# Arduino Quick Reference Card

*by Ernest Neijenhuis PA3HCM*
*version 2.2, 26 Apr 2021*

## Basic structure

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

## Comments

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

/*
  This is a
  multi-line
  comment.
*/
```

## Punctuation

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

## Constants

```
const int motorPin = 9;
const float pi = 3.14;
```

The Arduino IDE has some built-in constants, including:
`INPUT INPUT_PULLUP OUTPUT LED_BUILTIN`
`HIGH LOW true false`

## Include library

```
#include <myLibrary.h>
```

## Variables

**byte** A small integer, 0 to 255 (8 bits)
**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 (8 bits)
**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 = 10000000000000000000000000;
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 a 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(LED_BUILTIN, HIGH);
} else {
  digitalWrite(LED_BUILTIN, 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), e.g.:

```
for(int i=0; i<10; i++){
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  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 to 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 determine whether its input is HIGH (pulled to +5 volts) or LOW (pulled to ground).

```
int buttonPressed = digitalRead(4);

if (digitalRead(4)) {
  digitalWrite(LED_BUILTIN, HIGH);
}
```

## 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, for example:

```
for (int i=255; i>0; i--) {
  analogWrite(6, i); // LED + resistor on pin 6
  delay(5);
}
```

```
int analogRead(pin);
```

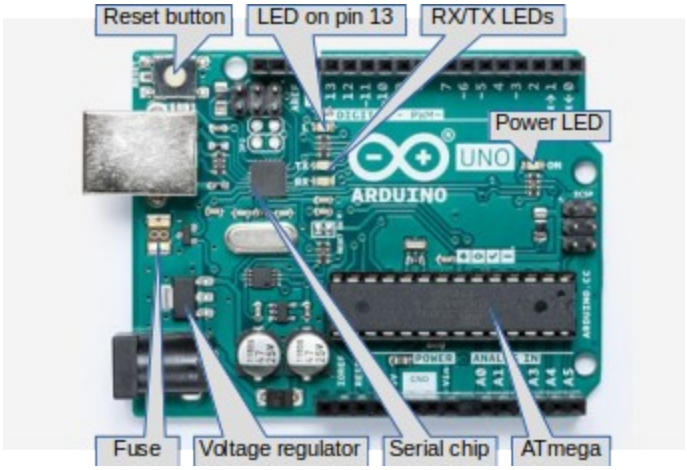
Returns the input value of an analog pin. A value between 0 (for 0 volts) and 1023 (for 5 volts) will be returned.

The full Arduino code reference can be found here:
**https://www.arduino.cc/reference**

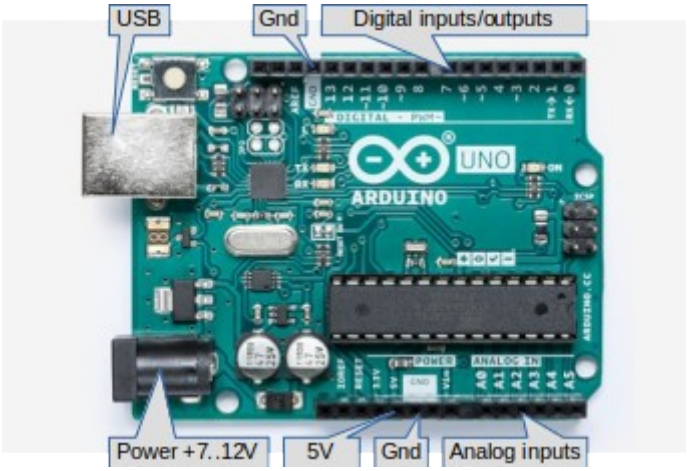
# Arduino Quick Reference Card

*by Ernest Neijenhuis PA3HCM*
*version 2.2, 26 Apr 2021*

## Components (Uno)




## Connectors (Uno)




## Specifications (Uno)

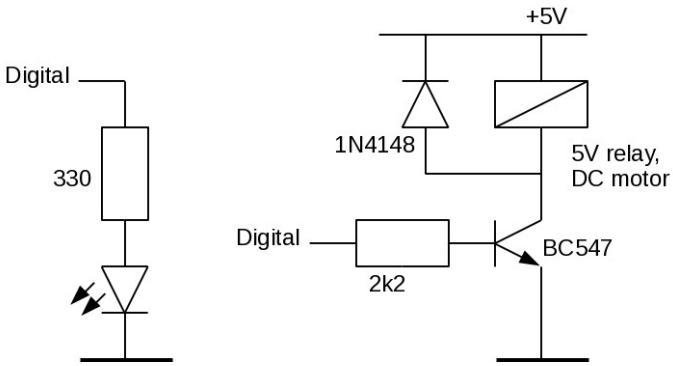
| | |
|---|---|
| Processor: | ATmega328 8-bit AVR |
| Clock: | 16 MHz |
| Flash memory: | 32 kB |
| SRAM: | 2 kB |
| EEPROM: | 1kB |
| Digital I/O pins: | 14 (of which 6 PWM capable) |
| Analog inputs: | 6 |
| Interrupts: | 2 |
| Protocols: | Serial, I2C/TWI, SPI, PWM |
| Size: | 68.6 x 53.4 mm |
| Weight: | 25 g |

## Power ratings

Maximum DC input: 6-20V (7-12V recommended)
Maximum DC current per I/O pin: 20mA
Maximum DC current on VCC, GND: 200mA

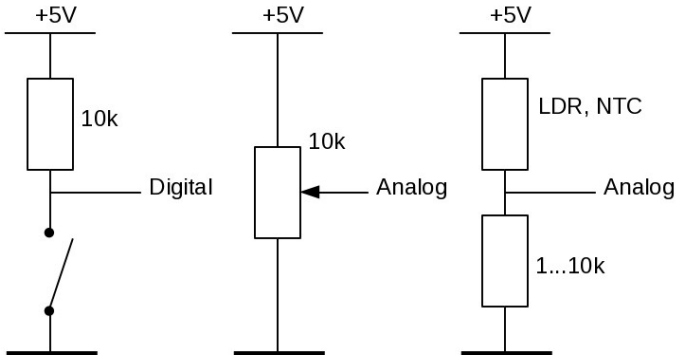
## Basic output circuits

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

Check my website for the latest version of this document, as well as lots of inspiring projects:
**http://www.pa3hcm.nl/**

This document is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.