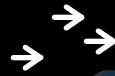




Robótica aplicada con Arduino



# ROBÓTICA APLICADA CON ARDUINO

15 JULIO 2013

Programación en ARDUINO

Gabriel J. García Gómez

# Índice



- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca



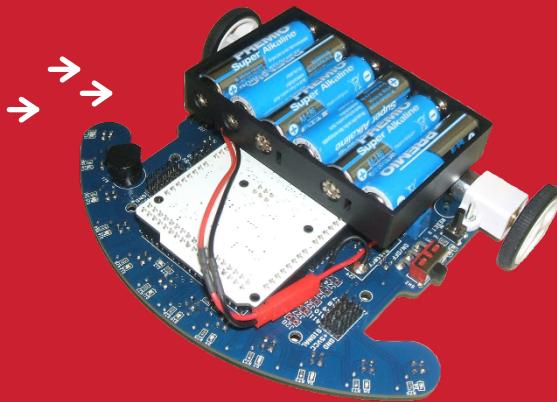
Robótica aplicada  
con Arduino



dfests  
ua.es



Robótica aplicada con Arduino



# LENGUAJE C A BAJO NIVEL

# El lenguaje C a bajo nivel



11110000	0xF0	$2^7 + 2^6 + 2^5 + 2^4 = 240$
00010001	0x11	$2^4 + 2^0 = 17$
00011111	0x1F	$2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 31$



Robótica aplicada  
con Arduino



# Operadores a nivel de bit en C

char a = 0x37;  
char b = 0x76;

Operador	Descripción	Ejemplo		
&	AND	a & b	$\begin{array}{r} 00110111 \\ 01110110 \\ \hline 00110110 \end{array}$	0x36
	OR	a   b	$\begin{array}{r} 00110111 \\ 01110110 \\ \hline 01110111 \end{array}$	0x77
^	XOR	a ^ b	$\begin{array}{r} 00110111 \\ 01110110 \\ \hline 01000001 \end{array}$	0x41
~	NOT	a ~ b	$\begin{array}{r} 00110111 \\ 11001001 \end{array}$	0xc9
<< X	Desplaza X bits a la izda	a << 3	00110111 << 3 = 10111000	0xb8
>> X	Desplaza X bits a la dcha	a >> 3	00110111 >> 3 = 00000110	0x06



Robótica aplicada  
con Arduino





# Máscaras para manejar bits en C

- Funciones interesantes:

- `char Consultar_bit(char *byte, char posicion_bit);`
- `void Bit_a_0(char *byte, char posicion_bit);`
- `void Bit_a_1(char *byte, char posicion_bit);`



Robótica aplicada  
con Arduino





# Función Consultar\_bit

```
char Consultar_bit(char *byte, char posicion_bit)
{
    char resultado;

    resultado = *byte >> posicion_bit;
    resultado = resultado & 0x01;

    return resultado;
}
```

## Ejemplo:

```
char a = 0x41;
void main () {
    char b,c;
    b = Consultar_bit(&a, 3);
    c = Consultar_bit(&a, 6);
}
```

0x41 >> 3	01000001 >> 3	00001000	0x08
0x08 & 0x01		00001000 00000001 <u>00000000</u>	b = 0x00
0x41 >> 6	01000001 >> 6	00000001	0x01
0x01 & 0x01		00000001 00000001 <u>00000001</u>	c = 0x01



Robótica aplicada  
con Arduino



# Función Bit\_a\_0

```
void Bit_a_0(char *byte, char posicion_bit)
{
    char mascara;
    mascara = ~(0x01 << posicion_bit);
    *byte &= mascara;
}
```

## Ejemplo:

```
char a = 0x41;
void main () {
    Bit_a_0(&a, 3);
    Bit_a_0(&a, 6);
}
```



Robótica aplicada  
con Arduino



0x01 << 3	00000001 << 3	00001000	0x08
~0x08	<u>00001000</u> <u>11110111</u>		0xf3
0x41 & 0xf3	<u>01000001</u> <u>11110111</u> <u>01000001</u>	b = 0x41	
0x01 << 6	00000001 << 6	01000000	0x40
~0x40	<u>01000000</u> <u>10111111</u>		0xbf
0x41 & 0xbf	<u>01000001</u> <u>10111111</u> <u>00000001</u>	c = 0x01	

# Función Bit\_a\_1

```
void Bit_a_1(char *byte, char posicion_bit)
{
    char mascara;
    mascara = 0x01 << posicion_bit;
    *byte |= mascara;
}
```

## Ejemplo:

```
char a = 0x41;
void main () {
    Bit_a_1(&a, 3);
    Bit_a_1(&a, 6);
}
```



Robótica aplicada  
con Arduino



0x01 << 3	00000001 << 3	00001000	0x08
0x41   0x08	01000001 00001000 01001001	b = 0x49	

0x01 << 6	00000001 << 6	01000000	0x40
0x41   0x40	01000001 01000000 01000001	c = 0x41	

# Índice

→ → →

- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca



Robótica aplicada  
con Arduino



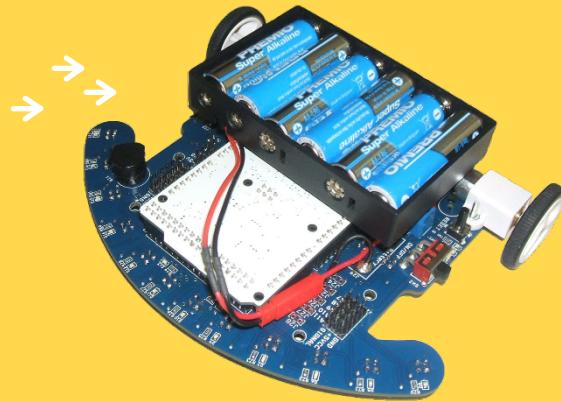


Robótica aplicada con Arduino



# INTRODUCCIÓN

## AL ROBOTÓICA





# Introducción

- Desarrollando la idea:
  - "Arduino es una **plataforma de electrónica abierta** para la **creación de prototipos** basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear **entornos u objetos interactivos.**"



Robótica aplicada  
con Arduino





# ¿Para qué se utiliza?

- Proyectos de Informática Industrial
  - Investigación
  - Instalaciones interactivas
  - Diseño rápido de prototipos
- 
- Para desarrollar nuevos proyectos e interacciones más allá del tradicional ratón, teclado y monitor



Robótica aplicada  
con Arduino





# ¿Qué se puede hacer?

- Sensores (detectar si pasa algo)
  - Pulsadores, teclados, interruptores.
  - Resistencias variables (ej. Control de volumen / deslizadores).
  - Fotoresistencias (detección de niveles de luz).
  - Termistores (temperatura).
  - Ultrasonidos.
- Actuadores (hacer que pase algo)
  - Luces, LED's.
  - Motores.
  - Altavoces.
  - Pantallas (LCD).



Robótica aplicada  
con Arduino



# ■ ¿Por qué Arduino?

- Open Source, en términos de hardware y software.
- Coste bajo ( $\approx 50\text{€}$  el ARDUINO Mega/DUE).
- Comunicación con el ordenador: serie, USB, Bluetooth, Ethernet...
- Alimentación a través de USB o con corriente continua para su funcionamiento autónomo.

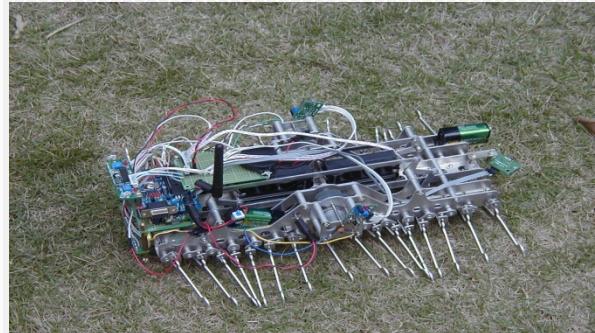


Robótica aplicada  
con Arduino



# ■ ¿Por qué Arduino?

- Puede funcionar sin estar conectado al ordenador (chip programable) y dispone de una pequeña cantidad de memoria.
- Trabaja con señales digitales y analógicas. Sensores y actuadores.
- Se pueden utilizar incluso para desarrollar robots sencillos.



Robótica aplicada  
con Arduino



# Índice



- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca

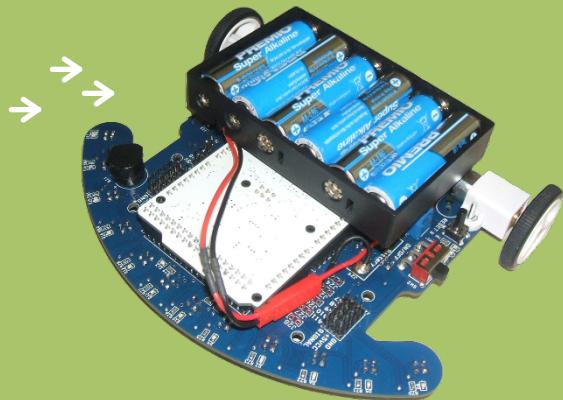


Robótica aplicada  
con Arduino





Robótica aplicada con Arduino



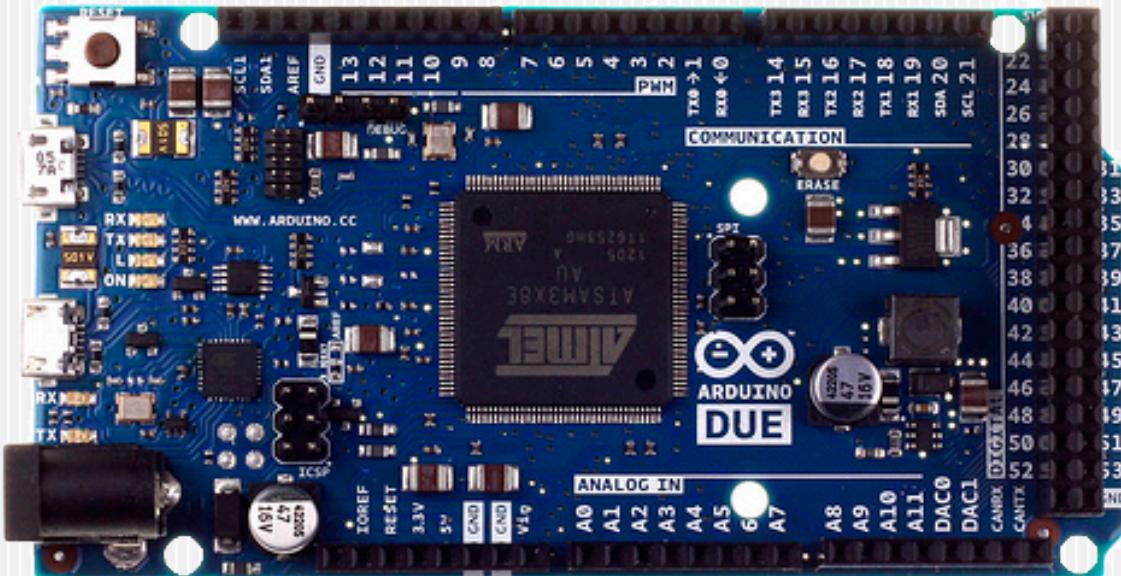
# COMPONENTES DEL ARDUINO





# Elementos básicos

- Entradas
  - Salidas
  - Entrada de programación



# Robótica aplicada con Arduino



 dfests   
.ua.es

# Tipos de señales

Digitales



Diferencia entre 2 posibles estados  
Ej: frio – calor o sistema binario usado por ordenadores (switch)

Analógicas



Diferencia entre múltiples estados.  
Rango: tibio  
Ej: dimmer



Robótica aplicada  
con Arduino



# Entradas/salidas digitales (DI-DO)

- Pines digitales

- son salidas o entradas que reciben niveles altos (5V) o bajos (0V) de tensión y que son interpretados como un 1 o un 0 respectivamente (3,3V en el caso de DUE)
- los pines 0 (TX) y 1 (RX) se emplean para la comunicación en serie o comunicación de Arduino con otros dispositivos.

- Comandos:

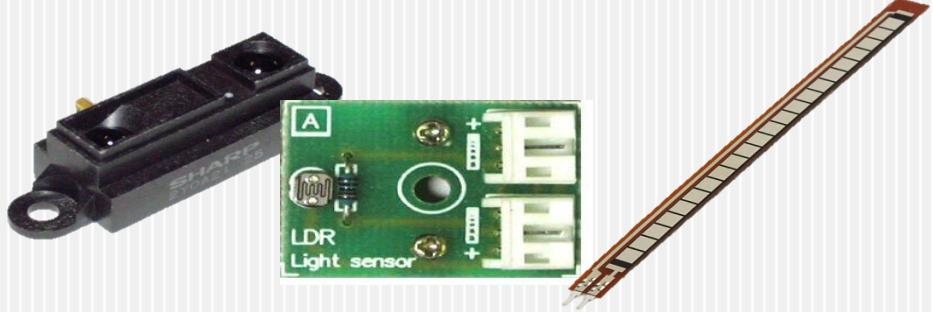
- **pinMode(pin, mode)**
  - configura el pin como entrada o salida
  - pin corresponde al número del pin y mode puede ser INPUT o OUTPUT
- **digitalWrite(pin, value)**
  - escribe un 0 o un 1 (0 o 5V) en el pin especificado (3,3V en el caso de DUE)
- **int digitalRead(pin)**
  - lee el valor de un pin digital



Robótica aplicada  
con Arduino



# Entradas analógicas



- **Pines analógicos**
  - Reciben tensiones **entre** 3.3V y 0V
  - No necesitan ser declarados como modo INPUT o OUTPUT (al contrario que los pines digitales)
- **Conversión Analógico-Digital (ADC)**
  - Convierten tensiones de 0 a 3.3 voltios en números enteros que van del 0 al 4095. En otras palabras, representa la información en números de 12 bits. Por compatibilidad usa por defecto sólo 10 bits (de 0 a 1023).
- **Comandos básicos:**
  - **analogRead(pin)**
    - lee o captura el valor de entrada del pin analógico especificado.
- **Ejemplos:** resistencias variables (potenciómetro – fotocélula)
- Cada sensor posee su propia escala por lo que probablemente se deba realizar el cálculo pertinente para obtener el valor correcto de la medición.



Robótica aplicada  
con Arduino





# Salidas analógicas – PWM y DAC

- Pulse-Width Modulation o modulación por anchura de pulso.
- Ancho de pulso, representa al ancho (en tiempo) del pulso con una modulación o cantidad de pulsos (estado on/off) por segundo.
- El periodo es medido en segundos y la frecuencia en Hz.
- La señal PWM se utiliza como técnica para controlar circuitos analógicos, comúnmente usadas para el control de motores DC (si se decrementa la frecuencia, la inercia del motor es mas pequeña y el motor se mueve más lentamente), ajustar la intensidad de brillo de un LED, etc.
- En ARDUINO DUE la señal de salida PWM (pines 2 a 13) es una señal de frecuencia constante (aproximadamente 490 Hz) y que sólo permite cambiar el "duty cycle" o el tiempo que el pulso esta activo (on) o inactivo (off), utilizando la función **analogWrite()**. Por defecto la salida tiene una precisión de 8 bits por compatibilidad con programas anteriores de ARDUINO, pero se puede configurar a 12.
- ARDUINO DUE tiene dos salidas específicas para circuitos analógicos (DAC0 y DAC1). Al contrario que las salidas PWM estas salidas son conversores digitales-analógicos y actúan como salidas analógicas reales.

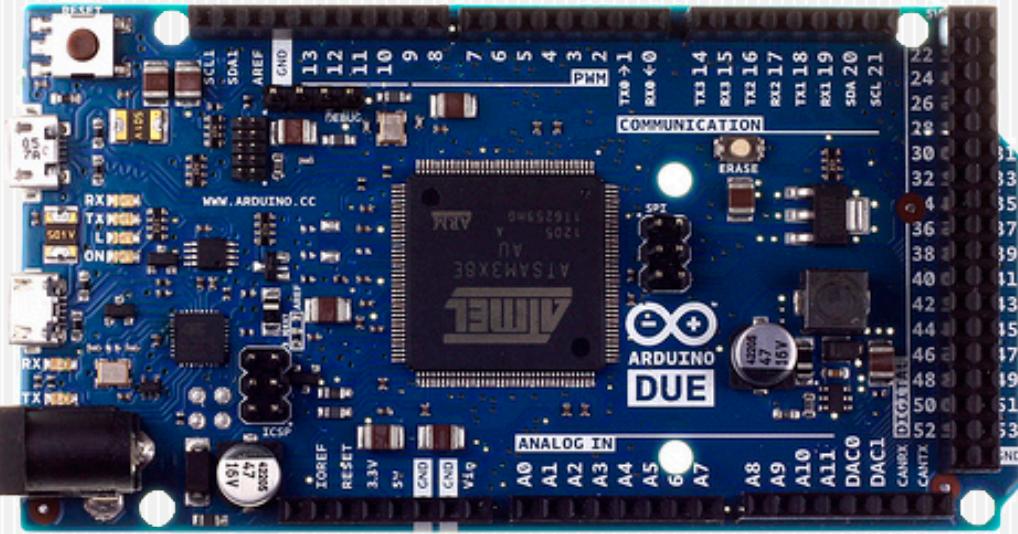


Robótica aplicada  
con Arduino



# Elementos de la placa

- Voltaje de referencia para entrada analógica
- PWM:
  - Valores high/low.
  - PWM para escritura analógica
- Serie (4)
  - Comunicación Serie TX/RX
- I2C: SDA y SCL.
- USB:
  - Nativo
  - De programación



Robótica aplicada  
con Arduino

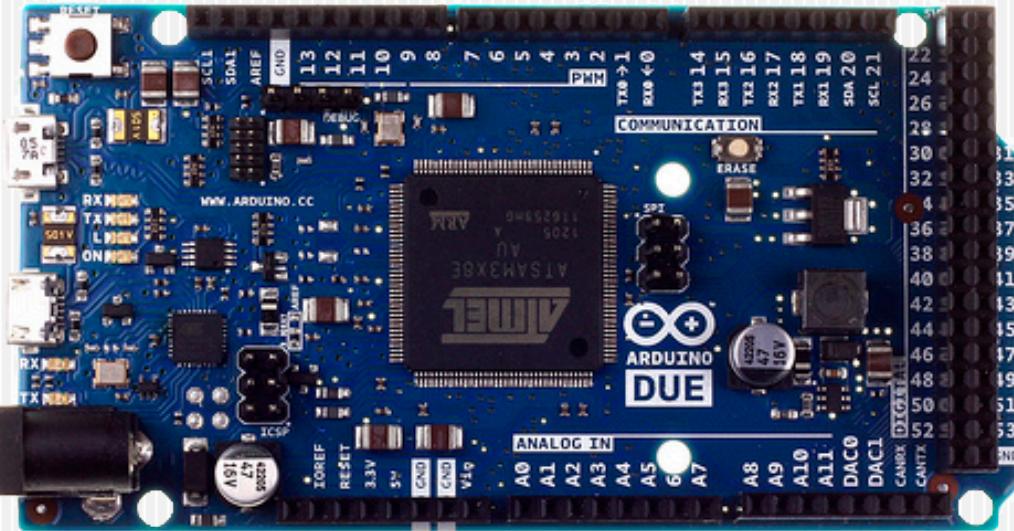


Máster Oficial en  
**Automática y Robótica**  
Universidad de Alcalá

dfests  
ua.es

# Elementos de la placa

- Alimentación externa
  - Batería, pared
  - Extensión de botón reset
  - Fuentes de alimentación de 3.3V y 5V estabilizados
- Vin
  - Conecta con Alimentación externa
- In analógicas
  - Lecturas analógicas 0-1023 niveles
- Out analógicas: DAC0 y DAC1.

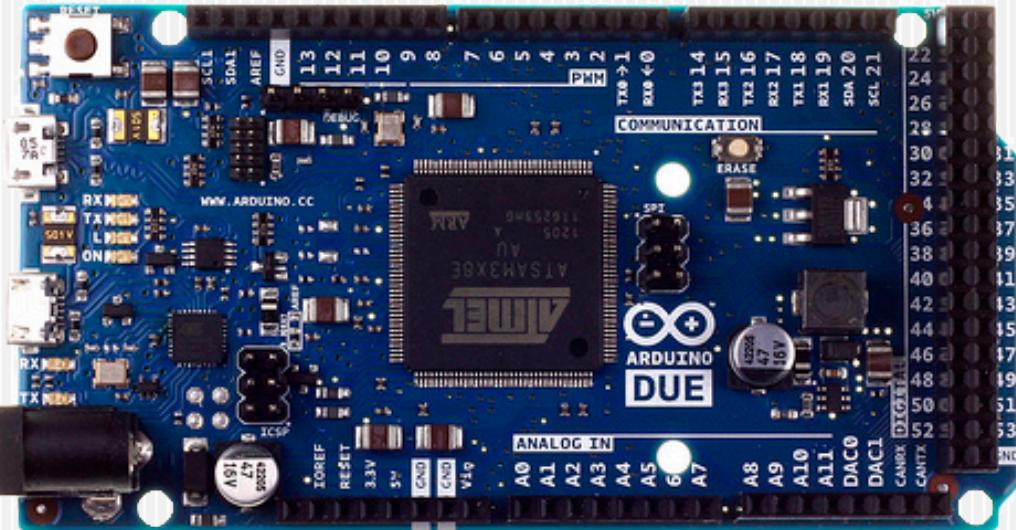


Robótica aplicada  
con Arduino



# Características del microcontrolador ARM:

- AT91SAM3X8E.
- Microcontrolador de 32-bit, permite operaciones de 4 bytes de datos en un simple ciclo de reloj.
- Reloj de la CPU a 84Mhz.
- 96 Kbytes de SRAM.
- 512 Kbytes de memoria Flash para código.
- Una controladora DMA, que permite liberar a la CPU de tareas intensivas de memoria.

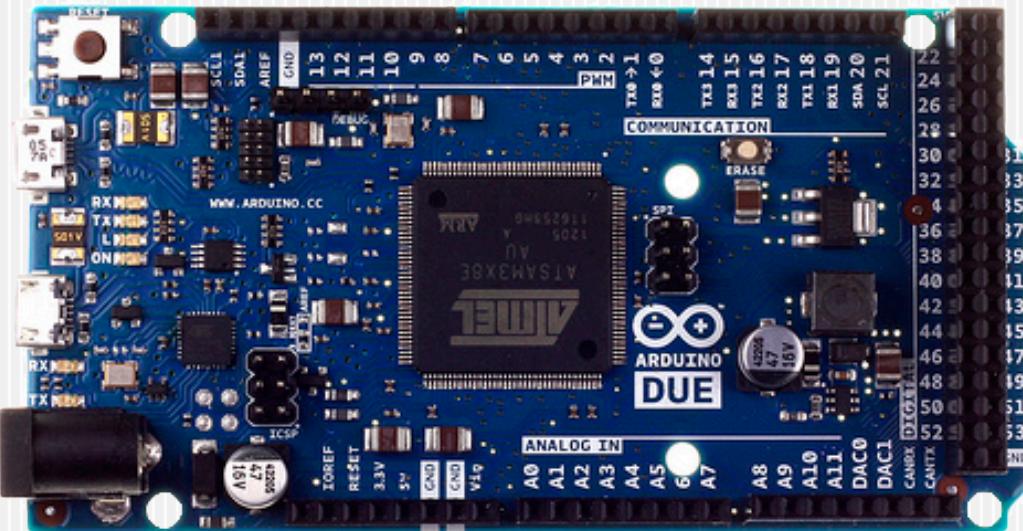


Robótica aplicada  
con Arduino



## Otras características:

- Voltaje de operación 3.3V
  - Voltaje de entrada recomendado: 7-12V
  - Min-Max voltaje de entrada: 6-16V
  - Digital I/O Pins: 54 (de los cuales 12 son PWM)
  - Entradas analógicas: 12
  - Salidas analógicas: 2
  - Corriente de salida por pin: 130 mA
  - Corriente DC para el pin 3.3V: 800 mA
  - Corriente DC para el pin 5V: 800 mA



# Robótica aplicada con Arduino



dfests  
.ua.es

# Índice

→ → →

- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- **Sensores**
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca

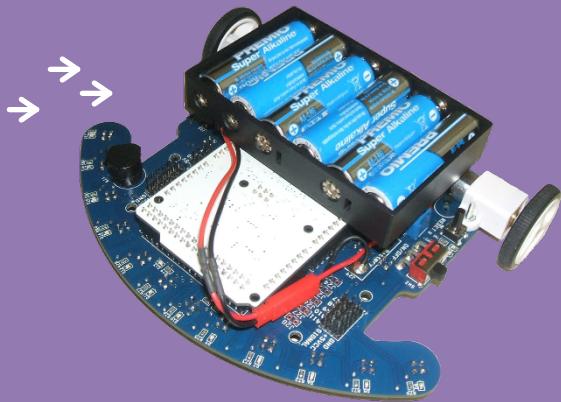


Robótica aplicada  
con Arduino





Robótica aplicada con Arduino



# SENSORES

20142015



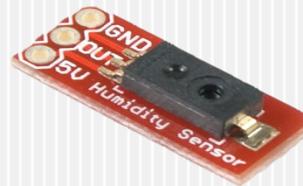


# Sensores

- Sensor de humedad
  - Control de toldos, ventanas, persianas...
  - Riego de plantas
  - Alerta de inundación



Digital  
Consumo 30 µW  
0-100% HR  
Temperatura



Analógico  
Salida Lineal AREF 5V  
Consumo 200 µW  
0-100% HR  
Temperatura

- Sensor de luminosidad
  - Control de luces, ventanas, persianas...
  - Alerta de seguridad



Analógico  
1kOhm - 10kOhm  
0-100% HR  
Temperatura



Comportamiento similar  
a transistor



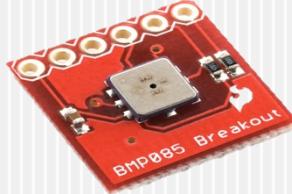
Robótica aplicada  
con Arduino





# Sensores

- Sensor de presión atmosférica
  - Control de toldos, Ventanas, Persianas...
  - Riego de plantas, Aviso meteorológico...



Digital  
Conexión I<sup>2</sup>C  
Bajo consumo  
30KPa - 110KPa  
Temperatura



Digital  
Conexión SPI  
Bajo consumo  
30KPa - 120KPa  
Temperatura

- Sensor de presencia
  - Control de luces, Control de energía, calefacción...
  - Alerta de seguridad
  - Accionamientos mecánicos



Comportamiento similar  
a transistor



Comunicación Serie 9600bps  
Salida analógica 10mV/pulgada  
0-255 niveles (6,45 metros)



Robótica aplicada  
con Arduino





# Sensores

- Sensor de temperatura
  - Control de calefacción, ventanas, persianas,...
  - Alertas de seguridad, Aplicaciones gráficas
  - Accionamiento de sistemas contra hielo
  - Seguimiento temperatura pacientes



Resistencia  
Analógico  
10kOhm



Analógico  
Salida Lineal  $10\text{mV}^{\circ}\text{K}$   
 $2.98\text{V} = 298^{\circ}\text{K} = 25^{\circ}\text{C}$



Digital  
Resolución 12 bits ( $0.065^{\circ}\text{C}$ )  
 $-25^{\circ}\text{C} / 80^{\circ}\text{C}$   
Conexión I<sup>2</sup>C



Atmega + sensor  
Atmega 328P  
Sensor temp. infrarrojo  
Solución pequeño tamaño



Robótica aplicada  
con Arduino





# Sensores

- Sensor biométrico
  - Control de acceso e identificación
  - Alerta y control de intoxicación (calderas)



Analógico  
Detector Monóxido de carbono  
20-2000ppm



Digital  
Comunicación serie  
9600bps  
Devuelve una única id



Analógico  
Detector partículas de humo  
0.5V/0.1mg/m<sup>3</sup>



Robótica aplicada  
con Arduino





# Sensores

- Acelerómetros, giroscopios, brújulas, GPS
  - Anticipación y control del sistema domótico
  - Control automático de dispositivos móviles
  - Localización de personas, Captura de movimientos



Acelerómetro  
Analógico  
Dos ejes  
Salida lineal



Brújula  
Comunicación I<sup>2</sup>C  
Resolución 0.2°



Giroscopio  
Analógico  
Dos ejes  
Salida lineal  
Velocidad angular  
500°/s



Acelerómetro  
Analógico  
Tres ejes  
+/- 3G eje vertical (z)  
Salida lineal

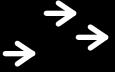
GPS  
Comunicación serie  
NMEA / SIRF  
20 canales  
5m de precisión



Robótica aplicada  
con Arduino



# Índice



- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca



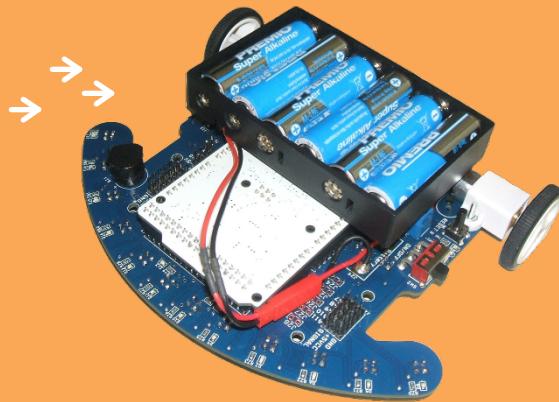
Robótica aplicada  
con Arduino



dfests  
ua.es



Robótica aplicada con Arduino



# PROGRAMACIÓN

ROBOTICA



# Proceso

## ■ Diseñar el circuito:

- ¿Qué requisitos eléctricos necesitan los sensores o actuadores?
- Identificar las entradas (entradas analógicas).
- Identificar las salidas digitales.

## ■ Escribir el código:

- Construirlo de manera incremental:
  - Obtener primero el resultado más sencillo.
  - Añadir complejidad y probar los resultados cada vez.
  - Guardar copias de seguridad con frecuencia.
- Usar variables, no constantes.
- Realizar comentarios.



Robótica aplicada  
con Arduino



# Escribiendo y cargando el código

Escribir el sketch en el PC



Cargar el sketch en el ARDUINO



<http://tech-freaks.net/arduino/instalar-driver-puerto-usb-nativo-en-arduino-due/>

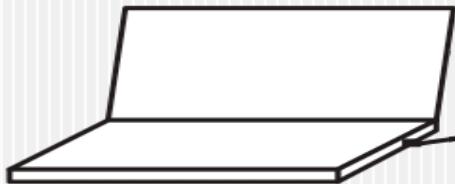


Robótica aplicada  
con Arduino



# Ejecutando el código conectados al ordenador

Ejecuta el sketch en Arduino y envía los datos de vuelta al PC



Arduino interactúa con su entorno



Comunicación serie de vuelta al host

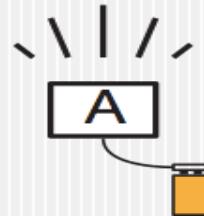


Robótica aplicada con Arduino





# Ejecutando el código de manera autónoma



Arduino interactúa con su entorno y funciona con baterías



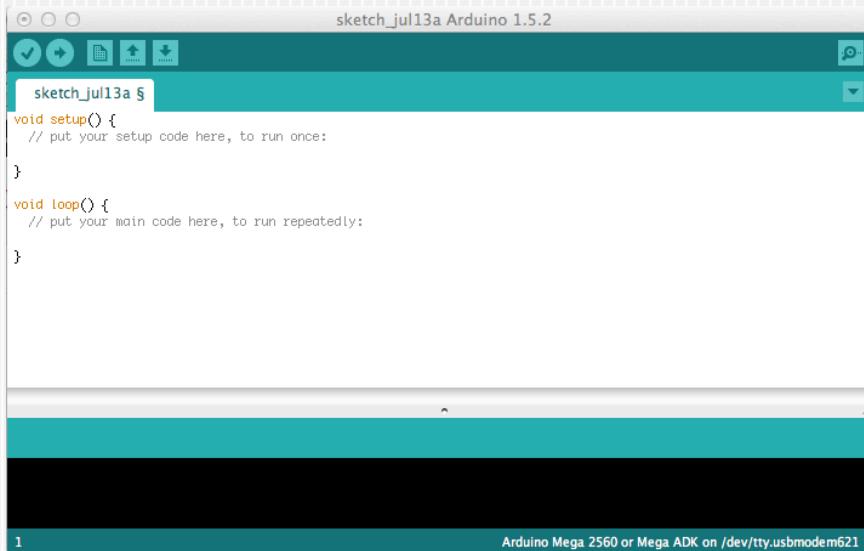
Robótica aplicada  
con Arduino





# Arduino IDE 1.5.2 (beta)

<http://arduino.cc/en/Main/Software>



<http://www.arduino.cc/en/Guide/Environment>

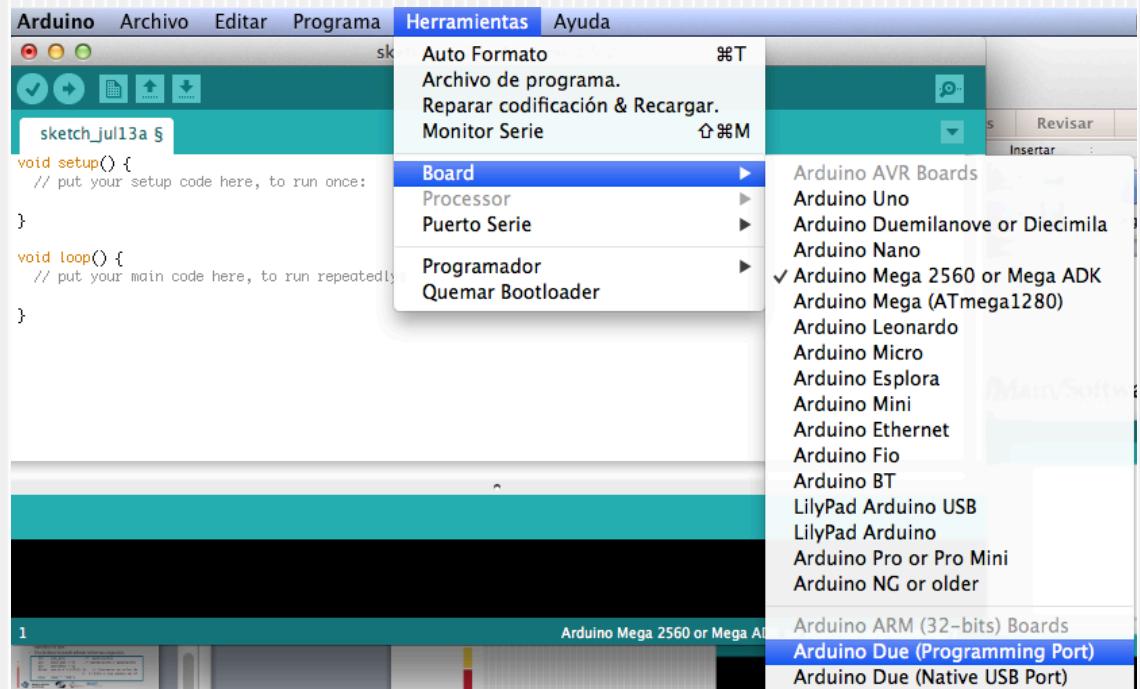


Robótica aplicada  
con Arduino



# Configuración del IDE

- Selección del tipo de placa:
  - Arduino Uno
  - Arduino Duemilanove
  - Arduino Diecimila
  - Arduino Mega
  - Arduino DUE
  - ...
- Selección del puerto serie correcto.



Robótica aplicada  
con Arduino



# Asignando y usando variables

- Definiendo y usando variables:
  - Todas las variables deben declararse antes de su uso.
  - La declaración consiste en indicar el nombre de la variable y especificar su tipo.
  - Una declaración puede además incluir una asignación.

```
int      red_pin;           //  declaración
int      blue_pin = 5;       //  declaración y asignación
int      greenPin = 0;
float   scale = 3.3/1023.0; //  Convierte un valor de
                           //  10 bits a una escala de 3.3V
char    name = 'Bob';
```



Robótica aplicada  
con Arduino



# Variables int

- Un int es un entero con signo de 32 bits
  - Ver: <http://www.arduino.cc/en/Reference/Int>
  - El almacenamiento requiere cuatro bytes.
  - Dividido en un rango positivo y negativo: -2,147,483,648 a 2,147,483,647
  - Los cálculos se redondean y se desbordan (rolled-over) si se supera el límite
- Ejemplos:

```
int sensorVal;          // declaración  
int sensorPin = 3;      // declaración y asignación
```

```
sensorVal = analogRead(sensorPin); // asignación
```



Robótica aplicada  
con Arduino



# Variables float

- Un float es un número con signo y con una parte fraccional

- Ver: <http://www.arduino.cc/en/Reference/Float>
- Se almacena en 32 bits
- Rango:  $-3.4028235 \times 10^{38}$  a  $3.4028235 \times 10^{38}$
- La aritmética de punto flotante introduce redondeos

- Ejemplos:

```
int sensorVal; // valor devuelto de una entrada analógica input
int sensorPin = 3; // pin asignado a una entrada analógica
int maxInput = 1023; // Salida máxima del A/D de 10 bit
float voltage; // Voltaje de la señal de entrada
float maxVoltage = 5.0; // Maximum range of analog input

sensorVal = analogRead(sensorPin); // obtener la lectura
voltage = sensorVal*maxVoltage/maxInput; // convertir a voltaje
voltage = ((float)sensorVal)*maxVoltage/((float)maxInput);
```



Robótica aplicada  
con Arduino



# ■ Referencia del lenguaje

- Basado en C/C++
- Vincula la librería AVR Libc

## Estructura

- `setup()`
  - Al inicio de un sketch
  - Inicia variables, estado de pins
  - inicio de librerías, ...
- `loop()`
  - Función principal del sketch.
  - Ejecución indefinida



Robótica aplicada  
con Arduino



Máster Oficial en  
**Automática y Robótica**  
Universidad de Alicante





# Referencia del lenguaje

## ■ Estructuras de control

- if
- if...else
- for
- while
- do...while
- switch case
- break
- continue
- return



Robótica aplicada  
con Arduino





# Referencia del lenguaje

## ■ Más sintaxis

- ; (punto y coma)
- {} (llaves)
- // (comentarios de una sola línea)
- /\* \*/ (comentarios multilínea)
- #define (definición de precompilador)
- #include (inclusión de código externo)

## ■ Operadores aritméticos

- = (operador de asignación)
- + (suma)
- - (resta)
- \* (multiplicación)
- / (división)
- % (módulo)



Robótica aplicada  
con Arduino



# Referencia del lenguaje

- Operadores de comparación
  - == (igual que)
  - != (distinto que)
  - < (menor que)
  - > (mayor que)
  - <= (menor o igual que)
  - >= (mayor o igual que)
- Operadores a nivel de bits
  - & ('y' a nivel de bits)
  - | ('o' a nivel de bits)
  - ^ (xor a nivel de bits)
  - ~ (not a nivel de bits)
  - << (desplazamiento de bits a la izquierda)
  - >> (desplazamiento de bits a la derecha)



Robótica aplicada  
con Arduino



# Referencia del lenguaje

- Operadores booleanos
  - `&&` ('y' lógico)
  - `||` ('o' lógico)
  - `!` (negación lógica)
- Operadores compuestos
  - `++` (incremento)
  - `--` (decremento)
  - `+=` (suma compuesta)
  - `-=` (resta compuesta)
- Operadores de acceso a punteros
  - `*` operador de indirección
  - `&` acceso a memoria
- `*=` (multiplicación compuesta)
- `/=` (división compuesta)
- `&=` ('y' a nivel de bits compuesto)
- `|=` ('o' a nivel de bits compuesto)



Robótica aplicada  
con Arduino





# Referencia del lenguaje

- <http://arduino.cc/en/Reference/HomePage>



Robótica aplicada  
con Arduino



# Índice

→ → →

- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca

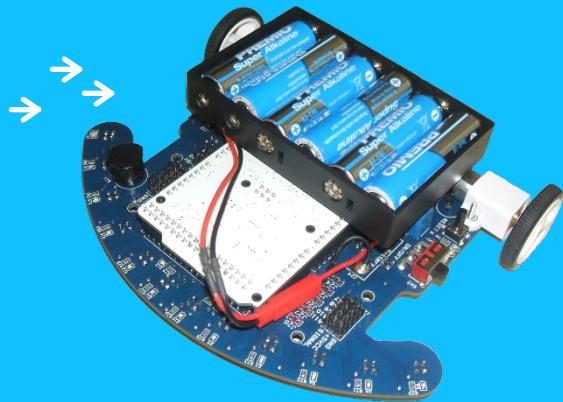


Robótica aplicada  
con Arduino





Robótica aplicada con Arduino



# COMUNICACIÓN

COMUNICACIÓN

# Comunicación en ARDUINO

- Comunicación por serie:
  - Se utiliza para la comunicación entre el ARDUINO y un ordenador u otros dispositivos.
  - Se comunica a través de los pines digitales 0 (RX) y 1 (TX), así como con el ordenador mediante USB.
  - Si se utilizan estas funciones, no puedes usar los pines 0 y 1 como entrada o salida digital.
  - Se puede utilizar el monitor del puerto serie incorporado en el entorno Arduino para comunicarse con la placa ARDUINO.
  - El ARDUINO DUE tiene tres puertos adicionales de serie: **Serial1** en los pines 19 (RX) y 18 (TX), **Serial2** en los pines 17 (RX) y 16 (TX), **Serial3** en los pines 15 (RX) y 14 (TX).
  - No se debe conectar estos pines directamente a un puerto serie RS232, que operan a +/- 12V ya que esto puede dañar el ARDUINO.



Robótica aplicada  
con Arduino





# Comunicación en ARDUINO

## Ejemplo de comunicación serie:

```
int incomingByte;  
  
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    if (Serial.available() > 0) {  
        incomingByte = Serial.read();  
        if (incomingByte == 'A') {  
            Serial.println("Se ha escrito una A");  
        }  
    }  
}
```



Robótica aplicada  
con Arduino



## Funciones:

- [begin\(\)](#)
- [end\(\)](#)
- [available\(\)](#)
- [read\(\)](#)
- [flush\(\)](#)
- [print\(\)](#)
- [println\(\)](#)
- [write\(\)](#)

# Comunicación en ARDUINO

- Comunicación por I2C:
  - I2C es un bus de comunicaciones en serie.
  - Diseñado por Philips.
  - Velocidad de 100kbits/s hasta 3.4Mbits/s.
  - Se dispone de 7 bits para el direccionamiento de cada dispositivo conectado al bus. Como se tienen 16 direcciones reservadas se pueden conectar hasta 112 dispositivos.
  - Es un bus muy utilizado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas empotrados.
  - Utiliza dos líneas para transmitir la información: una para los datos (SDA) y por otra la señal de reloj (SCL).
  - Los dispositivos conectados al bus I2C tienen una dirección única para cada uno.
  - El dispositivo *maestro* inicia la transferencia de datos y además genera la señal de reloj.



Robótica aplicada  
con Arduino



# Comunicación en ARDUINO

## Funciones:

- [begin\(\)](#)
- [begin\(address\)](#)
- [requestFrom\(address, count\)](#)
- [beginTransmission\(address\)](#)
- [endTransmission\(\)](#)
- [send\(\)](#)
- byte [available\(\)](#)
- byte [receive\(\)](#)
- [onReceive\(handler\)](#)
- [onRequest\(handler\)](#)



Robótica aplicada  
con Arduino



## Blink (maestro):

```
#include <Wire.h>
int ledPin = 13;
void setup() {
    pinMode(ledPin, OUTPUT);
    Wire.begin();
}
```

```
void loop() {
    digitalWrite(ledPin, HIGH);
    Wire.beginTransmission(4);
    Wire.send(HIGH);
    Wire.endTransmission();
    delay(1000);
    digitalWrite(ledPin, LOW);
    Wire.beginTransmission(4);
    Wire.send(LOW);
    Wire.endTransmission();
    delay(1000);
}
```



dfests  
.ua.es

# Comunicación en ARDUINO

## Blink (esclavo):

```
#include <Wire.h>

int ledPin = 13;

void setup() {
    pinMode(ledPin, OUTPUT);
    Wire.begin(4);
    Wire.onReceive(receiveEvent);
}

void loop() {
    delay(100);
}

void receiveEvent(int howMany) {
    digitalWrite(ledPin, Wire.receive());
}
```



Robótica aplicada  
con Arduino



# Índice

→ → →

- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca

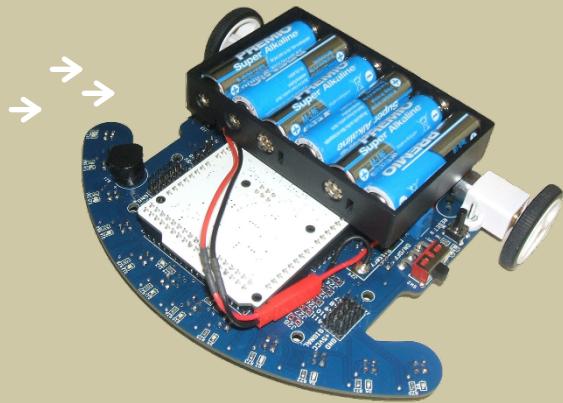


Robótica aplicada  
con Arduino





Robótica aplicada con Arduino



# INTERRUPCIONES

INTERRUPCIONES



# Interrupciones

- Una interrupción hardware causa que el procesador salve su estado de ejecución y comience la ejecución de una rutina de interrupción.
- Resuelve el problema del “polling” (sondeo o muestreo), que provoca excesiva carga de trabajo al procesador.



Robótica aplicada  
con Arduino



# Interrupciones en ARDUINO

Las interrupciones en el ARDUINO DUE se pueden implementar en cualquier pin.

- `attachInterrupt(pin, function, mode)`
  - pin: número de pin donde se quiere capturar la interrupción.
  - function: nombre de la rutina. La rutina no debe tener parámetros ni devolver nada.
  - mode: indica cuándo se debe lanzar la interrupción:
    - **LOW** para lanzarla siempre que el pin esté en 0,
    - **CHANGE** para lanzarla siempre que el pin cambie de valor,
    - **RISING** para lanzar la interrupción cuando el pin pasa de 0 a 1,
    - **FALLING** para cuando el pin pasa de 1 a 0.



Robótica aplicada  
con Arduino



# ■ Interrupciones en ARDUINO

- Dentro de la rutina de interrupción:
  - No funciona la función delay(), sí que lo hace la función delayMicroseconds().
  - Tampoco se incrementa la función millis().
- Los datos recibidos por el puerto serie mientras se está en la rutina se pueden perder.
- Cualquier variable que se modifique dentro de la rutina se debe declarar como *volatile*.



Robótica aplicada  
con Arduino



# ■ Interrupciones en ARDUINO

- Otras funciones:
  - detachInterrupt(interrupt)
    - interrupt: el número de la interrupción a deshabilitar.
  - noInterrupts()
    - Deshabilita todas las interrupciones de manera temporal.
  - interrupts()
    - Habilita todas las interrupciones.



Robótica aplicada  
con Arduino



# Índice



- Lenguaje C a bajo nivel
- Introducción a ARDUINO
- Componentes del ARDUINO
- Sensores
- Programación
- Comunicación:
  - Serie
  - I2C
- Interrupciones
- Ejercicios:
  - Blink
  - El coche fantástico
  - Cruce de semáforos
  - Aumentar luminosidad de led con pulsador
  - Código Morse
  - Ruleta loca



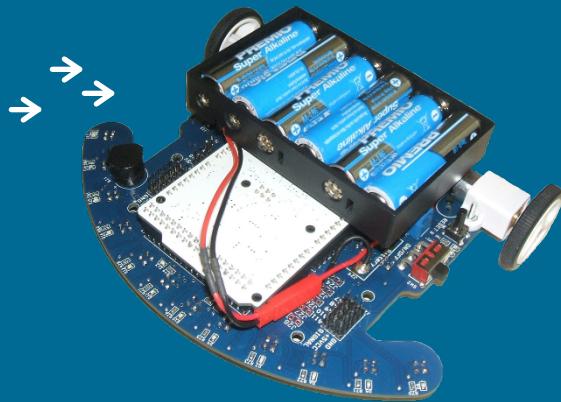
Robótica aplicada  
con Arduino



dfests  
ua.es



Robótica aplicada con Arduino



# EJERCICIOS

EJERCICIOS



# Blink

- Programa “Hola mundo” de ARDUINO.
- Objetivo: Hacer parpadear el LED verde del shield GR conectado a la salida digital 34.



Robótica aplicada  
con Arduino



# ||| Blink: sketch

```
void setup() {  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(1000);  
}
```



Robótica aplicada  
con Arduino

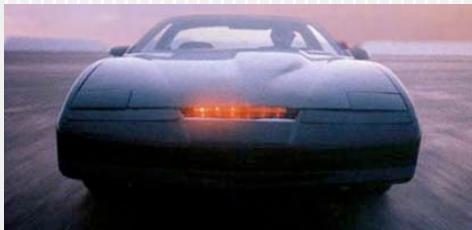


Máster Oficial en  
Automática y Robótica  
Universidad de Alicante



# El coche fantástico

- Programación de bucles de control y salidas digitales.
- Objetivo: Emular con 6 LED el frontal del coche fantástico.



Robótica aplicada  
con Arduino



- Utilizar los leds conectados al shield de GR:
  - D4 a D17 (excluyendo los azules D8 y D13).
  - D4 conectado a pin 52.
  - D5 conectado a pin 50.
  - D6 conectado a pin 48.
  - D7 conectado a pin 46.
  - D9 conectado a pin 42.
  - D10 conectado a pin 36.
  - D11 conectado a pin 34.
  - D12 conectado a pin 32.
  - D14 conectado a pin 28.
  - D14 conectado a pin 26.
  - D14 conectado a pin 24.



# El coche fantástico: sketch

```
void loop() {  
    for (int thisPin = 2; thisPin < 8; thisPin++) {  
        digitalWrite(thisPin, HIGH);  
        delay(timer);  
        digitalWrite(thisPin, LOW);  
    }  
  
    for (int thisPin = 7; thisPin >= 2; thisPin--) {  
        digitalWrite(thisPin, HIGH);  
        delay(timer);  
        digitalWrite(thisPin, LOW);  
    }  
}
```



Robótica aplicada  
con Arduino



```
int timer = 100;
```

```
void setup() {  
    for (int thisPin = 2; thisPin < 7; thisPin++) {  
        pinMode(thisPin, OUTPUT);  
    }  
}
```

dfests  
.ua.es



# Cruce de semáforos

- Se trata de un cruce de semáforos controlado por arduino, para ello utilizaremos en el primer semáforo los pines 52 (led rojo), 48 (led ambar), 42 (led verde), en el segundo semáforo utilizaremos los pines 22 (led rojo), 26 (led ambar) y 32 (led verde). La secuencia de funcionamiento debe ser : rojo 1 – verde 2 durante 3 segundos, rojo 1 – ambar 2 durante 500 ms, verde 1 – rojo 2 durante 3 segundos, ambar 1 - , rojo 2 durante 500 ms.



Robótica aplicada  
con Arduino



# Código Morse

- Se trata de un zumbador que en código morse (pitidos largos/cortos) especifica una palabra, en nuestro caso SOS. Para el que no lo sepa, la S son tres señales acústicas de corta duración y la O tres señales acústicas de larga duración.
- El zumbador está conectado en el shield de GR al pin 23, los pitidos cortos tendrán una duración de 100 ms y los largos 300 ms. Entre letra y letra debe pasar un tiempo de 300 ms y entre SOFs debe haber un tiempo de 1000 ms.
- Nota: Debes usar variables para guardar los tiempos que vas a usar.



Robótica aplicada  
con Arduino





# Aumentar luminosidad de led con pulsador

- Se trata de aumentar la luminosidad de un diodo led conectado a una salida PWM a través de la activación de un pulsador. El pulsador estará conectado al pin 13. Mientras el pulsador está conectado aumenta la luminosidad del led hasta llegar a su valor máximo (255), si el pulsador se desactiva se mantendrá su luminosidad hasta que el valor de luminosidad llegue a su máximo (255) pulsando nuevas veces, si esto ocurre la luminosidad pasará a valor nulo (0).



Robótica aplicada  
con Arduino



# Rueda loca

- Se trata de utilizar los leds del frontal del shield de GR de forma que se vayan encendiendo y apagando formando una secuencia. El jugador debe dar al pulsador cuando el led intermedio (D11 en pin 34) se enciende. Si acierta funciona el zumbador y la velocidad de la secuencia aumenta.

Los leds están conectados tal y como se comentó en el ejercicio del coche fantástico. El zumbador al pin , el pulsador al pin 23. El tiempo inicial entre encendido y encendido de leds debe ser 200 ms, si se acierta se decrementa el tiempo en 20 ms, si el tiempo entre encendidos llegase a 10 ms, se activa el zumbador durante 3 veces el tiempo normal de funcionamiento establecido para un acierto y se devuelve el tiempo a 200 ms.



Robótica aplicada  
con Arduino





# Páginas de interés

- <http://arduino.cc/en>
- <http://processing.org>
- <http://wiring.org.co/>
- <http://www.creativeapplications.net>
- <http://www.instructables.com/>
- <http://www.ladyada.net/learn/arduino/lesson2.html>
- [http://web.media.mit.edu/~leah/LilyPad/03\\_arduino\\_intro.html](http://web.media.mit.edu/~leah/LilyPad/03_arduino_intro.html)
- ...



Robótica aplicada  
con Arduino





Robótica aplicada con Arduino



# ROBÓTICA APLICADA CON ARDUINO

15 JULIO 2013

Programación en ARDUINO

Gabriel J. García Gómez