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) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions
```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	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 subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

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

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings
```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

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

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

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5
```
pinMode(pin,
  [INPUT, OUTPUT, INPUT_PULLUP])
int digitalread(pin)
digitalWrite(pin, [HIGH, LOW])
```

Analog In - pins A0-A5
```
int analogRead(pin)
analogReference(
  [DEFAULT, INTERNAL, EXTERNAL])
```

PWM Out - pins 3 5 6 9 10 11
```
analogWrite(pin, value)
```

Advanced I/O
```
tone(pin, freq_Hz)
tone(pin, freq_Hz, 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)
```

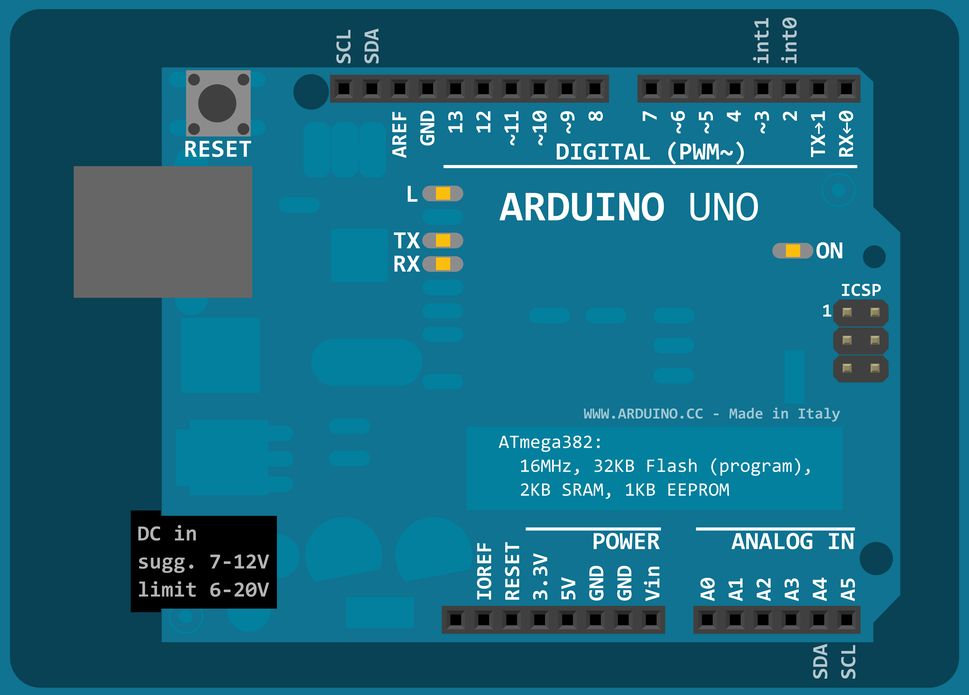
Math
```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers
```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
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(val)       byte(val)
int(val)        word(val)
long(val)       float(val)
```

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



Libraries

Serial - comm. with PC or via RX/TX
```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - comm. on any pin
```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()       // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - access non-volatile memory
```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - control servo motors
```
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()
```

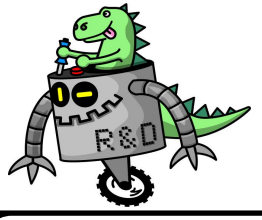
Wire.h - I²C communication
```
begin()     // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```


by Mark Liffiton

Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org

Arduino Cheat Sheet V.02c

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace



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
0b11011111 //**Binary**
0x7B // **Hex** number
7**U** // Force unsigned
10**L** // 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 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)

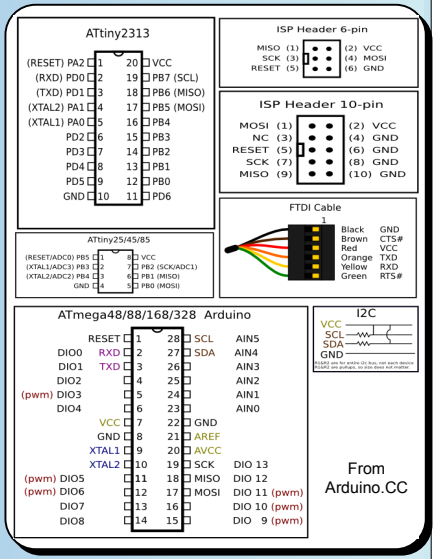
Servo (#include <Servo.h>)
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000, 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()

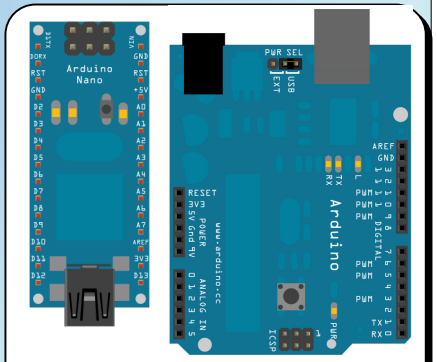
SoftwareSerial(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
begin(longSpeed) // up to 9600
char **read**() // blocks till data
print(myData) or **println**(myData)

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 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL





Pics from Fritzing.Org under C.C. license

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

ARDUINO CHEAT SHEET

JEROEN DOGGEN, AP UNIVERSITY COLLEGE ANTWERP

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Structure

void **setup**()
void **loop**()

Control Structures

```
if(x<5){}
for(int i = 0; i < 255; i++ ){}
while((x < 6 ){}
```

Further Syntax

// Single line comment
/ * .. * / Multi line comment
#define ANSWER 42
#include <myLib.h>

General Operators

=	assignment
+, −	addition, substraction
*, /	multiplication, division
%	modulo
==	equal to
! =	not equal to
<	less than
<=	less than or equal to

Pointer Access

&	reference operator
*	dereference operator

Bitwise Operators

&	bitwise AND
|	bitwise OR
∧	bitwise XOR
∼	bitwise NOT

Compound Operators

++	Increment
−−	Decrement
+ =	Compound addition
& =	Compound bitwise AND

Constants

```
HIGH, LOW
INPUT, OUTPUT
true, false
```

53 : Decimal
B11010101 : Binary
0x5BA4 : Hexadecimal

Data Types

void	
boolean	0, 1, false, true
char	e.g. 'a' -128 → 127
unsigned char	0 → 255
int	-32.768 → 32.767
unsigned int	0 → 65535
long	-2.147.483.648 → 2.147.483.647
float	-3,4028235E+38 → 3.402835E+38
sizeof (myint)	returns 2 bytes

Arrays

int myInts[6];
int myPins[]=2,4,8,5,6;
int myVals[6]=2,-4,9,3,5;

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';
char S4[]="Arduino";
char S5[8] = "Arduino";
char S6[15] = "Arduino";

Conversion

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

Qualifiers

static	Persist between calls
volatile	Use RAM (nice for ISR)
const	Mark read-only
PROGMEM	Use flash memory

Interrupts

attachInterrupt(interrupt, function, type)
detachInterrupt(interrupt)
boolean(interrupt)
interrupts()
noInterrupts()

Advanced I/O

tone(pin, freqhz)
tone(pin, freqhz, duration_ms)
noTone(pin)
shiftOut (dataPin, clockPin, how, value)
unsigned long **pulseIn**(pin, [HIGH,LOW])

Time

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

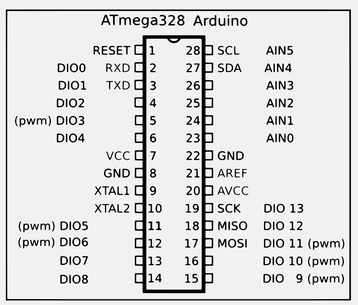
Math

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

Pseudo Random Numbers

randomSeed(seed)
long **random**(max)
long **random**(min, max)

ATmega328 Pinout



I/O Pins

	Uno	Mega
# of IO	14 + 6	54 + 11
Serial Pins 3	0 - RX, 1 -TX	RX1 → RX4
Interrupts	2,3	2,3,18,19,20,21
PWM Pins	5,6 - 9,10 - 3,11	0 → 13
SPI (SS, MOSI, MISO, SCK)	10→ 13	50→ 53
I2C (SDA, SCK)	A4, A5	20,21

Analog I/O

```
analogReference(EXTERNAL, INTERNAL)
analogRead(pin)
analogWrite(pin, value)
```

Digital I/O

```
pinMode(pin,[INPUT,OUTPUT])
digitalRead(pin)
digitalWrite(pin, value)
```

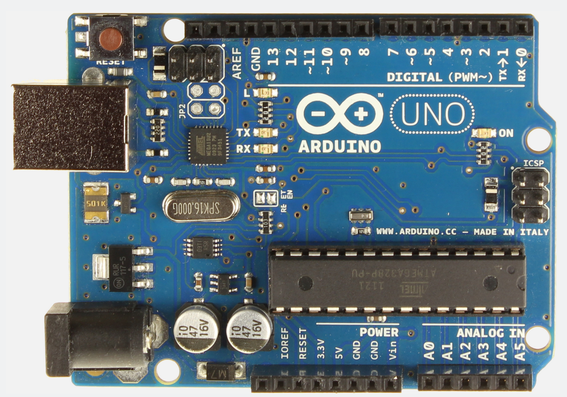
Serial Communication

Serial.begin(speed)
Serial.print("Text")
Serial.println("Text")

Websites

```
forum.arduino.cc
playground.arduino.cc
arduino.cc/en/Reference
```

Arduino Uno Board



ARDUINO CHEAT SHEET

For more information visit: http://arduino.cc/en/Reference/

Structure

```
/* Each Arduino sketch must contain the
following two functions. */
void setup()
{
/* this code runs once at the beginning of
the code execution. */
}

void loop()
{
/* this code runs repeatedly over and over
as long as the board is powered. */
}
```

Comments

```
// this is a single line
/* this is
a multiline */
```

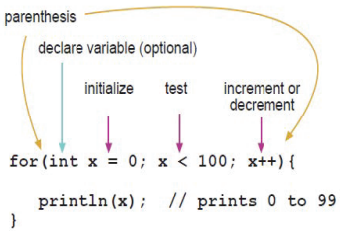
Setup

```
pinMode(pin, [INPUT \ OUTPUT \ INPUT_PUL-
LUP]);
/* Sets the mode of the digital I/O pin.
It can be set as an input, output, or an
input with an internal pull-up resistor.
*/
```

Control Structures

```
if(condition)
{
// if condition is TRUE, do something here
}
else
{
// otherwise, do this
}

for(initialization; condition; increment)
{
// do this
}
/* The 'for' statement is used to repeat
a block of statements enclosed in curly
braces. An increment counter is usually
used to increment and terminate the loop.
*/
```



Digital I/O

```
digitalWrite(pin, val);
/* val = HIGH or LOW write a HIGH or a LOW
value to a digital pin. */
int var = digitalRead(pin);
/* Reads the value from a specified digital
pin, either HIGH or LOW. */
```

Analog I/O

```
analogWrite(pin, val);
/* Writes an analog value to a pin.
val = integer value from 0 to 255 */
int var = analogRead(pin);
/* Reads the value from the specified
analog pin. */
```

Advanced I/O

```
tone(pin, freq);
/* Generates a square wave of the specified
frequency to a pin. Pin must be one of the
PWM (~) pins. */
tone(pin, freq, duration);
/* Generates a square wave of the specified
frequency to a pin for a duration in
milliseconds. Pin must be one of the PWM (~)
pins. */
noTone(pin);
// Turns off the tone on the pin.
```

Time

```
delay(time_ms);
/* Pauses the program for the amount of time
(in milliseconds). */
delayMicroseconds(time_us);
/* Pauses the program for the amount of time
(in microseconds). */
millis();
/* Returns the number of milliseconds since
the board began running the current program.
max: 4,294,967,295 */
micros();
/* Returns the number of microseconds since
the board began running the current program.
max: 4,294,967,295 */
```

Data Types

```
void      // nothing is returned
boolean   // 0, 1, false, true
char      // 8 bits: ASCII character
byte      // 8 bits: 0 to 255, unsigned
int       // 16 bits: 32,768 to 32,767, signed
long      /* 32 bits: 2,147,483,648
             to 2,147,483,647, signed */
float     // 32 bits, signed decimal
```

Constants

```
HIGH \ LOW
INPUT \ OUTPUT
true \ false
```

Mathematical Operators

```
=   // assignment
+   // addition
-   // subtraction
*   // multiplication
/   // division
%   // modulus
```

Logical Operators

```
==  // boolean equal to
!=  // not equal to
<   // less than
>   // greater than
<=  // less than or equal to
>=  // greater than or equal to
&&  // Boolean AND
||  // Boolean OR
!   // Boolean NOT
```

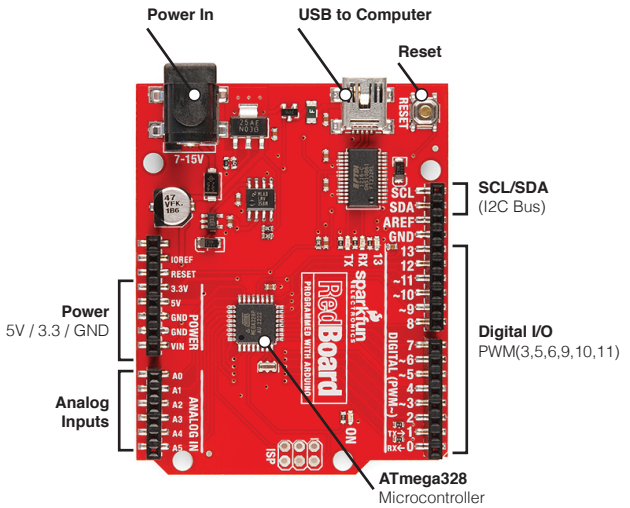
Bitwise Operators

```
&     // bitwise AND
|     // bitwise OR
^     // bitwise XOR
~     // bitwise INVERT
var << n    // bitwise shift left by n bits
var >> n    // bitwise shift right by n bits
```

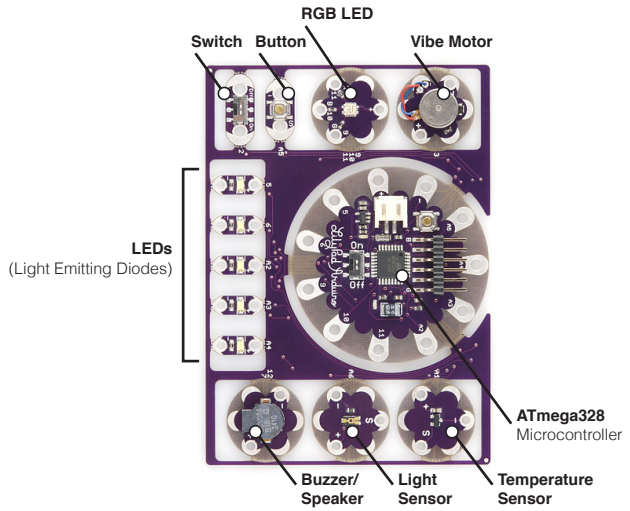
Libraries

```
#include <libraryname.h>
/* this provides access to special
additional functions for things such as
servo motors, SD card, wifi, or bluetooth.
*/
```

RedBoard:



LilyPad ProtoSnap Simple:



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
0b11011111 //**Binary**
0**x**7B // **Hex** number
7**U** // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4**e**5 // 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)

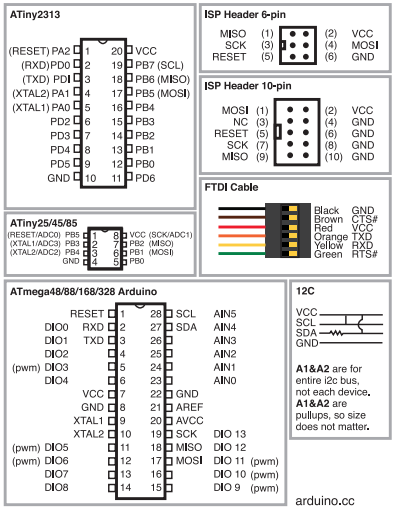
Servo (#include <Servo.h>)
  attach(pin , [min_uS, max_uS])
  write(angle) // 0-180
  writeMicroseconds(uS) //1000-2000,1500 is midpoint
  read() // 0-180
  attached() //Returns boolean
  detach()

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 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 1 - (Int 1)	2,3,21,20,19,18 (IRQ0 - IFQ5)
PWM Pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0 - 13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL



arduino.cc