



**DEEC**

DEPARTAMENTO DE ENGENHARIA  
ELETROTÉCNICA E DE COMPUTADORES

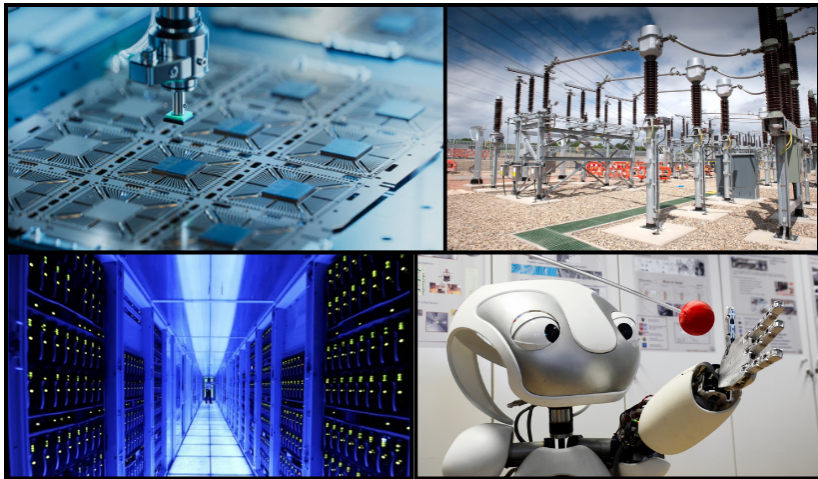
TÉCNICO LISBOA

# Arduino 101

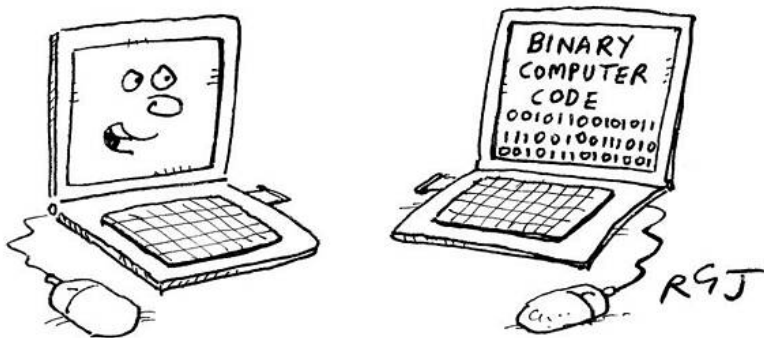
A sprint presentation on MCUs and I/O devices

DEEC-IST | HackerSchool | NEEC | IEEE-SB

# Why are we where?



# How do computers communicate?



"NOW YOU'RE TALKING MY LANGUAGE"

# How do computers communicate?

- ❖ **bit**: The most basic unit of computing information
  - ❖ Either 0
  - ❖ Or 1
- ❖ **byte**: Normally the smallest addressable unit of memory in most computing systems
  - ❖ It consists of **8** bits

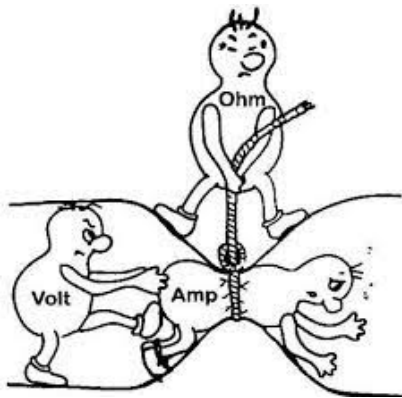


*"Oh, what the hell, I'll add another zero."*

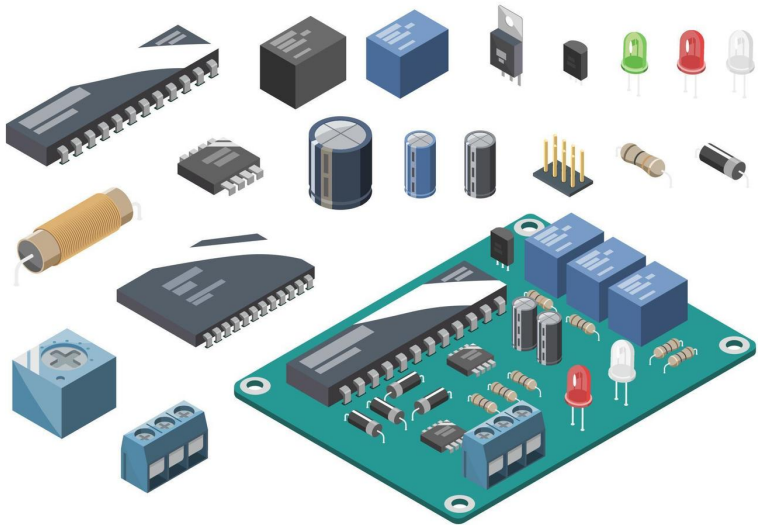


# Ohm's Law

We are Electrical and Computer Engineers, not Computer Scientists! We have to understand the underlying hardware too, lets start with the basics!

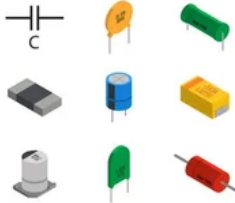


# Simple electronic components

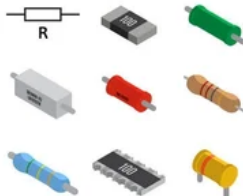


# Simple electronic components

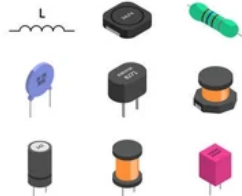
## Capacitors



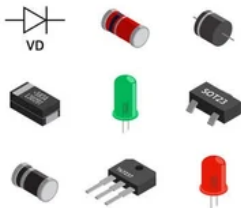
## Resistors



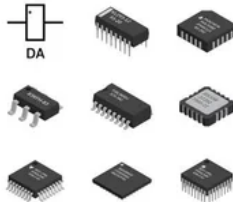
## Inductors



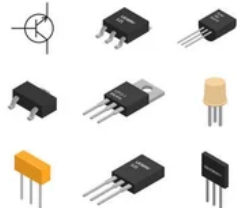
## Diodes



## Microchips



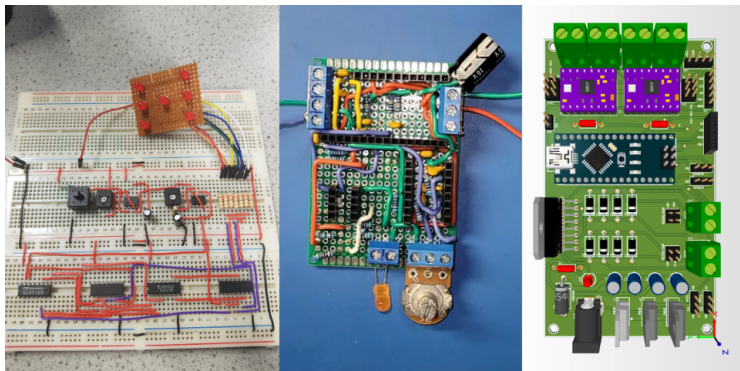
## Transistors



# But how can I make the connections?

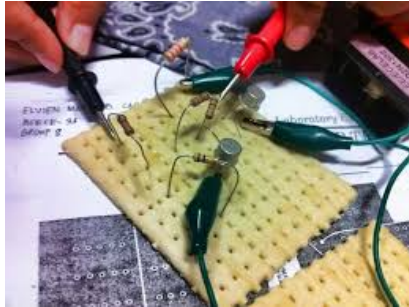
I need to have a conductive material between my components so they are electrically connected. . . But how?

Short Answer: **ECAD and Soldering**, in many shapes and forms.



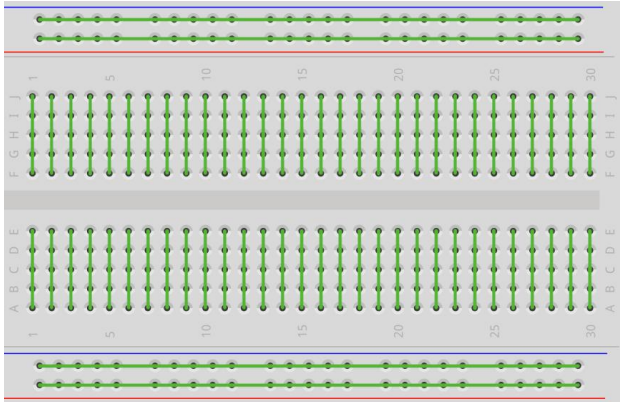
But is soldering viable for **prototyping**?

# Breadboard?



Use a breadboard they say. . . .

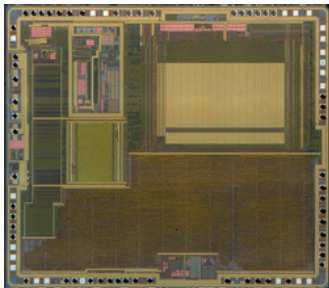
# Breadboard.



He now have a conductive array where we can connect cables to prototype our circuit!

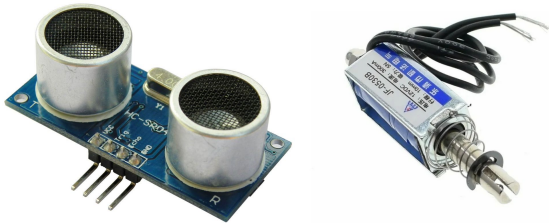
# What is a Microcontroller?

- ❖ Small embedded computer
- ❖ Generically includes:
  - ❖ processor (CPU)
  - ❖ memory (RAM/Flash)
  - ❖ and I/O pins
- ❖ Normally the code runs on the baremetal
- ❖ Alternatively it can run a simple, deterministic OS
  - ❖ **FreeRTOS**



# What are Sensors and Actuators?

- ❑ **Sensors:** collect data from the environment
- ❑ **Actuators:** perform actions over the environment





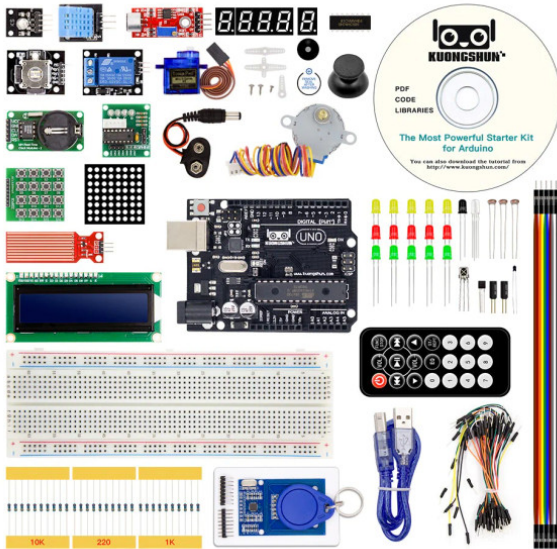
# What are Sensors and Actuators?

These devices can communicate with the MCU through various protocols:

- ❖ General Purpose I/O (GPIO) and ADC I/O
- ❖ Inter-Integrated Circuit (I<sup>2</sup>C)
- ❖ Serial Peripheral Interface (SPI)
- ❖ Universal Asynchronous Receiver-Transmitter (UART)
- ❖ etc. . .



## Some of the sensors you have....



## ATmega328P

- ❖ Single-core 8-bit AVR microprocessor
- ❖ 20 MHz max frequency
- ❖ 2 kB of on-chip SRAM for data and instructions
- ❖ 23 GPIO pins
- ❖ 6 channel 10-bit Analog-to-Digital converter

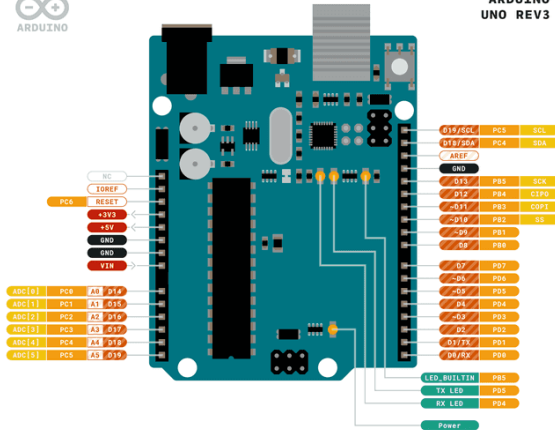
## Arduino UNO

- ❖ Board that integrates a ATmega328P chip
- ❖ Adds peripherals such as
  - ❖ Quartz Oscillator for clock signal
  - ❖ USB Controller
  - ❖ Interfaces for Power signals

# Arduino UNO Pinout



ARDUINO  
UNO REV3



Ground	Internal Pin	Digital Pin	Microcontroller's Port
Power	SWD Pin	Analog Pin	
LED	Other Pin	Default	



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or write to Creative Commons, P.O. Box 1868, Mountain View, CA 94042, USA.

## Setup and Microcontroller programming

For ease of use we will be using **Arduino IDE**

```
File Edit Sketch Tools Help

embedded

DallasTemperature waterTempSensor(1oneWire);

const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 0; //Replace with your GMT offset (seconds)

const int daylightOffset_sec = 3600;
//const int daylightOffset_sec = 0;

char verboseWaterTime[512];
String timeRTDB;
time_t waterTime, timeThreshold;
bool manualWater, enAutoWater, enWaterSaving, enWaterSchedule;
float turbThreshold, humiThreshold, tempThreshold, waterTempThreshold;
// Define the Firebase Data object
FirebaseData fdbdo;
FirebaseAuth auth;
FirebaseConfig config;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(pumpWater, OUTPUT);
  //pinMode(pumpRemove, OUTPUT);
  //pinMode(pumpAdd, OUTPUT);
  digitalWrite(pumpWater, INACTIVE);
  pinMode(waterLevel, INPUT);
  //digitalWrite(waterLevel, LOW);
  pinMode(waterTurbidity, INPUT);
  //digitalWrite(waterTurbidity, LOW);
  pinMode(waterTemperature, INPUT);

  Compiling sketch...

/home/jbcr/1ST/Meng/RMIC/GreenThumb/EmbeddedSystem/embedded/embedded.ino:57:1: warning: 'typedef' was ignored in this declaration
57 | typedef enum TankCode {NO_WATER, DIRTY_WATER, HOT_WATER};
    | ^~~~~~

3300cded, Use pgm_read macros for IRAMPROGMEM, dt (aka nodemcu, 26 MHz, 40MHz, DOUT (compatib), 1MB (FS:64KB OTA~470KB), 2, nonos-adv 2.2.1+100 (190703), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/ttyUSB0
```

# Digital I/O - How does it work?

The information can only be in two states:

- **HIGH** (between 5V and 3.0V)
- **LOW** (between 1.5V and 0V)

Our information is a **single binary digit** - a bit!

# Analog I/O - How does it work?

**Question:** If my ADCs have a 10-bit (binary digit) resolution, how many states can I have?

# Analog I/O - How does it work?

A bit represents one of two possible states

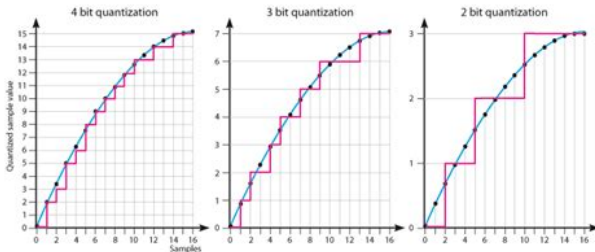
- **HIGH** (0)
- **LOW** (1)

Therefore with 10-bits:



# Analog I/O - How does it work?

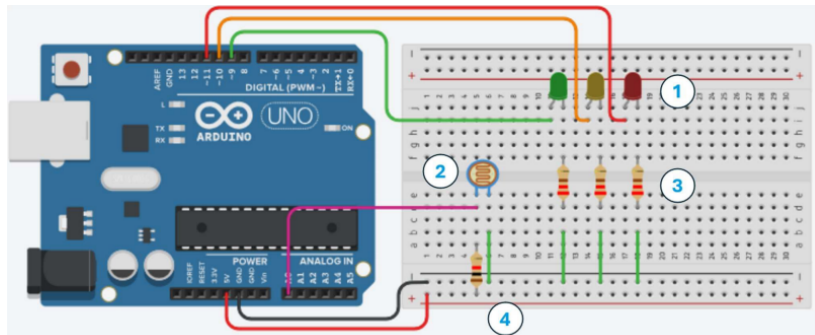
So our ADC acquired a **continuous** signal and converted it into a **discrete** one! This is called **quantization**!



# Let's build something! - Concept

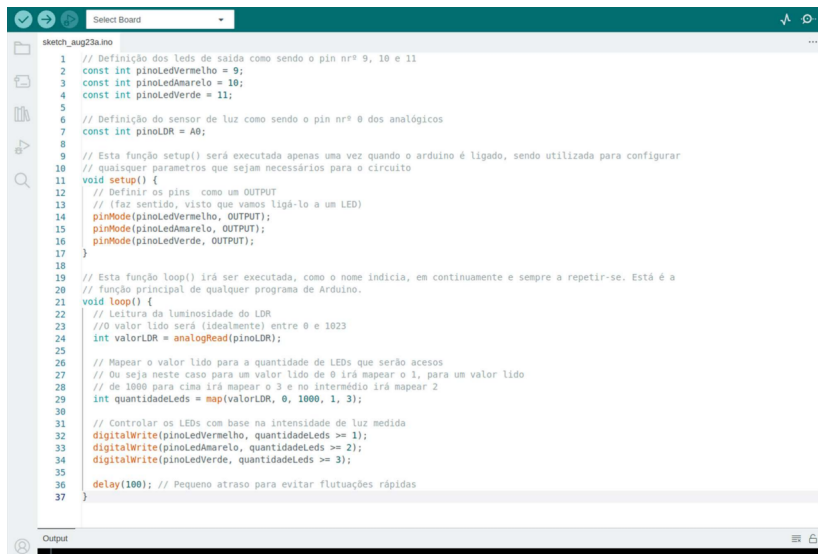
Let's build a **very** simple **luminosity detection system**

# Let's build something! - Schematic



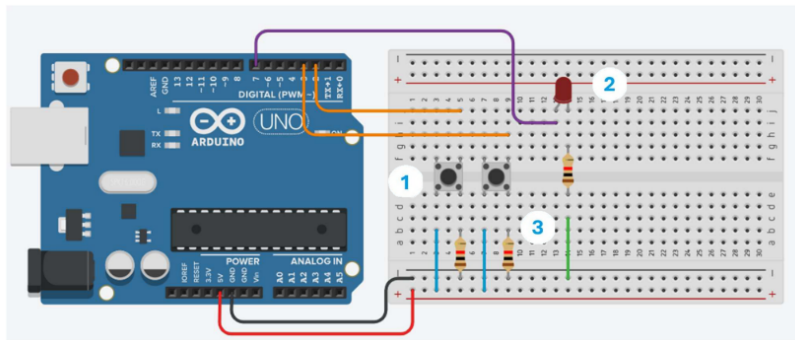
- ① 3 x LED's    ② 1 x LDR    ③ 3 x Resistência 220  $\Omega$     ④ 1 x Resistência 10k $\Omega$

# Let's build something! - Code



```
1 // Definição dos leds de saída como sendo o pin nº 9, 10 e 11
2 const int pinoLedVermelho = 9;
3 const int pinoLedAmarelo = 10;
4 const int pinoLedVerde = 11;
5
6 // Definição do sensor de luz como sendo o pin nº 0 dos analógicos
7 const int pinoLDR = A0;
8
9 // Esta função setup() será executada apenas uma vez quando o arduino é ligado, sendo utilizada para configurar
10 // quaisquer parametros que sejam necessários para o circuito
11 void setup() {
12     // Definir os pins como um OUTPUT
13     // (faz sentido, visto que vamos ligá-lo a um LED)
14     pinMode(pinoLedVermelho, OUTPUT);
15     pinMode(pinoLedAmarelo, OUTPUT);
16     pinMode(pinoLedVerde, OUTPUT);
17 }
18
19 // Esta função loop() irá ser executada, como o nome indicia, em continuamente e sempre a repetir-se. Está é a
20 // função principal de qualquer programa de Arduino.
21 void loop() {
22     // Leitura da luminosidade do LDR
23     //O valor lido será (idealmente) entre 0 e 1023
24     int valorLDR = analogRead(pinoLDR);
25
26     // Mapear o valor lido para a quantidade de LEDs que serão acesos
27     // Ou seja neste caso para um valor lido de 0 irá mapear o 1, para um valor lido
28     // de 1000 para cima irá mapear o 3 e no intermédio irá mapear 2
29     int quantidadeLeds = map(valorLDR, 0, 1000, 1, 3);
30
31     // Controlar os LEDs com base na intensidade de luz medida
32     digitalWrite(pinoLedVermelho, quantidadeLeds >= 1);
33     digitalWrite(pinoLedAmarelo, quantidadeLeds >= 2);
34     digitalWrite(pinoLedVerde, quantidadeLeds >= 3);
35
36     delay(100); // Pequeno atraso para evitar flutuações rápidas
37 }
```

# One more challenge

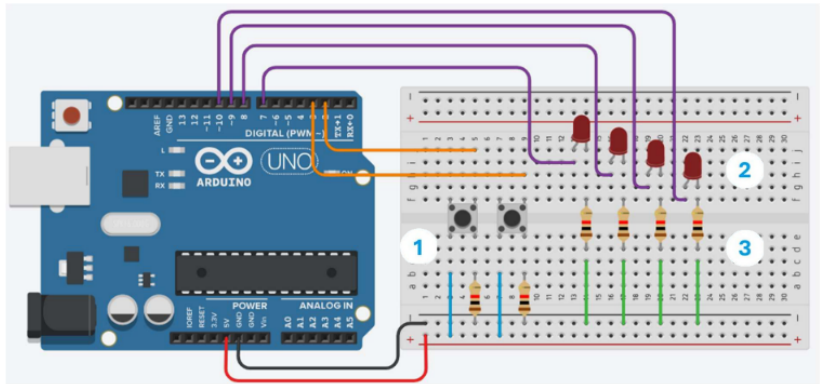


1 2 x Botões

2 1 x LED

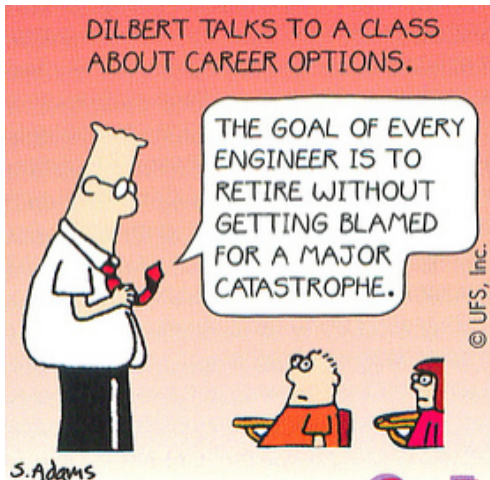
3 3 x Resistência

# ...And one more for good luck



# Thank you for coming

And always remember:

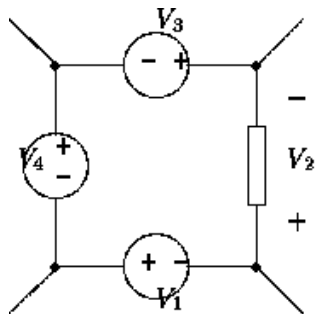


# Electronic Fundamentals (Revisions)

- ❖ Ohm's Law
- ❖ Kirchhoff's Voltage Law (KVL)
- ❖ Kirchhoff's Current Law (KCL)



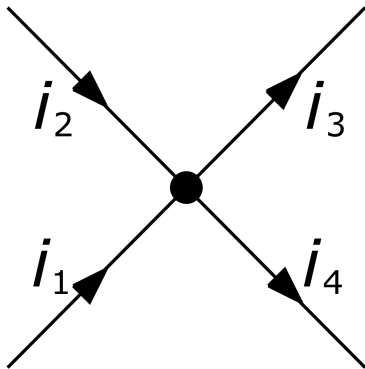
# Kirchoff's Current Law



$$V_1 + V_2 + V_3 + V_4 = 0$$

“In loop there can be no residual Voltage”

# Kirchoff's Voltage Law



$$i_1 + i_2 = i_3 + i_4$$

“In a node there can be no residual Current”

# Common C datatypes

**bool**

*/\* A binary value -> stored in a byte\*/*

**int**

*/\* A signed integer value -> stored in 4 bytes\*/*

**unsigned int**

*/\* A signed integer value -> stored in 4 bytes\*/*

**long int**

*/\*A signed integer value -> stored in 8 bytes\*/*

**float**

*/\* A decimal value -> stored in 4 bytes\*/*

**double**

*/\*A decimal values -> stored in 8 bytes\*/*

and many, many more....

# Functions

```
int add (int arg0, int arg1){  
    int temp = 0;  
    temp = arg0 + arg1;  
    return temp  
}
```

# Applying C to arduino

```
void setup(){  
    /*code here runs once, at the start of our routine*/  
}  
  
void loop(){  
    /*code here will run until the board is reset!*/  
}
```

## Digital GPIO

```
digitalRead(PIN);  
digitalWrite(PIN, value);
```

# Analog I/O - Code example

## Analog GPIO

```
analogRead(PIN);  
analogWrite(PIN, value);
```

## I<sup>2</sup>C

```
#include <Wire.h>
```

```
Wire.begin(SDA_PIN, SCL_PIN);
```

```
Wire.write(data);
```

## UART

```
HardwareSerial mySerial(2); // Use UART2
```

```
mySerial.begin(BAUD_RATE, SERIAL_8N1, RXD_PIN, TXD_PIN);
```