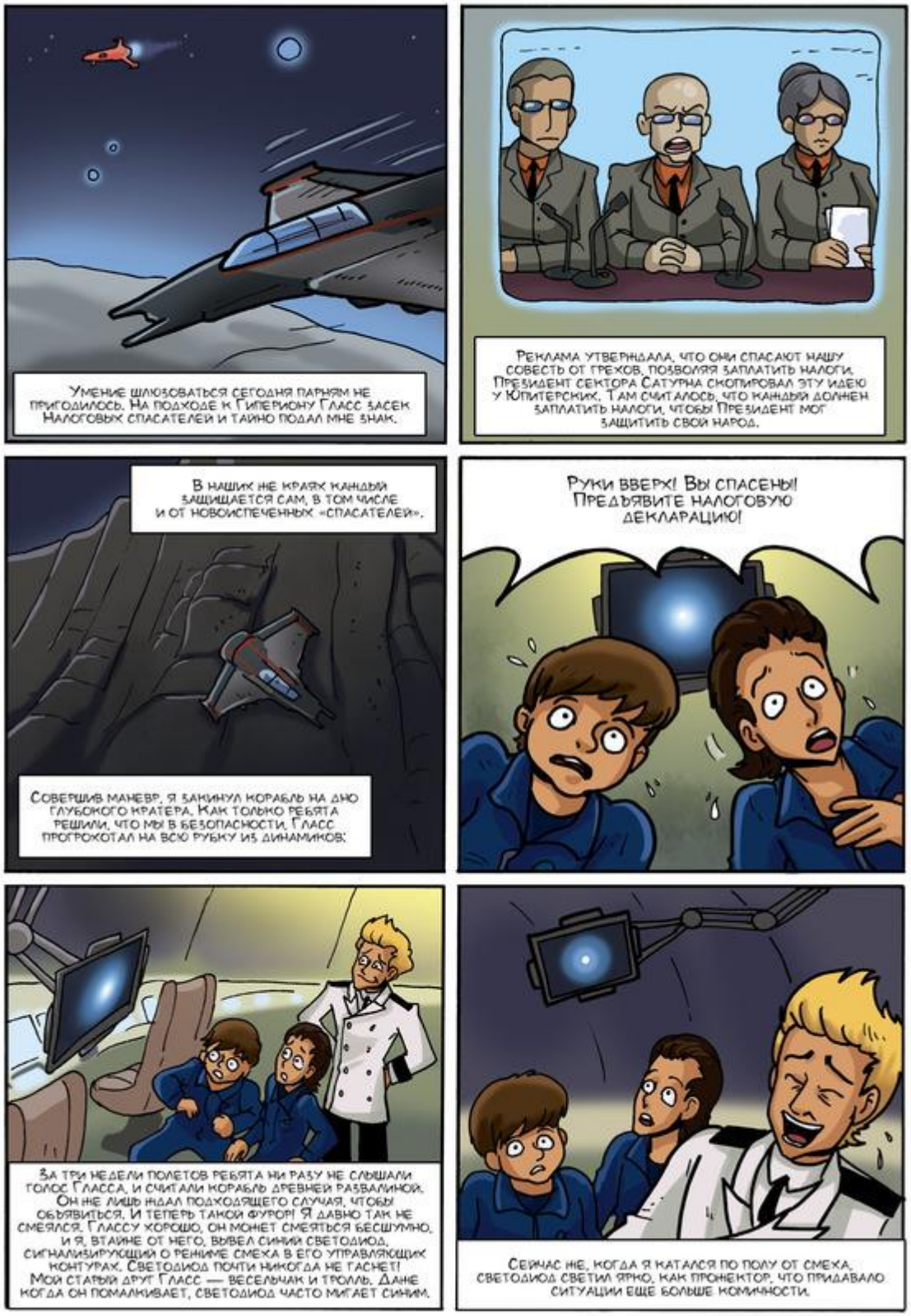
Первый блок ***если*** проверяет одновременное выполнение двух условий, размер шлюза должен быть более 300 (это означает, что вы приблизились к нему на нужное расстояние), и расстояние между центрами спрайтов должно быть менее 2 (это означает, что вы точно соединились со шлюзом, и не погнете корпус корабля). В случае одновременного выполнения этих условий шлюз сообщит об удачной стыковке, и работа программы будет завершена.

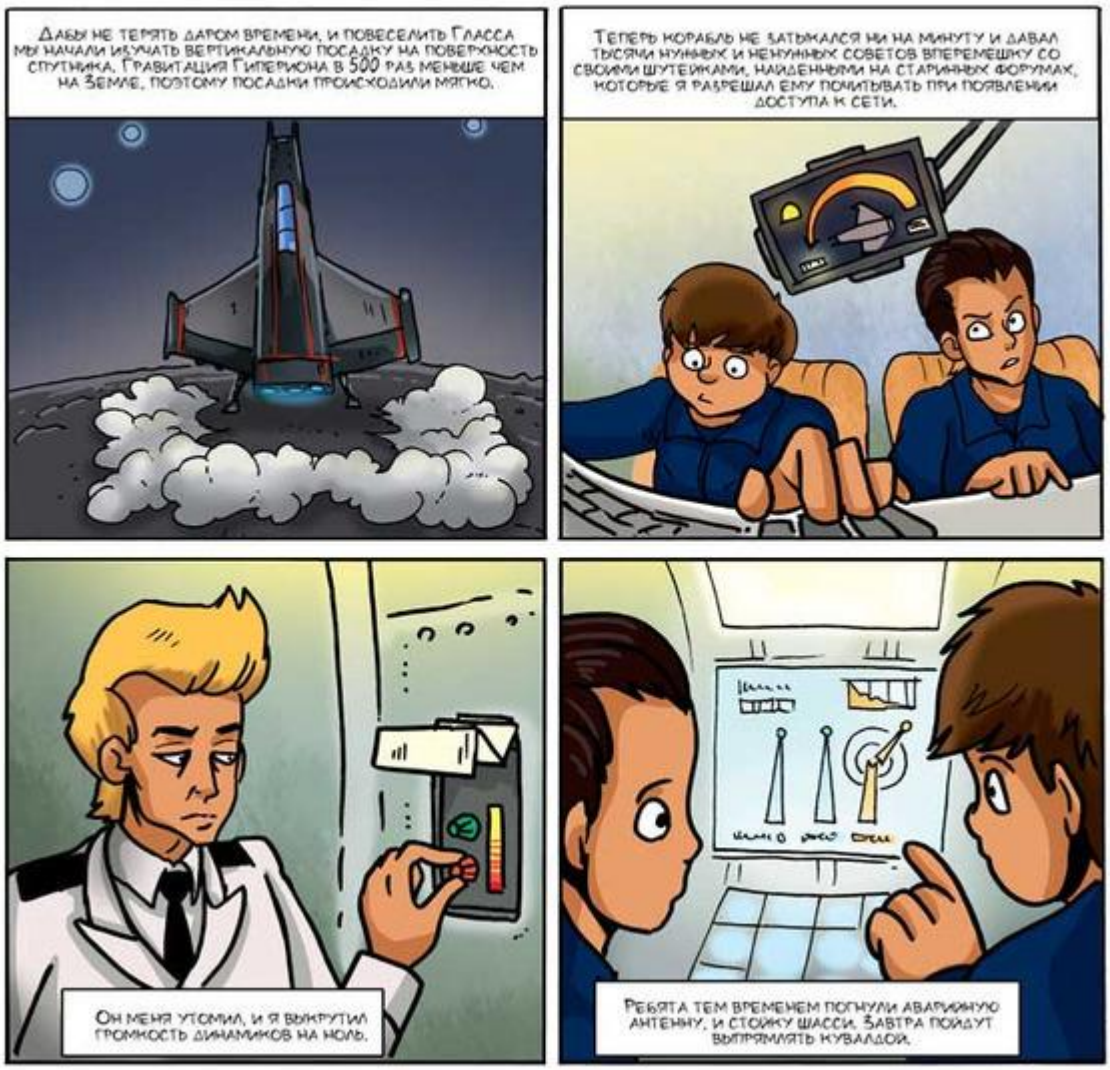
Второй блок ***если*** проверяет одновременное выполнение двух условий, размер шлюза должен быть более 330 (это означает, что вы влепились в шлюз), и расстояние между центрами спрайтов должно быть более 2 (это означает, что вы поцарапали корпус корабля). В случае одновременного выполнения этих условий шлюз сообщит о неудачной стыковке, работа программы будет завершена, а вы должны выполнить 100 приседаний.

Сохраните проект и запустите его, нажав на зеленый флажок. Будьте очень аккуратны! Берегите корабль!


*Мой метод работает! Я становлюсь опытным воспитателем. Скоро смогу работать в школе, или в тюрьме, что в наших местах почти одно и то же. Не уверен, что лучше помогло: учебное голодание, прописи, или те 2000 приседаний, но сегодня ребятки смогли аккуратно соединить корабль со шлюзовой камерой!*

*Заметил, что ножка табуретки стала немного короче. Значит, парнишки нашли способ, как обвести меня вокруг пальца. Как же они ее ели то, без воды? Бедняги. Ну, ничего, назначил им еще 3000 приседаний, и накормил по-человечески.*






# Посадка на астероид

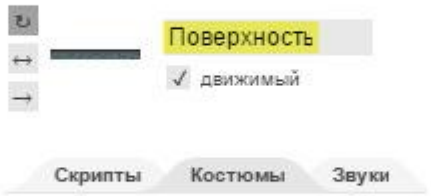
## Подготовка спрайтов

Создайте новый проект и подготовьте его к работе. Импортируйте спрайт космического корабля, окрасьте сцену в темный цвет и подключите плату Arduino как описано в приложении 1.

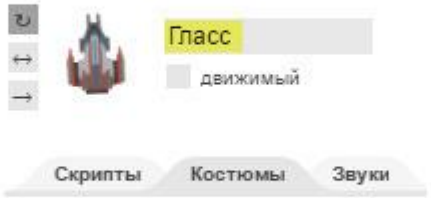
Создайте новый спрайт, нажав на кнопку добавления спрайтов (**add a new Turtle sprite)**.

Импортируйте изображение поверхности из файла fon1.png. Если поверхность размещена не горизонтально, то дважды кликните на блок ***указывать в направлении 90***, и она ляжет ровно.

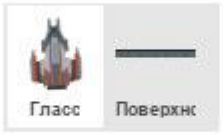

Поместите спрайт поверхности в нижней части сцены и переименуйте его в **Поверхность**.




Спрайту космического корабля дайте имя **Гласс**.



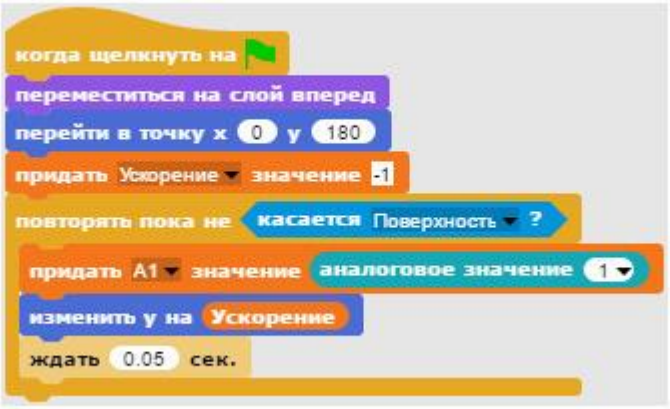
Область спрайтов должна выглядеть вот так.




Объявите переменные ***Ускорение*** и ***A1*** и сделайте их видимыми, установив галочки.
Сцена должна выглядеть вот так.

## Программирование

Теперь запрограммируем корабль, у него будет четыре скрипта. Первый скрипт отвечает за движение корабля.



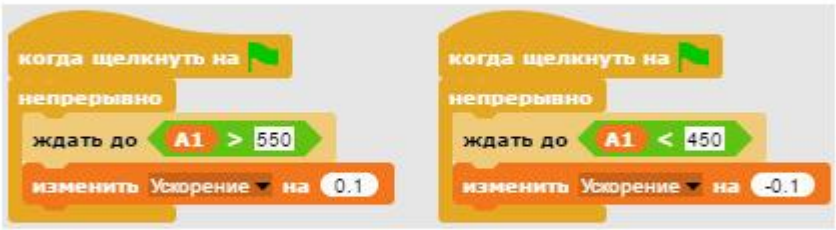

Первый скрипт корабля

При запуске программы корабль перейдет на слой вперед, чтобы не прятаться за спрайт поверхности, затем перейдет в верхнюю часть сцены и задаст начальное значение переменной ***Ускорение***, при котором корабль будет плавно опускаться на поверхность.

Затем вместо блока ***непрерывно*** мы используем блок ***повторять пока не***, который прекратит свое выполнение при касании кораблем поверхности. Внутри блока циклически записываем в переменную ***A1*** аналоговое значение с пина A1, а также изменяем вертикальное положение корабля в соответствии со значением переменной ***Ускорение***.

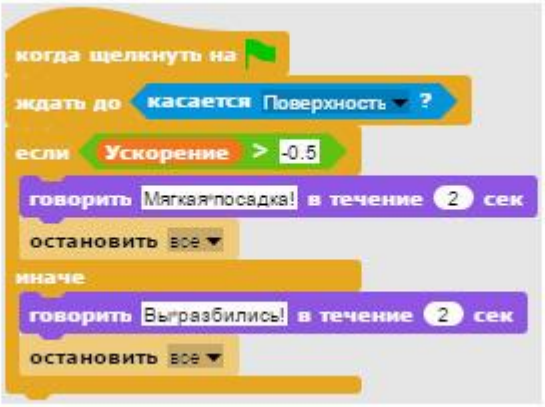
Блок ***ждать*** необходим для замедления работы проекта на быстрых компьютерах. Если ваш компьютер недостаточно быстрый, то можете удалить его.

Следующие два скрипта изменяют значение переменной ***Ускорение*** в соответствии со значением переменной ***A1***. Если ***A1*** больше 550 (ручка джойстика вперед), то скорость падения корабля замедлится, и наоборот, если ***A1*** меньше 450 (ручка джойстика назад), то падение корабля ускорится.




Второй и третий скрипты корабля

Четвертый скрипт определяет «мягкость» посадки.



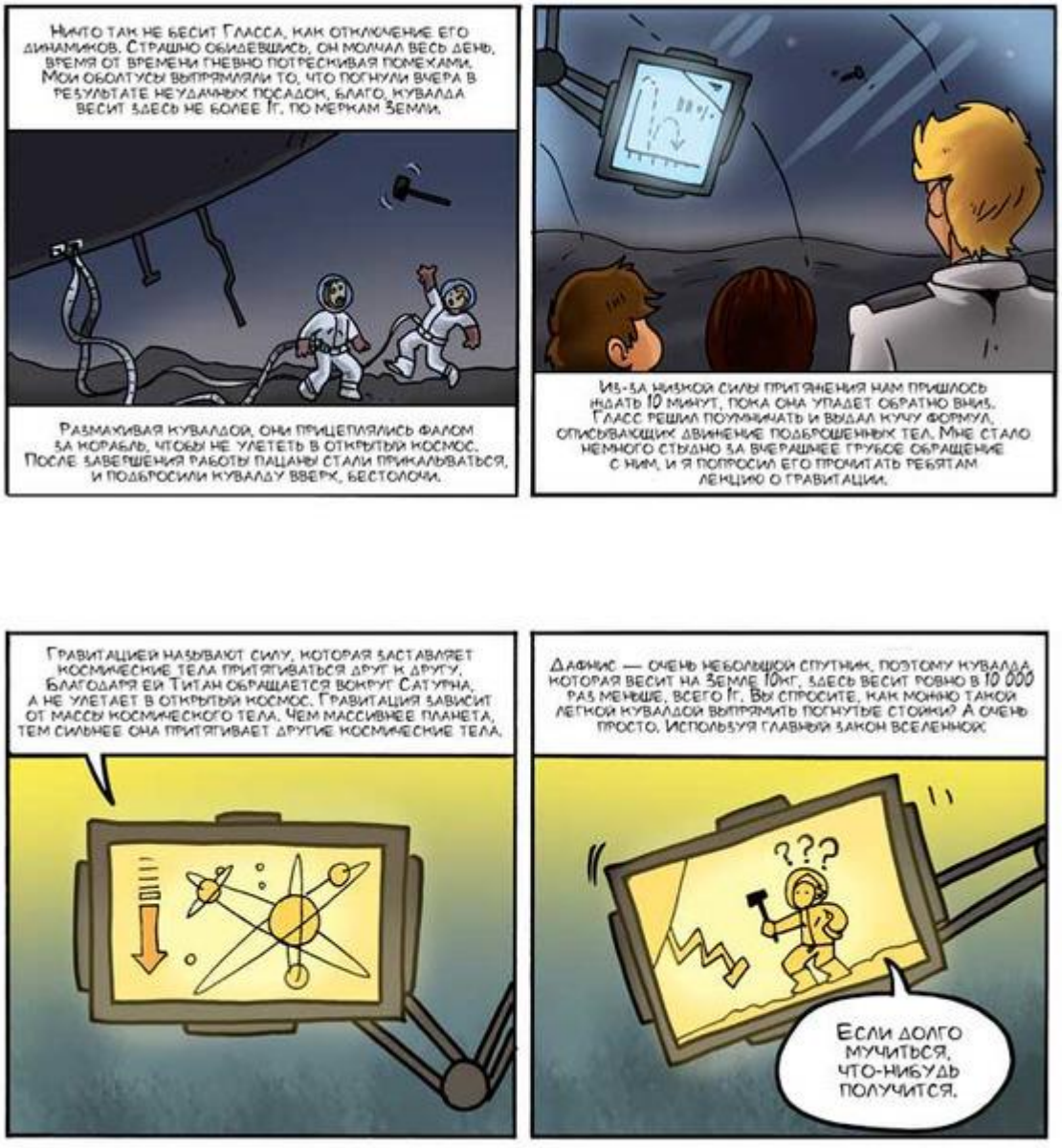

Четвертый скрипт корабля

Он начинает выполняться только тогда, когда корабль коснется поверхности. Блок ***если иначе*** проверяет, было ли значение переменной ***Ускорение*** меньше -0.5, и если да, то посадка будет мягкой, а если нет, то корабль разобьется.

Сохраните проект и запустите его, нажав на зеленый флажок. Будьте осторожны, не поцарапайте корабль!

**Задания.**

1. Увеличьте скорость посадки корабля.
2. Усложните в два раза условие мягкой посадки.
3. Нарисуйте второй костюм корабля в виде взрыва.
4. Измените последний скрипт с использованием блока следующий костюм, чтобы в момент неудачной посадки корабль взрывался.




# Сигнализация

*Отсиживаемся на Гиперионе. Я отключил освещение, чтобы налоговым спасателям было сложнее заметить наш корабль. Дежурим по очереди, но на пацанов я особо не рассчитываю, могут заснуть и прозевать незваных гостей. На всякий случай я собрал схему, реагирующую на свет от прожектора. Если на нас будет направлен луч яркого света, то прозвучит сигнал тревоги, и мы успеем удрать, поставив дымовую завесу.*

## Знакомство с фоторезистором

Датчик освещенности – это фоторезистор, электронный компонент, который, как и потенциометр, изменяет свое сопротивление, только не из-за вращения ручки, а в зависимости от яркости света, падающего на него.

Фоторезистор

Чем ярче свет, тем меньше сопротивление фоторезистора.

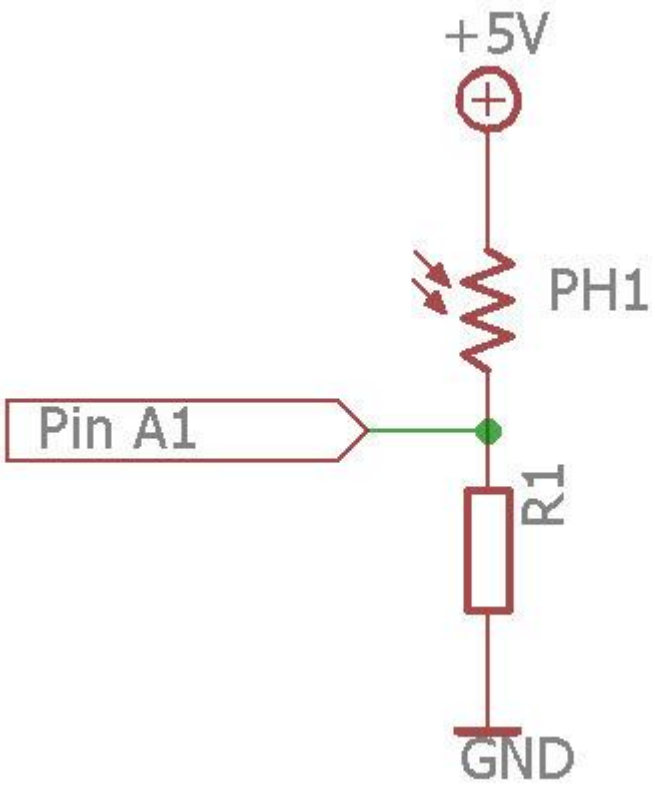
Подключите фоторезистор к мультиметру с помощью «крокодилов» и измерьте его сопротивление.

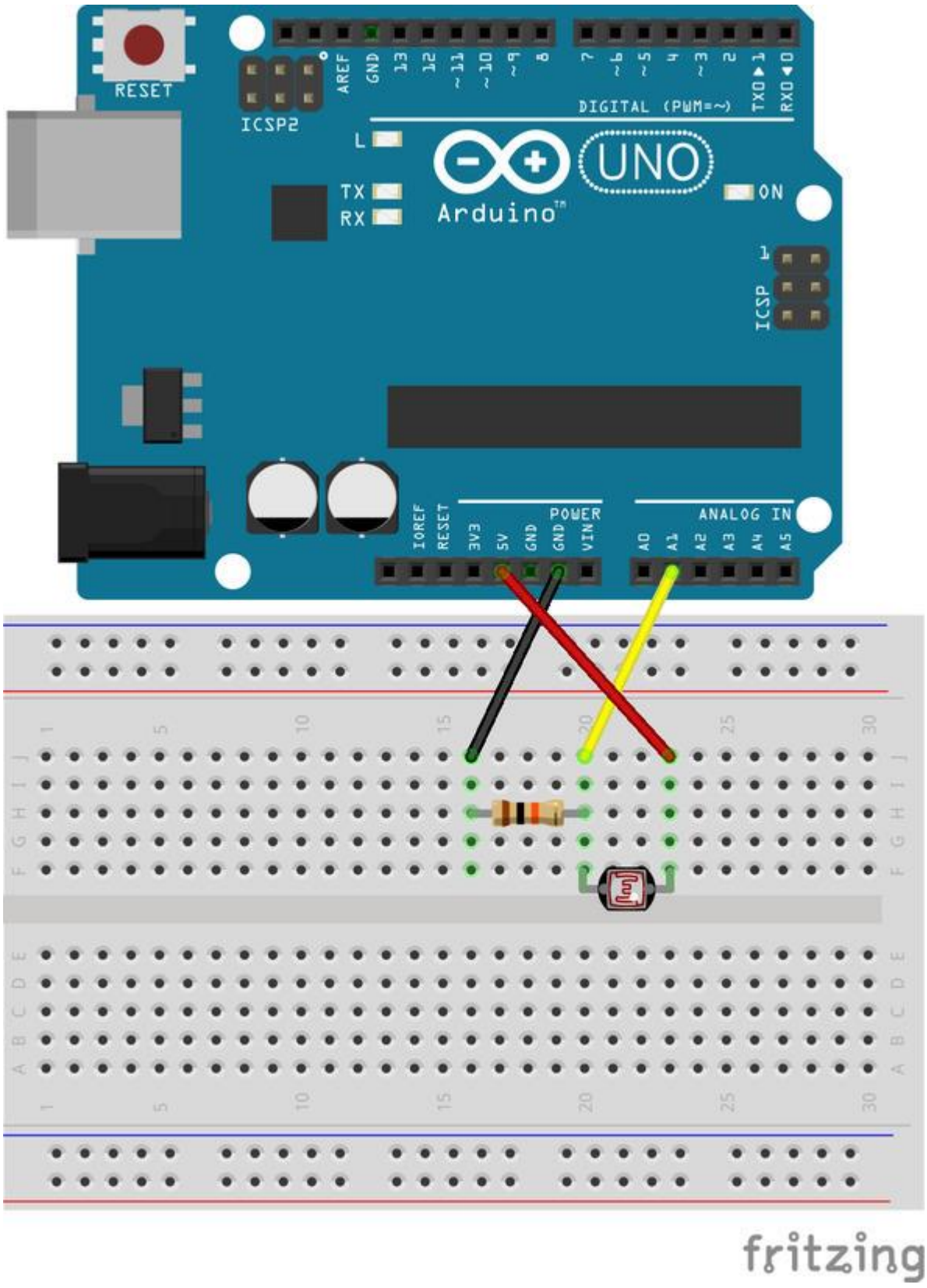

Установите переключатель мультиметра как показано на рисунке.

Направьте на фоторезистор свет от фонарика, и посмотрите, как будет изменяться его сопротивление. Изменяйте световой поток, приближая и удаляя фонарик от фоторезистора, понаблюдайте за изменением его сопротивления.

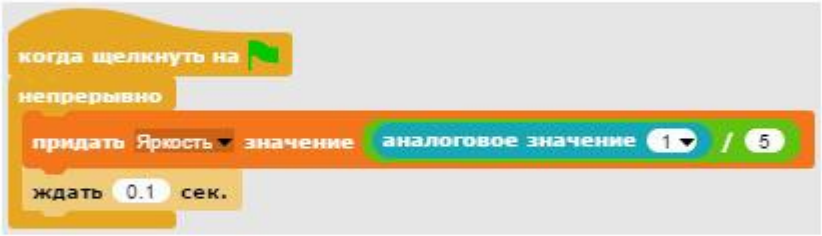
Сопротивление фоторезистора должно изменяться в диапазоне от 2 до 20 кОм (в полной темноте менее 200 кОм).

Соберите вот такую схему для тестирования работы фоторезистора. Подайте питание на одну ножку фоторезистора, вторую соедините с резистором 10 кОм. Вторую ножку резистора соедините с землей. Точку соединения резистора и фоторезистора соедините в пином A1 на плате Arduino.





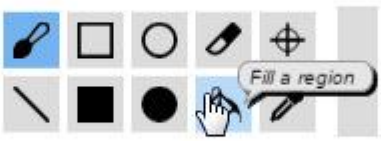

Подключите плату Arduino к компьютеру и соберите вот такой скрипт для спрайта стрелки, предварительно создав переменную ***Яркость***.



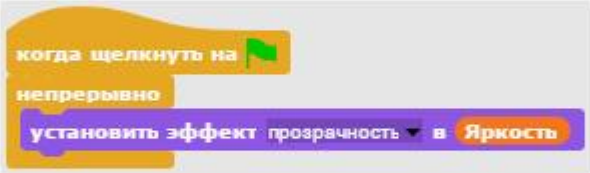

Скрипт стрелки

Выберите сцену и создайте новый фон нажав на кнопку **Нарисовать новый костюм (Paint a new costume)**.


Выберите черный цвет и закрасьте весь фон с помощью инструмента **Заливка (Fill a region)**.



Перейдите на вкладку **Скрипты** и соберите вот такой скрипт для сцены.




Скрипт сцены

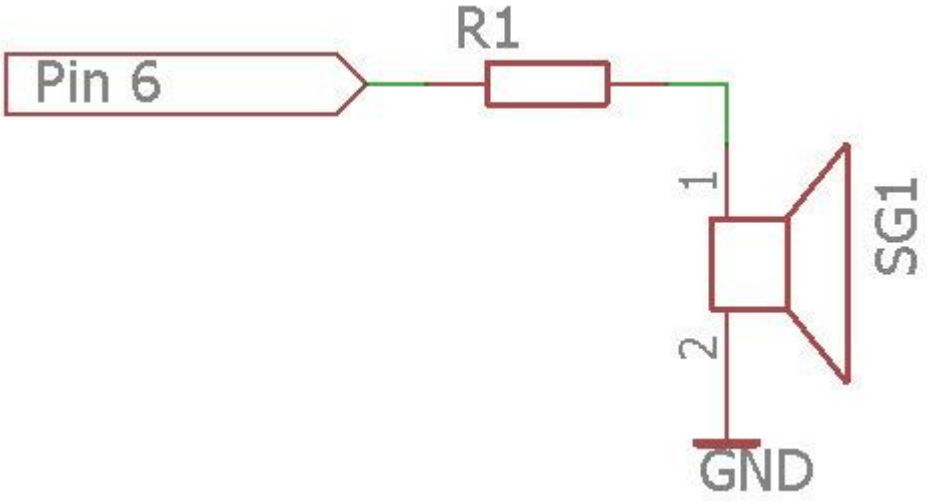
Запустите проект, и закрывайте фоторезистор руками от яркого света. Чем сильнее вы его закроете, тем темнее станет сцена. Если закрывать фоторезистор медленно, то это будет похоже на заход Солнца и наступление сумерек.

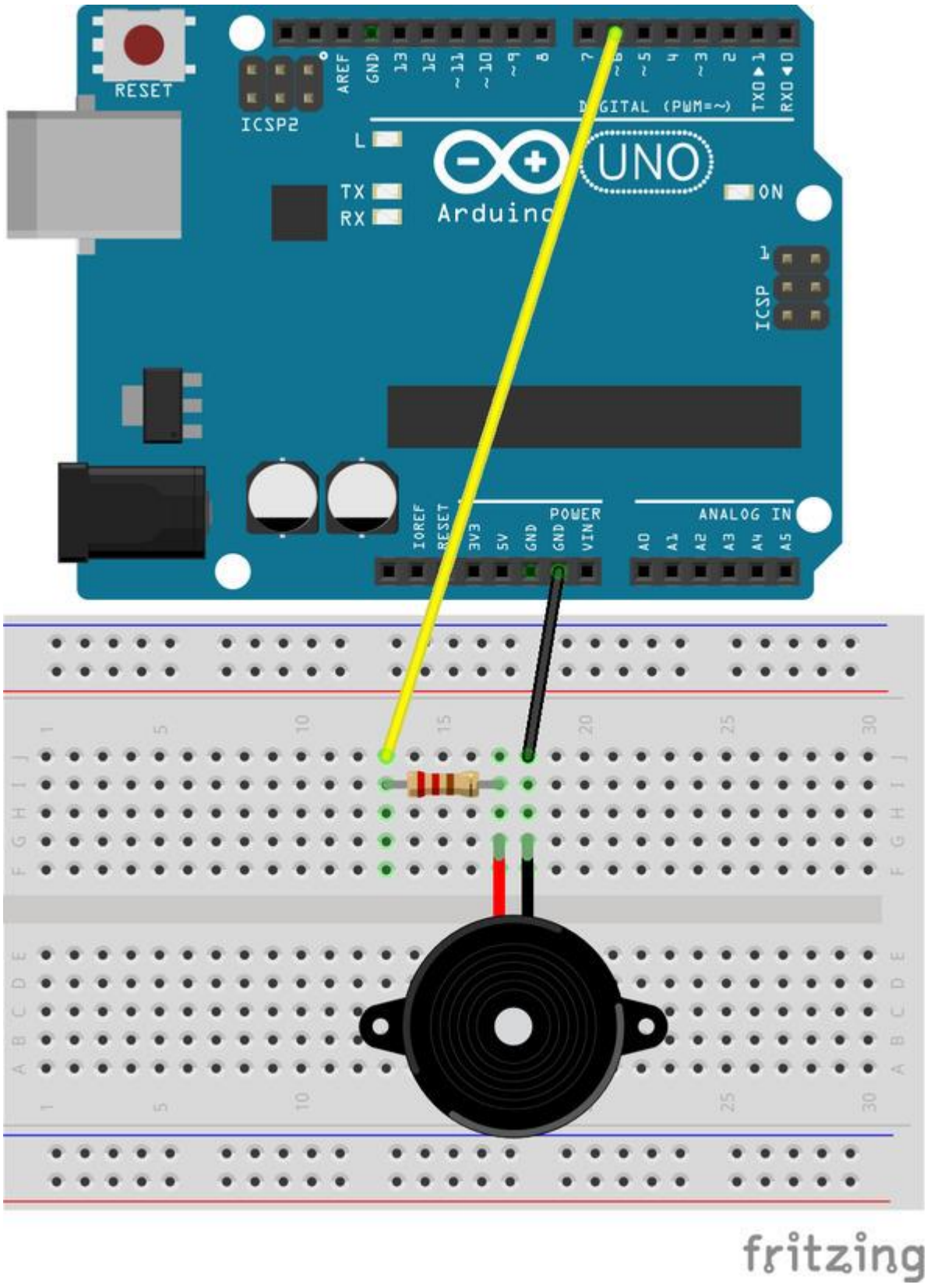
## Знакомство с зуммером

Зуммер – это электронный компонент, который издает звук при подаче на него переменного напряжения высокой частоты. Звук издается при помощи тонкой мембраны, установленной внутри зуммера, которая колеблется с частотой поданного напряжения.

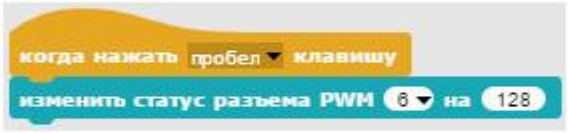
Частота измеряется в Герцах. 1 Герц – это одно колебание в секунду. 1000 Герц называются килогерц – это 1000 колебаний в секунду.

Соберите вот такую схему для тестирования работы зуммера. Пин 6 соедините с резистором в 330 Ом, резистор с одной ножкой зуммера, а вторую ножку зуммера с землей.



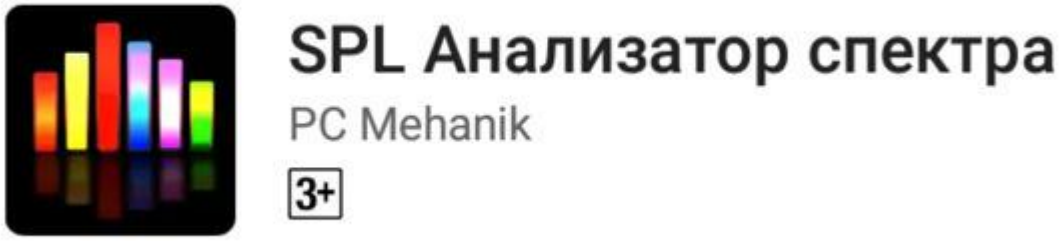



Подключите плату Arduino к компьютеру и соберите вот такой скрипт.

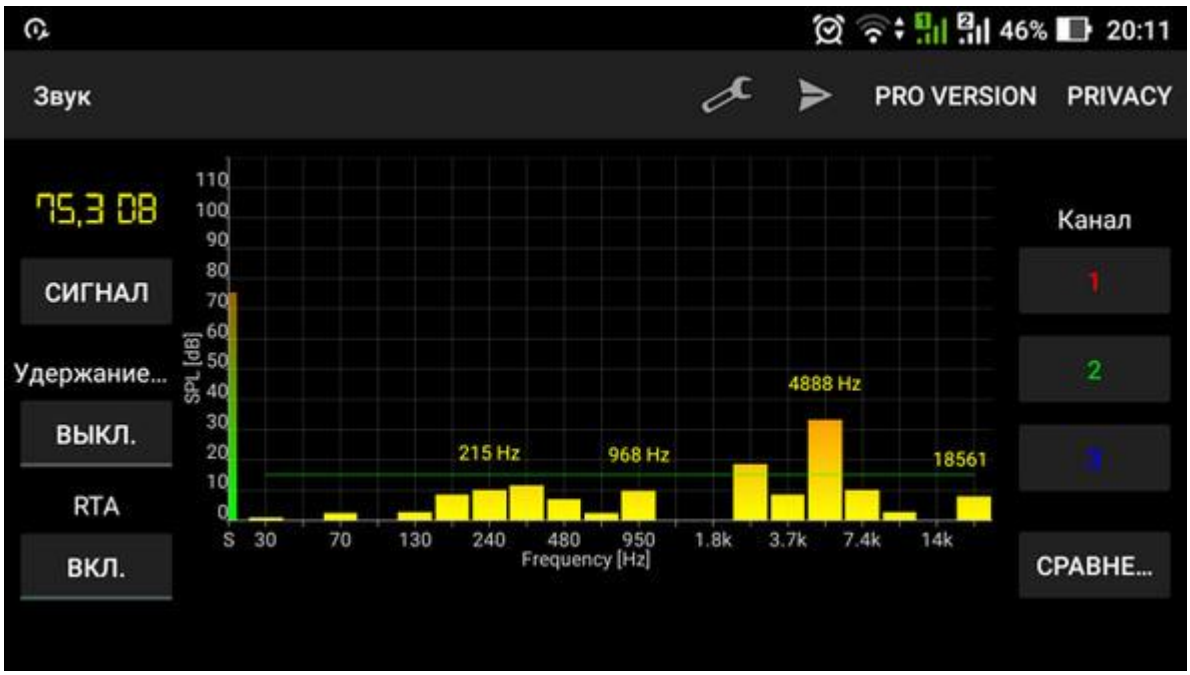



Нажмите на пробел – зуммер дико запищит. Изменяйте значение 128 в диапазоне от 0 до 255 и послушайте, как будет изменяться звук.

Для того чтобы определить частоту издаваемого звука можно использовать любой смартфон. Установите любое приложение измеряющее частоту звука, например «SPL анализатор спектра».




Запустите приложение и обратите внимание на самый высокий столбик. Мембрана зуммера сейчас колеблется с частотой 4888 Гц, и вы слышите звук именно такой частоты.

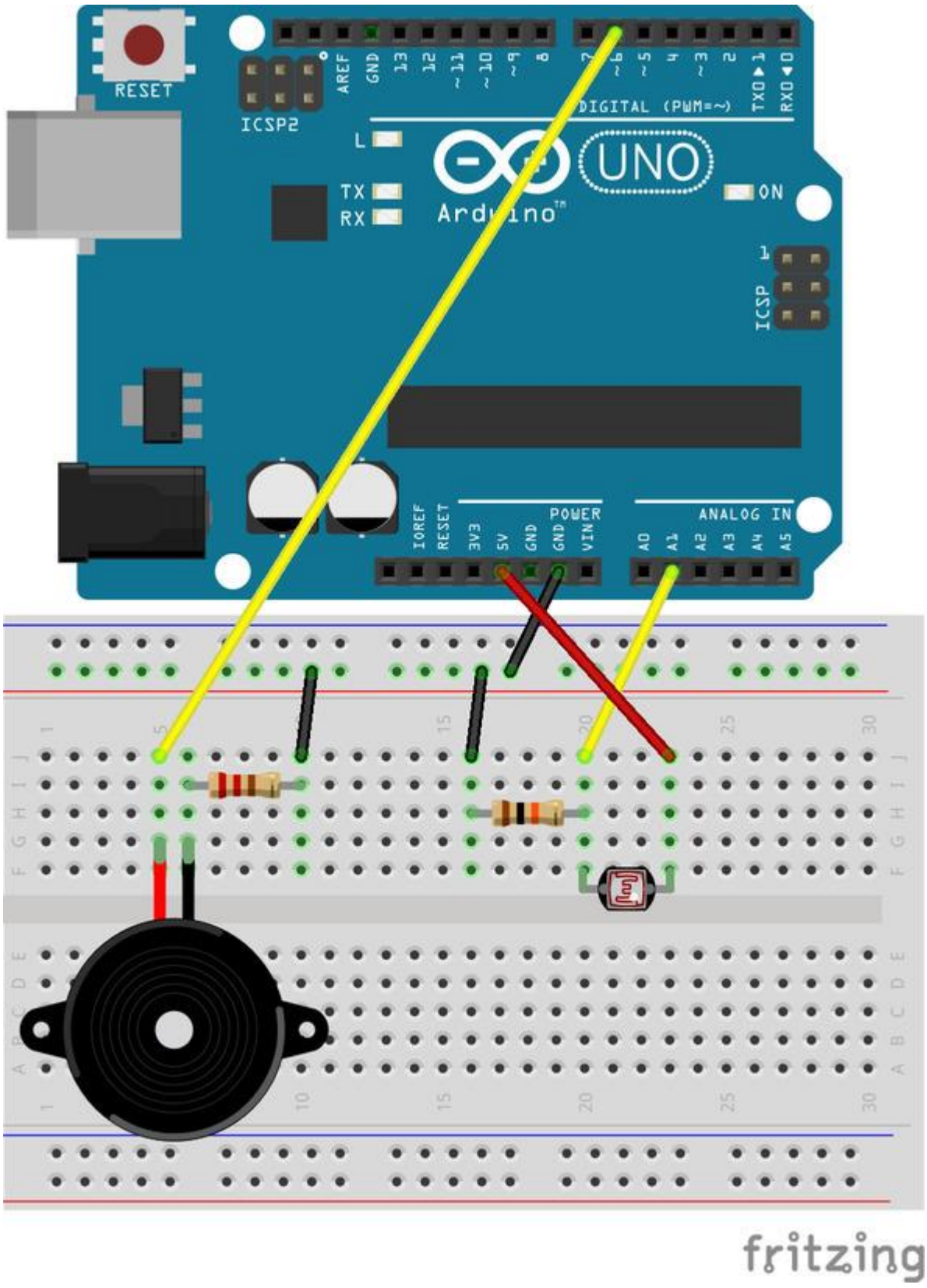



Произнесите несколько слов, и вы увидите, с какой частотой колеблются ваши голосовые связки.

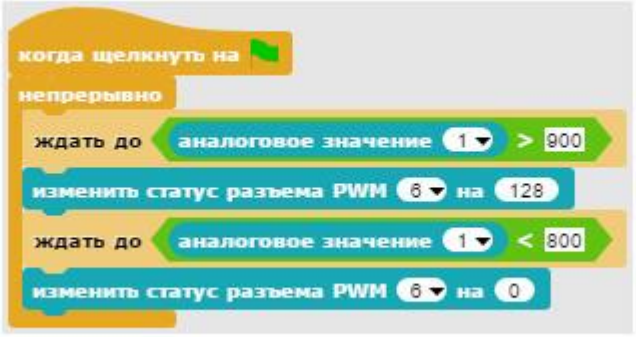
Задания.
1. Если у вас есть гитара или пианино, то заставьте звучать одну ноту и измерьте частоту звука.
2. Измерьте частоту нот соседних октав и вычислите, во сколько раз они отличаются.

## Сигнализация с зуммером срабатывающая от яркого света

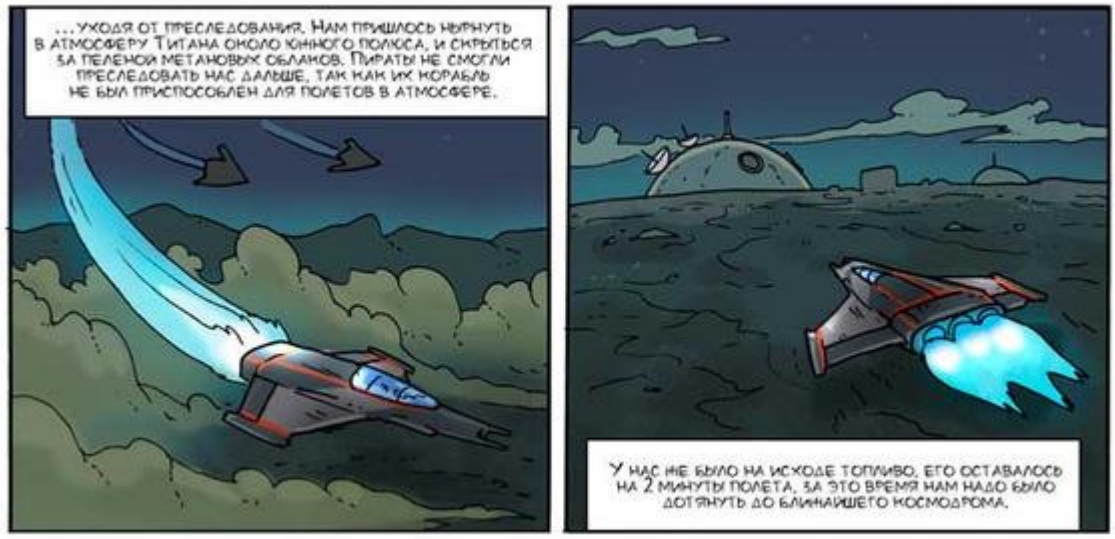
Соберите следующую схему. Сигал от фоторезистора подайте на аналоговый вход A1. На зуммер подайте сигнал от пина 6. К фоторезистору подключите резистор номиналом 10 кОм, а к зуммеру номиналом 330 Ом.




Подключите плату Arduino к компьютеру и соберите вот такой скрипт.




После запуска проекта скрипт будет постоянно ожидать момента, когда свет станет ярким и аналоговое значение, считанное с пина A1, станет больше 900. При этом статус пина 6 станет равным 128, и зуммер запищит. Пищать он будет до тех пор, пока аналоговое значение, считанное с пина A1, не станет меньше 800.



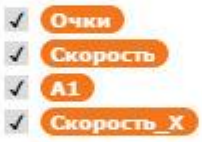

# Полет в атмосфере

## Подготовка спрайтов

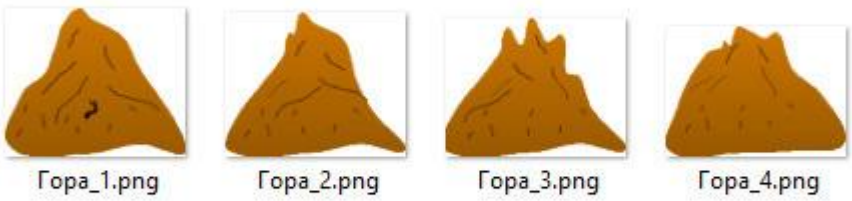
Создайте новый проект и подготовьте его к работе. Импортируйте спрайт космического корабля glass2_2.png, окрасьте сцену в светло-желтый цвет и подключите плату Arduino как описано в приложении 1.

Создайте три новых спрайта, нажав на кнопку добавления спрайтов **add a new Turtle sprite**.

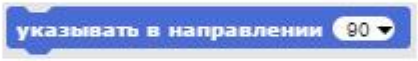

Объявите переменные ***Очки***, ***Скорость***, ***A1***, ***Скорость_X***.




Выберите второй спрайт и импортируйте четыре изображения горы (у нее появятся четыре костюма).

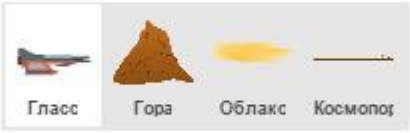


Если спрайт расположился не ровно, то выровняйте его дважды кликнув по блоку ***указывать в направлении 90***.




Выберите третий спрайт и импортируйте изображения облаков (у них тоже появятся четыре костюма).


Затем выберите четвертый спрайт и импортируйте изображение космопорта Космодром.png.
Выровняйте все спрайты и переименуйте их.

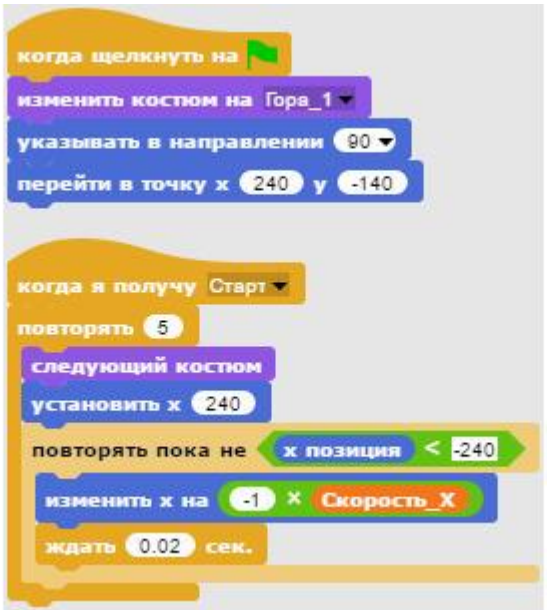



Сцена должна выглядеть вот так, видны переменные ***Очки***, и ***Скорость*.**


## Программирование

Этот проект будет похож на игру Flappy Bird. Управлять полетом корабля будем, поднимая ручку джойстика вверх. Начало полета – нажатие на ручку джойстика, при этом будет передано сообщение ***Старт***. Сообщение видно сразу всем спрайтам, и они одновременно начнут выполнять скрипты, начинающиеся с блока ***когда я получу Старт***.

Запрограммируйте гору. У нее будет всего два скрипта.



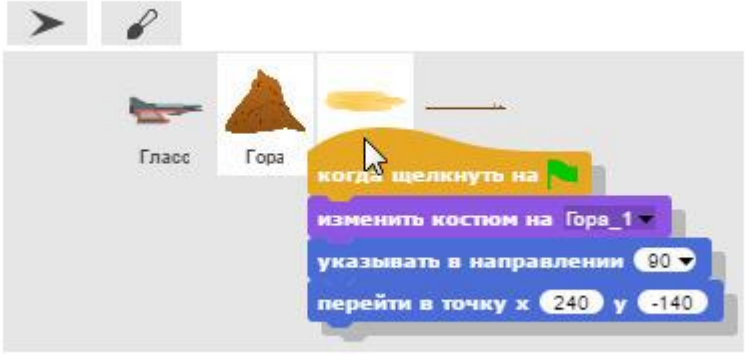

Два скрипта горы

Первый скрипт выбирает первый костюм горы, устанавливает ее в направление 90, и перемещает на правую границу сцены.

Следующий скрипт запускается после получения сообщения ***Старт***, передаваемого спрайтом Гласса при нажатии на ручку джойстика. Этот спрайт сделает так, что гора пять раз проплывет справа налево. При этом будет казаться, что корабль летит над землей (как в игре Flappy Bird). Каждый раз, начиная движение, гора будет надевать новый костюм, а затем будет перемещаться влево со скоростью ***Скорость_X***. Значение переменной ***Скорость_X*** мы умножаем на -1 для того, чтобы координата X изменялась на отрицательную величину. Если этого не сделать, то гора будет перемещаться вправо.

Блок ***ждать*** необходим для замедления работы проекта на быстрых компьютерах. Если ваш компьютер недостаточно быстрый, то можете удалить его.

Теперь запрограммируйте облако. У него будет всего два скрипта. Они практически такие же, как и у горы, поэтому скопируйте скрипты горы облаку. Для этого перетащите их в панель спрайтов на иконку спрайта облака.

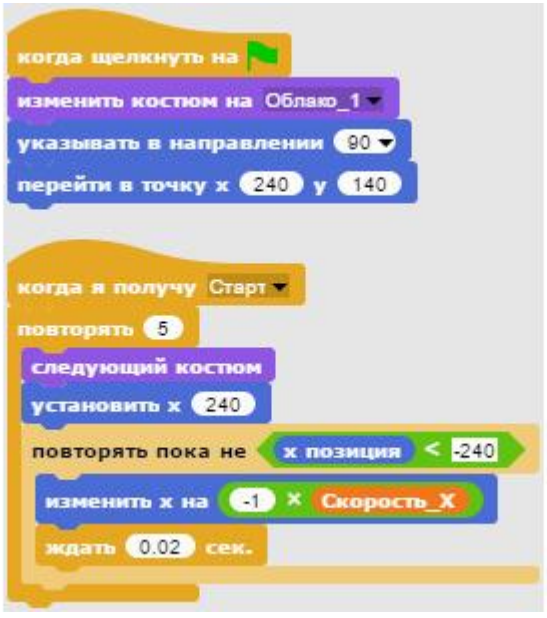



Отпускайте левую кнопку мышки только тогда, когда курсор окажется над спрайтом облака.

Скопируйте на облако второй спрайт горы.

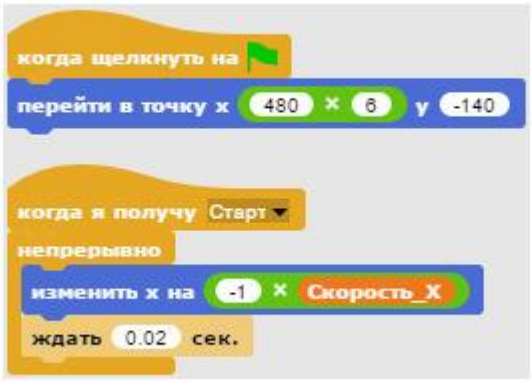

Отпускайте левую кнопку мышки только тогда, когда курсор окажется над спрайтом облака.

Затем измените название костюма в первом блоке. В итоге скрипты облака должны выглядеть вот так (координата Y в первом блоке равна не -140, а 140).




Скрипты облака

Теперь запрограммируйте космопорт. У него тоже всего два скрипта.

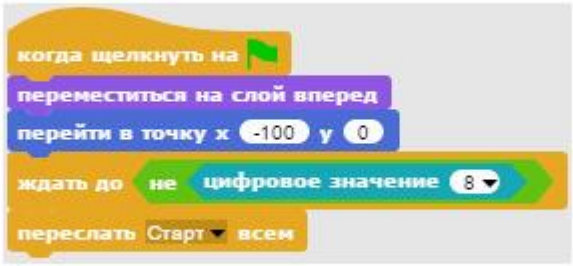



Скрипты космопорта

Первый скрипт перемещает космопорт далеко вправо за границу сцены на расстояние равное шести размерам сцены по горизонтали (480×6). Таким образом, когда под нашим кораблем проплывут все пять горных вершин, то за ними покажется космопорт.

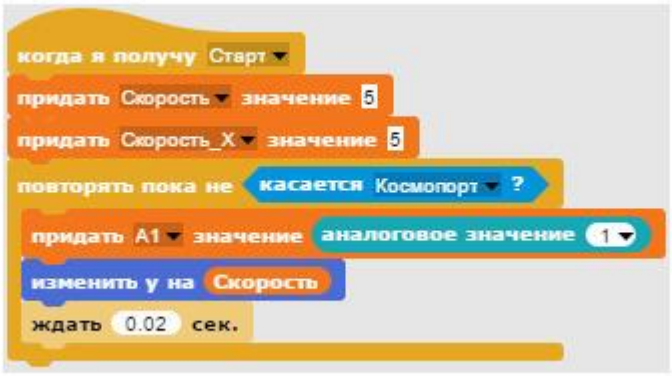
Второй скрипт постоянно перемещает космопорт влево со скоростью ***Скорость_X***. Такой же блок перемещает влево гору и облако.

Теперь запрограммируем космический корабль. У него самая сложная программа из шести скриптов.




Первый скрипт Гласса

Первый скрипт перемещает Гласса на слой вперед, чтобы он летел перед горами и облаками, затем перемещает в левую часть сцены и ждет нажатия на ручку джойстика. При нажатии на ручку джойстика скрипт перешлет всем спрайтам сообщение **Старт**. Это сообщение получит и сам Гласс, и приступит к выполнению следующих пяти скриптов, начинающихся с блока ***когда я получу Старт***.




Второй скрипт Гласса

Второй скрипт задает начальные значения переменных ***Скорость*** и ***Скорость_X***. Переменная ***Скорость*** отвечает за скорость вертикального перемещения корабля, а переменная ***Скорость_X*** отвечает за горизонтальную скорость перемещения гор и облаков, то есть за горизонтальную скорость корабля.

После установки начальных значений переменных корабль будет постоянно сохранять в переменной ***A1*** аналоговое значение, считанное с пина A1. Это будет происходить до тех пор, пока корабль не долетит до космопорта.

Также этот скрипт будет постоянно изменять вертикальное положение корабля, изменяя значение координаты Y на величину ***Скорость***.

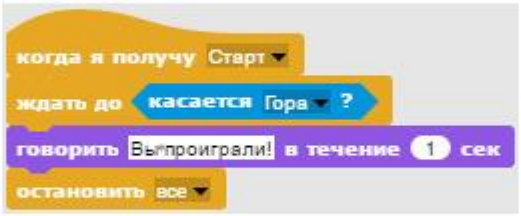

Третий скрипт Гласса

Третий скрипт корабля постоянно следит за величиной переменной ***A1***, и если оно станет больше чем 550 (ручка джойстика отклонена вверх), то значение переменной ***Скорость***увеличится до 5, а если ручка не отклонена, и корабль в свободном полете, то скорость будет непрерывно уменьшаться, изменяя значение переменной ***Скорость*** на -0.5, при этом Гласс будет плавно опускаться вниз.

Протестируйте работу проекта. Корабль должен лететь вперед, не взаимодействуя с горами, облаками и космопортом.

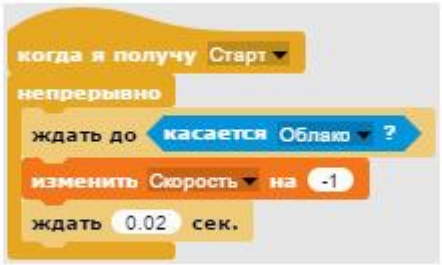
Будьте внимательны! Не перепутайте блоки ***придать*** и ***изменить***!

Если корабль нормально летит и управляется, то добавьте ему следующие три скрипта.



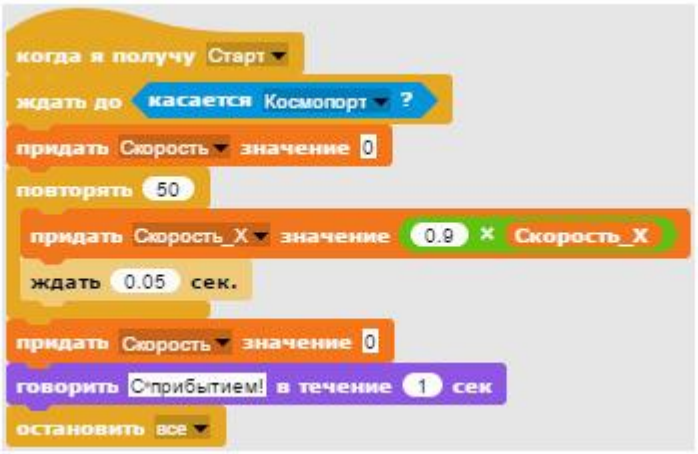

Четвертый скрипт Гласса

Четвертый скрипт постоянно ждет касания горы. Если это произойдет, то гора сообщит о проигрыше и завершит работу проекта.




Пятый скрипт Гласса

Пятый скрипт постоянно ждет касания облака. Если это произойдет, то скорость корабля резко замедлится. Не на -0.5, как обычно, а сразу на -1, ведь облака достаточно плотные и скорость движения в них быстро падает.




Шестой скрипт Гласса

Шестой скрипт – скрипт победы. Он постоянно ждет касания космопорта. Если это произойдет, то Гласс перестанет опускаться (***Скорость*** будет установлена в значение ноль), корабль плавно остановится, 50 раз подряд уменьшив значение переменной ***Скорость_X***, а затем установив ее значение в ноль. После остановки корабля он скажет: «С прибытием!», и остановит выполнение программы.

Сохраните проект, и протестируйте его работу.

**Задания.**

1. Измените кнопку запуска проекта на самую правую.
2. Увеличьте гравитацию, сделайте так, чтобы значение переменной скорость уменьшалось быстрее.
3. Увеличьте количество гор и облаков до 10.
4. Замедлите скорость подъема корабля в два раза.
5. Увеличьте плотность облаков, скорость в облаке должна падать в два раза быстрее.


Уже три дня живем на южной базе. Директор базы — один из моих лучших клиентов. Я продал ему несколько антикварных вещичек. Больше всего он дорожит древним, как хобот мамонта вездеходом с двигателем внутреннего сгорания «Армата». Он переделан из земного, где кислород находится в атмосфере на титановый лад, где и кислород и топливо находятся в баллонах.


Директор обожает шокировать путешественников, в первый раз оказавшихся на его космодроме. С диким ревом он подруливает к ним на своем вездеходе и просит предъявить документы. Целая стена в его кабинете украшена фотографиями ошарашенных путешественников, впервые в жизни увидевших ревущее железное чудовище.


От директора южной базы я узнал о давно интересующем меня объекте. Этот спутник располагался непосредственно внутри колец Сатурна, и считался необитаемым. Разведчики не обнаружили на нем ни больших запасов воды, ни полезных ископаемых. Завтра вылетаем на обследование.


Конечно, даже ребенок знает, что кольца Сатурна кишат пиратами, но я надеюсь, что нам они не повстречаются.


# Полет через кольца Сатурна

## Подготовка спрайтов

Создайте новый проект и подготовьте его к работе. Импортируйте спрайт космического корабля glass2_2.png, окрасьте сцену в темный цвет и подключите плату Arduino как описано в приложении 1. Переименуйте спрайт космического корабля в **Гласс**.

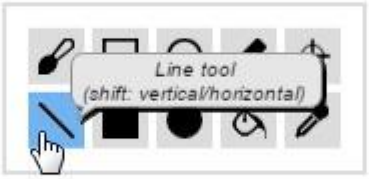
Создайте три новых спрайта, нажав на кнопку добавления спрайтов **(add a new Turtle sprite)**.


Выберите второй спрайт и импортируйте изображение метеороида. Дайте этому спрайту имя **Метеороид**.

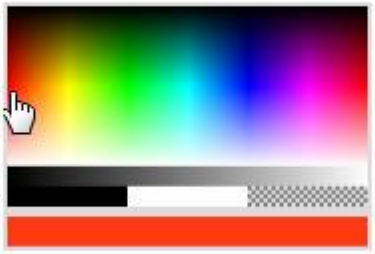

Выберите третий спрайт и импортируйте изображение звезды Звезда.png. Переименуйте этот спрайт в **Звезду**.

Затем выберите четвертый спрайт, и нарисуйте лазерный луч. Перейдите на вкладку **Костюмы** и нажмите на кнопку **Paint a new costume**.

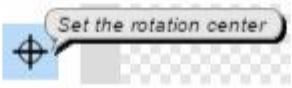

Выберите инструмент **Line tool**.



Выберите красный цвет.



Удерживая нажатой клавишу <Shift>, нарисуйте ровный горизонтальный лазерный луч, затем выберите инструмент **установить центр костюма (Set the rotation center)**.

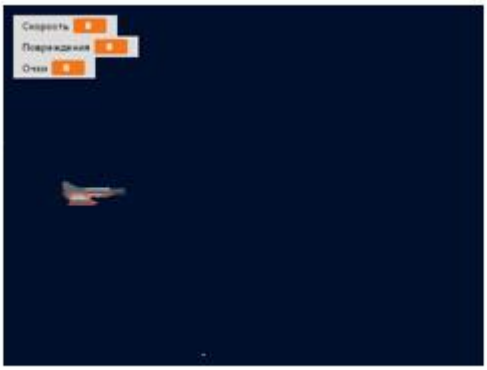



Установите центр костюма на левый конец лазерного луча.

Все спрайты готовы, можно программировать.


Все спрайты проекта

Объявите переменные ***Скорость***, ***Повреждения***, ***Очки*** и ***Пролетело***.
Сцена должна выглядеть вот так, переменная ***Пролетело*** должна быть скрыта.

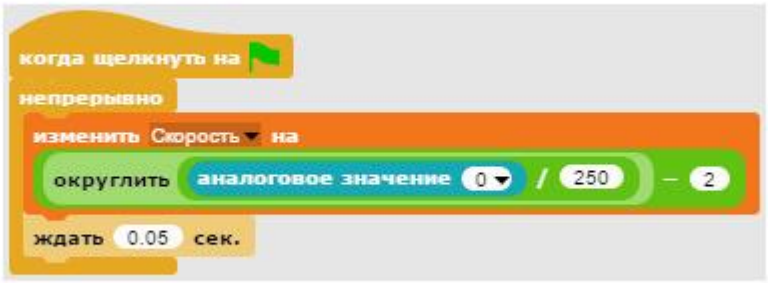


## Программирование

Выберите спрайт космического корабля и создайте для него программу из пяти скриптов.


Первый скрипт Гласса

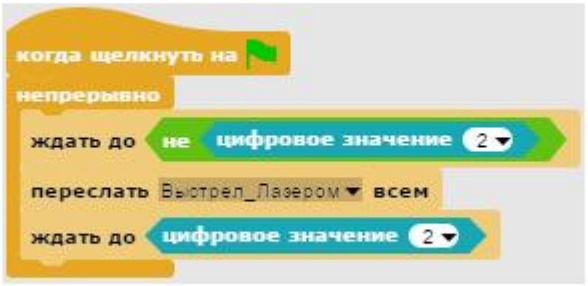
Первый скрипт Гласса обнуляет значения всех переменных, перемещает его в левую половину сцены, в точку (-150; 0), а затем непрерывно изменяет его координату Y в соответствии с аналоговым значением A1. Величина изменения координаты Y находится в диапазоне от -5 до 5. Таким образом, мы можем маневрировать, и уклоняться от столкновения с метеороидами, управляя вертикальным перемещением корабля.




Второй скрипт Гласса

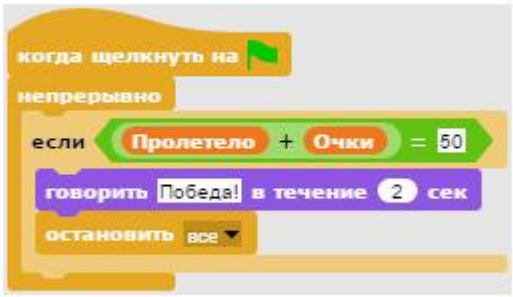
Второй скрипт Гласса непрерывно изменяет значение переменной ***Скорость***, отвечающую за скорость полета корабля. Переменная ***Скорость*** изменяется на величину из диапазона от -2 до 2, зависящую от аналогового значения, считанного с A0. На самом деле Гласс не будет перемещаться в горизонтальном направлении, перемещаться будут только метеороиды со скоростью, зависящей от значения переменной ***Скорость***.

Блок ***ждать*** создает небольшую задержку для корректной работы проекта на быстрых компьютерах. Если на вашем компьютере проект работает недостаточно быстро, то можете удалить этот блок.



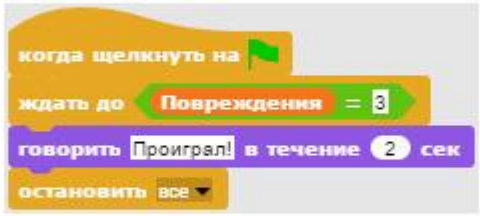

Третий скрипт Гласса

Третий скрипт космического корабля отслеживает нажатие на верхнюю кнопку и передает сообщение ***Выстрел_Лазером***.




Четвертый скрипт Гласса

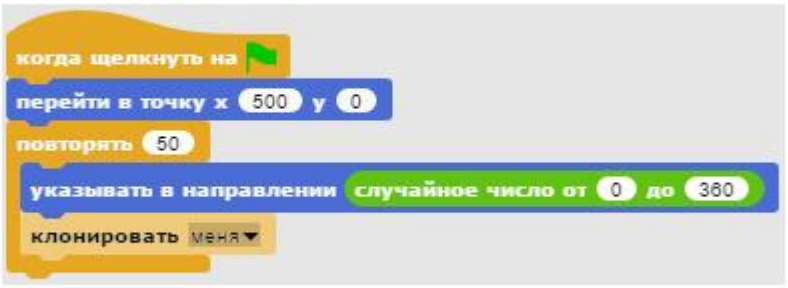
Четвертый скрипт Гласса непрерывно подсчитывает количество уничтоженных и пролетевших мимо метеороидов. Если сумма переменных ***Пролетело*** и ***Очки*** достигнет 50, то он сообщит о победе, и остановит выполнение программы.




Пятый скрипт Гласса

Пятый скрипт Гласса ждет, пока значение переменной ***Повреждения*** не станет равным 3 (пока корабль 3 раза не столкнется с метеороидом), после этого сообщает о проигрыше и останавливает выполнение проекта.

Теперь запрограммируйте метеороид. У него четыре скрипта.




Первый скрипт метеороида

Первый скрипт перемещает спрайт метеороида за пределы сцены, и создает 50 его клонов, каждый из которых повернут в случайном направлении.

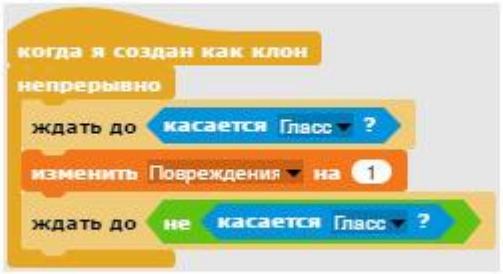

Второй скрипт метеороида

Этот скрипт отвечает за перемещение клонов метеороида. Он использует локальную переменную ***X_M***, которая у каждого клона своя. Она хранит начальную координату X метеороида. Значение этой переменной задается случайным образом из диапазона 500—10000. Блок ***установить X*** устанавливает метеороид в точку с координатой X, равной значению переменной ***X_M***. Координата Y также задается случайным образом из диапазона от -180 до 180.

После установки клона в начальную точку начинает непрерывно изменяться значение переменной ***X_M*** на значение переменной ***Скорость*** умноженное на -1 (для полета влево). Как только значение переменной уменьшится менее 240, клон переходит в соответствующую точку и показывается на экране.

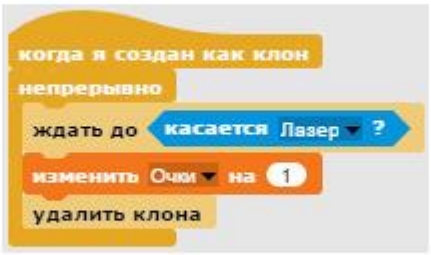
После того, как клон пролетит через всю сцену и достигнет точки с координатой менее -240, он изменит счетчик пролетевших метеороидов на 1 (переменную ***Пролетело***) и будет удален.

Блок ***ждать*** создает небольшую задержку для корректной работы проекта на быстрых компьютерах. Если на вашем компьютере проект работает недостаточно быстро, то можете удалить этот блок.




Третий скрипт метеороида

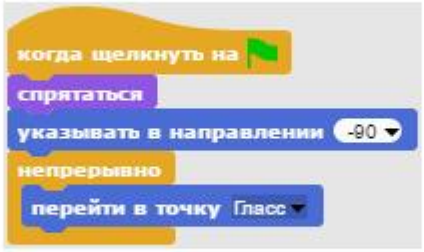
Третий скрипт метеороида следит за касанием космического корабля. Как только это произойдет, значение переменной ***Повреждения*** будет увеличено на 1.




Четвертый скрипт метеороида

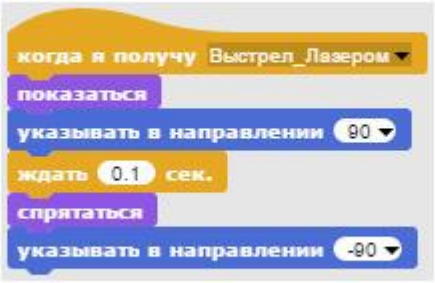
Четвертый скрипт метеороида следит за касанием лазерного луча. Как только клон метеороида коснется спрайта Лазер, то значение переменной ***Очки*** будет увеличено на единицу, и клон метеороида будет удален.

Теперь запрограммируем поведение лазерного луча. В его программе два скрипта.




Первый скрипт лазера

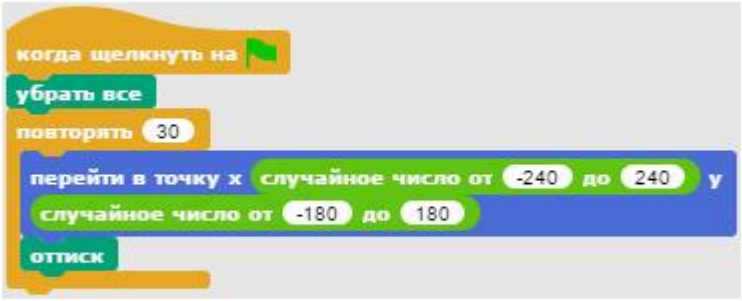
Первый скрипт лазера скрывает его и поворачивает влево. Таким образом, выключенный Лазер не будет касаться летящих метеороидов, и сбивать их.




Второй скрипт лазера

Второй скрипт лазера начинает работать при получении сообщения ***Выстрел_Лазером***. В этот момент Лазер покажется, повернется вправо, а через долю секунды снова исчезнет, и отвернется в левую сторону.

Теперь запрограммируйте звезду, у нее всего один скрипт.



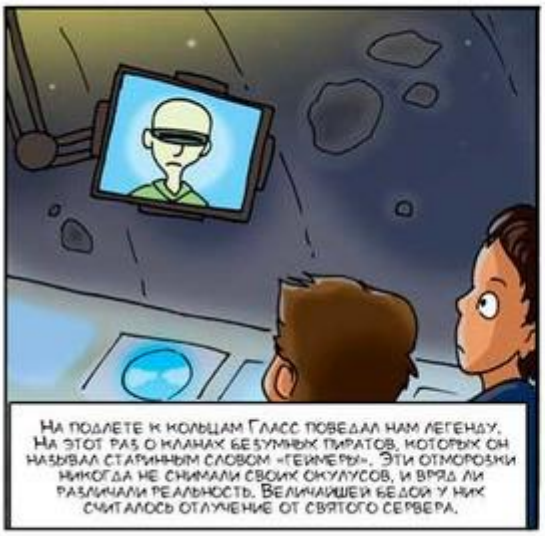

Скрипт звезды

При запуске программы звезда будет очищать сцену, а затем 30 раз переместится в случайную точку сцены и отпечатается на ней с помощью блока ***оттиск***. При этом сцена станет больше похожа на настоящий космос.

Сохраните проект и протестируйте его работу.

## Задания.

1. Измените количество метеороидов с 50 до 70.
2. Уменьшите скорость разгона Гласса.
3. Запрограммируйте на стрельбу самую правую кнопку джойстика.
4. Ускорьте перемещение Гласса по вертикали.



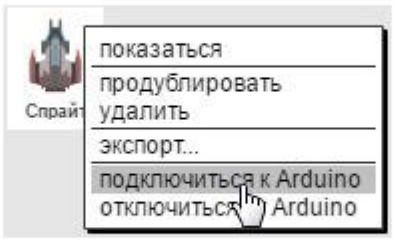




# Защита Дафниса

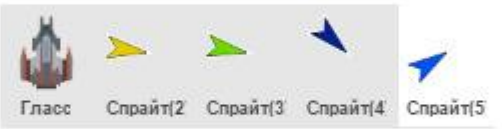
## Подготовка спрайтов

Создайте новый проект. Импортируйте изображение космического корабля glass_2.png, дайте спрайту имя **Гласс**. Подключите плату Arduino к спрайту космического корабля. Для этого кликните по спрайту правой кнопкой мышки и выберите команду **подключиться к Arduino**.




Создайте 4 спрайта, нажав на кнопку добавления спрайтов (**add a new Turtle sprite**).


Всего получится 5 спрайтов.

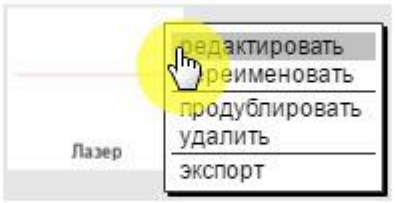


Выберите второй спрайт и импортируйте изображение поверхности. Если поверхность размещена не горизонтально, то дважды кликните на блок ***указывать в направлении 90***, и она ляжет ровно.

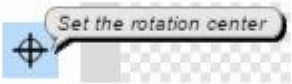

Поместите спрайт поверхности в нижней части сцены и переименуйте его в **Поверхность.**

Выберите третий спрайт, импортируйте изображение звезды, и дайте спрайту новое имя **Звезда.**

Затем выберите четвертый спрайт, импортируйте изображение лазера, дайте спрайту имя **Лазер**, и перейдите на вкладку **Костюмы**. Кликните правой кнопкой на костюм лазера и выберите **редактировать.**



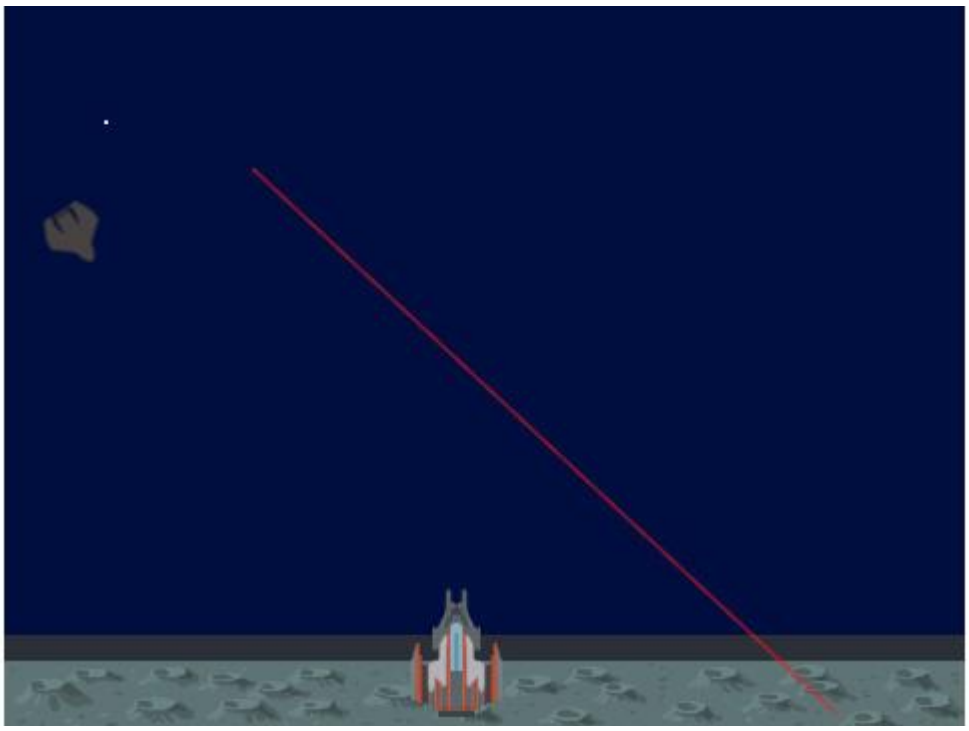

Выберите инструмент **установить центр костюма (Set the rotation center)**.




Установите центр костюма на левый конец лазерного луча и нажмите **ОК**.

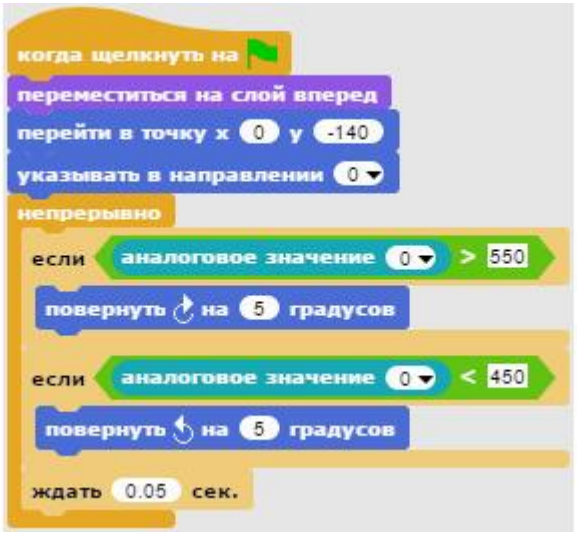
Затем выберите пятый спрайт, дайте ему имя **Враг**, и импортируйте поочередно три изображения пиратских космических кораблей из файлов pirate_1.png, pirate_2.png, pirate_3.png.

Сцена должна выглядеть вот так, не забудьте окрасить ее в темный цвет. Обратите внимание, пока программа не запущена, лазерный луч никак не связан со спрайтом Гласса, и расположен как попало.



## Программирование

Теперь запрограммируйте спрайт космического корабля, у него всего два скрипта.



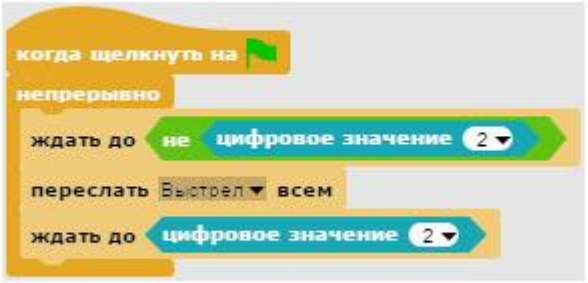

Первый скрипт Гласса

Первый скрипт Гласса перемещает его в самый верхний слой, в нижнюю часть сцены и устанавливает в вертикальном направлении. Затем непрерывно изменяет направление корабля в соответствии с аналоговым значением A0, считанным с джойстика. Если значение A0 больше 550, то Гласс поворачивается на 5 градусов по часовой стрелке, а если оно меньше 450, то на 5 градусов против часовой.

Блок ***ждать*** создает небольшую задержку для корректной работы проекта на быстрых компьютерах. Если на вашем компьютере проект работает недостаточно быстро, то можете удалить этот блок.

Перед созданием второго скрипта Гласса вытащите в область скриптов блок ***переслать*** и создайте новое событие ***Выстрел***.

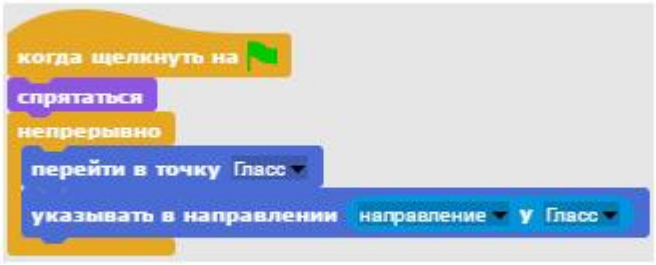

Соберите второй скрипт Гласса.




Второй скрипт Гласса

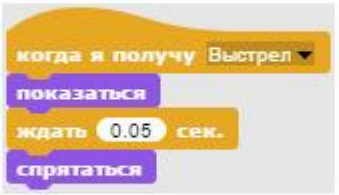
Второй скрипт Гласса ждет нажатия на верхнюю кнопку и передает сообщение ***Выстрел***. Это сообщение активирует лазерный луч.

Теперь создайте два скрипта для Лазера.




Первый скрипт Лазера

Первый скрипт Лазера прячет его, совмещает с Глассом и поворачивает его в том же направлении что и у него. Теперь при повороте космического корабля луч лазера будет поворачиваться вместе с ним.




Второй скрипт Лазера

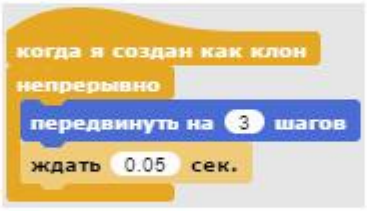
Второй скрипт космического корабля запускается при получении сообщения ***Выстрел***. При этом Лазер покажется на долю секунды и спрячется снова.

Запрограммируйте спрайт Врага, у него всего три скрипта.


Первый скрипт Врага

Первый скрипт врага перемещает его в начальную точку, поворачивает в направлении 180 градусов, и удаляет всех клонов, оставшихся от прошлого запуска игры. Если этого не сделать, то при втором запуске игры планету будет атаковать не 9, а 99 клонов, что доставит нам некоторые неудобства.

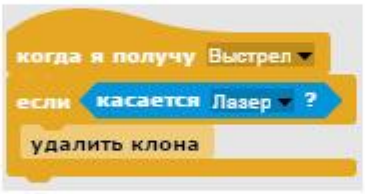
После этого с интервалом в 2 секунды создаются 9 атакующих клонов пиратских кораблей различного типа, ведь внешний вид корабля каждый раз изменяется при помощи блока ***следующий костюм***.




Второй скрипт Врага

Второй скрипт врага программирует полет клонов к поверхности. Они постоянно двигаются в направлении 180 градусов (вниз) со скоростью 3 шага за раз.

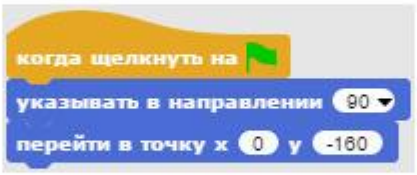
Блок ***ждать*** создает небольшую задержку для корректной работы проекта на быстрых компьютерах. Если на вашем компьютере проект работает недостаточно быстро, то можете удалить этот блок. Если же наоборот, враги летают слишком шустро, то можете увеличить значение в блоке до 0.1 или 0.15.




Третий скрипт врага

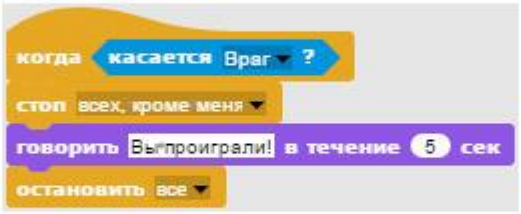
Третий скрипт выполняется при получении сообщения ***Выстрел***. Если в этот момент клон касается лазера, то он будет удален.

Теперь запрограммируйте поверхность, у нее будет два скрипта.




Первый скрипт поверхности

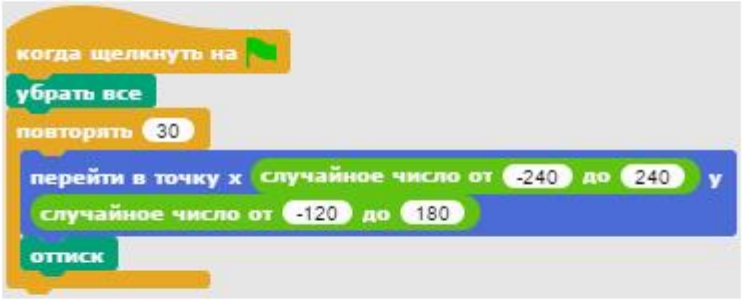
Первый скрипт поверхности просто перемещает ее в самый низ сцены и поворачивает в направлении 90 градусов.




Второй скрипт поверхности

Второй скрипт запускается только тогда, когда поверхности касается враг. Это означает, что вам не удалось защитить Дафнис от нападения. Когда враг коснется поверхности, блок ***стоп*** остановит выполнение всех скриптов проекта кроме этого, затем он сообщит о проигрыше и полностью остановит выполнение проекта.

Теперь запрограммируйте звезду, у нее всего один скрипт.



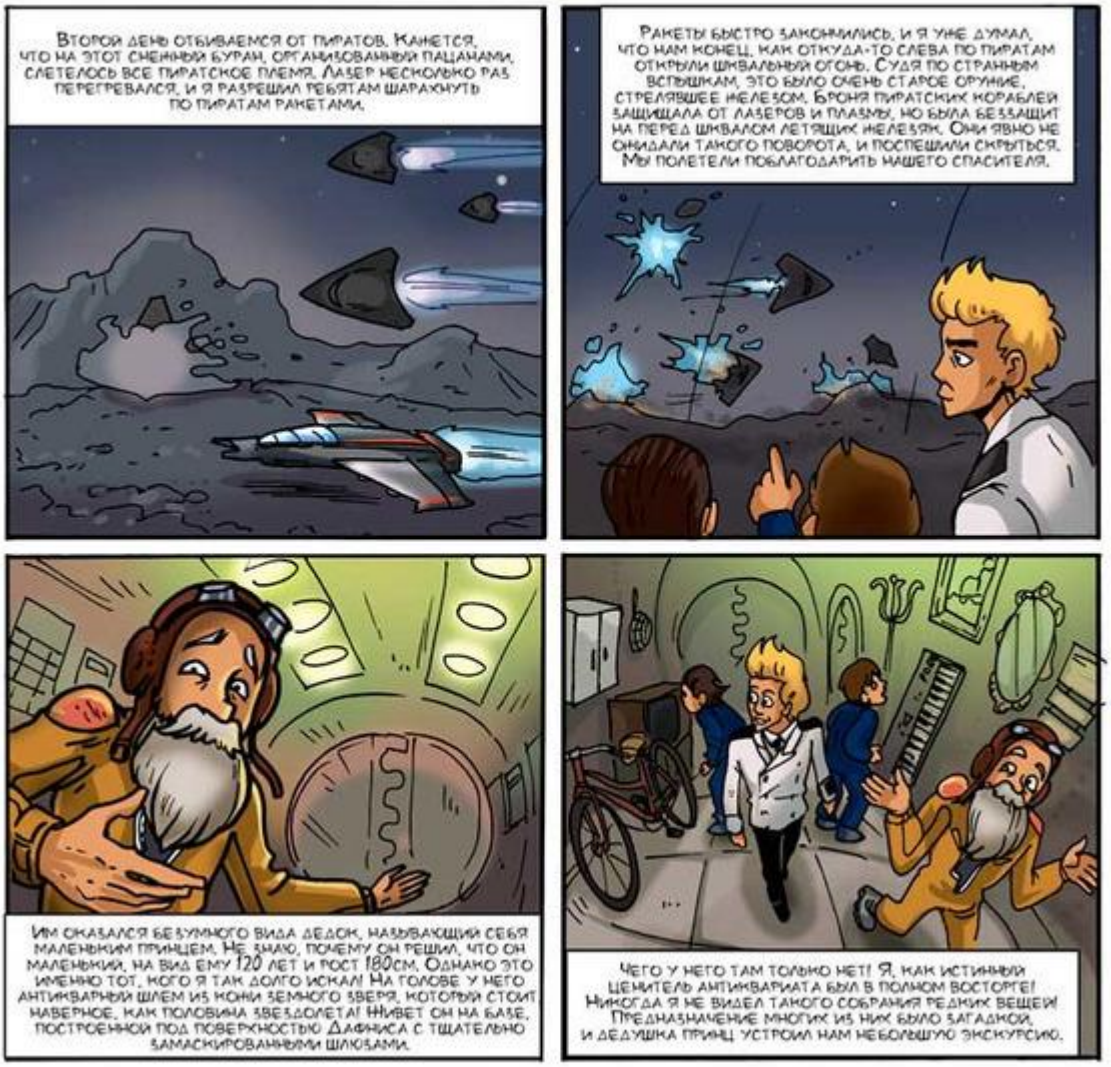

Скрипт звезды

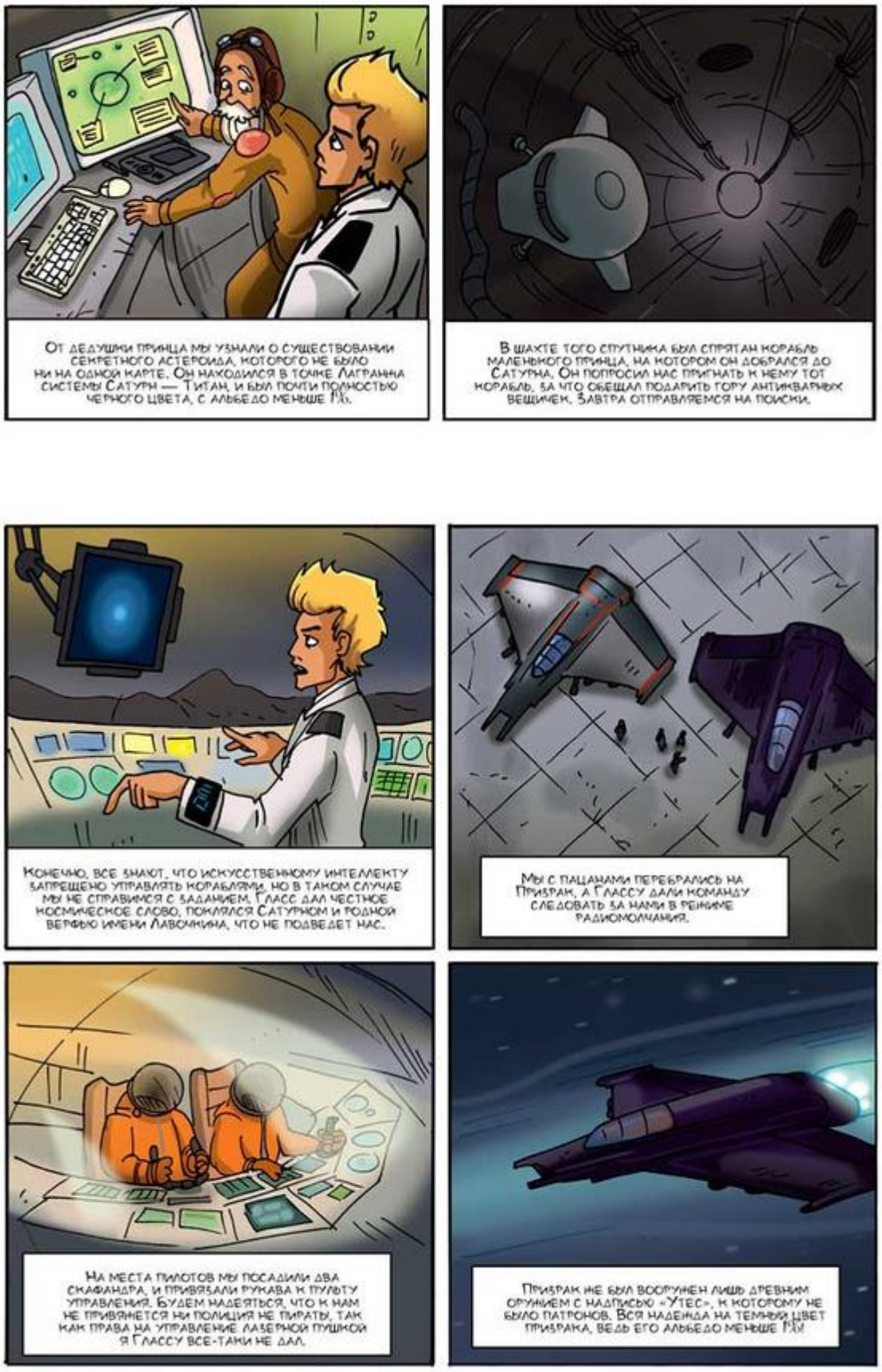
При запуске программы звезда будет очищать сцену, а затем 30 раз переместится в случайную точку сцены и отпечатается на ней с помощью блока ***оттиск***. При этом сцена станет больше похожа на настоящий космос.

Сохраните проект, а затем запустите его и защитите Дафнис от незваных гостей!

## Задания.

1. Замедлите движение врагов.
2. Ускорьте поворот корабля.
3. Добавьте вражеским кораблям по 2 жизни. Для этого используйте локальные переменные.
4. Увеличьте количество вражеских клонов до 20.
5. Сделайте так, что проигрыш будет наступать в тот момент, когда поверхности коснулось пять пиратских кораблей.





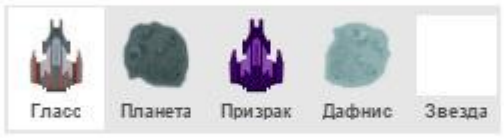

# Поиск таинственной планеты

## Подготовка спрайтов

Создайте новый проект. Импортируйте изображение космического корабля glass.png, дайте спрайту имя **Гласс**. Окрасьте фон в темный цвет.

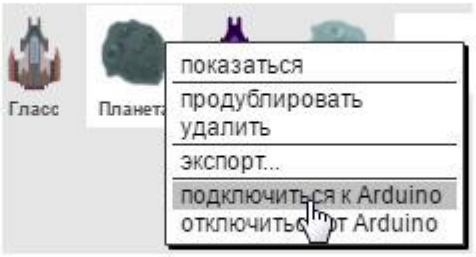
Создайте 4 спрайта, нажав на кнопку добавления спрайтов (**add a new Turtle sprite**).


Всего получится 5 спрайтов. Выберите второй спрайт и импортируйте изображение планеты из файла aster-1.png. Выберите третий спрайт и импортируйте изображение призрака из файла prizrak.png. Выберите четвертый спрайт и импортируйте изображение планеты из файла aster-1.png. Выберите пятый спрайт и импортируйте изображение звезды из файла Звезда.png.

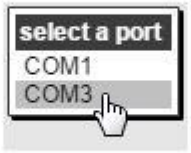



Объявите четыре переменные ***Случайное, Расстояние, A0, A1***.

Подключите Arduino к спрайту Планеты.

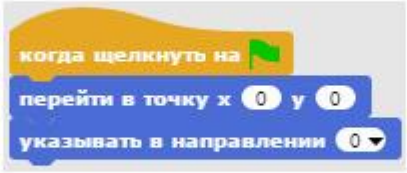



Выберите порт.




## Программирование

Запрограммируйте космический корабль. Он будет постоянно находиться в центре сцены, перемещаться будут только небесные тела – Дафнис, Планета и звезды. У Гласса будет всего три скрипта.

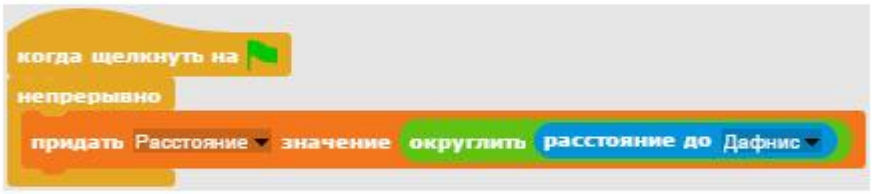



Первый скрипт Гласса

Первый скрипт устанавливает его в центре сцены и поворачивает в направлении 0 градусов.


Второй скрипт Гласса

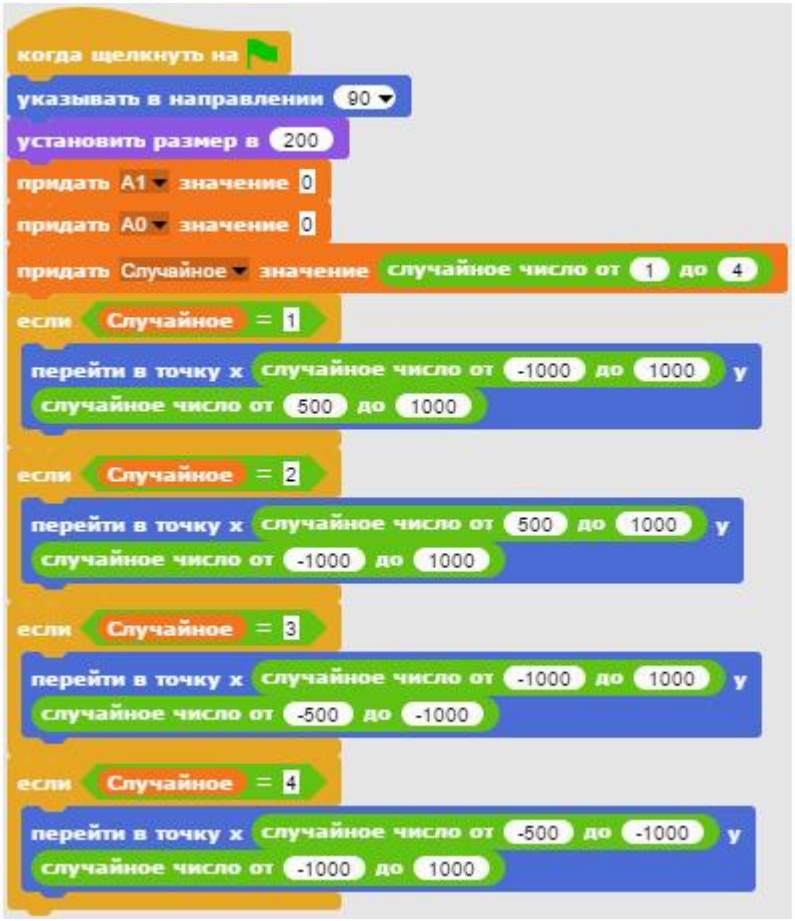
Второй скрипт постоянно ждет того, что Гласс коснется спрайта таинственной планеты. В этот момент будет передано событие ***Нашел***, и появится космический корабль Призрак.




Третий скрипт Гласса

Третий скрипт помогает игроку ориентироваться в бескрайнем космосе. Для этого он постоянно отображает на экране значение переменной ***Расстояние***, равное округленному расстоянию до Дафниса. По изменению значения этой переменной вы сможете определить направление вашего движения, приближаетесь ли вы к Дафнису, или удаляетесь от него.

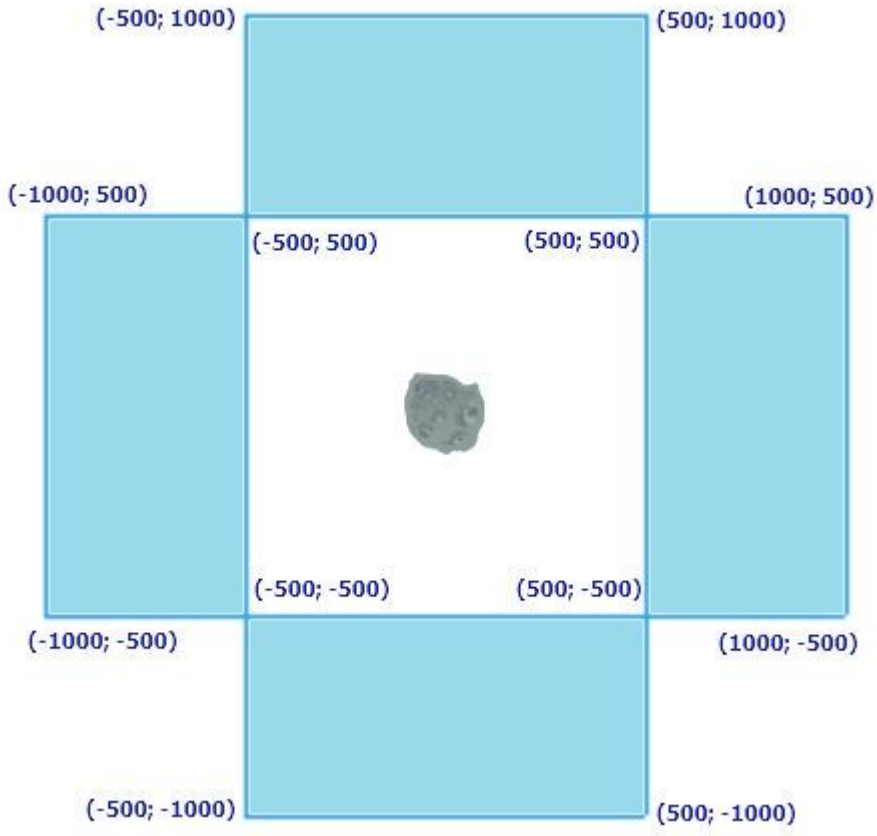
Теперь запрограммируйте планету, ее программа состоит из шести скриптов.



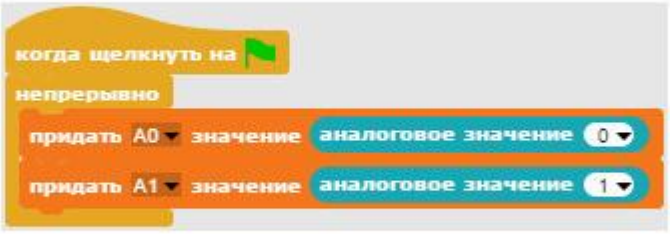

Первый скрипт планеты

Первый скрипт планеты поворачивает ее в направлении 180 градусов, увеличивает в размере до 200%. Далее случайным образом вычисляются координаты X и Y, определяя положение планеты в космическом пространстве.

Планета должна быть не слишком близко и не слишком далеко от Дафниса, чтобы была возможность ее обнаружить. Она будет располагаться в одном из четырех прямоугольников размером 500 на 1000 пикселов, в каком именно определит переменная ***Случайное***. Ее значение выбирается из следующего ряда чисел 1, 2, 3, 4. Если значение этой переменной равно 1, то Планета появится в верхнем прямоугольнике, если 2, то в правом, если 3, то в нижнем, а если 4, то в левом.

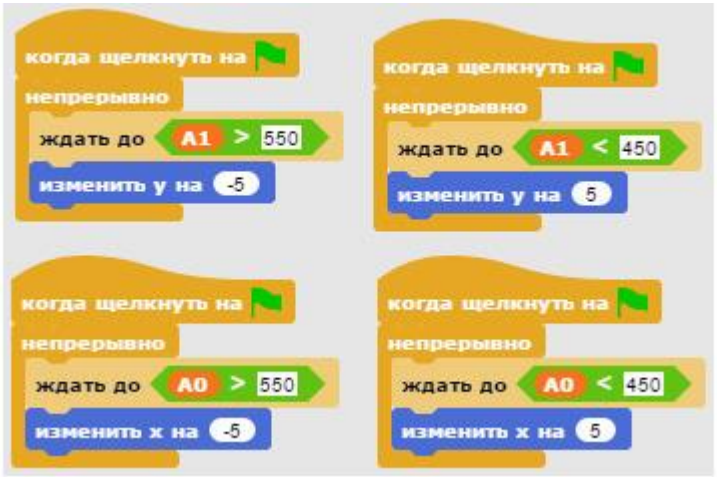



Следующий скрипт планеты постоянно считывает аналоговые значения с пинов A0 и A1 и запоминает их в соответствующих переменных.



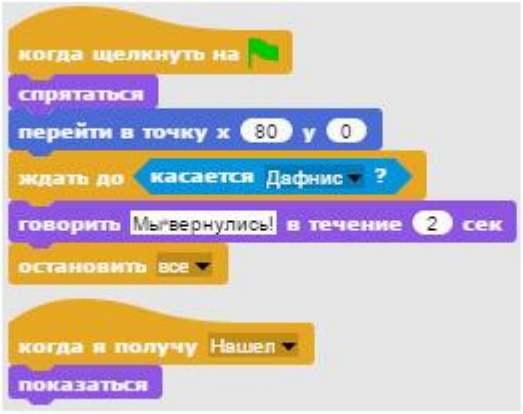

Второй скрипт планеты

Следующие четыре скрипта планеты перемещают ее в соответствии со значениями переменных ***A0*** и ***A1***.




Третий – шестой скрипты планеты

Теперь запрограммируйте космический корабль Призрак, который будет обнаружен на таинственной планете.

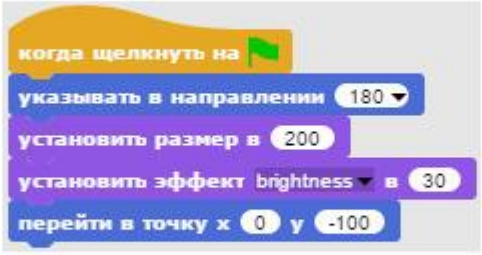



Первый и второй скрипты Призрака

При запуске программы Призрак спрячется и переместится в точку с координатами (80; 0), это немного правее Гласса. Затем Призрак ждет, пока не коснется Дафниса, при касании скажет: «Мы вернулись!», и работа проекта будет остановлена.

Второй скрипт Призрака делает его видимым при получении сообщения ***Нашел***.

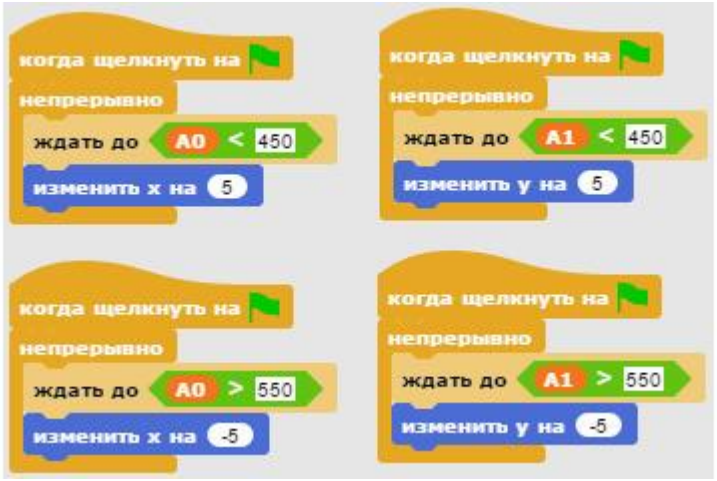
Теперь сделайте пять скриптов для Дафниса.




Первый скрипт Дафниса

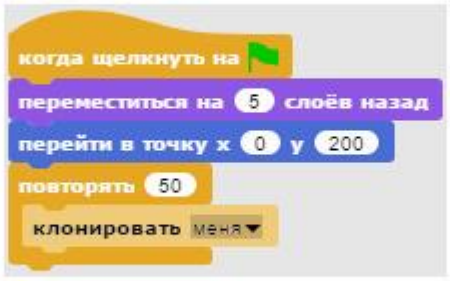
Первый скрипт поворачивает спрайт в направлении 180 градусов, увеличивает в два раза (до 200%), немного осветляет его (чтобы отличался от Планеты) и перемещает в точку (0; -100).

Следующие четыре скрипта Дафниса перемещают его в соответствии со значением переменных ***A0*** и ***A1***. Эти скрипты идентичны скриптам Планеты, можете скопировать их у нее.



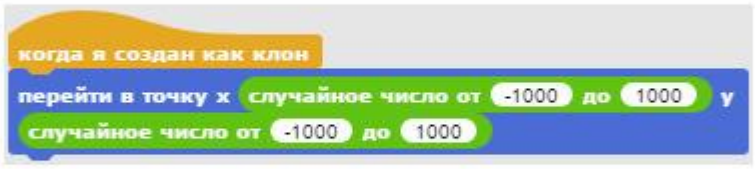

Второй – пятый скрипты Дафниса

Для того чтобы перемещения в космосе были реалистичнее, мы добавили спрайт звезды. Запрограммируйте ее, у нее шесть маленьких скриптов. Некоторые из них идентичны скриптам Планеты, можете скопировать их у нее, перетащив на Звезду в области спрайтов.




Первый скрипт звезды

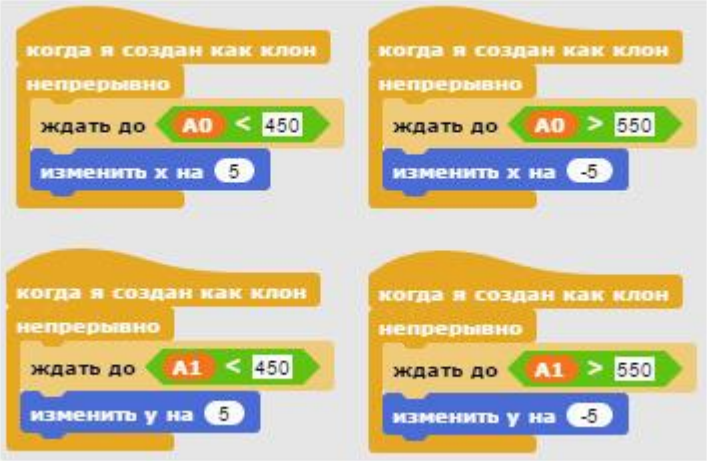
Первый скрипт перемещает звезду на 5 слоев назад (чтобы она была на заднем плане), отправляет в точку (0; 200), и 50 раз клонирует ее. Следующие скрипты управляют пятью созданными клонами звезды.




Второй скрипт звезды

Этот скрипт размещает созданные клоны звезды в случайно выбранных точках сцены.

Следующие четыре скрипта перемещают клоны звезды в соответствии со значением переменных ***A0*** и ***A1***. Эти скрипты идентичны скриптам Планеты, и Дафниса, можете скопировать их оттуда.




Третий – шестой скрипты звезды

Проект готов. Сохраните его и протестируйте. Искать загадочную планету непросто. Внимательно следите за направлением вашего полета и изменением значения переменной ***Расстояние***.

**Задания.**

1. Увеличьте количество звезд.
2. Сделайте так, чтобы Гласс поворачивался в направлении полета.
3. Установите Призраку эффект прозрачности в 90%.
4. Увеличьте скорость полета.


# Охота на дикую шаурму

## Подготовка спрайтов

Создайте новый проект, импортируйте изображение прицела из файла Прицел.png, и переименуйте спрайт в **Прицел**. Подключите к нему плату Arduino. Окрасьте сцену в светло-оранжевый цвет, и создайте новый спрайт, нажав на кнопку добавления спрайтов (**add a new Turtle sprite**).

Выделите новый спрайт и импортируйте четыре изображения дикой шаурмы.


Переименуйте новый спрайт во **Врага**.


## Программирование

Перед началом программирования объявите переменную ***Очки***, затем запрограммируйте прицел, у него будет три скрипта.

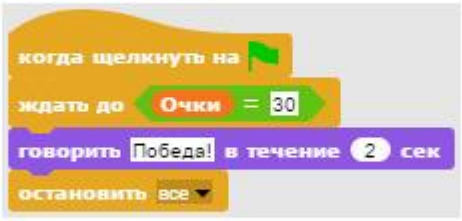

Первый скрипт прицела

Первый скрипт прицела обнуляет значение переменной ***Очки***, перемещает его в начальную точку и постоянно изменяет координаты прицела в зависимости от положения ручки джойстика и значений считанных с аналоговых пинов A0 и A1.

Блок ***ждать*** необходим для замедления работы проекта на быстрых компьютерах. Если ваш компьютер недостаточно быстрый, то можете удалить его.


Второй скрипт прицела

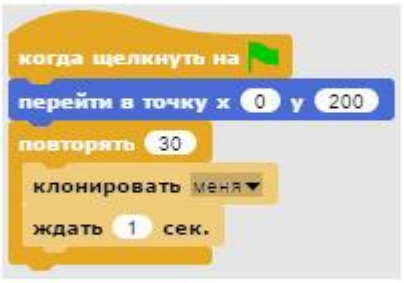
Второй скрипт прицела постоянно передает сообщение ***Нет выстрела***, и ожидает нажатия на верхнюю кнопку джойстика. При нажатии на нее скрипт передает сообщение ***Выстрел***.




Третий скрипт прицела

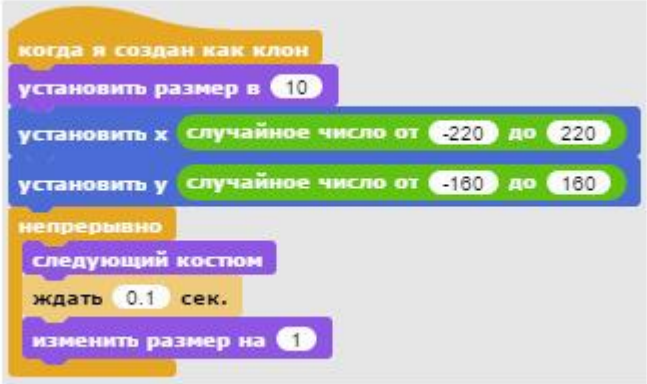
Третий скрипт прицела ждет, пока значение переменной ***Очки*** не станет равно 30, тогда он сообщает о победе и останавливает выполнение проекта.

Теперь запрограммируйте спрайт врага, у него всего четыре скрипта.



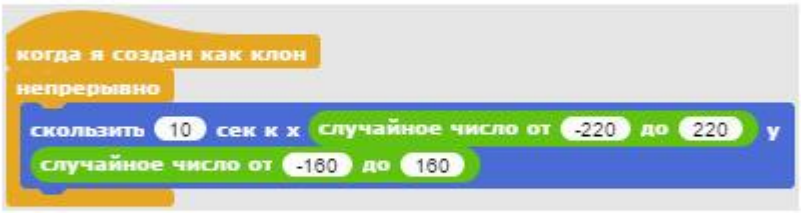

Первый скрипт врага

Первый скрипт врага перемещает его в начальную точку, а затем создает 30 его клонов с интервалом в 1 секунду.



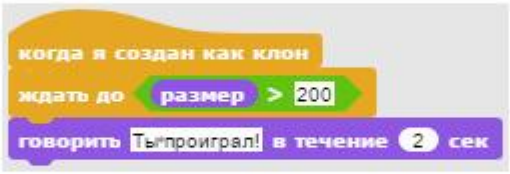

Второй скрипт врага

Новый клон сразу становится маленьким (размер 10%), и переходит в случайную точку сцены. Затем он непрерывно, с интервалом в 0.1 секунду меняет костюмы для анимации крыльев, и немного увеличивает свой размер. При этом будет казаться, что враг приближается.




Третий скрипт врага

Все клоны не стоят на месте, а плавно скользят в случайно выбранную точку сцены, это делает полет врагов непредсказуемым и усложняет игру.




Четвертый скрипт врага

Четвертый скрипт врага проверяет, насколько большим он стал. Если размер клона больше 200%, то это значит, что враг подлетел очень близко, и дикая шаурма вас съела.


Пятый скрипт врага

Пятый скрипт врага ждет одновременного наступления двух условий: выстрела и касания клона прицелом. Как только это произойдёт (вы попали во врага), то значение переменной ***Очки*** изменится на 1, и клон будет удален.

Сохраните проект и протестируйте его работу. Удачной охоты!

## Задания.

1. Упростите игру, уменьшив количество клонов.
2. Упростите игру, замедлив «приближение» врагов.
3. Упростите игру, ускорив скорость перемещения прицела.
4. Сделайте так, чтобы игрок проигрывал только тогда, когда долетят пять врагов.

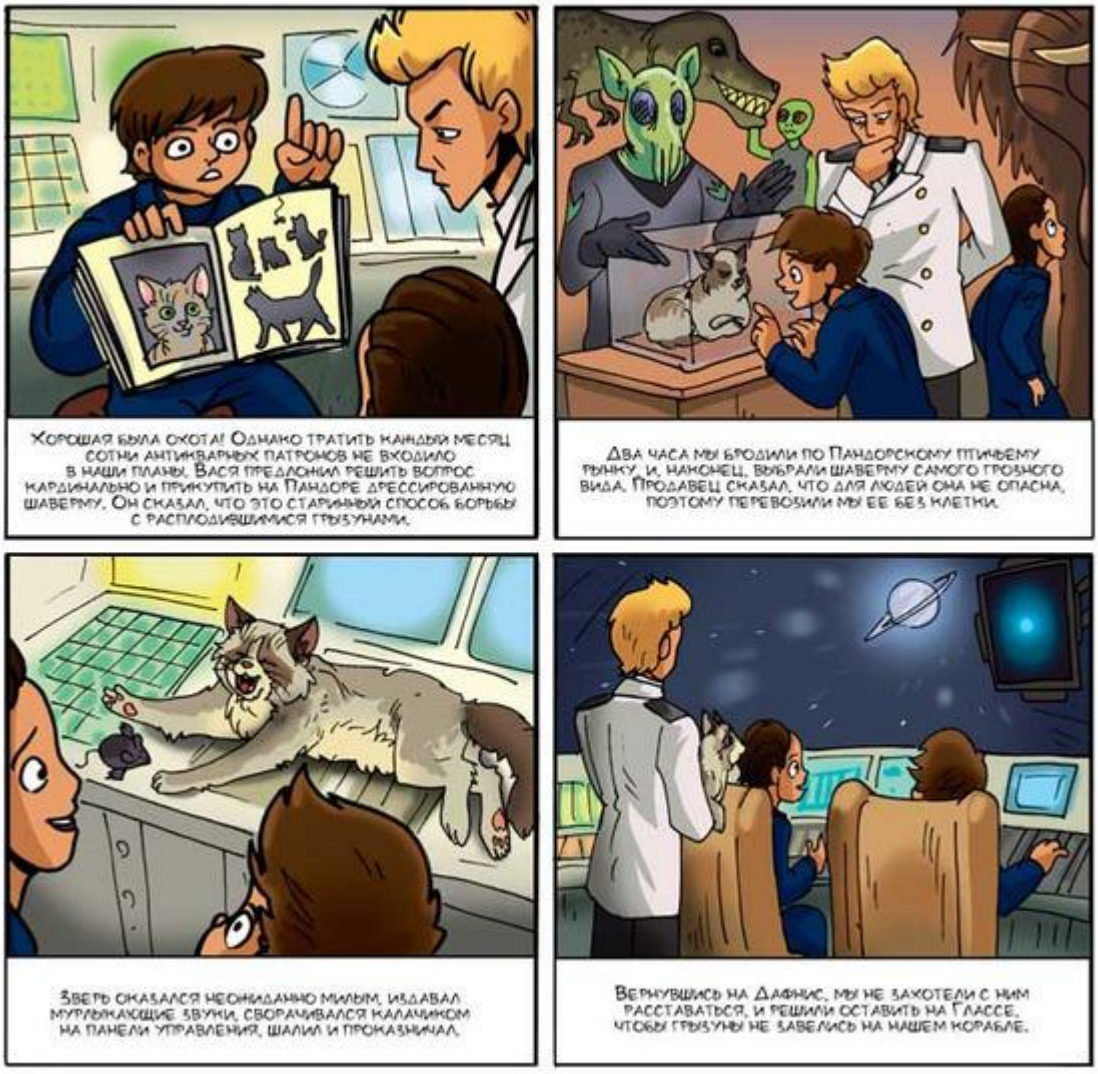
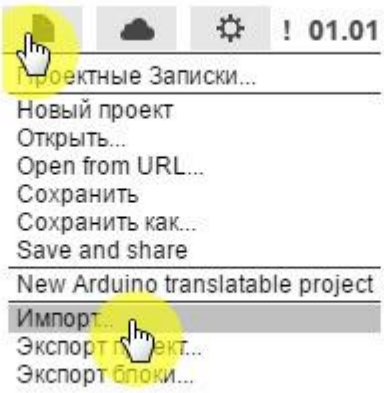


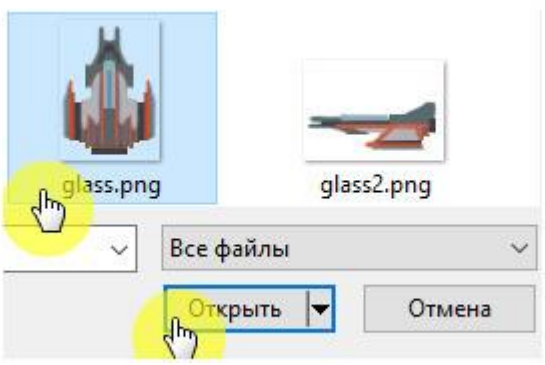
Фото питомца

# Приложение 1
# Подготовка спрайтов и сцены

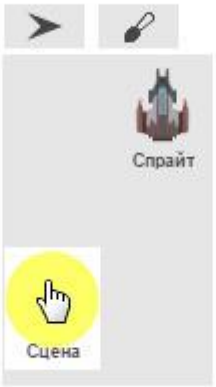
Создайте новый проект и импортируйте изображение космического корабля.



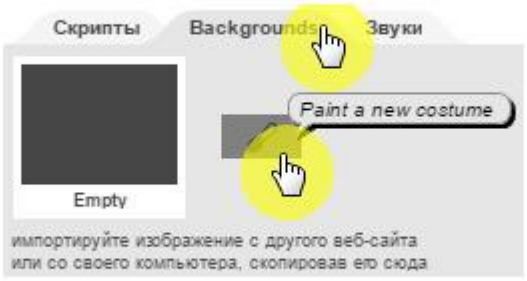
Выберите файл glass.png.



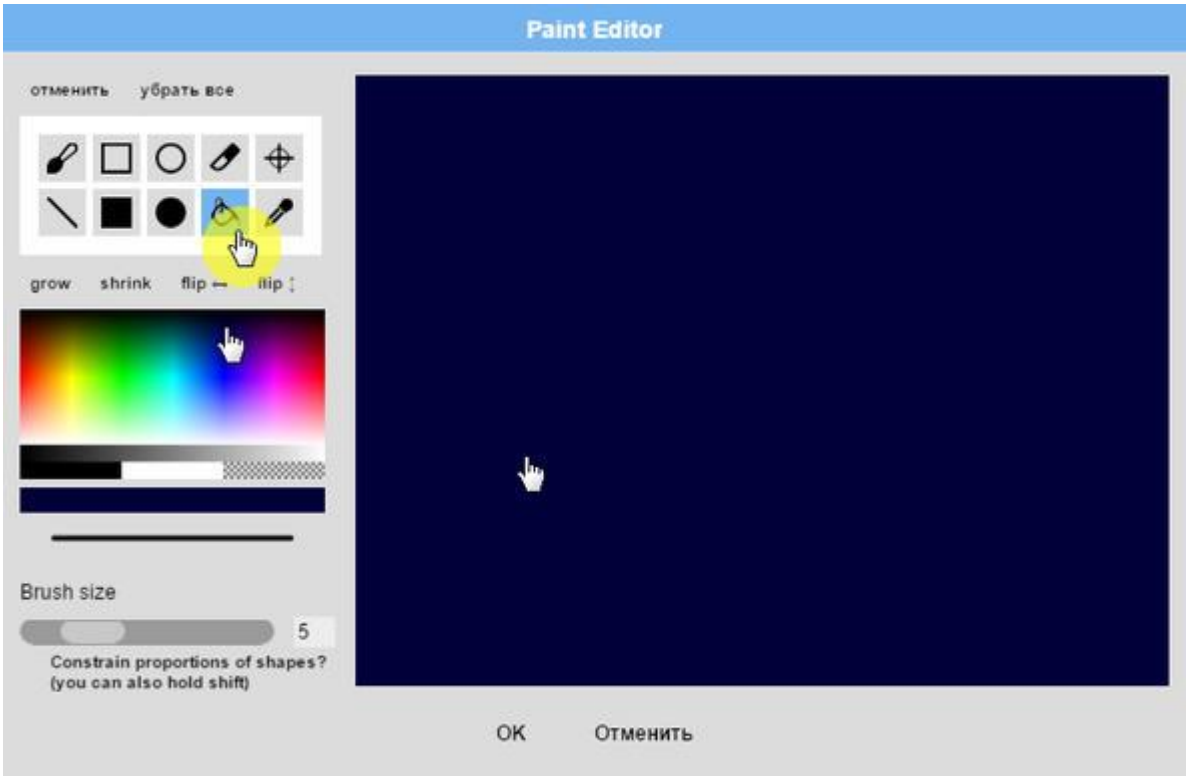

Теперь измените белый фон на что-нибудь более похожее на космос. Выберите сцену.




Перейдите на вкладку **Backgrounds** и нажмите на кисточку.

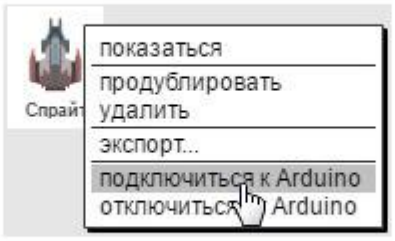



В открывшемся редакторе выберите инструмент **Заливка** (Fill the region), выберите цвет, кликнув по палитре цветов, залейте фон темным цветом, и нажмите **ОК**.

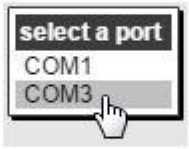



Почти настоящий космос!

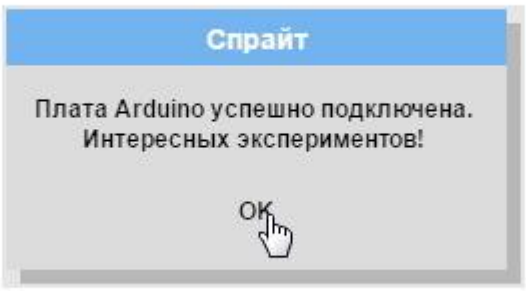
С помощью USB провода подключите плату Arduino к компьютеру. Затем кликните по спрайту корабля правой кнопкой мышки и выберите команду **подключиться к Arduino**.




Выберите порт, к которому подключена плата Arduino (обычно это не COM1, а другой порт).

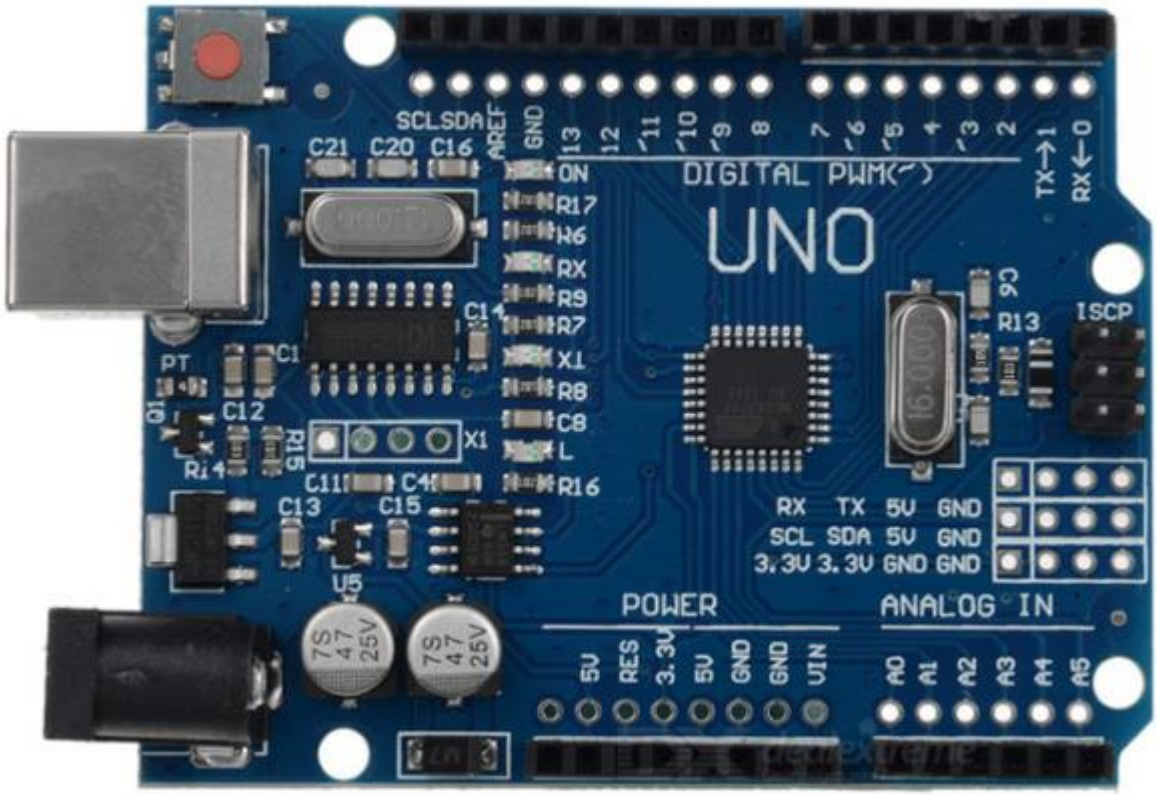



Если подключение прошло успешно, то вы увидите следующее сообщение. Нажмите на **ОК** и приступайте к созданию проекта.




# Приложение 2
# Принцип работы платы Arduino

## Общая информация




Плата Arduino UNO сконструирована на базе микроконтроллера ATmega328 (это самая большая микросхема на плате). Питание на плату подается от компьютера по проводу USB, напряжение питания 5 вольт. Плата Arduino UNO имеет 14 цифровых выводов (пинов) и 6 аналоговых выводов.

## Цифровые выводы

Цифровые выводы сообщают плате Arduino о поданном на них напряжении. Эти пины различают только два вида напряжения – высокий уровень (5 вольт) и низкий уровень (0 вольт). Если на цифровом пине 2 высокий уровень напряжения (5 вольт), то это означает, что Snap4Arduino прочитав значение с этого пина с помощью блока ***цифровое значение 2***, получит информацию о его состоянии **истина** (**true**).

Кликните на блоке для просмотра возвращаемого значения.

Если на цифровом пине 2 низкий уровень напряжения (0 вольт), то это означает, что Snap4Arduino прочитав значение с этого вывода, получит информацию о его состоянии **ложь**(**false**).


При установке на Arduino платы Joystick Shield, ее кнопки подключаются к следующим цифровым выводам.

Кнопка вверх – Пин 2

Кнопка вправо – Пин 3

Кнопка вниз – Пин 4

Кнопка влево – Пин 5

Кнопки подключены таким образом, что пока они не нажаты, на соответствующих пинах присутствует высокое напряжение и Snap4Arduino получает значения **истина** от каждого из четырех пинов. И, наоборот, при нажатии на кнопки Snap4Arduino будет получать значения **ложь** от соответствующих пинов.

## Аналоговые выводы

Аналоговые выводы также сообщают плате Arduino о поданном на них напряжении, но, в отличие от цифровых выводов, они могут измерять поданное на них напряжение. Например, об уровне поданного на пин A0 напряжения Snap4Arduino может узнать с помощью блока ***аналоговое значение 0***.

Аналоговое значение, считанное с пина, может изменяться от 0 (при напряжении на пине 0 вольт), до 1023 (при напряжении на пине 5 вольт).

К аналоговым пинам A0 и A1 подключен джойстик, установленный на плате Joystick Shield. Джойстик состоит из двух переменных резисторов, напряжение на которых изменяется в зависимости от уровня наклона ручки джойстика.

Ручка влево – Пин A0 = 0

Ручка вправо – Пин A0 = 1023

Ручка вверх – Пин A1 = 1023

Ручка вниз – Пин A1 = 0

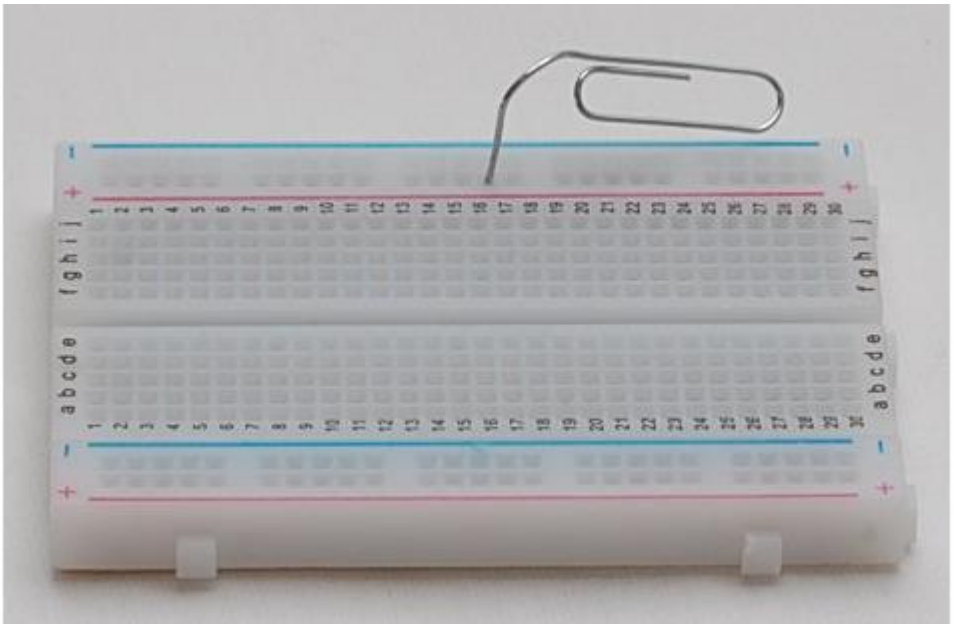
### Совет

На плате Joystick Shield установлен яркий светодиод, который неприятно светит прямо в глаза. Закрасьте его полупрозрачным лаком для ногтей или канцелярской замазкой.

# Приложение 3
# Решение проблем.

## Не получается установить электронный компонент в отверстие на макетной плате

Разработайте отверстие при помощи скрепки или иголки.



## Snap4Arduino не реагирует на нажатие некоторых кнопок на плате Joystick Shield

Иногда по неизвестным причинам среда Snap4Arduino отказывается реагировать на нажатие кнопок на плате с джойстиком. Если это произойдет, то попробуйте выполнить одно из следующих действий:

– отключить Arduino от компьютера и подключить снова;

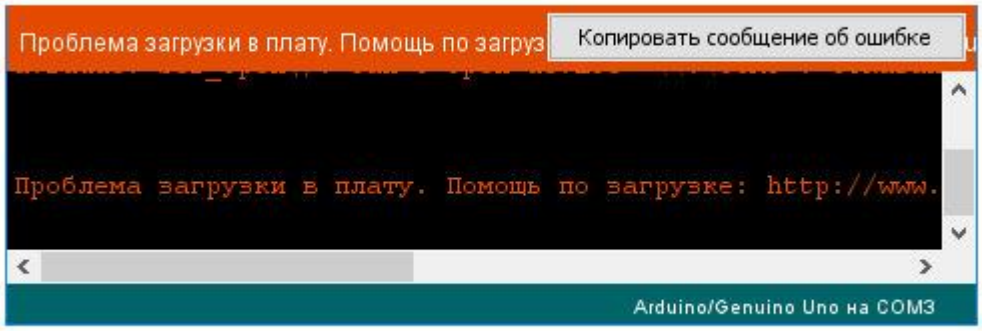
– выключить Snap4Arduino и включить снова;

– снять плату с джойстиком с платы Arduino и установить ее снова.

Если эти действия не помогают, то внесите изменения в скрипт, подключившись к одной из соседних кнопок.

## Скетч StandardFirmata не загружается в плату Arduino

Иногда в процессе загрузки скетча в плату Arduino возникает следующая ошибка.




Полный текст сообщения об ошибке приведен ниже.

*Arduino: 1.8.0 (Windows 10), Плата: «Arduino/Genuino Uno»*

*Скетч использует 11134 байт (34%) памяти устройства. Всего доступно 32256 байт.*

*Глобальные переменные используют 1021 байт (49%) динамической памяти, оставляя 1027 байт для локальных переменных. Максимум: 2048 байт.*

*avrdude: ser_open (): can't open device "\\.\COM3»: Отказано в доступе.*

*Проблема загрузки в плату. Помощь по загрузке: http://www.arduino.cc/en/Guide/Troubleshooting#upload.*

*Этот отчёт будет иметь больше информации с*

*включенной опцией Файл -> Настройки ->*

*«Показать подробный вывод во время компиляции»*

Как можно догадаться из текста сообщения, плате Arduino отказано в доступе к порту COM3.

Для решения проблемы попробуйте выполнить одно из следующих действий:

– закрыть все программы, которые могут использовать COM порт, к которому присоединена плата Arduino, и снова попробовать загрузить скетч;

– перезагрузить компьютер, сразу же запустить Arduino IDE, подключиться к нужному порту и загрузить скетч.

### Проект перестал работать правильно

Иногда проекты перестают корректно работать без видимых причин. В этом случае помогает старый проверенный способ – все выключить и включить снова.

Для решения проблемы попробуйте выполнить следующее.

Выключить Snap4Arduino, отсоединить плату Arduino, а затем снова загрузить Snap4Arduino и присоединить Arduino через USB.

### Проект работает очень медленно

Если у вас не очень современный компьютер, то некоторые проекты с большим количеством спрайтов, клонов и скриптов могут работать медленно.

Для решения проблемы попробуйте выполнить одно из следующих действий:

– убрать блоки ***ждать 0.05***, которые мы добавляли для замедления работы проекта на быстрых компьютерах;

– убрать спрайты, не влияющие на работу программы, например звезды, которые нужны только для украшения проекта;

– не загружать готовые спрайты, а нарисовать их самостоятельно в графическом редакторе;

– уменьшить количество клонов.

# Приложение 4
# Крупнейшие спутники сатурна

| Название | Расстояние до центра Сатурна, км. | Период обращения в сутках | Диаметр, км. | Масса, кг. | Открыт |
|---|---|---|---|---|---|
| Пан | 133 600 | 0,6 | 20 | $4{,}9 \cdot 10^{15}$ | 1981 |
| Дафнис | 136 500 | 0,9 | 7 | $1{,}5 \cdot 10^{14}$ | 2005 |
| Пегги | 137 000 | | 1 | | 2013 |
| Атлас | 137 700 | 0,6 | 32 | $6{,}6 \cdot 10^{15}$ | 1980 |
| Прометей | 139 400 | 0,6 | 100 | $1{,}6 \cdot 10^{17}$ | 1980 |
| Пандора | 141 700 | 0,6 | 84 | $1{,}4 \cdot 10^{17}$ | 1980 |
| Эпиметей | 151 400 | 0,7 | 119 | $5{,}3 \cdot 10^{17}$ | 1980 |
| Янус | 151 500 | 0,7 | 178 | $1{,}9 \cdot 10^{18}$ | 1980 |
| Эгеон | 167 500 | 0,8 | 0,5 | | 2008 |
| Мимас | 185 539 | 0,9 | 397 | $3{,}7 \cdot 10^{19}$ | 1789 |
| Мефона | 194 000 | 1,01 | 3 | $1{,}5 \cdot 10^{13}$ | 2004 |
| Анфа | 197 700 | 1,04 | 1 | | 2007 |
| Паллена | 211 000 | 1,14 | 4 | $3{,}5 \cdot 10^{13}$ | 2004 |
| Энцелад | 238 042 | 1,37 | 499 | $1{,}1 \cdot 10^{20}$ | 1789 |
| Тефия | 294 672 | 1,89 | 1060 | $6{,}2 \cdot 10^{20}$ | 1684 |
| Калипсо | 294 720 | 1,89 | 19 | $3{,}6 \cdot 10^{15}$ | 1980 |
| Телесто | 294 720 | 1,89 | 24 | $7{,}2 \cdot 10^{15}$ | 1980 |
| Полидевк | 377 220 | 2,74 | 4 | $3{,}0 \cdot 10^{13}$ | 2004 |
| Диона | 377 415 | 2,74 | 1118 | $1{,}1 \cdot 10^{21}$ | 1684 |
| Елена | 377 440 | 2,74 | 32 | $2{,}5 \cdot 10^{15}$ | 1980 |
| Рея | 527 068 | 4,52 | 1528 | $2{,}3 \cdot 10^{21}$ | 1672 |
| Титан | 1 221 865 | 15,95 | 5150 | $1{,}3 \cdot 10^{23}$ | 1655 |
| Гиперион | 1 500 933 | 21,28 | 266 | $5{,}7 \cdot 10^{18}$ | 1848 |
| Япет | 3 560 854 | 79,33 | 1436 | $2{,}0 \cdot 10^{21}$ | 1671 |
| Кивиок | 11 111 000 | 449,20 | 16 | $3{,}3 \cdot 10^{16}$ | 2000 |
| Иджирак | 11 124 000 | 451,40 | 12 | $1{,}2 \cdot 10^{15}$ | 2000 |
| Феба | 12 944 300 | 548,20 | 240 | $8{,}3 \cdot 10^{18}$ | 1898 |
| Палиак | 15 200 000 | 686,90 | 22 | $8{,}2 \cdot 10^{15}$ | 2000 |
| Скади | 15 541 000 | 728,20 | 8 | $3{,}1 \cdot 10^{14}$ | 2000 |
| Альбиорикс | 16 182 000 | 783,50 | 32 | $2{,}1 \cdot 10^{16}$ | 2000 |

# Приложение 5
# гравитация на спутниках сатурна

На Титане при падении с высоты 20 метров (высота 6 этажа) скорость падающего предмета будет такой же, как на Земле при падении с высоты 2.65м. Время падения на Титане с высоты 20 метров 5 секунд.

На Рее при падении с высоты 20 метров скорость падающего предмета будет такой же, как на Земле при падении со стула (с высоты 45см.) Время падения на Рее с высоты 20 метров составит целых 12 секунд.

| Название | Масса, кг. | Радиус, км. | Время падения с высоты 20 м, сек. | Скорость в конце падения, м/с | Гравитация слабее земной во столько раз |
|---|---|---|---|---|---|
| Пан | 4,9E+15 | 10 | 116 | 0,3 | 3267 |
| Дафнис | 1,5E+14 | 3,5 | 200 | 0,2 | 9800 |
| Атлас | 6,6E+15 | 16 | 141 | 0,3 | 4900 |
| Прометей | 1,6E+17 | 50 | 100 | 0,4 | 2450 |
| Пандора | 1,4E+17 | 42 | 89 | 0,4 | 1960 |
| Эпиметей | 5,3E+17 | 60 | 63 | 0,6 | 980 |
| Мимас | 3,7E+19 | 199 | 25 | 1,6 | 156 |
| Энцелад | 1,1E+20 | 250 | 18 | 2,2 | 83 |
| Тефия | 6,2E+20 | 530 | 16 | 2,4 | 66 |
| Диона | 1,1E+21 | 559 | 13 | 3,1 | 42 |
| Рея | 2,3E+21 | 764 | 12 | 3,2 | 37 |
| Титан | 1,3E+23 | 2575 | 5 | 7,2 | 7 |
| Гиперион | 5,7E+18 | 133 | 43 | 6,6 | 454 |
| Япет | 2,0E+21 | 718 | 12 | 3,2 | 38 |