КОМЕНТАРИИ

// - Однострочный комментарий

/*
* Многострочный коментарий
*/

*ВСЕ КОМЕНТАРИИ НЕ ЗАНИМАЮТ ПАМЯТЬ

БИБЛИОТЕКИ

#include - подключить файл(библиотеку)

ФУНКЦИИ

void - функция которая не возращает значения

void setup() {

- Всё находящееся внутри {} будет выполнено 1 раз при загрузке Arduino.

- Прописываем настройки и режим работы ардуино.
- Инициализируем всё, что к ней подключено.

}

void loop() {

- Всё находязееся внутри {} бесконечно повторяется сверху вниз.

- Идёт чтение значений приборов, их обработка.
- Выводы на дисплей, вращение моторчиков и т.д.
- Произведение расчётов.

}

Функция - фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы

Функция объявляется вне другой функции

Два типа функций:

Не возвращает результат

```
void <имяФункции>(){
}
```

Возвращает результат

```
<типДанных> <имяФункции> () {
}
```

void myFunction() { — не возвращающая функция

- **void - слово, показывающее, что функция ничего не возвращает**
- **myFunction - название функции**

- **return - оператор, возвращающий результат**

- **myFunction(); - обращение к функции в коде**

}

Временные функции

delay(); — задержка, в скобках указывается число миллисекунд (в 1 сек 1.000 миллисекунд).
Максимальное значение 4.294.967.295 мс – ~1200 часов – ~ 50 суток

delayMicroseconds() — задержка, в скобках указывается число микросекунд (в 1 сек 1.000.000 микросекунд)
Максимальное значение 16.383 мкс – 16 миллисекунд

! ИСПОЛЬЗОВАТЬ НЕ РЕКОМЕНДУЕТСЯ, ПОТОМУ ЧТО ОСТАНАВЛИВАЕТСЯ ВЕСЬ КОД !

millis() - возвращает количество миллисекунд, прошедших с момента включения микроконтроллера
Макс.значение: 4.294.967.295 мс – ~50 суток
Разрешение: 1 миллисекунда

micros() - возвращает количество микросекунд, прошедших с момента включения микроконтроллера
Макс.значение: 4.294.967.295 мкс – ~70 мин
Разрешение: 4 микросекунды

пример таймера

```
unsigned long last_time;

void setup() {

}

void loop() {
    if (millis() - last_time > 5000) {
        last_time = millis();
        <код> // выполняется параллельно с другим кодом и не тормозит его!
    }
}
```

ТИПЫ ДАННЫХ

объявление:

- **изначально без значения - "тип данных" "имя";**
- **изначально с значением - "тип данных" "имя" = "значение";**

примеры:

- **int hello1; // по умолчанию 0.**
- **int hello2 = 5; // равна 5.**

Название	Вес	Диапазон	Особенность
boolean	1 байт	0 или 1	Логическая переменная, может принимать значения true (1) и false (0) **Знак инверсии - !**
char	1 байт	-128 ... 127	Хранит номер символа из таблицы символов ASCII
byte	1 байт	0 ... 255	Используется для хранения небольших значений чисел

Название	Вес	Диапазон	Особенность
int	2 байта	-32.768 ... 32.767	Используется для хранения чисел
unsigned int	2 байта	0 ... 65.535	Используется для хранения только чётных чисел
word	2 байта	0 ... 65.535	тоже самое, что и unsigned int

Название	Вес	Диапазон	Особенность
long	4 байта	-2.147.483.648 ... 2 147 483 647	Используется для хранения больших чисел -2 млрд ... 2 млрд
unsigned long	4 байта	0 ... 4.294.967.295	Используется для хранения только чётных чисел

Название	Вес	Диапазон	Особенность
float	4 байта	-3.4028235E+38 ... 3.4028235E+35	Хранит числа с плавающей точкой (десятичные дроби) **Точность: 6-7 знаков**
double	4 байта		То же самое, что и float (не используется)

Особенности float

1. Присваивать только значения с точкой, даже если оно целое (10.0).

2. Делить тоже только на числа с точкой, даже если они целые (переменная / 2.0).

3. При делении целочисленного типа с целью получить число с плавающей точкой, писать (float) перед вычислением!

   Пример:

   float <name> = (float) <int переменная> / 2.314

P.S. Операции с числами типа float занимают гораздо больше времени, чем с целыми! Если нужна высокая скорость вычислений, лучше применять всякие хитрости.

ПЕРЕМЕННЫЕ

Действия с переменными

- +,-,*,/ - сложить, вычесть, умножить, поделить

- pow(x, a); - Возвести "x" В степень "a" (x^a) pow может возводить в дробную степень!
- sqr(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
- ceil(x); - округлить в бóльшую 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

Константы

const <тип> <имя> = <значение>; — объявить константу

#define <имя> <значение> "без ;" — объявить константу через define (занимает меньше памяти, и присвоенные значения через define являются именем переменных, т.е #define light1 7 — в коде обращаемся к 7.

! Константа значений не меняет !

Область видимости переменных

Все переменные объявленные в коде вне функций — глобальные

Переменные объявленные в коде в функции — локальные

ПОРТЫ

COM порт

- **Serial - объект библиотеки Serial для работы с последовательным портом (COM портом)**
- **Serial.begin(<скорость>); — открыть порт**
- **Serial.print(); — вывод в порт. Переменные и цифры напрямую, текст в ""**
- **Serial.println(); — вывод с переводом строки**
- **Serial.println(val, n); — вывод переменной val 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!**

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

Порты ввода/вывода

- **Аналоговые и цифровые порты могут работать как ВХОДЫ и как ВЫХОДЫ**
- **По умолчанию все работают как ВЫХОДЫ**

pinMode(pin, mode); — настройка порта

pin — номер порта:
Цифровые: 0-13
Аналоговые: 14-19 (A0-A5)
mode — режим работы порта:
INPUT - вход, принимает сигнал
OUTPUT - выходи, выдает 0 или 5 Вольт
INPUT_PULLUP - вход с подтяжкой к 5 Вольт

digitalWrite(pin, signal); — подать цифр. сигнал

pin — пин, куда подаем сигнал (см.выше)
signal — какой сигнал подаем:
LOW - 0 (false) — 0 Вольт
HIGHT - 1 (true) — 5 Вольт

digitalRead(pin); — прочитать цифровой сигнал

pin — номер пина, с которого считываем

Аналоговые порты ввода/вывода

analogRead(pin); — возвращает значение 0 ... 1023 в зависимости от напряжения на пине от 0 до опорного напряжения (гробо 5В)

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 min и max значения на выходе

constrain(val, min, max); — ограничить диапазон переменной val до min и max (map и constrain обычно используются вместе, дополняя друг друга)

ОПЕРАТОРЫ

Операторы сравнения

- == — равно
- != — неравно
- > — больше
- < — меньше
- >= — больше или равно
- <= — меньше или равно

Логические операторы

- && — логическое И
- || — логическое ИЛИ
- ! — логическое отрицание

Условные операторы

if () {

- **Условный оператор, проверяет условие в () и выполняет код в {} если оно верно**

}

if () {

- **Условный оператор, проверяет условие в () и выполняет код в {} если оно верно**

} else {

- **Выполнить кусок кода, если не сработал if**

}

if () {

- **Условный оператор, проверяет условие в () и выполняет код в {} если оно верно**

} else if () {

- **Выполнить кусок кода, если не сработало условие выше**

} else if() {

- **Выполнить кусок кода, если не сработало условие выше**

} else {

- **Выполнить кусок кода, если не сработали условия выше**

}

```
switch (val) { — рассматриваем переменную val
    case 1: <код> — если она равна 1, выполнить код здесь
    break;

    case 2: <код> — если она равна 2, выполнить код здесь
    break;

    ..........

    default: (необязательно) — если, что-то ещё, выполнить код
    здесь
}
```

ШИМ

Стандартный Arduino ШИМ генератор

Разрядность: 8 бит

Вход: цифровое значение 0 … 255

Выход: ШИМ сигнал со скважностью 0 … 100%

Функция: analogWrite(pin, duty);

- pin - пин, на который пойдёт ШИМ
- duty - значение 0 … 255

ЦИКЛЫ

for (counter; condition; change) {

- **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;**

}

break; - выйти из цикла
continue; - пропустить итерацию

while (condition) {

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

}

break; - выйти из цикла
continue; - пропустить итерацию

***while (1) {**

- **Бесконечный цикл**

}

do {} while (condition); - цикл с постусловием

- **condition** - условие, при котором выполняется блок кода, заключённый в {}. В отличие от предыдущего цикла, выполнится **хотя бы один раз**, даже если условие изначально неверно

break; - выйти из цикла
continue; - пропустить итерацию

РАНДОМ

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)

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

- **value** - любое число типа **long**

МАССИВЫ

Массив - структура данных в виде набора Компонентов (элементов массива), расположенных в памяти непосредственно друг за другом, и имеющих адреса, по которым их можно читать или перезаписывать.

Объявление массива

```
<тип> <имя> [<число элементов>];
<тип> <имя> [] = {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 сразу 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";
```

Нумерация элементов начинается с 0!

Обращение к массиву

<имяМассива>[<номерЯчейки>] = <значение>;>] = <значение>;

<типДанных> <имяПеременной> = <имяМассива>[<номерЯчейки>];

Двумерные массивы

```
int my2array[2][3] = { - 2 строки, 3 столбца
   {10, 300, 1000},
   {666, 20, 1300}
};
```

my2array[0][0]; - это 10

my2array[0][2]; - это 1000

my2array[1][1]; - это 20

my2array[1][0]; - это 300

	столбец 0	столбец 1	столбец 2	столбец 3
строка 0	arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]
строка 1	arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]
строка 2	arr[2][0]	arr[2][1]	arr[2][2]	arr[2][3]

АППАРАТНЫЕ ПРЕРЫВАНИЯ

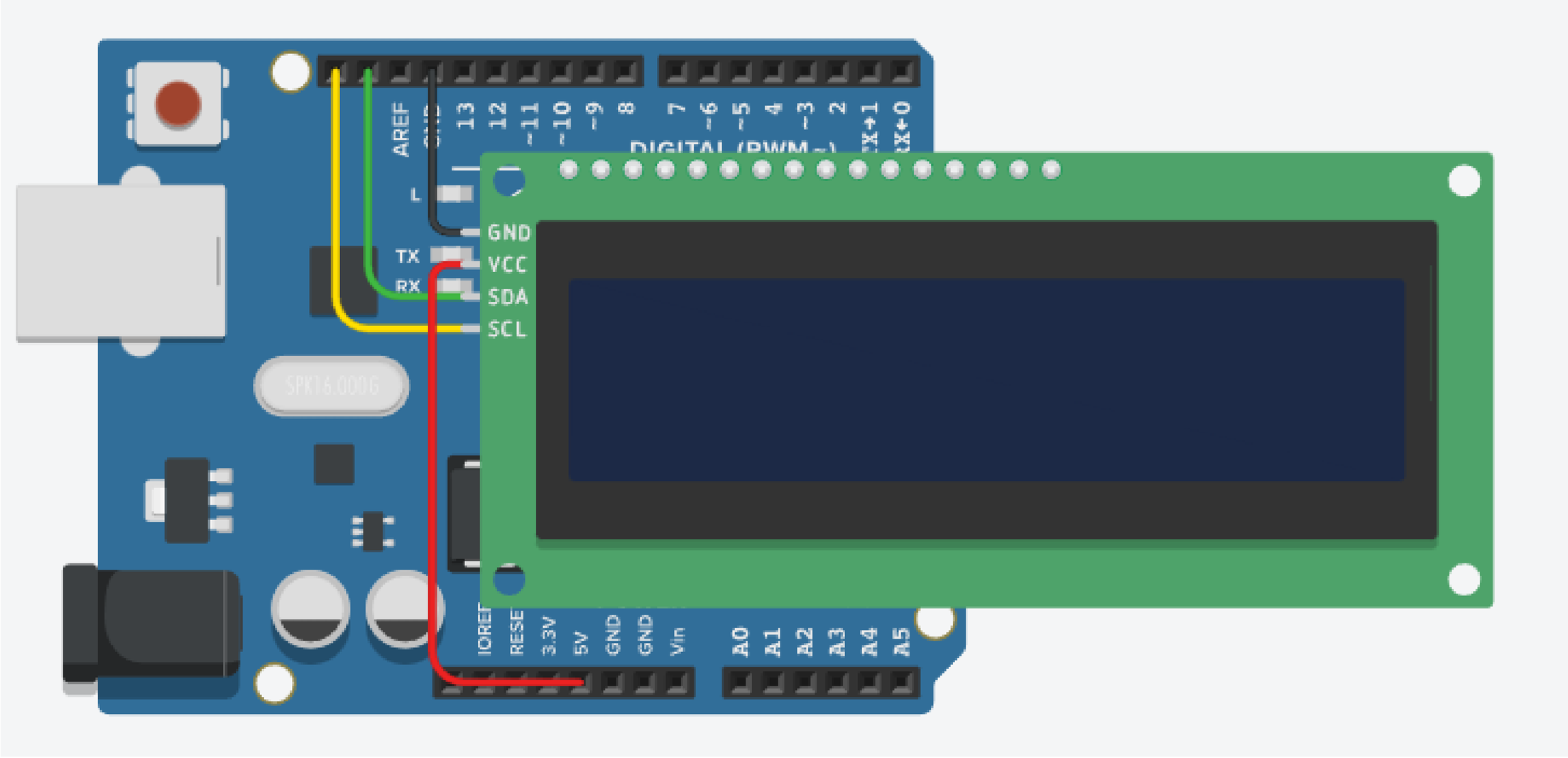
attachInterrupt(pin, function, mode); подключить прерывание
detachInterrupt(pin); - отключить прерывание

- **pin** - пин, на который настроена обработка
- **function** - вызываемая функция
- **mode** - режим работы. Их несколько:
- LOW - срабатывает, когда на пине LOW
- RISING - срабатывает, когда сигнал меняется с LOW на HIGH
- FALLING - срабатывает, когда сигнал меняется с HIGH на LOW
- CHANGE - срабатывает, когда сигнал меняется с LOW на HIGH и наоборот

- **noInterrupts();** приостановить обработку прерывания
- **interrupts();** - продолжить обработку прерывания

УПРАВЛЕНИЕ LCD

Схема подключения:



- #include <Adafruit_LiquidCrystal.h> — подключение библиотеки
- Adafruit_LiquidCrystal <имяДисплея>(0); — подключение LCD
- <имяДисплея>.begin(16, 2); — запуск дисплея
- <имяДисплея>.print(<текст>); — вывод текста на экран дисплея
- <имяДисплея>.setCursor(<символ>, <строка>); — откуда начать писать
- <имяДисплея>.setBacklight(<значение>); — яркость экрана:
  - 0 - темно
  - 1 - светло
- <имяДисплея>.blink(); — мигающий квадрат, типо курсор
- <имяДисплея>.clear(); — очистить написанное

библиотека: https://github.com/adafruit/Adafruit_LiquidCrystal