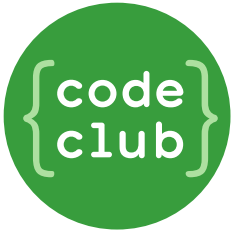
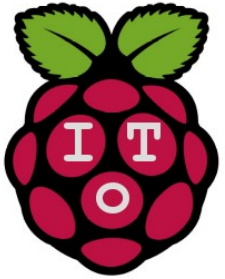


Internet of Things

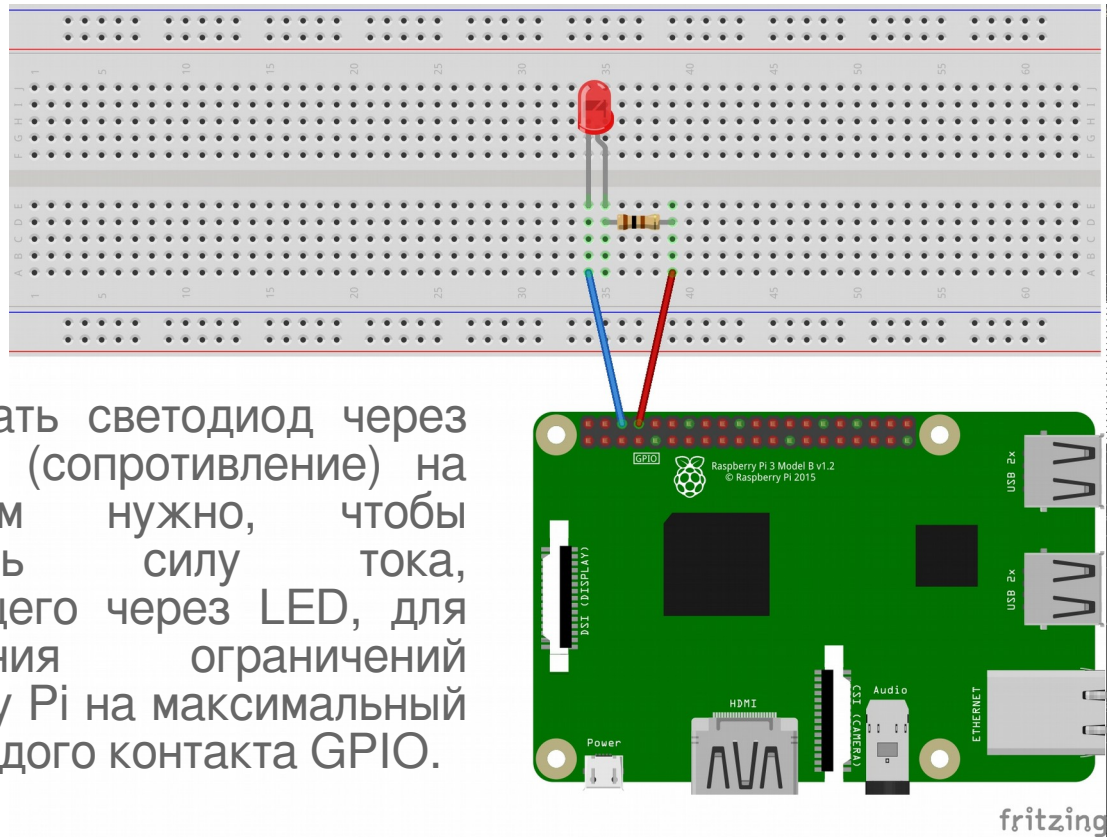


Язык программирования **Ruby** в проектах **IoT**

Шадринск
2018-2019

М. В. Шохирев

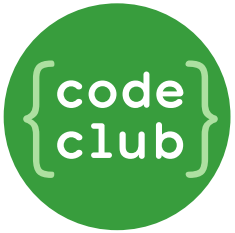
Светодиод (LED): подключение



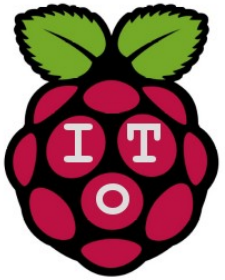
Подключать светодиод через резистор (сопротивление) на 330 Ом нужно, чтобы уменьшить силу тока, проходящего через LED, для соблюдения ограничений Raspberry Pi на максимальный ток с каждого контакта GPIO.

Короткую ножку **светодиода** (LED) подключить (обычно синим или чёрным проводом) к контакту [6] «земля» (GND) на GPIO у Raspberry Pi.

Длинную ножку **светодиода** подключить (любым цветным проводом) к резистору на 330 Ом, который подключить к соседнему физическому контакту [8] на GPIO у Raspberry Pi, имеющему логический номер BCM (25).



LED: запуск программы на языке Ruby



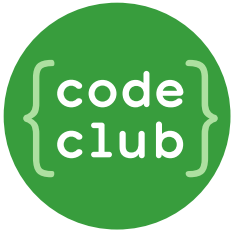
Теперь мы знаем, что можно управлять физическими объектами, считывая данные (показания датчиков) из определённых системных файлов и записывая в них значения (посылая сигналы включения и выключения).

Но записывать эти действия в командных файлах на **bash** громоздко и неудобно: из-за его ограниченных возможностей и необходимости подробно описывать все действия.

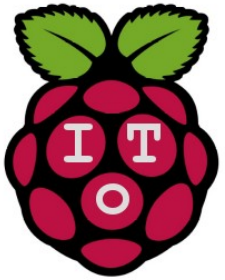
Поэтому программы управления обычно пишут на языках высокого уровня, где есть готовые библиотеки для взаимодействия с различными физическими устройствами. Мы научимся писать программы на языке **Ruby**.

Для примера можно в терминальном окне запустить программу управления светодиодом на языке **Ruby**:

```
cd ~/projects
./test_led.rb
```

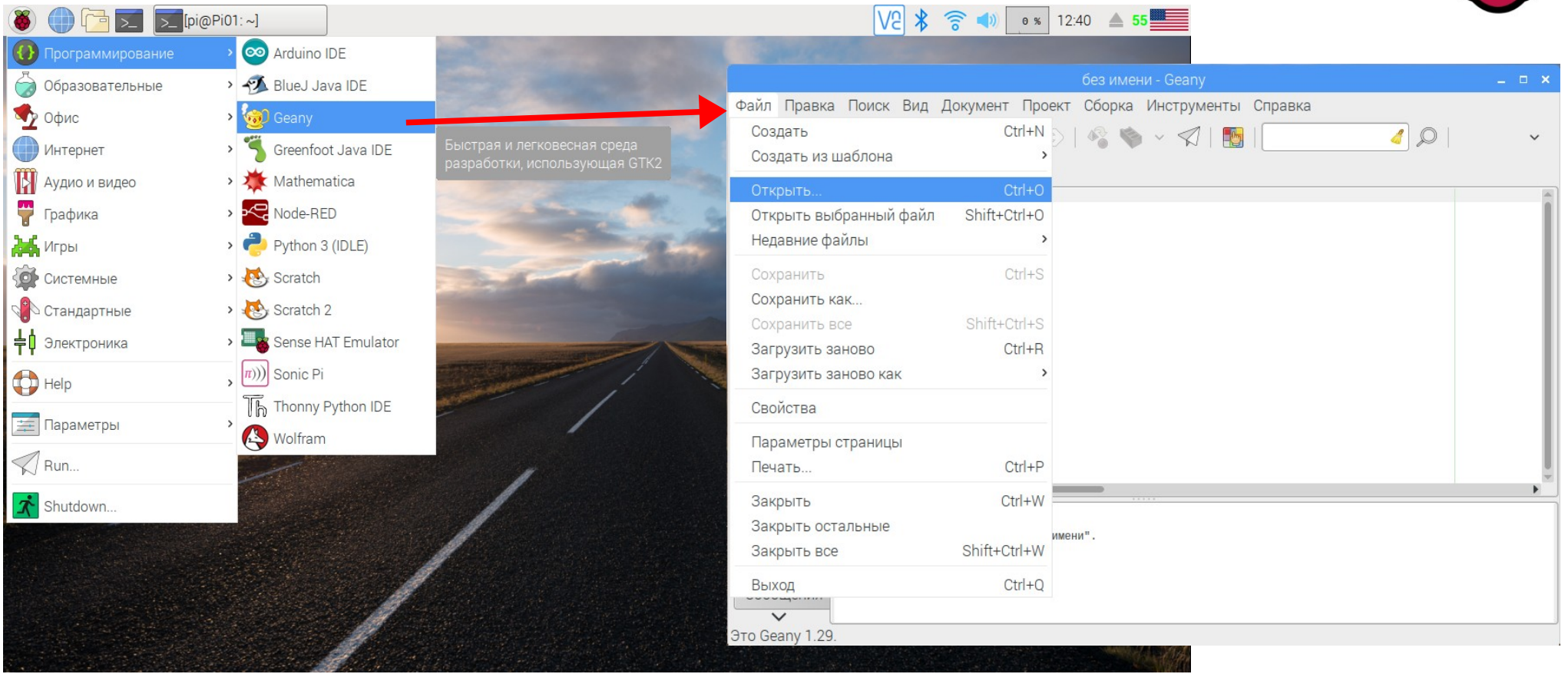


LED: управление из программы на Ruby

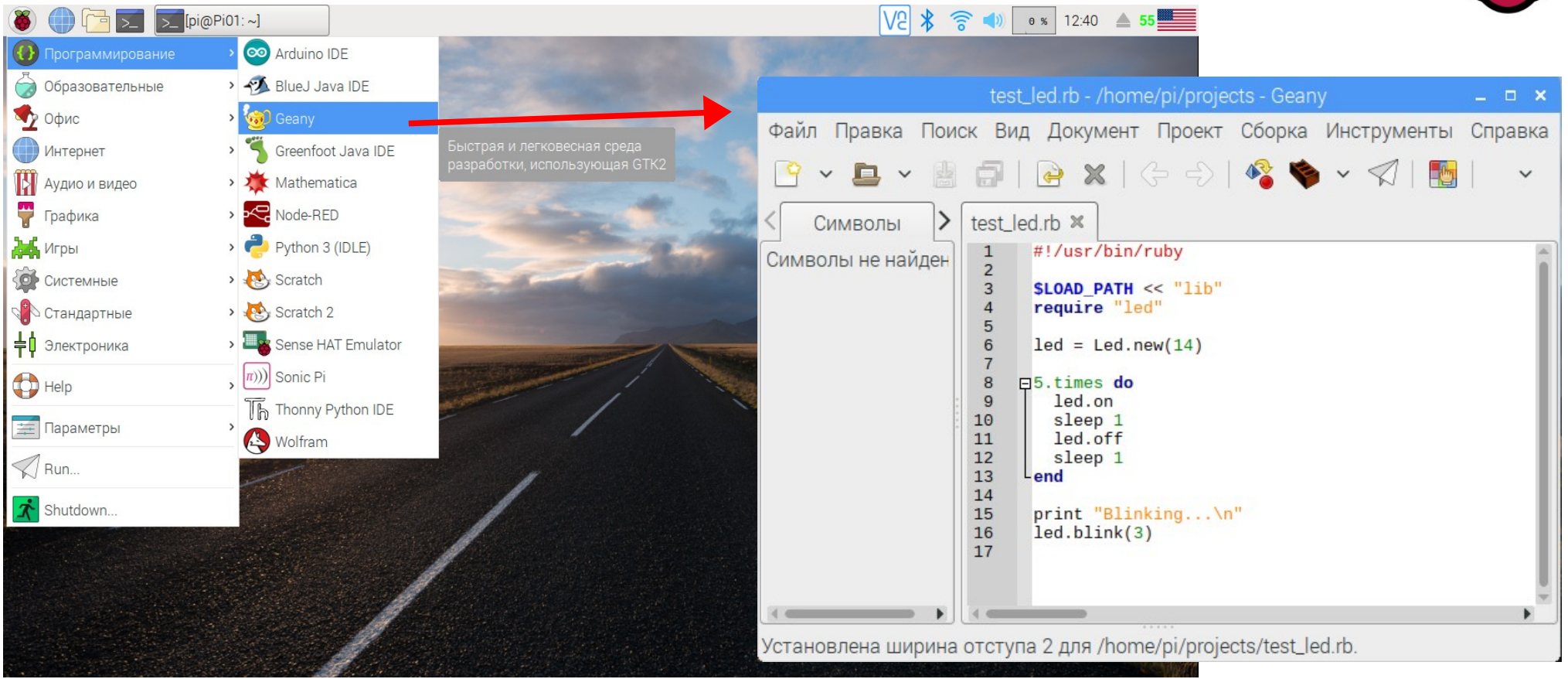



```
#!/usr/bin/ruby    # указать путь к интерпретатору ruby

require "led"      # подключить библиотеку led.rb

led = LED.new(14) # создать объект led класса Led (Светодиод)
5.times do        # 5 раз повторить команды от do до end
  led.on()        # скомандовать объекту led включиться (on)
  sleep 1         # приостановить выполнение на 1 секунду
  led.off()       # скомандовать объекту led выключиться (off)
  sleep 1         # приостановить выполнение на 1 секунду
end

led.blink(3)      # скомандовать объекту led помигать (blink) 3 раза

# Запуск в терминальном окне: ./test_led.rb
```

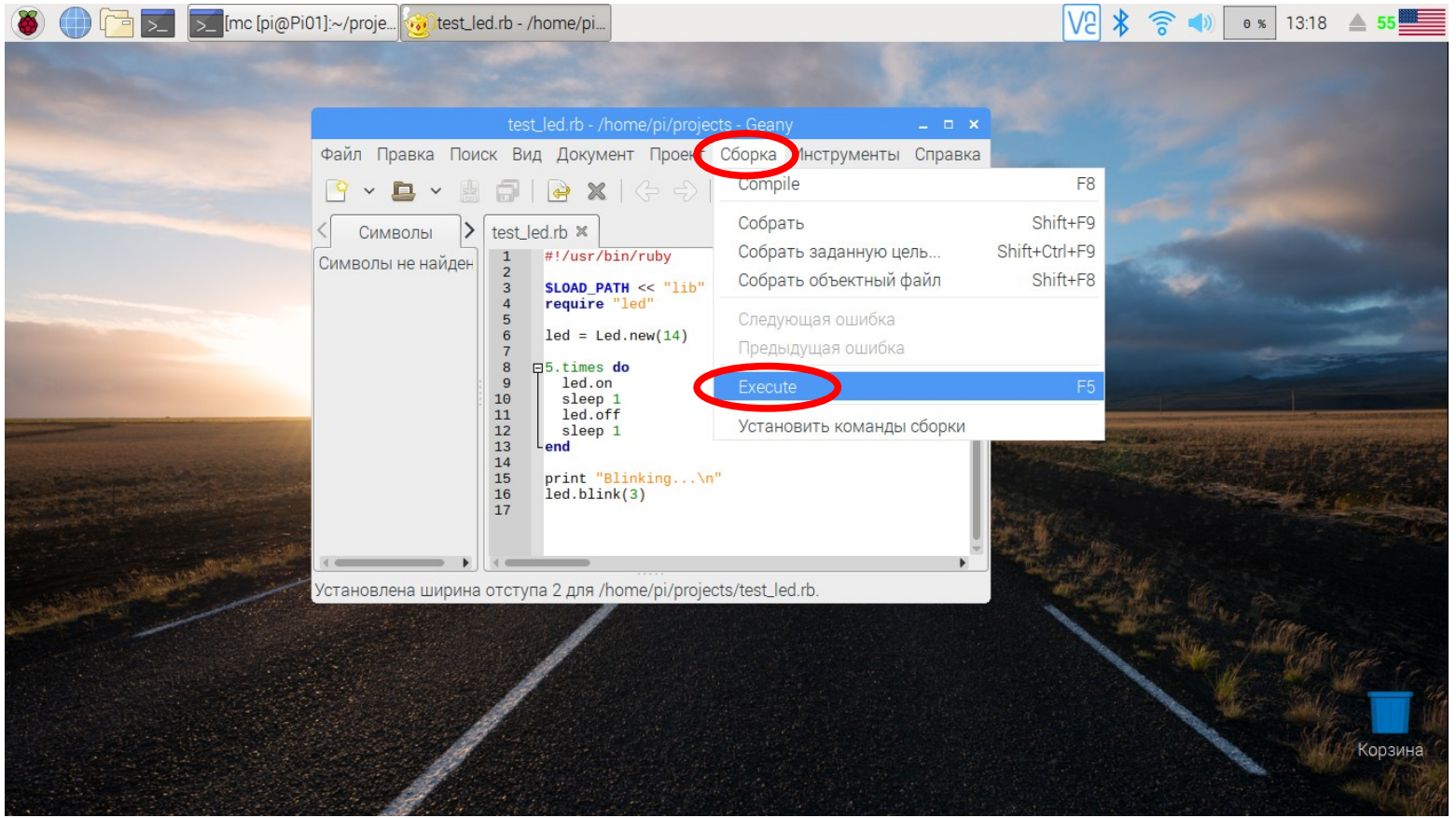
Среда разработки Geany



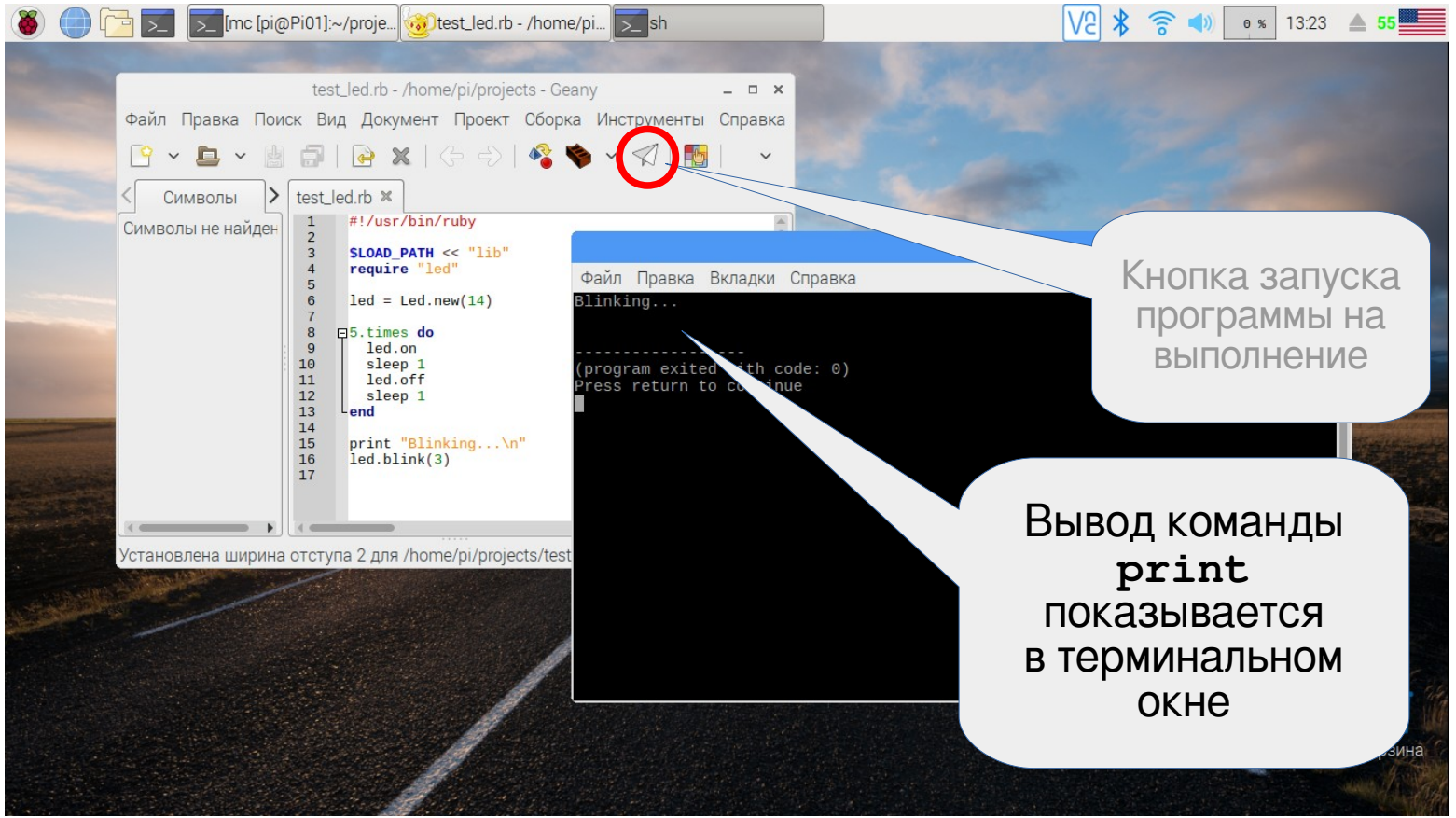
Текст программы на Ruby в Geany



Запуск программы на Ruby из Geany



Выполнение программы на Ruby из Geany



Классы и объекты: модель реальности

Ruby – объектно-ориентированный язык, который позволяет в программах моделировать объекты реального мира с их характерными свойствами и уникальным состоянием, а также взаимодействие между ними.

При разработке программы создаётся программная модель:

- предметы и понятия реальности описываются как **классы**, в которых
- описываются **объекты** внешнего мира (светодиоды, кнопки, датчики, реле, моторы),
- их **свойства** (название, цвет, контакты, показания, состояние) и
- возможные **действия** с ними (включить, выключить, считать показания, изменить состояние, увеличить скорость).

Классы и объекты: термины

Реалии	Объяснение	Термины ООП	Описание на Ruby

Несколько однотипных светодиодов (LED) разных цветов 	Похожие объекты с одинаковыми характеристиками	*Класс* «Светодиод»	`class LED`
			`  # описания`
			`end`

Цвет, № контакта, рабочее напряжение, потребляемый ток, ...	Характеристики (свойства) этих объектов	*Атрибуты* в описании класса	`attr_reader :pin`
			`attr_reader :color`
			`# другие атрибуты`

Светодиоды можно включать и выключать	Их поведение заключается в переходах от состояния «включено» к «выключено» и наоборот	Объекты класса имеют *методы* on() и off()	`def on()# включить`
			`end`
			`def off()# выключить`
			`end`

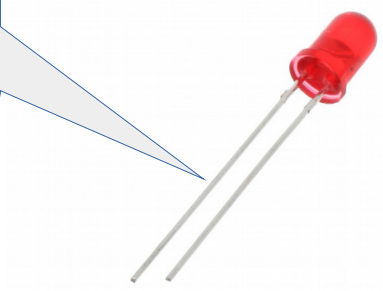
Красный светодиод на пине 23 сначала включить, а потом выключить	Перевести светодиод в соответствующее состояние	Изменить *состояние* с помощью определённого метода	`r = LED.new(23,:red)`
			`r.on()# включить`
			`r.off()# включить`

Классы и объекты: моделирование

Во время выполнения программы

- создаются объекты классов методом **`new`**: **`led = LED.new(18,:red)`**
- над объектами выполняются действия с помощью *методов*: **`led.on()`**
- и у них изменяется *состояние* (то есть значения свойств): **`@state = :on`**
- которое можно выяснить: **`print "включен" if led.is_on?`**

Физический объект в реальном мире

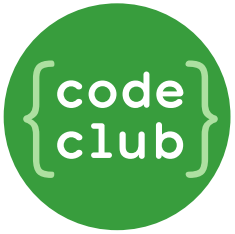


LED

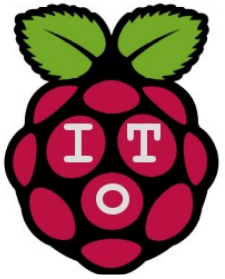
`@pin = 18`
`@color = :red`
`@state = :on`

`on()`
`off()`
`is_on?()`
`is_off?()`

Информационный объект в программной модели

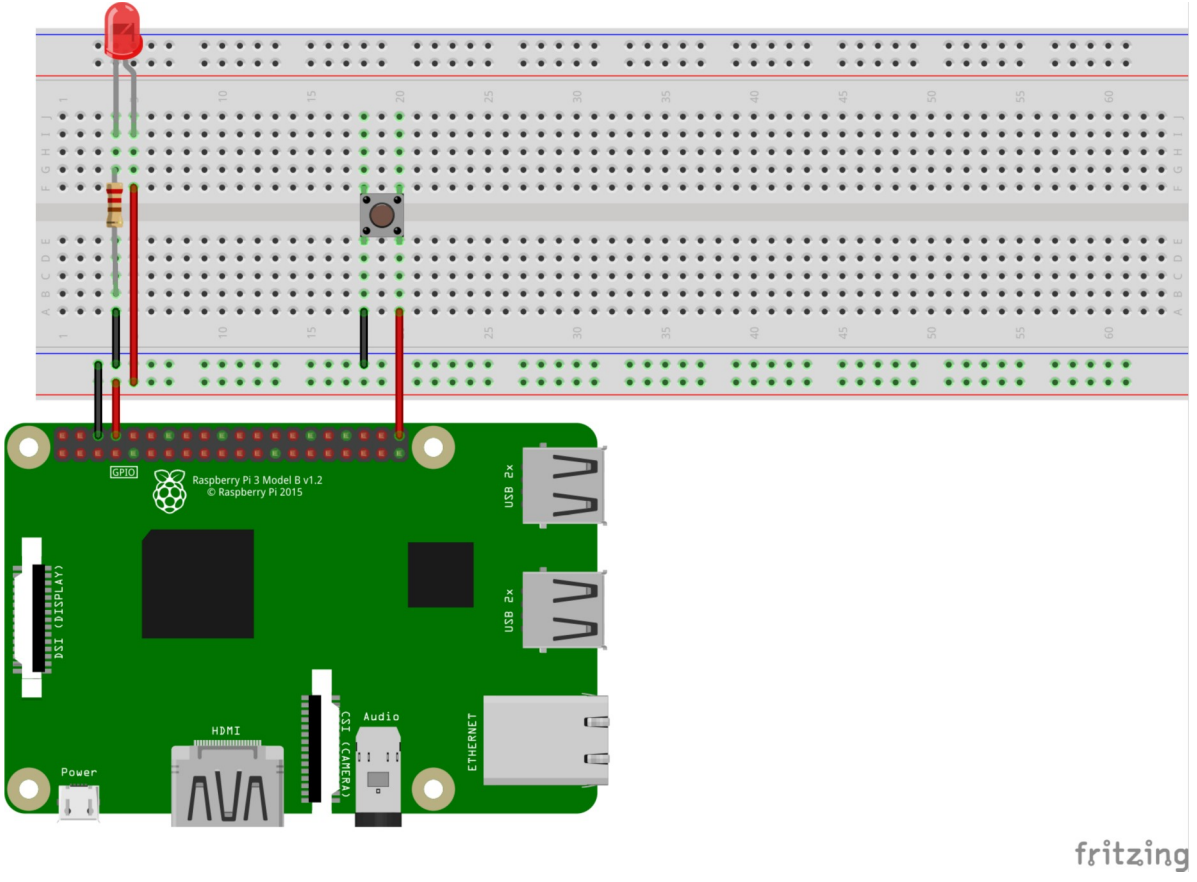


Классы и объекты: пример




```
class LED                  # класс объектов: светодиоды
  attr_reader :pin         # атрибут - свойство объекта @pin: № контакта
  attr_reader :color       # атрибут - свойство объекта @color: цвет
  attr_reader :state       # атрибут - свойство объекта @state: состояние

  def on                   # метод - действие с объектом: включить
    @state = :on           # установить значение свойства объекта
  end
  def is_on?               # действие с объектом: проверить состояние
    return @state==:on     # вернуть значение свойства @state объекта
  end
end

red = LED.new(18)          # создать объект red класса Led с параметром 17
red.on                     # скомандовать объекту red: «Включись!»
print(red.is_on?)          # спросить у объекта его состояние методом is_on?

blue = LED.new(22)         # создать ещё объект класса Led — с именем blue
print(blue.is_on?)         # спросить у объекта его состояние
```

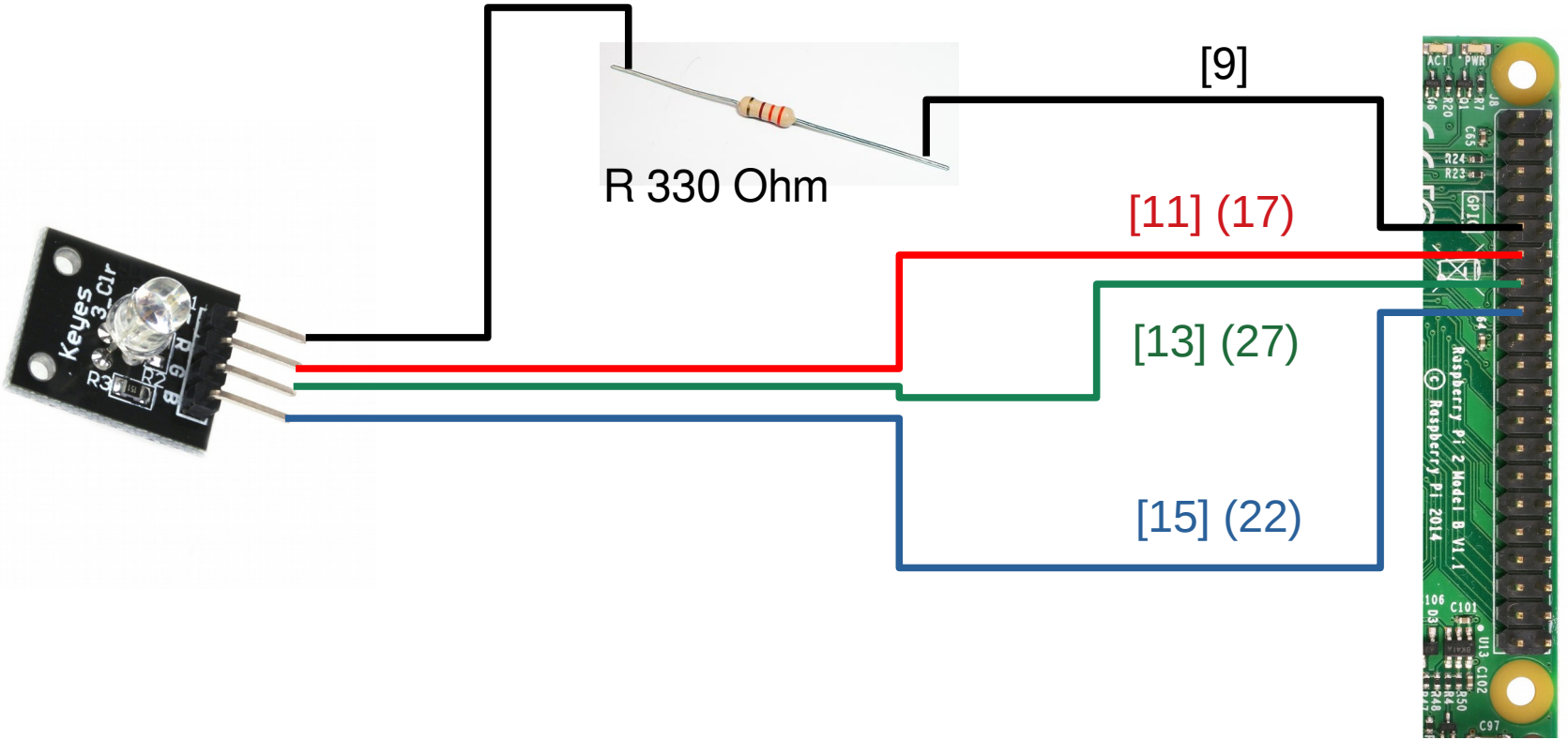
Кнопка:
пример рецептора



Кнопка: управление на языке Ruby

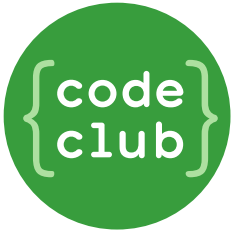
```
#!/usr/bin/ruby
require "button"                    # подключить библиотеку button.rb

button = Button.new(21)             # создать объект класса Button

button.wait_for_press               # ожидать нажатия на кнопку
if  button.long_press?              # было длинное нажатие?
  print "Длинное!\n"
else
  print "Короткое.\n"
end

button.wait_for_presses(2)          # ожидать 2 нажатия на кнопку
if button.double_press?             # было двойное нажатие?
  print "Двойное...\n"
else                                # было одиночное нажатие
  print "2 одинарных\n"
end
```

Светодиод RGB: подключение

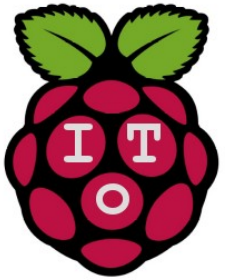


RGB: управление из программы на Ruby

```
#!/usr/bin/ruby

require "led_rgb"    # подключить библиотеку lib/led.rb

rgb = LED::RGB.new(17,27,22) # объект rgb класса LedRgb
3.times do           # 3 раза повторить команды от do до end
  [:red,:green,:blue,:yellow,:aqua].each do |color|
    rgb.on(color)    # скомандовать rgb включиться очередным цветом
    sleep 1          # приостановить выполнение на 1 секунду
    rgb.off          # скомандовать rgb выключиться
    sleep 1          # приостановить выполнение на 1 секунду
  end
end

rgb.blink(:red, 5)   # скомандовать rgb помигать красным 5 раз
```



На каком языке легче программировать?



Что вам не понятно
из изученного материала?