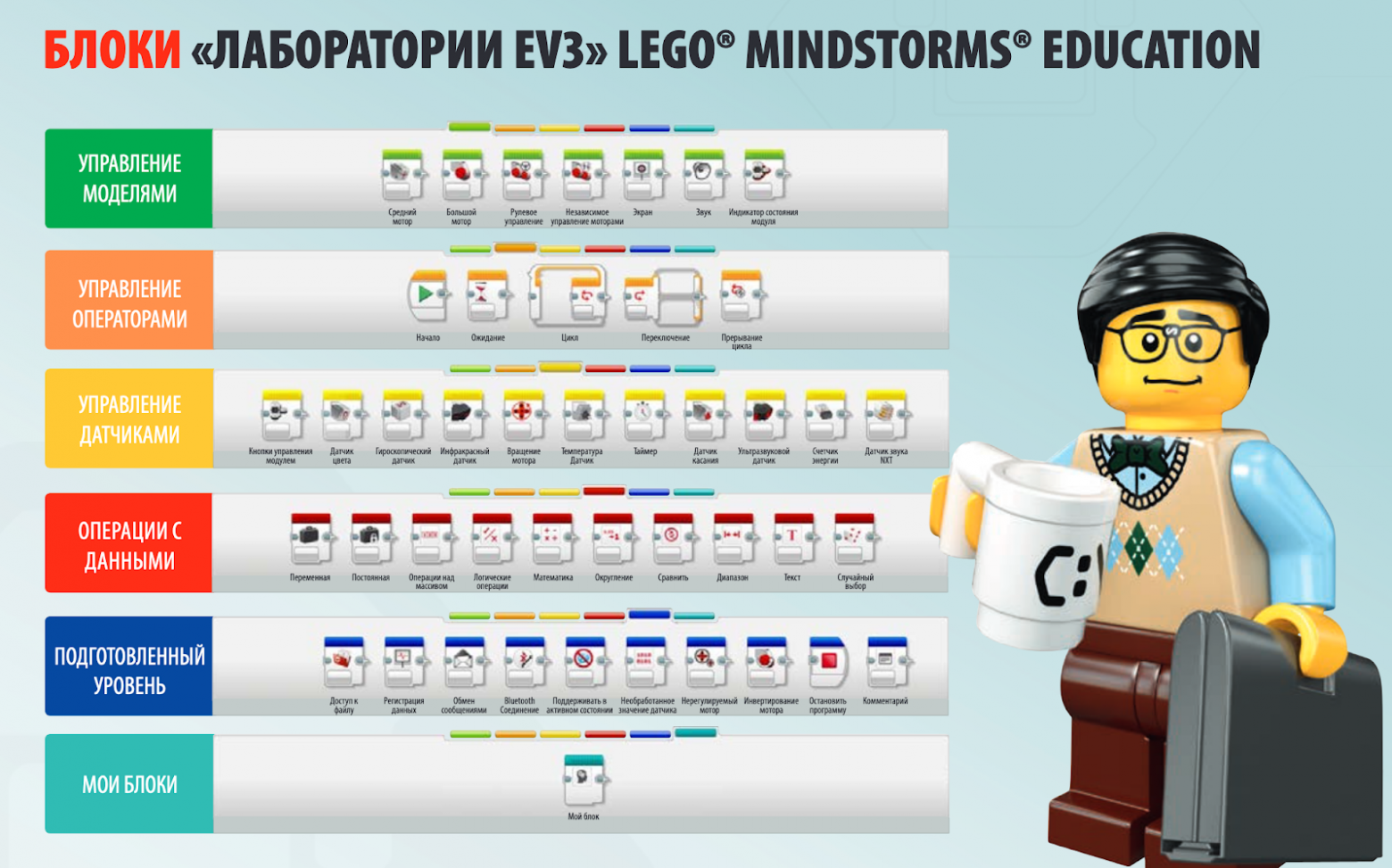
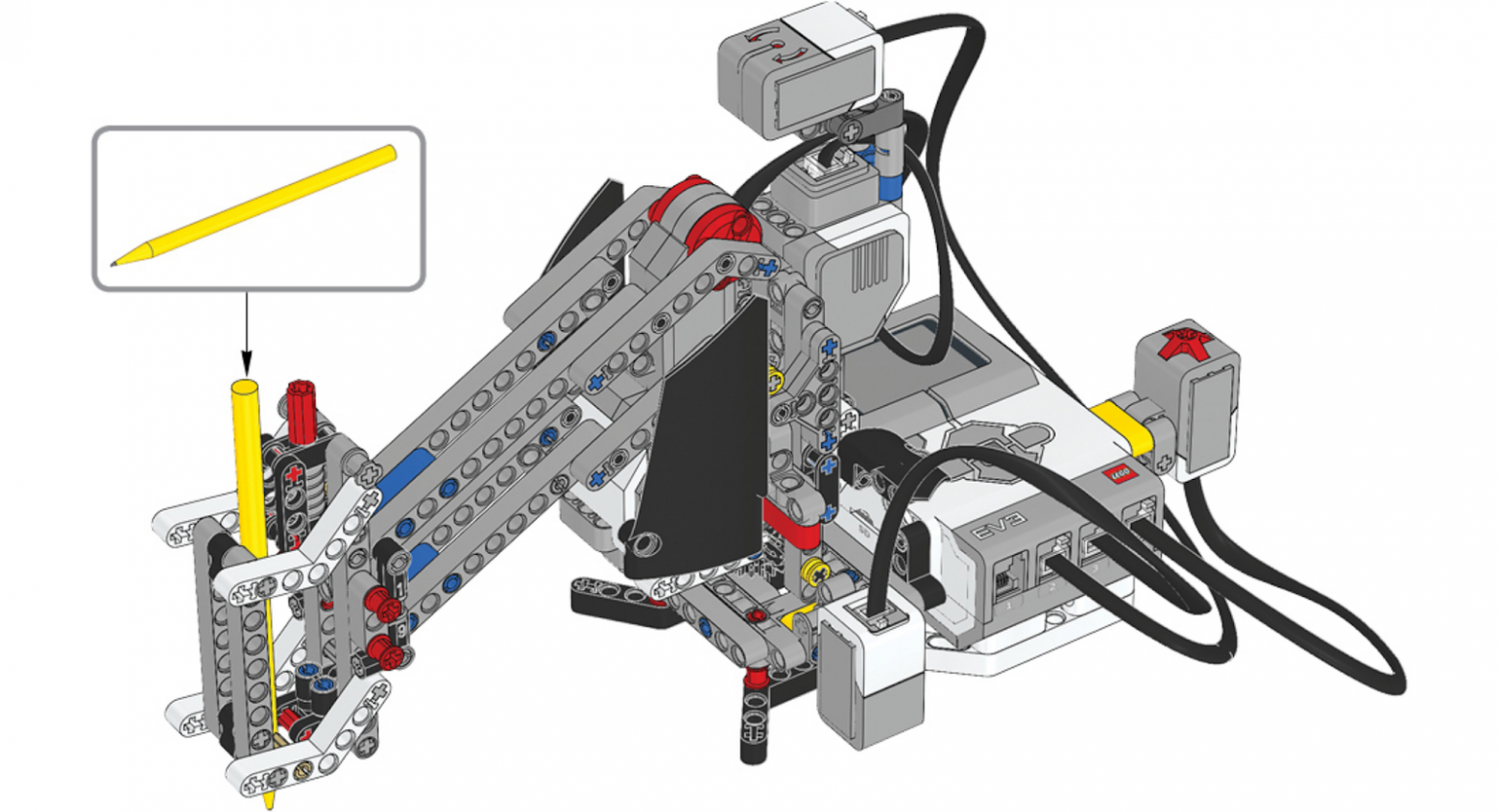
Учим программированию с помощью EV3

Современные школьники хотят видеть красочный результат. Да, им скучно, если программа выводит в консоль числа, и они хотят рассматривать цветные графики, диаграммы и создавать настоящих роботов, движущихся и выполняющих команды. Обычный код тоже кажется детям слишком сложным, поэтому обучение лучше начинать с чего-нибудь полегче.  
  
Базовая среда программирования EV3 создана на основе графического языка LabVIEW и позволяет задавать алгоритмы для робота визуально: команды представлены в виде блоков, которые можно перетаскивать и соединять.  
  
  
  
Такой способ хорошо работает, когда нужно показать, как строятся алгоритмы, но он не подходит для программ с большим количеством блоков. При усложнении сценариев необходимо переходить на программирование с помощью кода, но детям трудно сделать этот шаг.   
  
Здесь есть несколько хитростей, одна из которых — показать, что код выполняет те же задачи, что и блоки. В среде EV3 это можно сделать благодаря интеграции с MicroPython: дети создают одну и ту же программу в базовой среде программирования с помощью блоков и на языке Python в Visual Studio Code от Microsoft. Они видят, что оба способа работают одинаково, но кодом решать сложные задачи удобнее.

Переходим на MicroPython

Среда EV3 построена на базе процессора ARM9, и разработчики специально оставили архитектуру открытой. Это решение позволило накатывать альтернативные прошивки, одной из которых стал образ для работы с MicroPython. Он позволяет использовать Python для программирования EV3, что делает работу с набором еще ближе к задачам из реальной жизни.   
  
Чтобы начать работать, нужно скачать [образ EV3 MicroPython](https://education.lego.com/ru-ru/support/mindstorms-ev3/python-for-ev3) на любую microSD-карту, установить ее в микрокомпьютер EV3 и включить его. Затем нужно установить [бесплатное расширение](https://marketplace.visualstudio.com/items?itemName=lego-education.ev3-micropython) для Visual Studio. И можно приступить к работе. 

Программируем первого робота на MycroPython

  
  
На нашем [сайте](https://education.lego.com/ru-ru/lessons/ev3-cim) есть несколько уроков для освоения базовых понятий робототехники. Модели на EV3 знакомят детей с азами, которые используются в самоуправляемых автомобилях, заводских роботах-сборщиках, станках с ЧПУ.   
  
Мы возьмем для примера чертежную машину, которую можно научить рисовать узоры и геометрические фигуры. Данный кейс является упрощенным вариантом взрослых роботов-сварщиков или фрезеровщиков и показывает, как можно использовать EV3 совместно с MicroPython для обучения школьников. А еще чертежная машина может разметить отверстия в печатной плате для папы, но это уже другой уровень, требующий математических расчетов.  
  
Для работы нам понадобятся:

* базовый набор LEGO MINDSTORMS Education EV3;
* большой лист клетчатой бумаги;
* цветные маркеры.

Сборка самого робота есть в [инструкции](https://le-www-live-s.legocdn.com/sc/media/lessons/mindstorms-ev3/ev3-dep/building%20instructions/pen-arm-bi-30b1246d4a98f95ca6ab254da5487f6c.pdf), а мы рассмотрим пример программирования.  
  
Сначала инициализируем библиотеку модулей EV3:

#!/usr/bin/env pybricks-micropython

**from** pybricks **import** ev3brick **as** brick

**from** pybricks.ev3devices **import** (Motor, TouchSensor, ColorSensor, GyroSensor)

**from** pybricks.parameters **import** Port, Stop, Direction, Color, ImageFile

**from** pybricks.tools **import** wait

Настраиваем платформу, которая вращает ручку как мотор в порте B. Задаем передаточное отношение двухступенчатой зубчатой передачи с количеством зубьев 20-12-28 соответственно.

turntable\_motor = Motor(Port.B, Direction.CLOCKWISE, [20, 12, 28])

Настраиваем подъемный механизм для ручки как мотор в порте C:

seesaw\_motor = Motor(Port.C)

Настраиваем гироскоп, измеряющий угол наклона ручки, в порте 2:

gyro\_sensor = GyroSensor(Port.S2)

Настраиваем цветовой датчик в порте 3. Датчик используется, чтобы определять белую бумагу под чертежной машиной:

color\_sensor = ColorSensor(Port.S3)

Настраиваем датчик касания в порте 4. Робот начинает рисовать, когда датчик нажат:

touch\_sensor = TouchSensor(Port.S4)

Определяем функции, которые поднимают и опускают ручку:

**def** **pen\_holder\_raise**():

  seesaw\_motor.run\_target(50, 25, Stop.HOLD)

  wait(1000)

**def** **pen\_holder\_lower**():

  seesaw\_motor.run\_target(50, 0, Stop.HOLD)

  wait(1000)

Определяем функцию для поворота ручки на заданный угол или до определенного угла:

**def** **pen\_holder\_turn\_to**(target\_angle):

**if** target\_angle > gyro\_sensor.angle():

Если целевой угол больше, чем текущий угол гироскопического датчика, продолжаем движение по часовой стрелке с положительной скоростью:

    turntable\_motor.run(70)

**while** gyro\_sensor.angle() < target\_angle:

**pass**

**elif** target\_angle < gyro\_sensor.angle():

Если целевой угол меньше, чем текущий гироскопического датчика, то двигаемся против часовой стрелки:

    turntable\_motor.run(-70)

**while** gyro\_sensor.angle() > target\_angle:

**pass**

Останавливаем вращающуюся платформу, когда целевой угол будет достигнут:

  turntable\_motor.stop(Stop.BRAKE)

Устанавливаем начальное положение ручки в верхнем положении:

pen\_holder\_raise()

Теперь идет основная часть программы — бесконечный цикл. Сначала EV3 ожидает, когда датчик цвета обнаружит белую бумагу или синюю стартовую клетку, а датчик касания будет нажат. Затем он рисует узор, возвращается в исходное положение и повторяет все заново.  
  
Когда устройство не готово, светодиоды на контроллере принимают красный цвет, и на ЖК-экране отображается изображение «палец вниз»:

**while** True:

  brick.light(Color.RED)

  brick.display.image(ImageFile.THUMBS\_DOWN)

Дожидаемся, когда датчик цвета считает синий или белый цвет, устанавливаем цвет светодиодов зеленым, отображаем на ЖК-экране изображение «палец вверх» и сообщаем, что устройство готово к работе:

**while** color\_sensor.color() **not** **in** (Color.BLUE, Color.WHITE):

wait(10)

  brick.light(Color.GREEN)

  brick.display.image(ImageFile.THUMBS\_UP)

Дожидаемся нажатия датчика касания, присваиваем гироскопическому датчику значение угла 0 и начинаем рисовать:

**while** **not** touch\_sensor.pressed():

wait(10)

  gyro\_sensor.reset\_angle(0)

  pen\_holder\_turn\_to(15)

  pen\_holder\_lower()

  pen\_holder\_turn\_to(30)

  pen\_holder\_raise()

  pen\_holder\_turn\_to(45)

  pen\_holder\_lower()

  pen\_holder\_turn\_to(60)

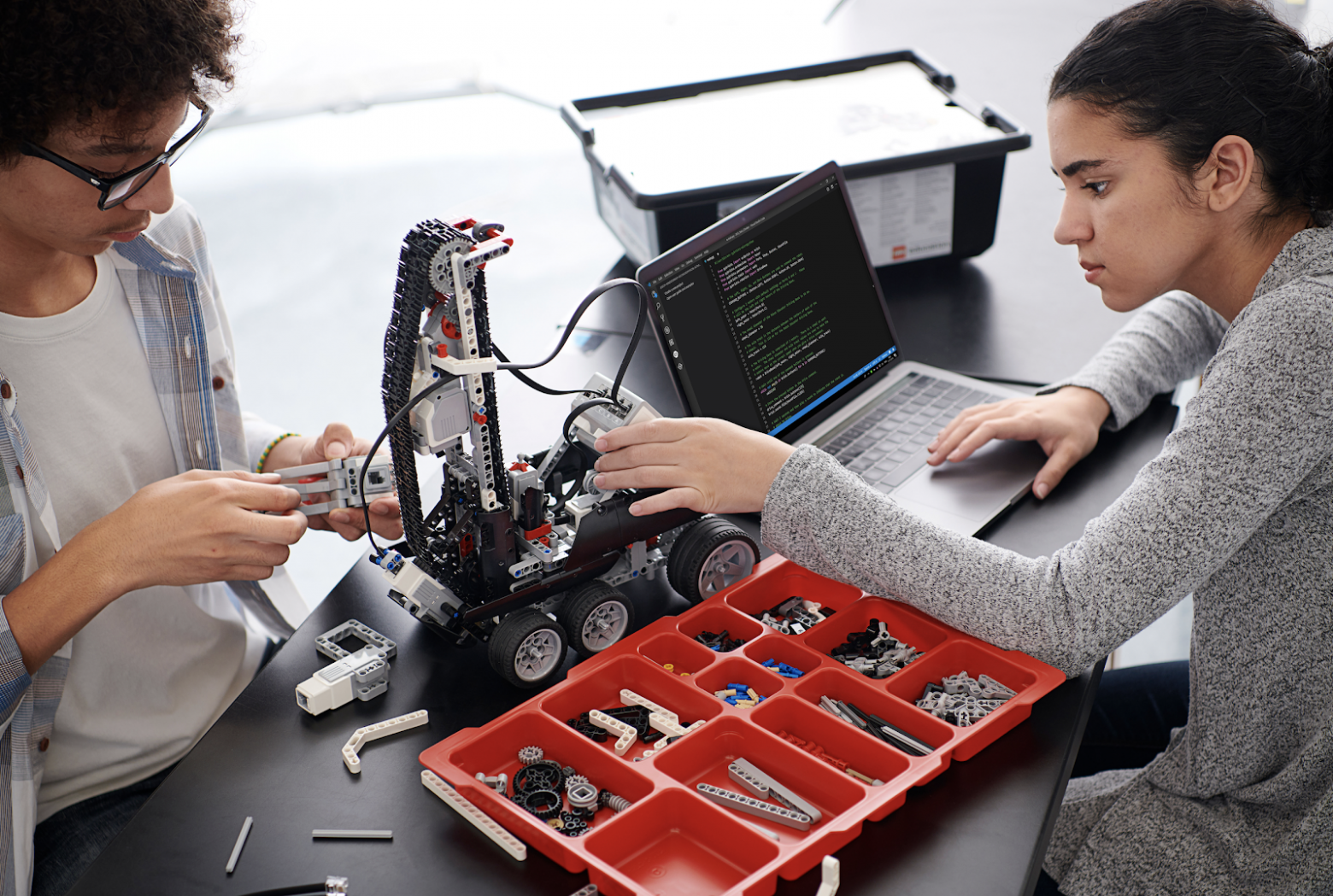
Поднимаем держатель ручки и возвращаем его в исходное положение:

  pen\_holder\_raise()

  pen\_holder\_turn\_to(0)</i>

Вот такая несложная программа у нас получилась. И теперь запускаем ее и смотрим на робота-чертежника в деле. 

Что дают такие примеры

  
  
EV3 — это инструмент для профориентации в рамках профессий STEM и точка входа в инженерные специальности. Так как на нем можно решать практические задачи, дети получают опыт технических разработок и создания промышленных роботов, учатся моделировать реальные ситуации, понимать программы и анализировать алгоритмы, осваивают базовые конструкции программирования.  
  
Поддержка MicroPython делает платформу EV3 подходящей для обучения в старших классах. Ученики могут попробовать себя в роли программистов на одном из самых популярных современных языков, познакомиться с профессиями, связанными с программированием и инженерным проектированием. Наборы EV3 показывают, что писать код — это не страшно, готовят к серьезным инженерным задачам и помогают сделать первый шаг к освоению технических специальностей. А для тех, кто работает в школе и связан с образованием, у нас подготовлены [программы занятий](https://education.lego.com/ru-ru/support/mindstorms-ev3/coding-activities) и учебные материалы. В них детально расписано, какие навыки формируются при выполнении тех или иных задач, и как полученные навыки соотносятся со стандартами обучения.