УРОКИ ARDUINO

Данное пособие является сборником команд, операторов и функций (ШПАРГАЛКА), стандартных для языка C++ и *Arduino Wiring*. Пособие создано с целью иметь в одном месте список всех «инструментов» для работы с Arduino, чтобы всегда можно было его открыть и сразу найти нужное. Информация в пособии дублирует, а иногда даже дополняет теоретическую часть в видеоуроках канала «**Заметки Ардуинщика**» https://goo.gl/rgM5Kj
По следующему адресу находится **пополняемый сборник полезных алгоритмов** для различных областей применения http://alexgyver.ru/arduino-algorithms/
По жизненно важным вопросам обращаться на почту alexgyvershow@ya.ru
Принимаю пожертвования :3 http://alexgyver.ru/support_alex/
Приятного чтения!

С уважением, автор канала и пособия Александр «AlexGyver» Майоров

Оглавление

Урок 1 – данные и переменные

Видео вариант: https://youtu.be/CRRlbzzt3VA

Структура программы

// Однострочный комментарий

/* Многострочный комментарий (комментарии не занимают память) ***/**

#include - подключить файл (библиотеку)

void setup() {} - всё находящееся внутри {} будет выполнено 1 раз при загрузке Ардуино

void loop() {} - всё находящееся внутри {} бесконечно повторяется сверху вниз

После каждого «действия» ставится точка запятой;

Типы данных

<тип данных> <имя>;

int my_val; // объявить переменную my_val)

<тип данных> <имя> = <значение>;

int my_val = 2300; // объявить переменную my_val и присвоить ей число 2300

Название	Вес	Диапазон	Особенность
boolean	1 байт	0 или 1	Логическая переменная, может принимать значения **true** (1) и **false** (0)
char	1 байт	-128… 127	Хранит номер символа из таблицы символов ASCII
byte	1 байт	0… 255	
int	2 байта	-32 768… 32 767	
unsigned int	2 байта	0… 65 535	
word	2 байта	0… 65 535	То же самое, что unsigned int
long	4 байта	-2 147 483 648… 2 147 483 647	- 2 миллиарда… 2 миллиарда
unsigned long	4 байта	0… 4 294 967 295	0… 4 миллиарда…
float	4 байта	-3.4028235E+38… 3.4028235E+38	Хранит числа с плавающей точкой (десятичные дроби). **Точность: 6-7 знаков**
double	4 байта		То же самое, что float

Особенности использования переменных

- Внимательно следите за значением, которое принимает переменная. Если значение превысит максимальное или принизит минимальное (выйдет из диапазона) для этого типа данных, то переменная сбросится в 0, либо выдаст вообще случайное число. Такую ошибку потом будет трудно отследить.

- Тип данных указывается при объявлении переменной ТОЛЬКО ОДИН РАЗ, далее переменная используется чисто по имени (обращение к переменной). При попытке сменить тип переменной (переобъявить переменную) вы получите ошибку. Но только в том случае, если переменная глобальная, либо когда локальная переобъявляется внутри функции, в которой она была объявлена.

Особенности float

- Присваивать только значение с точкой, даже если оно целое (10.0)
- Делить тоже только на числа с точкой, даже если они целые (переменная / 2.0)
- При делении целочисленного типа с целью получить число с плавающей точкой, писать (float) перед вычислением!
- Операции с числами типа float занимают гораздо больше времени, чем с целыми! Если нужна высокая скорость вычислений, лучше применять всякие хитрости =)

Урок 1.1 – операции с переменными и константами

Видео вариант: https://youtu.be/tm831gRkscY

Типы переменных

Глобальная переменная – объявляется ВНЕ функций, например в самом начале скетча. Обращаться к глобальной переменной (использовать её значение) можно использовать ВЕЗДЕ.
Локальная переменная – объявляется ВНУТРИ функции, и обращаться к ней можно только внутри этой функции.
Локальных переменных может быть несколько с одинаковым именем, но разными значениями. Это связано с тем, что локальная переменная выгружается из оперативной памяти микроконтроллера при выходе из функции. То есть две переменные с одинаковым именем и разным значением существуют не одновременно, а в разное время.

Константы

const <тип> <имя> = <значение>; - объявить константу
const int my_val = 2300; // объявить константу my_val и присвоить ей число 2300
#define <имя> <значение> - объявить константу через define
#define my_val 2300 // определить константу my_val и присвоить ей число 2300
При попытке сменить значение константы ПОСЛЕ её объявления, вы получите ошибку!

Математические операторы

+, -, *, / - сложить, вычесть, умножить...
pow(x, a); - возвести "x" в степень "a" (x^a), **pow** может возводить в дробную степень!
sq(x); - возвести число "x" в квадрат (x^2)
sqrt(x); - взять квадратный корень числа "x"
abs(x); - найти модуль числа, |x|
sin(x), cos(x), tan(x); - синус, косинус, тангенс
round(x); - математическое округление (если после запятой больше или равно 5, то округляем в большую сторону)
ceil(x); - округлить в бОльшую сторону
floor(x); - округлить в меньшую сторону

x += a; - прибавить "a" к "x"
x -= a; - вычесть "a" из "x"
x *= a; - домножить "x" на "a"
x /= a; - разделить "x" на "a"
x++; - увеличить "x" на 1
x--; - уменьшить "x" на 1

Урок 2 – последовательный порт

Видео вариант: https://youtu.be/gmgw6nLgzbY

Операторы библиотеки Serial

Serial - объект библиотеки Serial для работы с последовательным портом (COM портом)
Serial.begin(<скорость>); - открыть порт
Serial.begin(9600); // открыть порт на 9600 БОД
ВНИМАНИЕ! Скорость, установленная в begin(), должна быть равна скорости монитора порта (в самом мониторе правый нижний угол). Иначе в выводе получите крокозябры!
Serial.print(); - вывод в порт. Переменные и цифры напрямую, текст – в кавычках " "
Serial.println(); - вывод с переводом строки
Serial.println(val, n); - вывод *переменной val (типа float)* с n числом знаков после запятой
Serial.println(val, <базис>); - вывод с указанным базисом:

- **DEC** - десятичный (человеческие числа)
- **HEX** - 16-ричная система
- **OCT** - 8-ричная система
- **BIN** - двоичная система

Данные с компьютера попадают в буфер с объёмом 64 байта, и ждут обработки
Serial.available(); - проверить буфер на наличие входящих данных
Serial.read(); - прочитать входящие данные в символьном формате! Согласно ASCII
Serial.read() - '0'; - прочитать данные в целочисленном формате. По одной цифре!
Serial.parseInt(); - прочитать данные в целочисленном формате. Число целиком!
Serial.flush(); - очистить буфер порта

Код		Код		Код		Код		Код		Код		Код		Код	
32	пробел	44	,	56	8	68	D	80	P	92	\	104	h	116	t
33	!	45	-	57	9	69	E	81	Q	93	]	105	i	117	u
34	"	46	.	58	:	70	F	82	R	94	^	106	j	118	v
35	#	47	/	59	;	71	G	83	S	95	_	107	k	119	w
36	$	48	0	60	<	72	H	84	T	96	`	108	l	120	x
37	%	49	1	61	=	73	I	85	U	97	a	109	m	121	y
38	&	50	2	62	>	74	J	86	V	98	b	110	n	122	z
39	'	51	3	63	?	75	K	87	W	99	c	111	o	123	{
40	(	52	4	64	@	76	L	88	X	100	d	112	p	124	|
41	)	53	5	65	A	77	M	89	Y	101	e	113	q	125	}
42	*	54	6	66	B	78	N	90	Z	102	f	114	r	126	~
43	+	55	7	67	C	79	O	91	[	103	g	115	s	127	⌂

Урок 3

Видео вариант: https://youtu.be/hnKImcN3jYE

Условный оператор if

if () {} - условный оператор, проверяет условие в () и выполняет код в {} если оно верно
if () { - проверяет условие, если верно,
выполняет эту часть кода
} else { - если неверно
выполняет вот эту часть кода
}

if () { - проверяет условие, выполняет если верно
} else if () { - если неверно, проверяет новое
} else if () { - если неверно, проверяет новое
} else if () { - если неверно, проверяет новое
} else { - если неверно, выполняет то, что ниже
}

Операторы сравнения

a == b - если a равно b
a != b - если a не равно b
> - если a больше b (строго)
< - если a меньше b (строго)
>= - если a больше или равна b
<= - если a меньше или равна b

Логические операторы

&& - логическое И (одно условие И второе)
|| - логическое ИЛИ (либо одно, либо второе)
! – отрицание (например if (!val) - если val - ложь, т.е. 0)

Добавлено от WakeUp4L1fe
*Использовать **boolean (bool)** лучше со значениями **true** и **false**.*
*C++ приравнивает ноль к **false**, а любое число к **true**.*
К примеру:

```
bool x = 2;
if (x == 1) then {
Serial.println("истина");
} else {
Serial.println("ложь");
}
// В порт выведется слово истина, хотя присваивали двойку!
```

Добавлено от Alexei Belousov
Небольшое добавление по условиям. Существует и укороченная запись условий

```
(a > b) ? c == true : c == false;
```

Если A больше B то C равно истина, иначе C равно ложь..

Также имеет место запись присваивания переменной значения результата сравнения:
c == (a > b);

Оператор выбора switch.. case

switch (val) { - рассматриваем переменную val
case 1: - если она равна 1, выполнить код здесь
break;

case 2: - если она равна 2, выполнить код здесь
break;

case 3: - если она равна 3, выполнить код здесь
break;

default: - если что-то ещё, выполнить код здесь (default необязателен)
break;
}

Добавлено от WakeUp4L1fe
Еще стоило указать возможность использования одновременно нескольких условий ***switch*** *оператора:*

switch (val) {
case 1:
case 2:
Serial.println("1 или 2");
break;
case 3:
Serial.println("3");
break;
}
// то есть при пропуске ***break;*** *будут проверены и отработаны оба условия*

Урок 4

Видео вариант: https://youtu.be/Ik7SwQ477mA

Функции задержек

delay() - задержка, в скобках указывается число миллисекунд (в 1 сек 1'000 миллисекунд). Максимальное значение типа *unsigned long* (4 байта), 4’294’967’295 мс, или около 1200 часов, или 50 суток.

delayMicroseconds() - задержка, в скобках указывается число микросекунд (в 1 сек 1'000'000 микросекунд). Максимальное значение 16'383 мкс, или 16 миллисекунд.

ИСПОЛЬЗОВАТЬ ЗАДЕРЖКИ НЕ РЕКОМЕНДУЕТСЯ! ОНИ ПОЛНОСТЬЮ "ВЕШАЮТ" СИСТЕМУ!

Функции таймера

millis() - возвращает количество миллисекунд, прошедших с момента включения МК.

Макс. значение: 4’294’967’295 мс или 50 суток.

Разрешение: 1 миллисекунда.

micros() - возвращает количество микросекунд, прошедших с момента включения МК.

Макс. значение: 4’294’967’295 мкс или 70 мин

Разрешение: 4 микросекунды

Пишем (unsigned long) или (long) перед умножением, чтобы программатор выделил нужное количество памяти для проведения операции! (для работы с большими числами).

*Пример: (unsigned long)23*24*60*60*1000, если хотим получить ПРАВИЛЬНЫЙ результат умножения в виде числа миллисекунд, равного 23 дням.*

Решаем проблему переполнения таймера в конструкции

```
 if (millis() - last_time > 5000)  {
// что-то сделать
last_time = millis();
}
```

Добавлено от Steep TR

А что будет с этим условием, если пройдет 50 суток? Ведь last_time будет содержать число ближе к максимальному, а millis() ближе к начальному т.е. нужно еще одно условие сделать, чтобы по прошествию 50 суток все не умерло =)

```
 if (millis() < last_time) last_time = millis(); // выполнится, когда таймер переполнится
```

А ЕЩЁ МОЖНО СДЕЛАТЬ ТАК

Добавлено от **WakeUp4L1fe**

```
if (nextTime <= millis())  {
// что-то сделать
nextTime = millis() + 5000;
 }
```

Получаем слегка другую логику работы: «опережаем» время на 5 секунд, и выполняем действие, когда таймер догоняет.

Получился таймер, который не переполняется! Спасибо WakeUp4L1fe!

Урок 5

Видео вариант: https://youtu.be/3UwgMAdV4xQ

Режимы работы цифровых портов

• Аналоговые и цифровые порты могут работать как ВХОДЫ и как ВЫХОДЫ

• По умолчанию все порты работают КАК ВХОДЫ

pinMode(pin, mode); - настроить порт

• **pin** - номер порта. Цифровые: 0 – 13. Аналоговые: 14 - 19, либо A0 - A5

• **mode** - режим работы порта

  • INPUT - вход, принимает сигнал

  • OUTPUT - выход, выдаёт 0 или 5 Вольт

  • INPUT_PULLUP - вход с подтяжкой к 5 В

Генерация цифрового сигнала

digitalWrite(pin, signal); - подать цифровой сигнал

• **pin** - номер порта. Цифровые: 0 – 13. Аналоговые: 14 - 19, либо A0 - A5

• **signal** - какой сигнал подаём

  • **LOW**, или 0 (ноль), или false - 0 Вольт

  • **HIGH**, или 1, или true - 5 Вольт

Чтение цифрового сигнала

digitalRead(pin); - прочитать цифровой сигнал

• **pin** - номер порта. Цифровые: 0 – 13. Аналоговые: 14 - 19, либо A0 - A5

Урок 10

Видео вариант: https://youtu.be/ypH3W8r41Cw

Чтение аналогового сигнала

analogRead(pin); - прочитать аналоговый сигнал (оцифровать)

• **pin** - номер пина. Аналоговые: 0 - 7

• Функция возвращает значение 0.. 1023 в зависимости от напряжения на пине от 0 до опорного напряжения (грубо 5 В)

Изменение диапазона значений

map(val, min, max, new_min, new_max); - возвращает величину в новом диапазоне

• **val** - входная величина

• **min, max** - минимальное и максимальное значение на входе в map

• **new_min, new_max** – соответственно мин. и макс. значения на выходе

constrain(val, min, max); - ограничить диапазон переменной val до min и max

Урок 11

Видео вариант: https://youtu.be/rCmaMST8qkg

Генерация ШИМ сигнала

analogWrite(pin, duty);

• **pin** – пин, На котором генерировать ШИМ

• **duty** – величина 0.. 255, соответствует скважности ШИМ 0.. 100%

ШИМ пины Arduino NANO, UNO: 3, 5, 6, 9, 10, 11

ШИМ пины Arduino MEGA: все до 13

Урок 12

Видео вариант: https://youtu.be/gIVcKbSeqFo

Цикл for, «счётчик»

for (counter; condition; change) {} - цикл for

• **counter** – переменная счётчика, обычно создают новую «локальную», в стиле **int i = 0**;

• **condition** – условие, при котором выполняется цикл, например *«счётчик меньше 5»* **i < 5;**

• **change** – изменение, т.е. увеличение или уменьшение счётчика, например **i++**, **i--**, **i += 10**;

Пример:

```
for (byte i = 0; i < 100; i++) {   // счётчик от 0 до 99
Serial.println(i);                 // вывести в монитор порта числа от 0 до 99
}
```

Цикл while, «с предусловием»

while (condition) {}

• **condition** – условие, при котором выполняется блок кода, заключённый в **{}**

Пример:

```
while (flag) {
// какой-то кусок кода, который выполняется, пока flag равен логической 1
}
```

Цикл do while, «с постусловием»

do {} while (condition) ;

• **condition** – условие, при котором выполняется блок кода, заключённый в **{}**

Пример:

```
do {
// какой-то кусок кода, который выполняется, пока flag равен логической 1
// но в отличие от предыдущего цикла, выполнится ХОТЯ БЫ ОДИН РАЗ, даже если Flag равен 0
} while (flag);
```

break – выход из цикла

Пример:

```
for (byte i = 0; i < 100; i++) {   // счётчик от 0 до 99
if (i > 50) break;                 // выйти из цикла, если i больше 50
Serial.println(i);                 // вывести в монитор порта числа от 0 до 50!!!
}
```

continue – пропустить ход

Пример:

```
for (byte i = 0; i < 100; i++) {        // счётчик от 0 до 99
if (i > 50 && i < 60) continue;         // если i от 50 до 60, перейти в начало цикла
Serial.println(i);                      // в монитор порта пойдут числа от 0 до 50, затем от 60 до 99
}
```

Урок 13

Видео вариант:

Функция, которая ничего не берёт и не возвращает

```
void myFunction() {}
```

• **void** – слово, показывающее, что функция ничего не возвращает

• **myFunction** – название функции

Пример:

```
void myFunction() {             // задаём функцию
Serial.println("HELLO!");       // которая выведет в порт HELLO!
}
```

В другом месте программы вызываем функцию как **myFunction();** и в этом месте будет выслано слово HELLO! в порт.

Функция, которая берёт и ничего не возвращает

Пример:

```
void myFunction(int val) {   // создаём функцию, на вход которой подаётся одно число

Serial.println(val);          // и выводим его в порт

}
```

В другом месте программы вызываем функцию как **myFunction(50);** и в этом месте будет выслано в порт число 50.

Функция, которая берёт и возвращает

Пример:

```
int mySum(int val1, int val2) {     // создаём функцию, на вход которой подаётся два числа

return val1 + val2;                 // возвращаем сумму чисел

}
```

• **return** – оператор, возвращающий результат

В другом месте программы вызываем функцию как **mySum(50, 70);** и функция вернёт результат 120. Как пример:

```
Serial.println(mySum(50, 70));     // в порт будет послана сумма чисел, т.е. 120
```

Урок 14

Видео вариант:

Генерируем случайные числа

randomSeed(value); - функция, задающая начало отсчёта генератору псевдослучайных чисел

• **value** – любое число типа *long* (смотри урок №1)

random(min, max); – функция, возвращающая случайное число в диапазоне от **min** до **max - 1**

random(max); - то же самое, но возвращает от 0 до **max - 1**

Пример:

```
Serial.println( random(20) );     // вывести в порт случайное число от 0 до 19
```

Как получать максимально случайную последовательность чисел?

```
randomSeed(analogRead(0));   // в качестве опорного числа взять сигнал с НЕПОДКЛЮЧЕННОГО НИКУДА аналогового пина
```

Урок 15

Видео вариант:

Создание массива

<тип данных> <имя массива>[<число элементов>];

<тип данных> <имя массива>[<число элементов>] = {элемент1, элемент2…};

Если не указываются элементы, то обязательно нужно указать размер массива, чтобы под него выделилось место в памяти. Размер можно не указывать в том случае, если сразу указываются все элементы. Примеры:

```
int myInts[6];

int myPins[] = {2, 4, 8, 3, 6};

int mySensVals[6] = {2, 4, -8, 3, 2};

char message[6] = "hello";
```

Чтение – запись

Главное помнить, что нумерация элементов НАЧИНАЕТСЯ С НУЛЯ!

```
myArray[5] = 10;          // присвоить пятому элементу число 10

if (myArray[5] == 20) …..  // если  элемент массива под номером 5 равен 20…
```

Пример. Забивка массива случайными числами

```
byte myArray[50];              // создать массив myArray на 50 ячеек

for (byte i = 0; i < 50; i++) {      // счётчик от 0 до 49

myArray[i] = random(100);      // присвоить случайное число от 0 до 99 элементам массива под номерами 0.. 49

}
```

Урок 16

Видео вариант:

Аппаратные прерывания

attachInterrupt(pin, function, state); - подключить прерывание

detachInterrupt(pin); - отключить прерывание

• **pin** – пин прерывания, для NANO и UNO это пины D2 и D3, соответствуют номерам 0 и 1

• **function** – название функции, которая будет вызвана при срабатывании прерывания

• **state** – режим обработки, их несколько:

- **LOW** - вызывает прерывание, когда на порту LOW
- **CHANGE** - прерывание вызывается при смене значения на порту, с LOW на HIGH и наоборот
- **RISING** - прерывание вызывается только при смене значения на порту с LOW на HIGH
- **FALLING** - прерывание вызывается только при смене значения на порту с HIGH на LOW

Пример:
Кнопка подключена к D2 и GND

```
void setup() {
pinMode(2, INPUT_PULLUP);                    // пин D2 подтянут к питанию
attachInterrupt(0, myInterrupt, FALLING);    // подключить прерывание на пин D2, обрабатывать при падении сигнала и вызывать функцию myInterrupt
}

void myInterrupt() {                         // функция обработчика прерываний
Serial.println("INTERRUPT!");                // при срабатывании вывести в порт слово INTERRUPT
}
```