A convenient arduino cheatsheet, v1.0

Program Flow / Control

```
/* Basic sketch structure */
void setup()
{ // runs only once.
}
void loop()
{ // runs repeatedly.
}
```

Basic Logic

Simple `if()`-else

```
if(condition)
{ //true condition code here
} else
{ //false statement code here
}
```

Functions

Declaration:

```
<return type> function_name ([arguments])
```

return type: returned value type or ***void***

arguments: list of arguments, preceded by the corresponding types

Examples:

```
int addNum(int a, int b) { return value; }
void returnLess(byte a) { return; }
```

Looping

```
while(condition)
{ }
for(init; condition; update variable)
{ }
continue; // jumps to the next loop iteration
break; // exits a loop
```

Pin Configuration - INPUT vs OUTPUT

```
pinMode(pin, INPUT/OUTPUT/INPUT_PULLUP);
```

Reading INPUTs

```
buttonPress = digitalRead(pin); // any pin
sensorVal = analogRead(pin); // A0-A5 pins
```

DEI Departamento de Engenharia Informática

OUTPUT Control (and PWM)

```
digitalWrite(pin, val); // val: HIGH or LOW
analogWrite(pin, val); // val: 0 to 255.
```

Data / Variable Types

```
void // null data type
byte // small integer, unsigned
int // integer, signed
long // big integer, signed
float // floating point / decimal numbers
String // array of characters
char // character
bool or boolean //holds either true of false
<type> arrayName[] //array of <type> elements
```

Timing

```
delay(time_millis); // pauses program in ms
millis();  //returns # of milliseconds (long)
```

Communications

```
Serial.begin(baudrate);
Serial.print(); // print data out
Serial.println(); // print with new line
Serial.println(val,base); // base BIN,HEX,DEC
x = Serial.read(); // reads a single byte/char
x = Serial.readStringUntil(terminator);
// reads String from the serial buffer, until
the terminator character is found (such as '\n')
```

Strings

```
myString.toInt(); //converts string to int
myString.trim(); // removes leading/trailing
whitespaces
myString.length(); // string size, in bytes
```

String constructor:

```
String(val); // converts val to string
String(val, base); // same, but specific base
```

Math Operators

+	addition	-	subtraction
*	multiplication	/	division
%	modulus	=	Assignment

Useful functions

```
random([min,]max);//gets random number (long)
randomSeed(number); //initializes the
pseudo-random number generator
abs(value); // returns absolute value
sizeof(variable); // returns size of a variable
type or array, in bytes (size_t)
```

Logic Operators

==	is equal to?	>	greater than
!=	is not equal to?	<=	less than or equal
<	less than	>=	greater than or equal
&&	compound AND	||	compound OR
!	NOT (inverse)		

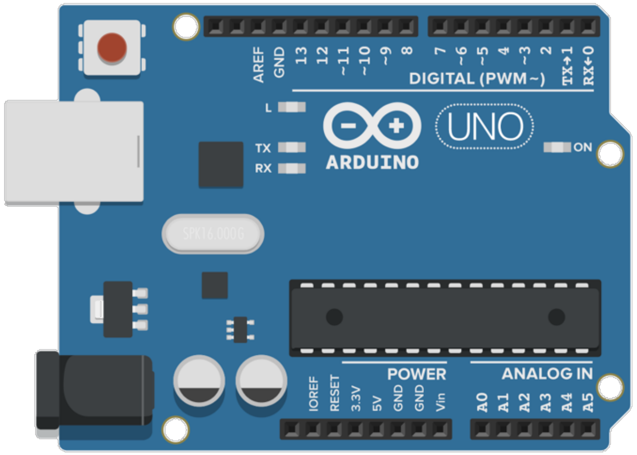
Bitwise Operators

<<	Shift left	>>	Shift right
&	AND	|	OR
^	XOR	~	NOT

Bit manipulation (note: *var* is the target variable)

```
bitSet(var, n) // sets 1 in position n
bitRead(var, n) // reads bit in position n
bitWrite(var, n, b) // writes value b in position n
bitClear(var, n) // sets 0 in in position n
```

Pinout



Examples

Basic button/LED control

```
const int buttonPin = 2;
const int ledPin =  13;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Print all elements stored in an array

```
const int sizeOFarray = 5;
int b[sizeOFarray] = {10, 20, 30, 40, 50};
int sum = 0;

void setup ()
{
    Serial.begin(9600);
}
void loop ()
{
   for ( int i = 0; i < sizeOFarray; i++ )
      sum += b[ i ];  // here, sum = sum + b[i]
   Serial.print('Sum of total elements of an array:') ;
   Serial.print(sum) ;
}
```

Debounced button/LED control

```
const int buttonPin = 2;
const int ledPin = 13;

int ledState = HIGH;
int buttonState;
int lastButtonState = LOW;

unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);
}

void loop() {
  int reading = digitalRead(buttonPin);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
    }
  }
  digitalWrite(ledPin, ledState);
  lastButtonState = reading;
}
```