

Introducción a Arduino

Fernando Cerezal López

9 de Noviembre de 2012

- Plataforma
 - Estándar
 - Electrónica
 - IDE
- Conceptos básicos electrónica
 - Ley de Ohm y efecto Joule
 - Ruido, puntos sin conexión y efecto rebote
 - Semiconductores
- Conceptos básicos programación
 - Funciones y variables
 - Programación en micros
 - Interrupciones (internas y externas)

- **Plataforma**
 - Estándar
 - Electrónica
 - IDE
- **Conceptos básicos electrónica**
 - Ley de Ohm y efecto Joule
 - Ruido, puntos sin conexión y efecto rebote
 - Semiconductores
- **Conceptos básicos programación**
 - Funciones y variables
 - Programación en micros
 - Interrupciones (internas y externas)

- Plataforma
 - Estándar
 - Electrónica
 - IDE
- Conceptos básicos electrónica
 - Ley de Ohm y efecto Joule
 - Ruido, puntos sin conexión y efecto rebote
 - Semiconductores
- Conceptos básicos programación
 - Funciones y variables
 - Programación en micros
 - Interrupciones (internas y externas)

- Plataforma
 - Estándar
 - Electrónica
 - IDE
- Conceptos básicos electrónica
 - Ley de Ohm y efecto Joule
 - Ruido, puntos sin conexión y efecto rebote
 - Semiconductores
- Conceptos básicos programación
 - Funciones y variables
 - Programación en micros
 - Interrupciones (internas y externas)

- Plataforma
 - Estándar
 - Electrónica
 - IDE
- Conceptos básicos electrónica
 - Ley de Ohm y efecto Joule
 - Ruido, puntos sin conexión y efecto rebote
 - Semiconductores
- Conceptos básicos programación
 - Funciones y variables
 - Programación en micros
 - Interrupciones (internas y externas)

- μ Cs y dispositivos
 - Pinout, memoria y velocidad
 - GPIO
 - ADC (y DAC)
 - PWM
 - Timers
 - Buses: UART, SPI, I2C
- API Arduino
 - Repaso de la API
- Uso del IDE
 - Configuración: placa y directorio de trabajo
 - Formato de un programa: setup y loop
 - Añadir una biblioteca
 - Verificación y carga de un programa

- μ Cs y dispositivos
 - Pinout, memoria y velocidad
 - GPIO
 - ADC (y DAC)
 - PWM
 - Timers
 - Buses: UART, SPI, I2C
- API Arduino
 - Repaso de la API
- Uso del IDE
 - Configuración: placa y directorio de trabajo
 - Formato de un programa: setup y loop
 - Añadir una biblioteca
 - Verificación y carga de un programa

- μ Cs y dispositivos
 - Pinout, memoria y velocidad
 - GPIO
 - ADC (y DAC)
 - PWM
 - Timers
 - Buses: UART, SPI, I2C
- API Arduino
 - Repaso de la API
- Uso del IDE
 - Configuración: placa y directorio de trabajo
 - Formato de un programa: setup y loop
 - Añadir una biblioteca
 - Verificación y carga de un programa

- μ Cs y dispositivos
 - Pinout, memoria y velocidad
 - GPIO
 - ADC (y DAC)
 - PWM
 - Timers
 - Buses: UART, SPI, I2C
- API Arduino
 - Repaso de la API
- Uso del IDE
 - Configuración: placa y directorio de trabajo
 - Formato de un programa: setup y loop
 - Añadir una biblioteca
 - Verificación y carga de un programa

- μ Cs y dispositivos
 - Pinout, memoria y velocidad
 - GPIO
 - ADC (y DAC)
 - PWM
 - Timers
 - Buses: UART, SPI, I2C
- API Arduino
 - Repaso de la API
- Uso del IDE
 - Configuración: placa y directorio de trabajo
 - Formato de un programa: setup y loop
 - Añadir una biblioteca
 - Verificación y carga de un programa

Plataforma

¿Por qué nace Arduino?

- Multitud de kits de prototipado.
- Cada kit con su propio IDE propietario.
- Placas con esquemáticos cerrados.
- Cada micro con capacidades distintas.
- Cada micro con bibliotecas y lenguajes distintas.

¿Por qué nace Arduino?

- Multitud de kits de prototipado.
- Cada kit con su propio IDE propietario.
- Placas con esquemáticos cerrados.
- Cada micro con capacidades distintas.
- Cada micro con bibliotecas y lenguajes distintas.

¿Por qué nace Arduino?

- Multitud de kits de prototipado.
- Cada kit con su propio IDE propietario.
- Placas con esquemáticos cerrados.
- Cada micro con capacidades distintas.
- Cada micro con bibliotecas y lenguajes distintas.

¿Por qué nace Arduino?

- Multitud de kits de prototipado.
- Cada kit con su propio IDE propietario.
- Placas con esquemáticos cerrados.
- Cada micro con capacidades distintas.
- Cada micro con bibliotecas y lenguajes distintas.

¿Por qué nace Arduino?

- Multitud de kits de prototipado.
- Cada kit con su propio IDE propietario.
- Placas con esquemáticos cerrados.
- Cada micro con capacidades distintas.
- Cada micro con bibliotecas y lenguajes distintas.

¿Por qué nace Arduino?

- Multitud de kits de prototipado.
- Cada kit con su propio IDE propietario.
- Placas con esquemáticos cerrados.
- Cada micro con capacidades distintas.
- Cada micro con bibliotecas y lenguajes distintas.

¿Qué pretende Arduino?

- Estandarizar kits de prototipado.
- Kits de prototipado baratos.
- Hacer sencillo programar micros.

¿Qué pretende Arduino?

- Estandarizar kits de prototipado.
- Kits de prototipado baratos.
- Hacer sencillo programar micros.

¿Qué pretende Arduino?

- Estandarizar kits de prototipado.
- Kits de prototipado baratos.
- Hacer sencillo programar micros.

¿Qué pretende Arduino?

- Estandarizar kits de prototipado.
- Kits de prototipado baratos.
- Hacer sencillo programar micros.

¿Qué es Arduino?

- Placas electrónicas: Pinout estable y esquemático libre. Basado en micros de Atmel.
- IDE (mostrar IDE)
- Bibliotecas
- Lenguaje y API

¿Qué es Arduino?

- Placas electrónicas: Pinout estable y esquemático libre. Basado en micros de Atmel.
- IDE (mostrar IDE)
- Bibliotecas
- Lenguaje y API

¿Qué es Arduino?

- Placas electrónicas: Pinout estable y esquemático libre. Basado en micros de Atmel.
- IDE (mostrar IDE)
- Bibliotecas
- Lenguaje y API

¿Qué es Arduino?

- Placas electrónicas: Pinout estable y esquemático libre. Basado en micros de Atmel.
- IDE (mostrar IDE)
- Bibliotecas
- Lenguaje y API

¿Qué es Arduino?

- Placas electrónicas: Pinout estable y esquemático libre. Basado en micros de Atmel.
- IDE (mostrar IDE)
- Bibliotecas
- Lenguaje y API

Electrónica básica

Ley de Ohm y Efecto Joule

- Ley de Ohm: $I = V/R$.
- La tensión está, no se mueve. La corriente se mueve.
- Efecto Joule: $W = I * R^2 = I * V$
- Los componentes se calientan al pasar corriente por ellos.
- Cuidado con las resistencias pequeñas o las tensiones altas.
- Hay corrientes y tensiones límite antes de ruptura.

Ley de Ohm y Efecto Joule

- Ley de Ohm: $I = V/R$.
- La tensión está, no se mueve. La corriente se mueve.
- Efecto Joule: $W = I * R^2 = I * V$
- Los componentes se calientan al pasar corriente por ellos.
- Cuidado con las resistencias pequeñas o las tensiones altas.
- Hay corrientes y tensiones límite antes de ruptura.

Ley de Ohm y Efecto Joule

- Ley de Ohm: $I = V/R$.
- La tensión está, no se mueve. La corriente se mueve.
- Efecto Joule: $W = I * R^2 = I * V$
- Los componentes se calientan al pasar corriente por ellos.
- Cuidado con las resistencias pequeñas o las tensiones altas.
- Hay corrientes y tensiones límite antes de ruptura.

Ley de Ohm y Efecto Joule

- Ley de Ohm: $I = V/R$.
- La tensión está, no se mueve. La corriente se mueve.
- Efecto Joule: $W = I * R^2 = I * V$
- Los componentes se calientan al pasar corriente por ellos.
- Cuidado con las resistencias pequeñas o las tensiones altas.
- Hay corrientes y tensiones límite antes de ruptura.

Ley de Ohm y Efecto Joule

- Ley de Ohm: $I = V/R$.
- La tensión está, no se mueve. La corriente se mueve.
- Efecto Joule: $W = I * R^2 = I * V$
- Los componentes se calientan al pasar corriente por ellos.
- Cuidado con las resistencias pequeñas o las tensiones altas.
- Hay corrientes y tensiones límite antes de ruptura.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25mv$ a $25^{\circ}C$
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a $V+$ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25mv$ a $25^{\circ}C$
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a $V+$ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25mv$ a $25^{\circ}C$
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a $V+$ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25mv$ a $25^{\circ}C$
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a $V+$ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25mv$ a $25^{\circ}C$
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a $V+$ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25mv$ a $25^{\circ}C$
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a $V+$ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Ruido, puntos sin conexión y efecto rebote

Ruido EM

- Sólo a tener en cuenta si usamos sensores muy sensibles.
 $V_t = 25\text{mv}$ a 25°C
- O cerca hay aparatos muy potentes. (Señal a 50 Hz.)

Puntos sin conexión

- Una patilla a V_+ es 1 y una a GND es 0
- Una patilla al aire es indeterminado.

Efecto rebote

- La naturaleza no permite cambios abruptos.
- Cualquier contacto mecánico presenta efecto rebote.

Semiconductores: LEDs, diodos, transistores

- Están polarizados.
- Presentan una caída de tensión fija en sus puntos de unión.
- Cuidado, hay que absorber la tensión "sobrante".

Semiconductores: LEDs, diodos, transistores

- Están polarizados.
- Presentan una caída de tensión fija en sus puntos de unión.
- Cuidado, hay que absorber la tensión "sobrante".

Semiconductores: LEDs, diodos, transistores

- Están polarizados.
- Presentan una caída de tensión fija en sus puntos de unión.
- Cuidado, hay que absorber la tensión "sobrante".

Semiconductores: LEDs, diodos, transistores

- Están polarizados.
- Presentan una caída de tensión fija en sus puntos de unión.
- Cuidado, hay que absorber la tensión "sobrante".

Programación

Programación básica

- Variable = zona de memoria donde se almacena un valor que puede cambiar.
- Función = conjunto de órdenes que transforman unos datos de entrada en datos de salida.
- Arduino usa algo parecido a C++.

Ejemplo

```
int a=3;
int b=2;
int c = multiplicar(a,b);

int multiplicar(a,b){
    //Los micros sí pueden multiplicar, pero esto es más gráfico
    for(int c=0; c < b; c++){
        a+=a;
    }
    return a;
}
```

Programación en micros

- Nuestra función main es la única que se ejecuta. Debe ser un bucle infinito (o parecido).
- Hay que inicializar los dispositivos al arrancar.
- Nunca da error.
- Arduino nos quita esto de encima.

Programación en micros

- Nuestra función main es la única que se ejecuta. Debe ser un bucle infinito (o parecido).
- Hay que inicializar los dispositivos al arrancar.
- Nunca da error.
- Arduino nos quita esto de encima.

Programación en micros

- Nuestra función main es la única que se ejecuta. Debe ser un bucle infinito (o parecido).
- Hay que inicializar los dispositivos al arrancar.
- Nunca da error.
- Arduino nos quita esto de encima.

Programación en micros

- Nuestra función main es la única que se ejecuta. Debe ser un bucle infinito (o parecido).
- Hay que inicializar los dispositivos al arrancar.
- Nunca da error.
- Arduino nos quita esto de encima.

Programación en micros

- Nuestra función main es la única que se ejecuta. Debe ser un bucle infinito (o parecido).
- Hay que inicializar los dispositivos al arrancar.
- Nunca da error.
- Arduino nos quita esto de encima.

Interrupciones

- Una interrupción es algo que necesita atención en el momento: puerto, botón, etc. . .
- Una interrupción cambia el contador de programa, de forma que lo siguiente a ejecutar es la función de la interrupción.
- La rutinas de interrupción tienen que ser pequeñas.
- El bucle principal debe comprobar si han saltado.

Interrupciones

- Una interrupción es algo que necesita atención en el momento: puerto, botón, etc. . .
- Una interrupción cambia el contador de programa, de forma que lo siguiente a ejecutar es la función de la interrupción.
- La rutinas de interrupción tienen que ser pequeñas.
- El bucle principal debe comprobar si han saltado.

Interrupciones

- Una interrupción es algo que necesita atención en el momento: puerto, botón, etc. . .
- Una interrupción cambia el contador de programa, de forma que lo siguiente a ejecutar es la función de la interrupción.
- Las rutinas de interrupción tienen que ser pequeñas.
- El bucle principal debe comprobar si han saltado.

Interrupciones

- Una interrupción es algo que necesita atención en el momento: puerto, botón, etc. . .
- Una interrupción cambia el contador de programa, de forma que lo siguiente a ejecutar es la función de la interrupción.
- Las rutinas de interrupción tienen que ser pequeñas.
- El bucle principal debe comprobar si han saltado.

Interrupciones

Interrupciones internas

- Una interrupción interna sucede cuando un dispositivo interno la lanza: un timer salta, un puerto recibe o envía datos, etc. . .
- Las gestiona Arduino internamente.

Interrupciones externas

- Una interrupción externa sucede cuando cambia la tensión en una patilla.
- Sólo algunas patillas permiten usarse como interrupciones externas: Interrupciones

Programación de interrupciones

- Cuidado con las variables dentro de las interrupciones: declaradlas Volatile

Interrupciones

Interrupciones internas

- Una interrupción interna sucede cuando un dispositivo interno la lanza: un timer salta, un puerto recibe o envía datos, etc. . .
- Las gestiona Arduino internamente.

Interrupciones externas

- Una interrupción externa sucede cuando cambia la tensión en una patilla.
- Sólo algunas patillas permiten usarse como interrupciones externas: Interrupciones

Programación de interrupciones

- Cuidado con las variables dentro de las interrupciones: declaradlas Volatile

Interrupciones

Interrupciones internas

- Una interrupción interna sucede cuando un dispositivo interno la lanza: un timer salta, un puerto recibe o envía datos, etc. . .
- Las gestiona Arduino internamente.

Interrupciones externas

- Una interrupción externa sucede cuando cambia la tensión en una patilla.
- Sólo algunas patillas permiten usarse como interrupciones externas: Interrupciones

Programación de interrupciones

- Cuidado con las variables dentro de las interrupciones: declaradlas Volatile

Interrupciones

Interrupciones internas

- Una interrupción interna sucede cuando un dispositivo interno la lanza: un timer salta, un puerto recibe o envía datos, etc. . .
- Las gestiona Arduino internamente.

Interrupciones externas

- Una interrupción externa sucede cuando cambia la tensión en una patilla.
- Sólo algunas patillas permiten usarse como interrupciones externas: Interrupciones

Programación de interrupciones

- Cuidado con las variables dentro de las interrupciones: declaradlas Volatile

Interrupciones

Interrupciones internas

- Una interrupción interna sucede cuando un dispositivo interno la lanza: un timer salta, un puerto recibe o envía datos, etc. . .
- Las gestiona Arduino internamente.

Interrupciones externas

- Una interrupción externa sucede cuando cambia la tensión en una patilla.
- Sólo algunas patillas permiten usarse como interrupciones externas: Interrupciones

Programación de interrupciones

- Cuidado con las variables dentro de las interrupciones: declaradlas Volatile

Interrupciones

Interrupciones internas

- Una interrupción interna sucede cuando un dispositivo interno la lanza: un timer salta, un puerto recibe o envía datos, etc. . .
- Las gestiona Arduino internamente.

Interrupciones externas

- Una interrupción externa sucede cuando cambia la tensión en una patilla.
- Sólo algunas patillas permiten usarse como interrupciones externas: Interrupciones

Programación de interrupciones

- Cuidado con las variables dentro de las interrupciones: declaradlas Volatile

μ Cs y dispositivos

¿Qué es un microcontrolador (μ C)?

¿Qué es un microcontrolador (μ C)?

- Un microprocesador con una serie de periféricos a su alrededor
- No hay S.O., sólo nuestro programa.
- El programa se guarda en una FLASH
- Sólo hace operaciones "básicas".
- Sólo hay enteros (en el μ C, no en Arduino) $\rightarrow$ map en la API

¿Qué es un microcontrolador (μ C)?

¿Qué es un microcontrolador (μ C)?

- Un microprocesador con una serie de periféricos a su alrededor
- No hay S.O., sólo nuestro programa.
- El programa se guarda en una FLASH
- Sólo hace operaciones "básicas".
- Sólo hay enteros (en el μ C, no en Arduino) → map en la API

¿Qué es un microcontrolador (μ C)?

¿Qué es un microcontrolador (μ C)?

- Un microprocesador con una serie de periféricos a su alrededor
- No hay S.O., sólo nuestro programa.
- El programa se guarda en una FLASH
- Sólo hace operaciones "básicas".
- Sólo hay enteros (en el μ C, no en Arduino) → map en la API

¿Qué es un microcontrolador (μ C)?

¿Qué es un microcontrolador (μ C)?

- Un microprocesador con una serie de periféricos a su alrededor
- No hay S.O., sólo nuestro programa.
- El programa se guarda en una FLASH
- Sólo hace operaciones "básicas".
- Sólo hay enteros (en el μ C, no en Arduino) → map en la API

¿Qué es un microcontrolador (μ C)?

¿Qué es un microcontrolador (μ C)?

- Un microprocesador con una serie de periféricos a su alrededor
- No hay S.O., sólo nuestro programa.
- El programa se guarda en una FLASH
- Sólo hace operaciones "básicas".
- Sólo hay enteros (en el μ C, no en Arduino) → map en la API

¿Qué caracteriza un μ C?

¿Qué caracteriza un μ C? (Básicamente)

- Velocidad del μ C.
- Tamaño memorias: RAM, EEPROM, FLASH
- Dispositivos
- Pinout
- Atmega2560

¿Qué caracteriza un μC ?

¿Qué caracteriza un μC ? (Básicamente)

- Velocidad del μC .
- Tamaño memorias: RAM, EEPROM, FLASH
- Dispositivos
- Pinout
- Atmega2560

¿Qué caracteriza un μ C?

¿Qué caracteriza un μ C? (Básicamente)

- Velocidad del μ C.
- Tamaño memorias: RAM, EEPROM, FLASH
- Dispositivos
- Pinout
- Atmega2560

¿Qué caracteriza un μC ?

¿Qué caracteriza un μC ? (Básicamente)

- Velocidad del μC .
- Tamaño memorias: RAM, EEPROM, FLASH
- Dispositivos
- Pinout
- Atmega2560

¿Qué caracteriza un μC ?

¿Qué caracteriza un μC ? (Básicamente)

- Velocidad del μC .
- Tamaño memorias: RAM, EEPROM, FLASH
- Dispositivos
- Pinout
- Atmega2560

GPIO: General Purpose Input Output

- Todas las patillas pueden usarse como entradas o salidas digitales.
- Hay que configurar al principio si son entradas o salidas.
- API

GPIO: General Purpose Input Output

- Todas las patillas pueden usarse como entradas o salidas digitales.
- Hay que configurar al principio si son entradas o salidas.
- API

GPIO: General Purpose Input Output

- Todas las patillas pueden usarse como entradas o salidas digitales.
- Hay que configurar al principio si son entradas o salidas.
- API

GPIO: General Purpose Input Output

- Todas las patillas pueden usarse como entradas o salidas digitales.
- Hay que configurar al principio si son entradas o salidas.
- API

ADC: Analogic Digital Converter (y DAC)

- Transforma una señal analógica a digital.
- Usa cuantificación sobre una referencia.(def: V_+ , pero hay entrada aref)
- Lo define la resolución (10 bits) y muestreo (15 ksps en mega)
- Un DAC hace lo contrario, como un MP3.
- API

ADC: Analogic Digital Converter (y DAC)

- Transforma una señal analógica a digital.
- Usa cuantificación sobre una referencia.(def: V_+ , pero hay entrada aref)
- Lo define la resolución (10 bits) y muestreo (15 ksps en mega)
- Un DAC hace lo contrario, como un MP3.
- API

ADC: Analogic Digital Converter (y DAC)

- Transforma una señal analógica a digital.
- Usa cuantificación sobre una referencia.(def: V_+ , pero hay entrada aref)
- Lo define la resolución (10 bits) y muestreo (15 ksps en mega)
- Un DAC hace lo contrario, como un MP3.
- API

ADC: Analogic Digital Converter (y DAC)

- Transforma una señal analógica a digital.
- Usa cuantificación sobre una referencia.(def: V_+ , pero hay entrada aref)
- Lo define la resolución (10 bits) y muestreo (15 ksps en mega)
- Un DAC hace lo contrario, como un MP3.
- API

ADC: Analogic Digital Converter (y DAC)

- Transforma una señal analógica a digital.
- Usa cuantificación sobre una referencia.(def: V_+ , pero hay entrada aref)
- Lo define la resolución (10 bits) y muestreo (15 ksps en mega)
- Un DAC hace lo contrario, como un MP3.
- API

PWM: Pulse Width Modulator

- Envía una señal con un pulso, variando el ciclo de carga.
- Se usa para el control de dispositivos.
- API

PWM: Pulse Width Modulator

- Envía una señal con un pulso, variando el ciclo de carga.
- Se usa para el control de dispositivos.
- API

PWM: Pulse Width Modulator

- Envía una señal con un pulso, variando el ciclo de carga.
- Se usa para el control de dispositivos.
- API

Contadores

- Permiten contar ciclos.
- Permiten lanzar interrupciones cuando desbordan.
- En Arduino nos abstraemos, sólo pedimos el tiempo desde el inicio o pedimos esperar un tiempo.
- API

Contadores

- Permiten contar ciclos.
- Permiten lanzar interrupciones cuando desbordan.
- En Arduino nos abstraemos, sólo pedimos el tiempo desde el inicio o pedimos esperar un tiempo.
- API

Contadores

- Permiten contar ciclos.
- Permiten lanzar interrupciones cuando desbordan.
- En Arduino nos abstraemos, sólo pedimos el tiempo desde el inicio o pedimos esperar un tiempo.
- API

Contadores

- Permiten contar ciclos.
- Permiten lanzar interrupciones cuando desbordan.
- En Arduino nos abstraemos, sólo pedimos el tiempo desde el inicio o pedimos esperar un tiempo.
- API

UART: Universal Asynchronous Receiver/Transmitter

- Puerto serie para los amigos.
- Permite comunicarse de forma sencilla con un ordenador (u otros dispositivos).
- En Arduino nos abstraemos, pero hay que configurarlo al arranque.
- Biblioteca Serial

UART: Universal Asynchronous Receiver/Transmitter

- Puerto serie para los amigos.
- Permite comunicarse de forma sencilla con un ordenador (u otros dispositivos).
- En Arduino nos abstraemos, pero hay que configurarlo al arranque.
- Biblioteca Serial

UART: Universal Asynchronous Receiver/Transmitter

- Puerto serie para los amigos.
- Permite comunicarse de forma sencilla con un ordenador (u otros dispositivos).
- En Arduino nos abstraemos, pero hay que configurarlo al arranque.
- Biblioteca Serial

SPI: Serial Peripheral Interface

- Puerto para comunicación con otros dispositivos.
- Usa cuatro hilos:
 - MOSI: Master Output Slave Input
 - MISO: Master Input Slave Output
 - SCK: Reloj común
 - SS: Slave select, es uno distinto por esclavo
- En Arduino nos abstraemos, es una biblioteca.
- Biblioteca SPI

SPI: Serial Peripheral Interface

- Puerto para comunicación con otros dispositivos.
- Usa cuatro hilos:
 - MOSI: Master Output Slave Input
 - MISO: Master Input Slave Output
 - SCK: Reloj común
 - SS: Slave select, es uno distinto por esclavo
- En Arduino nos abstraemos, es una biblioteca.
- Biblioteca SPI

SPI: Serial Peripheral Interface

- Puerto para comunicación con otros dispositivos.
- Usa cuatro hilos:
 - MOSI: Master Output Slave Input
 - MISO: Master Input Slave Output
 - SCK: Reloj común
 - SS: Slave select, es uno distinto por esclavo
- En Arduino nos abstraemos, es una biblioteca.
- Biblioteca SPI

I2C: Inter-Integrated Circuit

- Puerto para comunicación con otros dispositivos.
- Usa dos hilos:
 - SDA: Datos
 - SCL: Reloj común.
- En Arduino nos abstraemos, es una biblioteca.
- Biblioteca Wire

I2C: Inter-Integrated Circuit

- Puerto para comunicación con otros dispositivos.
- Usa dos hilos:
 - SDA: Datos
 - SCL: Reloj común.
- En Arduino nos abstraemos, es una biblioteca.
- Biblioteca Wire

I2C: Inter-Integrated Circuit

- Puerto para comunicación con otros dispositivos.
- Usa dos hilos:
 - SDA: Datos
 - SCL: Reloj común.
- En Arduino nos abstraemos, es una biblioteca.
- Biblioteca Wire

API de Arduino

API

API

IDE de Arduino

IDE de Arduino

- Configuración: placa y directorio de trabajo
- Formato de un programa: setup y loop
- Añadir una biblioteca: Ej: servo
- Verificación y carga de un programa

IDE de Arduino

- Configuración: placa y directorio de trabajo
- Formato de un programa: setup y loop
- Añadir una biblioteca: Ej: servo
- Verificación y carga de un programa

IDE de Arduino

- Configuración: placa y directorio de trabajo
- Formato de un programa: setup y loop
- Añadir una biblioteca: Ej: servo
- Verificación y carga de un programa

IDE de Arduino

- Configuración: placa y directorio de trabajo
- Formato de un programa: setup y loop
- Añadir una biblioteca: Ej: servo
- Verificación y carga de un programa

IDE de Arduino

- Configuración: placa y directorio de trabajo
- Formato de un programa: setup y loop
- Añadir una biblioteca: Ej: servo
- Verificación y carga de un programa

¿?

Gracias por venir y nos vemos en marzo.