# Arduino Cheatsheet

A quick reference guide for Arduino development, covering essential concepts, code snippets, and hardware information for both beginners and advanced users.

## Arduino Fundamentals

### Basic Structure

Every Arduino program (sketch) has two essential functions:

`void setup() {}` - Runs once at the beginning of the program. Used for initialization.

`void loop() {}` - Runs repeatedly after `setup()` finishes. This is where the main program logic goes.

**Example:**

```
void setup() {
  // Initialize serial communication
  Serial.begin(9600);
}

void loop() {
  // Print "Hello, world!" to the serial
monitor
  Serial.println("Hello, world!");
  delay(1000); // Wait for 1 second
}
```

### Data Types

| | |
|---|---|
| `int` | Integer (whole number). Typically 2 bytes (-32,768 to 32,767). |
| `byte` | Unsigned integer (0 to 255). |
| `long` | Long integer. Typically 4 bytes (-2,147,483,648 to 2,147,483,647). |
| `float` | Floating-point number (number with decimal point). 4 bytes. |
| `boolean` | Boolean value (true or false). |
| `char` | Character. 1 byte. |

### Digital I/O

| | |
|---|---|
| `pinMode(pin, mode)` | Sets the specified pin as `INPUT`, `OUTPUT`, or `INPUT_PULLUP`. |
| `digitalWrite(pin, value)` | Writes `HIGH` or `LOW` to a digital pin (only if the pin is set as `OUTPUT`). |
| `digitalRead(pin)` | Reads the value (`HIGH` or `LOW`) from a digital pin. |

## Analog I/O & Serial Communication

### Analog I/O

| | |
|---|---|
| `analogRead(pin)` | Reads the value from the specified analog pin (0 to 1023). |
| `analogWrite(pin, value)` | Writes an analog value (PWM signal) to a pin (0 to 255). Only works on PWM enabled pins (marked with ~ on the board). |
| `analogReference(type)` | Configures the reference voltage used for analog input (`DEFAULT`, `INTERNAL`, `EXTERNAL`). |

### Serial Communication

| | |
|---|---|
| `Serial.begin(baudRate)` | Initializes serial communication at the specified baud rate (e.g., 9600, 115200). |
| `Serial.print(data)` | Sends data to the serial port as human-readable ASCII text. |
| `Serial.println(data)` | Sends data to the serial port, followed by a carriage return and line feed. |
| `Serial.available()` | Gets the number of bytes (characters) available for reading from the serial port. |
| `Serial.read()` | Reads the first available byte of incoming serial data. |

### Time Functions

| | |
|---|---|
| `delay(milliseconds)` | Pauses the program for the amount of time (in milliseconds) specified as parameter. |
| `millis()` | Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero) after approximately 50 days. |
| `micros()` | Returns the number of microseconds since the Arduino board began running the current program. This number will overflow (go back to zero) after approximately 70 minutes. |

## Control Structures & Operators

### Control Structures

`if (condition) { ... }` - Executes code block if the condition is true.

`if (condition) { ... } else { ... }` - Executes one code block if the condition is true and another if the condition is false.

`for (initialization; condition; increment) { ... }` - Repeats a block of code a specific number of times.

`while (condition) { ... }` - Repeats a block of code as long as the condition is true.

`do { ... } while (condition);` - Executes a block of code once, and then repeats as long as the condition is true.

`switch (expression) { case value1: ... break; case value2: ... break; default: ... }` - Selects one of several code blocks to execute based on the value of an expression.

### Operators

| | |
|---|---|
| `=` | Assignment operator (assigns a value to a variable). |
| `==` | Equality operator (checks if two values are equal). |
| `!=` | Inequality operator (checks if two values are not equal). |
| `>` | Greater than operator. |
| `<` | Less than operator. |
| `&&` | Logical AND operator (returns true if both conditions are true). |
| `\|\|` | Logical OR operator (returns true if at least one condition is true). |
| `!` | Logical NOT operator (reverses the logical state of its operand). |

### Math Operators

| | |
|---|---|
| `+` | Addition. |
| `-` | Subtraction. |
| `*` | Multiplication. |
| `/` | Division. |
| `%` | Modulo (returns the remainder of a division). |

## Advanced Arduino

## Interrupts

| | |
|---|---|
| `attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)` | Attaches an interrupt to the specified pin. `ISR` is the interrupt service routine, and `mode` can be `LOW`, `CHANGE`, `RISING`, or `FALLING`. |
| `detachInterrupt(digitalPinToInterrupt(pin))` | Detaches the interrupt on the specified pin. |
| `interrupts()` | Re-enables interrupts (after they have been disabled by `noInterrupts()`). |
| `noInterrupts()` | Disables interrupts. |

## Libraries

Libraries provide extra functionality to your sketches. To include a library:

```
#include <LibraryName.h>
```

**Examples:**

- `#include <Wire.h>` - For I2C communication
- `#include <SPI.h>` - For SPI communication
- `#include <Servo.h>` - For controlling servo motors
- `#include <LiquidCrystal.h>` - For LCD display

## EEPROM

| | |
|---|---|
| `EEPROM.write(address, value)` | Writes a byte to the EEPROM at the specified address. |
| `EEPROM.read(address)` | Reads a byte from the EEPROM at the specified address. |
| `EEPROM.update(address, value)` | Writes a byte to the EEPROM at the specified address, only if the new value is different from the old value. |