Вступление

Мне попал в руки на тестирование робототехнический набор Makeblock mBot Educational Robot Kit, благодарю за это Алексея Филимонова. Робот уже был собран, поэтому мне очень интересно было изучить, как он программируется, чтобы прикинуть как с ним можно выстроить учебные занятия по образовательной робототехнике для школьников.

Прежде всего рассмотрим его характеристики:

Среда программирования	Arduino IDE, mBlock (Mac OS, Windows, Linux*), mBlock APP (Android, IOS)
Устройства ввода	Датчик света (Light sensor), кнопка (button), инфракрасный приемник (infrared receiver), ультразвуковой дальномер (ultrasonic sensor), датчик линии (line follower), датчик освещенности
Устройства вывода	Пьезопищалка (buzzer), RGB-светодиоды (RGB LED), инфракрасный передатчик (infrared transmitter), 2 порта для моторов (two motor ports)
Микроконтроллер	ATmega328, Arduino UNO-совместимый
Питание	4 AA батарейки по 1,5 Вольт
Беспроводная связь	Bluetooth
Размеры	17 x 13 x 9 см
Масса	500 грамм

Поскольку дома работаю на Ubuntu Linux, то первым делом начал установку mBlock, скачав с сайта http://www.mblock.cc/download/ установочный deb-пакет. Кстати, на этом сайте можно

скачать версии не только для Linux, но и для других операционных систем: Windows, Mac OS, Android, IOS. Установить среду под Linux получилось, но вот загрузить в робота простую тестовую программу сходу не удалось.

Ну, хорошо, попробуем программировать робота в Arduino IDE. Обзор информации и готовых проектов на ресурсах, посвященных Makeblock, привел к пониманию, что для упрощения программирования требуется специальная библиотека, и такая библиотека на просторах интернета была найдена. Приятно, что Makeblock придерживается философии открытых исходных кодов (OpenSource)!

Набор MakeBlock mBot укомплектован программным обеспечением (ПО), которое относится к классу OpenSource. Аппаратная часть наборов MakeBlock относится к классу OpenHardwareSource.
Для образовательного процесса это имеет явные преимущества перед проприетарными программами и продуктами:

1) Доступ к исходным текстам открывает перед учащимся перспективу углубленного изучения программирования. Проприетарное ПО, созданное под девизом «используй как есть», закрывает возможность изучения кода и даже запрещает это законодательно.
2) Если педагог планирует задавать домашние задания с использованием ПО, установленного на домашних компьютерах детей, то свободное программное обеспечение (в данном случае оно бесплатное!) позволяет избежать излишних трат со стороны родителей на ПО, необходимое только для учебных занятиях в ограниченный период времени.
3) Открытые лицензии, такие как GPL (General Public License), специально разработаны для защиты клиента. Они гарантируют, что вы имеете право использовать программное обеспечение так, как вам нужно и без каких-либо законодательных ограничений.
4) OpenSourceHardware дает возможность углубленного изучения схемотехники в перспективе на основе схемы контроллера.
5) Вокруг продуктов OpenSource в следствие их открытости образуются многочисленные интернет-сообщества разработчиков и пользователей открытого кода, где вы получите новый опыт, ценные советы, передовые методы и идеи от других участников. Есть такие интернет-сообщества и вокруг продуктов MakeBlock.

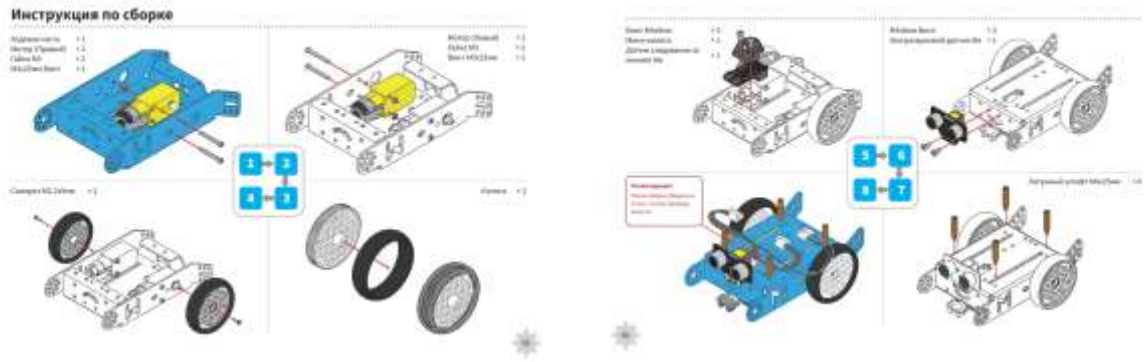
В данной книге практически в каждой главе, рассматриваются примеры программирования в двух средах параллельно, то есть, материал специально составлен так, чтобы одни и те же программы для робота вы могли составить как в среде mBlock, так и написать в Arduino IDE. Конечно, между такими программами будут незначительные различия, но зато такой подход позволит вам познакомиться и с введением в программирование для начинающих в «пазлоподобной», визуальной среде mBlock, где очень сложно ошибиться, и «прокачать скиллсы» начинающего программиста в «продвинутом» Си-подобном языке программирования (более сложном, но и более мощном) в среде Arduino IDE. И при этом мы будем использовать одного и того же робота mBot!

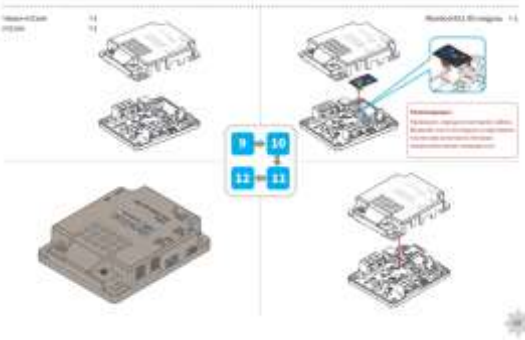
Сборка робота mBot

Если ваш робот еще не собран, то собрать его не сложно. В коробке вложена хорошо иллюстрированная инструкция, руководствуясь которой можно осуществить сборку.

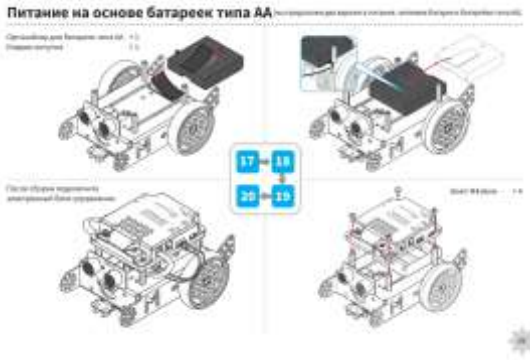
Из инструментов вам потребуется универсальная отвертка, у которой двухсторонний наконечник: с одного конца он шестигранный, с другой — крестовой. Наконечник можно вытащить из ручки, перевернуть и вставить в ручку. Таким образом универсальной отверткой можно закручивать винты с шестигранными головками (например, ими крепятся датчики, корпус контроллера mCore) и винты с крестовыми шляпками (ими крепятся моторы и колеса).

Соберите робота по инструкции из 20 шагов, описанных с 4 страницы по 8.






Питание на основе литиевой батареи

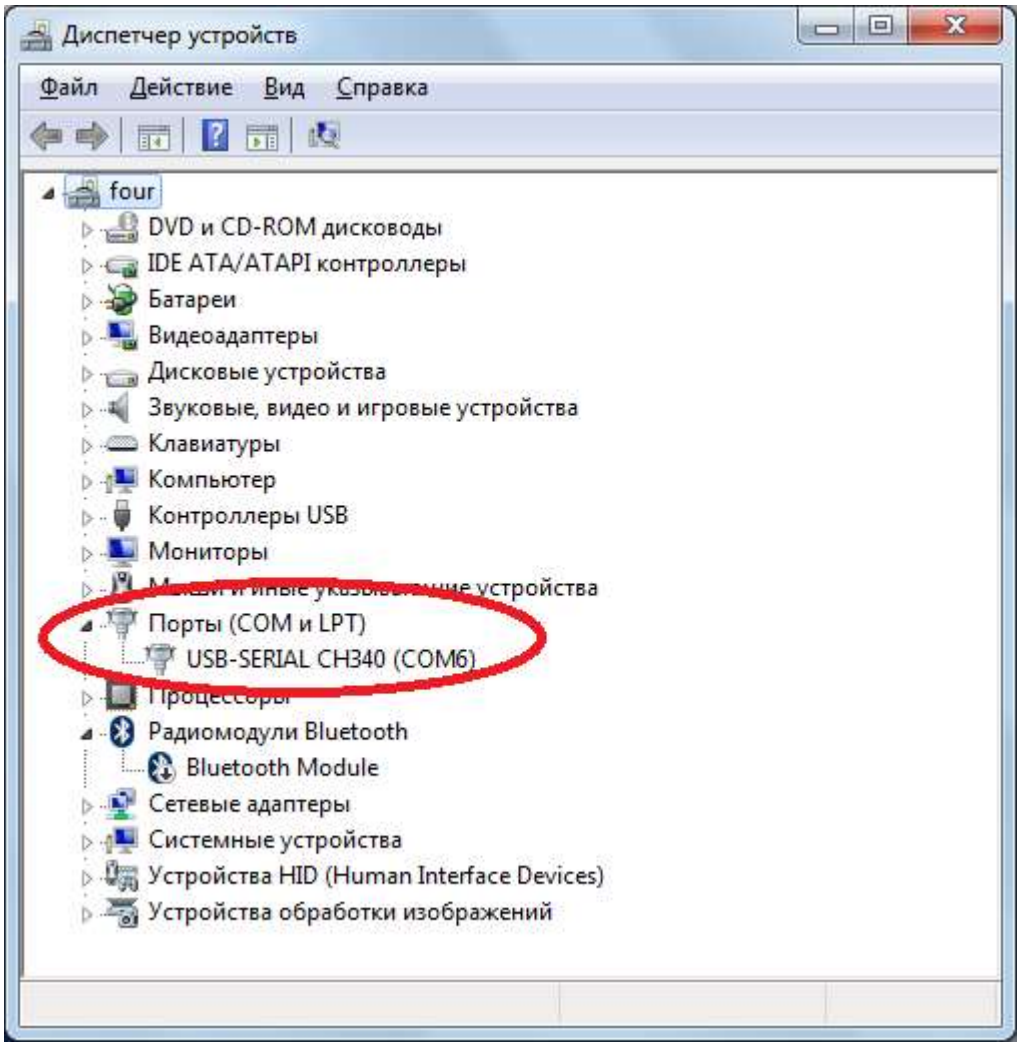

Питание на основе батареек типа AA


MakeBlock

Подключение mBot к персональному компьютеру

К собранному роботу mBot подсоедините блок питания из четырех батареек AA, которые в быту называют «пальчиковыми», или литий-ионный аккумулятор. На мой взгляд практичнее использовать «пальчиковые» никель-магнивые аккумуляторы AA на 1,2 Вольт, так как их можно многократно заряжать, и они считаются безопаснее литий-ионных аккумуляторов, которые при повреждении оболочки могут воспламениться.

Подсоедините mBot к персональному компьютеру (ПК) кабелем USB и включите Power switch в положение ON. Если робот собран правильно и блок питания выдает достаточное напряжение, то на роботе загорятся светодиоды, а на компьютере скорее всего появятся сообщения о подключении нового устройства и установке драйверов.
Если вы работаете на ПК под управлением Windows, то в «Диспетчере устройств» в разделе «Порты (COM и LPT)» появится устройство «USB-SERIAL CH340 (COM6)». Нас интересует номер COM порта, который написан в скобках. Скорее всего на вашем ПК название такого устройства будет отличатся номером в скобках. Запомните этот номер COM-порта, который появился на вашем ПК, его в дальнейшем нужно будет использовать для программирования робота mBot.



Резюме:

1. Подсоедините блок питания к mBot.

2. Подсоедините кабелем USB робота mBot к ПК
3. Включите робота mBot (включатель на правом борту)
4. Определите номер COM-порта «USB-SERIAL CH340» в «Диспетчере устройств». Возможно, устройство автоматически не распознается и на компьютере потребуется вручную установить драйвера. Как это сделать, будет рассказано в следующей главе.

Настройки для программирования в mBlock

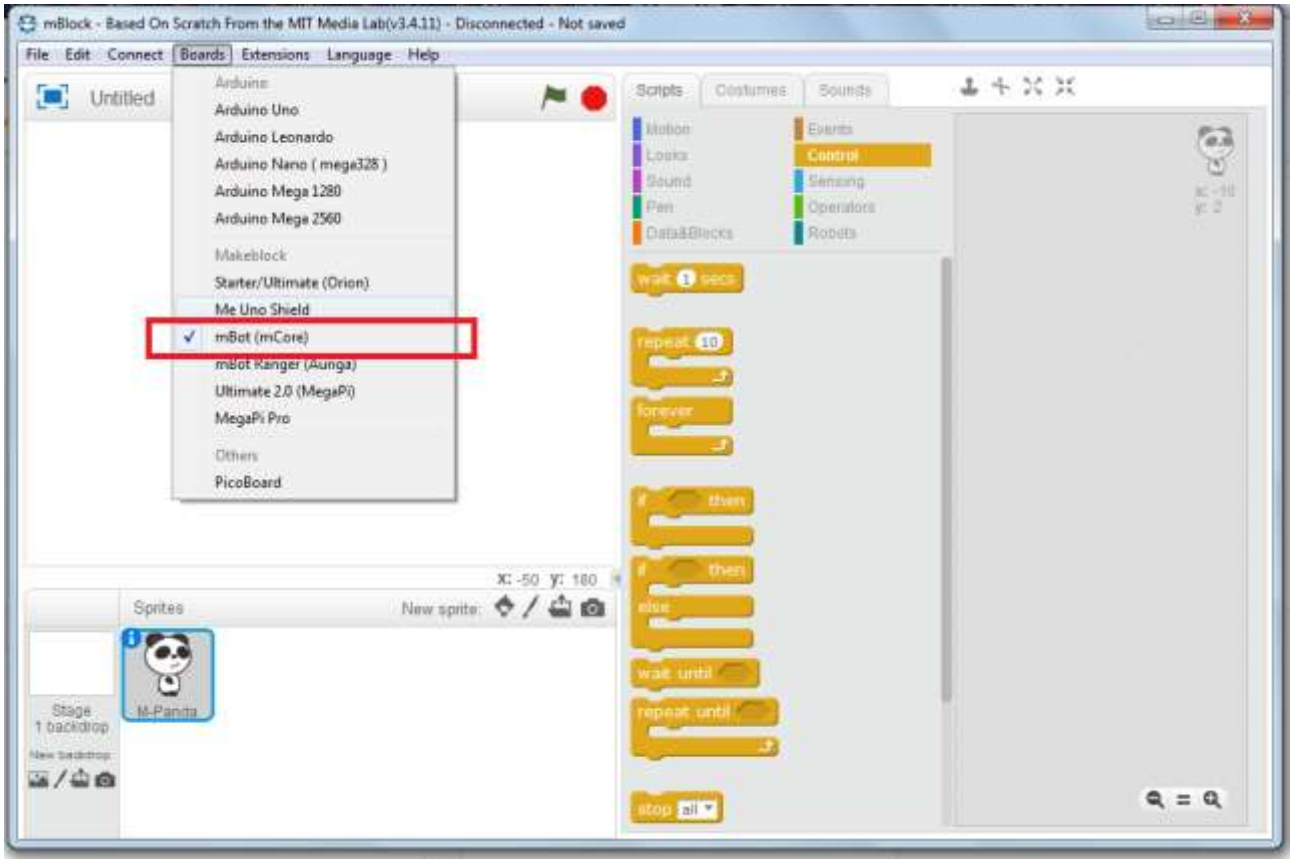
Для программирования mBot сначала нужно скачать среду программирования mBlock. В интернете на сайте http://www.mblock.cc/download/ нужно выбрать версию mBlock, соответствующую для операционной системы на вашем персональном компьютере. Например, для Windows нужно выполнить следующие шаги:

1. Открыть в браузере сайт http://www.mblock.cc/download/
2. Выбрать ссылку «mblock for PC»
3. Выбрать версию mBlock 3 (рекомендуемая). В настоящий момент доступно две версии: «mBlock 3» и «mBlock 5», - но вторая еще на стадии разработки.
4. Скачать версию для вашей операционной системы, например, «Windows 7 and above V3.4.11»

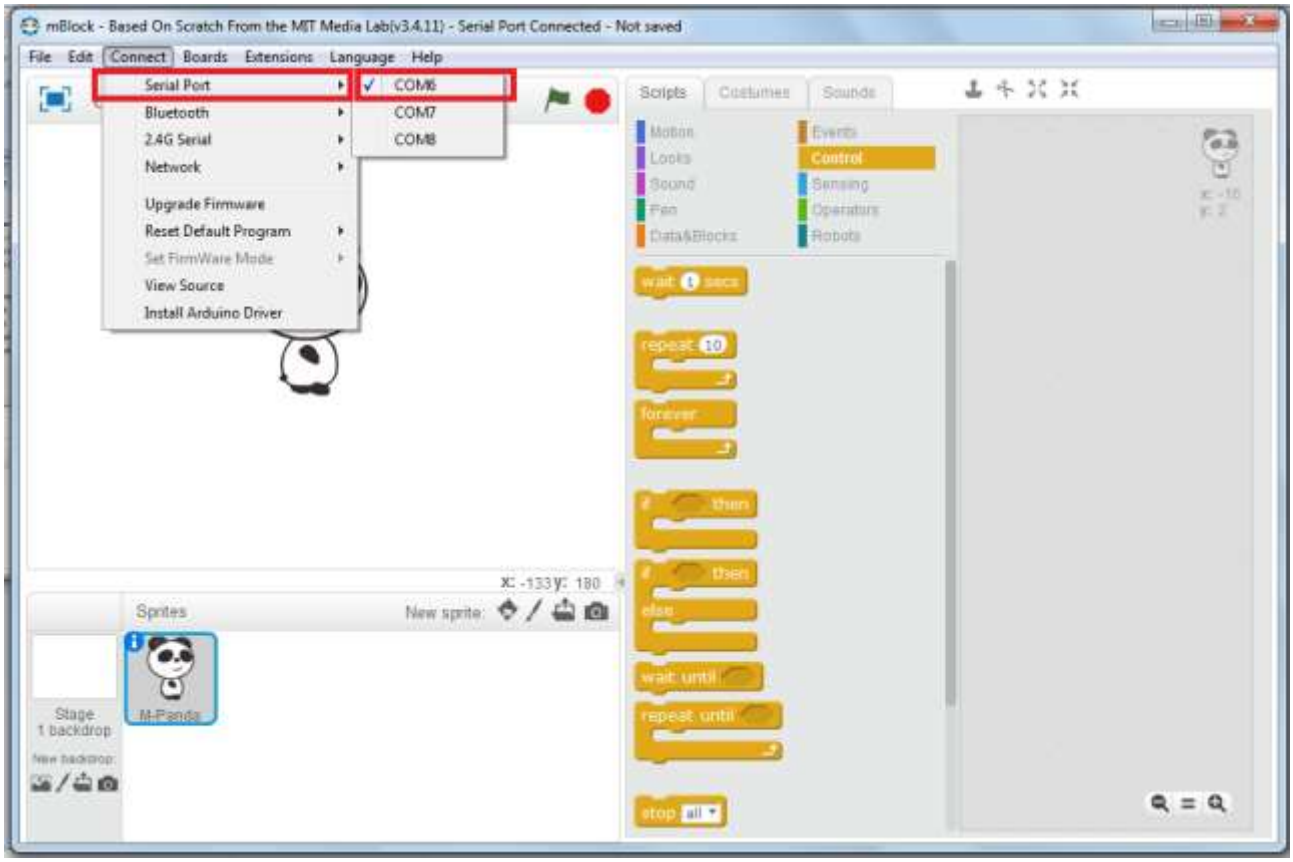
Теперь нужно установить среду mBlock на вашем компьютере. Если у вас недостаточно системных прав для установки программ, то обратитесь к вашему системному администратору.

После установки программы mBlock нужно ее правильно настроить:

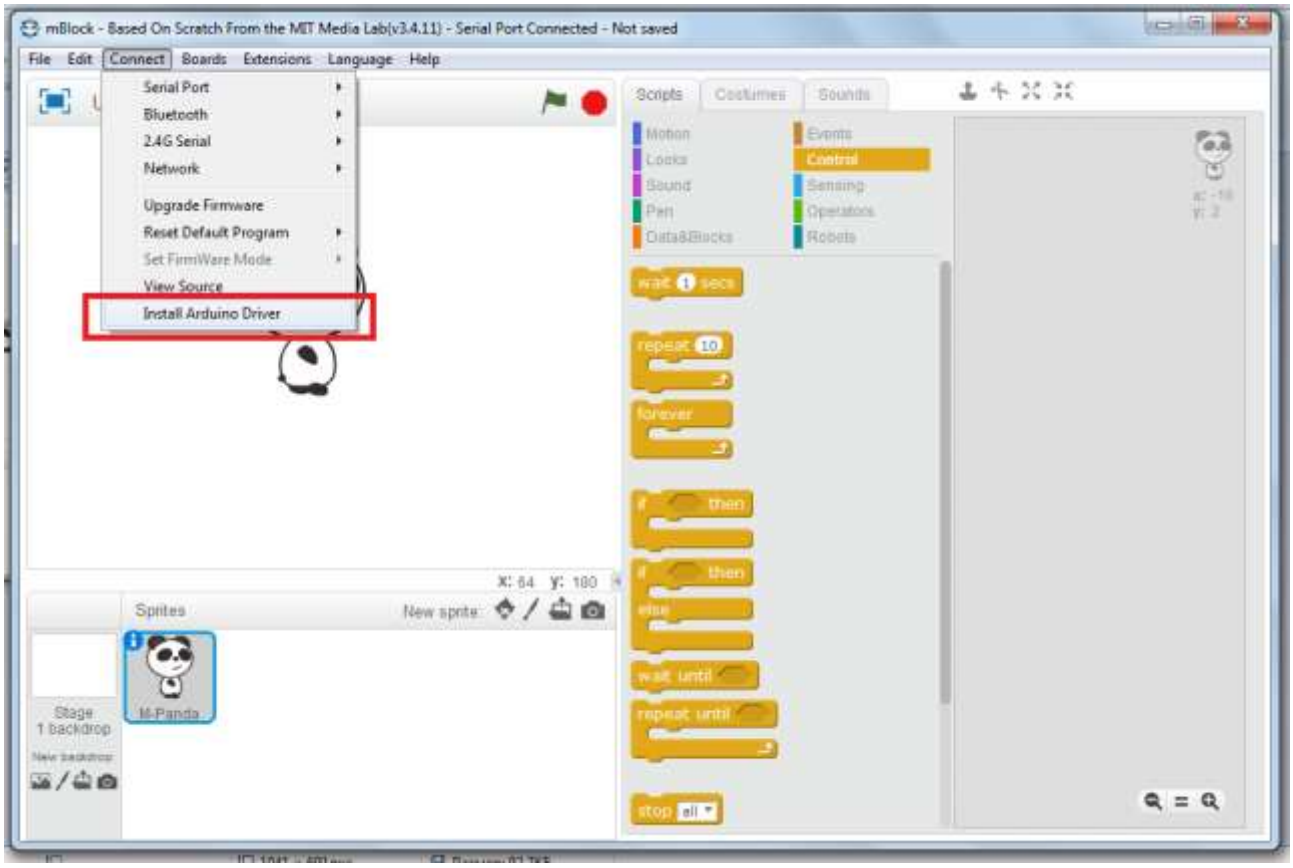
1. Запускаем mBlock.
2. С начала в верхнем меню нужно выбрать пункт «Board» (плата устройства) и в открывшемся списке поставить галочку напротив «mBot».



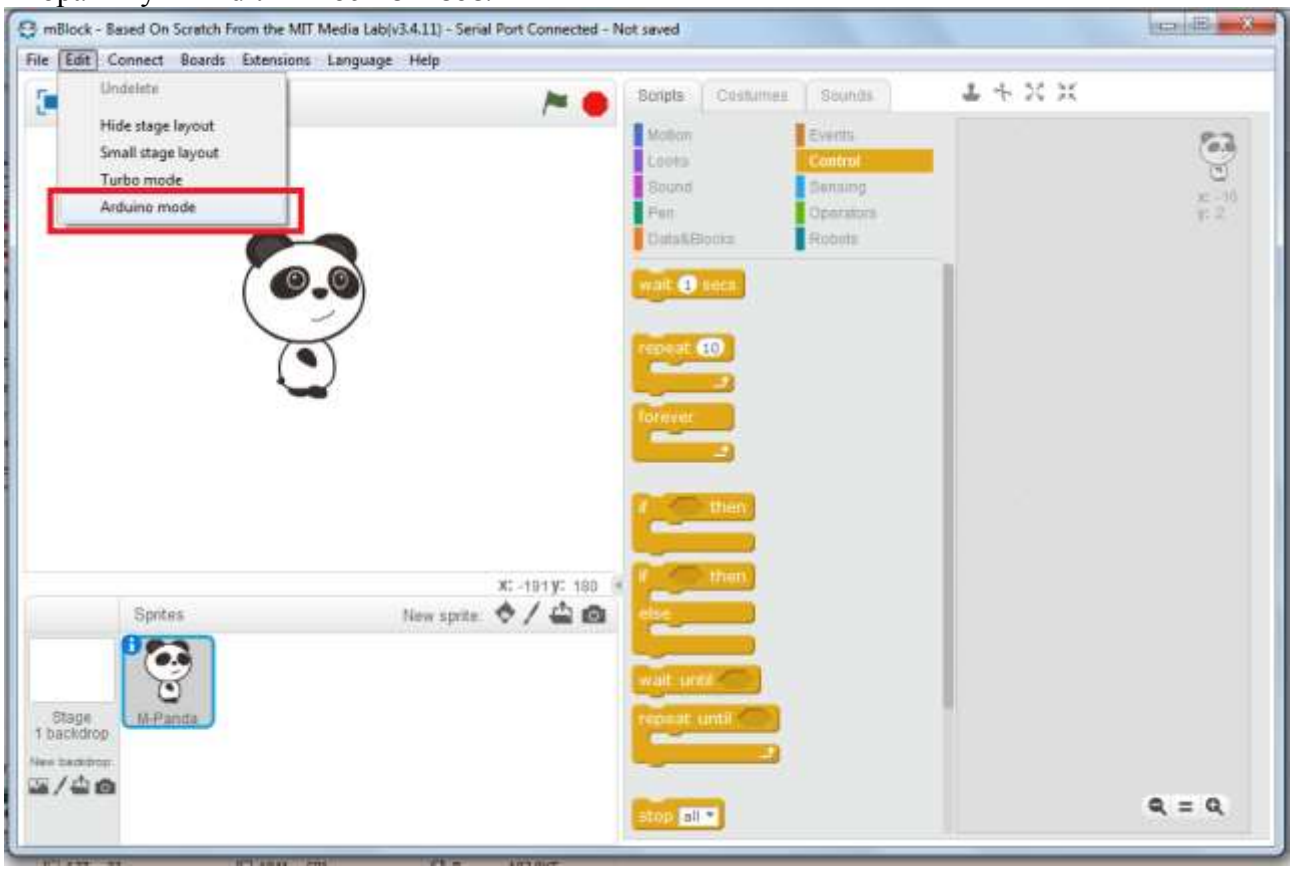
3. Затем для соединения с роботом нужно настроить COM-порт, который был присвоен вашему роботу в «Диспетчере устройств» (смотри предыдущую главу). Для этого в верхнем меню нужно выбрать Connect->Serial Port->COM6 (здесь COM6 указан для примера, вам нужно указать COM-порт вашего робота).



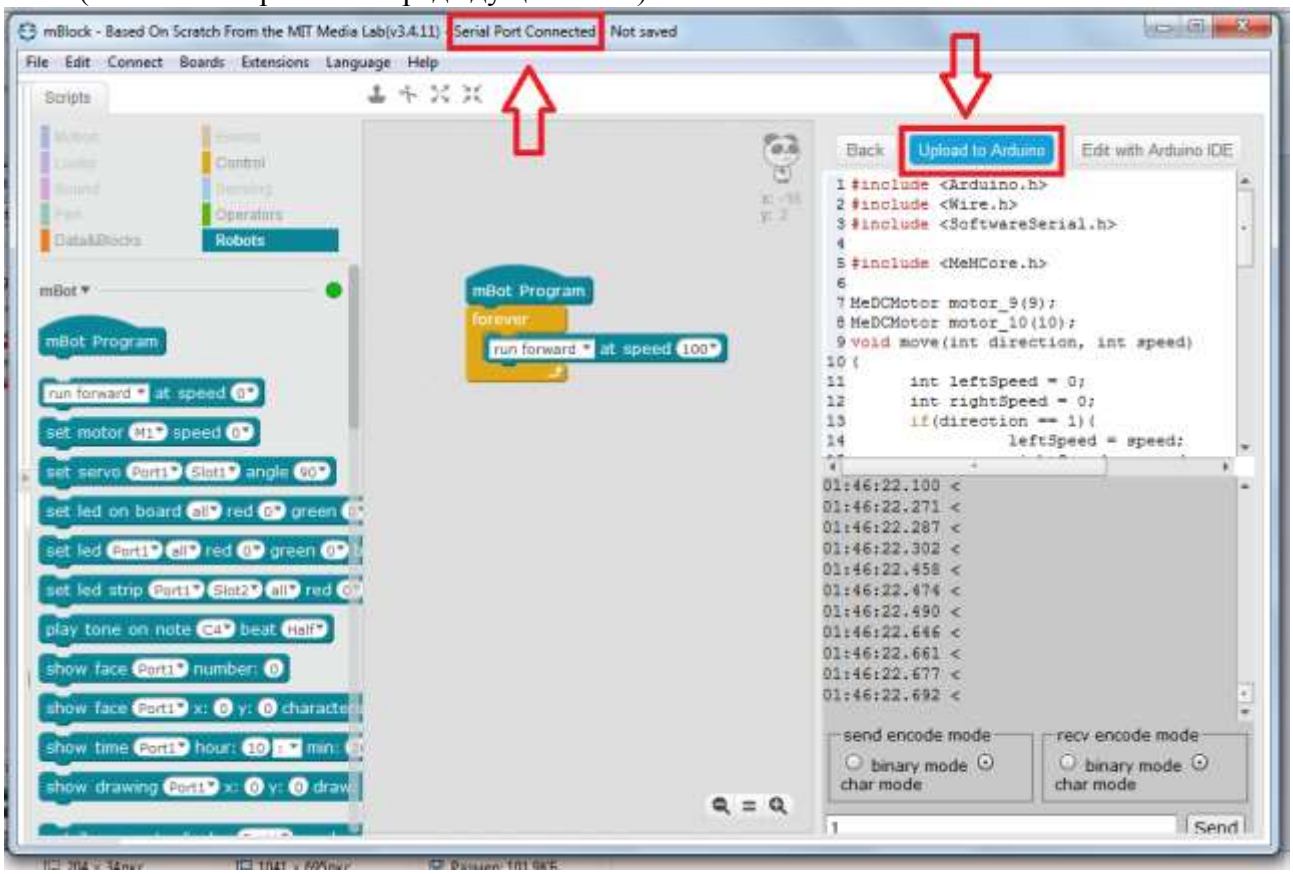
Если в «Диспетчере устройств» роботу не был присвоен COM-порт, то попробуйте установить драйвера. Для этого в программе mBlock в верхнем меню нужно выбрать пункт Connect->Install Arduino Driver. Либо нужно скачать драйвера по ссылке https://raw.githubusercontent.com/Makeblock-official/Makeblock-USB-Driver/master/Makeblock_Driver_Installer.zip и установить их вручную.



Среда mBlock сразу после запуска позволяет писать программы на языке Scratch.
Чтобы иметь возможность писать собственные программы для робота и загружать их в mBot, нужно переключить среду mBlock в режим «Arduino mode», для этого в верхнем меню нужно выбрать пункт Edit-> Arduino mode.



После этого вид mBlock изменится и можно будет составлять программы для mBot и загружать их в робота mBot щелкнув мышкой по кнопке «Upload to Arduino». Важно: перед загрузкой программы в робота обязательно убедитесь, что он подключен: в верхней строке окна mBlock должна быть надпись «Serial Port Connected» иначе проверьте соединение mBot с ПК (об этом говорилось в предыдущей главе).

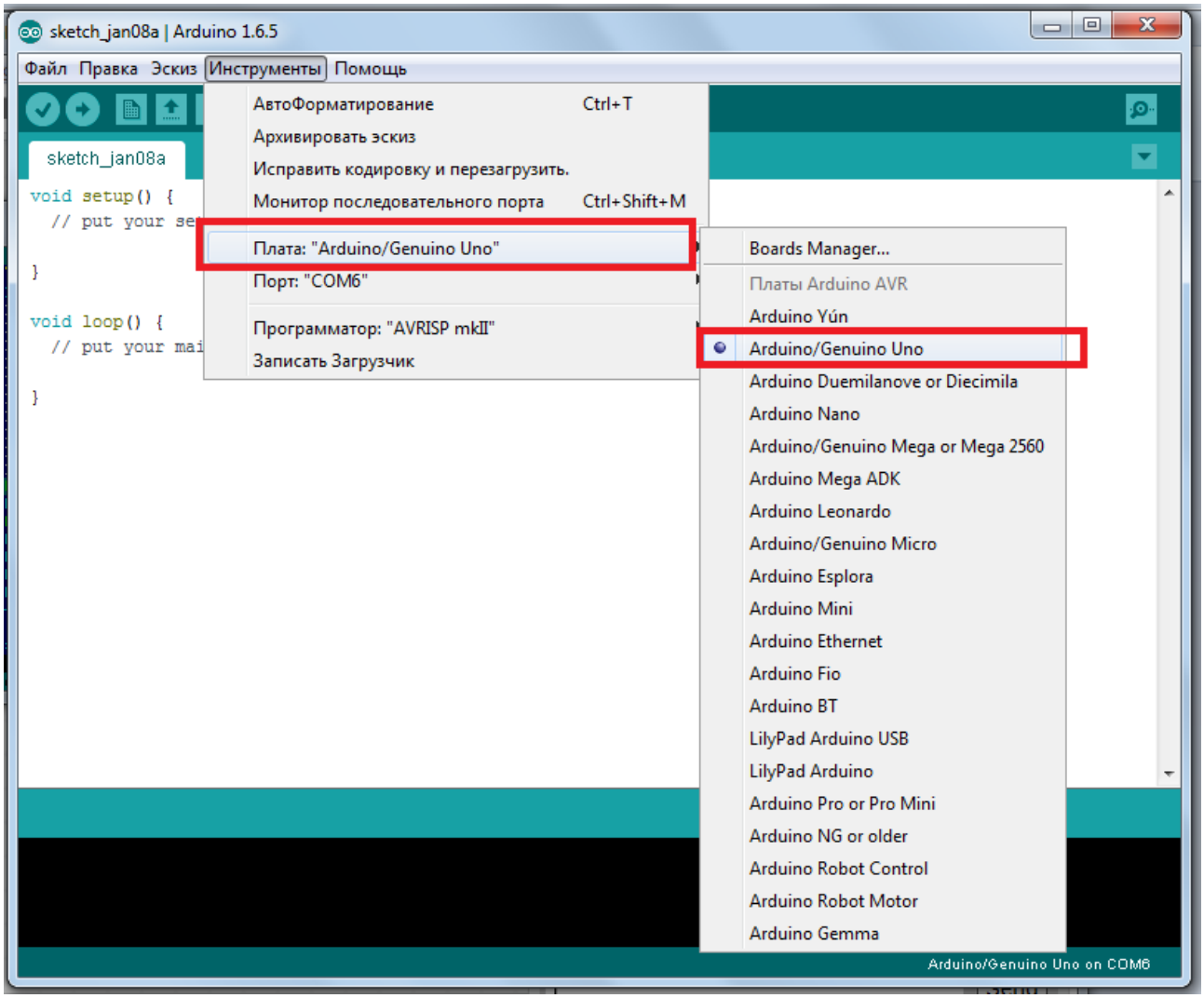


Если вам удобнее работать с меню и блоками на русском языке, то выберете в верхнем меню пункт «Language» и поставьте галочку напротив «Русский».

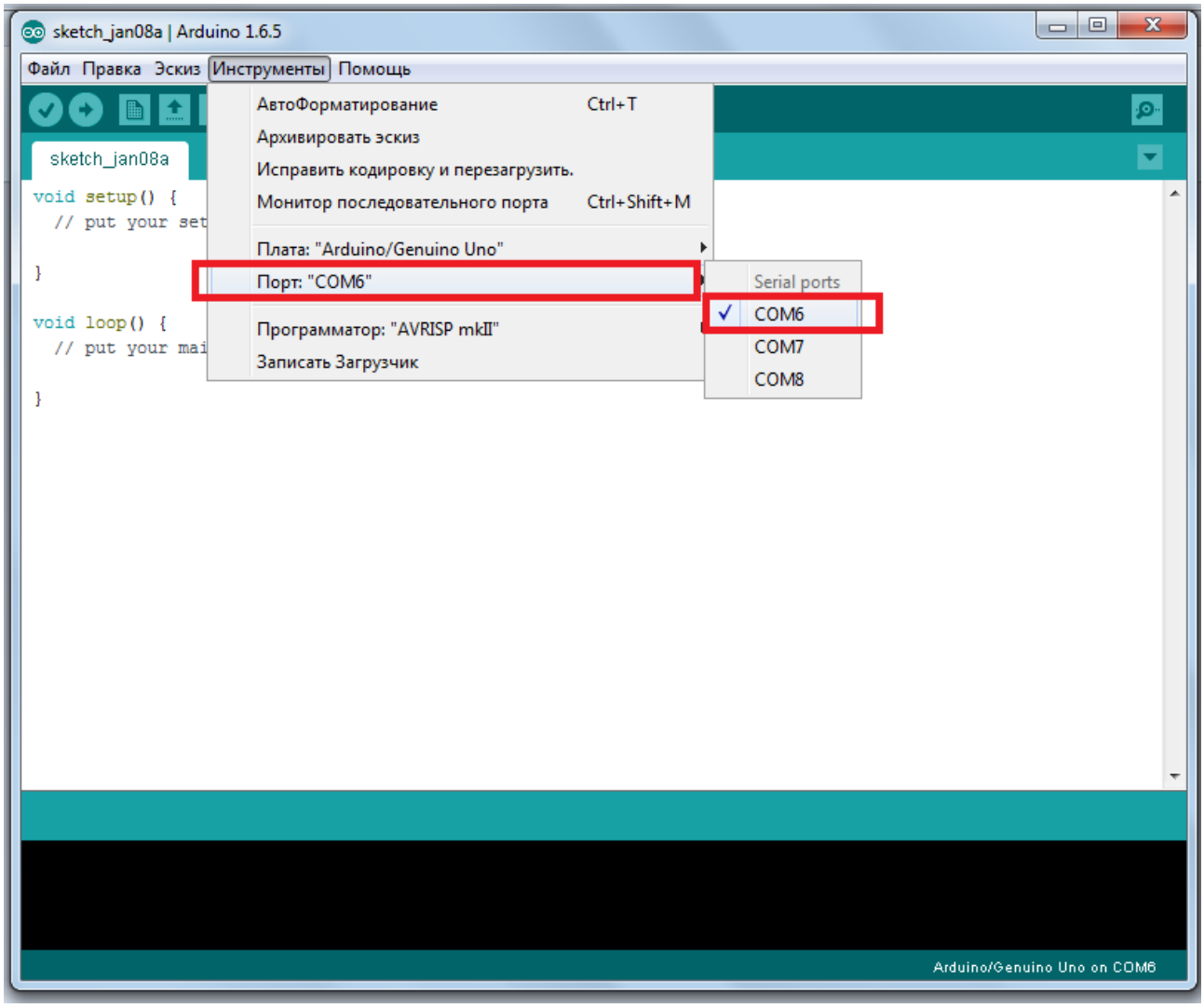
Настройки для программирования в Arduino IDE

Робота mBot можно программировать на языке программирования C/C++ с помощью среды Arduino IDE.

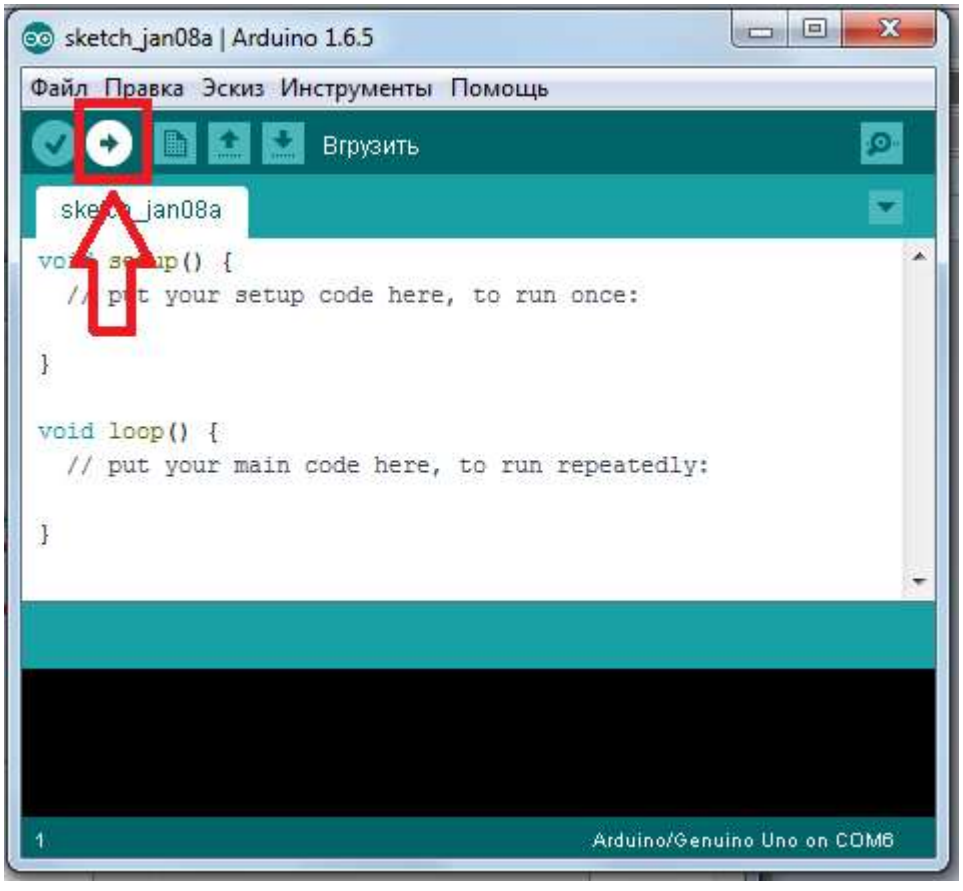
1. Для этого загрузите по интернет-ссылке http://arduino.cc/download версию Arduino IDE для операционной системы, установленной на вашем персональном компьютере.
2. Установите Arduino IDE на вашем компьютере либо распакуйте из скаченного архивного файла.
3. Запустите Arduino IDE (arduino.exe).
4. Плата робота mBot под названием mCore совместима с Arduino UNO, поэтому в верхнем меню в пункте «Инструменты (Tools)» нужно выбрать Плата->Arduino/Genuno UNO



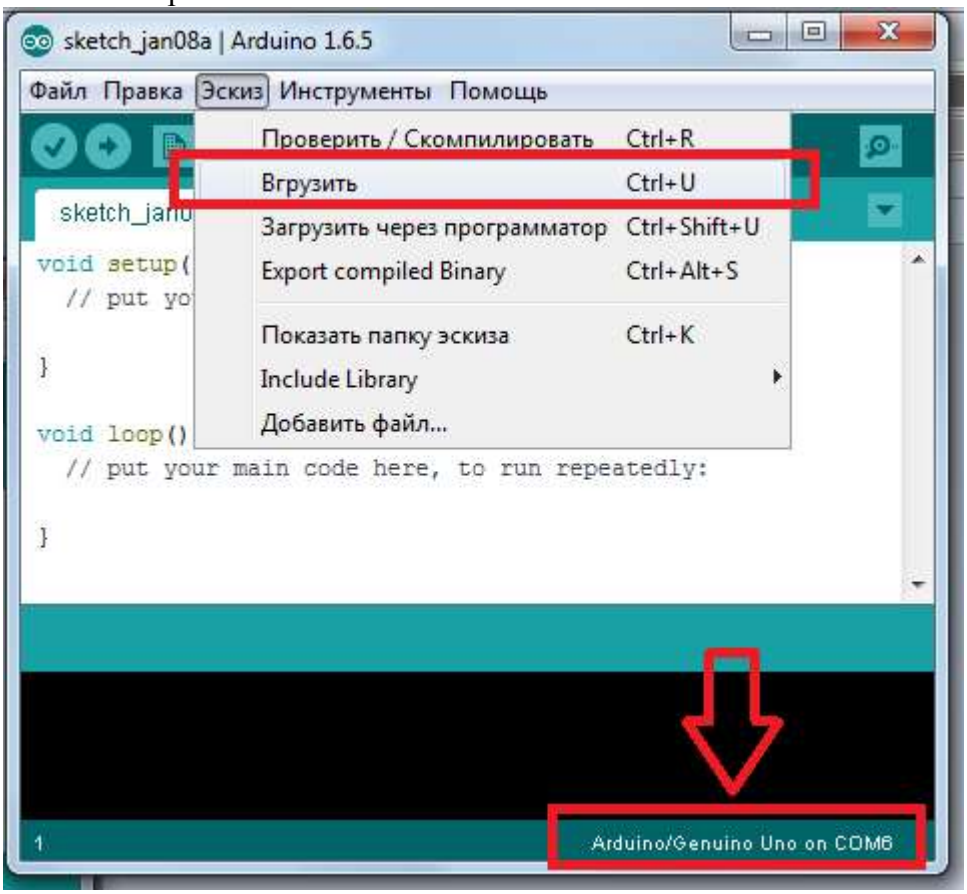
5. Для соединения робота mBot с персональным компьютером по USB кабелю в верхнем меню в пункте «Инструменты (Tools)» нужно выбрать Порт->COM6 (здесь COM6 приведен для примера, вам следует указать COM-порт вашего робота. см. главу «Подключение mBot к персональному компьютеру»)



6. Для загрузки программы в память робота mBot можно щелкнуть мышкой по стрелке «Вгрузить» на верхней панели инструментов.



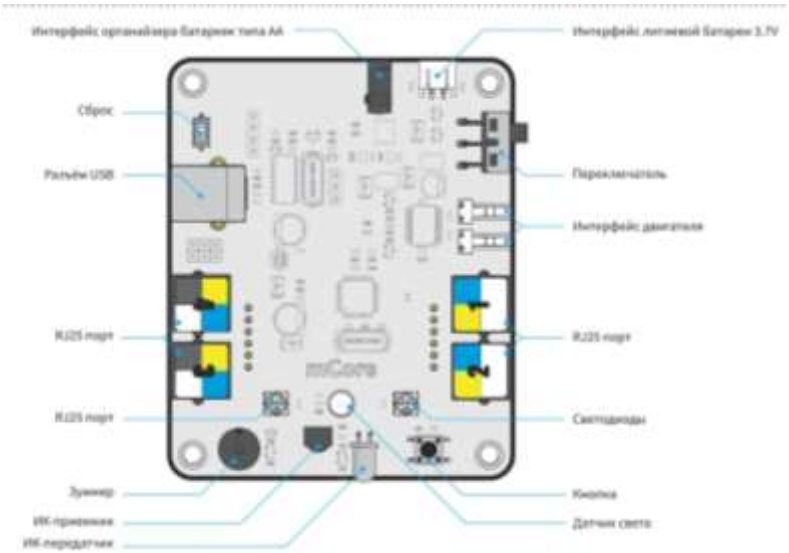
Либо выбрать в верхнем меню пункт Эскиз->Вгрузить, либо на клавиатуре одновременно нажать комбинацию клавиш Ctrl+U. Важно, перед загрузкой программы в робота обязательно убедитесь, что он подключен: в нижнем правом углу окна Arduino IDE выводится информация о соединении платы «Arduino/Genuino UNO» с COM-портом.



7. Если в «Диспетчере устройств» роботу не был присвоен COM-порт, то нужно скачать драйвера по ссылке https://raw.githubusercontent.com/Makeblock-official/Makeblock-USB-Driver/master/Makeblock_Driver_Installer.zip и установить их вручную.
8. Для программирования робота mBot в среде Arduino IDE потребуется добавить библиотеку MakeBlock. Скачайте библиотеку по ссылке https://github.com/Makeblock-official/Makeblock-Libraries и распакуйте ее и скопируйте папку makeblock в подпапку libraries, которая находится в папке arduino (например, в C:\Program Files\Arduino).

Плата mCore

В этой главе приведено на первый взгляд очень сложное техническое описание платы mCore. Однако, в последствие по мере вашего продвижения в программировании, информация из этой главы будет полезной. Пока можно с ней ознакомиться для понимания возможностей



платы mCore и робота mBot.

Общие характеристики платы mCore:

Среда программирования	Arduino IDE, mBlock (Mac OS, Windows, Linux*), mBlock APP (Android, IOS)
Устройства ввода	Датчик света (Light sensor), кнопка (button), инфракрасный приемник (infrared receiver), датчик света (Light sensor)
Устройства вывода	Зуммер (buzzer), RGB-светодиоды (RGB LED), инфракрасный передатчик (infrared transmitter), 2 порта для моторов (two motor ports)
Микроконтроллер	ATmega328, Arduino UNO-совместимый
Напряжение питания	От 3,7 Вольт до 6 Вольт
Беспроводная связь	Bluetooth
Связь с ПК	USB
Разъемы для электропитания	Разъем для батареек 4-х батареек AA, разъем для литий-ионной батареи
Сброс	Кнопка Reset для рестарта программы в mCore
Переключатель	Включение питания (Power switch) - ON, выключение — OFF
Разъемы для подключения двигателей	Motor Connector M1 - левый мотор, M2 — правый мотор

Устройства ввода — устройства, с помощью которых данные (информация) извне поступают в mCore.
Устройства вывода — устройства, через которые данные (информация) от mCore передаются во внешний мир.

Если рассмотреть электрическую схему mCore, то можно заметить следующее:

- Зуммер подключен к pin 8.
- Кнопка подключена к pin 8.
- Датчик света подключен к pin A6.
- RGB-светодиоды (WS2812 LEDs) подключены к pin 13.
- Мотор1 соединен с pin 5 (ШИМ определяет скорость вращения) и с pin 4 (направление вращения).
- Мотор2 соединен с pin 6 (ШИМ определяет скорость вращения) и с pin 7 (направление вращения).
- ИК приемник подключен к pin 2.
- ИК передатчик подключен к pin 3.

Логически в библиотеку makeblock введены «Порты», логически совмещенные из двух pin-портов:
Port_1 { 11 , 12 } = RJ25 port 1
Port_2 { 9 , 10 } = RJ25 port 2
Port_3 { A2 , A3 } = RJ25 port 3
Port_4 { A0 , A1 } = RJ25 port 4
Port_5 { NC , NC } (Not Connected)
Port_6 { 8 , A6 } = Buzzer, Light sensor
Port_7 { A7 , 13 } = Button, two WS2812 LEDs
Port_8 { 8 , A6 } = Buzzer, Light sensor
Port_9 { 6 , 7 } = PWM motor 2, DIR Motor 2
Port_10 { 5 , 4 } = PWM motor 1, DIR Motor 1

Для подключения внешних датчиков или модулей на плате mCore имеется четыре порта RJ25 с цветовой кодировкой, которая обозначает следующее:

- Черный цвет — один или два аналоговых порта.
- Синий цвет — два цифровых порта.
- Желтый цвет — один цифровой порт.
- Белый цвет — I2C порт.

Вопросы на внимательность

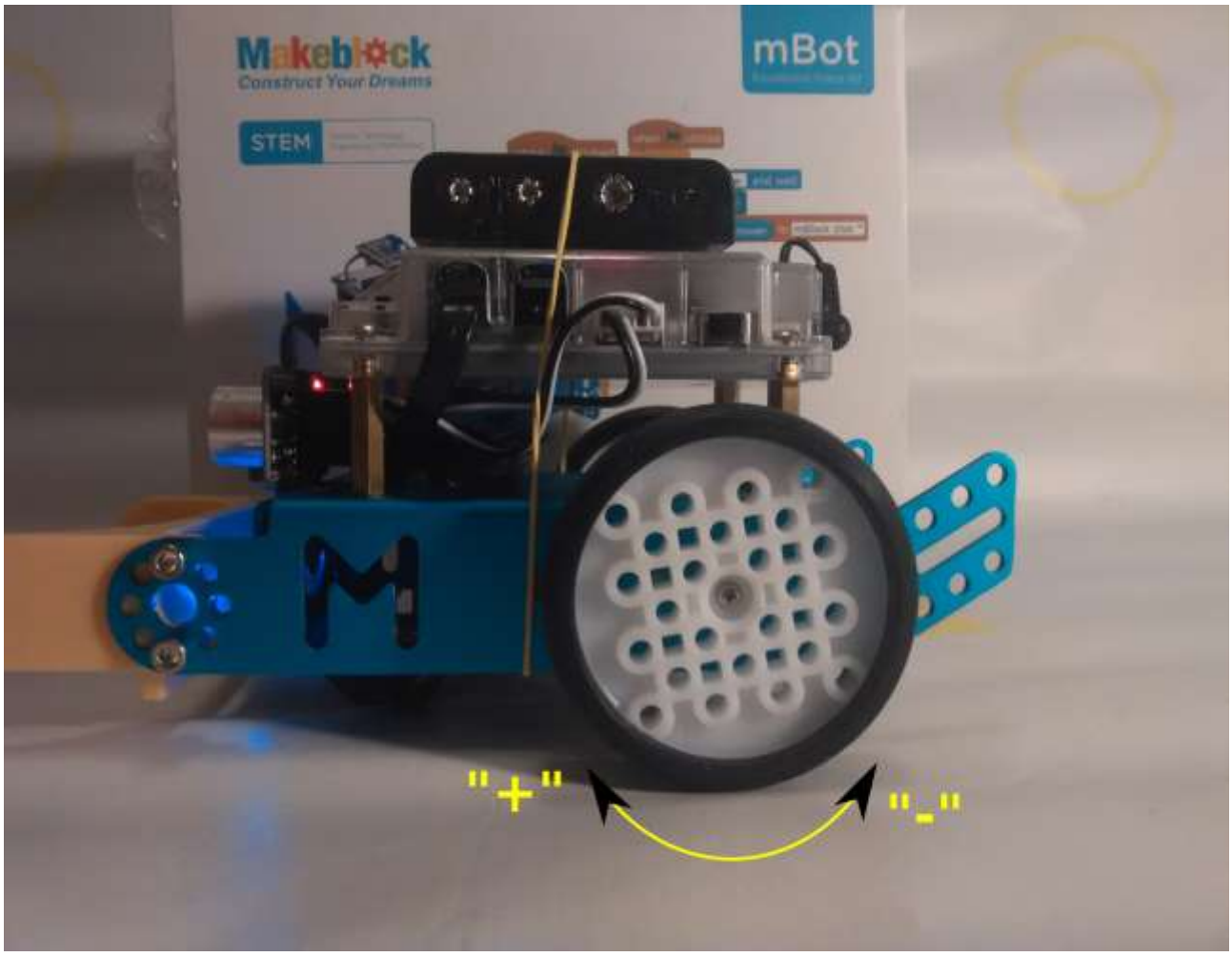
1. Сколько устройств ввода на плате mCore?
2. Сколько устройств вывода на плате mCore?
3. Какое максимальное число внешних датчиков или модулей можно одновременно подключить к mCore, используя порты RJ25?
4. Как легче запомнить, к какому разъему M1 или M2 нужно подключать левый

Управление моторами mBot

Робот mBot собран и хотелось бы сразу приступить к его программированию. Лично мне не терпится увидеть как он ездит. Поэтому начнем программирование с управления двигателями. У робота два мотора M1 - левый и M2 - правый. Мы сможем программировать скорость вращения и направление вращения каждого из них по отдельности и, таким образом, программировать движения робота mBot вперед-назад и повороты влево-вправо.

Скорость вращения моторами на mBot варьируется в диапазоне от -255 до 255, причем от знака «+» или «-» числового значения скорости зависит направление вращения двигателем.

Если «+» - вращение DC-мотора по часовой стрелке, если смотреть на колесо, прикрепленное к валу двигателя. А если «-», то вращение против часовой стрелки. Учтем это при написании скетча для mBota: двигатели развернуты друг относительно друга на 180 градусов, поэтому, если нам нужно, чтобы робот двигался вперед, то для левого мотора нужно подать отрицательное значение скорости, а для правого — положительное.

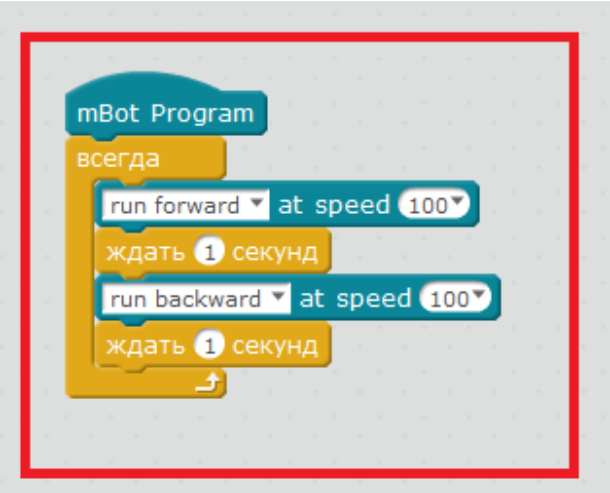


Программируем движение робота mBot вперед-назад в mBlock

Запустите программу mBlock и проверьте настройки, описанные в главе «Настройки для

программирования в mBlock».

Используя расположенные справа-сверху в палитре Скрипты (Scripts) блоки из разделов Контроль (Control) и Робот (Robots) и перетаскивая их на рабочее поле в центре mBlock, составьте следующую программу.

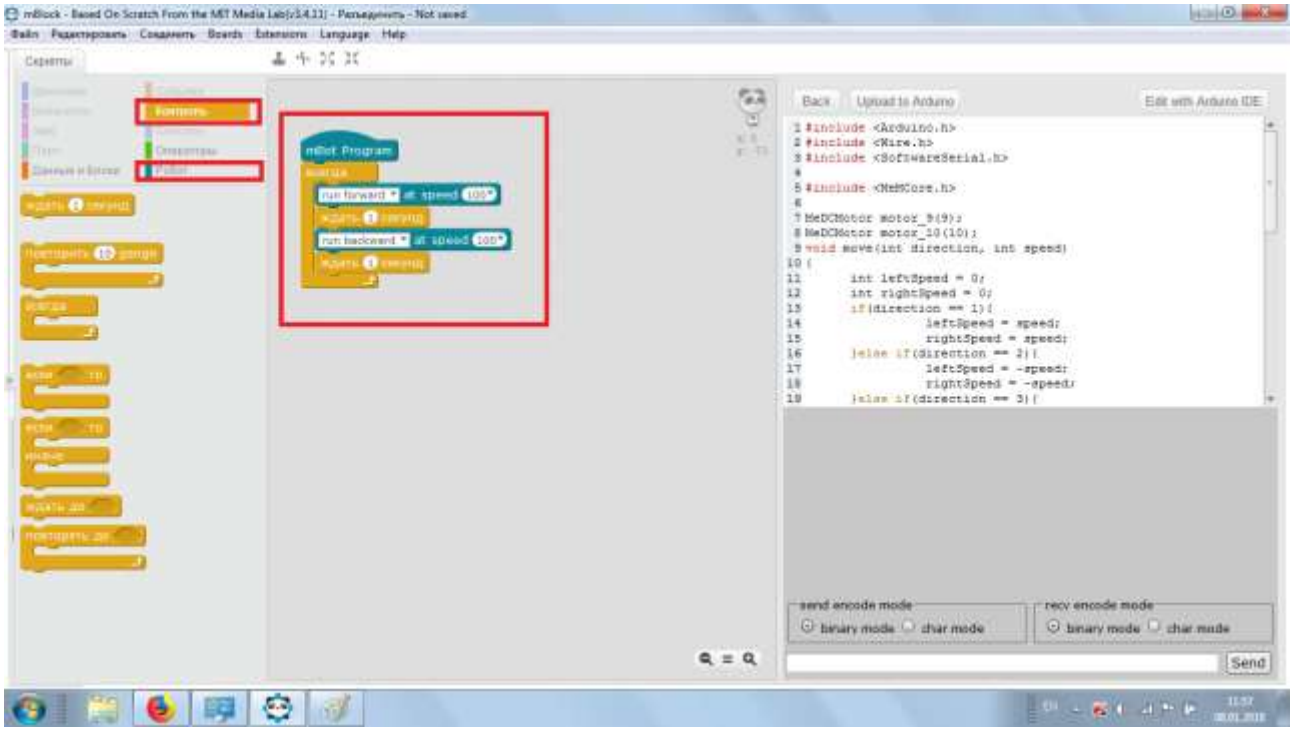


Как это работает

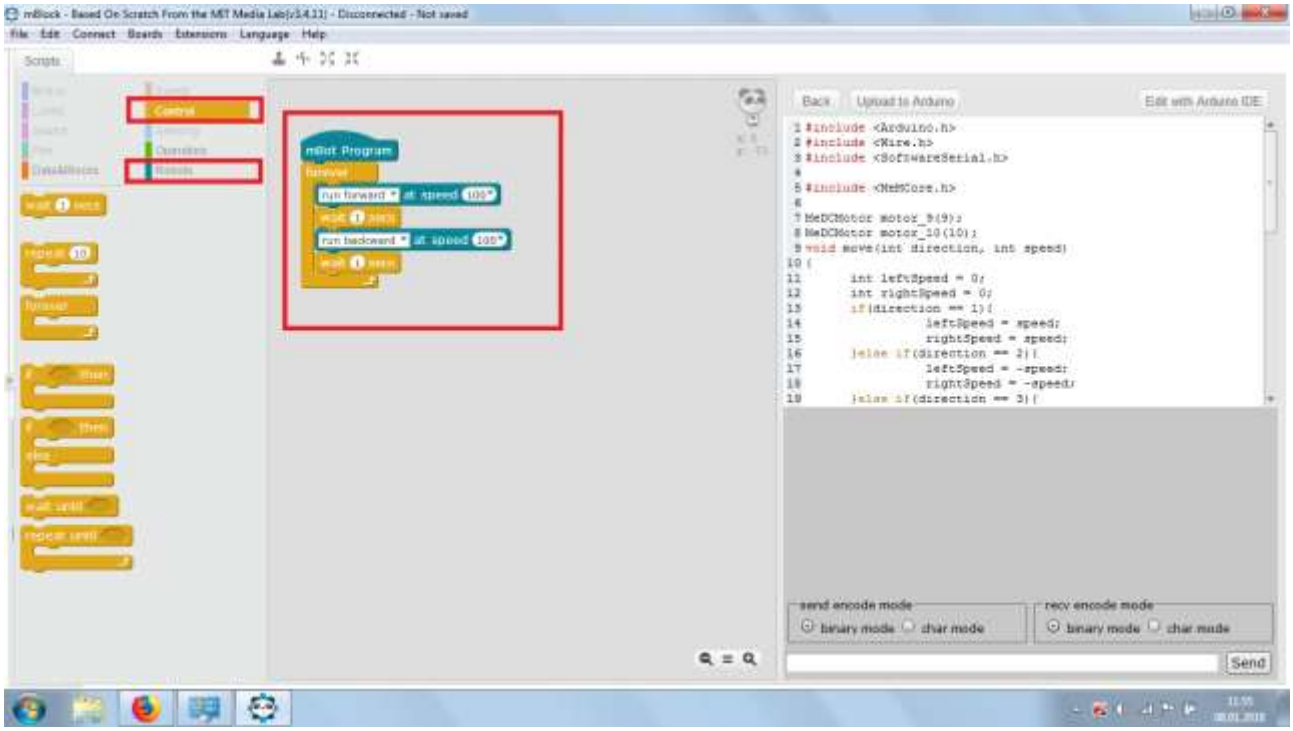
Давайте пошагово разберемся как эта программа работает.

1. Блок «mBot Program» - заголовок программы для робота mBot.
2. Блок «всегда» - это бесконечный цикл, внутри которого все блоки с командами выполняются много-много раз, т.е. бесконечно, пока не выключите питание робота.
3. Блок «run forward at speed 100» (этот блок не переведен на русский язык, но смысл переводится так: «бежать вперед на скорости 100») - включает вращение моторов робота mBot так, чтобы он двигался вперед на скорости 100. Это не самая быстрая скорость для моторов, но для нас сейчас главное разобраться в программировании, а не быстрые гонки.
4. Блок «ждать 1 секунд» нужен для того, чтобы робот двигался вперед 1 секунду. Программа в это время ничего не будет делать, просто ждать пока истечет 1 секунда, зато робот mBlock за это время преодолеет некоторое расстояние, двигаясь вперед.
5. Блок «run backward at speed 100» (этот блок тоже не переведен на русский язык, но смысл переводится так: «бежать назад на скорости 100») - это команда для робота mBot двигаться назад (задним ходом) на скорости 100.
6. Блок «ждать 1 секунд» нужен для того, чтобы робот двигался назад 1 секунду.
7. После этого цикл «всегда» повторит все команды, начиная с блока «run forward at speed 100» в пункте 3, и так до бесконечности.

Специально приводим изображение программы mBlock для русского языка:



Если вы не выбирали в настройках программы mBlock русский язык, то увидите следующие надписи, но в следующих иллюстрациях мы будем использовать только русский язык.



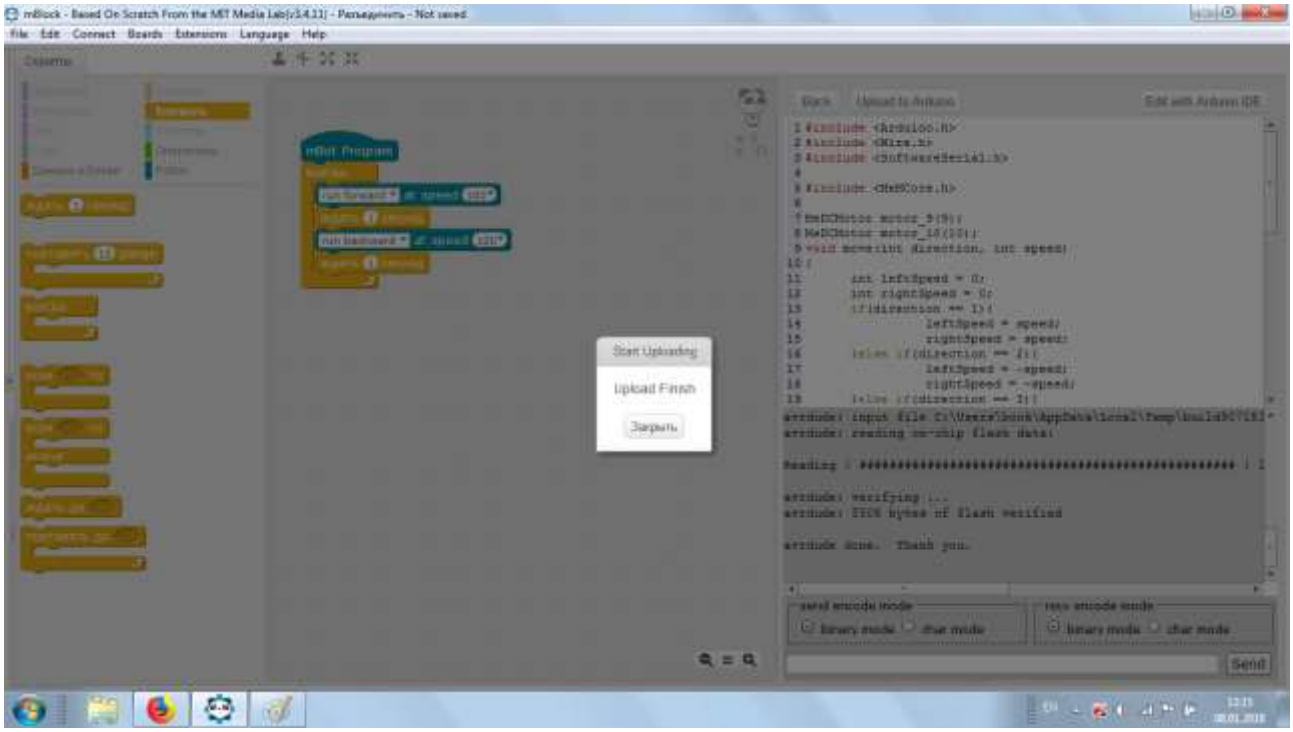
Сохраните созданную программу под названием «mBot_kvadrat», выбрав в верхнем меню File->Save Project As.

Загрузка программы в память робота из mBlock

Перед загрузкой программы в память mBot проверьте соединение робота с персональным компьютером и включите питание робота.

Внимание! После успешной загрузки программы робот сразу начнет вращать колесами вперед-назад! Будьте готовы к этому, держите робота в руках на весу, чтобы он не упал со стола!

Когда будете готовы к загрузке, кликните по кнопке «Upload to Arduino» и подождите несколько минут, пока программа не загрузится в робота mBot. По окончании появится сообщение об успешной загрузке «Upload Finish»:



И mBot сразу начнет вращать колесами вперед-назад! Теперь можно выключить питание, отсоединить USB-кабель от робота и перенести робота к месту испытаний.

Будьте осторожны, ведь робот теперь в зоне вашей ответственности. Не допускайте его падения или поломки, когда он выполняет вашу программу. Установив робота на безопасную от падения поверхность, например, на пол, включайте питание и начинайте его испытания.

Программируем движение робота mBot вперед-назад в Arduino IDE

Язык программирования C/C++ считается сложнее, чем Scratch, но зато позволяет более тонко управлять роботом. К тому же загрузка программы в память робота mBlock из Arduino IDE происходит гораздо быстрее, чем в mBlock.

Запустите программу Arduino IDE и наберите в ней следующий скетч (программу):

```
/*
Робот mBot движется вперед-назад
(c) Sergey Kosachenko
*/
```

```
//подключаем библиотеку Makeblock
#include "MeMCore.h"

//определяем двигатели
MeDCMotor motorLeft(M1); //левый двигатель
MeDCMotor motorRight(M2); //правый двигатель

//Скорость моторов 100
uint8_t motorSpeed = 100;

void setup()
{
}

void loop()
{

  //ДВИЖЕНИЕ РОБОТА ВПЕРЕД
  //вперед левый мотор
  motorLeft.run(-motorSpeed);
  //вперед правый мотор
  motorRight.run(motorSpeed);

  delay(1000); //ждать 1 секунду пока робот едет вперед

  //ДВИЖЕНИЕ РОБОТА НАЗАД
  //назад левый мотор
  motorLeft.run(motorSpeed);
  //назад правый мотор
  motorRight.run(-motorSpeed);

  delay(1000); //ждать 1 секунду пока робот едет назад
}
```

Как это работает

Подключаем библиотеку makeblock для платы mCore директивой:

```
#include "MeMCore.h"
```

Затем создаем объекты — левый и правый моторы, указывая в скобках порты, к которым они подключены на плате mCore:

```
MeDCMotor motorLeft(M1); //левый двигатель
MeDCMotor motorRight(M2); //правый двигатель
```

Значение скорости моторов 100 будем хранить в числовой переменной motorSpeed — это удобно для внесения изменений в программу в будущем:

```
uint8_t motorSpeed = 100;
```

Обязательная процедура void setup() выполняется один раз в момент включения робота или при нажатии на кнопку Сброс. Но нам не нужно, чтобы она содержала команды. Поэтому она «пустая»:

```
void setup()
{
}
```

Обязательная процедура void loop() - это бесконечный цикл. т.е. аналог блока «всегда» в mBlock (Scratch). Она содержит все команды для движения робота, которые расположены между фигурными скобками { }.

При движении робота вперед учитываем, что левый двигатель перевернуть относительно правого на 180 градусов, следовательно, чтобы робот mBot двигался вперед, его левый мотор должен вращаться против часовой стрелки: motorLeft.run(-motorSpeed),- обратите внимание, что числовое значение скорости со знаком минус. А правый мотор должен вращаться по часовой стрелке motorRight.run(motorSpeed) (числовое значение скорости со знаком плюс):

```
  //ДВИЖЕНИЕ РОБОТА ВПЕРЕД
  //вперед левый мотор
  motorLeft.run(-motorSpeed);
  //вперед правый мотор
  motorRight.run(motorSpeed);
```

Пауза в Arduino IDE измеряется в миллисекундах. В 1 секунде — 1000 миллисекунд, поэтому команда выглядит так:

```
delay(1000); //ждать 1 секунду пока робот едет вперед
```

Для начала движения робота назад, нам достаточно в командах управления мотороами поменять знаки для значения скорости на противоположные:

```
 //ДВИЖЕНИЕ РОБОТА НАЗАД
  //назад левый мотор
  motorLeft.run(motorSpeed);
  //назад правый мотор
  motorRight.run(-motorSpeed);
```

Еще раз применим команду «ждать секунду», пока робот движется назад (задним ходом):

```
delay(1000); //ждать 1 секунду пока робот едет назад
```

После этого подпрограмм void loop() снова начнет выполнять команды, которые между фигурными скобками {}, начиная с движения робота вперед, и так до бесконечности.

Вы, наверное, обратили внимание, что текст программы содержит много ценных пояснений, которые начинаются с символов // - таким образом мы можем комментировать программы, чтобы улучшить их понимание. Эти комментарии не влияют на команды программы, но весьма удобны для программистов, использование их считается очень хорошим тоном и указывает на высокий класс программиста, поэтому не стесняйтесь их применять и они вас неоднократно выручат в будущем.

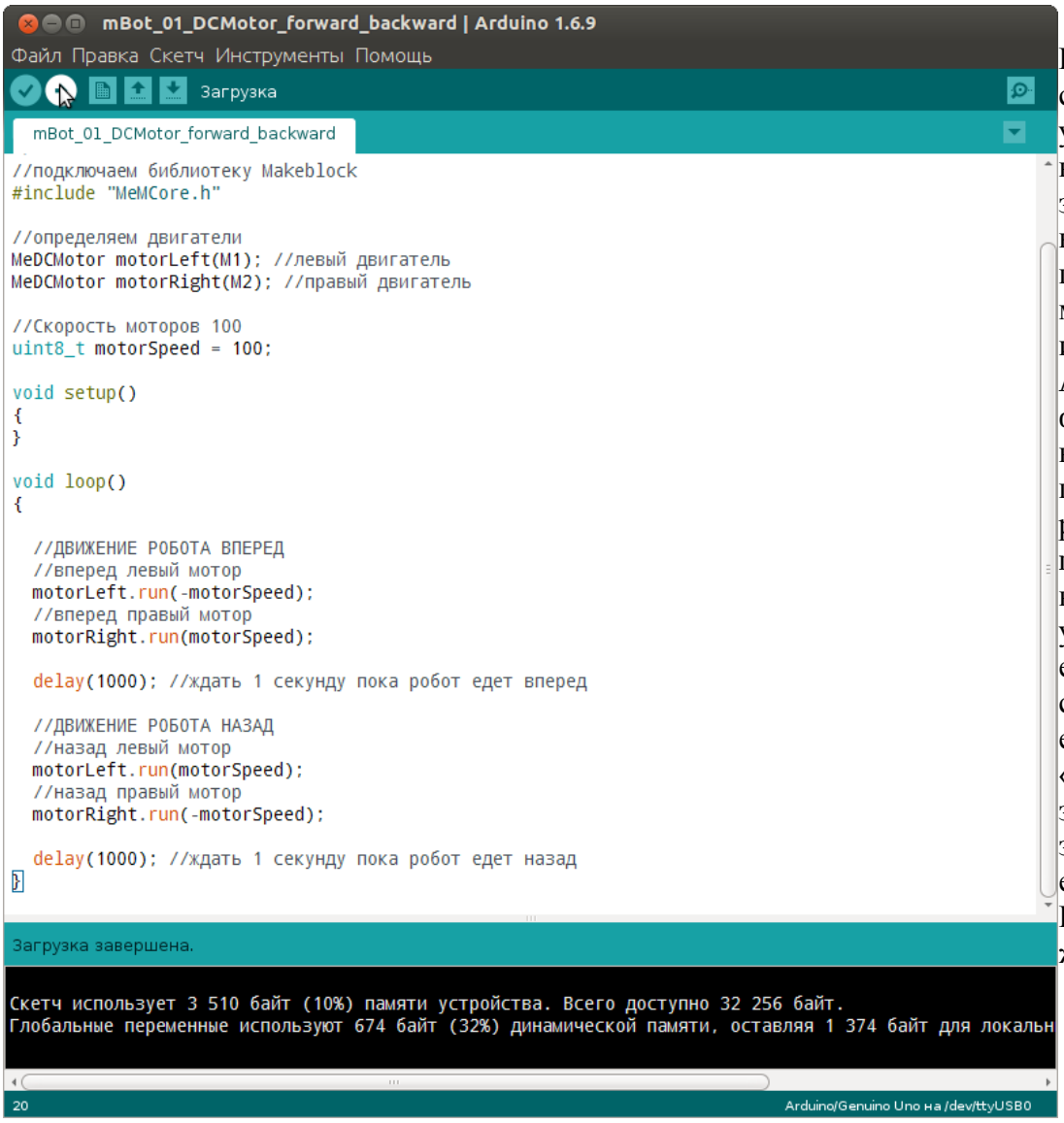
Сохраните созданную программу под названием «mBot_kvadrat».

Загрузка программы в память робота из Arduino IDE

Перед загрузкой программы в память mBot проверьте соединение робота с персональным компьютером и включите питание робота.

Внимание! После успешной загрузки программы робот сразу начнет вращать колесами вперед-назад! Будьте готовы к этому, держите робота в руках на весу, чтобы он не упал со стола!

Когда будете готовы к загрузке, кликните по кнопке «Загрузка» (Вгрузить):



В случае успешной загрузки программы из Arduino IDE в память робота mBot вы увидите сообщение «Загрузка завершена». Если же вы

допустили какую-либо ошибку, то сообщение об этом появится в нижней части окна на черном фоне, а загрузка программы будет аварийно прервана. В этом случае прочитайте сообщение об ошибке, исправьте ее и повторите загрузку. Безусловно, вам будет полезен краткий справочник на русском языке по программированию в Arduino IDE http://arduino.ru/Reference

После успешной загрузки можно отключить питание робота, отсоединить USB-кабель от mBot и переходить к месту испытания движения робота.

Вопросы на внимательность

Сколько сантиметров проезжает робот за 1 секунду на скорости 100?

Движение робота mBot по квадрату

Попробуем усложнить движение робота, добавив повороты на 90 градусов. Чтобы запрограммировать поворот робота, нужно дать команду, чтобы моторы вращались в разные стороны. Через некоторое время, которое вам нужно установить экспериментально, робот повернется на 90 градусов, после чего нужно остановить вращение моторов. Этот прием называется **поворот по таймингу**.

Программируем в mBlock

Изменим предыдущую программу, изменив или добавив в нее несколько блоков:

Сохраните программу на компьютере под названием «mBot_kvadrat».

Как это работает

В блоке «всегда» бесконечного цикла будут выполнятся следующие блоки:

1. Блок «run forward at speed 100» («бежать вперед на скорости 100») включает моторы робота для движения вперед.
2. Блок «ждать 2 секунд» ожидает 2 секунды пока робот движется вперед.
3. Блок «run forward at speed 0» («бежать вперед на скорости 0») останавливает моторы робота.
4. Блок «ждать 0.5 секунд» ожидает полсекунды пока робот остановится окончательно.
5. Блок «turn left at speed 100» («поворачивать влево на скорости 100») включает моторы робота для поворота налево.
6. Блок «ждать 0.7 секунд» ожидает 0,7 секунды пока робот осуществляет поворот. **Это время вам нужно подбирать экспериментально, чтобы робот повернул как можно точнее на 90 градусов.**
7. Блок «run forward at speed 0» («бежать вперед на скорости 0») останавливает моторы робота.

8. Блок «ждать 0.5 секунд» ожидает полсекунды пока робот остановится окончательно.

После этого заново выполнятся все блоки, начиная с блока №1 по №8, и так бесконечно.

Выполняя данную программу, робот mBot проедет прямо в течение 2 секунд, остановится, повернет налево на 90 градусов, остановится и будет повторять эти шаги в бесконечном цикле. В результате выполнения такой программы робот mBot будет ездить по траектории, близкой к квадрату.

Важно экспериментально подобрать время в блоке «ждать 0.7 секунд» в строке №6, если робот будет поворачиваться на угол меньше, чем 90 градусов, то попробуйте увеличить время на 0.1 секунды, а если угол поворота будет больше, чем прямой угол, то уменьшайте время на 0.1 до тех пор, пока угол поворота робота не будет достаточно близок к 90 градусам.

Программируем в Arduino IDE

В программу, созданную по предыдущей главе, добавим в нашу программу остановку двигателей, чтобы они меньше изнашивались, и поворот по таймингу на 90 градусов налево.

Наберите в Arduino IDE следующий скетч и загрузите его в память робота mBot:

```
/*
Робот mBot движется по квадрату
(c) Sergey Kosachenko
*/
//подключаем библиотеку Makeblock
#include "MeMCore.h"
//определяем двигатели
MeDCMotor motorLeft(M1); //левый двигатель
MeDCMotor motorRight(M2); //правый двигатель

//Скорость моторов 100
uint8_t motorSpeed = 100;

void setup()
{
}

void loop()
{
  //ДВИЖЕНИЕ РОБОТА ВПЕРЕД
  //вперед левый мотор
  motorLeft.run(-motorSpeed);
  //вперед правый мотор
  motorRight.run(motorSpeed);
```

```
  delay(2000); //ждать 2 секунды пока робот едет вперед

  //стоп моторы!
  motorLeft.stop();
  motorRight.stop();
  delay(500); // подождем 0,5 секунды пока робот остановится

  //поворот налево примерно на 90 градусов на месте
  motorLeft.run(motorSpeed); //левый мотор назад
  motorRight.run(motorSpeed); //правый мотор вперед
  delay(700); //пауза 0,7 секунд - время для поворота

  //стоп моторы!
  motorLeft.stop();
  motorRight.stop();
  delay(500); // подождем 0,5 секунды пока робот остановится
}
```

Сохраните программу на компьютере под названием «mBot_kvadrat».

Как это работает

При изменении программы, созданной к предыдущей главе, мы добавили команды «стоп моторы!», которые в течение полсекунды полностью останавливают двигатели робота перед следующим маневром, а значит бережнее управляют моторами, избегая резких нагрузок при резком изменении направления вращения:

```
  //стоп моторы!
  motorLeft.stop();
  motorRight.stop();
  delay(500); // подождем 0,5 секунды пока робот остановится
```

Для поворота робота mBot налево на 90 градусов напишем команду левому мотору вращаться назад, а правому — вперед со скоростью 100, после чего поставим команду для ожидания 0.7 секунды (это время нужно подбирать экспериментально и оно сильно зависит от заряда батарей и от коэффициента трения покрытия поверхности, по которой будет ездить робот):

```
  //поворот налево примерно на 90 градусов на месте
  motorLeft.run(motorSpeed); //левый мотор назад
  motorRight.run(motorSpeed); //правый мотор вперед
```

Выполняя данную программу, робот mBot проедет прямо в течение 2 секунд, остановится, повернет налево примерно на 90 градусов, остановится и будет повторять эти шаги в бесконечном цикле. В результате выполнения такой программы робот mBot будет ездить по траектории, близкой к квадрату.

Вопросы на внимательность

Нужно ли заново подбирать время для поворота робота на 90 градусов, если после программирования поменять батарейки на более свежие?

Нужно ли заново подбирать время для поворота робота на 90 градусов, если после программирования и испытания на одной поверхности, требуется запускать робота на более шершавой поверхности?

Почему команда для небольшой остановки моторов после движения, например, вперед или налево, может продлить срок исправной работы двигателей?

Добавляем нажатие кнопки

Программа для робота mBot в предыдущей главе была бы удобнее, если бы робот сначала ожидал нажатие на кнопку и только потом начинал движение. Это позволяет не торопясь точно установить робота на поле и по готовности быстро дать команду на старт робота. На плате mCore имеется кнопка, которую мы задействуем в нашей программе.

Программируем в mBlock

Изменим программу из предыдущей главы, добавив в нее только один блок «ждать до» и в качестве условия поставить параметр «on board button pressed» (что переводится «кнопка на плате нажата»):

Сохраните программу на компьютере под названием «mBot_button_krvadrat».

Как это работает

При включении питания робота или нажатии на кнопку Сброс программа запускается сначала. Первым блоком в программе стоит блок «ждать до». Условием, которое завершит ожидание данного блока, является нажатие кнопки на плате. Как только кнопка на роботе будет нажата, робот перейдет к выполнению следующей команды в программе, а это блок «всегда», который в бесконечном цикле выполняет команды, вложенные в него. Как они работают. Мы подробно рассмотрели в предыдущей главе.

При загрузке программы в память робота mBot, он будет ожидать нажатия кнопки, чтобы начать движение. Теперь нам не нужно удерживать робота на весу, ожидая, что сразу после окончания загрузки он начнет движение.

После успешной загрузки программы, отсоедините USB-кабель от робота и установите его на поверхность, по которой он будет ездить. Для начала движения нажмите кнопку на роботе, и mBot сразу начнет движение по квадрату.

Программируем в Arduino IDE

Чтобы запрограммировать старт движения робота только после нажатия на кнопку, воспользуемся информацией из главы «Плата mCore». В ней написано, что кнопка подсоединена к микроконтроллеру через pin A7. Команды, которые считывают статус нажатия на кнопку и ожидают пока кнопка не будет нажата, мы разместим в процедуре void setup(), т.к. она выполняется при включении робота или нажатии на кнопку Сброс один раз — это нас устраивает. Наберите данный скетч в Arduino IDE:

```
/*
Робот mBot ждет нажатия конпки и движется по квадрату
(c) Sergey Kosachenko
*/
//подключаем библиотеку Makeblock
#include "MeMCore.h"

//кнопка подключена к pin A7
#define pinButton A7

//определяем двигатели
MeDCMotor motorLeft(M1); //левый двигатель
MeDCMotor motorRight(M2); //правый двигатель

//Скорость моторов 100
uint8_t motorSpeed = 100;

//статус нажатия кнопки
uint8_t keyPressed;

void setup()
{
  Serial.begin(9600); //Инициализация последовательного порта
  Serial.println("Waiting...");

  pinMode(pinButton, INPUT); //пин подсоединения кнопки

  //считать показания кнопки
  // 255 - не нажата, 0 - нажата
  keyPressed = analogRead(pinButton);
```

```
  //цикл ожидания нажатия кнопки
  while (keyPressed > 100)
  {
    //считать показания кнопки
    // 255 - не нажата, 0 - нажата
    keyPressed = analogRead(pinButton);
    delay(10); //пауза 0.1 секунды
   }
   Serial.println("KEY pressed! Start!");
}

void loop()
{
 //ДВИЖЕНИЕ РОБОТА ВПЕРЕД
  //вперед левый мотор
  motorLeft.run(-motorSpeed);
  //вперед правый мотор
  motorRight.run(motorSpeed);

  delay(2000); //ждать 2 секунды пока робот едет вперед

  //стоп моторы!
  motorLeft.stop();
  motorRight.stop();
  delay(500); // подождем 0,5 секунды пока робот остановится

  //поворот налево примерно на 90 градусов на месте
  motorLeft.run(motorSpeed); //левый мотор назад
  motorRight.run(motorSpeed); //правый мотор вперед

  delay(700); //время для поворота

  //стоп моторы!
  motorLeft.stop();
  motorRight.stop();
  delay(500); // подождем 0,5 секунды пока робот остановится
}
```

Сохраните программу на компьютере под названием «mBot_button_krvadrat».

Как это работает

Несмотря на предельную простоту программы в mBlock, эта же программа в Arduino IDE выглядит несколько сложнее.

Для указания номера pin, к оторому подключена кнопка, мы использовали директиву переопределения, при компиляции программы везде, где встретится слово «pinButton» будет подставлено «A7». Это сделано для того, чтобы по названию «pinButton» нам было легче догадаться, что подключено к данному пину:

```
//кнопка подключена к pin A7
#define pinButton A7
```

Мы добавили переменную keyPressed, в которой будем хранить статус нажатия на кнопку:

```
//статус нажатия кнопки
uint8_t keyPressed;
```

В процедуре void setup() мы впервые применили передачу данных по последовательному порту, которые при работе mBot можно будет получать от робота на персональном компьютере по USB-кабелю и наблюдать в Мониторе последовательного порта:

```
Serial.begin(9600); //Инициализация последовательного порта на скорости 9600 бод
Serial.println("Waiting..."); //отправить на ПК сообщение об ожидании
```

Переключаем режим порта (pin), к которому подключена кнопка, в режим ввода (INPUT), чтобы считывать с него данные о нажатии на кнопку:

```
  pinMode(pinButton, INPUT); //пин подсоединения кнопки
```

Считываем статус нажатия на кнопку и присваиваем его переменной keyPressed командой analogRead(), которая считывает оцифрованные 10-битные данные с порта, к которому подключена кнопка, выполняющая в этот момент роль датчика касания:

```
  //считать показания кнопки
  // 255 - не нажата, 0 - нажата
  keyPressed = analogRead(pinButton);
```

После этого запускаем цикл ожидания нажатия на кнопку, условием выполнения которого является истинность сравнения «keyPressed > 100». Пока кнопка не нажата в keyPressed будет значение 255 и цикл будет продолжать работать, как только кнопка будет нажата, то команда analogRead(pinButton) передаст в переменную keyPressed значение 0, условие станет ложным и цикл закончится, а робот перейдет к выполнению к следующей за циклом команде:

```
  //цикл ожидания нажатия кнопки
  while (keyPressed > 100)
  {
    //считать показания кнопки
    // 255 - не нажата, 0 - нажата
```

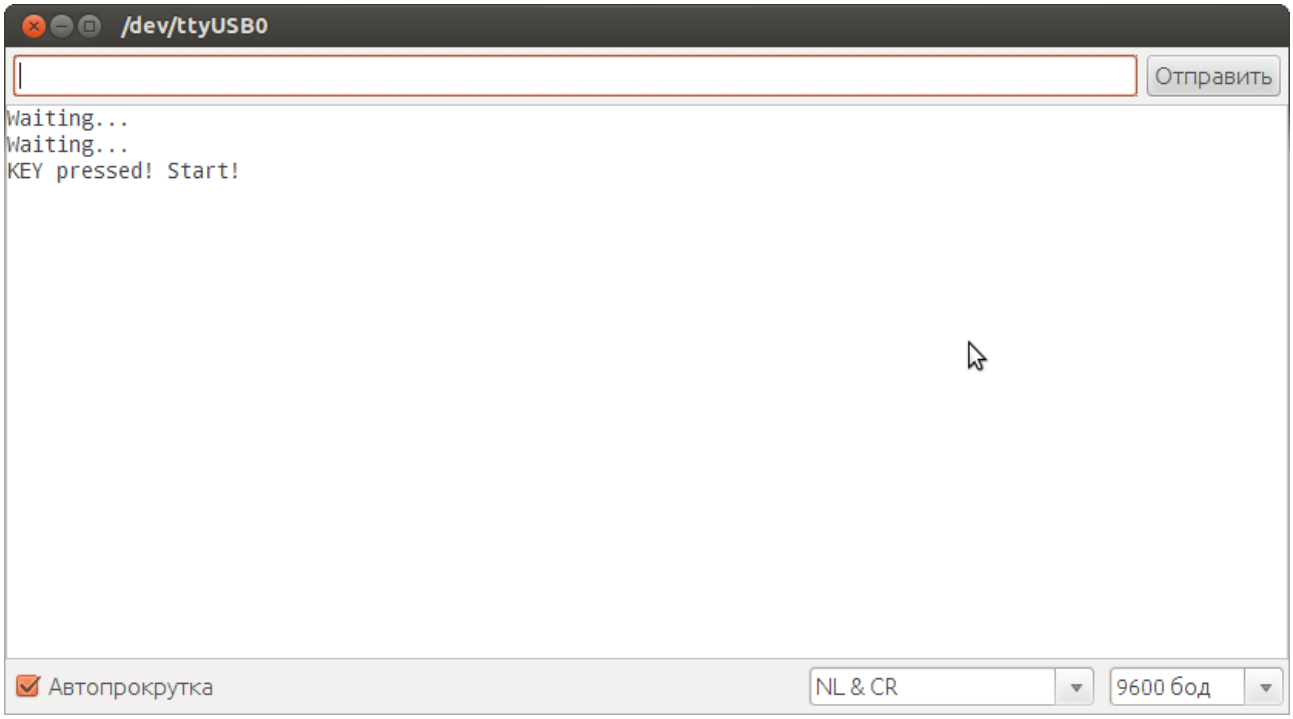
```
    keyPressed = analogRead(pinButton);
    delay(10); //пауза 0.1 секунды
  }
```

Следующая за циклом команда отправит на ПК сообщение о том, что кнопка нажата и робот начал движение:

```
  Serial.println("KEY pressed! Start!");
```

Остальные команды в процедуре void loop() мы оставили без изменения так же, как в предыдущей главе, где они объясняются.

После успешной загрузки программы, не отсоединяйте USB-кабель от робота и установите его на поверхность, по которой он будет ездить. В Arduino IDE откройте «Инструменты->Монитор порта», чтобы видеть получаемые от робота сообщения. Для начала движения нажмите кнопку на роботе, и mBot сразу начнет движение по квадрату. При этом в окне Монитора порта вы увидите сообщения, получаемые от робота:



Вопросы на внимательность

Можно ли запрограммировать робота так, чтобы нажатие на кнопку останавливало движение робота, т.е. ставило его «на паузу», а повторное нажатие на кнопку продолжало его движение по квадрату, т.е. снимало «с паузы»?

Полезно ли передавать от робота на ПК данные по USB-кабелю?

Источники информации

1. Example Codes mBot's main board is mCore https://docs.google.com/document/d/16uXDUmgN_9jM2sp_KGJtZZfQTpQ2-PzLDtjUFla_FcA/edit
2. Makeblock-library-for-Arduino V3.2.4 http://learn.makeblock.com/en/Makeblock-library-for-Arduino
3. Скачать библиотеку MakeBlock для Arduino IDE https://codeload.github.com/Makeblock-official/Makeblock-Libraries/zip/master
4. GitHub библиотека MakeBlock https://github.com/Makeblock-official/Makeblock-Libraries
5. Скачать программное обеспечение для программирования mBot http://www.mblock.cc/download/
6. Информационные материалы по mBot на русском языке https://yadi.sk/d/QHmzeMj13Mmy4p/mBot
7. Ответы на часто задаваемые вопросы по mBot http://learn.makeblock.com/en/mbot-faq/
8. Драйвера для mBot https://raw.githubusercontent.com/Makeblock-official/Makeblock-USB-Driver/master/Makeblock_Driver_Installer.zip
9. Справочник на русском языке по программированию в Arduino IDE