MINICAM24

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Набор для моделирования Ардуино Learning Kit с платой ATmega328 LY-F2

СОДЕРЖАНИЕ

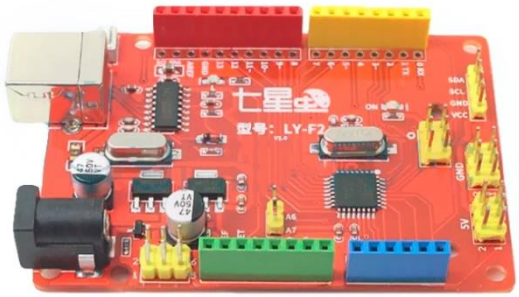
1. Чтение инструкции

Данная инструкция предназначена для использования в качестве вспомогательного материала. Нет необходимости прочитывать полностью весь написанный текст. Воспользуйтесь оглавлением, чтобы выбрать нужный пункт.

2. Обзор компонентов

2.1 Основная плата

ATmega328 LY-F2



Контроллер / плата / микропроцессор Arduino UNO R3 считывает показания датчиков, управляет исполнительными механизмами (актуаторами) и выводит результат для пользователей.

2.2 Расширительная плата Arduino

Расширительная плата или Shield, шилд отвечает за дополнительные возможности устройства. В данном наборе используется расширительная плата AS008.

2.3 Датчики Arduino

Сенсоры**,** или по-другому, датчики, помогают Arduino воспринимать окружающий его мир.

В данный набор включен большой комплект датчиков, охватывающий полный список параметров окружающей среды: здесь есть, например, датчик пламени, влажности, Холла и так далее.

2.4 Исполнительный механизм Arduino

Исполнительные механизмы или актуаторы воздействуют на окружающую среду. Различные моторы, насосы, светодиоды, воздушные винты и так далее.

2.5 Соединения

Провода и кабели обеспечивают подключение составных частей системы Arduino между собой.

2.6 Электронные компоненты

Транзисторы, микросхемы, конденсаторы и резисторы помогают правильно собрать электрическую цепь Arduino, обеспечивая её надежную, бесперебойную работу.

3. Установка Arduino IDE

Интегрированная среда разработки Arduino (IDE) — это программная часть платформы Arduino. В этом проекте вы узнаете, как настроить компьютер для использования Arduino и как приступить к последующим проектам.

Программное обеспечение Arduino, которое вы будете использовать можно запрограммировать под свой Arduino для Windows, Mac и Linux. Процесс установки отличается для всех трех платформ, и, к сожалению, для установки программного обеспечения требуется определенный объем ручной работы.

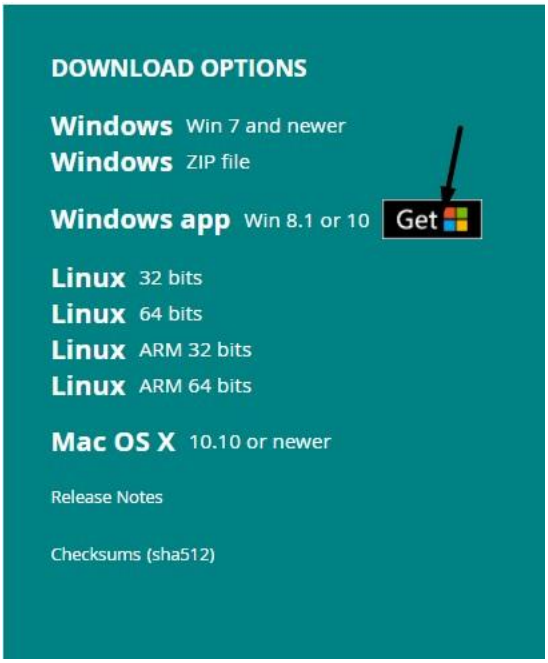
Перейдите по ссылке ниже и найдите следующую страницу:

[Software | Arduino]

Примечание:

Версия, доступная на этом веб-сайте, как правило новейшая, и реальная версия может быть новее, чем версия на фото.

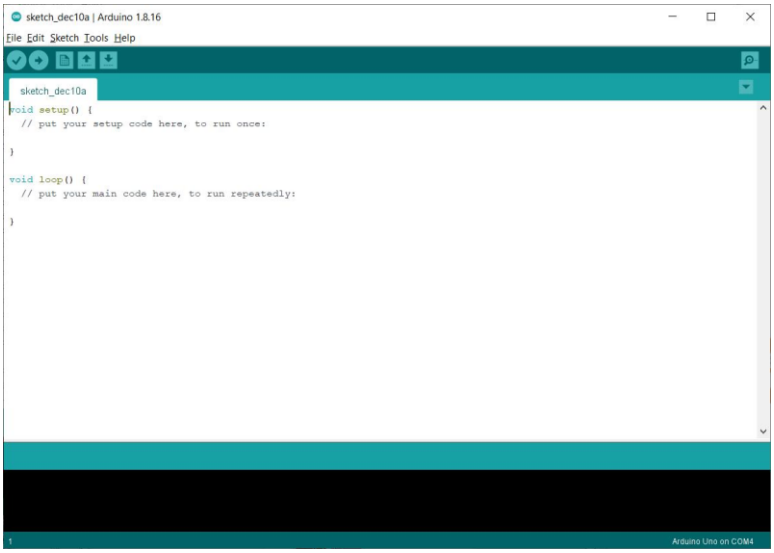
Загрузите программное обеспечение для разработки, совместимое с операционной системой вашего компьютера.



Нажмите кнопку «JUST DOWNLOAD» и выберите папку, куда будет сохранен файл. После загрузки дважды кликните на приложение “arduino-1.8.16-windows.exe”. Последовательно нажмите кнопки «I Agree» «Next» и «Install» для установки компонентов по умолчанию. Дождитесь завершения установки и нажмите «Close». Если появилось всплывающее окно от

«Windows Security» просто нажмите «Install» для продолжения. На экране появится следующая иконка 

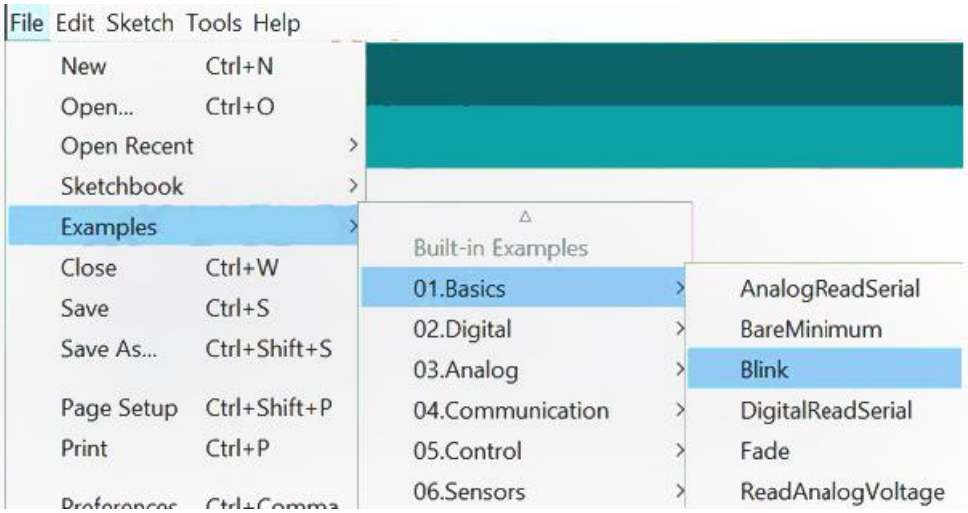
Дважды кликнете по иконке чтобы попасть в интегрированную среду разработки.



4. Как загрузить первую программу на плату

Первая программа нового языка программирования «Hello World» в мире Arduino – программа заставляющая светодиод мигать.

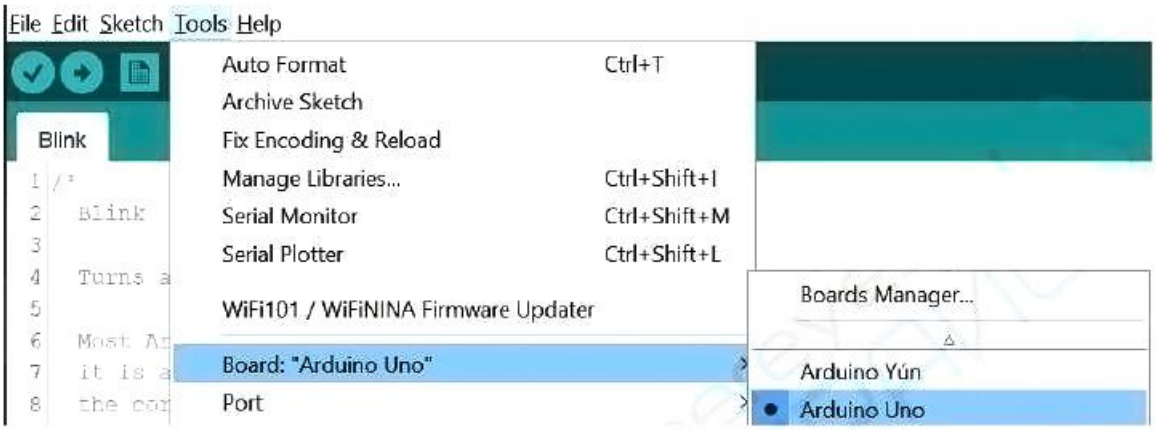
Загрузите пример в программу. Для этого пройдите по следующему пути:



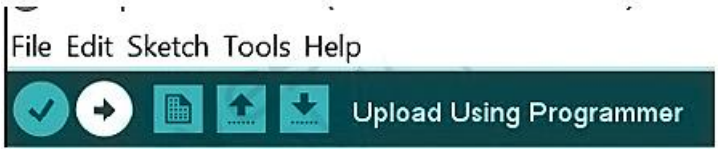
Для начала подключите Arduino R3 через USB-кабель к компьютеру. В поле Tools переходим по следующему пути

Tools -> Board: “Arduino Uno” -> Arduino Uno

Если в наборе другая плата, выберите её.



Чтобы скомпилировать программу, нажмите иконку с галочкой, если необходимо не только скомпилировать программу, но и загрузить её на плату, нажмите кнопку со стрелкой влево.



5. Программирование на C++

Данный раздел предназначен для людей, кто никогда не программировал на данном языке.

5.1 Структура программы

Каждая программа для Arduino состоит из двух главных частей-функций:

Void setup и void loop.

```
void setup()
{
  // put your setup code here, to run once:
}
```

```
void loop()
{
  // put your main code here, to run repeatedly:
}
```

void setup() - выполнится один раз. В неё обычно засовывают подготовку перед исполнением программы. Инициализация датчиков и переменных, предстартовая проверка - это всё стоит включить в **setup**.

void loop() - будет выполняться снова и снова, пока не сядет батарейка, или плата не будет перезагружена. В ней пишется основная программа.

Не обязательно писать и в теле **void setup()**, и в теле **void loop()**. Если нужна программа, которая исполняется один раз, оставляйте **void loop()** пустым. Если нужна программа, которая исполняется циклично, оставляйте **void setup()** пустым.. Например:

```
void setup()
{
  Serial.begin(9600);
  Serial.println("Hello");
}

void loop()
{
}
```

5.2 Переменные

Переменные используются для хранения значений (sic!). Переменная характеризуется типом и именем. В си переменная не может начинаться с числа. Переменная может включать в себя символы английского алфавита, цифры и знак подчёркивания. Переменная не должна совпадать с ключевыми словами (это специальные слова, которые используются в качестве управляющих конструкций, для определения типов и т.п.)

```
int a = 3;
```

Текст выше задает переменной a с типом int, значение числа 3.

```
int a = 5;
int b = a;
```

Текст выше задает переменной с именем a и типом int значение числа 5. Переменной с именем b и типом int задает значение переменной a.

Особенностью С-подобных языков программирования является статическая типизация. У каждой переменной есть свой тип значений, которые можно

добавить в эту переменную. Посмотреть доступные типы переменных можно здесь, в разделе Variables.

5.3 Точка с запятой

Команды нужно заканчивать точкой с запятой. Если не добавить соответствующий знак, Arduino IDE ласково выведет “expected ';' before...”.

```
int a = 3;
Serial.begin(9600);
delay(1000);
digitalWrite(LED_BUILTIN, LOW);
```

5.4 Комментарии

В тексте программы можно оставлять комментарии. Комментарии никак не используются компьютером и нужны только как пометка человеку. С помощью комментариев можно объяснить непонятный код, оставить напоминание или оставить послание. Писать комментарии к коду - полезная привычка.

```
/*
 Многострочный комментарий
*/

// Однострочный комментарий
// int a = 10; - это тоже комментарий. Переменная не будет создана
```

5.5 Методы Arduino

В языке Arduino добавили методы, которые помогают работать с контроллером:

```
// Попросить пин PIN работать в режиме входа
pinMode(PIN, INPUT);

// Считать показание на пине PIN. (1 или 0)
```

```
int digital_result = digitalRead(PIN);
// Установить LOW на пине PIN
digitalWrite(PIN, LOW);

// Считать показание на пине PIN. (0 - 1023)
int analog_result = analogRead(PIN);

// Ничего не делать 1 секунду (1000 мс)
delay(1000);
```

Методы можно найти здесь.

5.6 Serial monitor

В Serial Monitor можно смотреть данные, которые плата передает на компьютер и отправлять команды на устройство.

Чтобы передать данные в Serial Monitor, сначала нужно открыть соединение с помощью **Serial.begin(speed_in_bauds)**, а потом воспользоваться функциями **Serial.print()** или **Serial.write()** для отправки данных. Чтобы считать данные, используйте функцию **Serial.read()**.

- **Serial.begin(speed_in_bauds)** - откроет Serial соединение со скоростью передачи символов в speed_in_bauds символов в секунду. Бод (Baud) — это количество символов, которые пересылаются за секунду. В данном случае один символ — это 8 бит, скорость передачи в битах в секунду — это speed_in_bauds, умноженная на восемь.
- **Serial.print()** - отправит данные в читаемом ASCII формате
- **Serial.write()** - отправит данные в бинарном формате

Пример

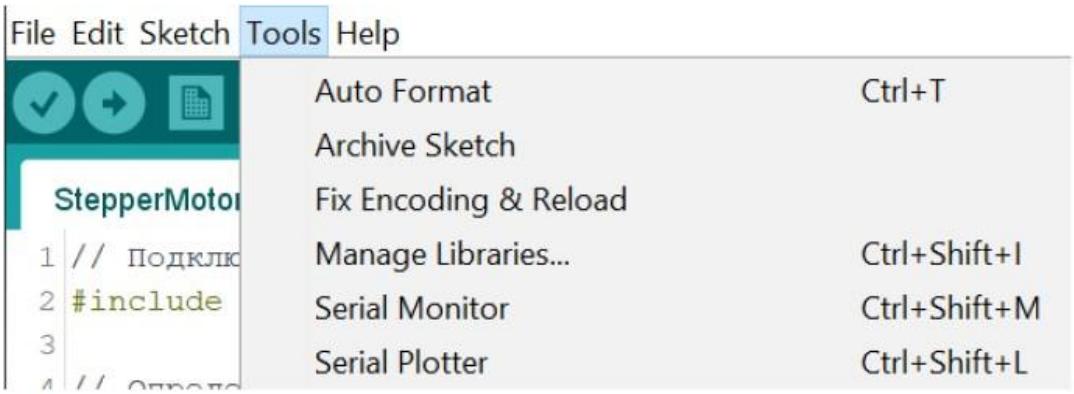
```
// Запускаем соединение со скоростью 9600 бод
Serial.begin(9600);
// Отправляем сообщение в Serial Monitor
Serial.println("Yay, new Serial Connection!");

// Проверяем, появились ли новые данные
if (Serial.available() > 0)
{
  // Считываем байт данных
  byte data = Serial.read();
  // Выводим данные в десятичном формате
  Serial.print("New data byte: ");
  Serial.println(data , DEC);
}
```

Чтобы открыть Serial Monitor, нажмите кнопку в правом верхнем углу или нажмите Ctrl+Shift+M

В правом нижнем углу Serial Monitor можно установить скорость соединения. Если скорости в Serial.begin() и в окне Serial Monitor не будут совпадать, отображение данных не будет корректным. Потому что плата и компьютер ожидают разное количество символов в один и тот же промежуток времени.

С помощью **Serial Plotter**-а можно визуализировать данные, которые приходят в **Serial Monitor**, а если возможностей Plotter-а недостаточно, то можно перейти в **Processing.**

Чтобы открыть **Serial Plotter** нажмите **Ctrl+Shift+L** или перейдите в меню Tools.



5.7 Ветвление

Функция ветвления применяется, когда нужно выполнять разные действия в зависимости от условия. Например:

```
int a = digitalRead(13);

if (a > 10)
{

}
else if (a < 10 && a >= 5)
{

}
else
{

}
```

Более сложные условия в if можно создавать с помощью логических операторов, например: || - логическое или, && - логическое и, ! - логическое не и других.

Информацию про логические операторы можно найти тут в разделе Structure.

5.8 Циклы

Часто нужно несколько раз запустить одни и те же строчки кода. Чтобы не копировать одно и то же, можно использовать циклы.

Например, чтобы вывести надпись Hello в консоль можно сделать так:

```
Serial.begin(9600);
Serial.println("Hello");
Serial.println("Hello");
Serial.println("Hello");
```

Или можно использовать цикл.

- Цикл for
  For пригодится когда мы точно знаем сколько раз нужно повторять действие.

```
Serial.begin(9600);
for(int i = 0; i < 10; i++)
{
  Serial.println("Hello!");
}
```

Циклу for нужна своя переменная-счётчик, которая будет изменяться в соответствии с правилом. Цикл for будет работать пока условие верно.

```
for(переменная; условие выхода; правило изменения)
{
  // Полезный код
}
```

переменная - здесь мы создаём переменную-счётчик. Например: int i = 0; **условие выхода** - когда условие не выполнится, цикл прекратится. Например: i < 3;

правило изменения - правило, по которому переменная-счётчик будет изменяться в конце каждой итерации цикла. Например: i *= 2;

- Цикл while
  Что делать если мы не знаем сколько раз нам понадобится цикл? Используйте цикл while если вы не знаете сколько раз вам понадобится цикл.

```
while(условие)
{
  // Полезный код
}
```

условие - когда условие не выполнится, цикл прекратится.

Цикл for - это преобразованный цикл while. Как он выглядит в процессе компиляции можно увидеть ниже:

```
int i = 0;

while(i < 10)
{
   i++;
}
```

- Цикл do while

Do while сначала сделает действие, а потом проверит условие. Полезно, когда действие нужно сделать перед проверкой условия цикла.

```
do
{
  // Полезный код
} while (условие);
```

условие - когда условие не выполнится, цикл прекратится.

6. Функции

Функции — это блоки кода, выполняющие определенные операции. Если требуется, функция может определять входные параметры, позволяющие вызывающим объектам передавать ей аргументы. При необходимости функция также может возвращать значение как выходное. Функции полезны для инкапсуляции основных операций в едином блоке, который может многократно использоваться. В идеальном случае имя этого блока должно четко описывать назначение функции.

```
bool usefulFunction(int arg1, float arg2)
{
bool result = false;

// Полезный код

// Возвращаем результат
return result;
}

// Вызываем одну и ту же функцию на разных входных данных
bool res1 = usefulFunction(1, 1.0);
bool res2 = usefulFunction(10, -3.0);
bool res3 = usefulFunction(0, 55.0);
```

Код выше создаст функцию с именем **usefulFunction**, которая принимает на вход два **аргумента arg1** и **arg2**, типа **int** и **float** соответственно, делает полезную работу и возвращает результат в виде значения переменной типа **bool**. Потом эту функцию можно использовать много раз.
Функции могут ничего не принимать и не возвращать. Например:

```
void function()
{
  // Полезный код
}

// Два раза вызываем функцию
function();
function();
```

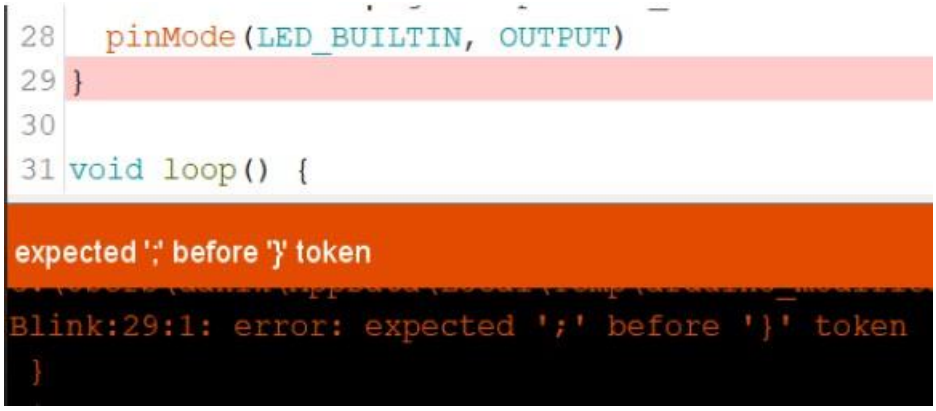
7. Использование библиотек для Arduino

Система Arduino позволяет писать скетчи самому и сохранять их в библиотеки. Вы можете использовать чужую библиотеку, чтобы не пришлось писать весь код «с нуля». Добавить библиотеку в программу можно с помощью команды #include, например:

```
#include <Servo.h>
// Теперь можно использовать класс Servo из библиотеки Servo.h
Servo myservo;
```

8. Ошибки компиляции

В процессе компиляции могут быть обнаружены ошибки в написанном вами коде. В таком случае строка будет подсвечена красным, а в сообщении ниже будет описана ошибка.



В данном случае программа пишет, что в коде допущена ошибка: перед фигурной скобкой должна стоять точка с запятой.

9. Использование --, --, +=

Команды для оптимизации умножения, вычитания и сложения.

```
int a = 10;
// Все три строчки ниже увеличат a на 1
a = a + 1;
a += 1;
a++;

// Все три строчки ниже вычтут 1 из a
a = a - 1;
a -= 1;
a--;

// Обе строчки ниже умножат a на 10
a = a * 10;
a *= 10;
```

10. #define

Часто нужно добавить какое-нибудь постоянное значение в нескольких местах, но если вдруг понадобится изменить, можно использовать макрос #define:

```
#define ИМЯ ЗНАЧЕНИЕ;
```

В процессе компиляции, препроцессор заменит все объявленные имена на значения.

```
// До препроцессора
#define VALUE 10;

int a = VALUE;
int b = VALUE;

// После препроцессора
int a = 10;
int b = 10;
```

Чтобы различать переменные и #define, имена в #define принято писать заглавными буквами. Данное действие не влияет на функционал, но желательно, тогда ваш код смогут сразу правильно понять.

Приятного использования!

Сайт: **minicam24.ru**
E-mail: info@minicam24.ru
Товар в наличии в 120 городах России и Казахстана
Телефон бесплатной горячей линии: **8(800)200-85-66**