Arduino Quick Reference Card

version 2, 1 Aug 2018 by Ernest Neijenhuis PA3HCM

Structure

```
// declarations and includes
void setup() {
  // this will run once at program
  // startup.
}
void loop() {
  // this will be repeated until power
  // is removed.
}
```

Syntax

```
// This is a single line comment.

/*
  This is a
  multiline
  comment.
*/
```

`{}` - Code is grouped by enclosing it in curly brackets.
`;` - Each line of code ends with a semicolon.

```
#define ledPin 13
#include myLibrary;
```

Variables

`byte`	A single byte (8 bits) value, 0 to 255.
`int`	Integer, stores a number in 2 bytes (16 bits). Has no decimal places and will store a value between -32,768 and 32,767.
`long`	Used when an integer is not large enough. Takes 4 bytes (32 bits) of RAM and has a range from -2,147,483,648 to 2,147,483,647.
`boolean`	A simple `true` or `false` variable. Useful because it only uses one bit of RAM.
`float`	Used for floating point math (decimals). Takes 4 bytes (32 bits) of RAM and has a range from -3.4028235E+38 to 3.4028235E+38.
`char`	Character, stores one character using the ASCII code (ie 'A' = 65). Uses one byte (8 bits) of RAM. Arduino handles strings as an array of char’s.

```
int Number = 4;
long Counter = 1000000000000000000000000;
boolean gotcha = true;
float pi = 3.1415927;
char userinput = ‘B’;
char hw[13] = “Hello, world”;
```

Arithmatic operators

=	(assignment) assigns a value.
%	(modulo) gives the remainder when one number is divided by another
+	(addition)
-	(subtraction)
*	(multiplication)
/	(division)

```
int product = 4 * 2;            // 8
int radius = 12 % 5;            // 2
int area = 2 * 3.14 * radius;   // 12.76
```

Comparison operators

==	(equal to)
!=	(not equal to)
<	(less than)
>	(greater than)
=>	(greater than or equal to)
<=	(less than or equal to)

Control structures

```
if( condition ){ }
else if( condition ){ }
else { }
```

This will execute the code between the curly brackets if the *condition* is true, and if not it will test the `else if` condition if that is also false the `else` code will execute.

```
if(i>5){
  digitalWrite(ledPin, HIGH);
} else {
  digitalWrite(ledPin, LOW);
}
```

```
for(int i = 0; i < #repeats; i++){ }
```

Used to repeat a chunk of code a number of times (can count up `i++` or down `i--` or use any variable).

```
for(int count=0; count<10; count++){
  digitalWrite(ledPin, true);
  delay(1000);
  digitalWrite(ledPin, false);
  delay(1000);
}
```

`delay``(time);`
Causes a delay of `time` milliseconds.

Digital

`pinMode``(pin, mode);`
Used to set a pin's mode, *pin* is the pin number you would like to address 0-19 (analog 0-5 are 14-19). The *mode* can either be `INPUT` or `OUTPUT`.

`digitalWrite``(pin, value);`
Once a pin is set as an `OUTPUT`, it can be set either `HIGH` (pulled to +5 volts) or `LOW` (pulled to ground).

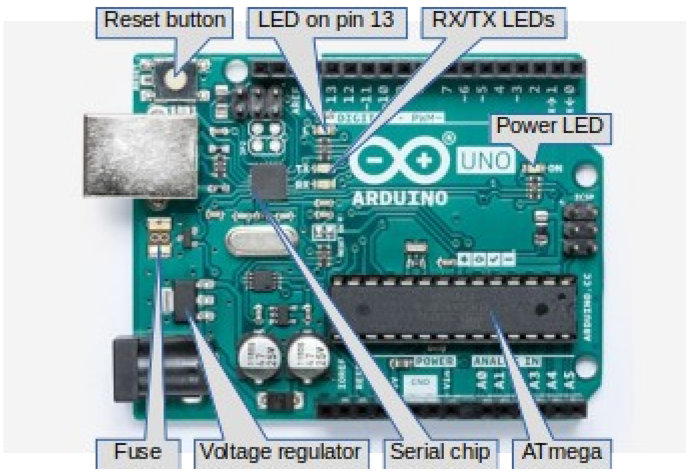
`int` **`digitalRead`**`(pin);`
Once a pin is set as an `INPUT` you can use this to return whether it is `HIGH` (pulled to +5 volts) or `LOW` (pulled to ground).

Analog

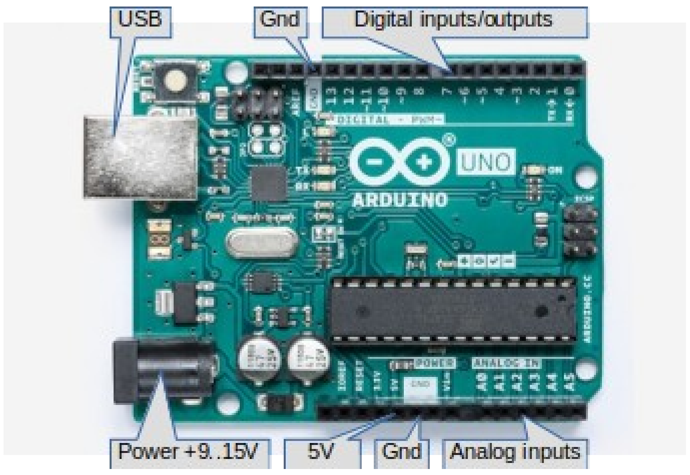
`analogWrite``(pin, value);`
Some of the Arduino's pins support pulse width modulation, which is basically a block wave signal. This function controls its duty cycle: 0 = 0% duty cycle, 255 = 100% duty cycle. You can use this to control a servo, or to control the brightness of an LED.

`int` **`analogRead`**`(pin);`
Returns the input value of an analog pin. A value between 0 (for 0 volts) and 1024 (for 5 volts) will be returned.

Components



Connectors



ATmega328 specifications (Uno, Duemilanove)

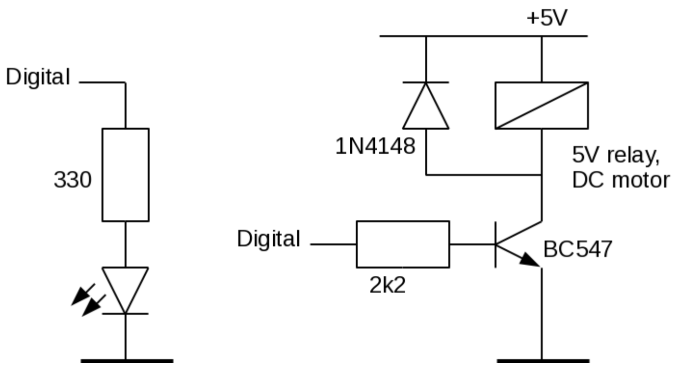
Processor:	8-bit AVR
Clock:	16 MHz
Flash memory:	32 kB
SRAM:	2 kB
EEPROM:	1kB
Digital I/O pins:	14 (of which 6 PWM capable)
Analog inputs:	6
Interrupts:	2
Protocols:	Serial, I2C/TWI, SPI, PWM
Size:	68.6 x 53.4 mm
Weight:	25 g

Power ratings

Power input : 6...20Vdc (recommended 8...12Vdc)
Maximum current per I/O pin: 20mA

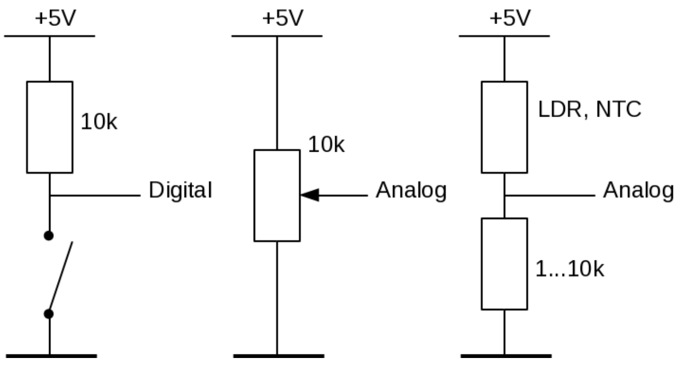
Basic output circuits

Left: LED
Right: 5V relay or small DC motor



Basic input circuits

Left: button or switch
Center: potmeter
Right: Light or temperature dependent resistor



Servo leads

Black or brown = GND
Red = +5V
White or orange = PWM