# Arduino Types

| NO | Board name | PROCESSOR | Voltage | Crystal |
|---|---|---|---|---|
| 1 | Arduino Uno | ATmega328 | 5 V/7-12 V | 16MHz |
| 2 | Due | AT91SAM3X8E | 3.3 V/7-12 V | 84 MHz |
| 3 | Leonardo | ATmega32u4 | 5 V/7-12 V | 16MHz |
| 4 | Mega 2560 | ATmega2560 | 5 V/7-12 V | 16MHz |
| 5 | Mega ADK | ATmega2560 | 5 V/7-12 V | 16MHz |
| 6 | Micro | ATmega32u4 | 5 V/7-12 V | 16MHz |
| 7 | Mini | ATmega328 | 5 V/7-9 V | 16MHz |
| 8 | Nano | ATmega168 ATmega328 | 5 V/7-9 V | 16MHz |
| 9 | Ethernet | ATmega328 | 5 V/7-12 V | 16MHz |
| 10 | Esplora | ATmega32u4 | 5 V/7-12 V | 16MHz |
| 11 | ArduinoBT | ATmega328 | 5 V/2.5-12 V | 16MHz |
| 12 | Fio | ATmega328P | 3.3 V/3.7-7 V | 8MHz |
| 13 | Pro (168) | ATmega168 | 3.3 V/3.35-12 V | 8MHz |
| 14 | Pro (328) | ATmega328 | 5 V/7-12 V | 16MHz |
| 15 | Pro Mini | ATmega168 | 3.3 V/3.35-12 V 5 V/5-12 V | 8MHz 16MHz |
| 16 | LilyPad | ATmega168V ATmega328V | 2.7-5.5 V/2.7-5.5 V | 8MHz |
| 17 | LilyPad USB | ATmega32u4 | 3.3 V/3.8-5V | 8MHz |
| 18 | LilyPad Simple | ATmega328 | 2.7-5.5 V/2.7-5.5 V | 8MHz |
| 19 | LilyPad SimpleSnap | ATmega328 | 2.7-5.5 V/2.7-5.5 V | 8MHz |
| 20 | Yun | ATmega32u4 | 5 V | 16MHz |

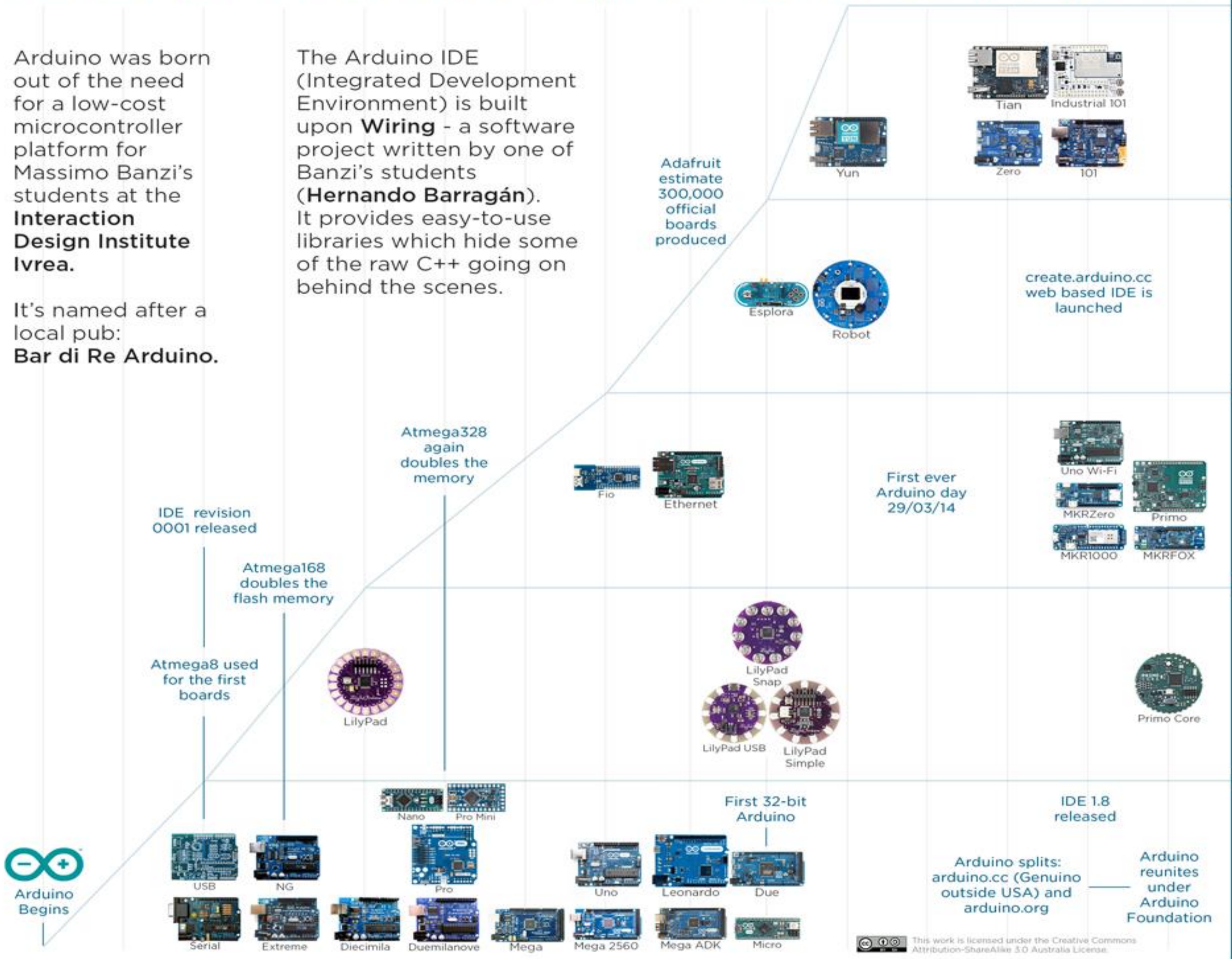
# HISTORY OF ARDUINO

2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017

Arduino was born out of the need for a low-cost microcontroller platform for Massimo Banzi's students at the **Interaction Design Institute Ivrea.**

It's named after a local pub: **Bar di Re Arduino.**

The Arduino IDE (Integrated Development Environment) is built upon **Wiring** - a software project written by one of Banzi's students (**Hernando Barragán**). It provides easy-to-use libraries which hide some of the raw C++ going on behind the scenes.



Atmega8 used for the first boards

IDE revision 0001 released

Atmega168 doubles the flash memory

Atmega328 again doubles the memory

Adafruit estimate 300,000 official boards produced

First 32-bit Arduino

First ever Arduino day 29/03/14

create.arduino.cc web based IDE is launched

IDE 1.8 released

Arduino splits: arduino.cc (Genuino outside USA) and arduino.org

Arduino reunites under Arduino Foundation

## ARDUINO TODAY

### Industrial

- Yun/Yun Mini
- Zero
- M0/M0 Pro
- Tian
- 101/Industrial 101

Powerful, smart technology

Rapid prototyping

Easily integrated with other devices

### Educational

- Esplora
- Robot

Classroom friendly

Modern, STEM learning

Hands-on and intuitive

### IoT

- MKR1000
- MKRZero
- MKRFOX1200
- Uno Wi-Fi
- Ethernet
- Primo

Connectivity and communication

Low power consumption

Easy to prototype with

### Wearables

- LilyPad
- LilyPad Simple
- LilyPad Snap
- LilyPad USB
- Primo Core

Thin, compact form factor

Battery powered

Easy to use with condutive material

### Maker

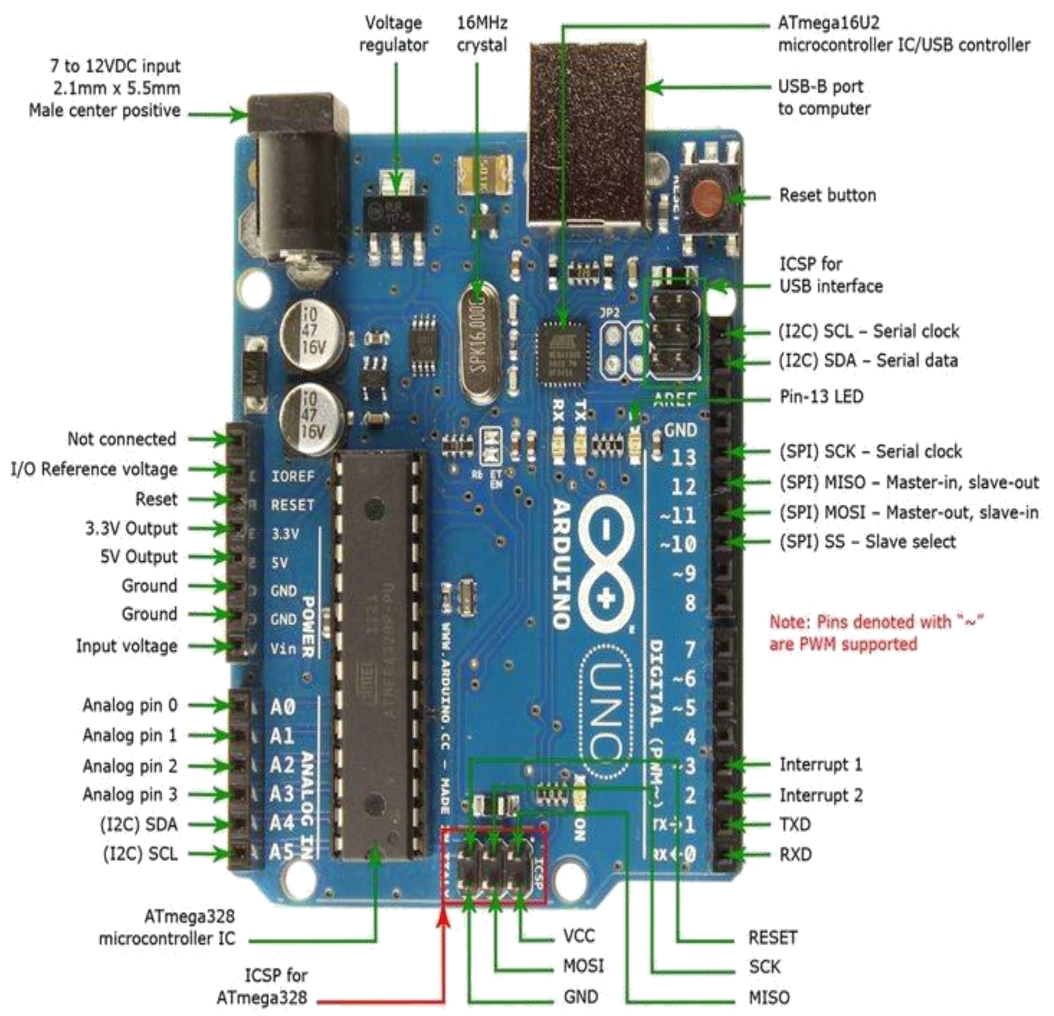
- Uno
- Leonardo
- Mini/Pro Mini
- Nano/Micro
- Mega2560/ADK
- Primo
- Due

Affordable

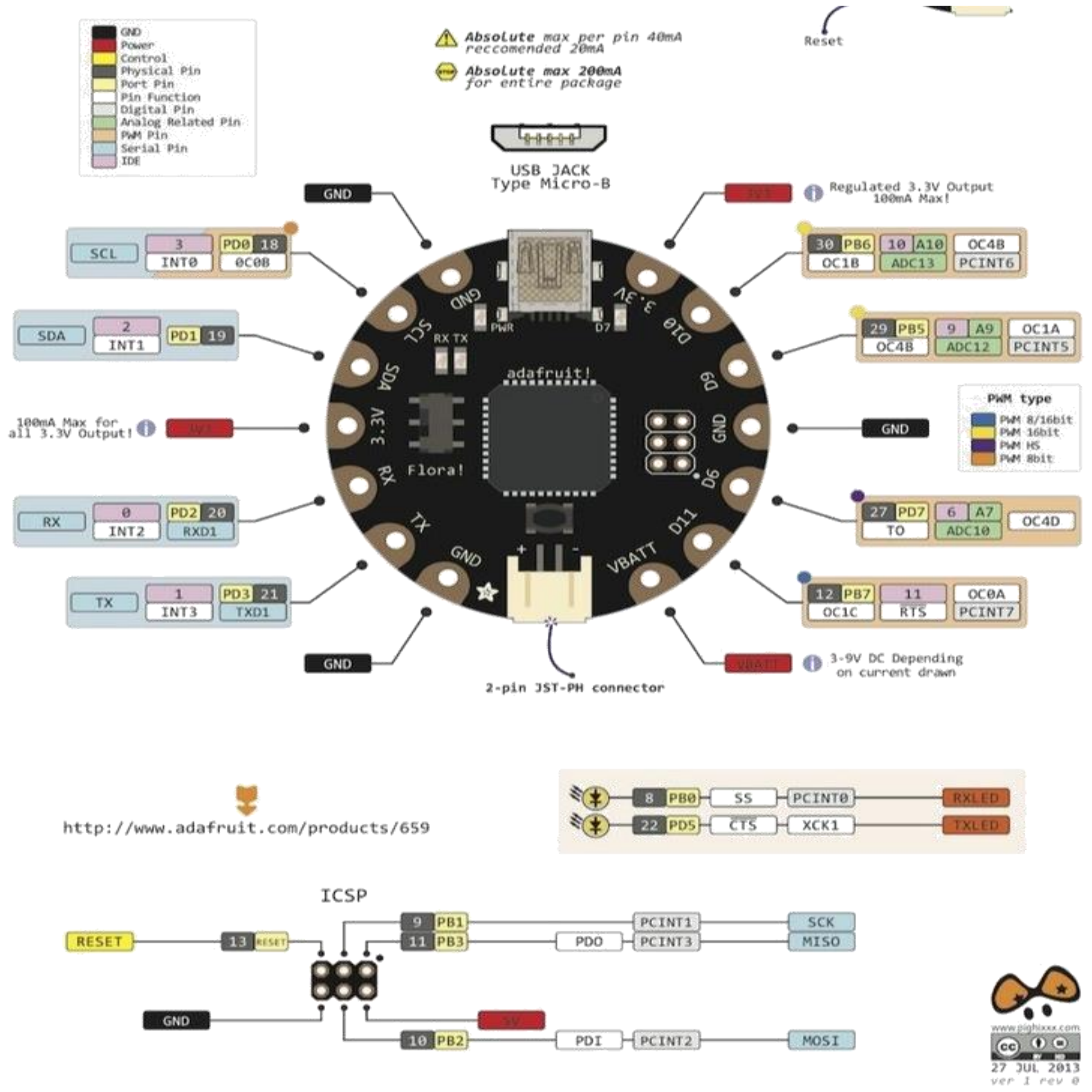
Community driven

Modular and adaptable

## UNO



## Arduino LilyPad



# Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
http://arduino.cc/en/Reference/

## Structure & Flow

### Basic Program Structure

```
void setup() {
  // runs once when sketch starts
}
void loop() {
  // runs repeatedly
}
```

### Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
do { ... } while ( x < 5);
for (int i = 0; i < 10; i++) { ... }
break;  // exit a loop immediately
continue;  // go to next iteration
switch (myVar) {
   case 1:
     ...
     break;
   case 2:
     ...
     break;
   default:
     ...
}
return x; // just return; for voids
```

## Operators

### General Operators

```
=  (assignment operator)
+  (add)             -  (subtract)
*  (multiply)        /  (divide)
%  (modulo)
== (equal to)        != (not equal to)
<  (less than)       >  (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and)    || (or)    ! (not)
```

### Compound Operators

```
++ (increment)
-- (decrement)
+= (compound addition)
-= (compound substraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)
```

### Bitwise Operators

```
& (bitwise and)    | (bitwise or)
^ (bitwise xor)    ~ (bitwise not)
<< (shift left)    >> (shift right)
```

## Built-in Functions

### Pin Input/Output

```
Digital I/O (pins: 0-13 A0-A5)
  pinMode(pin,[INPUT, OUTPUT])
  int digitalread(pin)
  digitalWrite(pin, value)
    // Write HIGH to an input to
    // enable pull-up resistors
Analog In (pins: 0-5)
  int analogRead(pin)
  analogReference(
    [DEFAULT, INTERNAL, EXTERNAL])
PWM Out (pins: 3 5 6 9 10 11)
  analogWrite(pin, value)
```

### Advanced I/O

```
tone(pin, freqhz)
tone(pin, freqhz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST,LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH,LOW])
```

### Time

```
unsigned long millis()
  // overflows at 50 days
unsigned long micros()
  // overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

### Math

```
min(x, y)     max(x, y)     abs(x)
sin(rad)      cos(rad)      tan(rad)
sqrt(x)       pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

### Random Numbers

```
randomSeed(seed) // long or int
long random(max)
long random(min, max)
```

### Bits and Bytes

```
lowByte(x)     highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

### Type Conversions

```
char()         byte()
int()          word()
long()         float()
```

### External Interrupts

```
attachInterrupt(interrupt, func,
 [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

## Libraries

### Serial (communicate with PC or via RX/TX)

```
begin(long Speed) // up to 115200
end()
int available() // #bytes available
byte read() // -1 if none available
byte peek()
flush()
print(myData)
println(myData)
write(myBytes)
SerialEvent() // called if data rdy
```

### SoftwareSerial (serial comm. on any pins)

```
  (#include <softwareSerial.h>)
SoftwareSerial(rxPin, txPin)
begin(long Speed) // up to 115200
listen()        // Only 1 can listen
isListening() // at a time.
read, peek, print, println, write
  // all like in Serial library
```

### EEPROM (#include <EEPROM.h>)

```
byte read(intAddr)
write(intAddr, myByte)
```

### Servo (#include <Servo.h>)

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()     // 0 to 180
bool attached()
detach()
```

### Wire (I²C comm.) (#include <Wire.h>)

```
begin()       // join a master
begin(addr) // join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(myByte)                // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission()           // Step 3
int available() // #bytes available
byte receive()  // get next byte
onReceive(handler)
onRequest(handler)
```

## Variables, Arrays, and Data

### Data types

```
void
boolean  (0, 1, true, false)
char     (e.g. 'a' -128 to 127)
int      (-32768 to 32767)
long     (-2147483648 to 2147483647)
unsigned char  (0 to 255)
byte           (0 to 255)
unsigned int   (0 to 65535)
word           (0 to 65535)
unsigned long  (0 to 4294967295)
float    (-3.4028e+38 to 3.4028e+38)
double   (currently same as float)
```

### Qualifiers

```
static     (persists between calls)
volatile   (in RAM (nice for ISR))
const      (make read only)
PROGMEM    (in flash)
```

### Arrays

```
int myInts[6]; // array of 6 ints
int myPins[]={2, 4, 8, 3, 6};
int mySensVals[6]={2, 4, -8, 3, 2};
myInts[0]=42;  // assigning first
               // index of myInts
myInts[6]=12;  // ERROR! Indexes
               // are 0 though 5
```

### Constants

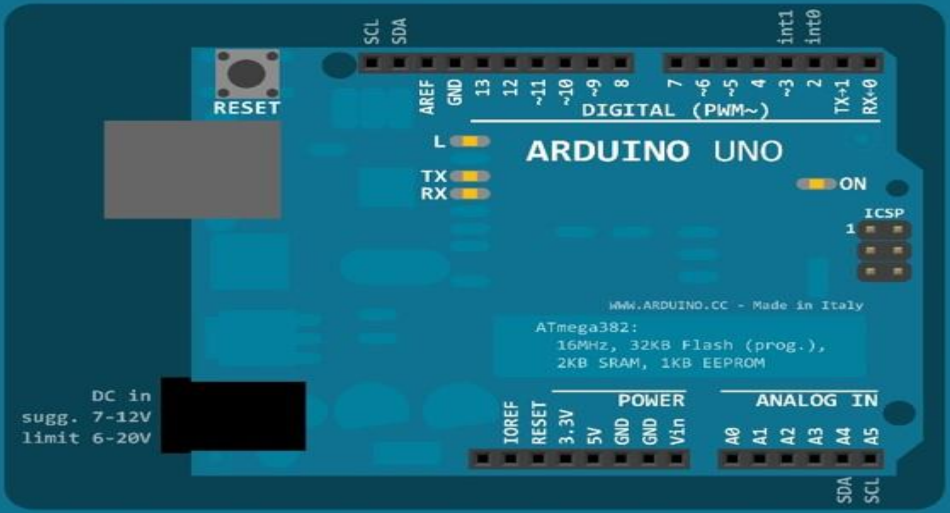
```
HIGH  | LOW
INPUT | OUTPUT
true  | false
143           (Decimal)
0173          (Octal - base 8)
0b11011111    (Binary)
0x7B          (Hexadecimal - base 16)
7U            (force unsigned)
10L           (force long)
15UL          (force long unsigned)
10.0          (force floating point)
2.4e5         (2.4*10^5 = 240000)
```

### Pointer Access

```
&  (reference: get a pointer)
*  (dereference: follow a pointer)
```

### Strings

```
char S1[8] =
 {'A','r','d','u','i','n','o'};
 // unterminated string; may crash
char S2[8] =
 {'A','r','d','u','i','n','o','\0'};
 // includes \0 null termination
char S3[]="Arduino";
char S4[8]="Arduino";
```



# ARDUINO CHEAT SHEET

Content for this Cheat Sheet provided by Gavin from Robots and Dinosaurs.
For more information visit: http://arduino.cc/en/Reference/Extended

sparkfun. ELECTRONICS

## Structure
void **setup**() void **loop**()

## Control Structures
```
if (x<5){ } else { }
switch (myvar) {
    case 1:
    break;
    case 2:
    break;
    default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } while (x<5);
continue; //Go to next in
do/for/while loop
return x; // Or 'return;' for voids.
goto // considered harmful :-)
```

## Further Syntax
// (single line comment)
/* (multi-line comment) */
#**define** DOZEN 12 //Not baker's!
#**include** <avr/pgmspace.h>

## General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) || (or) ! (not)

## Pointer Access
& reference operator
* dereference operator

## Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

## Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

## Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
**0b**11011111 //**Binary**
0x7B // **Hex** number
**7U** // Force unsigned
10L // Force long
15**UL** // Force long unsigned
10**.**0 // Forces floating point
2**.**4e5 // 240000

## Data Types
**void**
**boolean** (0, 1, false, true)
**char** (e.g. 'a' -128 to 127)
**unsigned char** (0 to 255)
**byte** (0 to 255)
**int** (-32,768 to 32,767)
**unsigned int** (0 to 65535)
**word** (0 to 655word (0 to 65535)
**long** (-2,147,483,648 to 2,147,483,647)
**unsigned long** (0 to 4,294,967,295)
**float** (-3.4028235E+38 to 3.4028235E+38)
**double** (currently same as float)
**sizeof**(myint) // returns 2 bytes

## Strings
```
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
//Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";
```

## Arrays
```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
```

## Conversion
**char**() **byte**()
**int**() **word**()
**long**() **float**()

## Qualifiers
**static** // persists between calls
**volatile** // use RAM (nice for ISR)
**const** // make read-only
**PROGMEM** // use flash

## Digital I/O
**pinMode**(pin, [INPUT,OUTPUT])
**digitalWrite**(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

## Analog I/O
**analogReference**([DEFAULT, INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source.
**analogWrite**(pin, value) // PWM

## Advanced I/O
**tone**(pin, freqhz)
**tone**(pin, freqhz ,duration_ms)
**noTone**(pin)
**shiftOut**(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin,[HIGH,LOW])

## Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
**delay**(ms)
**delayMicroseconds**(us)

## Math
**min**(x, y) **max**(x, y) **abs**(x)
**constrain**(x, minval, maxval)
**map**(val, fromL, fromH, toL, toH)
**pow**(base, exponent) **sqrt**(x)
**sin**(rad) **cos**(rad) **tan**(rad)

## Random Numbers
**randomSeed**(seed) // Long or int
long **random**(max)
long **random**(min, max)

## Bits and Bytes
**lowByte**()
**highByte**()
**bitRead**(x,bitn)
**bitWrite**(x,bitn,bit)
**bitSet**(x,bitn)
**bitClear**(x,bitn)
**bit**(bitn) //bitn: 0-LSB 7-MSB

## External Interrupts
**attachInterrupt**(interrupt, function, [LOW,CHANGE,RISING,FALLING])
**detachInterrupt**(interrupt)
**interrupts**()
**noInterrupts**()

## Libraries:

**Serial.**
**begin**([300, 1200, 2400, 4800, 9600,14400, 19200, 28800, 38400, 57600,115200])
**end**()
int **available**()
int **read**()
**flush**()
**print**()
**println**()
**write**()

**EEPROM** (#include <EEPROM.h>)
byte **read**(intAddr)
**write**(intAddr,myByte)

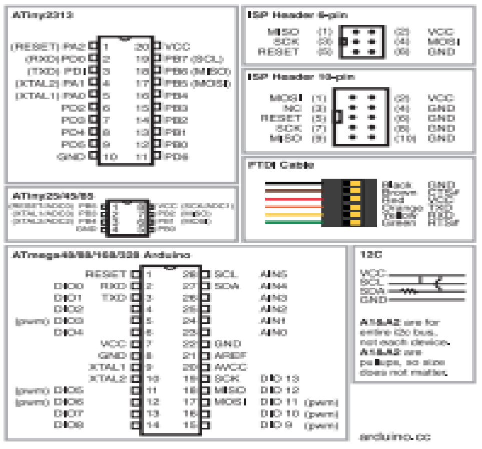
**Servo** (#include <Servo.h>)
**attach**(pin , [min_uS, max_uS])
**write**(angle) // 0-180
**writeMicroseconds**(uS) //1000-2000,1500 is midpoint
**read**() // 0-180
**attached**() //Returns boolean
**detach**()

**SoftwareSerial**(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
**begin**(longSpeed) // up to 9600
char **read**() // blocks till data
**print**(myData) or **println**(myData)

**Wire** (#include <Wire.h>) // For **I2C**
**begin**() // Join as master
**begin**(addr) // Join as slave @ addr
**requestFrom**(address, count)
**beginTransmission**(addr) // Step 1
**send**(mybyte) // Step 2
**send**(char * mystring)
**send**(byte * data, size)
**endTransmission**() // Step 3
byte **available**() // Num of bytes
byte **receive**() //Return next byte
**onReceive**(handler)
**onRequest**(handler)

| | ATMega168 | ATMega328 | ATMega1280 |
|---|---|---|---|
| Flash (2k for boobtloader) | 16kB | 32kB | 128kB |
| SRAM | 1kB | 2kB | 8kB |
| EEPROM | 512B | 1kB | 4kB |

| | Duemilanove/ Nano/ Pro/ ProMini | Mega |
|---|---|---|
| # of IO | 14 + 6 analog (Nano has 14 + 8) | 54 + 16 analog |
| Serial Pins | 0 - RX<br>1 - TX | 0 - RX1 1 - TX1<br>19 - RX2 18 - TX2<br>17 - RX3 16 - TX3<br>15 - RX4 14 - TX4 |
| Ext Interrupts | 2 - (Int 0)<br>1 - (Int 1) | 2,3,21,20,19,18 (IRQ0 - IFQ5) |
| PWM Pins | 5,6 - Timer 0<br>9,10 - Timer 1<br>3,11 - Timer 2 | 0 - 13 |
| SPI | 10 - SS<br>11 - MOSI<br>12 - MISO<br>13 - SCK | 53 - SS<br>51 - MOSI<br>50 - MISO<br>52 - SCK |
| I2C | Analog4 - SDA<br>Analog5 - SCK | 20 - SDA<br>21 - SCL |



arduino.cc