# Introducción a Arduino


Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Cristian González García
gonzalezcristian@uniovi.es


Basado en el material original de Jordán Pascual Espada

v 1.3.1 Septiembre 2021

# Componentes de un robot

# Componentes de un robot

- Algunas de las partes básicas
  - **Sensores**
    - Perciben información
  - **Actuadores**
    - Emiten respuestas (Señal, movimiento motor) locomoción, manipulación
  - **Controladores**
    - Computación
    - Control de sensores y actuadores
  - **Mecanismos y estructura**

- Pueden tener «infinitas» partes
- Pueden tener otras partes
  - Armazones, armas, protecciones, etc.

# Comportamientos

# Comportamientos I

- Comportamiento básico de los Robots
  - **Percepción**
    - ¿Cómo percibe? **Sensores**
  - **Procesamiento** de la lógica de negocio
    - ¿Cómo planifica en que estado se encuentra y que va a realizar?
    - Secuencia fija, **algoritmos**, **inteligencia artificial**, aprendizaje, etc…
  - **Acción**
    - ¿Cómo actúa? **Actuadores**

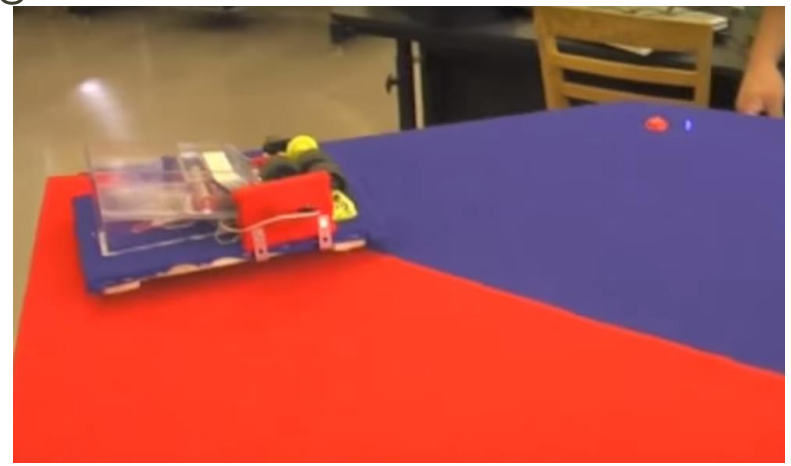
- En la mayoría de casos estos comportamientos están relacionados
  - **Desencadenadores**

# Comportamientos II

- Si no hay pared delante continúa por el camino

# Comportamientos III

- Reparte cartas entre los jugadores



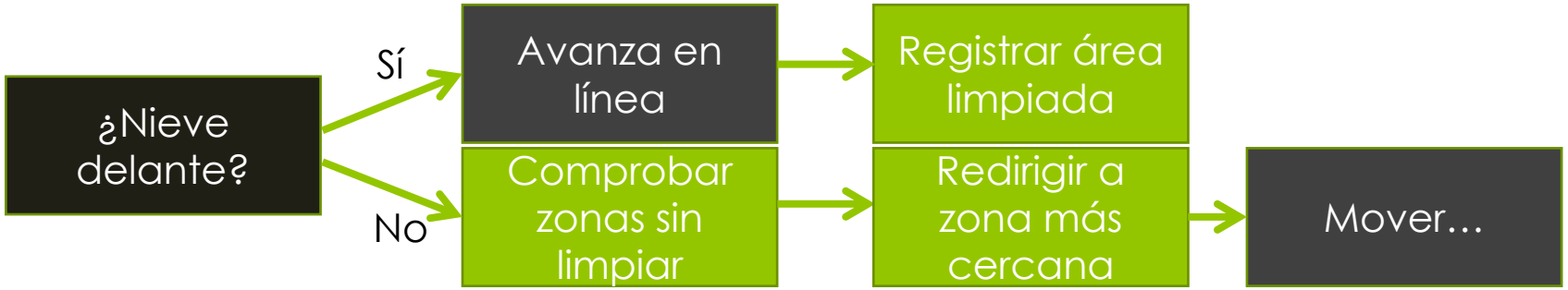
https://www.youtube.com/watch?v=3BGe4OZIAKQ#t=1m30s

# Comportamientos IV

- https://www.youtube.com/watch?v=sKPfwpGLXTg
- Percepción: ¿Qué puede percibir?
- Lógica de negocio: ¿Cómo planifica la tarea a realizar?
- Acción: ¿Qué acciones puede realizar?

# Comportamientos V

- **Percepción**: ¿Qué puede percibir?
  - Detección de nieve (sensor de distancia, color y/o presión)
  - Posición GPS, balizas, etc.
- **Lógica**: ¿Cómo planifica la tarea a realizar?



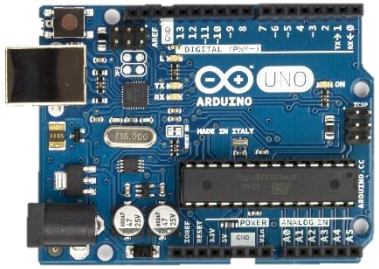

- **Acción**: ¿Qué acciones puede realizar?
  - Avanzar
  - Retroceder
  - Girar derecha / izquierda
  - Dar media vuelta
  - Subir/Bajar la pala
  - …

# Diseño y programación de robots

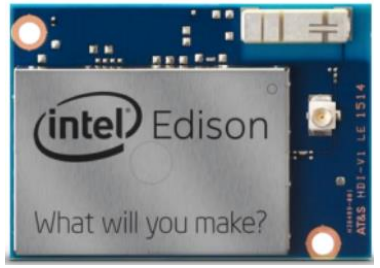
# Diseño y programación de robot

- Existen **muchas alternativas para construir y programar robots**
  - Kits completos (computación, electrónica, mecanismos y estructuras)
    - Lego Mindstorms, Makeblock, Fischertechnik,  Vex, Stem, …
  - Computación y control de componentes electrónicos
    - Arduino, Raspberry Pi, Intel Edison, Beagle Bone, Launchpad, Odroid-U3, …
  - Mecanismos y estructuras
    - Comprar componentes estándar, diseñarlos y fabricarlos
      - Metal, Poliácido Láctico (PLA), Acrilonitrilo Butadieno Estireno (ABS), etc.

https://commons.wikimedia.org/wiki/File:Lego_Mindstorms_Nxt-FLL.jpg



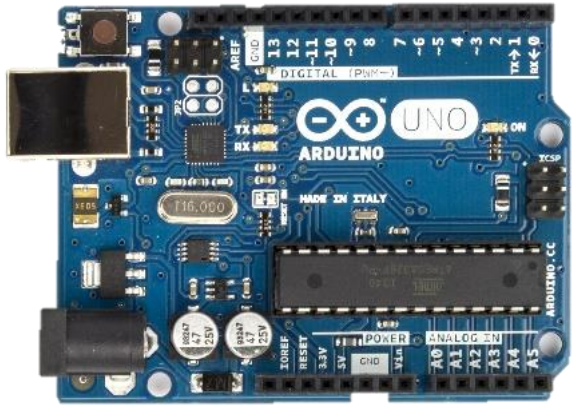

https://commons.wikimedia.org/wiki/File:Arduino_Uno_006.jpg




https://en.wikipedia.org/wiki/Intel_Edison#/media/File:Intel-Edison2.png

# Arduino

# Arduino I




https://commons.wikimedia.org/wiki/File:Arduino_Uno_006.jpg

- https://www.arduino.cc/
- Plataforma de **hardware libre**
  - Diagramas y especificaciones abiertas al público
- Apareció en el año **2005**
- Desarrollada por Massimo Banzi, David Mellis, David Cuartielles, Tom Igoe y Gianluca Martino
- Solo **se fabricaba en Italia**
- Placa con un **microcontrolador programable**
- Ofrece un **entorno de desarrollo**
  - Simple
  - Mantenido y actualizado cada poco

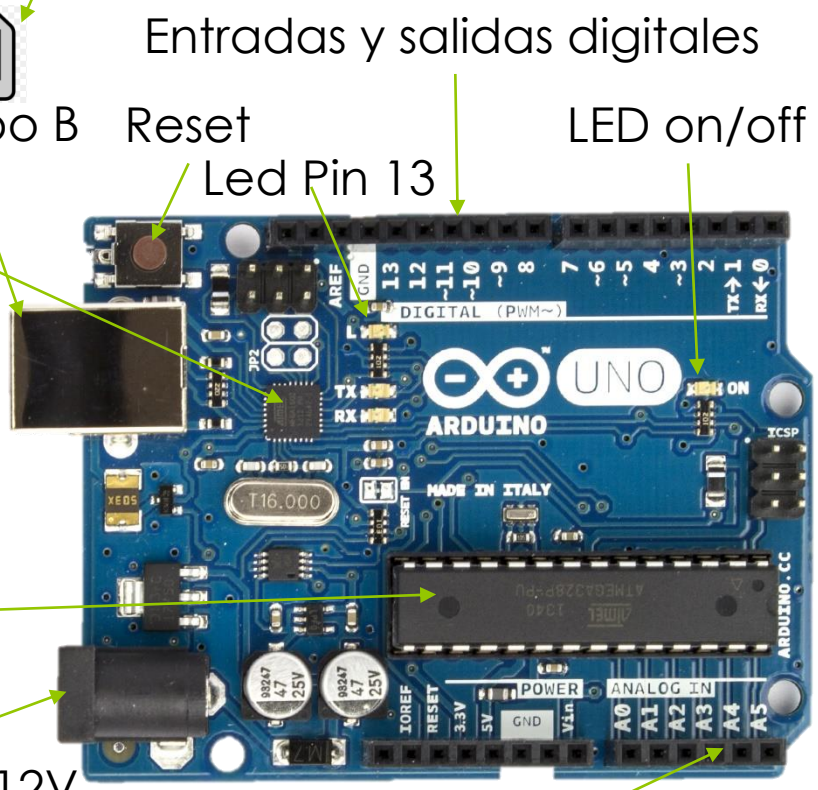
# Arduino o Genuino

- **Arduino LLC** fue el original en 2005
- Debido a **problemas con un socio** crearon Genuino en 2015
  - Montó una empresa (Smart Projects SRL) para fabricar placas Arduino y registró el nombre Arduino en Italia en secreto
    - Gianluca Martino registro la marca Arduino SRL en más de 40 países
  - El nombre Genuino de Arduino LLC era en todo el mundo y Arduino solo en EEUU
- Arduino LLC se podía fabricar libremente por cualquier empresa
- Diferencias
  - Arduino UNO vs Genuino UNO: lugar donde se fabrica
  - Nuevas placas: el equipo de desarrollo de estas
- http://playground.blogautore.repubblica.it/2015/02/11/la-guerra-per-arduino-la-perla-hi-tech-italiana-nel-caos/
- El 28 de julio de 2017 se vuelven a juntar las dos empresas
  - https://blog.arduino.cc/2017/07/28/a-new-era-for-arduino-begins-today/
  - Tras varias compras, Arduino LLC consiguió fusionar todo

https://es.wikipedia.org/wiki/Universal_Serial_Bus#/media/Archivo:USB_2.0_and_3.0_connectors.svg

# Arduino – Principales partes de la placa

- ATmega16U2
  - Se encarga del USB/Serial
  - Familia lógica CMOS
    - Transistores MOSFET
  - Arquitectura RISC
  - Se puede actualizar/borrar/etc.
- Atmega328p
  - Principal
  - Se encarga del código/programa
  - Arquitectura RISC




https://commons.wikimedia.org/wiki/File:Arduino_Uno_006.jpg

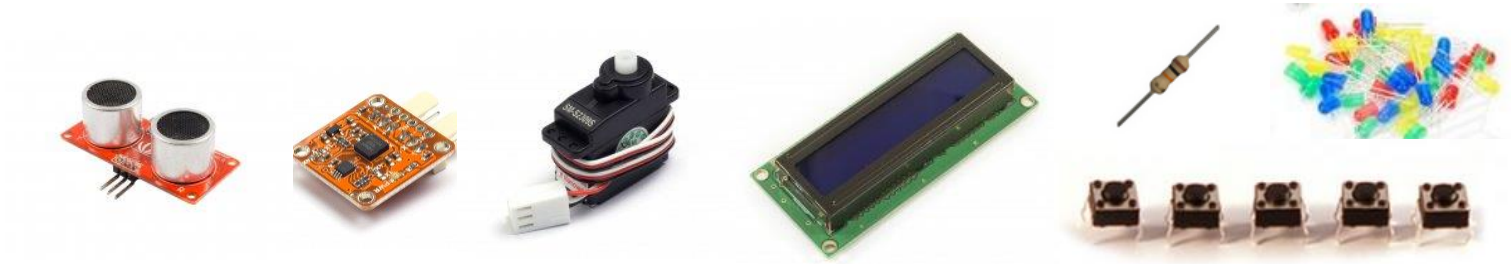
https://www.arduino.cc/en/pmwiki.php?n=Reference/Board

# Programación

- **Entornos de desarrollo**
  - Oficiales (incluido el web): https://www.arduino.cc/en/main/software
    - También en Microsoft Store
  - Terceros: MiniBloq, BitBloq (BQ), **TinkerCAD**, …
  - Investigaciones: Midgar, Paraimpu, etc.
- **Se comunica con el puerto serie USB**
  - Carga el programa en la placa, funciones de entrada/salida
- **Lenguaje de programación (Arduino)**
  - Basado en Wiring (Programado en C/C++)
  - Utiliza un set de funciones de C/C++
  - Todas las funciones estándar de C y C++ deberían de funcionar
    - Estructuras de control, variables,  tipos de datos, funciones, etc.
  - El código se compila utilizando un compilador de C/C++
    - https://arduino.github.io/arduino-cli/sketch-build-process/
  - Muchas bibliotecas para controlar componentes (de forma sencilla)
- **Comunicación de aplicaciones Arduino con otros lenguajes de programación** utilizando el USB
  - Java, C#, Objetive-C, C++, Perl, Ruby, Python, etc.
  - https://playground.arduino.cc/Interfacing/Java/
    - Más en las prácticas…

# Componentes usables

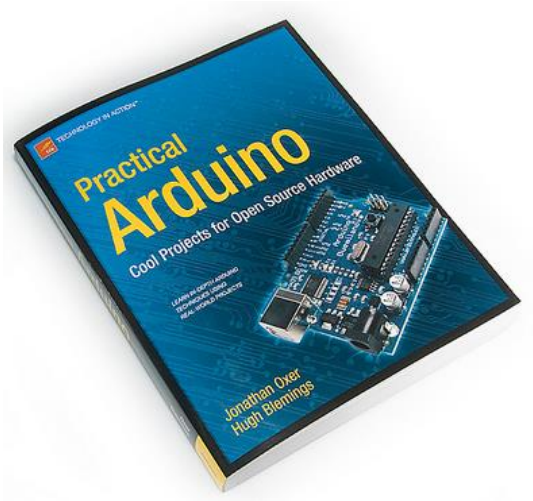
- **Facilita el uso de componentes electrónicos**
  - Sensores, motores, pantallas, resistencias , pulsadores, leds, etc.



- El hardware **provee entradas y salidas digitales, y entradas analógicas**
  - Conectamos los componentes electrónicos
  - Programadas para enviar o recibir datos

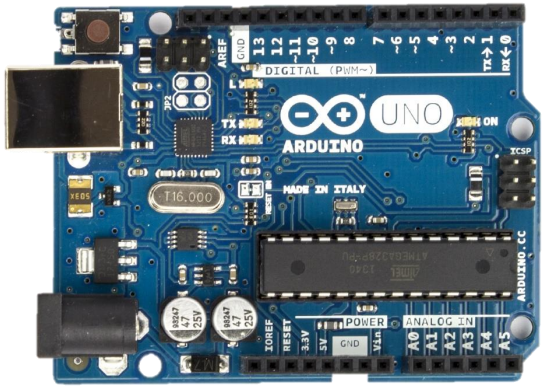

# Uso

- Se usa en **proyectos multidisciplinares**
  - Electrónica, informática, robótica, mecánica...
- Muy **popular para la práctica y el aprendizaje**
- **Gran comunidad de usuarios**
  - Desarrolladores
  - Material de consulta y aprendizaje
  - Eventos y cursos
  - Nuevos productos

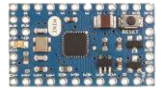


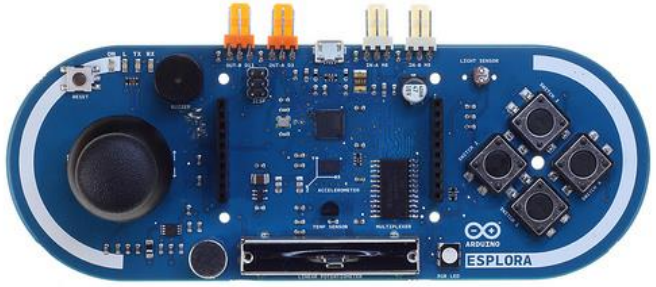
# Hardware

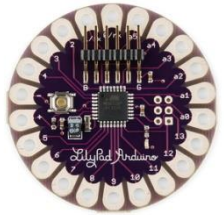
- **Diferentes modelos de placas**
  - Dimensiones (no están a escala)
  - Potencia: procesador, velocidad CPU, memoria
  - Número de entradas y salidas analógicas y digitales
  - Voltajes admitidos: alimentación y salidas
  - Otras características integradas
    - Ej.: Bluetooth, ethernet, botones, etc.

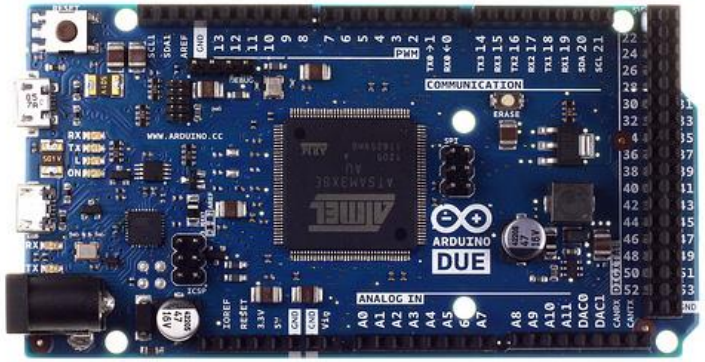



https://commons.wikimedia.org/wiki/File:Arduino_Uno_006.jpg








- https://www.arduino.cc/en/hardware
- https://docs.arduino.cc/retired/other/arduino-older-boards

# Componentes compatibles

- Un **gran número de componentes electrónicos compatibles**
  - Entradas y salidas digitales y entradas analógicas

- Placas de extensión/Escudos (Shields) Arduino y otros elementos propios
  - Permiten no «perder» pines, generalmente
  - https://store.arduino.cc/collections/shields

GSM

Relés

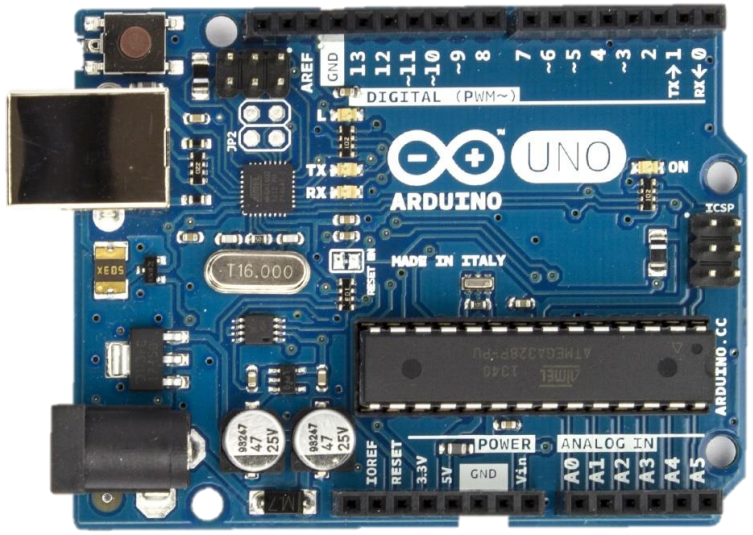
Carga Solar

Driver Motores

  - Ethernet, GPS, Wireless, TFT táctil, etc.

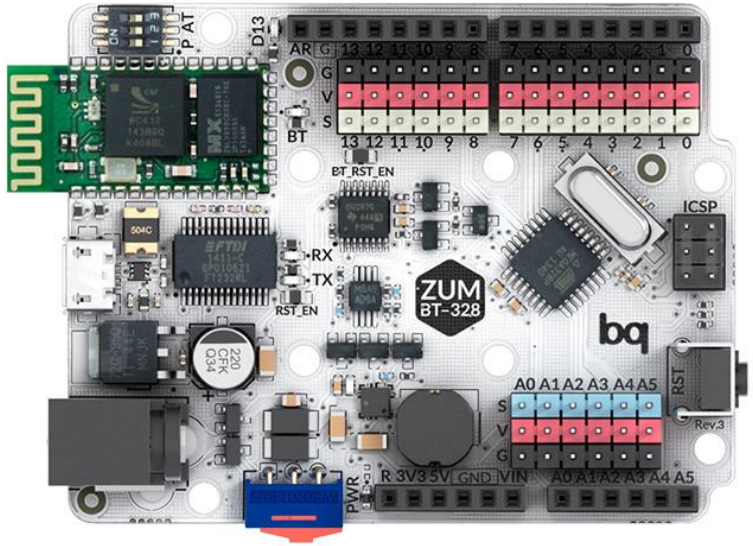
# Modelos que utilizaremos

- **Además** de muchos **componentes electrónicos** diferentes y alguna placa de extensión (**Shield**)




Arduino UNO

https://commons.wikimedia.org/wiki/File:Arduino_Uno_006.jpg

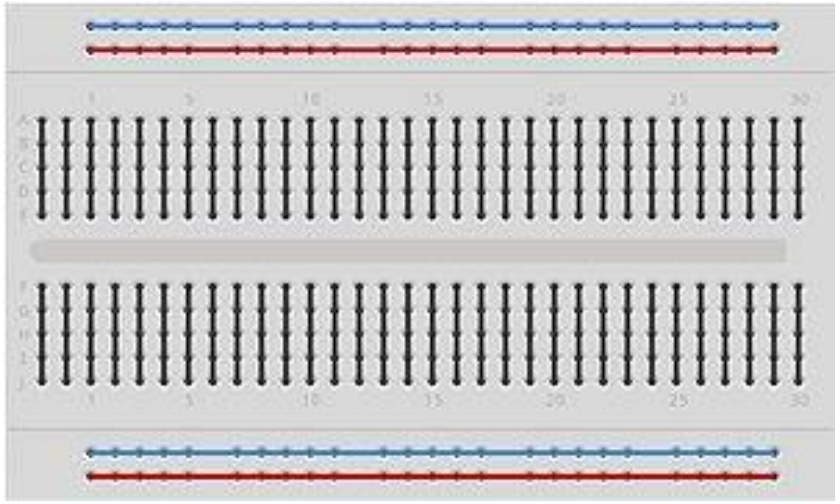


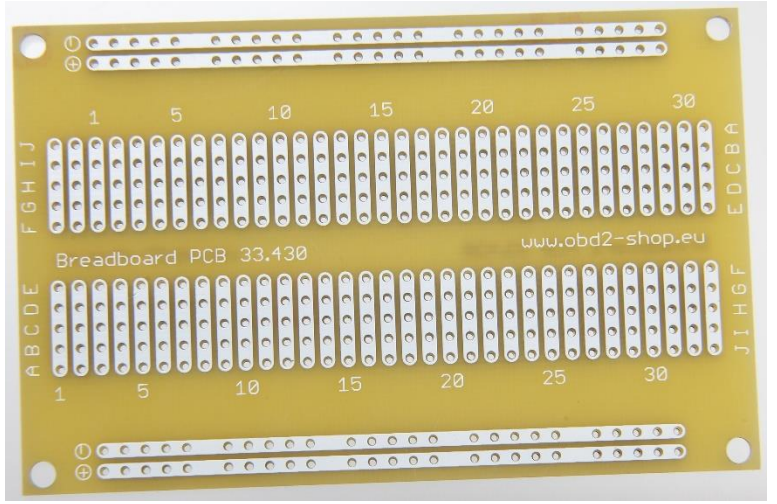

ZUM BT-328 - BQ

# BreadBoard

# ¿Cómo conectar componentes electrónicos? I

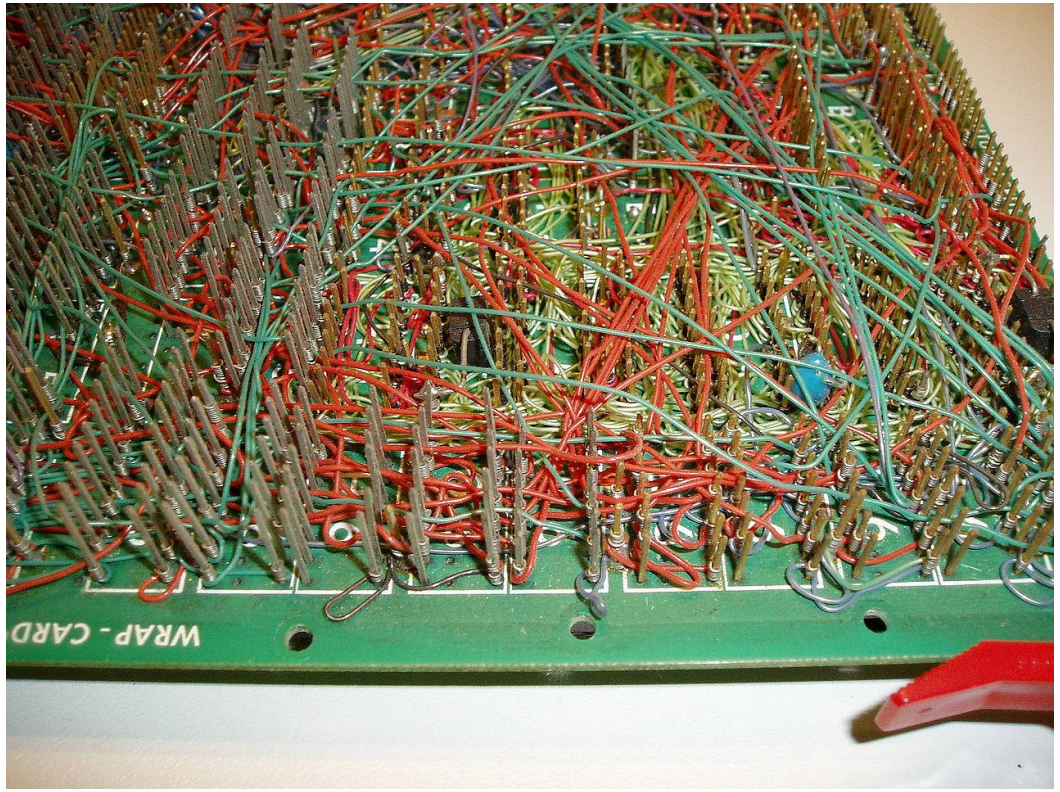
- **Crear un circuito** (sin o con la ProtoBoard)
- ProtoBoard / BreadBoard / Placa de pruebas
  - Crear y modificar circuitos encajando componentes
  - Basada en sistema de **raíles internos**
- Historia: https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard





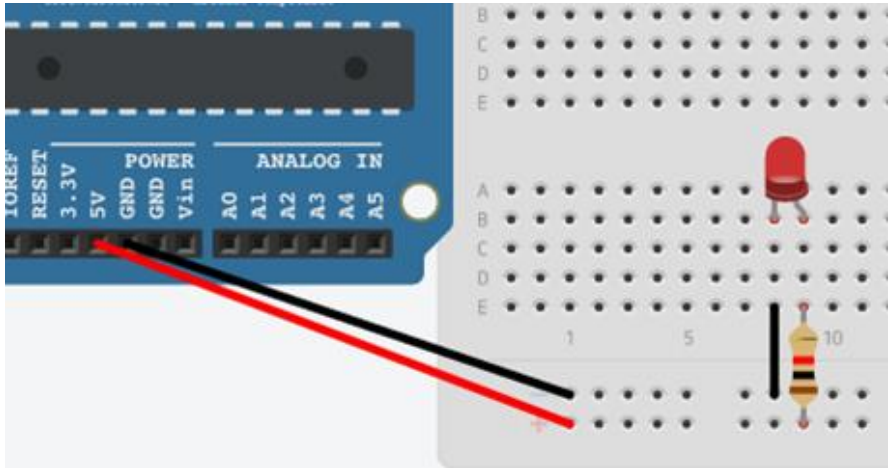
# Ayuda a…

- Evitar los líos antiguos de cables…

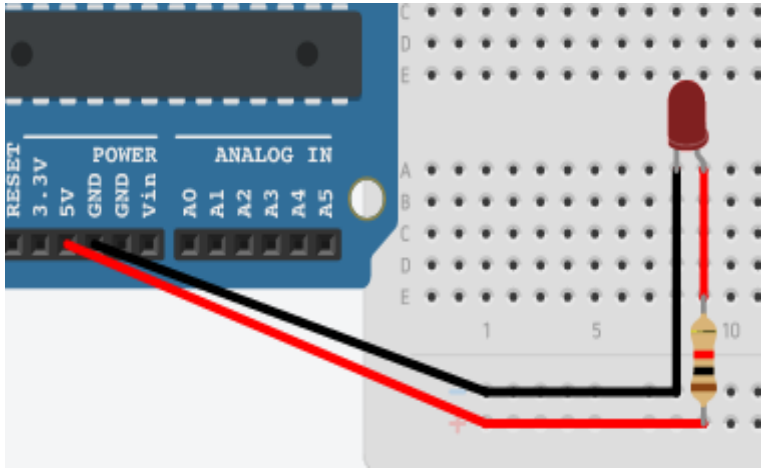



https://commons.wikimedia.org/wiki/File:Computerplatine_Wire-wrap_backplane_detail_Z80_Doppel-Europa-Format_1977.jpg

# Ejemplo I

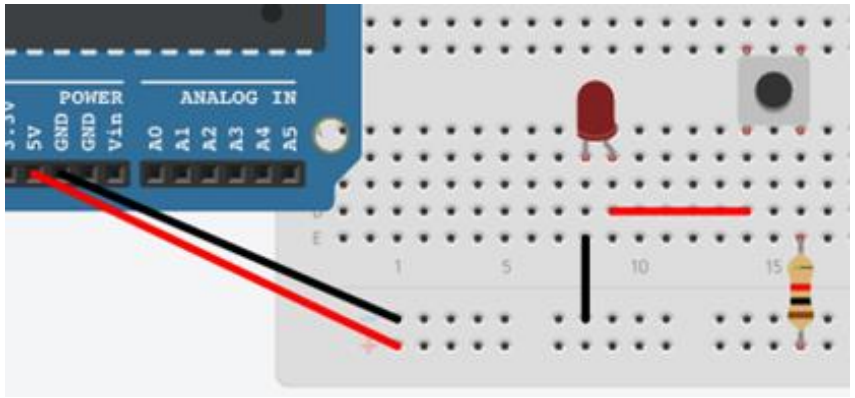



Conexión en la ProtoBoard

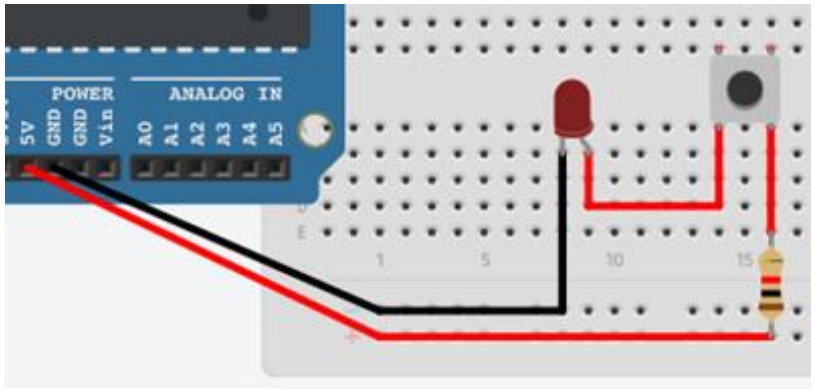



Circuito real equivalente

# Ejemplo II




Conexión en la ProtoBoard

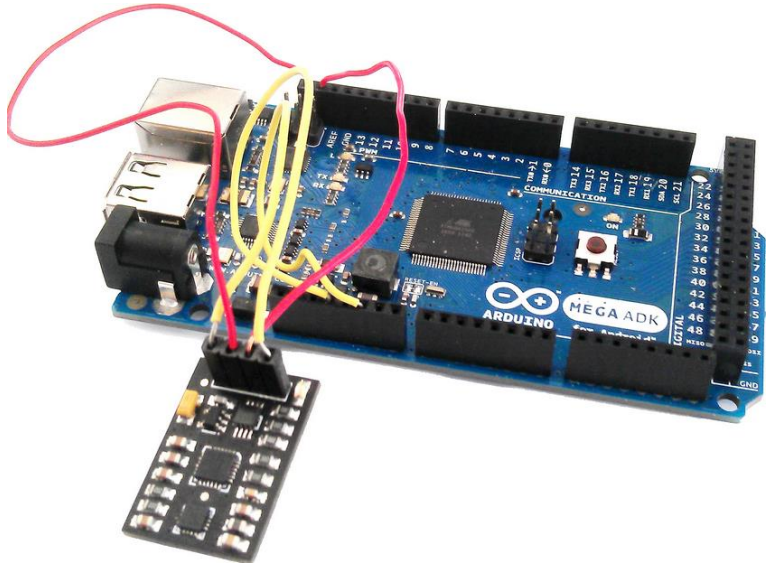


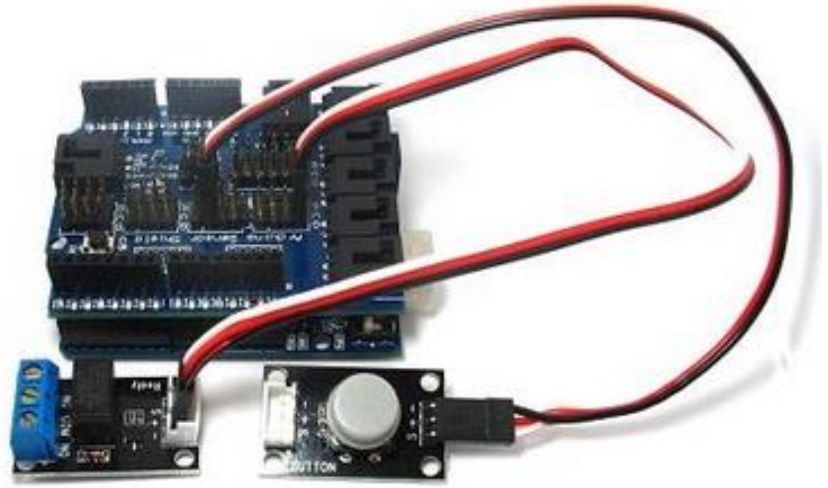

Circuito real equivalente

# Un poco de electrónica

# ¿Cómo conectar componentes electrónicos? II

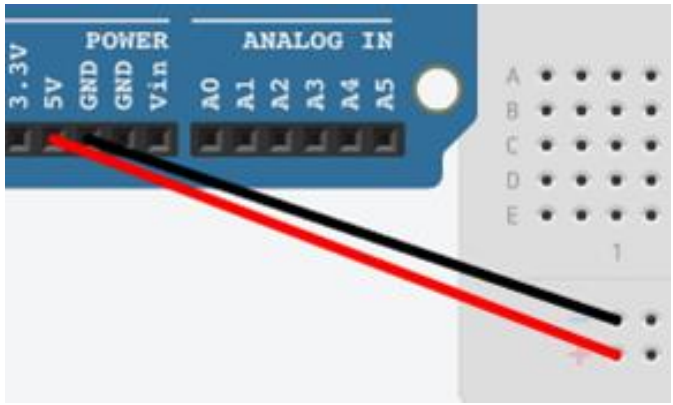
- Conectar directamente los componentes a Arduino
  - Conexión a los pines del Arduino
  - Placas de sensores (Sensor Shield)
  - Solo sensor vs Sensor + resistencias
- Utilizando una BreadBoard



# Creación de un circuito eléctrico I

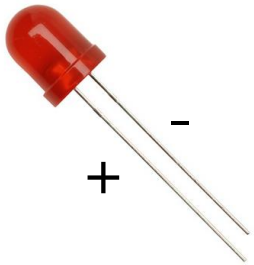
- Pin 5 voltios al carril + de la ProtoBoard
  - Esta placa tiene salidas de **5V** y 3.3V
- Pin GND (Tierra o negativo) al carril – de la ProtoBoard

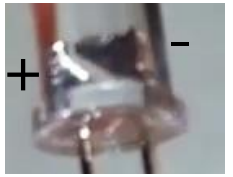




- **Respetar siempre el criterio de cables**: Rojo + y GND/Negativo –
  - Estándar
  - Evitar confundirnos (problema muy común)

# Creación de un circuito eléctrico II

- Conectar el componente electrónico al circuito
  - Diodo LED
  - Cada componente tiene sus especificaciones
    - Conexión
      - Ánodo + largo, cátodo- corto
      - Parte pequeña interna +, parte grande interna -
    - Voltaje (Voltios) o Intensidad (Amperios) requeridos
      - 10-20 mA (miliamperios)
      - Voltaje depende del color (1,7 a 4,6 v), y el tamaño
- Los componentes más complejos tienen otras especificaciones
  - Servomotor
    - Conexión
      - + -> Rojo, - -> Negro, Pin Digital -> Blanco/Naranja
    - Voltaje (Voltios) o Intensidad (Amperios) requeridos:
      - 4,8V – 6V
    - Torsión, velocidad, etc.

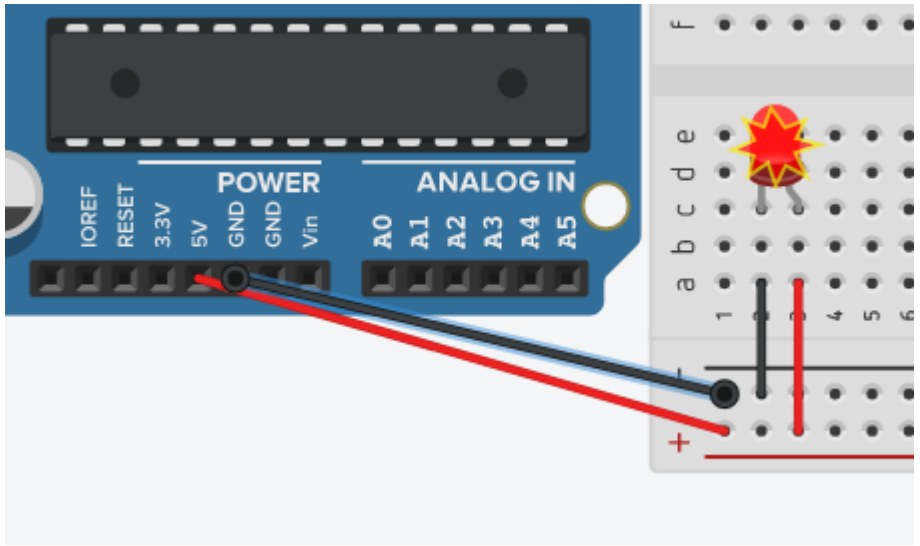



# Problemas

# Problema I

- ¿Conectar un led  10 - 20 mA a una alimentación de 5V?

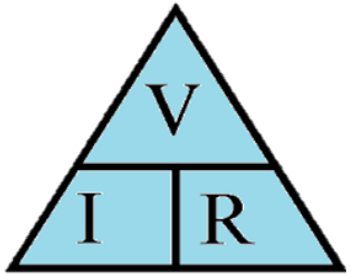



- ¿Qué esta pasando?

# Problema II

- **Ley de Ohm**

$$V = I \times R \qquad I = V / R \qquad R = V / I$$




- **Voltaje**: al conectar componentes al circuito -> el voltaje decae
  - *Casi todos los componentes (Un pulsador de clase no, un LED, etc.)
    - Esto hace que requiera una resistencia, para que haga impedancia
  - Un led (pequeño rojo) hace caer el voltaje 1,8V
    - **Los 5V del circuito pasan a: 5V – 1,8V = 3,2V**
- **Resistencia**: este circuito apenas tiene resistencia (cables, LED)
  - Unos 2Ω ohmios (Ω)
- **Intensidad**: podemos calcular la intensidad (A), que le llega al LED
  - V = I x R -> I = V / R
    - I = 3,2V / 2Ω = 1,6 A o lo que es lo mismo 1600 mA!!!
    - El LED requería entre 10-20 mA… ¡le llega entre 80-160 veces más!

# Problema III

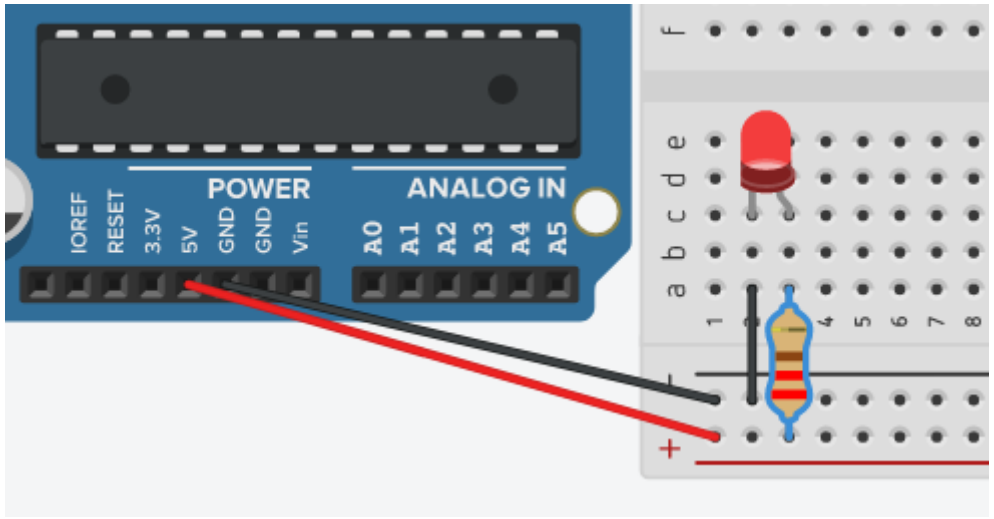
- ¿Cómo hacer que circule una intensidad de 10-20 mA?
  - 10 mA: brillo medio
  - 20 mA: brillo muy alto

  - Con menor voltaje o dificultando el paso de corriente (resistencia)
  - Queremos 15mA -> 0,015A ¿Qué resistencia hace falta?
    - **V = I x R -> V / I = R**
    - **5V – 1,8V (LED Rojo) = 3,2V**
    - **3,2V / 0,015 A = 213,33 Ω**
      - Arduino es 3,3V o 5V

  - Con **213,33 Ω** la intensidad de corriente seria aproximadamente 15mA
    - No hay resistencias comerciales de 213,33 Ω
    - Colocaríamos **la más próxima que es de 220 Ω**

# Problema IV

- ¡Solucionado con una resistencia de 220 Ω!
  - Sin resistencia iluminaría, pero su vida útil sería muy corta



# Problema V

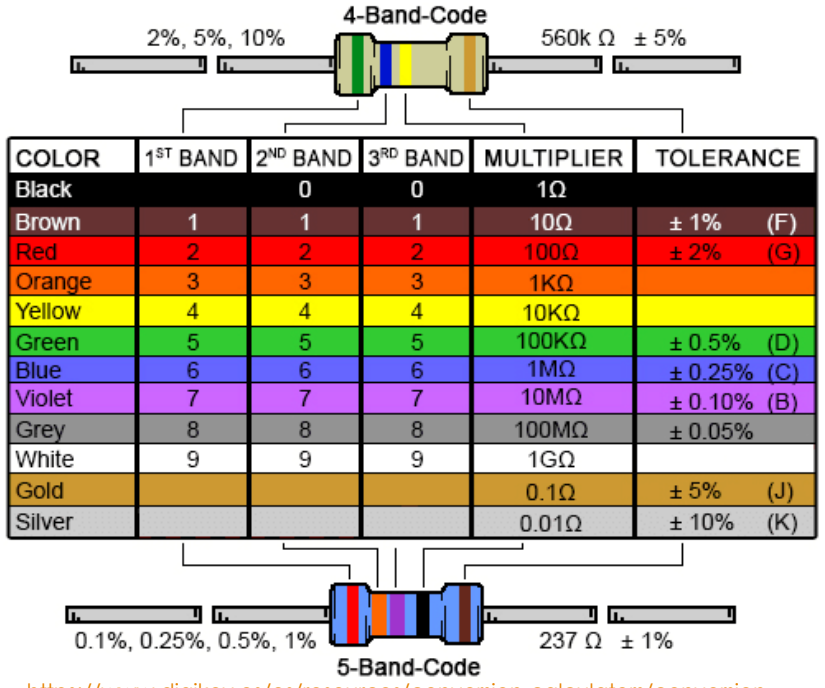
- ¿Por qué un LED de diferente color puede iluminar más o menos?
- ¿Estará estropeado?
- Depende del color… y del tamaño
  - Calculadora: http://gzalo.com/resistencias_led/

| Color | Caída de voltaje (Aproximada) | Resistencia (Ohmios) con 5v aprox. y 0,015A |
|---|---|---|
| **Infrarrojo** | 1,4 V | 270 |
| **Rojo (alto)** | **1,8 V** a 2,2 V | 220 |
| **Naranja** | **2,1 V** a 2,2 V | 200 |
| **Amarillo** | 2,1 V a 2,4 V | 200 |
| **Verde** | 2 V a 3,5 V | 150 |
| **Azul** | **3,5 V** a 3,8 V | 100 |
| **Violeta** | 3,6 V | 100 |
| **Blanco** | 3,8 V | 100 |

(Vcircuito – Vled) / A = R

# Problema VI

- ¿Cómo miro la resistencia?
  - Número de bandas: 4, 5 o 6
  - https://www.digikey.es/es/resources/conversion-calculators/conversion-calculator-resistor-color-code-5-band




| COLOR | 1ST BAND | 2ND BAND | 3RD BAND | MULTIPLIER | TOLERANCE |
|---|---|---|---|---|---|
| Black | | 0 | 0 | 1Ω | |
| Brown | 1 | 1 | 1 | 10Ω | ± 1% (F) |
| Red | 2 | 2 | 2 | 100Ω | ± 2% (G) |
| Orange | 3 | 3 | 3 | 1KΩ | |
| Yellow | 4 | 4 | 4 | 10KΩ | |
| Green | 5 | 5 | 5 | 100KΩ | ± 0.5% (D) |
| Blue | 6 | 6 | 6 | 1MΩ | ± 0.25% (C) |
| Violet | 7 | 7 | 7 | 10MΩ | ± 0.10% (B) |
| Grey | 8 | 8 | 8 | 100MΩ | ± 0.05% |
| White | 9 | 9 | 9 | 1GΩ | |
| Gold | | | | 0.1Ω | ± 5% (J) |
| Silver | | | | 0.01Ω | ± 10% (K) |

https://www.digikey.es/es/resources/conversion-calculators/conversion-calculator-resistor-color-code-5-band

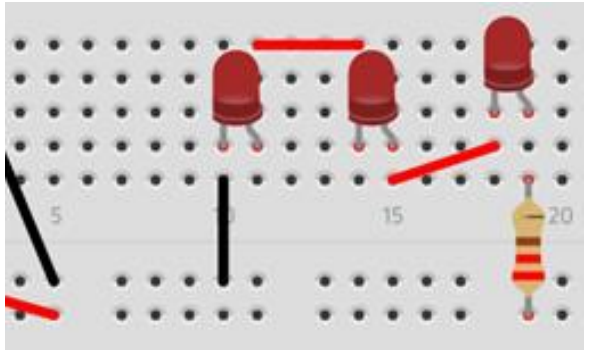
1st: 2 (Rojo)
2nd: 2 (Rojo)
Multiplier: 10
Tolerance: +-5%
Total: 220 Ohmios

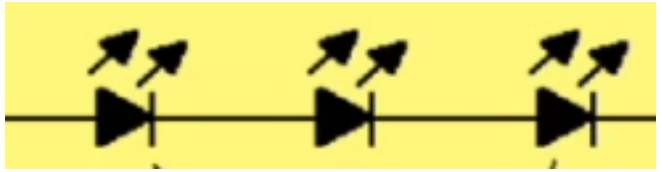
# Tipos de circuito

# Ejemplos

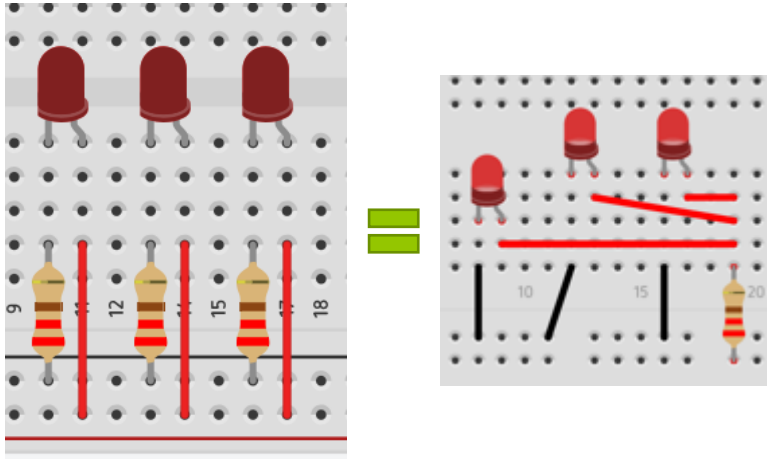
- 3 LEDs rojos
- Alimentación de 5 V
- Resistencia de 220 Ω

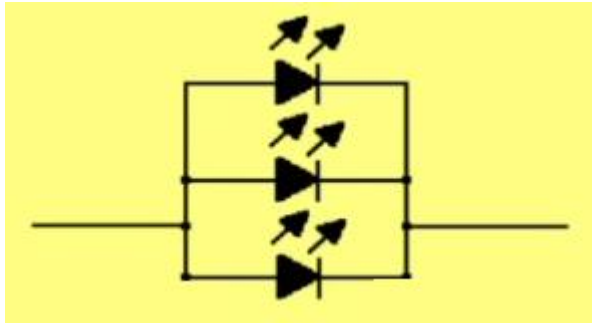
Serie



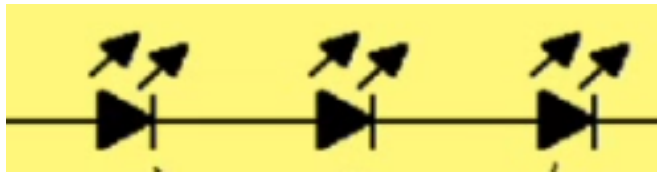


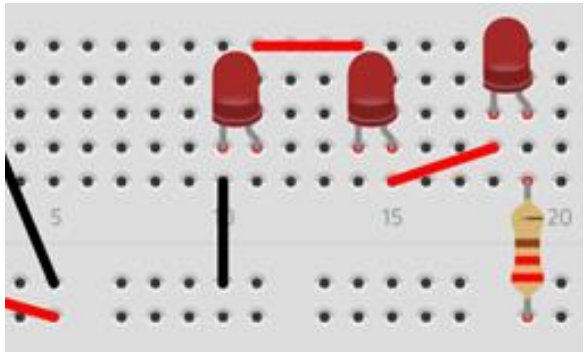
Paralelo





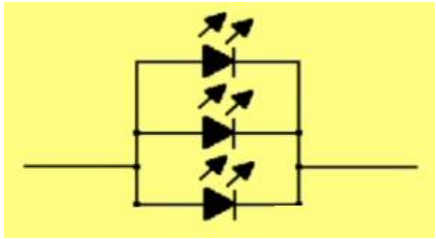
# Circuitos en serie

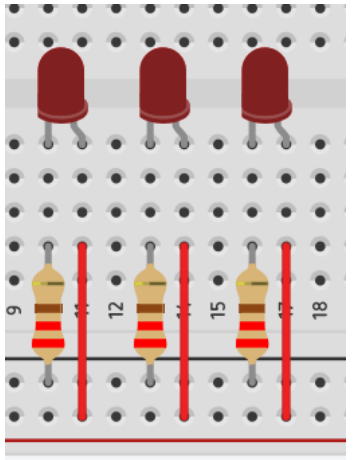




- **Las caídas de voltaje se suman** (aprox. 1,8V x 3 = 5,4 V)
  - Cada elemento tiene el suya propio
    - Si un elemento es eliminado, el voltaje de otro se podría hasta doblar
  - Necesitan 5,4V para funcionar y se esta alimentando con 3,3V
    - No funcionaría o funcionará mal, a medias, de vez en cuando, …
    - Si hubiera más voltaje: se calentará más, se puede estropear, funcionar diferente (+ rápido), etc.
  - V total = V1 + V2 + V3 + …
- **La intensidad es la misma en todo el circuito** (0,015 A = 15 mA)
  - I total = I1 = I2 = I3 = …
- **Todas las resistencias se suman** (220 Ω)
  - R total = R1 + R2 + R3 + …
  - *Con esta Rt los LEDs apenas brillarían
  - Si una resistencia se estropea, todo deja de funcionar
- **Si un elemento deja de funcionar, los demás también**
  - El elemento estropeado hace que se «corte» el circuito

# Circuitos en paralelo





- **Cada rama es como si fuera un circuito en serie**
- **Todas las ramas reciben el mismo voltaje** (5 V)
  - La caída es de 1,8V en cada rama
  - V total = V1 = V2 = V3 = …
- **Todos los caminos reciben la misma intensidad** (0,015 A = 15 mA)
  - El flujo de corriente total (intensidad) aumenta 15mA x 3 = 45 mA = 0,045 A
  - I total = I1 + I2 + I3 + …
- **La resistencia depende de la rama** donde esté (220 Ω)
  - Afecta solo a los elementos de esa rama
  - Las resistencias de cada rama son independientes
  - Si la resistencia está antes de las ramas, afecta a todas
  - R total del circuito = 1 / (1/R1 + 1/R2 + 1/R3 + …) (Rt = 73,3 Ω)
- **Si un elemento deja de funcionar, solo se estropea esa rama**, el resto de ramas siguen funcionando

# Programación en Arduino y pines

# Programación en Arduino

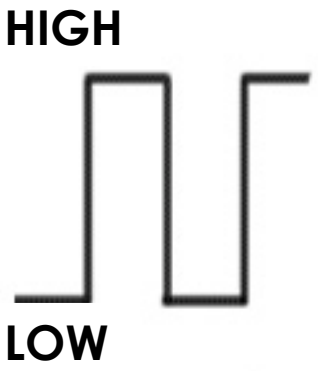
- Programas
  - Funciones básicas

```
1
2  void setup() {
3    // Inicialización
4
5  }
6
7  void loop() {
8    // funcionalidad continua a ejecutar en forma de bucle
9  }
```

  - Podemos: declarar #includes (importar librería), variables globales, otras funciones, etc.

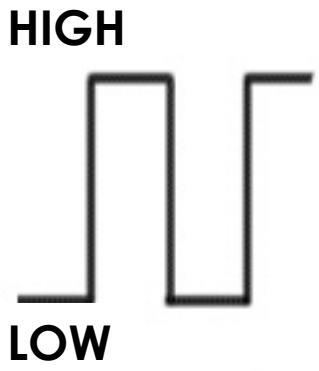
# Pines digitales I


- **Pines**: **2 – 13**
  - Los pines **0** (**RX** - recepción) y **1** (**TX** - transmisión) lo usa el chip ATMega
    - Se pueden reconfigurar
    - Se usan cuando se usa el puerto serie (USB)
  - El **13** tiene una **resistencia de 220 Ω incluida y un LED**
  - Pueden ser entradas o salidas
  - Al ser digitales, solo tienen dos valores
  - **I máxima** = 40 mA; **I recomendada** = 20 mA
    - 20 mA = salida de 5 voltios + resistencia > 200 Ω
  - **I totalSalidasArduino** = 300 mA; **I totalPuerto** = 150 mA
  - Suficiente para LEDs, servomotores pequeños, etc.
- Pines **PWM ~** -> 3, 5, 6, 9, 10, 11
  - Estos pines permiten enviar señales analógicas, además de las digitales



# Pines digitales II


- **Pines**: **2 – 13**
- Pines **PWM ~** -> 3, 5, 6, 9, 10, 11
- Como **salida**
  - Permiten emitir voltaje 5V HIGH o 0V LOW
    - High si hay 3v o más en 5 V
    - High si hay 2v o más en 3,3 V
  - Enviar señales a un sensor / actuador
  - Encender / apagar un componente, etc.
  - Objetos que tengan solo dos estados
- Como **entrada**
  - Detectan voltaje en base a un umbral (aprox. 2.5 V)
    - Detectado < umbral retornan LOW
    - Detectado > umbral retornan HIGH
  - Nunca deberían recibir voltajes fuera de 0 V – 5 V
  - Detectan si pasa corriente por un circuito
  - Recibir señales de un sensor, etc.

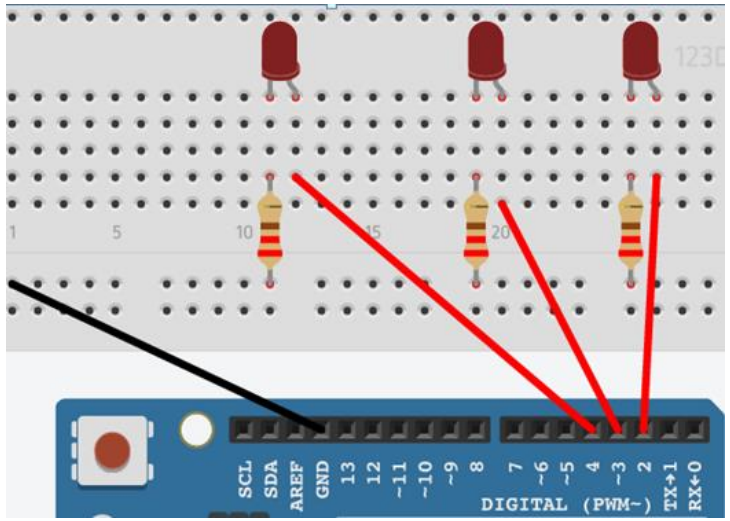


# Ejemplo – Salida digital – encender/apagar LEDs

- Inicializar Pin como salida
  - pinMode(pin, OUTPUT)

```
2  int led1 = 2;
3  int led2 = 3;
4  int led3 = 4;
5
6  void setup() {
7    pinMode(led1, OUTPUT);
8    pinMode(led2, OUTPUT);
9    pinMode(led3, OUTPUT);
10 }
```

- Escribir en la salida
  - digitalWrite(pin, HIGH / LOW)
    - HIGH: lo enciende
    - LOW: lo apaga

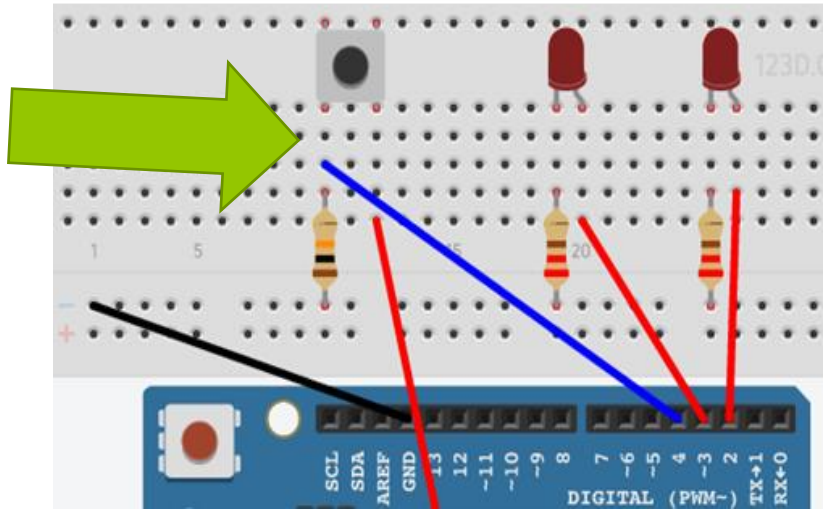



```
void loop() {
  digitalWrite(led1, HIGH);
  delay(500);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led2, LOW);
  digitalWrite(led3, HIGH);
  delay(500);
  digitalWrite(led3, LOW);
}
```

# Ejemplo – Entrada digital – detectar circuito cerrado

- Inicializar Pin como entrada
  - pinMode(pin, INPUT)

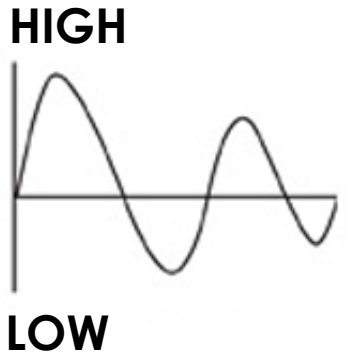
```
2  int boton = 4;
3
4  void setup() {
5    pinMode(boton, INPUT);
6  }
```




- Leer entrada
  - digitalRead(pin) -> retorna HIGH o LOW

```
9  void loop() {
10   int lectura = digitalRead(boton);
11   if( lectura == HIGH){
12       // ...
13   }
14 }
```

# Pines analógicos

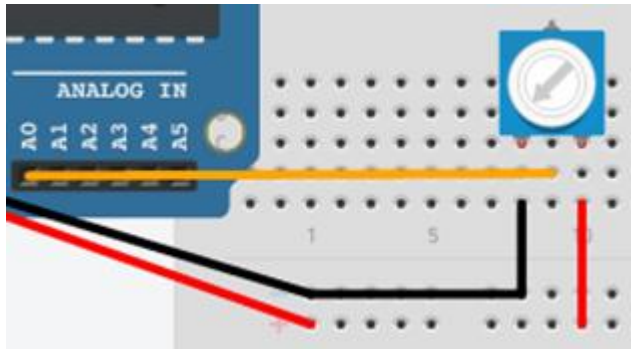




- Pines: **A0 – A5**
- Trabajan **solo** como **entradas** (Arduino Uno)
  - Hay otros Arduino que sí tienen salidas analógicas: Zero, Due, …
- Pueden tomar cualquier valor dentro de un intervalo
- Son **más escasas, lentas y caras que las digitales**
- **Ofrecen**
  - Arduino Uno: **1024 valores diferentes** (10 bits)
  - Transforma el valor analógico a un valor digital usando 10 bits (0 – 1023)
    - **Precisión relativa del 0,1% a 5 V**. Si fuera de 1V sería de de 0,5% (Depende del voltaje)
- **Entrada**
  - Detecta valores de tensión entre 0 V – 5 V
  - Detecta cuanta corriente pasa por un circuito
- **Usos**
  - **Leer datos de** un potenciómetro, motor, fotoresistores, u otro **elemento que pueda dar más de 2 estados**
  - Entrada de un sensor de temperatura analógico, etc.
- Pines **A4 y A5** incluyen la biblioteca WIRE para interfaces TWI
  - Son un tipo de comunicación y depende del dispositivo que se utilice
  - Ejemplo: I2C (bus de comunicación) de la LCD

# Ejemplo – Entrada analógica – leer valor de un potenciómetro

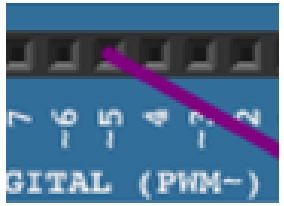
- No requiere configuración
- Leer entrada
  - analogRead(pin) -> valor entre [0 – 1023]

```
1  int potencia = A0;
2
3  void loop() {
4      int lectura = analogRead(potencia);
5  }
```

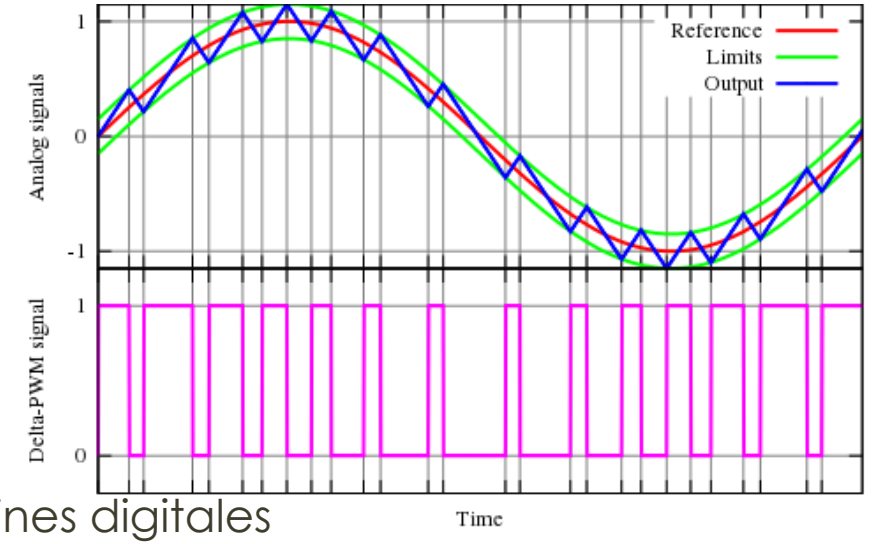


# Pines digitales PWM (Pulse-With-Modulation)

- Pulse-with-modulation
  - Modulación por ancho de pulso
  - Simula una salida analógica
    - Modifica el ancho del pulso




https://commons.wikimedia.org/wiki/File:Delta_PWM.svg

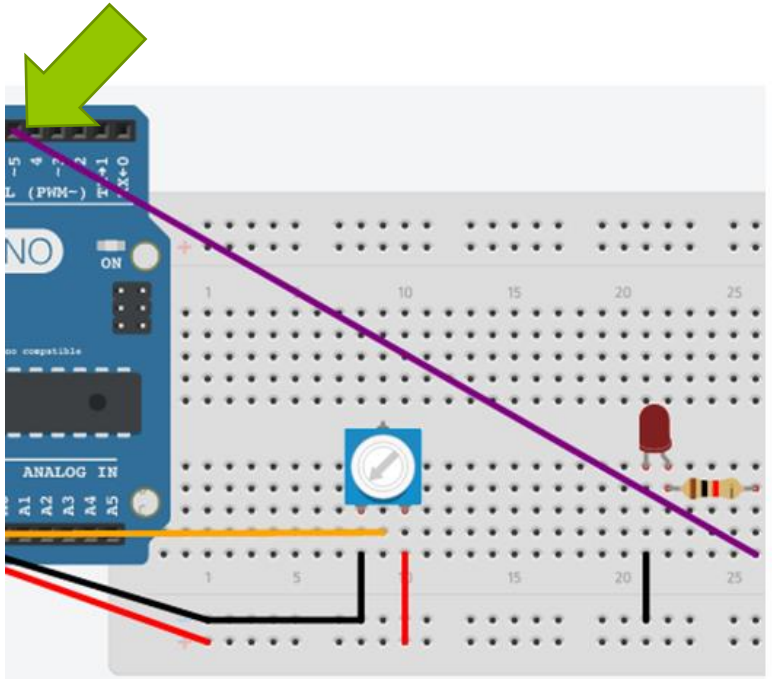



- Pueden generar salidas analógicas desde pines digitales
  - Pines: **~3, ~5, ~6, ~9, ~10, ~11**
  - Modifica el periodo de la señal digital para «emular» valores (0-255)
    - **~5, ~6**: tienen una mayor frecuencia que el resto, 980Hz
  - Escribir salida (solo una a la vez)
    - 0 – 255
  - Arduino: https://www.arduino.cc/en/Tutorial/PWM

# Ejemplo – Salida analógica – Regular el voltaje de un LED (regular brillo)

- Configurar Pin PWM como salida
  - pinMode(pin, OUTPUT)

```
int potencia = A0;
int led = 5;

void setup(){
    pinMode(led, OUTPUT);
}
```



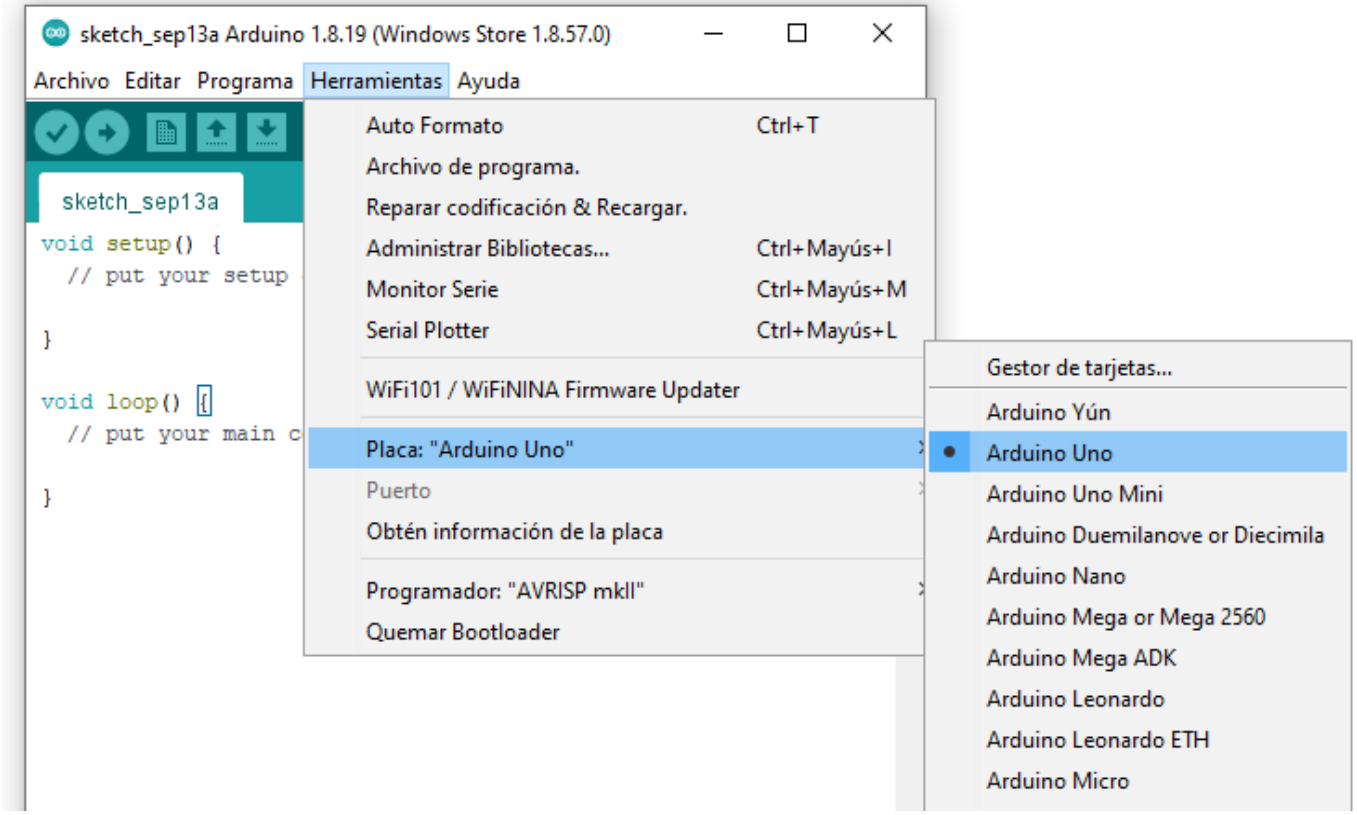

- Leer analógico y escribir PWM
  - analogWrite(pin, [0-255])
  - Map: cambia el rango A, B a A,C

```
void loop() {
    int lectura = analogRead(potencia);

    int voltajeLed = map(lectura, 0, 1023, 0, 255);

    analogWrite(led, voltajeLed);
}
```

# IDE

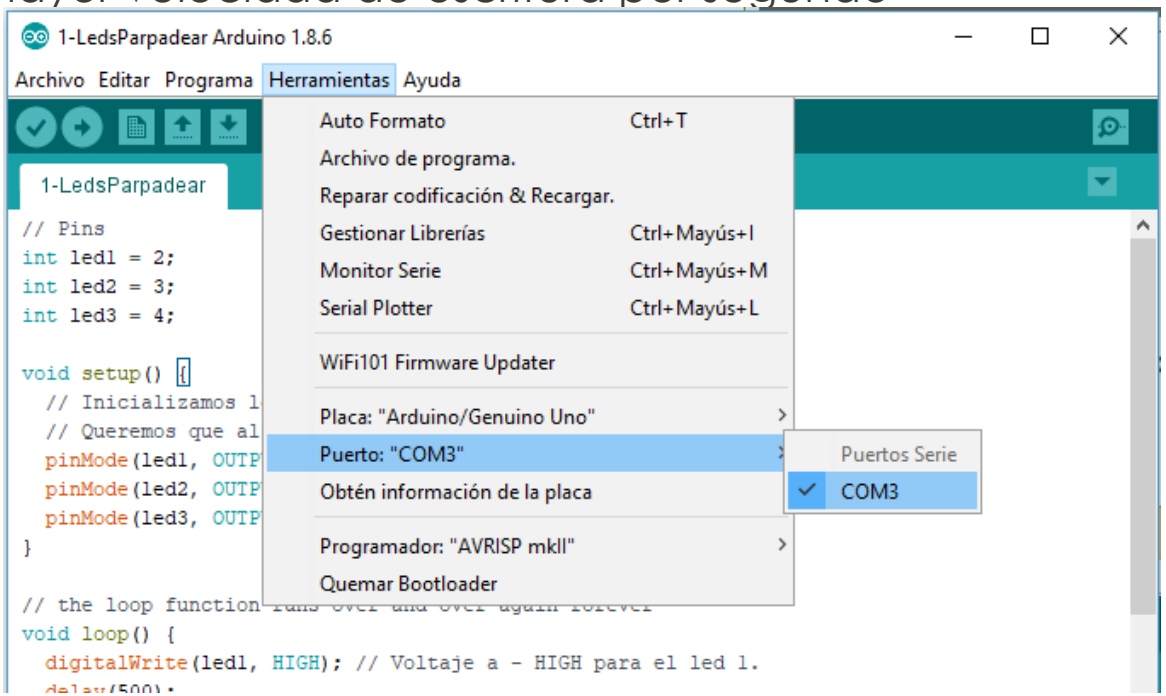
# IDE de desarrollo – Acciones básicas I

- Selección de modelo de placa



# IDE de desarrollo – Acciones básicas II

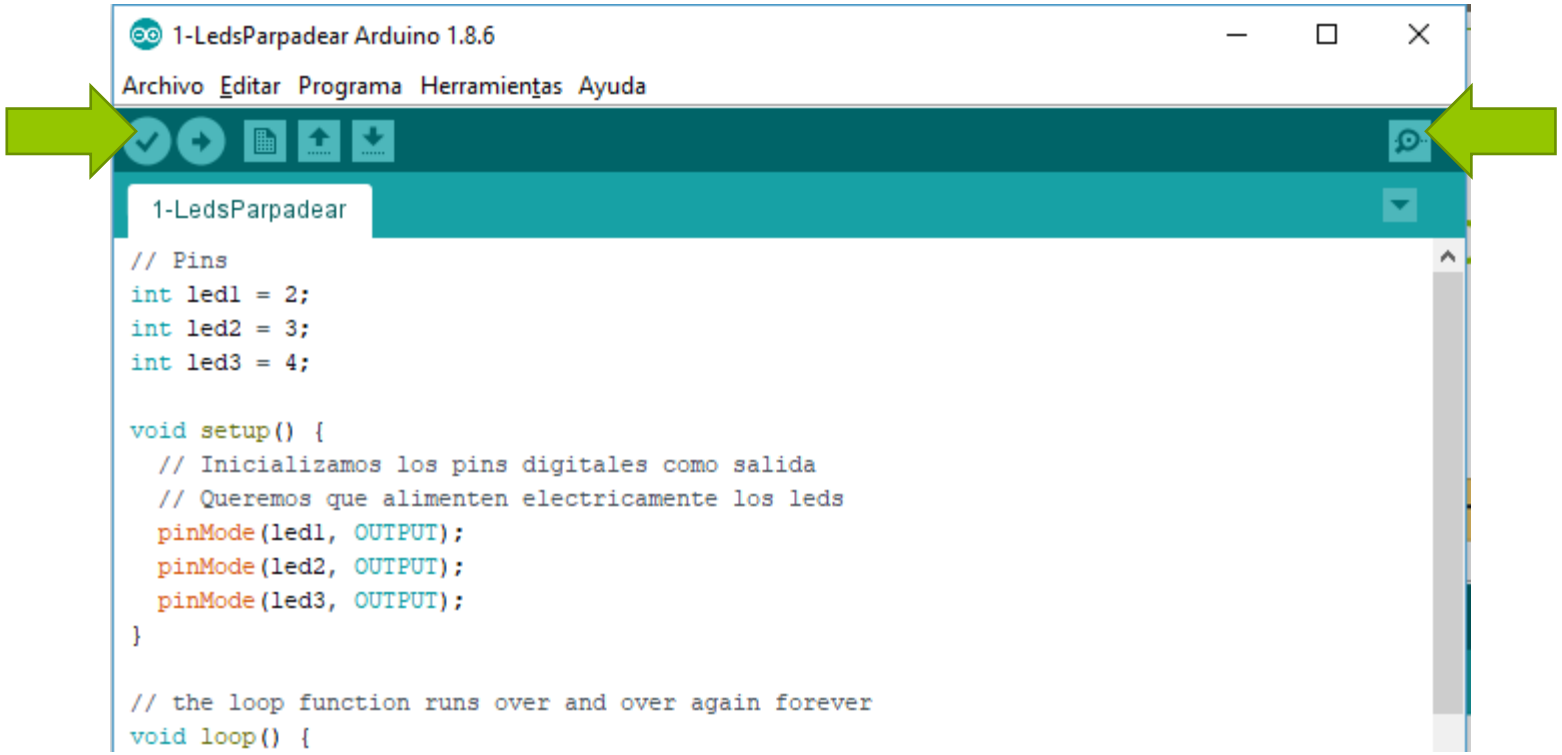
- Selección del puerto de conexión (USB)
  - Serial.begin(9600): conexión con el puerto serie usando 9600 baudios (lo típico)
  - https://www.arduino.cc/en/Serial/Begin
  - A más baudios, mayor velocidad de escritura por segundo



# IDE de desarrollo: acciones básicas

 Validar código  Subir código a la placa

 Abrir el monitor de serie (especie de consola E/S)



# Lenguaje de programación

- https://www.arduino.cc/reference/en/
- En general es muy sencillo

**Digital I/O**

digitalRead()
digitalWrite()
pinMode()

**Analog I/O**

analogRead()
analogReference()
analogWrite()

**Zero, Due & MKR Family**

analogReadResolution()
analogWriteResolution()

**Math**

abs()
constrain()
map()
max()
min()
pow()
sq()
sqrt()

**Trigonometry**

cos()
sin()
tan()

**Random Numbers**

random()
randomSeed()

**Bits and Bytes**

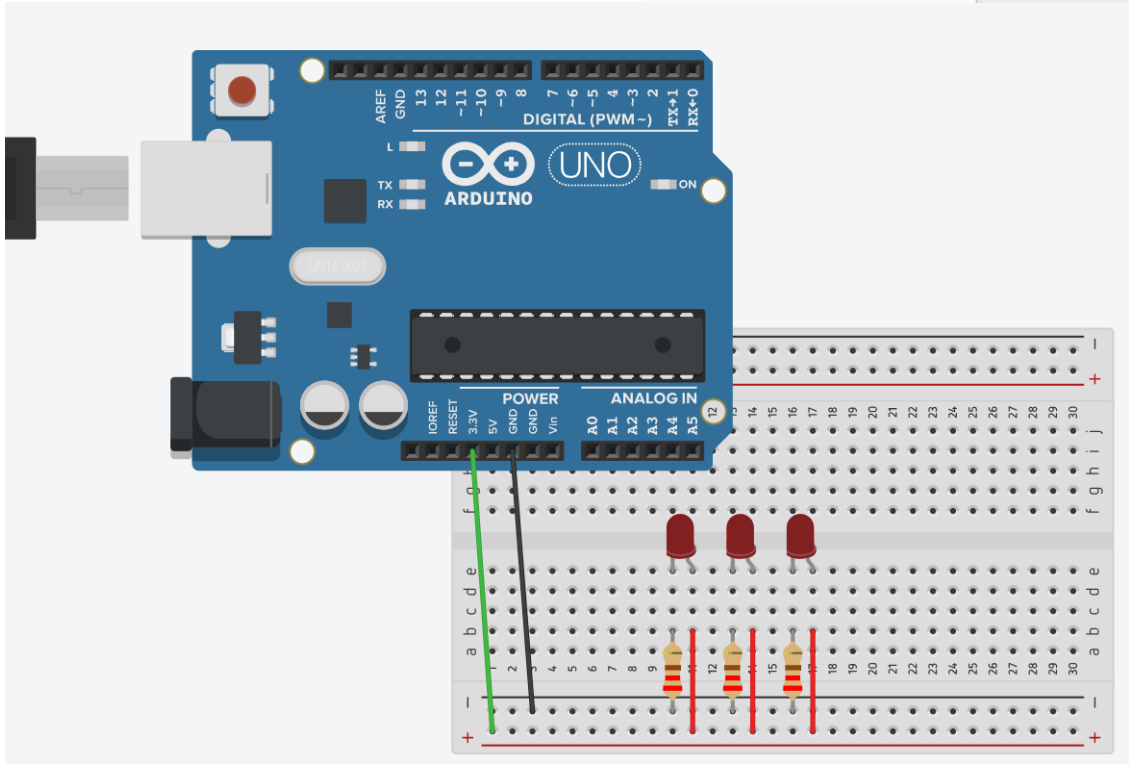
bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

**External Interrupts**

attachInterrupt()

# Emulación de Arduino

- Emulación: Tinkercad (Circuits)
  - https://www.tinkercad.com/

# Referencias

- Arduino Home
  - https://www.arduino.cc
- Emulación y ejemplos
  - https://www.tinkercad.com/
- Lenguaje de programación
  - https://www.arduino.cc/en/Reference/HomePage
- Ejemplos kit básicos
  - http://wiki.seeedstudio.com/Arduino/
- Libro: Arduino Curso práctico de formación 2014
- Foro Arduino StackExchange
  - https://arduino.stackexchange.com/

# Introducción a Arduino


Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Cristian González García
gonzalezcristian@uniovi.es


Basado en el material original de Jordán Pascual Espada

v 1.3.1 Septiembre 2021