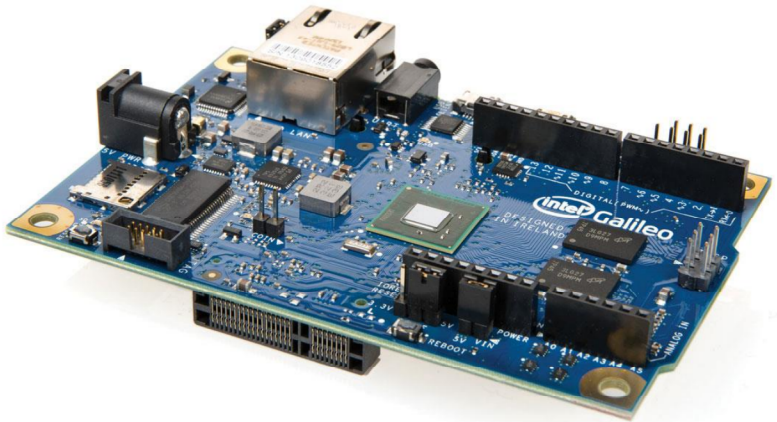
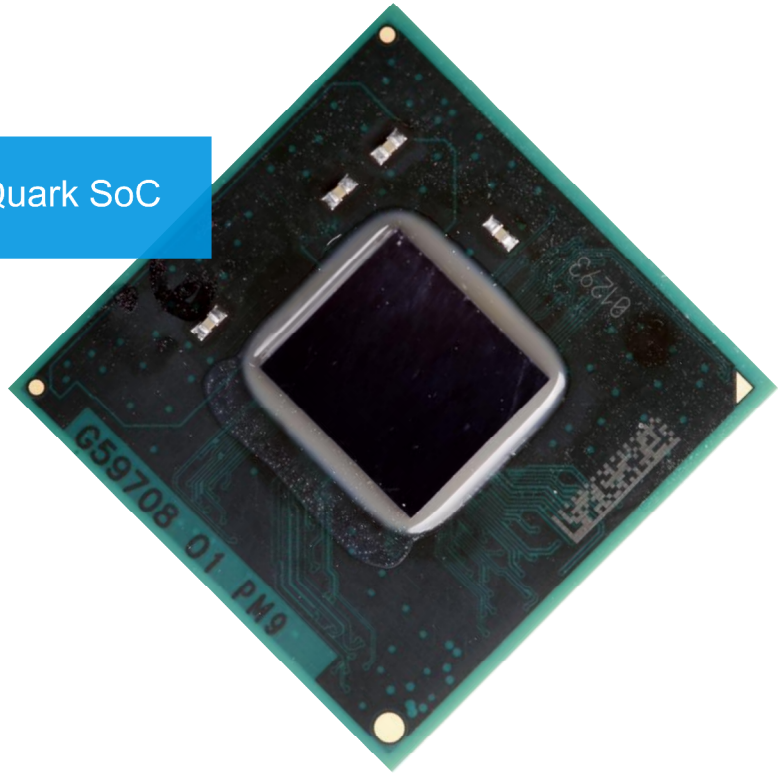


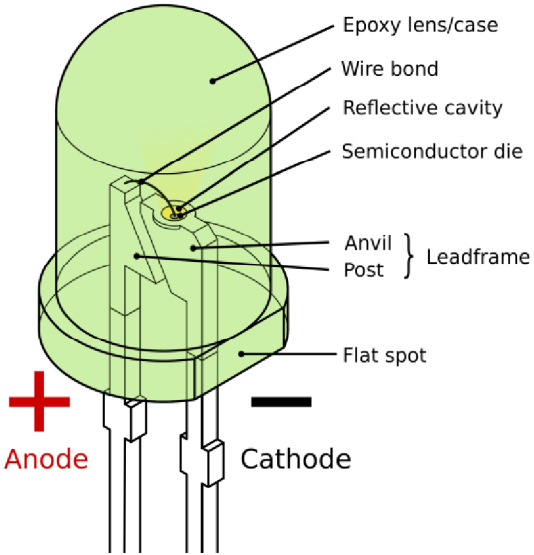
Intel Galileo: Medios de Salida

Intel Galileo: Medios de Salida

Intel Quark SoC



- Medios de Salida en Galileo
- Terminales de Salida Digital
- Salidas Análogas
- Salida de Comunicación Serial
- Aplicación: Control de un display LCD
- Aplicación: Control de Servomotores



Medios de Salida

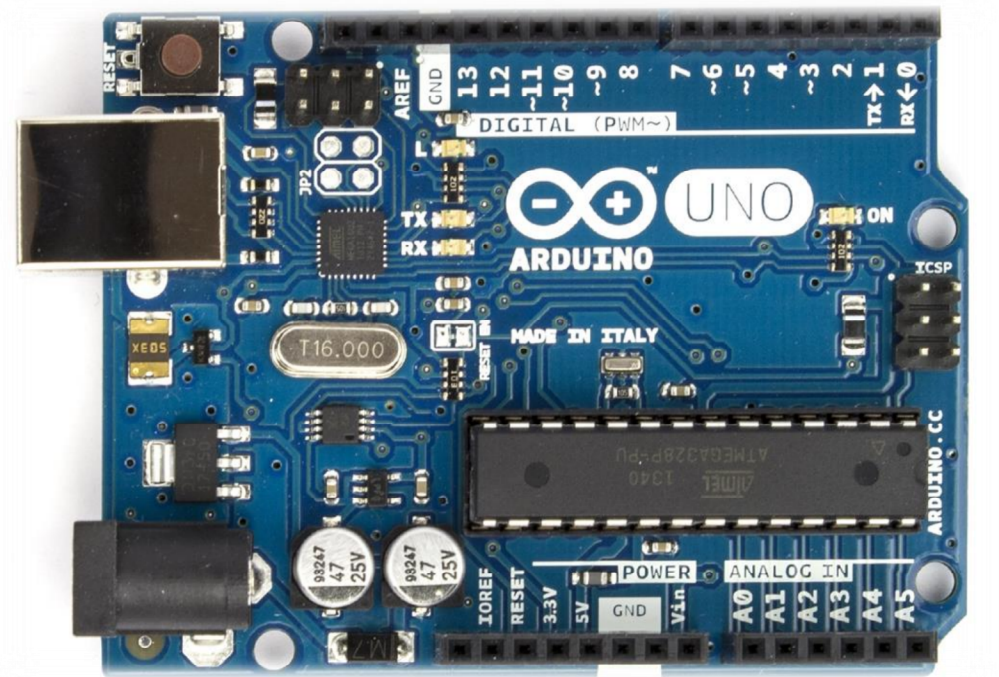
Que interfaces ofrece Galileo?

Los medios de salida de Galileo no solo permiten encender o pagar un Led, también son capaces de ***transmitir*** información a un usuario, hacer que algo se ***mueva*** o comunicarse con otros dispositivos electrónicos (***M2M***).

1. Permite el uso de terminales de salida digitales de propósito general
2. Generación de salidas análogas por medio de PWM
3. Salida serial con protocolos estándar como: UART, I2C o SPI

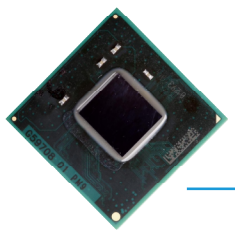
Motto: "It just works."

Header
Arduino R3



Galileo es compatible con los conectores del la placa Arduino UNO R3. En estos conectores, llamados headers, se pueden apreciar las salidas con las cuales disponemos.

La distribución entre salidas digitales y análogas, así como el numero identificador de cada pin se mantienen sin cambios entre Galileo y Arduino.

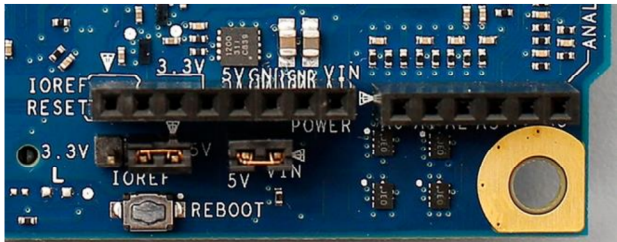


Salidas Digitales de Propósito General

Galileo tiene 14 terminales de salida digital de propósito general. Cada terminal esta representada por un identificador numérico.

Este identificador es utilizado para utilizar cada una de las terminales durante la programación del dispositivo.

Gran parte de los circuitos a conectar a Galileo se alimentan con ***5 V***, sin embargo dispositivos mas recientes tienen interfaces de ***3.3 V***. Galileo puede seleccionar el uso de cualquiera de los dos voltajes por medio del jumper ***IOREF***.



Bibliotecas Arduino

Galileo comparte la amplia librería de rutinas de software para el control de diversos periféricos, así como para facilitar el uso de tareas sencillas como la escritura a una terminal de salida digital.

Algunas bibliotecas requieren su importación al sketch, otras son accesibles de manera implícita en cualquier sketch para Galileo.

En esta sección se utilizan las funciones:

- **pinMode()**
- **delay()**
- **digitalWrite()**

Para usar las funciones anteriores no es necesario incluir librería alguna.

Buscar wiring_digital.c en : arduino/hardware/arduino/x86/cores/arduino

pinMode()
digitalWrite()
delay()

pinMode()

Al hacer una llama a la rutina pinMode se le deja saber a Galileo si una terminal será utilizada como entrada o salida.

Sintaxis:

pinMode(pin, MODO)

MODO: INPUT o OUTPUT.

digitalWrite()

Una vez que se a especificado que una terminal de Galileo será utilizada como salida, se usa esta función para escribir un valor digital alto o bajo a dicha terminal.

Sintaxis:

digitalWrite(pin, ESTADO)

ESTADO: HIGH o LOW.

delay()

Rutina de retardo universal, detiene el flujo de ejecución del sketch actual.

Sintaxis:

delay(ms)

ms: numero de milisegundos a esperar. (1000 ms = 1).

Código
blink.ino

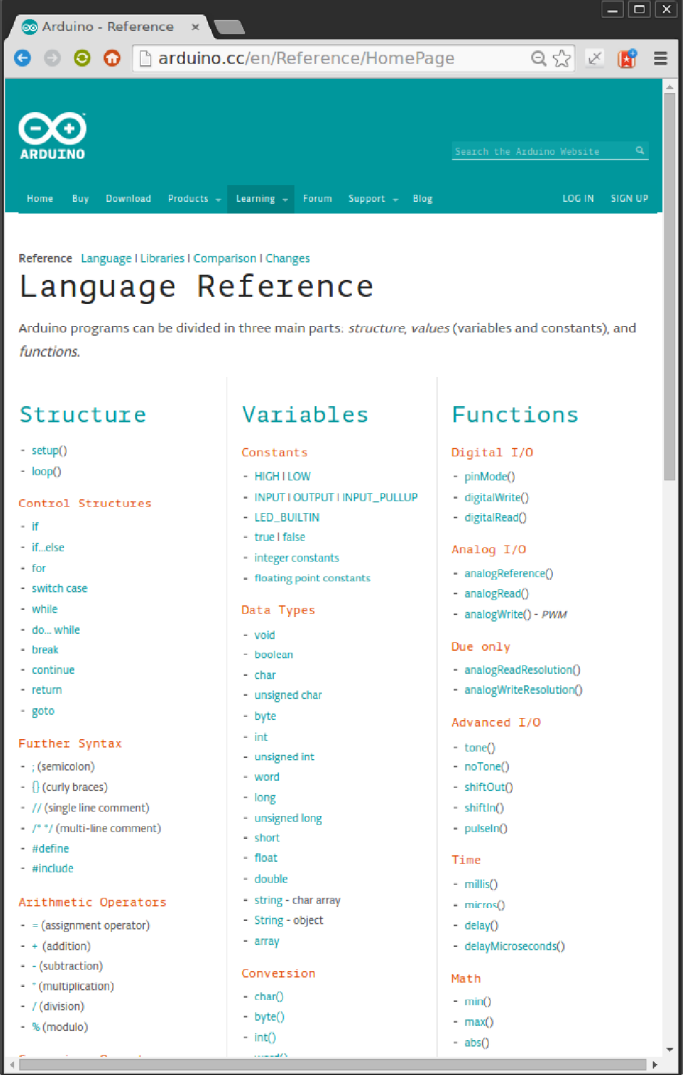
Lenguaje C

```
/* blink.ino */

int led = 13;

void setup() {

  pinMode(led, OUTPUT);

  }

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Asigna a la terminal 13 el rol de salida digital.

Escribe un valor lógico **ALTO** al pin 13.

Retardo de 1 segundo.



Arduino Language Reference

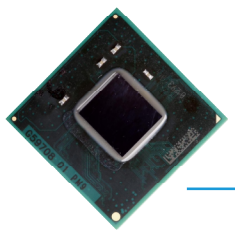
Galileo hereda de Arduino un **amplio compendio de documentación**, en especial el **Compendio de Referencia del Lenguaje Arduino**, el cual presenta una guía de todas las características de Arduino como lenguaje de programación.

En este, se listan **todas la características** comunes para todos los miembros de la familia Arduino. Por ejemplo, todas las funciones que se utilizaron en el ejemplo blink.ino se encuentran listadas.

La gran mayoría de las entradas en la referencia del lenguaje Arduino cuenta con un ejemplo de su uso.

http://arduino.cc/en/Reference/HomePage

Todo el contenido de la Arduino Language Reference forma parte de la biblioteca estándar de Arduino, por lo que no requiere la importación de ningún código a nuestro sketch para utilizarla.



Salidas Análogas

Galileo puede generar salidas análogas tan pronto como sale de su caja, y no requiere hardware extra.

Las salidas análogas son generadas por medio de señales **PWM**.

No todos las terminales de Galileo son capaces de generar una salida análoga, solo aquellos que están marcados con una tilde (**~**) junta a su identificador de terminal, son capaces de generar este tipo de señal.

analogWrire()

La llamada a esta rutina indica a una terminal de salida que debe de generar una señal PWM de manera continua con el ciclo de trabajo indicado

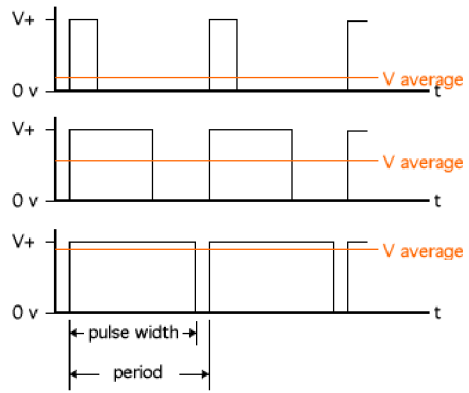
La señal PWM genera con esta rutina se mantiene de **manera continua** en la terminal de salida hasta que se escribe un nuevo analogWrite() a la misma terminal, o se ejecuta un digitalWrite() o digitalRead().

Sintaxis:

analogWrite(pin, CICLO)

CICLO: 0 a 255.

analogWrite()



PWM puede utilizarse para para el control de servo motores, motores de corriente directa o para regular la intensidad con la que brilla un led

Código
fade.ino

```
int led = 9;
int brightness = 0;
int fadeAmount = 5;

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

void loop()  {
  analogWrite(led, brightness);
  brightness = brightness + fadeAmount;

    if (brightness == 0 || brightness == 255)
      fadeAmount = -fadeAmount;

  delay(30);
}
```

Terminal 9 toma el rol de salida.

Escribir la cantidad de “brillo” que queremos obtener del led. En este casa el 100% de ciclo de trabajo esta representado por 255.

Una vez que se alcanza alguno de los valores máximos, se invierte de signo para alcanzar el valor opuesto de “brillo”

Lenguaje C vs Arduino

La caminata de leds

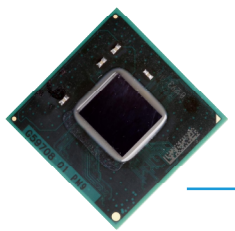
El equivalente para Galileo del famoso “Hola Mundo” seria hacer una caminata de leds.

Sin embargo utilizando la infraestructura de Arduino esta simple tarea puede resultar desafiante en un principio.

Lenguaje C

```
char        leds = 0x01;

while(1){
            portb = leds;
            delay(1000);
            leds = leds << 1;
            if (leds == 0)
                        leds = 0x01;
}
```

y Arduino … ?



Salida
Comunicación Serial

La **conexión USB** utilizada por Galileo para la carga de Sketch puede ser utilizada de igual forma para **intercambiar información** con el computadora.

Esta salida permite mostrar información en un formato amigable con los seres humanos!

La comunicación serial de Galileo es una herramienta invaluable para la depuración de aplicaciones, es tan sencillo su uso que basta 1 línea de código para habilitar este periférico.

Los datos transmitidos por el puerto serial pueden ser sometidos a un formato de transmisión. Mas información en ***Arduino Language Reference.***

Serial.print(32, HEX);

Serial.begin()
Serial.print()
Serial.println()

Serial.begin()

Rutina para abrir el puerto serial de Galileo. Junto a la apertura del puerto se debe de definir la velocidad de transmisión de datos.

Sintaxis:

Serial.begin(SPEED)

SPEED: Velocidad en bits por segundo.

Serial.print()

Esta rutina da inicio a la transmisión de caracteres a través del puerto serial.

Sintaxis:

Serial.print(VALOR)

VALOR: cualquier tipo de dato, desde cadenas hasta caracteres.

Serial.println()

Exhibe el mismo comportamiento que Serial.print(), pero a diferencia de este, Serial.println() agregar una salto de línea después de la cadena transmitida.

Código
blinkSerial.ino

```
int led = 13;

void setup() {
      pinMode(led, OUTPUT);
      Serial.begin(9600);
}

void loop() {
      Serial.print("ALTO\n");
      digitalWrite(led, HIGH);
      delay(1000);
      Serial.println("BAJO");
      digitalWrite(led, LOW);
      delay(1000);
}
```

Inicializa el puerto serial con una velocidad de transmisión de 9600 bits por segundo.

IMPORTANTE: El receptor de la comunicación serial debe estar configurado a la misma velocidad de transmisión, de lo contrario la transmisión de información no se llevara a cabo de manera satisfactoria.

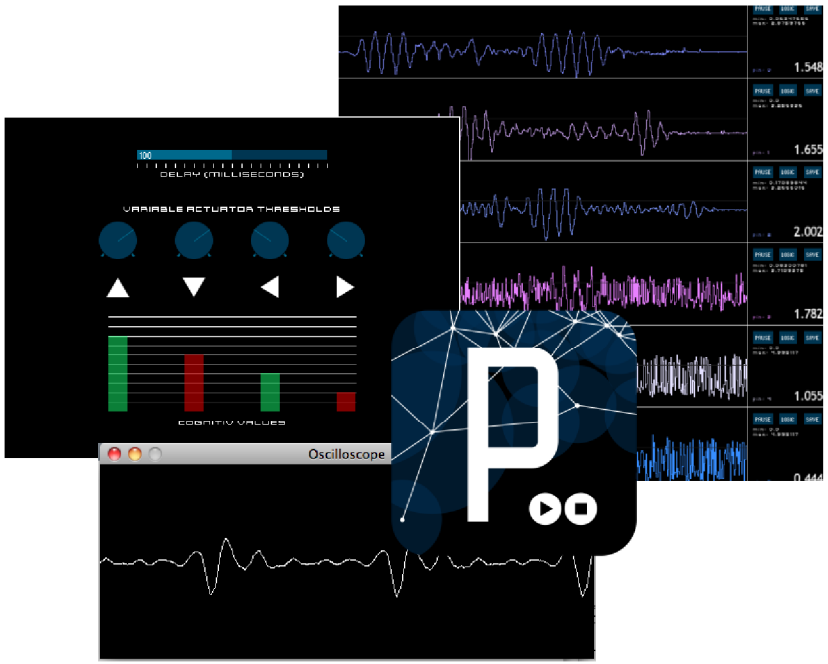
El uso de Serial.println() nos ahorra la necesidad de incluir el carácter de retorno de línea (“\n”) en cada cadena enviada.

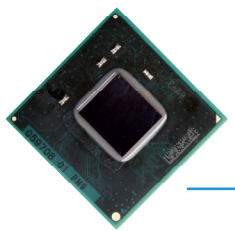
Puerto Serial
Processing

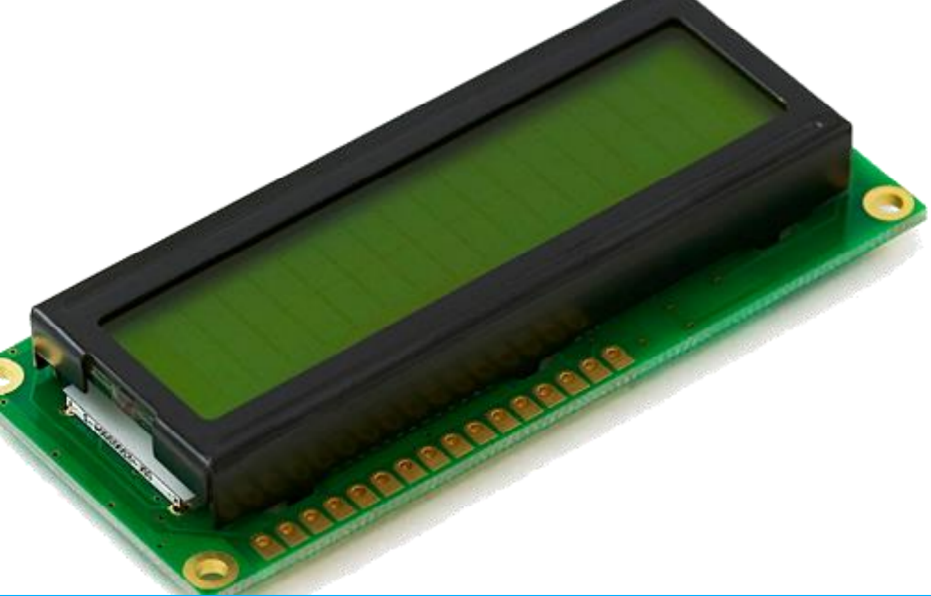
Processing

Processing es un programa que facilita el despliegue de información recibida a partir de un Puerto serial.

Este curso cubre una introducción a esta invaluable herramienta!







Pantalla
LCD 2x16

Galileo proporciona también una biblioteca para el control de pantallas LCD: **LiquidCrystal.h**

Esta biblioteca se basa en el controlador HD44780 de Hitachi, que se encuentra en la mayoría de las pantallas LCD.

La biblioteca trabaja tanto en el modo de 4 u 8 bits de datos, además de las señales de control RS y Enable y opcionalmente la señal de control RW.

Biblioteca LiquidCrystal.h

En esta sección se utilizan las funciones de la biblioteca **LiquidCrystal.h**:

- **LiquidCrystal ( )** — Inicializa la interfaz de la pantalla LCD
- **begin** (cols, rows)
- **print** (data) — Imprime texto en la LCD
- **setCursor** (col, row) — Posiciona el cursor en la LCD

.

Para ver todas las funciones: http://arduino.cc/en/Reference/LiquidCrystal

LiquidCrystal ( )

LiquidCrystal ( )

Esta función crea una variable
del tipo LiquidCrystal.

.**Sintaxis:**

LiquidCristal **name**(RS, E, D4, D5, D6, D7)

LiquidCristal **name**(RS, RW, E, D4, D5, D6, D7)

LiquidCristal **name**(RS, E, D0, D1, D2, D3, D4, D5, D6, D7)

LiquidCristal **name**(RS, RW, E, D0, D1, D2, D3, D4, D5, D6, D7)

Como se puede apreciar, la LCD puede ser controlada mediante 4 u 8 líneas de datos. Para controlarla con 4 líneas, simplemente omitir escribir las señales D0, D1, D2 y D3. El uso del pin RW es opcional; en caso de no utilizarlo se puede conectar directamente a tierra.

Configuración de pines del LCD 2x16

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
GND	VCC	VEE	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	LED+	LED-

Código
HelloWorld.ino

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}

void loop() {
  lcd.setCursor(0, 1);
  lcd.print(millis()/1000);
}
```

Utiliza 4 bits para configurar la LCD y no se usa el bit RW.

Para el shield que usarán, la configuración es distinta:

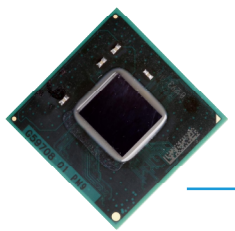
```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

Inicializa la pantalla de 16 caracteres por 2 líneas

Imprime una vez la palabra **hello, world!** en la primer línea de la pantalla LCd

Posiciona el cursor al inicio de la segunda línea

Imprime en la pantalla una cuenta en segundos



control Servos

Que es un Servo Motor?

Un servo, es un pequeño motor que puede colocar su eje en una posición deseada y mantener esa posición.

Las instrucciones al servomotor son enviadas en forma de señales PWM. Donde diferentes ciclos de trabajo indican diferentes posiciones la Servo motor.

Biblioteca Servo.h

La biblioteca Servo.h permite manejar de manera simple un servo motor.

Sin embargo requiere una actualización para mejorar la experiencia en el desplazamiento del motor.

Reemplazo Servo.h

1.- Obtener una copia de la librería modificada en: → https://github.com/mikalhart/galileo-Servo/releases

2.- Sustituir la versión original de la librería en la ruta: → ../hardware/arduino/x86/libraries/Servo

3.- Reinicia el IDE de Arduino

IMPORTANTE: Para información detallada de los cambios efectuados a la librería Servo.h visitar:

https://communities.intel.com/thread/48546

Librería Servo.h

Galileo tiene una resolución de pasos en los servomotores de aproximadamente 10° .

Servo myservo()

La librería Servo es una librería orientada a objetos, cada servo a controlar debe estar ligado a un objeto del tipo Servo.

Cada objeto es independiente.

Sintaxis:

Servo (ID)

ID: Identificador del Servo.

myservo.attach()

La rutina attach() debe ser invocada con un objeto del tipo servo. Esta rutina indica en que pin de Galileo se encuentra el servo motor físicamente.

Sintaxis:

myservo.attach(PIN)

PIN: Terminal a la que se encuentra conectado el servomotor.

myservo.write()

Por medio de esta función se indica la posición que el servo motor debe de tomar.

Sintaxis:

myservo.write(VALUE)

VALUE: Posición del servo motor.

IMPORTANTE: Este sketch funciona con la librería original de Galileo, sin embargo el motor se sentirá “forzado” durante sus cambios de posición.

Código
servo.ino

```
#include <Servo.h>

int servoPin = 9;

Servo myServo;

void setup() {
   myServo.attach(servoPin);
}

void loop() {
     myServo.write(19);
     delay(1000);
     myServo.write(180);
     delay(1000);

}
```

Importar la librería Servo.h

Creación de un objeto de la clase Servo. Igual que en programación orientada a objetos

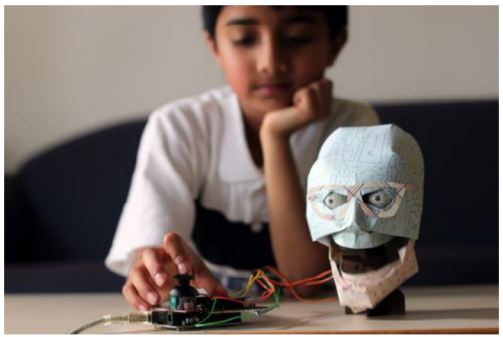
Asignar el pin 9 al objeto Servo. Todas las operaciones sobre el objeto ‘myservo’ se reflejaran en el pin 9.

Envió de señal de control al servo motor para asignar una nueva posición.

IMPORTANTE: Este sketch funciona con la librería original de Galileo, sin embargo el motor se sentirá “forzado” durante sus cambios de posición.

Servos
Para que Sirven?

http://www.uberreview.com/2012/11/animatronic-lamp-is-the-cutest-robot-you-will-see-all-day.htm



http://www.designboom.com/technology/tj-animatronic-puppet/