



# ARDUINO и ОСНОВЫ УПРАВЛЕНИЯ

# Содержание

□ Arduino

□ Основы управления

- последовательный ввод / вывод
- Основы управления аппаратным обеспечением

# Что такое Arduino

❑ Arduino - это платформа для аппаратного и программного обеспечения с открытым исходным кодом, основанная на микроконтроллерах Atmel.

: Открытый источник означает, что схемы и исходный код программного обеспечения, используемого в проектах, свободно доступны и могут быть изменены энтузиастами.

- Аппаратное обеспечение Arduino (одноплатный микроконтроллер)

: Микроконтроллер построен на одной печатной плате, которая обеспечивает все схемы, необходимые для полезной задачи управления.

- Программное обеспечение Arduino (IDE)

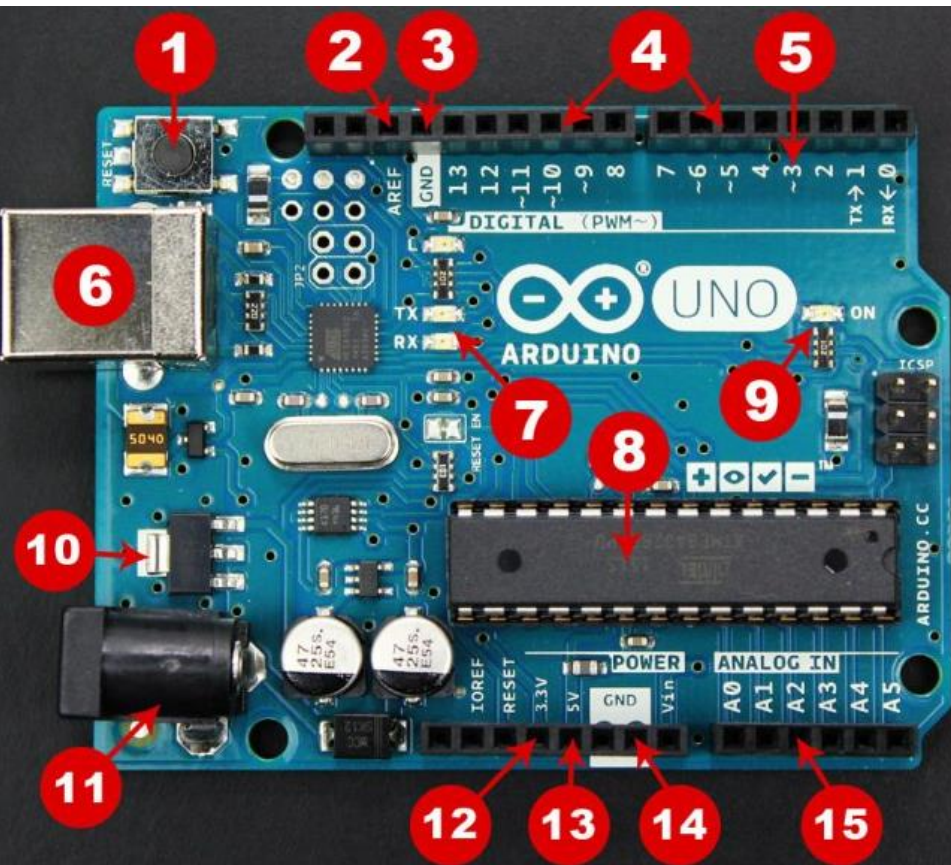
- Программное обеспечение Arduino с открытым исходным кодом (IDE) позволяет легко писать код и загружать его на борт.
- Среда разработки написана на Java и основана на обработке и другом программном обеспечении с открытым исходным кодом.

# Почему Arduino

- ❑ **недорогой:** платы Arduino относительно недороги по сравнению с другими платформами микроконтроллеров.
- ❑ **Кросс-платформенная:** программа Arduino (IDE) работает в операционных системах Windows, Macintosh OSX и Linux.
- ❑ **Простая и понятная среда программирования:** программное обеспечение Arduino (IDE) прост в использовании для новичков, но достаточно гибкое для продвинутых пользователей
- ❑ **Открытое и расширяемое оборудование:** планы плат Arduino публикуются под лицензией Creative Commons, поэтому опытные дизайнеры могут создавать собственную версию модуля, расширяя и улучшая его.
- ❑ **Открытое и расширяемое программное обеспечение:** программное обеспечение Arduino публикуется как инструменты с открытым исходным кодом, доступные для расширения опытными программистами. Язык может быть расширен через библиотеки C +

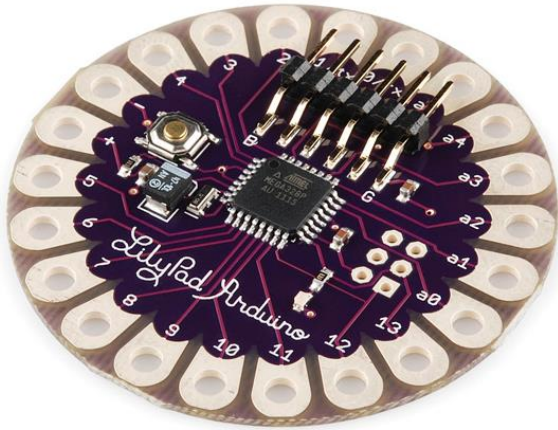


# плата микроконтроллера Arduino

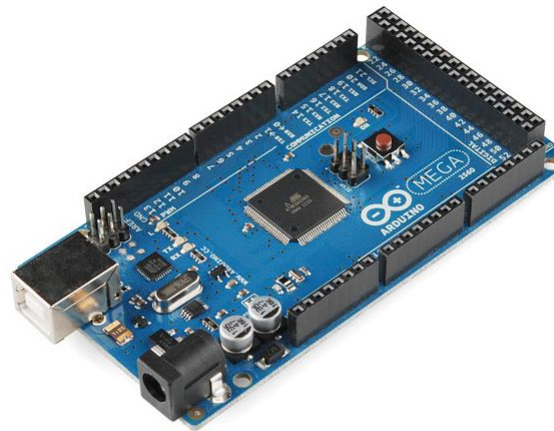


1. **Кнопка сброса:** перезагрузка любого загруженного на плату кода
2. **AREF:** Стенды для «Analog Reference» и используется для установки внешнего опорного напряжения
- 3, 14: **контакт заземления**
4. **Цифровой вход / выход:** контакты 0-13 могут использоваться для цифрового входа или выхода
5. **PWM:** пины отмеченные символом (~), могут имитировать выход PWM
6. **USB-соединение:** используется для включения питания и загрузки программного обеспечения
7. **TX / RX** - Светодиодные индикаторы передачи и приема данных
8. **ATmega Microcontroller:** это мозг и где хранятся программы
9. **Светодиодный индикатор питания**
10. **Регулятор напряжения:** контролирует количество напряжения, поступающего на плату Arduino
11. **Разъем питания постоянного тока:** для питания платы с источником питания
12. **3.3V Pin:** обеспт 3,3 вольт мощности для проекто
13. **5V Pin:** обесп.т 5 вольт мощности для проектов
15. **Аналоговые выходы:** могут считывать сигнал с аналогового датчика и преобразовывать его в цифровой

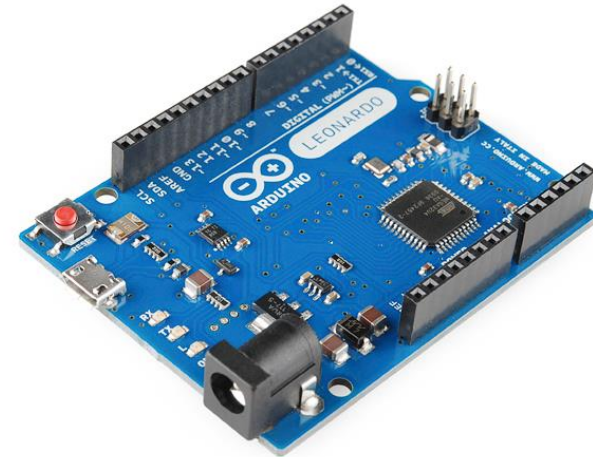
# Семейство Arduino



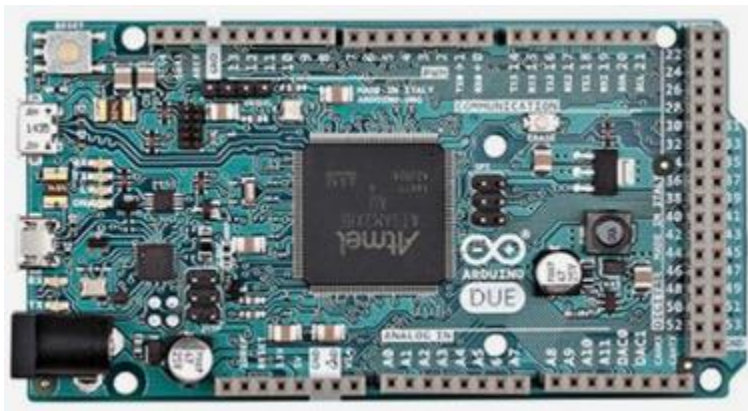
LilyPad Arduino



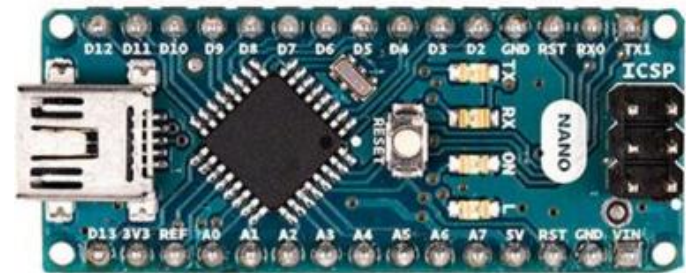
Arduino Mega (R3)



Arduino Leonardo



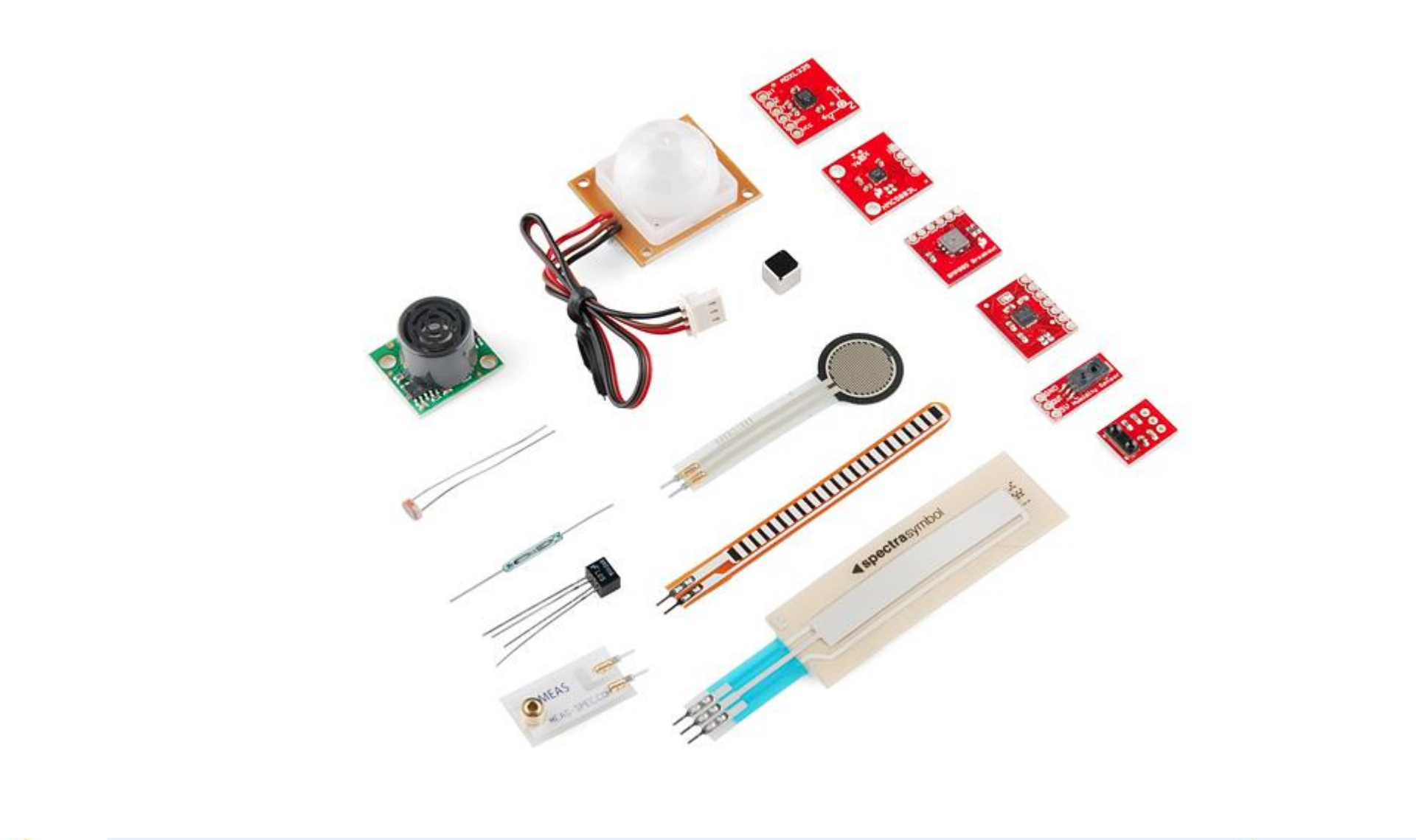
Arduino Due



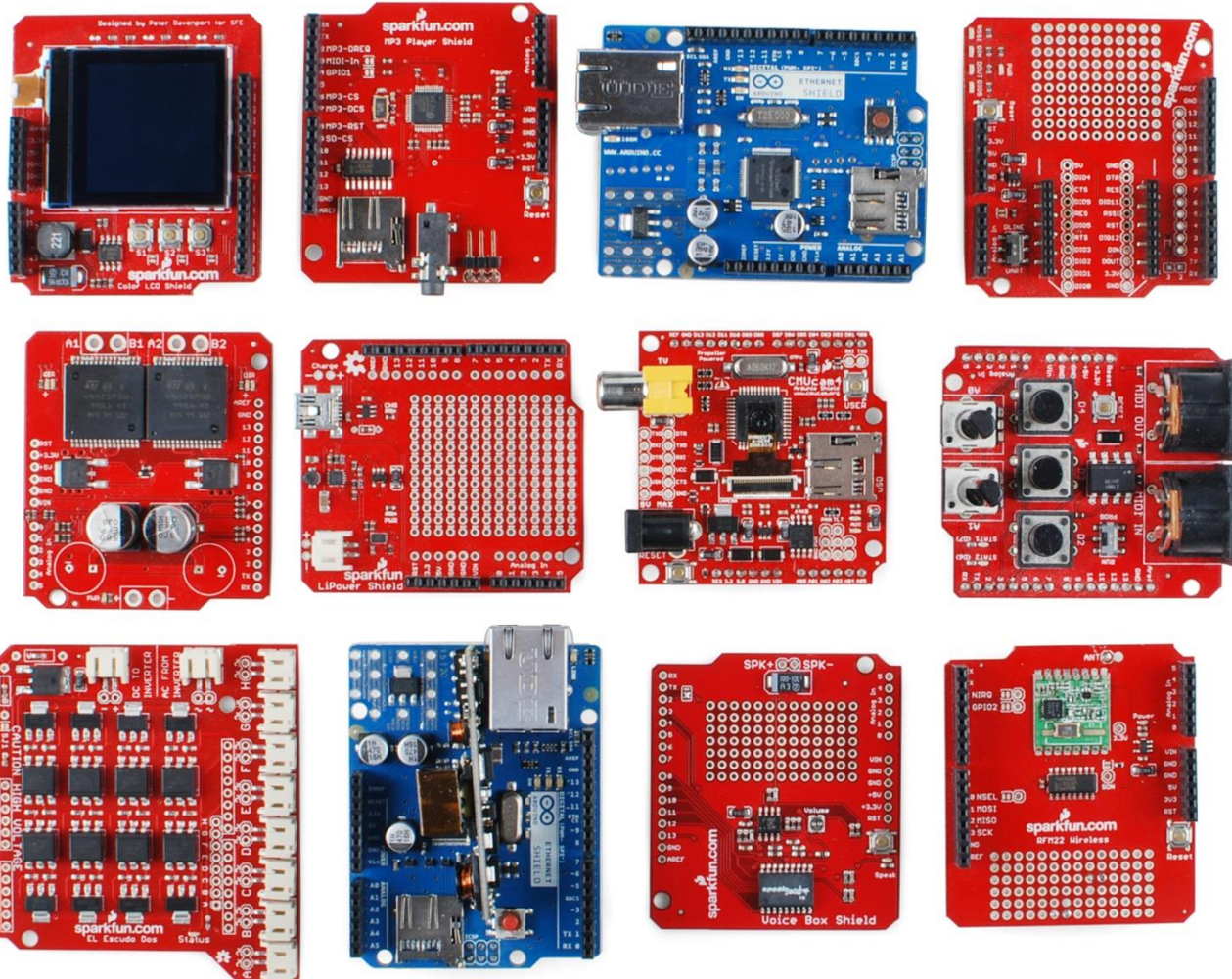
Arduino Nano



# Расширенные семейные датчики



# Расширенное семейство - щиты Shields





# Проекты с использованием Arduino UNO



Доступ к безопасности с помощью RFID-считывателя



Arduino UNO & Genuino UNO

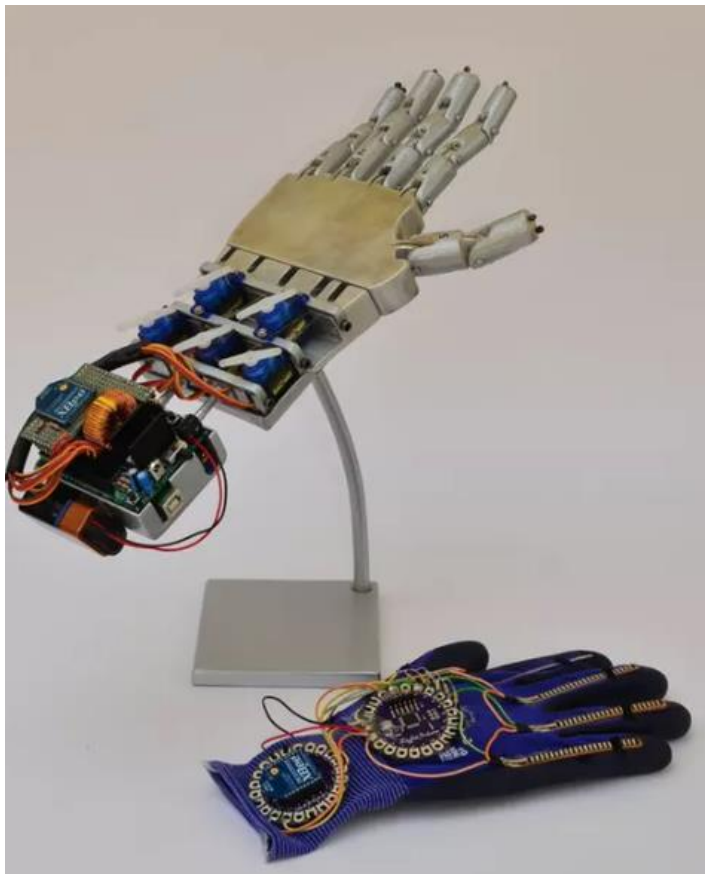
## Arduino code for RFID reader Arduino

In the piece of code above you need to change the if (content.substring(1) == "REPLACE WITH YOUR UID") and type the UID card you've written previously.

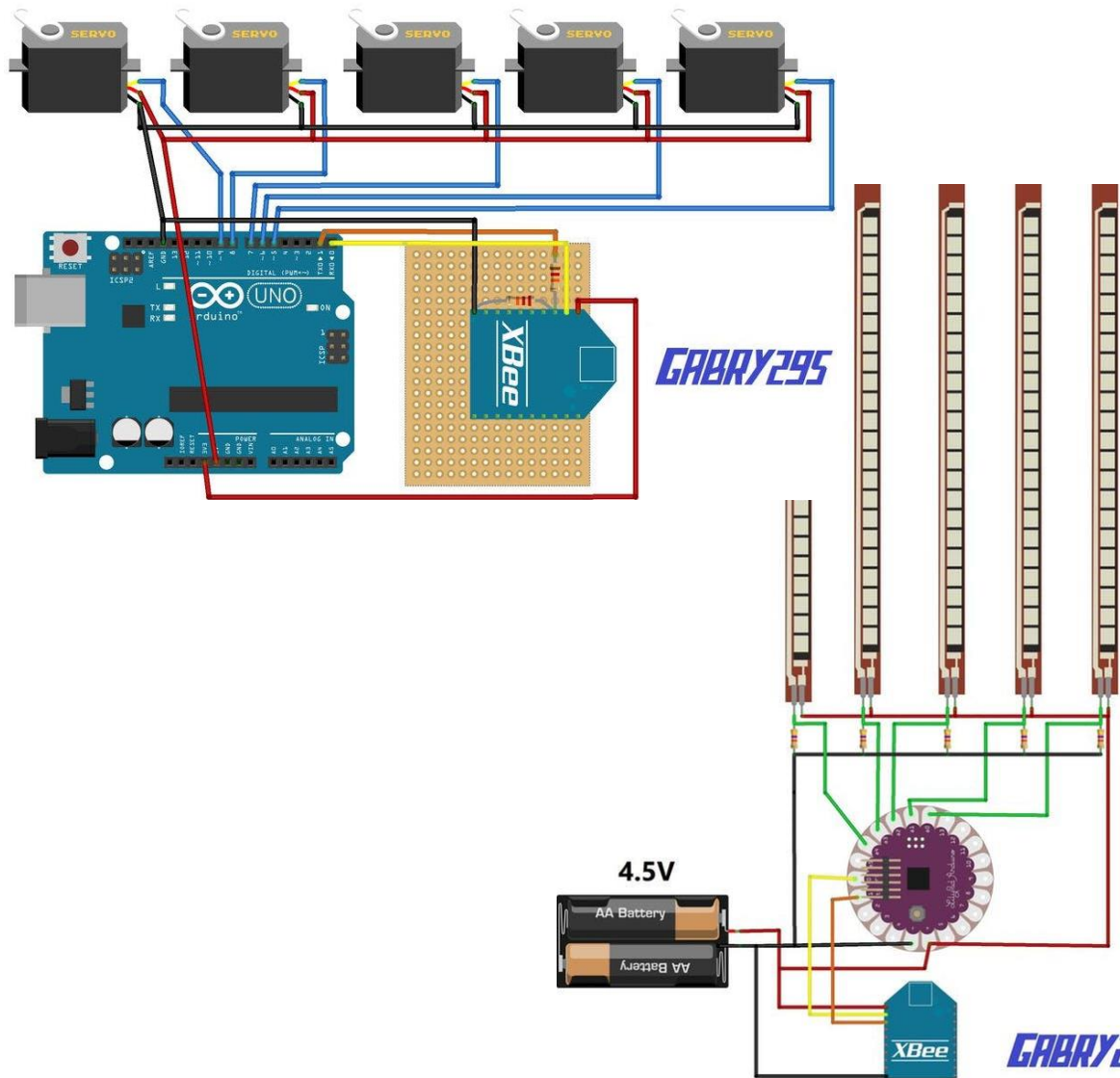
```
1  /*
2  *
3  * All the resources for this project: https://www.hackster.io/Aritro
4  * Modified by Aritro Mukherjee
5  *
6  *
7  */
8
9  #include <SPI.h>
10 #include <MFRC522.h>
11
12 #define SS_PIN 10
13 #define RST_PIN 9
14 MFRC522 mfc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
15
16 void setup()
17 {
18   Serial.begin(9600); // Initiate a serial communication
```

Код Arduino для считывателя RFID

# Проекты с использованием Arduino UNO



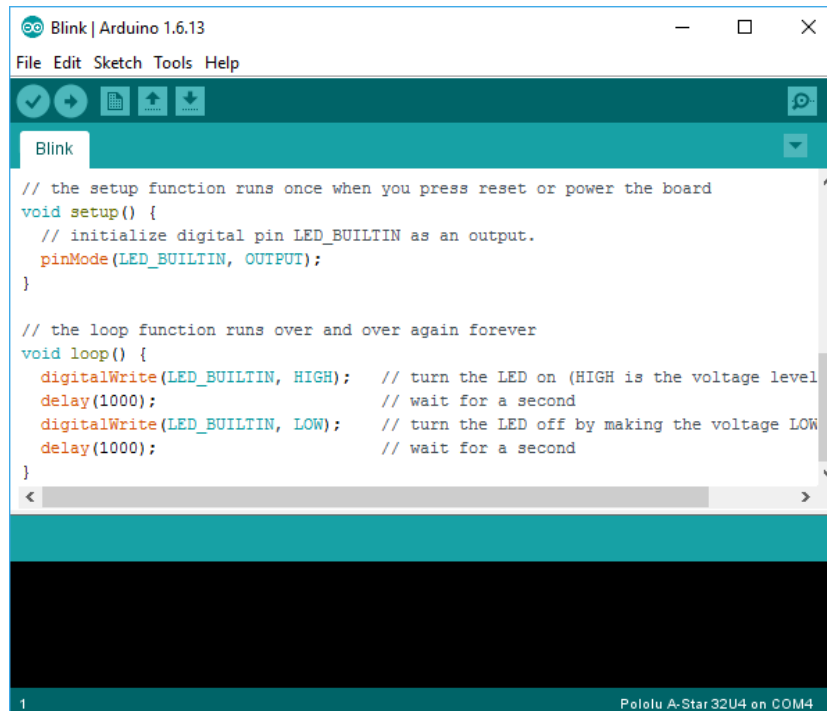
роботизированная рука с дистанционным управлением



# Программное обеспечение Arduino (IDE)

## ❑ Arduino IDE (интегрированная среда разработки)

- содержит текстовый редактор для написания кода, область сообщений, текстовую консоль, панель инструментов с кнопками для общих функций и ряд меню.
- подключается к оборудованию Arduino для загрузки программ и общения с ними



The screenshot displays the Arduino IDE window titled "Blink | Arduino 1.6.13". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for opening, saving, uploading, and downloading. The main text area shows the "Blink" sketch with the following code:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

At the bottom of the window, the status bar indicates "1" and "Pololu A-Star 32U4 on COM4".



# Программное обеспечение Arduino (IDE) - Эскизы

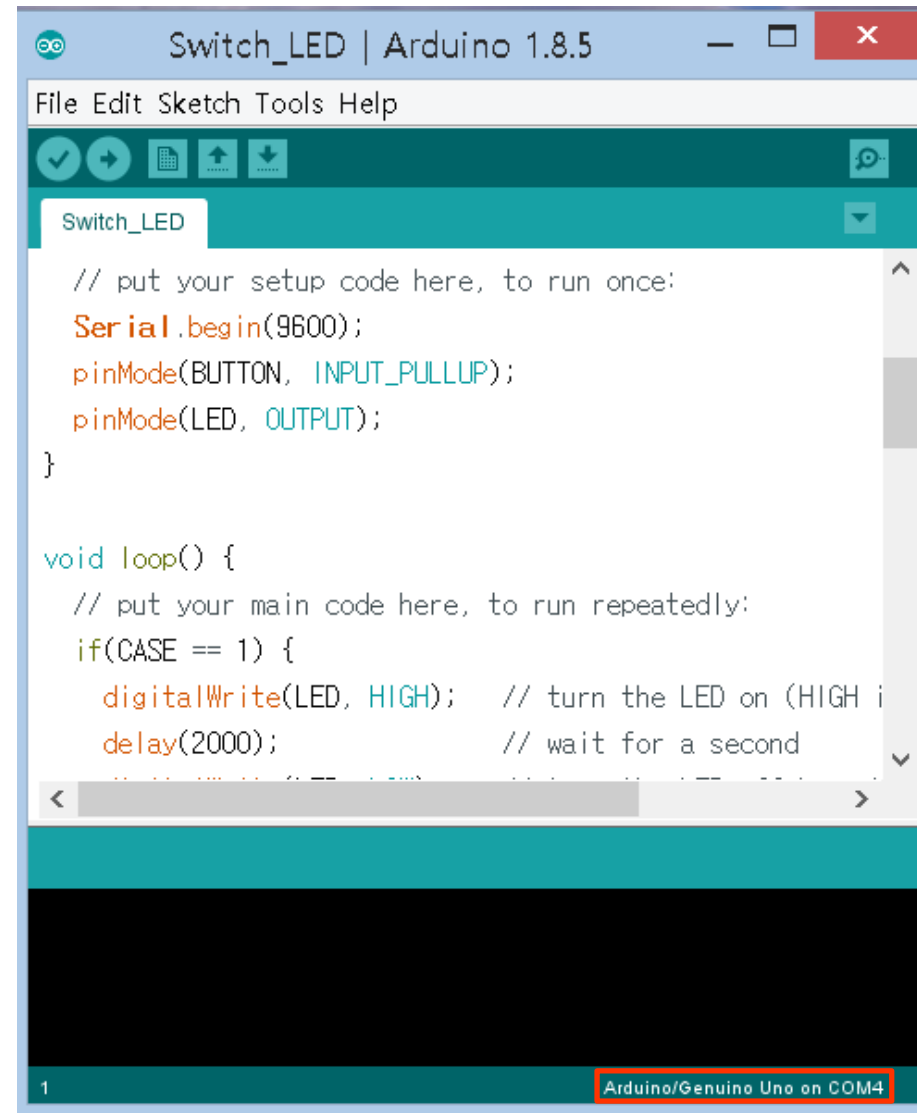
## □ Написание эскизов

- Программы, написанные с использованием Arduino Software (IDE), называются эскизами
- Эскизы записываются в текстовый редактор и сохраняются с расширением файла .ino.
- Кнопки панели инструментов позволяют проверять и загружать программы, создавать, открывать и сохранять эскизы и открывать последовательный монитор.
- ✓ Verify: проверяет ваш код на наличие ошибок.
- ➔ Upload: компилирует свой код и загружает его на настроенную плату.
- 📄 New: создает новый эскиз
- 📂 Open: представляет меню всех эскизов в вашем альбоме. Щелчок по одному откроет его в текущем окне, перезаписывая его содержимое.
- 💾 Save: сохраняет эскиз
- 🔍 Serial Monitor: открывает последовательный монитор

# Программное обеспечение Arduino (IDE) - Загрузка

## □ Загрузка

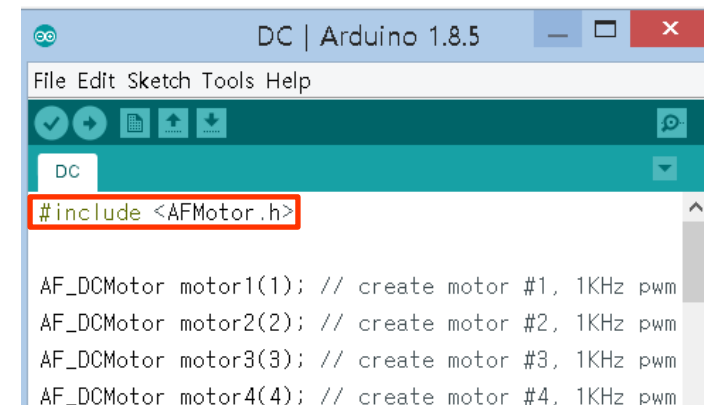
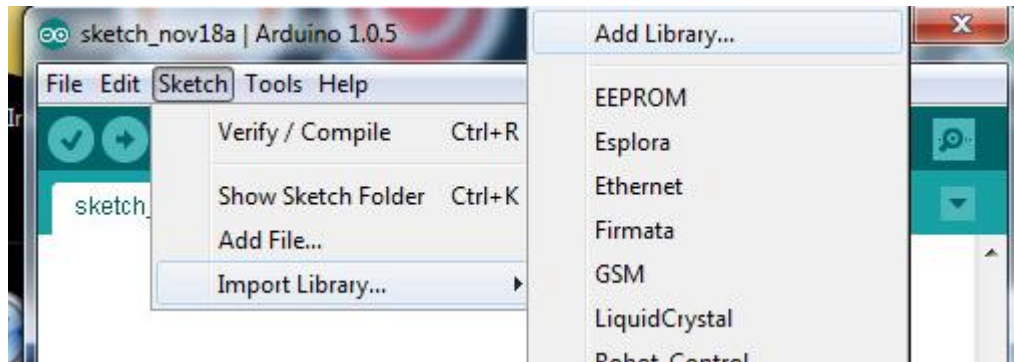
- Перед загрузкой эскиза вам нужно выбрать правильные элементы в меню Инструменты> **Board** и инструменты> **Port**
- Нажмите кнопку загрузки или выберите пункт «Загрузить» в меню «Эскиз».
- Текущие платы Arduino сбрасываются автоматически и начинают загрузку
- Когда эскиз загружается, светодиоды RX и TX мигают.
- После завершения загрузки среда IDE отобразит сообщение или покажет сообщение об ошибке



# Программное обеспечение Arduino (IDE) - Библиотеки

## □ Библиотеки

- Библиотеки предоставляют дополнительную функциональность для использования в эскизах, например. работа с оборудованием или манипулирование данными.
- Чтобы использовать библиотеку, выберите ее из меню Sketch> Import Library.
- Это вставляет один или несколько операторов **#include** в верхней части эскиза и скомпилирует библиотеку с вашим эскизом.
- Другие библиотеки можно загружать из разных источников или через Менеджер библиотек

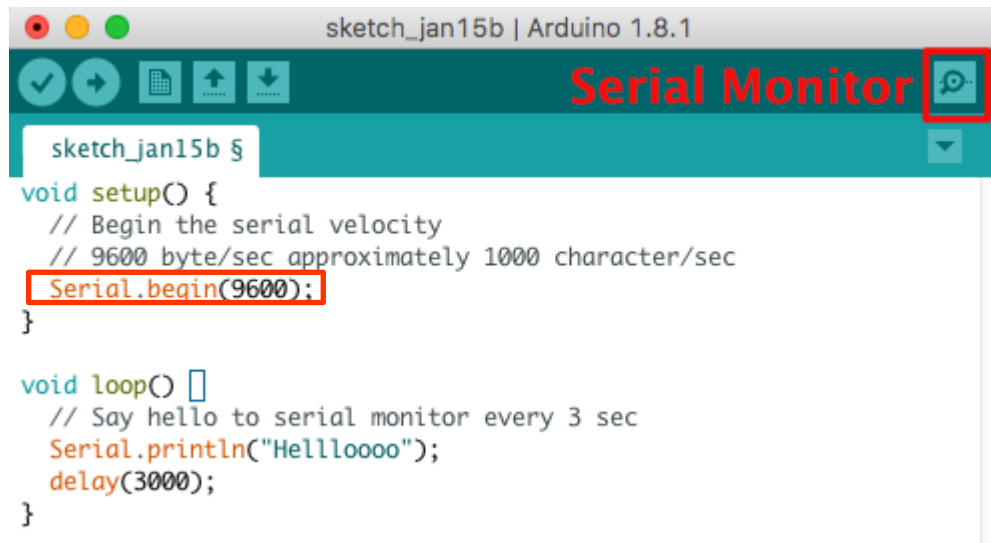




# Программное обеспечение Arduino (IDE) - последовательность

## ❑ Последовательный монитор

- Это отображает серийный номер, отправленный с платы Arduino через USB или последовательный разъем.
- Чтобы отправить данные на доску, введите текст и нажмите кнопку «Отправить» или нажмите «Ввод».
- Выберите скорость передачи в раскрывающемся меню, которая соответствует скорости, переданной Serial.begin в эскизе.



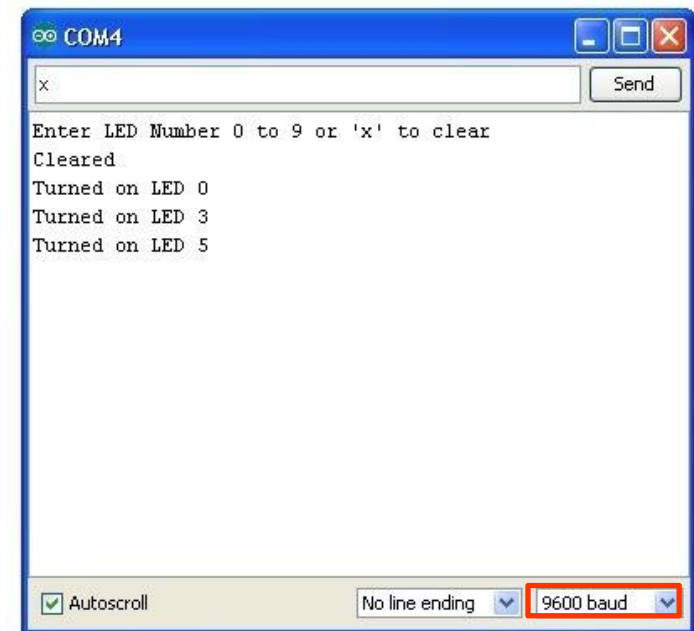
```
sketch_jan15b | Arduino 1.8.1

Serial Monitor

sketch_jan15b $

void setup() {
  // Begin the serial velocity
  // 9600 byte/sec approximately 1000 character/sec
  Serial.begin(9600);
}

void loop() {
  // Say hello to serial monitor every 3 sec
  Serial.println("Helllloooo");
  delay(3000);
}
```



# Arduino Software (IDE)-Структура

## ❑ Функция **setup( )**

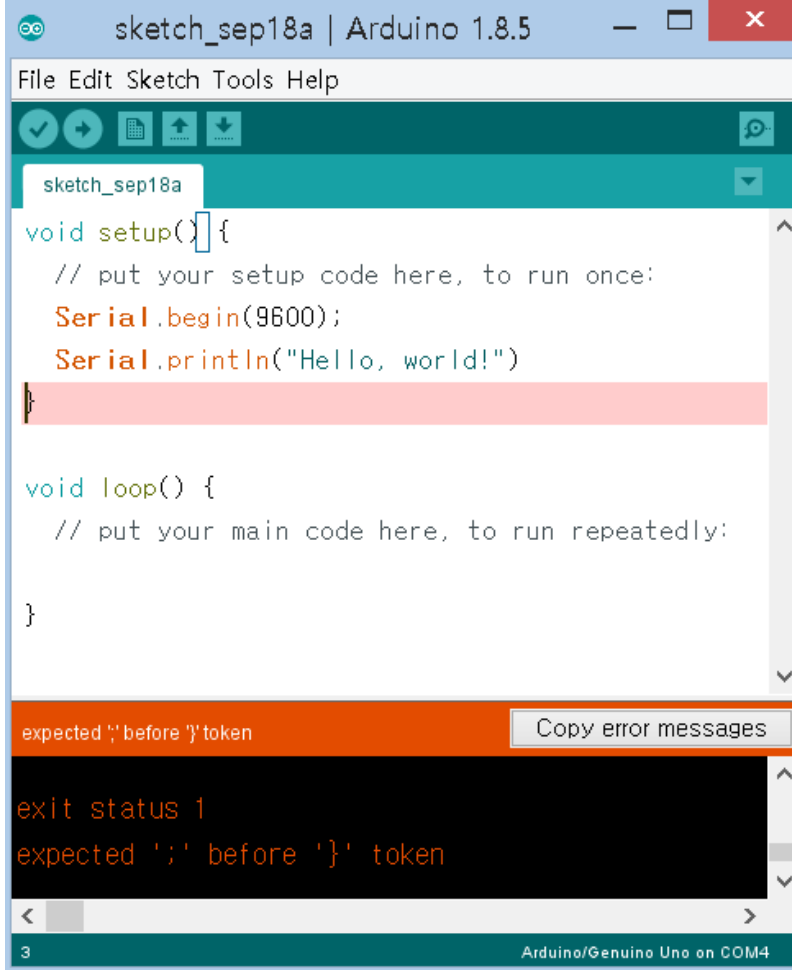
:операторы в функции setup ( ) выполняются только один раз, каждый раз, когда выполняется эскиз.

## ❑ Функция **loop( )**

: Выражения в функции loop ( ) будут выполняться непрерывно сверху вниз, а затем обратно в начало

## ❑ Текстовое сообщение от Arduino IDE

- Область сообщений дает обратную связь при сохранении и экспорте, а также отображает ошибки.
- Консоль отображает текстовый вывод, включая полные сообщения об ошибках и другую информацию



```
sketch_sep18a | Arduino 1.8.5
File Edit Sketch Tools Help

sketch_sep18a
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println('Hello, world!');
}

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

expected ';' before '}' token
Copy error messages
exit status 1
expected ';' before '}' token
3
Arduino/Genuino Uno on COM4
```

# Типы данных Arduino

Type	Keyword	Width	Description
Boolean (логическ)	bool	1bit	True(1) or False(0)
Character(символь)	char	1byte	a character value in the <a href="#">ASCII table</a>
	unsigned char		datatype for numbers from 0 to 255.
Byte(битовой)	byte	1byte	8-bit unsigned number, from 0 to 255
Integer(целочисл)	short	2byte	16-bit value, from -32768 to 32767
	int	2byte	16-bit value, from -32768 to 32767
	unsigned int	2byte	16-bit value, from 0 to 65535
Word(слово)	word	2byte	16-bit unsigned number, from 0 to 65535
Long(длина)	long	4byte	32 bits, from -2,147,483,648 to 2,147,483,647
	unsigned long	4byte	32 bits, from 0 to 4,294,967,295
Floating point(пл.то)	float	4byte	32 bits, -3.4028235E+38 ~ 3.4028235E+38
Double floating point	double	-	Arduino Uno: 4bytes, Arduino Due: 8bytes
String(строка)	string		Character array or an object of String class
Valueless(без знач)	void		no information



# Константы в Arduino

Type	Description
true	определяется как 1, Любое целое, отличное от нуля будет правд
false	Определяет как 0
HIGH	- Чтение: напряжение выше 3,0 V - Запись: вывод находится на 5 вольт
LOW	- Чтение: на выводе присутствует напряжение менее 1,5 V - Запись: вывод находится на 0 вольт
INPUT	высокоимпедансное состояние для считывания датчика. Чтобы обеспечи ть правильное считывание, когда переключатель разомкнут, необходимо использовать <u>подтягивающий или выталкивающий резистор</u>
OUTPUT	низкоомное состояние, которое может обеспечить значительное количес тво тока для других цепей
Integer Constants	Целочисленные константы - это числа, которые используются непосредс твенно в эскизе: десятичный (7), двоичный ( <b>B</b> 1111011), восьмеричный ( <b>0</b> 173), шестнадцатеричный ( <b>0x</b> 7B)
Floating Point Constants	константы с плавающей запятой используются, чтобы сделать код более читаемым: 0.005, 10.0, 2.34E5( $2.34 \cdot 10^5 = 234000$ ), 67e-10( $67.0 \cdot 10^{-10} = 0.0000000067$ )

# Преобразование типа в Arduino

- ❑ Преобразование значения в другой тип
  - **char()** : Преобразует значение в тип данных **char**
  - **byte()** : Преобразует значение в тип данных **byte**
  - **int()** : преобразует значение в тип данных **int**
  - **word()** : преобразует значение в тип данных **word**
  - **long()** : преобразует значение в тип данных **long**
  - **float()** : преобразует значение в тип данных **float**

# #1 задача: Обработка данных Arduino

## □ Task(Задание)

: объявите переменные и присвойте им значения, а затем преобразуйте значения в соответствующие типы данных, как показано ниже.

Имя переменной	Типы данных	значения	Тип преобразования
f_sw	boolean	false	byte
mark	character	'A'	word
id	byte	0x1F	character
rank	unsigned integer	123	float point
average	floating point	3.141592	integer

## □ Use Tip(Подсказка)

- Обратитесь к типам данных и константам Arduino



# Содержание

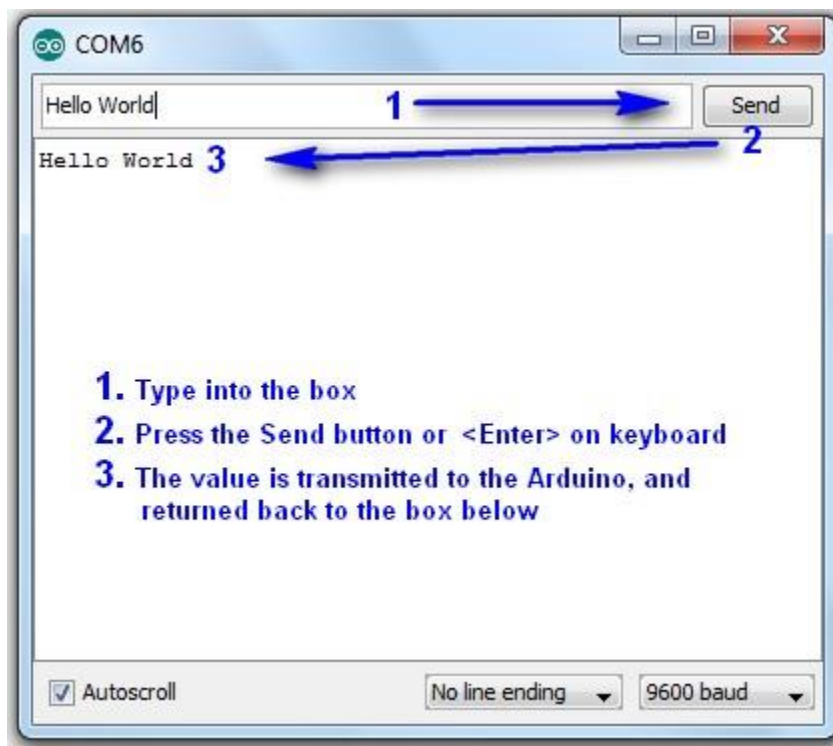
□ Arduino

□ Основы управления

- последовательный ввод / вывод
- Основы управления аппаратным обеспечением

# Последовательность в Arduino

- ❑ Используется для связи между платой Arduino и компьютером или другими устройствами. Все платы Arduino имеют как минимум один последовательный порт (также известный как UART или USART)
- ❑ Пользователь может вводить данные в поле ввода в окне последовательного монитора для отправки значений и данных в Arduino



# Последовательные функции

## ❑ Serial.begin()

: Устанавливает скорость передачи данных в битах в секунду (бод) для последовательной передачи данных

### ■ Синтаксис

#### ➤ Serial.begin(speed)

- speed: используйте одну из этих скоростей: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200

### ■ Пример

```
void setup() {  
    // открывает последовательный порт, устанавливает скорость передачи данных до 9600 бит / с  
    Serial.begin(9600);  
}  
void loop() { ... }
```

## ❑ Serial.end()

: Отключает последовательную связь. Для повторного включения вызовите Serial.begin ()

### ■ Синтаксис

#### ➤ Serial.end()

# Последовательные функции

## □ Serial.available()

: Получить количество байтов (символов), доступных для чтения из последовательного порта

### ■ Синтаксис

#### ➤ Serial.available()

- Возвращает значение: количество байтов, доступных для чтения

### ■ Пример

```
byte data = 0;                                //для входящих последовательных данных

void setup() {
    // открывает последо.й порт, устан.т скорость передачи данных до 9600 бит / с
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {              // отвечает только тогда, когда вы получаете данные
        ...
    }
```



# Последовательные функции

## ❑ Serial.read(), Serial.readBytes()

: Считывает последовательные данные

### ■ Синтаксис

- Serial.read(): считывает входящие последовательные данные
- Serial.readBytes(buffer, length): считывает символы из последовательного порта в буфер
  - buffer: буфер для хранения байтов в (char[] or byte[])
  - length: количество прочитанных байтов (int)

### ■ Пример

```
byte data = 0;                                // для входящих последовательных данных

void loop() {
  if (Serial.available() > 0) {                // отвечать только тогда, когда вы получаете данные
    data = Serial.read();                      // читать входящий байт:
```

# Последовательные функции

## ❑ Serial.print(), Serial.println()

: Распечатывает данные на последовательный порт в виде читабельного человека текста ASCII,

### ■ Синтаксис

- Serial.print(val, format)
- Serial.println(val, format): за которым следует символ возврата ('\r') и символ новой строки ('\n')
  - val: значение для печати - любой тип данных
  - format: указывает номер базы

### ■ Пример

```
Serial.print(78)           // gives "78"
Serial.print(1.23456)      // gives "1.23"
Serial.print('N')         // gives "N"
Serial.print("Hello world.") // gives "Hello world."

Serial.println(78, BIN)    // gives "1001110"
Serial.println(78, OCT)    // gives "116"
Serial.println(78, DEC)    // gives "78"
Serial.println(78, HEX)    // gives "4E"
```

# Пример кода

```
char data = 0;                                // для входящих последовательных данных

void setup() {
    Serial.begin(9600);                        // открывает порт, устан.т скоростьдо 9600 бит / с
}

void loop() {
    if (Serial.available() > 0) {              // отвечает тогда, когда получает данные:

        data = Serial.read();                  // читать входящий байт:

        Serial.print("I received: ");          // скажите, что вы получили
        Serial.println(data);
    }
}
```

# #2 задача: напечатать текст

## □ Task(Задание)

1. Введите символ в последовательный монитор
2. Распечатайте соответствующий текст, как показано ниже.

Character	Text statement
'F'	"Go Forward"
'B'	"Go backward"
'R'	"Turn Right"
'L'	"Turn Left"

## □ Use Tip(Подсказка)

- Установите скорость передачи данных с помощью `Serial.begin ()`
- Получите количество байтов, доступных для чтения из последовательного порта с `Serial.available ()`
- Прочитайте последовательные данные `Serial.read ()` и распечатайте данные на последовательный порт с помощью `Serial.println ()`



# #2-а задача: найти максимальное значение

## □ Task(Задание)

1. Введите новое значение в последовательный монитор
2. Сравните сохраненное максимальное значение с входным значением
3. Вывести максимальное значение

## □ Use Tip(Подсказка)

- Установите скорость пере данных с помощью `Serial.begin ()`
- Получите количество байтов, доступных для чтения из последовательного порта с `Serial.available ()`
- Прочитайте последовательные данные `Serial.read ()` и распечатайте данные на последовательный порт с помощью `Serial.print ()` и `Serial.println ()`

# Содержание

□ Arduino

□ Основы управления

- последовательный ввод / вывод
- Основы управления аппаратным обеспечением

# Управление цифровым I/O

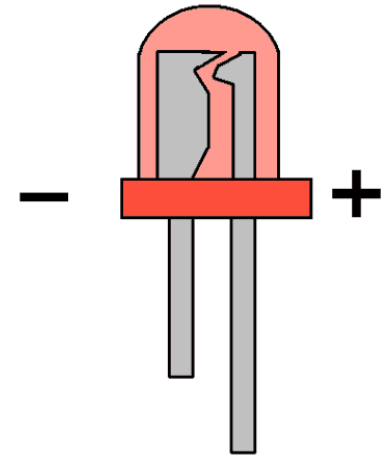
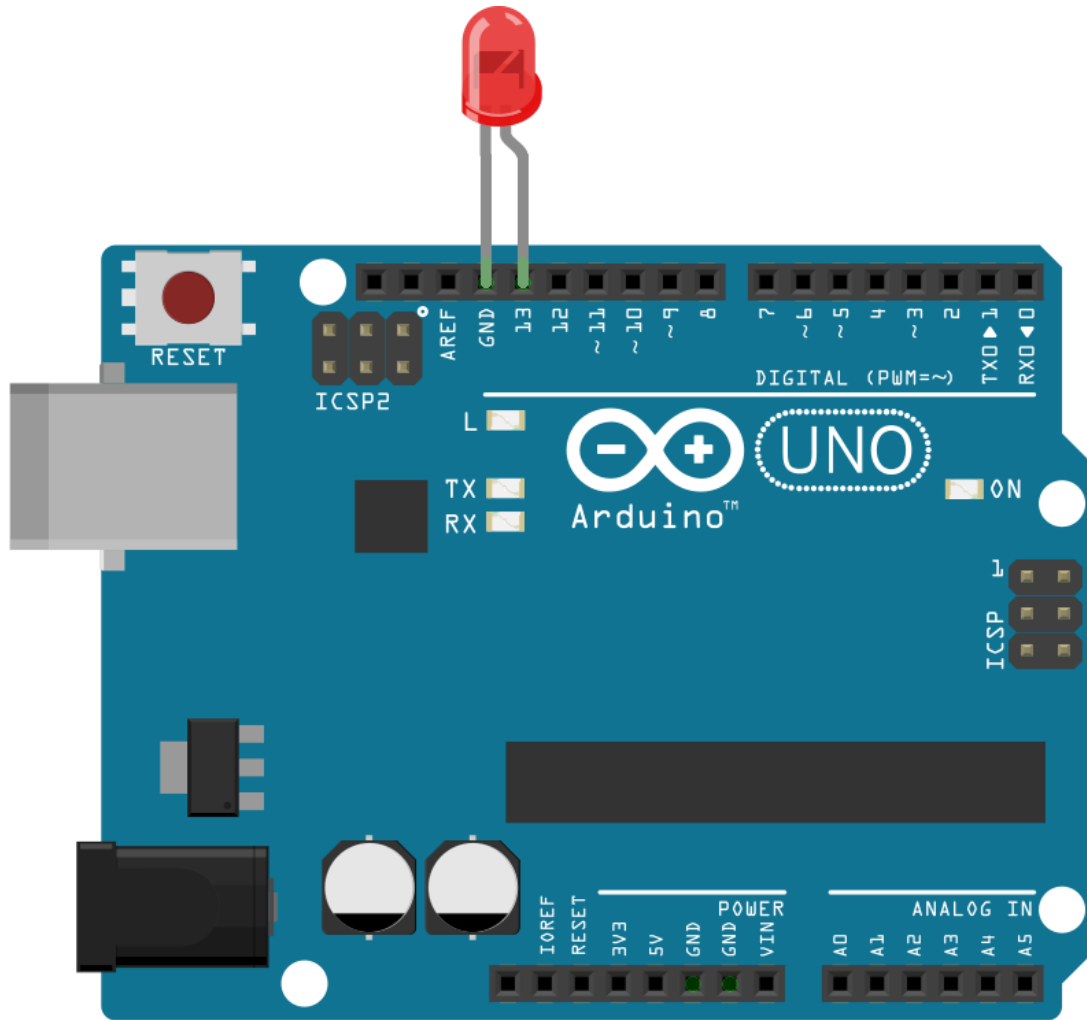
## ❑ Функция pinMode()

- заданный pin, может быть использован как вход или как выход
- Синтаксис
  - pinMode(pin, mode)
    - pin: номер контакта, режим которого вы хотите установить.
    - mode: это INPUT, OUTPUT, или INPUT\_PULLUP
- Пример

```
void setup()
{
  pinMode(2, INPUT_PULLUP); // устанавливает цифровой pin2 в качестве входа

  pinMode(13, OUTPUT);      // устанавливает цифровой pin13 в качестве выхода
}
```

# Arduino схема подключения: светодиода





# Управление цифровым I/O

## ❑ функция digitalWrite()

- Напишите значение HIGH или LOW для цифрового pin
- Синтаксис
  - digitalWrite(pin, value)
    - pin: номер pin
    - mode: HIGH or LOW (включить или выключить)
- Пример

```
void loop()
{
  digitalWrite(13, HIGH); // устанавливает pin 13 как включить
  delay(1000);           // ждет секунду
  digitalWrite(13, LOW);  // устанавливает pin 13 как выключить
  delay(1000);           // ждет секунду
}
```

# #3 задача: LED Blinking

## □ Task(Задание)

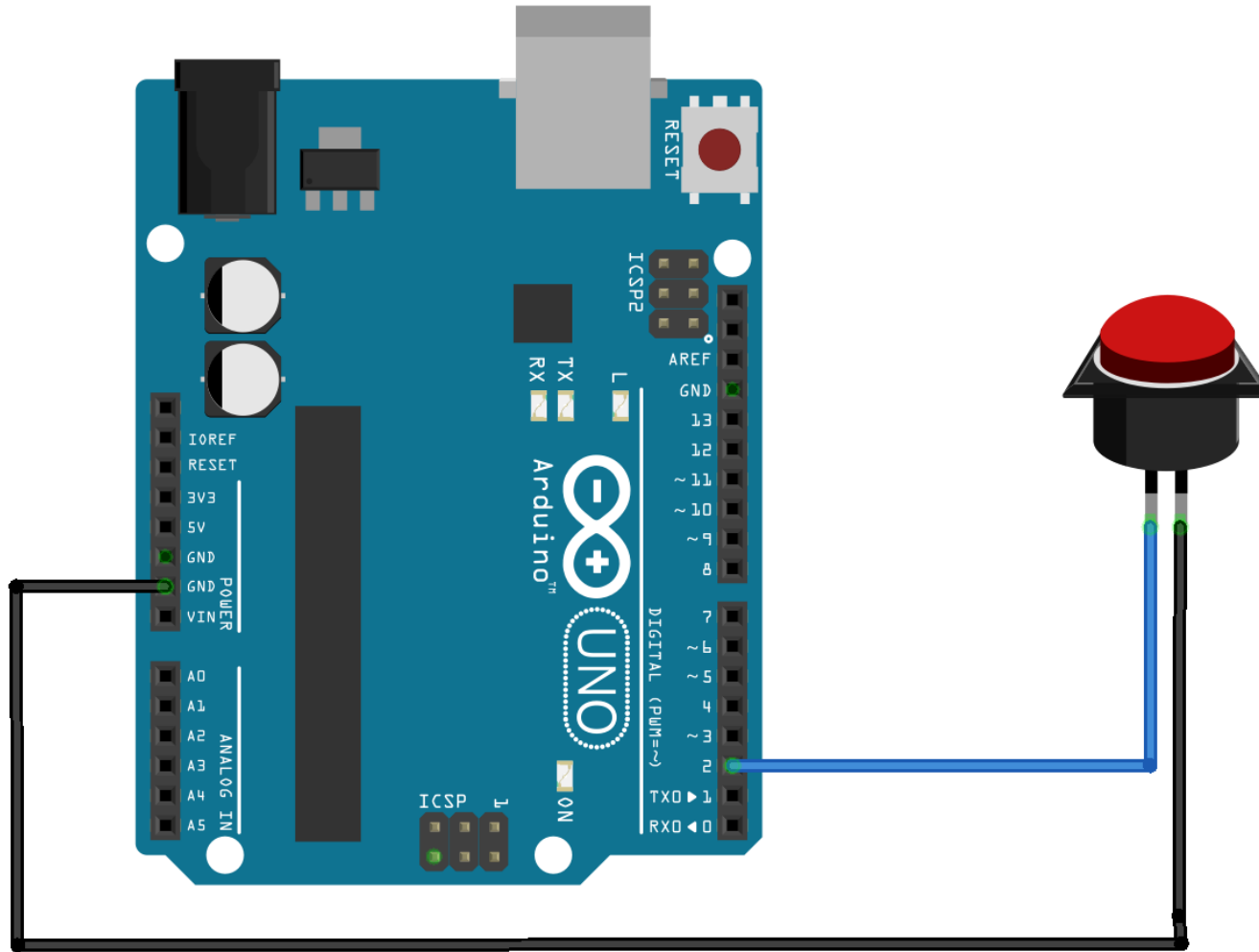
1. Введите значение, указывающее секунду (2: 2 секунды)
2. Сделать светодиод мигать с интервалом времени ввода

секунды	светодиод мигать
1	Светодиод мигает каждую 1 секунду
5	Светодиод мигает каждые 5 секунд

## □ Use Tip(Подсказка)

- Установите скорость передачи данных с помощью `Serial.begin()`
- Получить число байтов, доступных для чтения с последовательного порта с помощью `Serial.available ()` и Считать последовательные данные с помощью `Serial.read ()`
- Настройте выходной контакт с помощью `pinMode ()`
- Записать ВЫСОКОЕ или НИЗКОЕ значение на 13-контактный разъем с помощью `digitalWrite ()`

# схема подключения Arduino : Нажатие кнопки



# Управление цифровым I/O

## ❑ Функция digitalRead()

- Считывает значение с указанного цифрового вывода, либо HIGH, либо LOW
- Синтаксис
  - digitalRead(pin)
    - Нужно указать номер pin , с которого вы хотите прочитать
    - Возвращает значение: LOW или HIGH
- Пример

```
void loop()
{
  bool val;
  val = digitalRead(2);  // read the input pin(2)
}
```



# #4 задача: кнопка и светодиод

## □ Task(Задание)

1. Прочтите значение с 2х pin подключенного с помощью кнопки
2. Включите светодиод при нажатии кнопки
3. Выключите светодиод при отпускании кнопки

## □ Use Tip(подсказка)

- Настроить pin input/output используя функцию `pinMode()`
- Прочитать значения с 2х pin с помощью функции `digitalRead()`
- Записать значение HIGH или LOW в pin13 используя функцию `digitalWrite()`

# #4-а задача: кнопка и светодиод

## □ Task(Задание)

1. Прочтите значение с 2х pin подключенного с помощью кнопки
2. Включите светодиод при нажатии кнопки
3. Выключите светодиод при **повторном нажатии кнопки**

## □ Use Tip(подсказка)

- Настроить pin input/output используя функцию **pinMode()**
- Прочитать значения с 2х pin с помощью функции **digitalRead()**
- Записать значение HIGH или LOW в pin13 используя функцию **digitalWrite()**

# #приложение

## ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

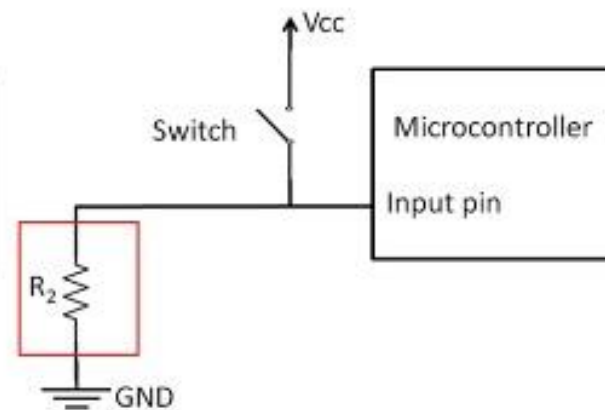
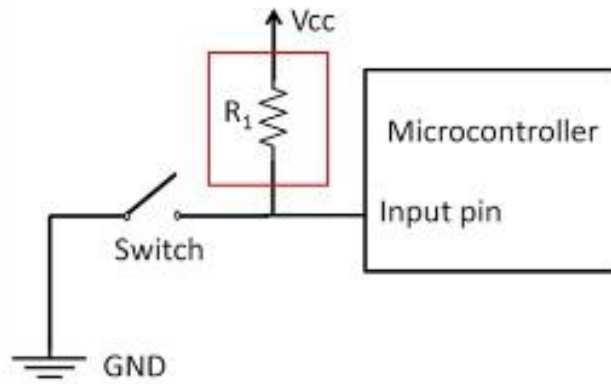


# #приложение

## Pull-up Resistor Circuit

## Pull-down Resistor Circuit

### Circuit Arrangement



When **Switch** is open

$R_1$  = Pull up resistor  
Current Path =  $V_{cc} \rightarrow$  input pin  
 $\therefore$  Voltage at input pin =  $V_{cc}$  (High)

$R_2$  = Pull down resistor  
Current Path = Input pin  $\rightarrow$   $GND$   
 $\therefore$  Voltage at input pin =  $GND$  (Low)

When **Switch** is closed

Current Path =  $V_{cc} \rightarrow$  input pin  $\rightarrow$   $GND$   
 $\therefore$  Voltage at input pin =  $GND$  (Low)

Current Path =  $V_{cc} \rightarrow$  input pin  $\rightarrow$   $GND$   
 $\therefore$  Voltage at input pin =  $V_{cc}$  (High)

