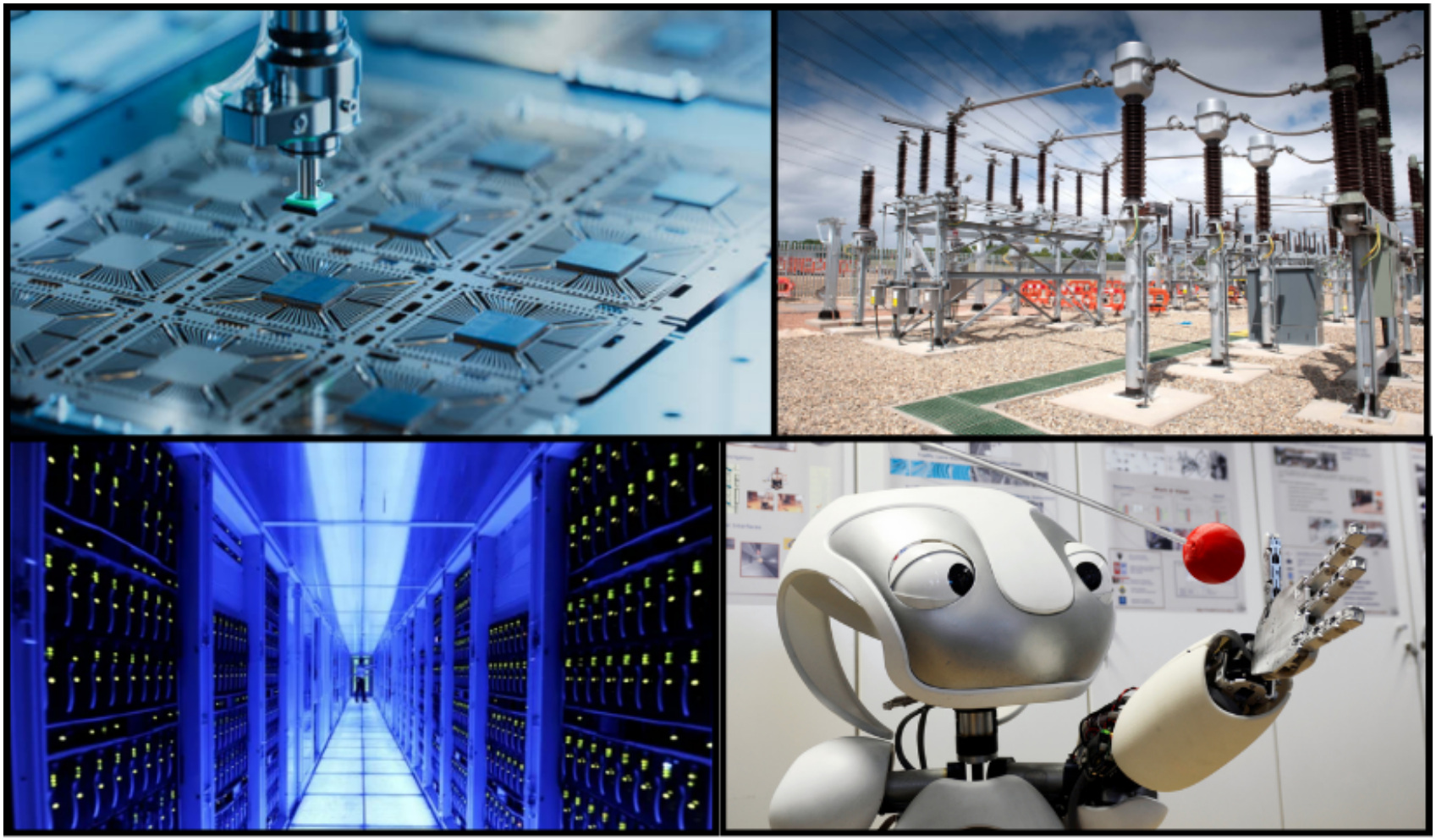
DEEC
DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Arduino 101

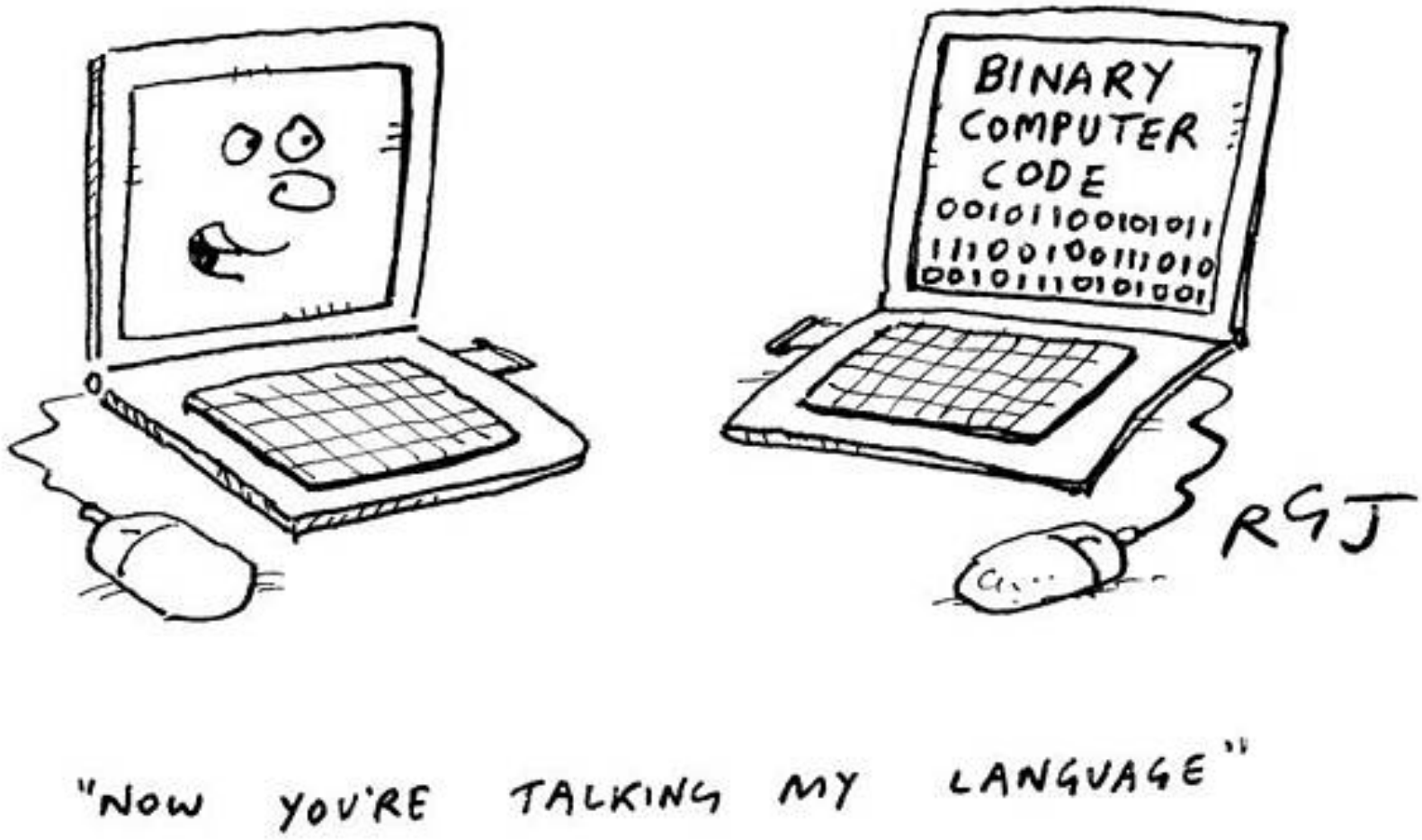
A sprint presentation on MCUs and I/O devices

DEEC-IST | HackerSchool | NEEC | IEEE-SB

Why are we where?



How do computers communicate?



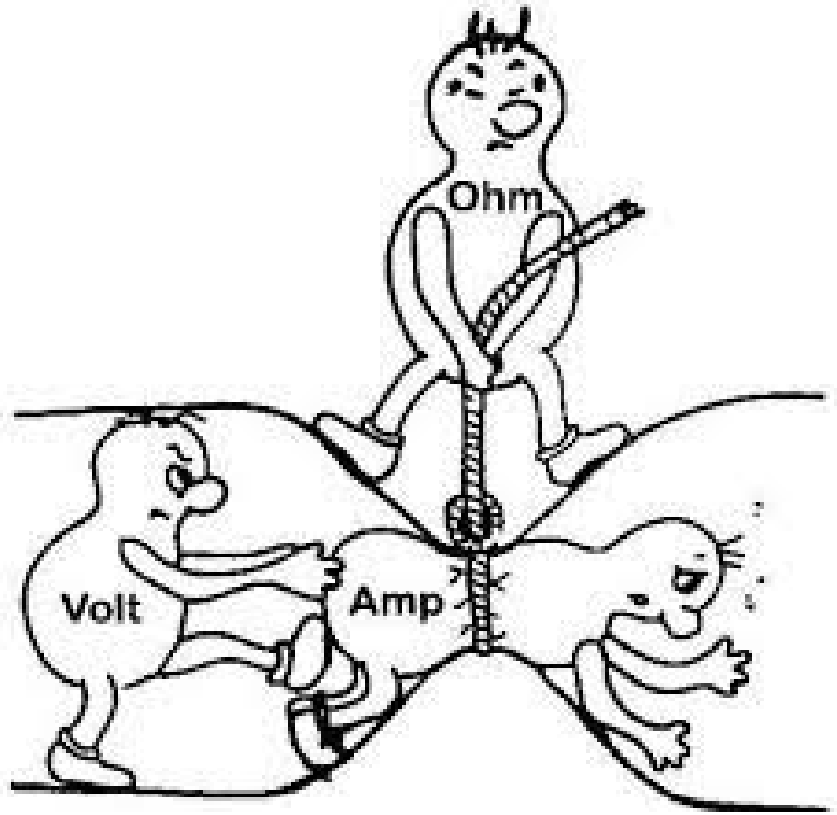
How do computers communicate?

- **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

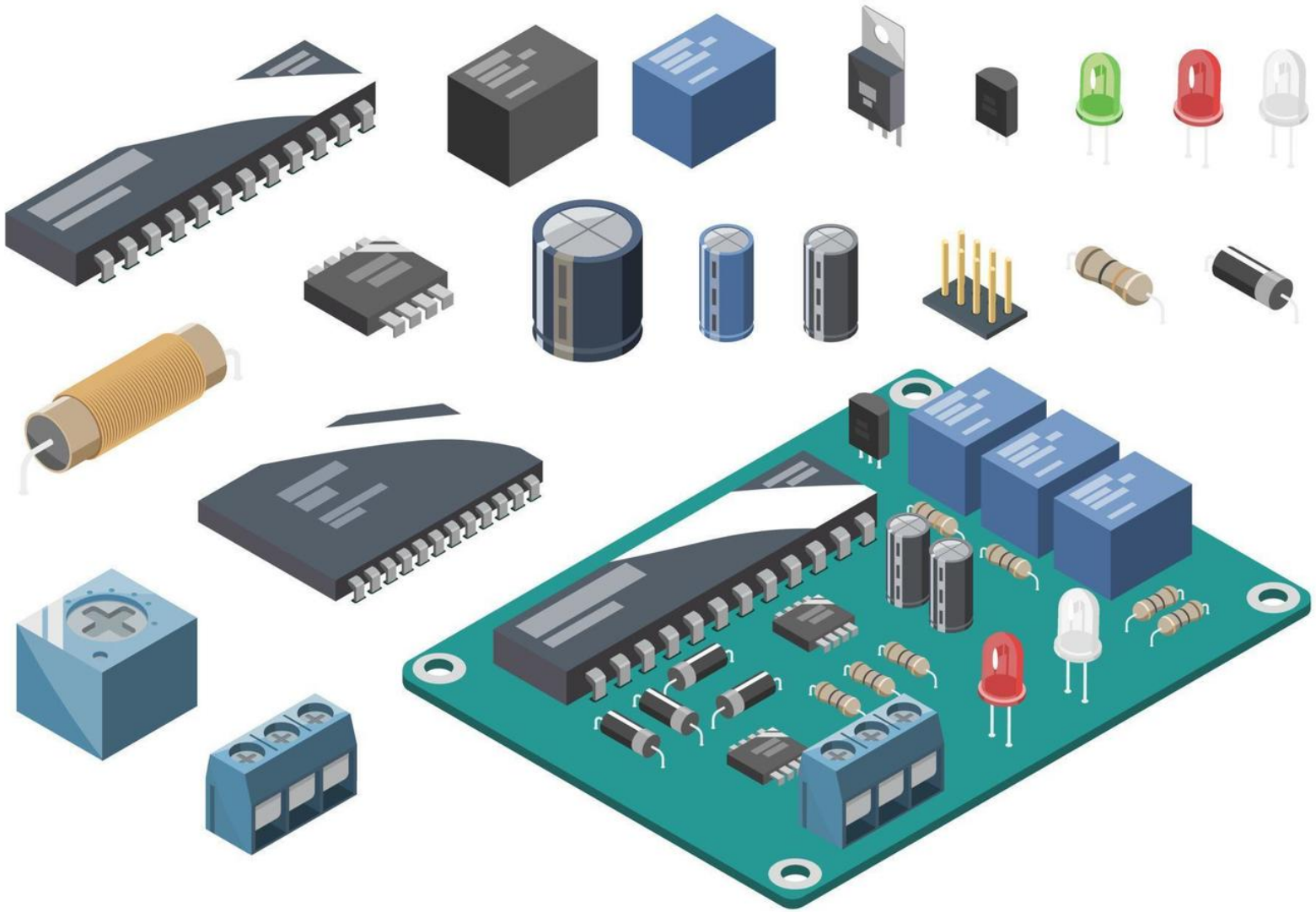
"Oh, what the hell, I'll add another zero."

Ohm's Law

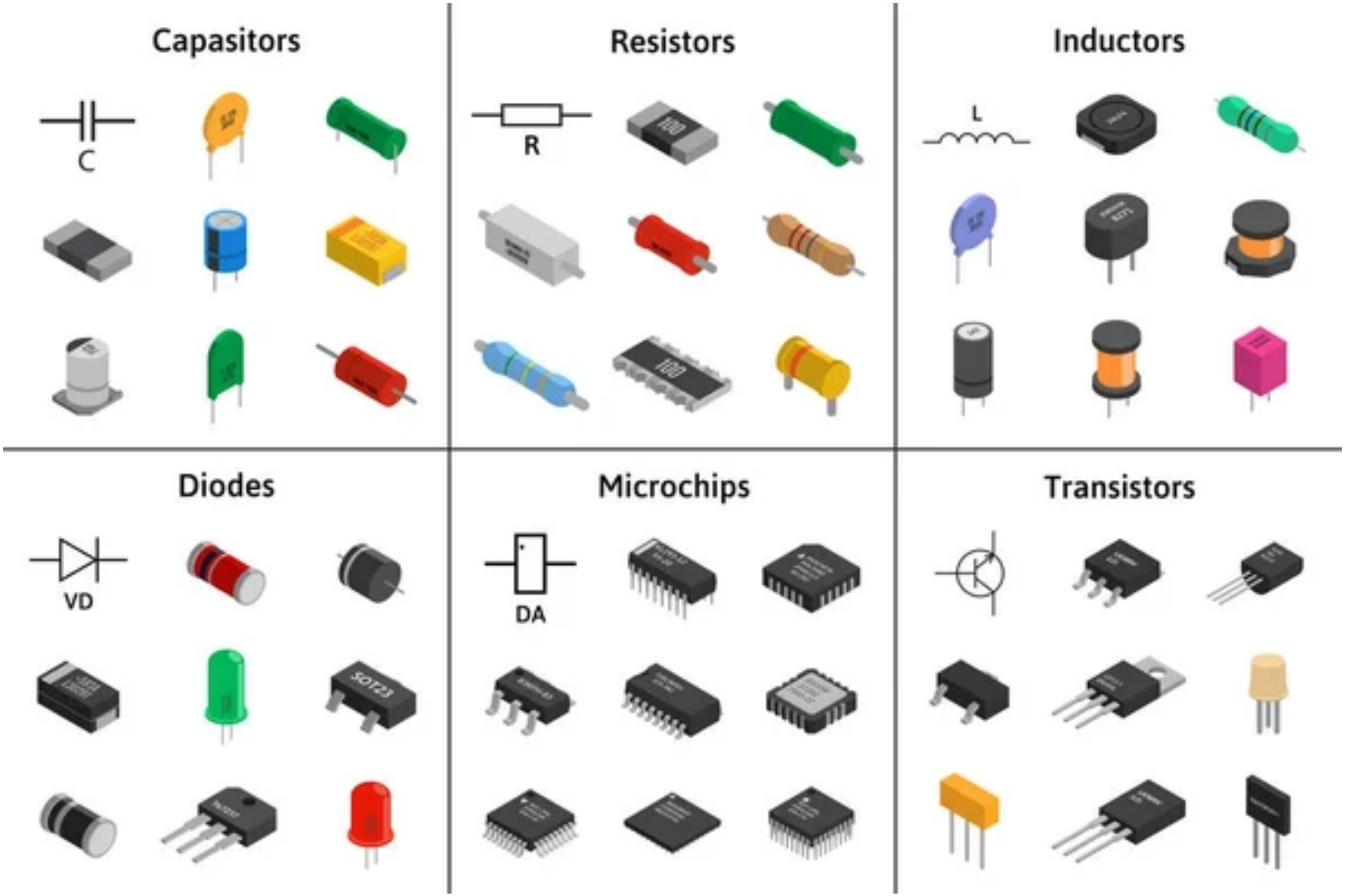
We are Electrical and Computer Engineers, not Computer Scientists! We have to understand the underlying hardware too, lets start with the basics!



Simple electronic components



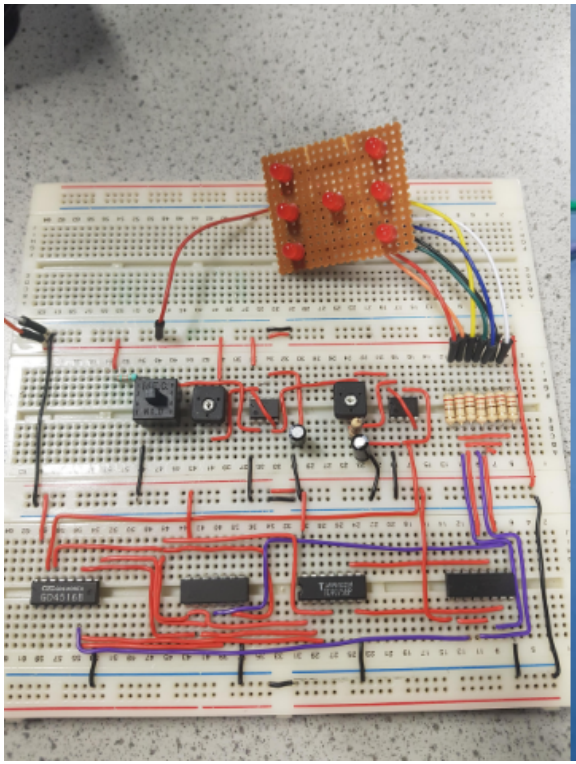
Simple electronic components

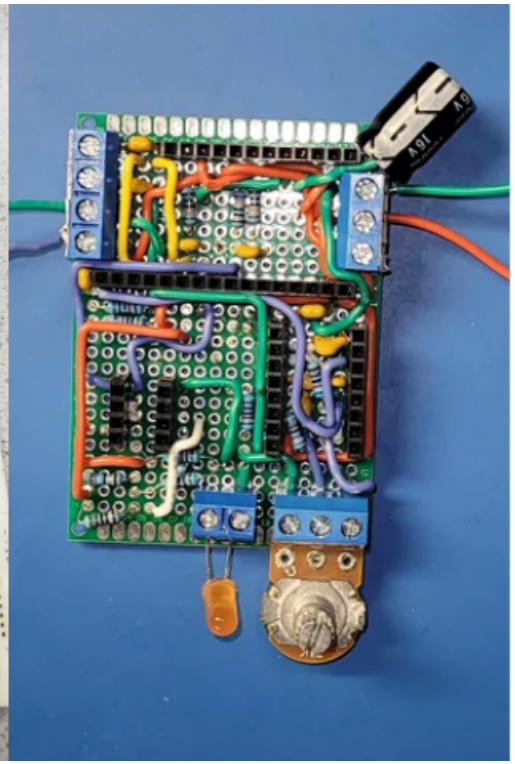


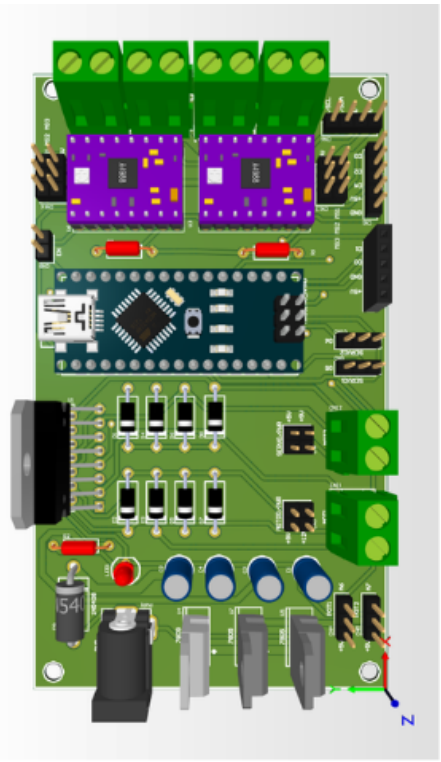
But how can I make the connections?

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.

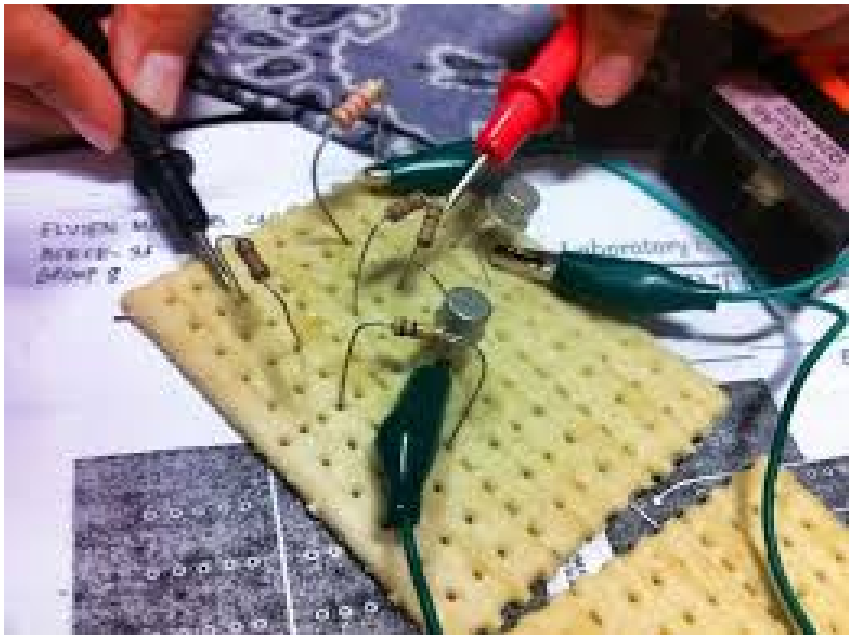






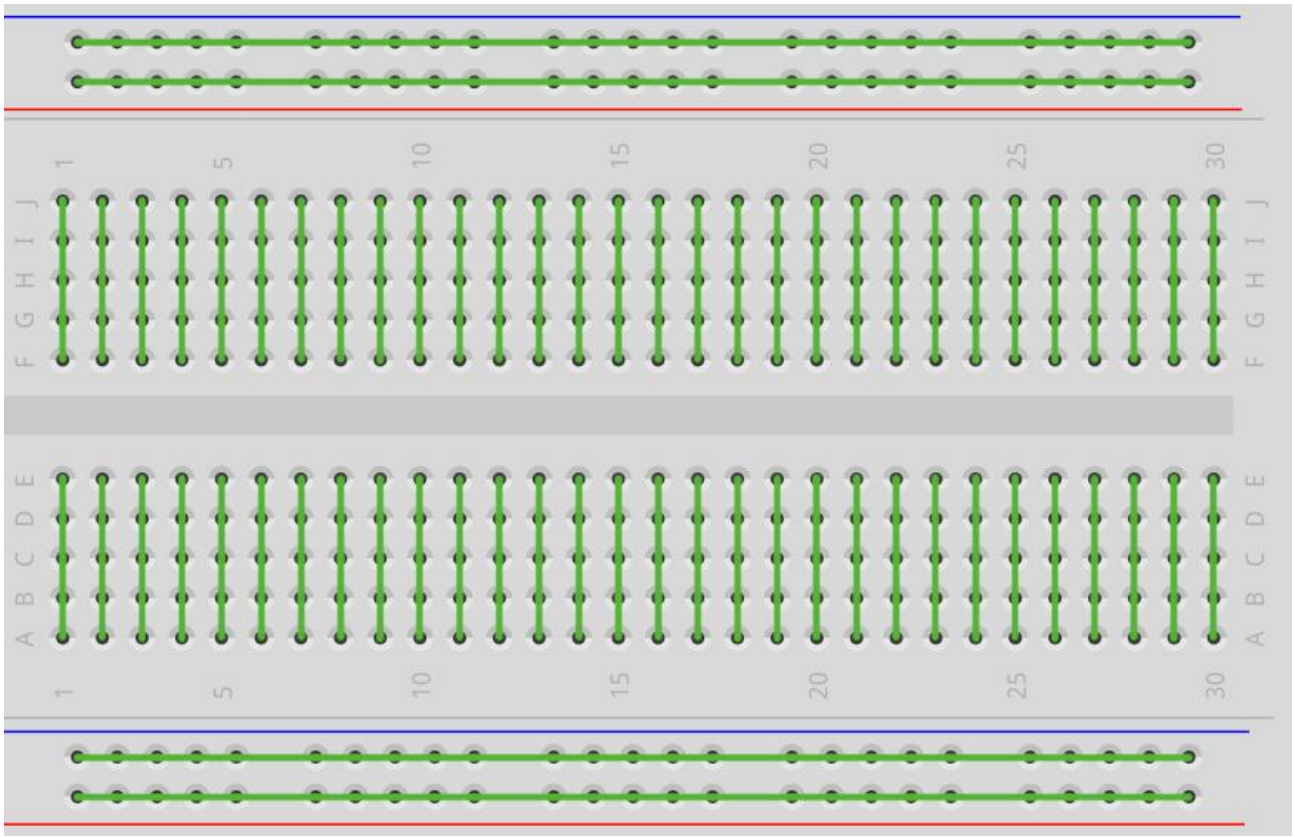
But is soldering viable for **prototyping**?

Breadboard?



Use a breadboard they say. . . .

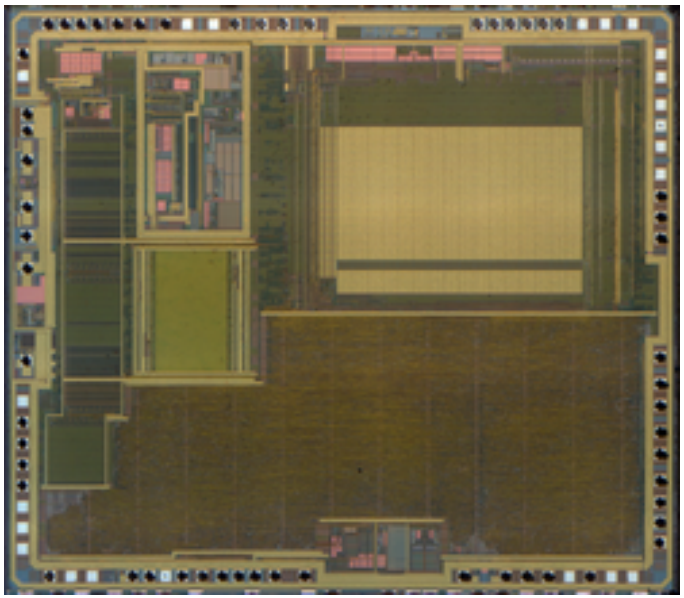
Breadboard.



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**



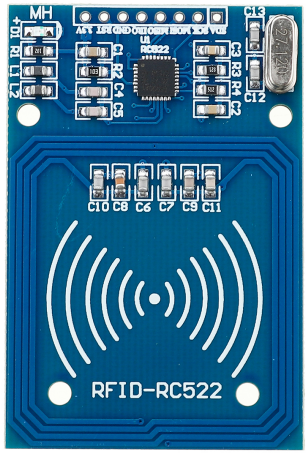
What are Sensors and Actuators?

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

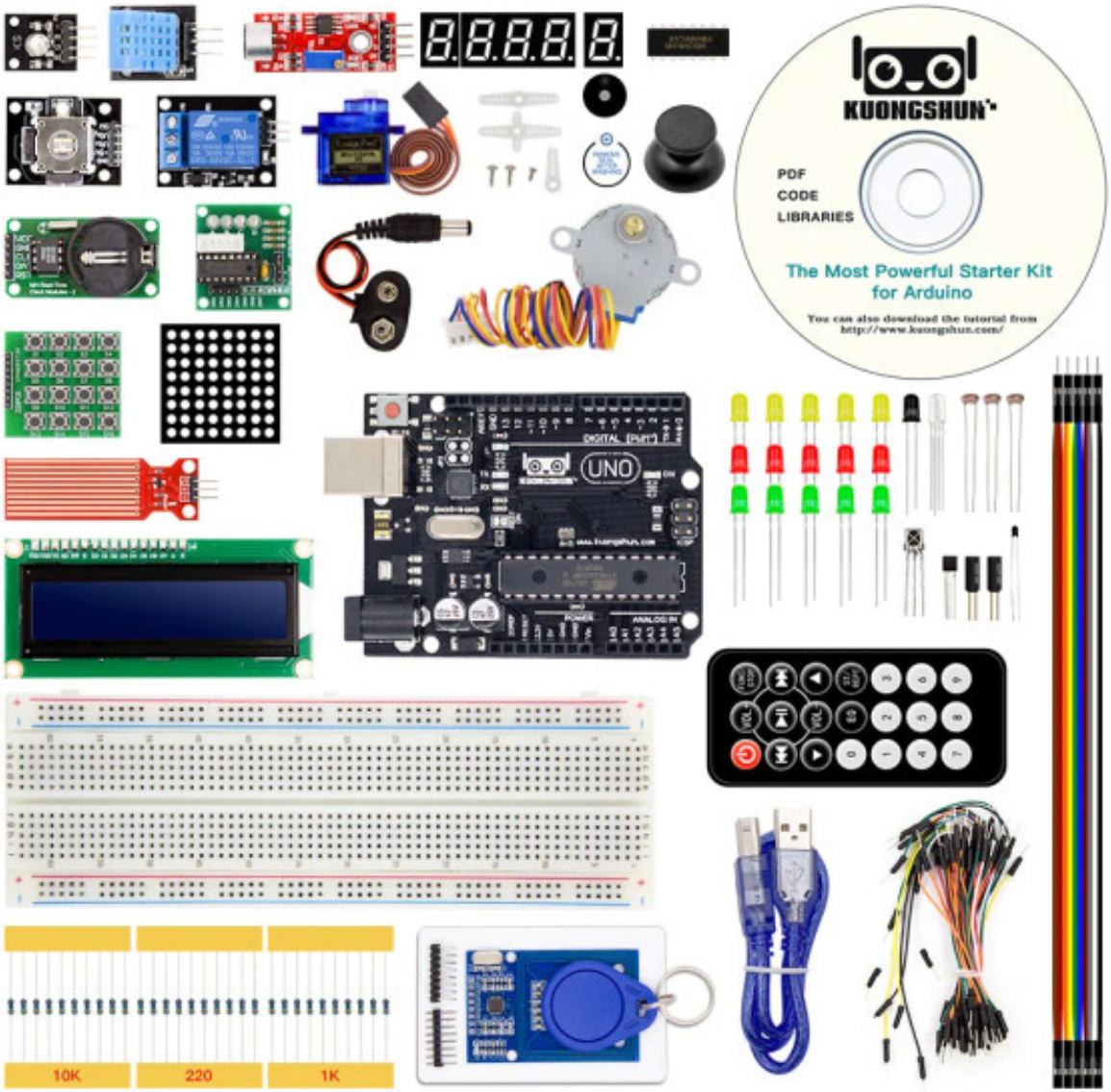
What are Sensors and Actuators?

These devices can communicate with the MCU through various protocols:

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



Some of the sensors you have. . . .



Arduino UNO

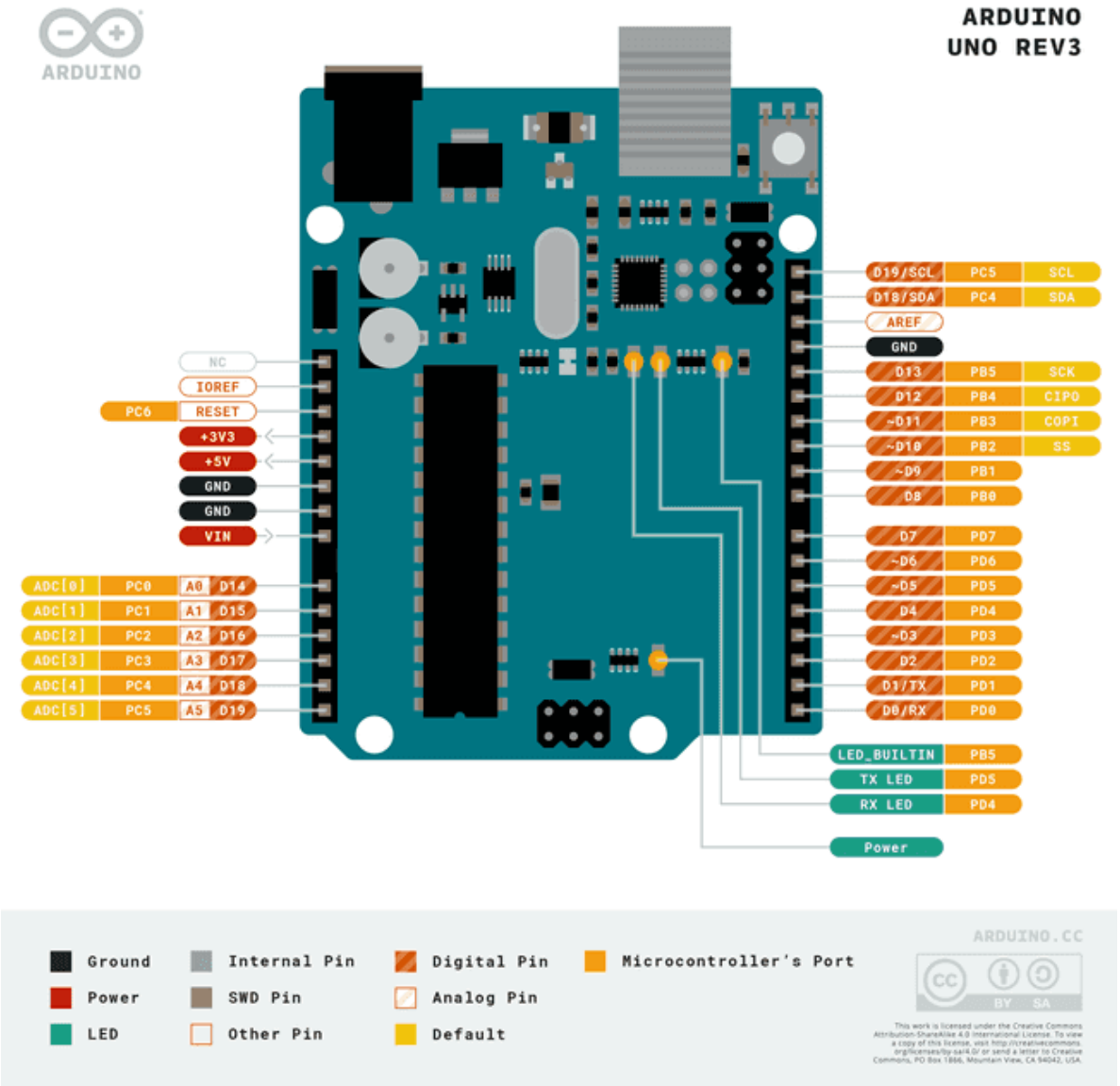
ATmega328P

- 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

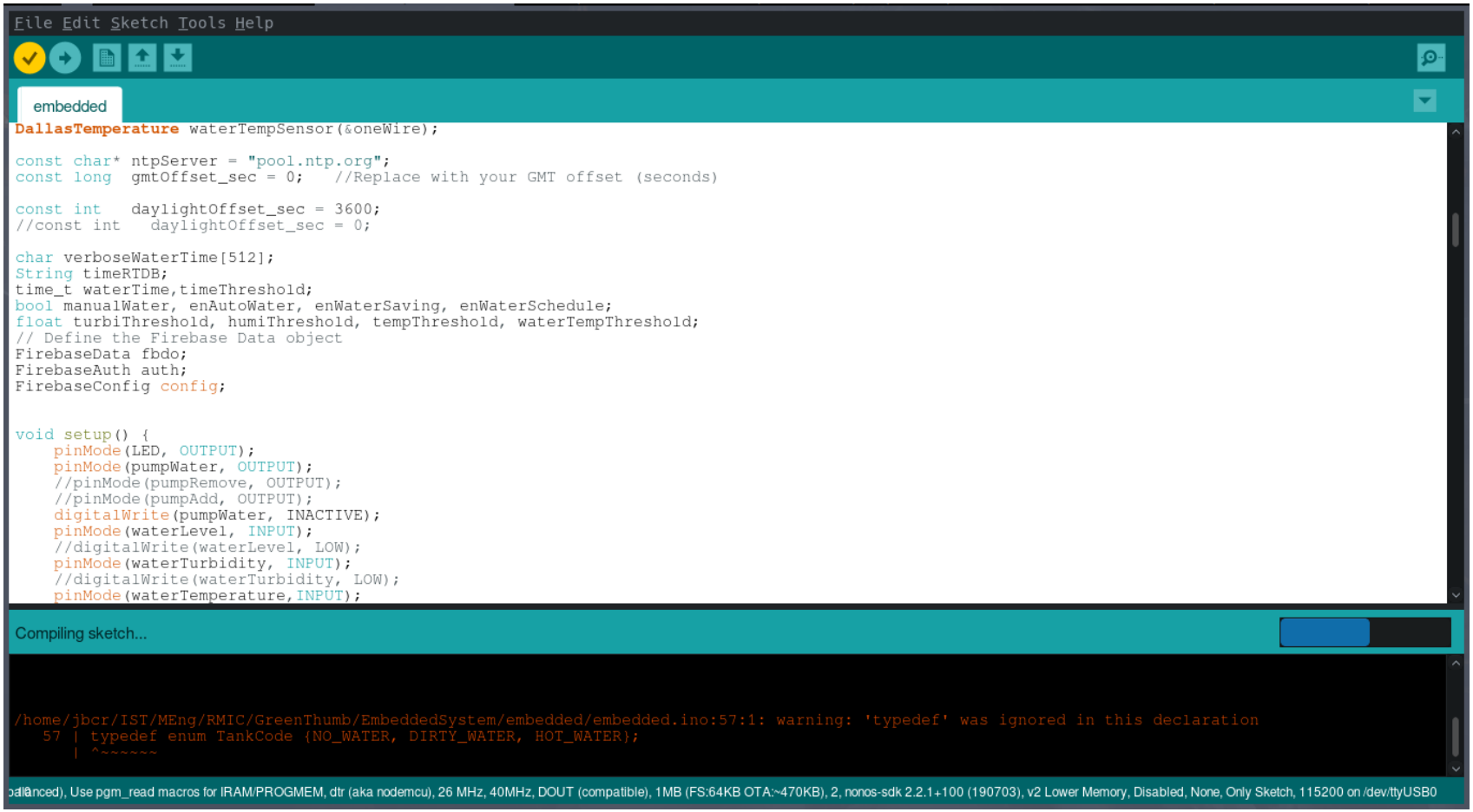
- Board that integrates a ATmega328P chip
- Adds peripherals such as
  - Quartz Oscillator for clock signal
  - USB Controller
  - Interfaces for Power signals

Arduino UNO Pinout



Setup and Microcontroller programming

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



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?

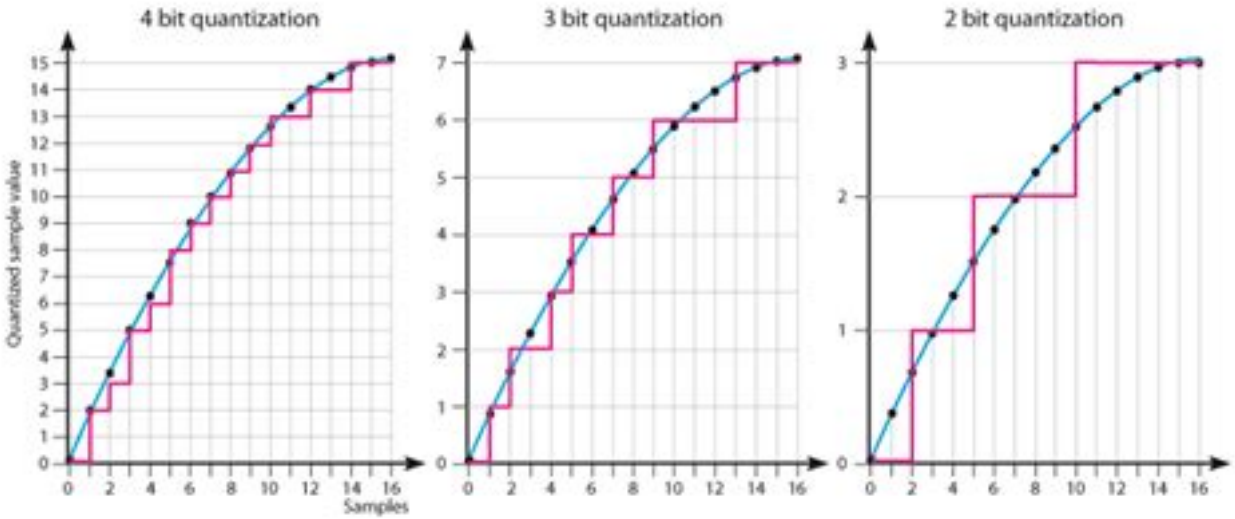
A bit represents one of two possible states

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

Therefore with 10-bits:

Analog I/O - How does it work?

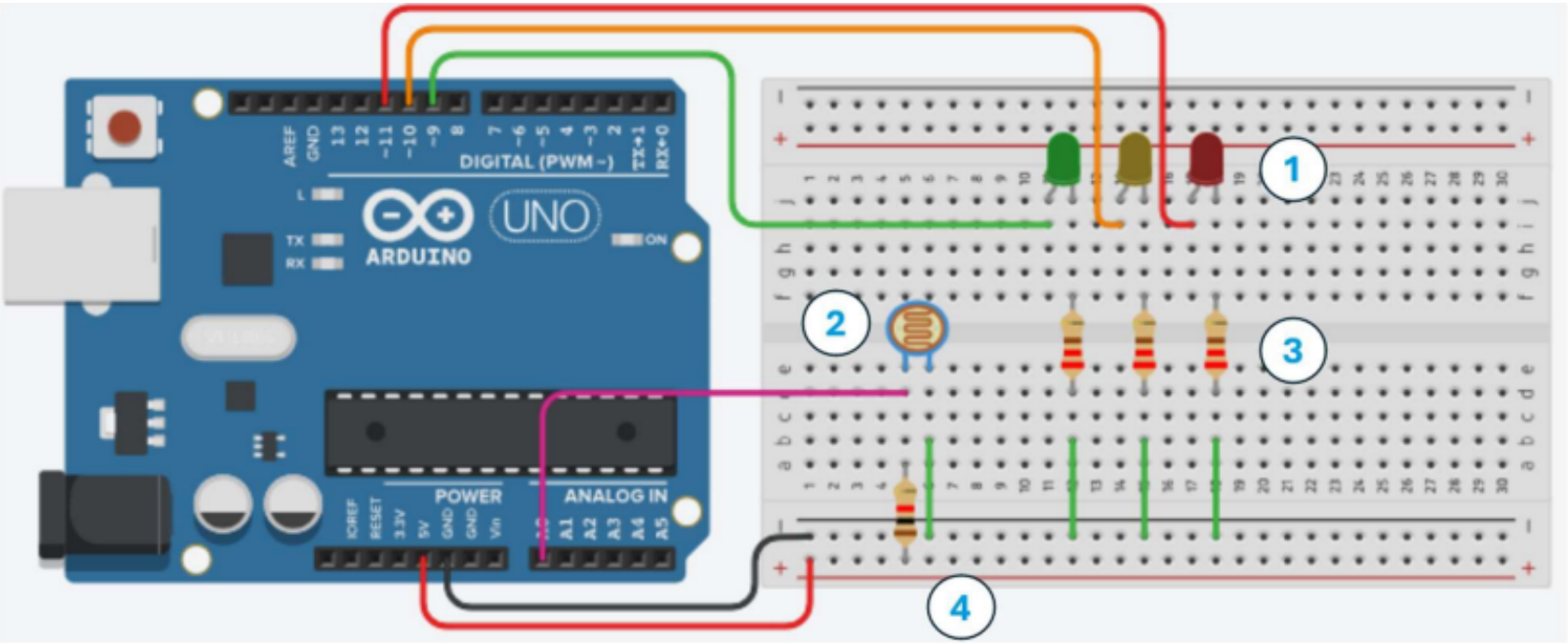
So our ADC acquired a **continous** signal and converted it into a **discrete** one! This is called **quantization**!



Let’s build something! - Concept

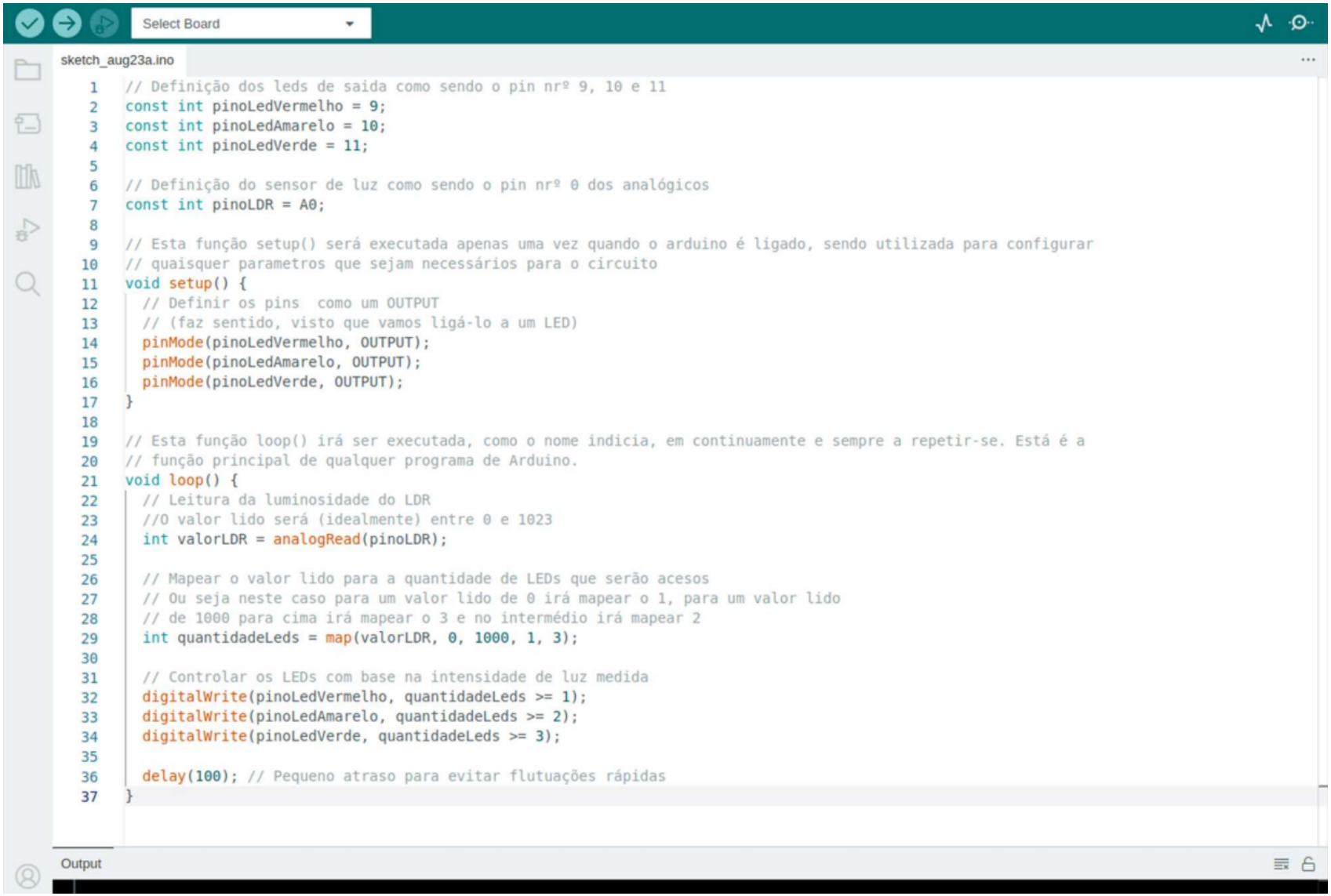
Let’s build a **very** simple **luminosity detection system**

Let’s build something! - Schematic



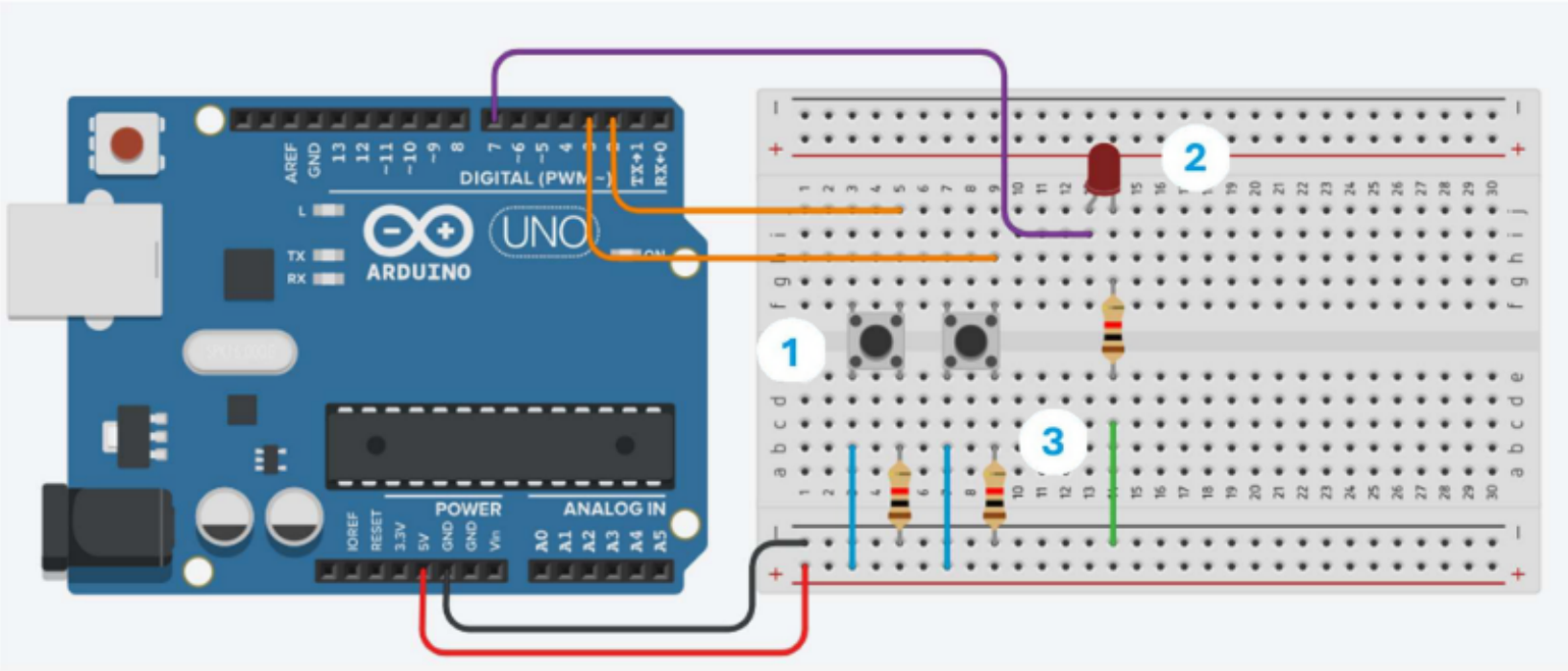
1 3 x LED’s 2 1 x LDR 3 3 x Resistência 220 Ω 4 1 x Resistência 10kΩ

Let’s build something! - Code




```ino
// Definição dos leds de saida como sendo o pin nrº 9, 10 e 11
const int pinoLedVermelho = 9;
const int pinoLedAmarelo = 10;
const int pinoLedVerde = 11;

// Definição do sensor de luz como sendo o pin nrº 0 dos analógicos
const int pinoLDR = A0;

// Esta função setup() será executada apenas uma vez quando o arduino é ligado, sendo utilizada para configurar
// quaisquer parametros que sejam necessários para o circuito
void setup() {
  // Definir os pins  como um OUTPUT
  // (faz sentido, visto que vamos ligá-lo a um LED)
  pinMode(pinoLedVermelho, OUTPUT);
  pinMode(pinoLedAmarelo, OUTPUT);
  pinMode(pinoLedVerde, OUTPUT);
}

// Esta função loop() irá ser executada, como o nome indicia, em continuamente e sempre a repetir-se. Está é a
// função principal de qualquer programa de Arduino.
void loop() {
  // Leitura da luminosidade do LDR
  //O valor lido será (idealmente) entre 0 e 1023
  int valorLDR = analogRead(pinoLDR);

  // Mapear o valor lido para a quantidade de LEDs que serão acesos
  // Ou seja neste caso para um valor lido de 0 irá mapear o 1, para um valor lido
  // de 1000 para cima irá mapear o 3 e no intermédio irá mapear 2
  int quantidadeLeds = map(valorLDR, 0, 1000, 1, 3);

  // Controlar os LEDs com base na intensidade de luz medida
  digitalWrite(pinoLedVermelho, quantidadeLeds >= 1);
  digitalWrite(pinoLedAmarelo, quantidadeLeds >= 2);
  digitalWrite(pinoLedVerde, quantidadeLeds >= 3);

  delay(100); // Pequeno atraso para evitar flutuações rápidas
}
```

One more challenge

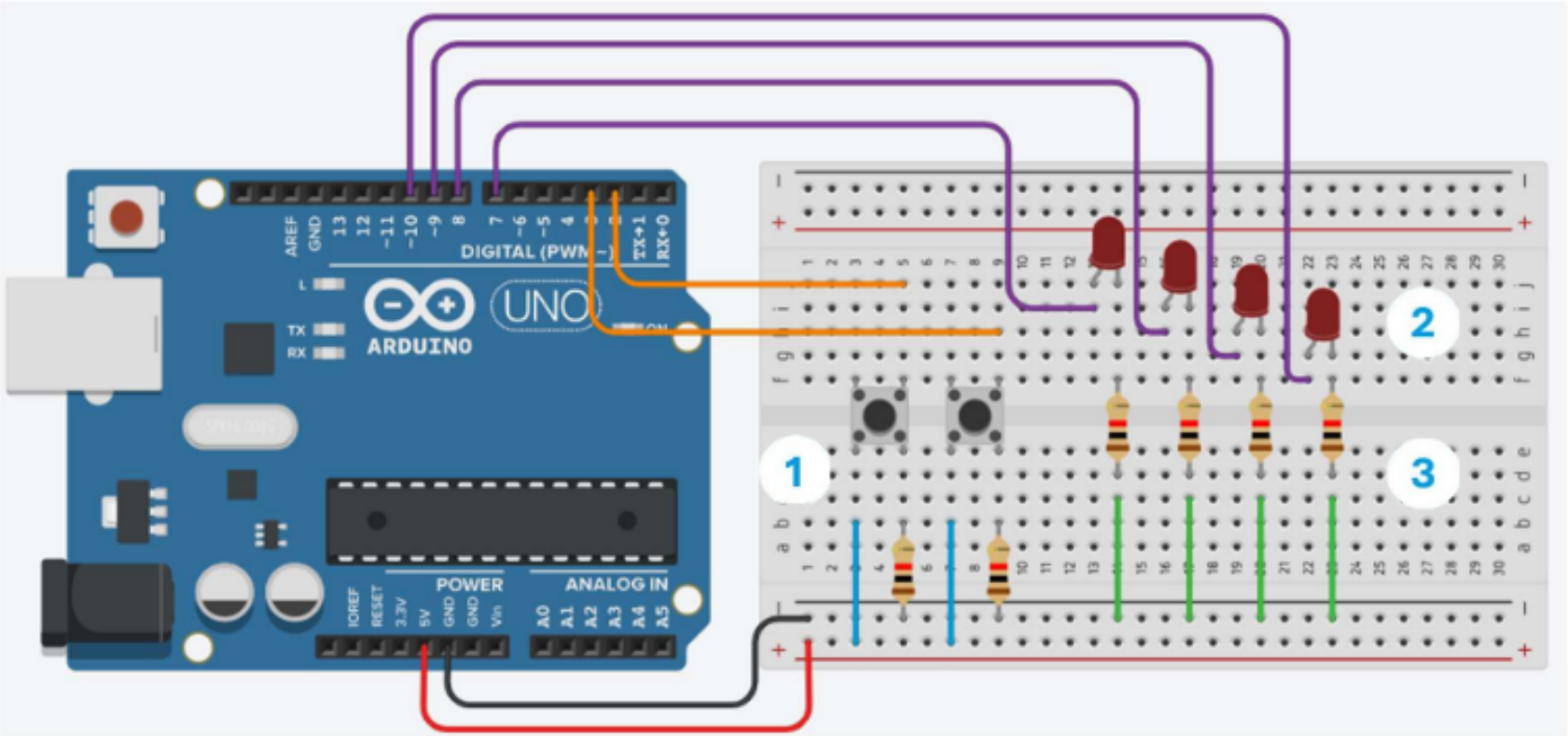


1 2 x Botões

2 1 x LED

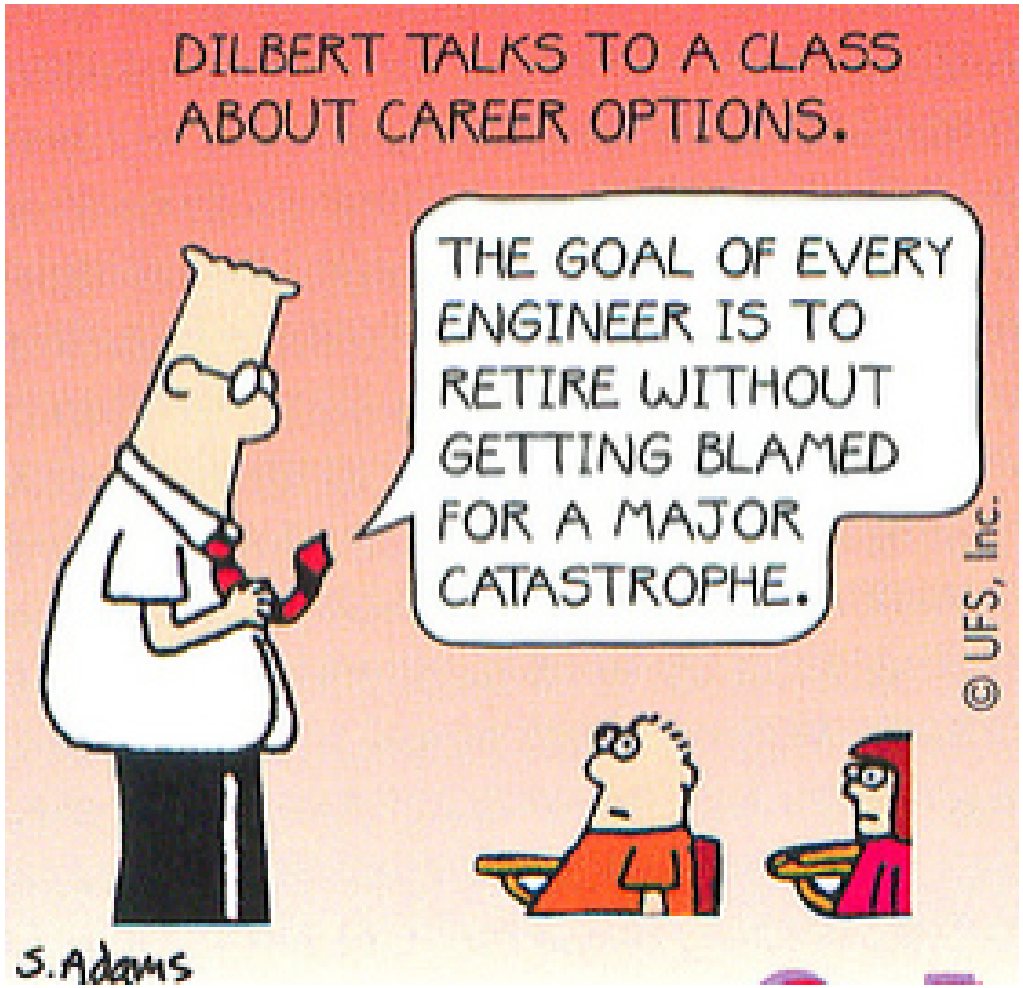
3 3 x Resistência

. . . And one more for good luck



Thank you for coming

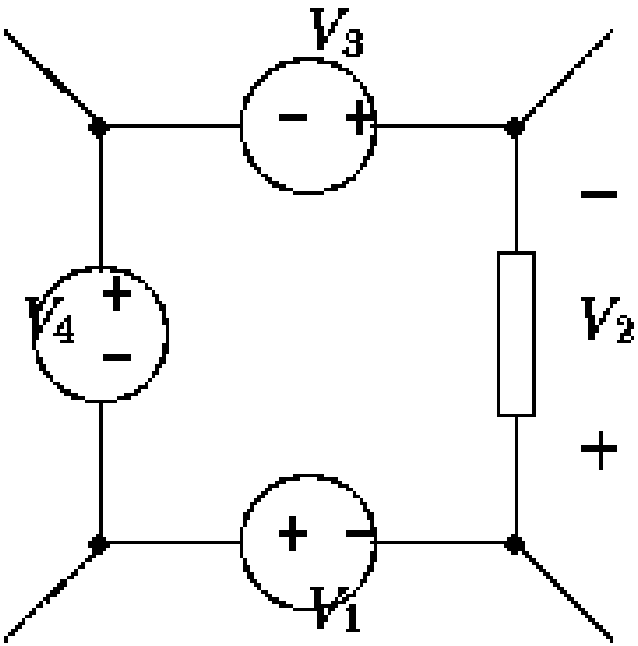
And always remember:



Eletronic Fundamentals (Revisions)

- Ohm's Law
- Kirchoff's Voltage Law (KVL)
- Kirchoff's Current Law (KCL)

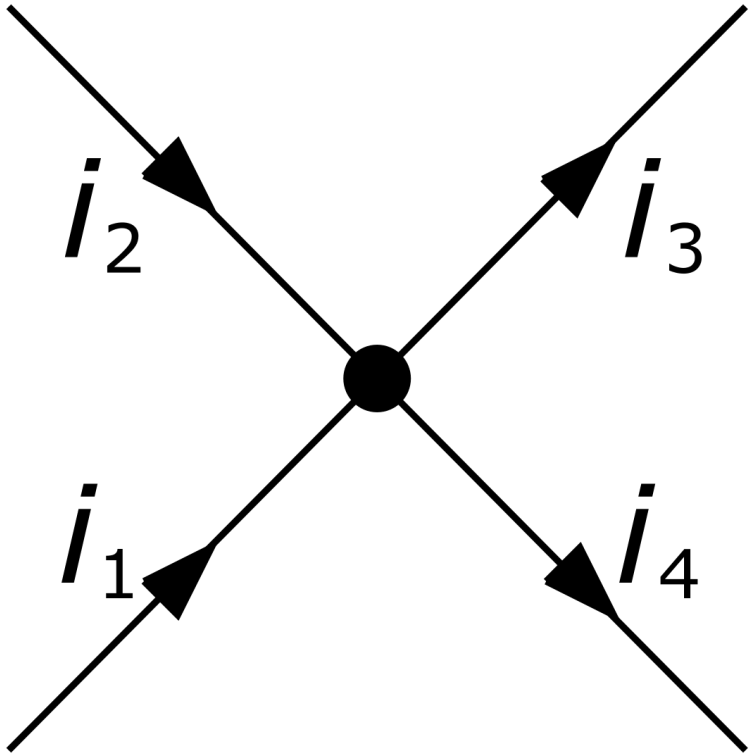
Kirchoff's Current Law



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

"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 I/O - Code example

Digital GPIO

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

Analog I/O - Code example

Analog GPIO

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

More I/O

I²C

```
#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);
```