

TECNOLOGÍAS EMERGENTES

Seminario 1

Arduino & Processing

Introducción a las Tecnologías Emergentes

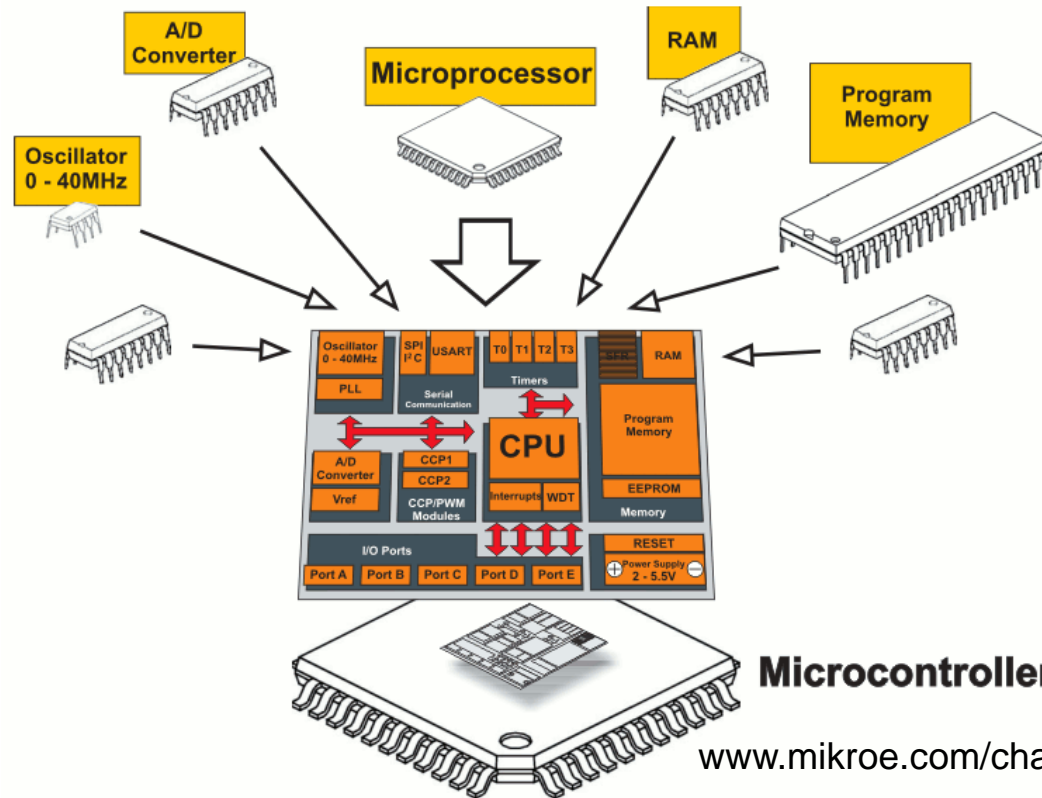
Qué vamos a ver hoy...

- Algunas definiciones previas
- Computación Ubicua y Vestible
- Aplicaciones de los sistemas vestibles
- Hardware para sistemas vestibles

- *HW libre*
- *Cosas q se pueden hacer con Arduino*
- *Formatos de placas Arduino (y breadboard)*
- *Shields*

- *Cosas q se pueden hacer con Processing*
- *Diferencias entre Arduino y Processing*
- *Plataformas (PC, web, móvil)*

Concepto de Microcontrolador

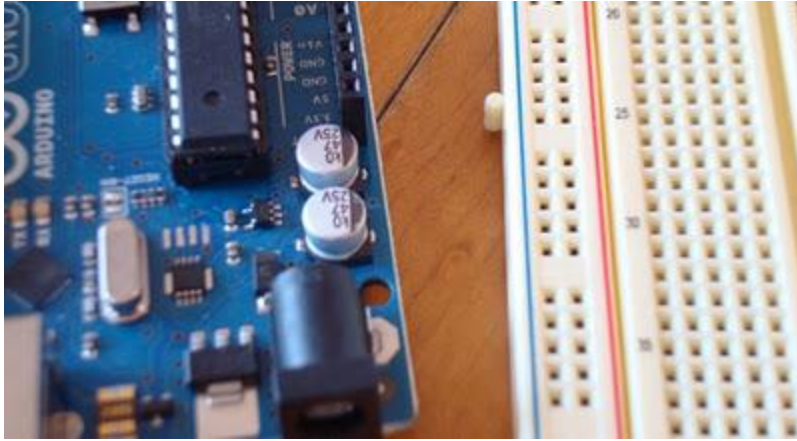


www.mikroe.com/chapters/view/1

Fig. 0-1 Microcontroller versus Microprocessor

- Es un computador en un solo chip
 - Incluye procesador, memoria y Entrada/Salida
- Normalmente, “empotrados” en otro dispositivo al que controlan
- Suelen ser pequeños y de bajo coste

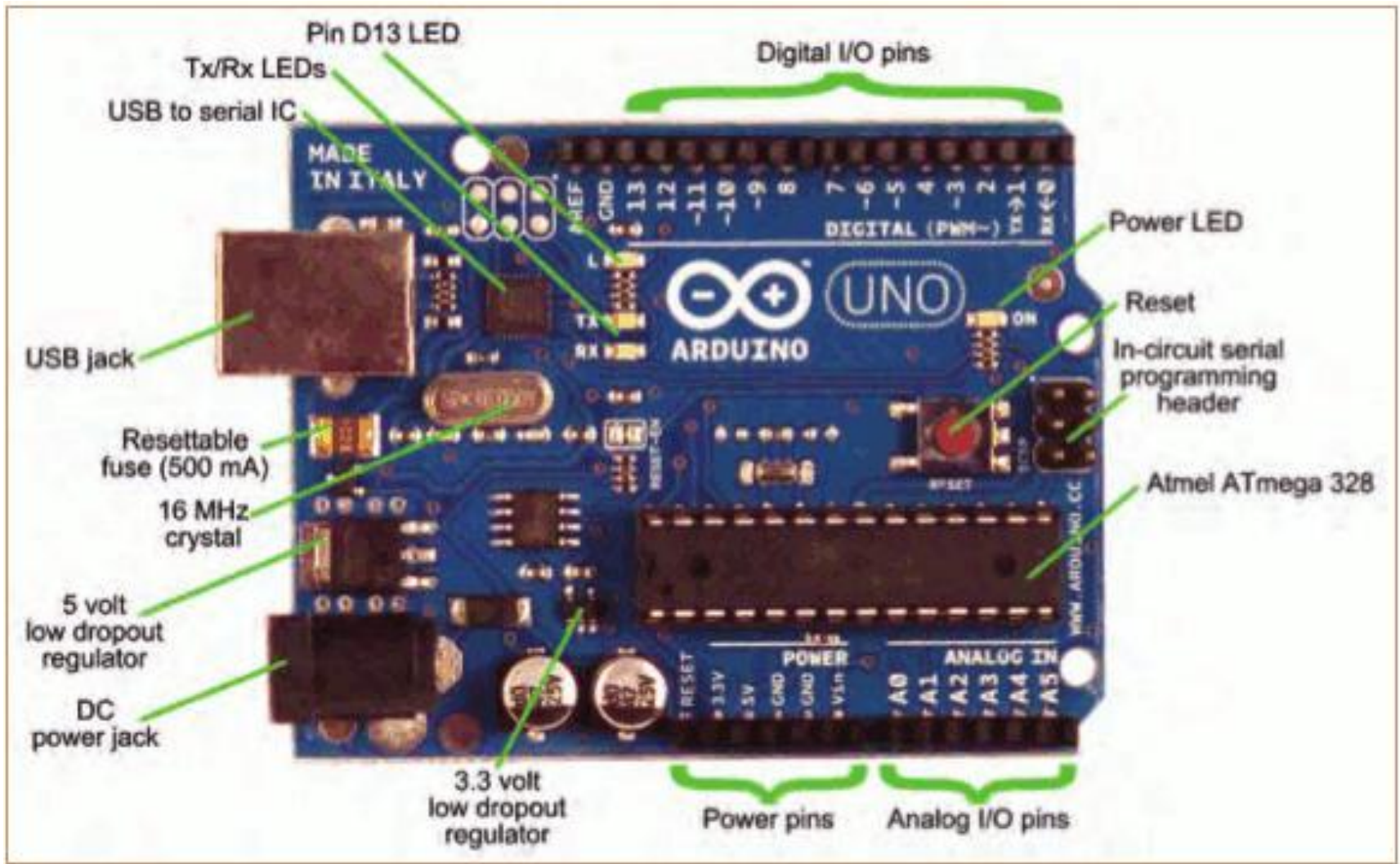
Placas de desarrollo



- *Son circuitos impresos especialmente fabricados para facilitar el diseño con un determinado microcontrolador*

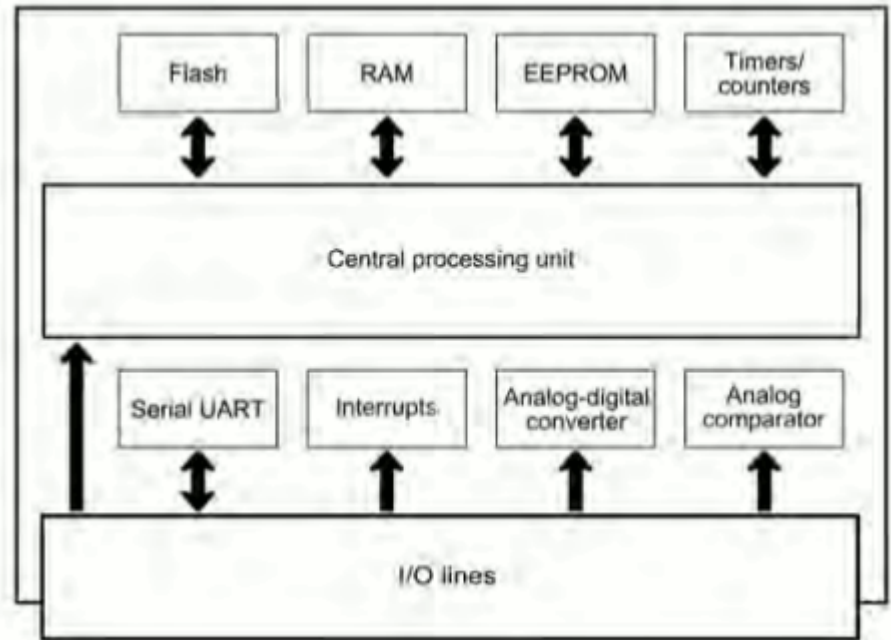
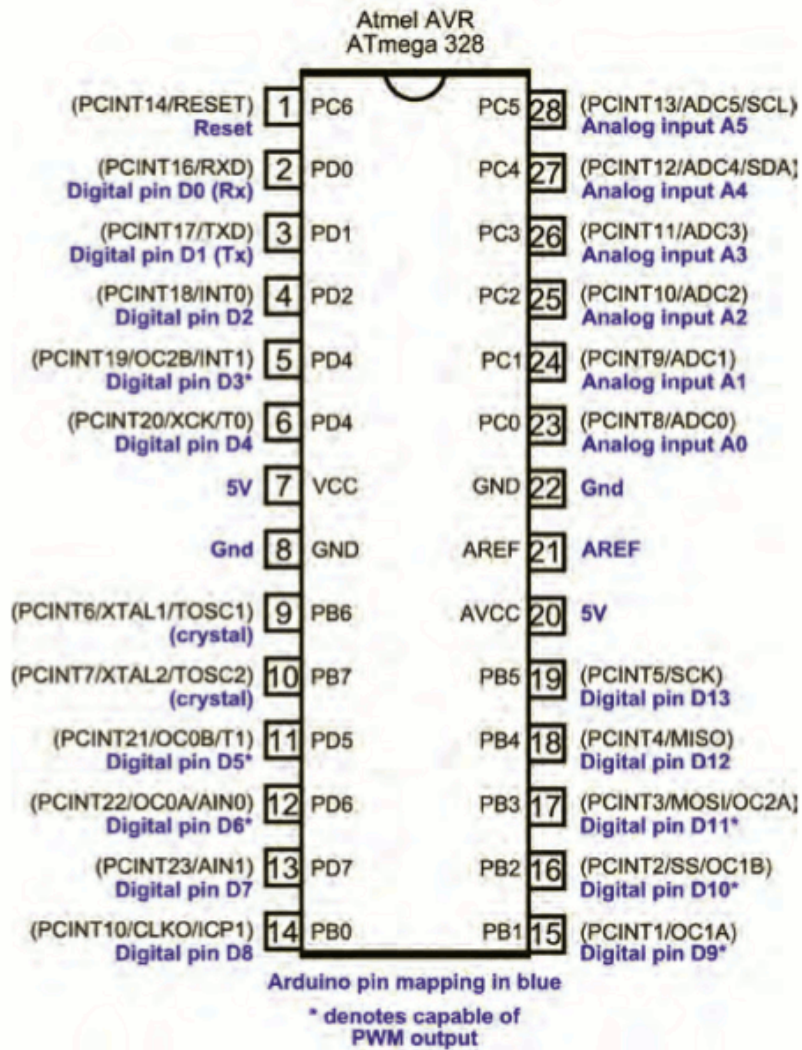
- Suelen incluir:
 - Alimentación
 - Interfaz de programación (USB, ...)
 - Una entrada/salida mínima (pulsadores y LEDs)

La placa de desarrollo Arduino (modelo UNO)



Making-robots-with-arduino.pdf

El microcontrolador de Arduino: Atmel ARV Atmega 328



Especificaciones

Making-robots-with-arduino.pdf

En qué consiste Arduino

The word “Arduino” can mean 3 things

A physical piece of hardware



A programming environment



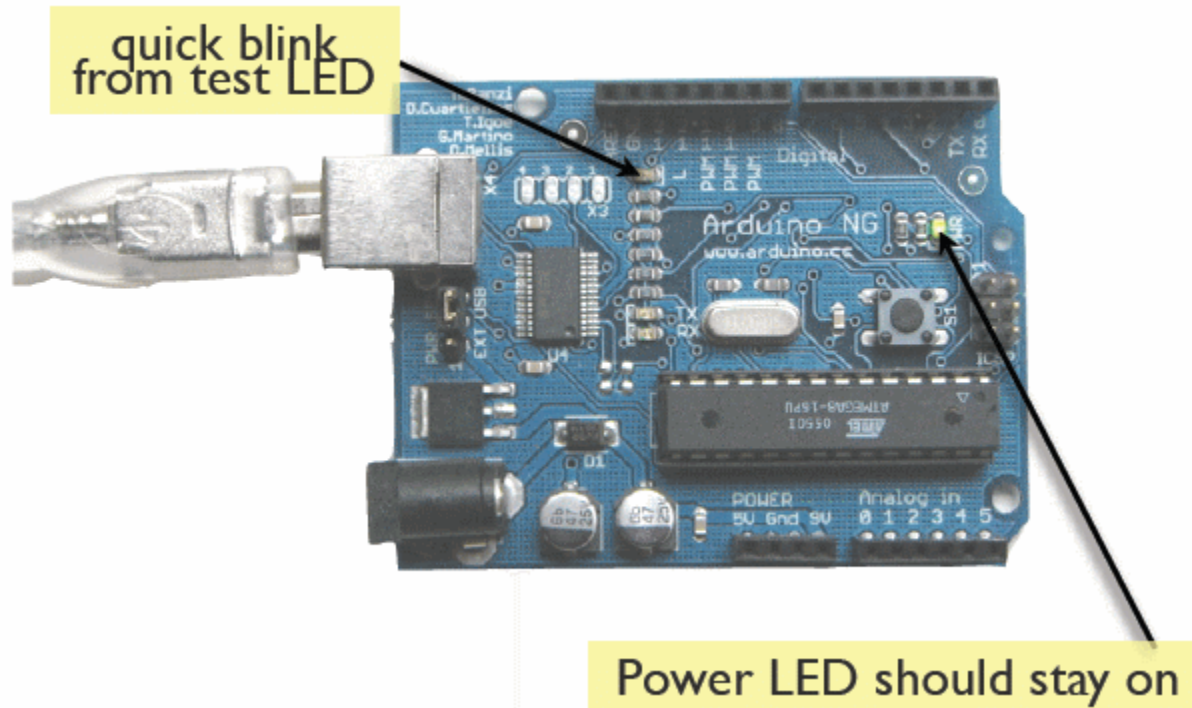
A community & philosophy



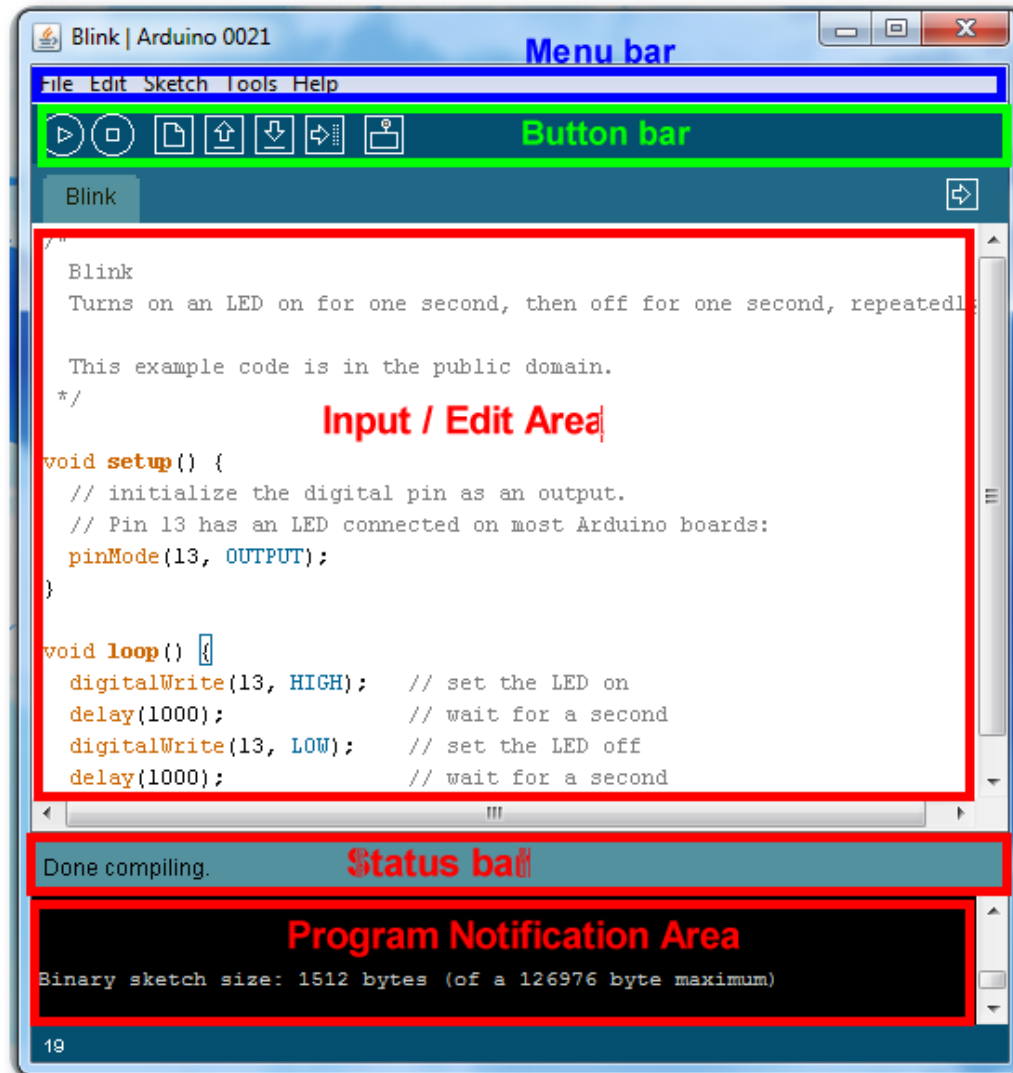
Cómo empezar

- *Ir a la página principal:* <http://arduino.cc/en/Guide/HomePage>
 1. Descargar e instalar el entorno de desarrollo (IDE)
 2. Conectar la placa al ordenador por USB (cable A-B)
 3. Instalar los drivers, si fuese necesario
 4. Abrir el IDE de Arduino
 5. Seleccionar en el menú el modelo de placa
 6. Seleccionar el puerto serie por el que nos comunicamos con la placa
 7. Abrir cualquier ejemplo (“blink” para ver un LED parpadeando)
 8. Cargar el programa (“sketch”) en la placa

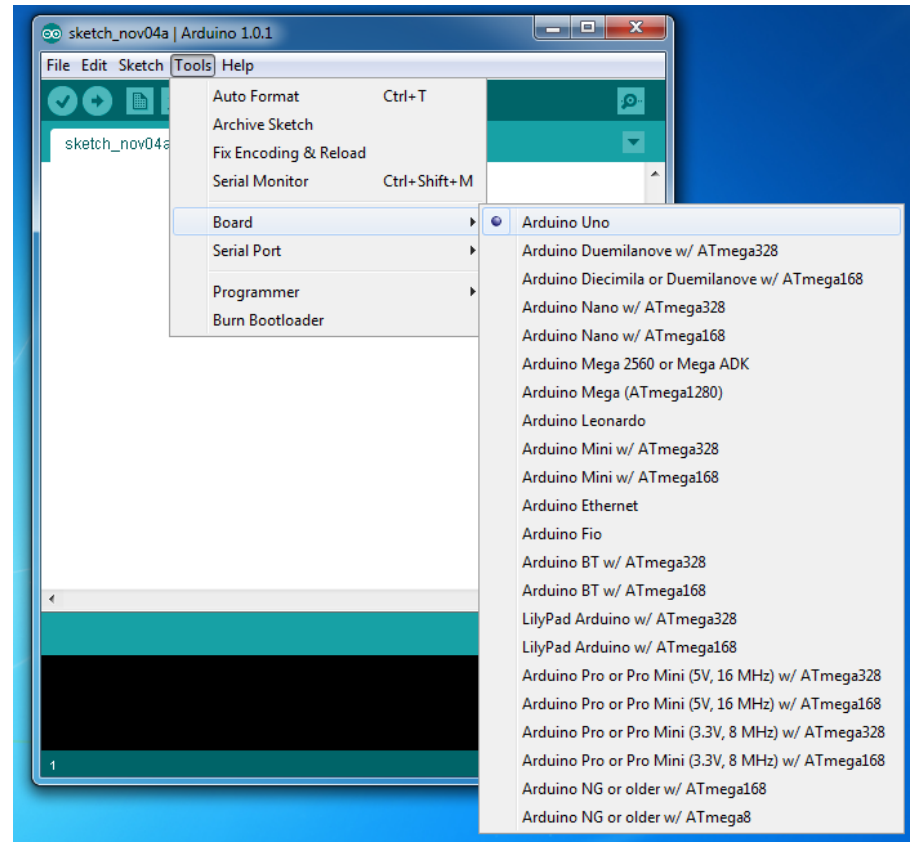
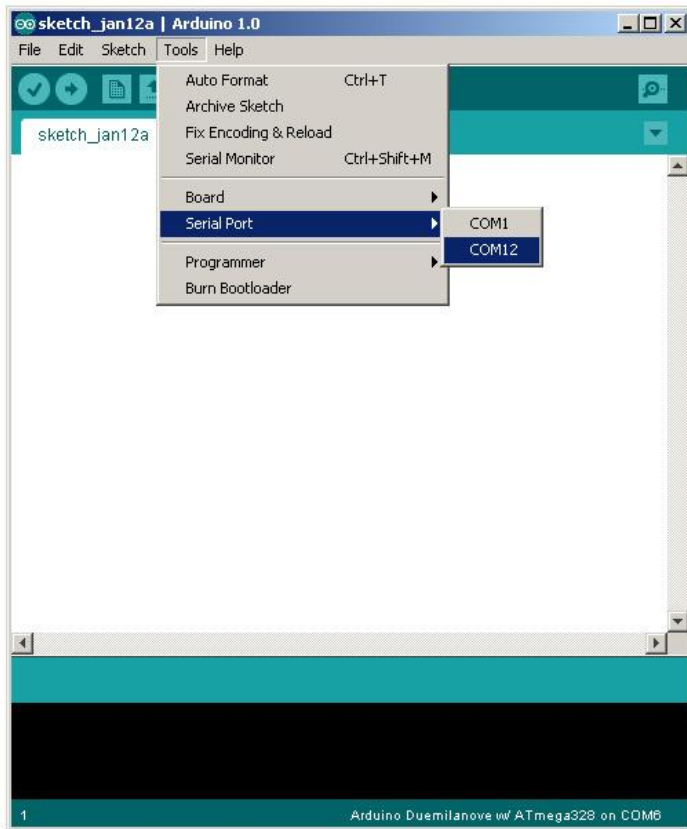
Comprobación de la conexión



Arduino IDE



Selección de puerto serie y modelo de placa



Mensajes de estado del IDE

Uploading worked

Size depends on complexity of your sketch

```
Done uploading.  
Binary sketch size: 1110 bytes (of a 14336 byte maximum)
```

Wrong serial port selected

```
Serial port '/dev/tty.usbserial-A4001qa8' not found. Did you select the  
java.awt.EventQueue$DispatchThread.run(EventDispatchThread.java:118)  
at  
java.awt.EventQueue$DispatchThread.run(EventDispatchThread.java:118)
```

Wrong board selected

```
Wrong microcontroller found. Did you select the right board from the T  
Binary sketch size: 000 bytes (of a 7168 byte maximum)  
avrdude: Expected signature for ATMEGA8 is 1E 93 07  
Double check chip, or use -F to override this check.
```

nerdy cryptic error messages

Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

Try it out with the “Blink” sketch!

Load “File/Sketchbook/Examples/Digital/Blink”

```
void setup() {  
  pinMode(ledPin, OUTPUT); // sets t  
}  
void loop() {  
  digitalWrite(ledPin, HIGH); // sets t  
  delay(1000); // waits  
  digitalWrite(ledPin, LOW); // sets t  
  delay(1000); // waits  
}
```



compile

Done compiling.



upload



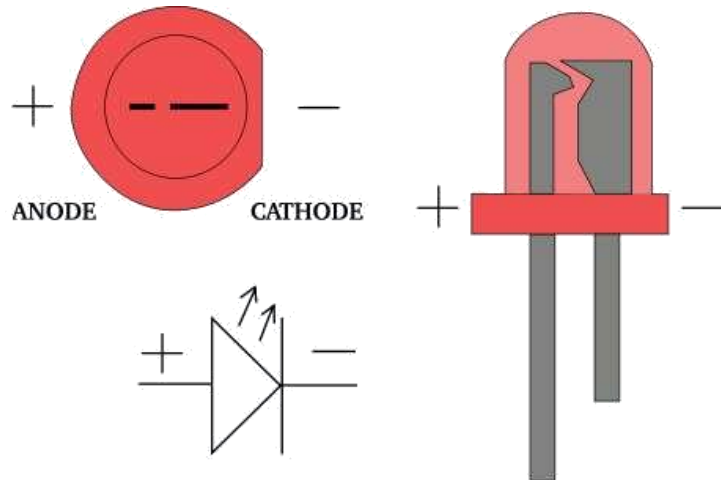
TX/RX flash



sketch runs

Añadiendo una salida externa (LED en pin 13)

- **File > Examples > Digital > Blink**
- Cuidado: los LED tienen polaridad
 - Negativo: patilla corta y lado plano de la carcasa



www.instructables.com

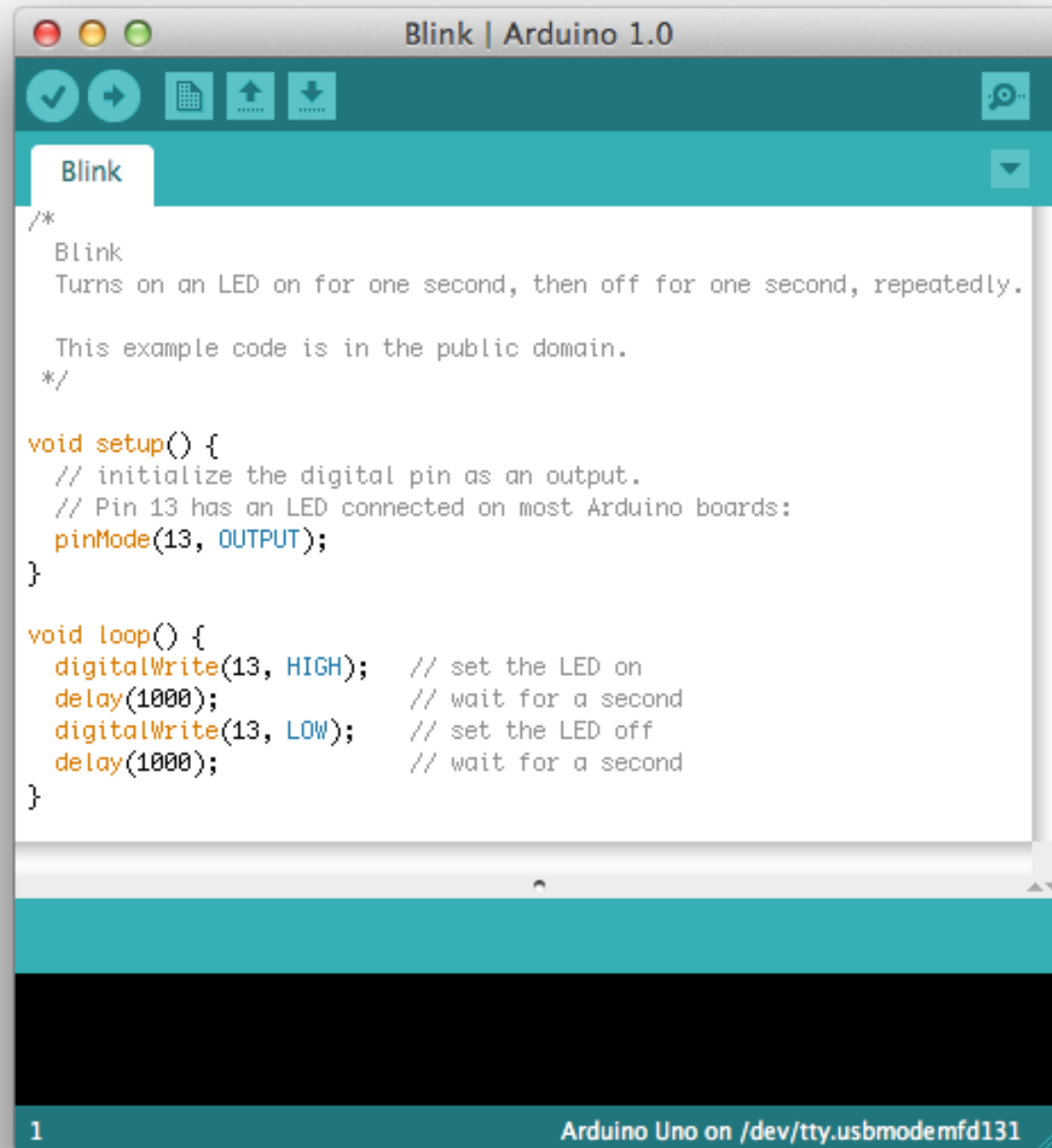


Sobre el lenguaje de programación



- Distingue mayúsculas de minúsculas.
- Las sentencias acaban en ;
- Comentarios tras // o entre /* */
- Dos estructuras:
 - Setup
 - Loop

El “hola mundo” de Arduino (parpadeo o “blink”)



The image shows a screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.0". The top toolbar contains icons for checking, running, uploading, and downloading. The "Blink" sketch is selected in the left sidebar. The main text area displays the following code:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);             // wait for a second
}
```

At the bottom of the window, the status bar shows "1" on the left and "Arduino Uno on /dev/tty.usbmodemfd131" on the right.

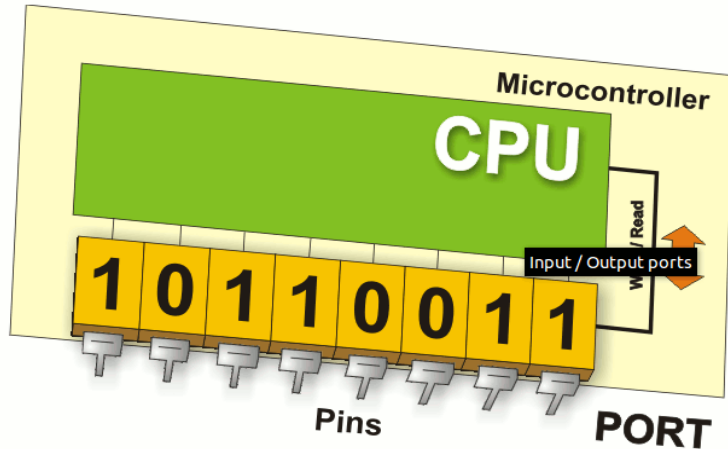
Terminología

“*sketch*” – a program you write to run on an Arduino board

“*pin*” – an input or output connected to something.
e.g. output to an LED, input from a knob.

“*digital*” – value is either HIGH or LOW.
(aka on/off, one/zero) e.g. switch state

“*analog*” – value ranges, usually from 0-255.
e.g. LED brightness, motor speed, etc.



Entradas / Salidas digitales

`pinMode(pin, mode)`

Fija el modo del pin en INPUT o OUTPUT

`digitalRead(pin)`

Consulta el valor del pin (HIGH o LOW)

`digitalWrite(pin, value)`

Escribe un valor en el pin (HIGH o LOW)

Ojo:

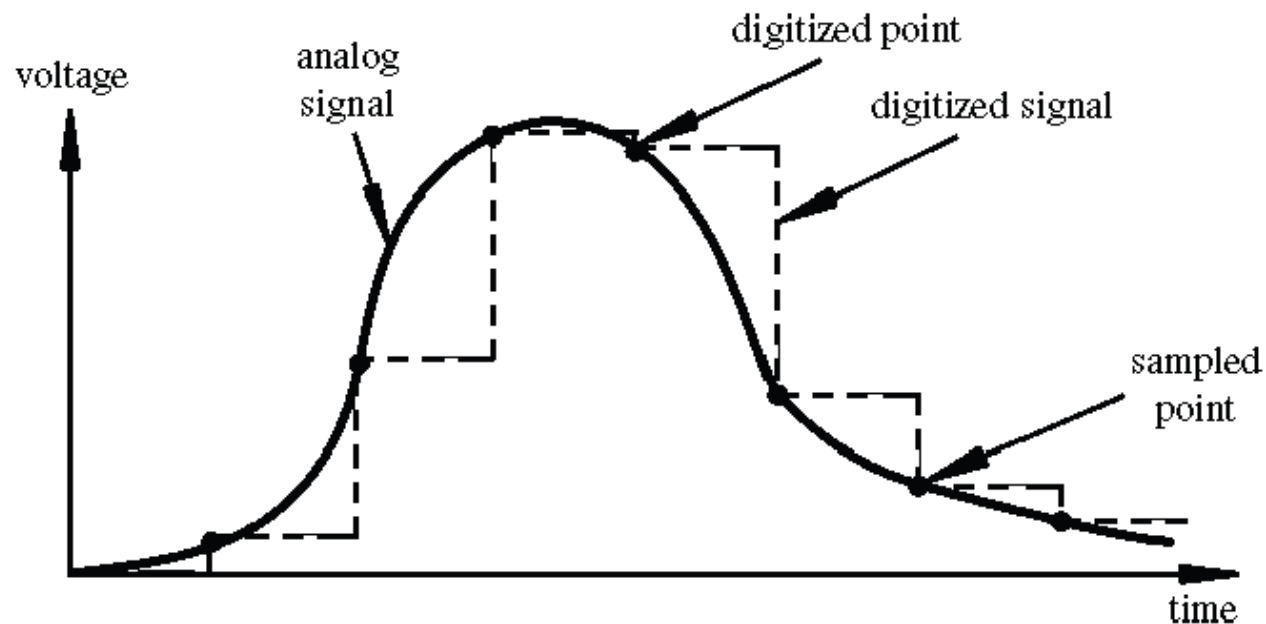
Los pines de salida no dan más de 40 mA de corriente

Temporización

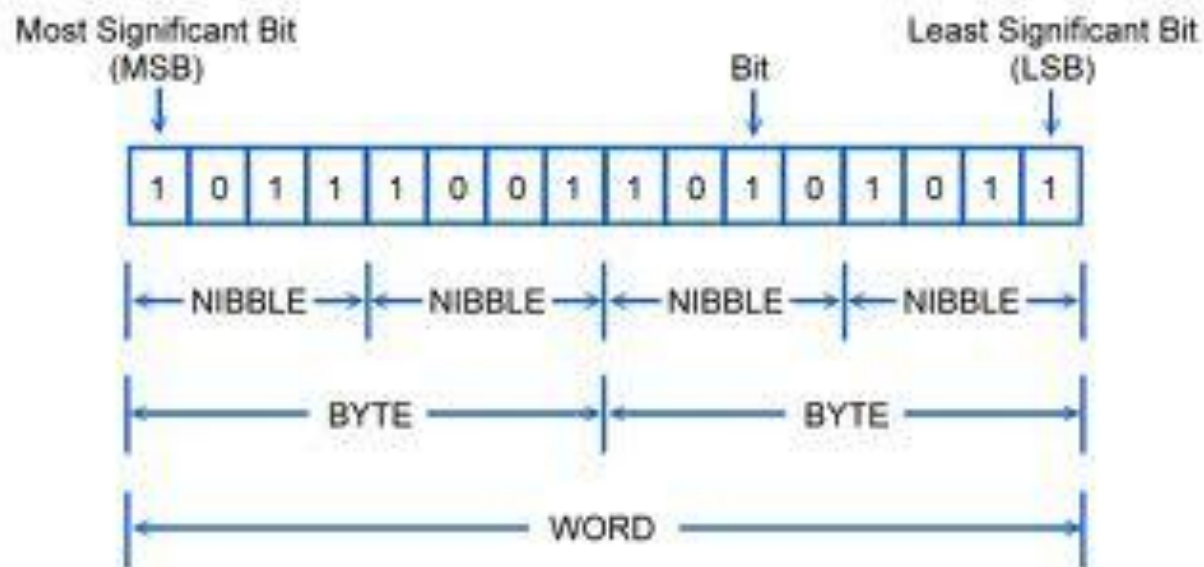
- `delay (ms)`
 - Proporciona una pausa en milisegundos
- `delayMicroseconds (us)`
 - Pausas de microsegundos
- Más órdenes en:
arduino.cc/en/Reference/HomePage

Digital vs. Analógico

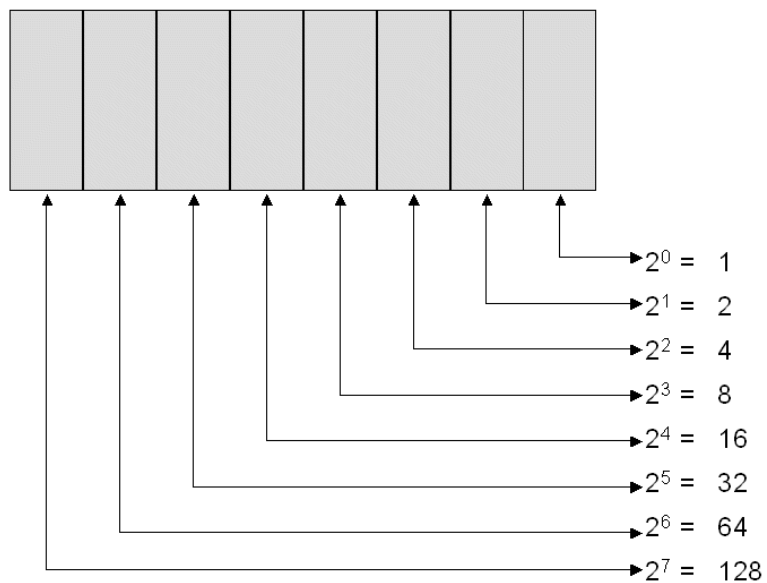
- Digital: sólo dos valores (HIGH/LOW)
- Analógico: “cualquier” valor intermedio
- En realidad, valores cuantizados en una serie de bits (precisión)



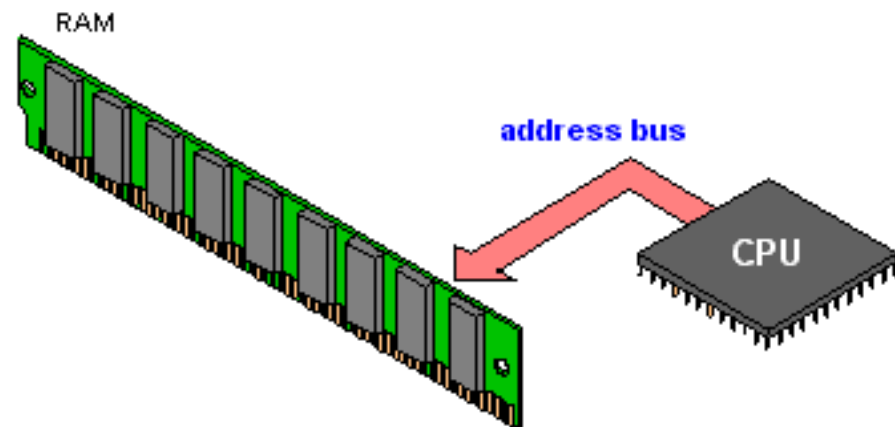
Bits y Bytes



Bit positions: 8 7 6 5 4 3 2 1



From Computer Desktop Encyclopedia
© 2004 The Computer Language Co. Inc.



Comentarios

- En cualquier parte del código
- Usando `//` o `/* y */`
- No afectan al código
- Muy recomendables

Operadores

El signo de igual se usa para

= asignación

== comparación

Operadores

–And & Or

–&& para hacer “and”

–|| para hacer “or”

Variables

Tipos básicos

Boolean

Integer

Character

Declaración de variables

Boolean: *boolean variableName;*

Integer: *int variableName;*

Character: *char variableName;*

String: *stringName [];*

Asignación de variables

- Boolean: ***variableName = true;***
 - o ***variableName = false;***
- Integer: ***variableName = 32767;***
 - o ***variableName = -32768;***
- Character: ***variableName = 'A';***
 - o ***stringName = "SparkFun";***

Ámbito o visibilidad de las variables

```
Blink$  
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
|  
const int variable1 = 1;  
int variable2 = 2;  
  
void setup() {  
  int variable3 = 3;  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino Boards.  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH); // set the LED on
```

Constant / Read only

Variable available
anywhere

Variable available only
in this function,
between curly brackets

Setup

void setup () {}

```
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}
```

Antes de loop, obligatorio en todos los sketches de Arduino

Setup

void setup () {}

```
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}
```

La cabecera es siempre fija

Setup
void setup () {
pinMode (13, OUTPUT); }

```
void setup() {  
  // initialize the digital pin as an output.  
  // pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}
```

*Las salidas se declaran en setup()
usando la función pinMode*
P. ej., fijamos el pin 13 como salida

Setup

*void setup () { **Serial.begin;**}*

```
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
  Serial.begin(9600);  
}
```

*Las comunicaciones serie también se
fijan en el setup()*

P. ej., fijamos la velocidad a 9600 bps

Setup, Resistencias internas de pull-up

***void setup () {
digitalWrite (12, HIGH); }***

```
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
  Serial.begin(9600);  
  digitalWrite(12, HIGH);  
}
```

*Podemos crear resistencias internas de pull-up usando
digitalWrite con el pin a HIGH*

Setup, Interrupciones

```
void setup ( ) {  
attachInterrupt (interrupt, function,  
mode) }
```

Podemos asignar una interrupción a los pines dos y tres de la placa

Así evitamos el procesamiento lineal de Arduino

Setup, Interrupciones

```
void setup ( ) {  
attachInterrupt (interrupt, function,  
mode) }
```

Interrupt: número de interrupción, 0 o 1, para los pines # 2 y 3 respectivamente

Function: función a la que llamar para atender a la interrupción

Mode: indica cuándo disparar la interrupción

Setup, Interrupciones

```
void setup ( ) {  
  attachInterrupt (interrupt, function,  
    mode) }
```

- **LOW** cuando el pin esté en valor bajo
 - **CHANGE** cuando el pin cambie su valor
 - **RISING** en flanco de subida
 - **FALLING** en flanco de bajada
- (valores predefinidos → mayúsculas)

If

if (this is true) { do this; }

```
void loop(){  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);  
  
  // check if the pushbutton is pressed.  
  // if it is, the buttonState is HIGH:  
  if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);  
  }  
  else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```

— If Statement

If

if (this is true) { do this; }

```
void loop(){  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);  
  
  // check if the pushbutton is pressed.  
  // if it is, the buttonState is HIGH:  
  if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);  
  }  
  else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```

IF

if (buttonState == HIGH) {

if (this is true) { do this; }

```
void loop(){  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);
```

```
  // check if the pushbutton is pressed.  
  // if it is, the buttonState is HIGH:
```

```
  if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);
```

```
  }  
  else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```

Conditional inside
parenthesis,
uses ==, <=, >= or !
you can also nest
using && or ||

Action

if (this is true) { do this; }

```
void loop(){  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);  
  
  // check if the pushbutton is pressed.  
  // if it is, the buttonState is HIGH:  
  if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);  
  }  
  else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```

Action that occurs if
conditional is true,
inside of curly brackets,
can be anything,
even more if statements

Else

else { do this; }

```
void loop(){  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);  
  
  // check if the pushbutton is pressed.  
  // if it is, the buttonState is HIGH:  
  if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);  
  }  
  else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```

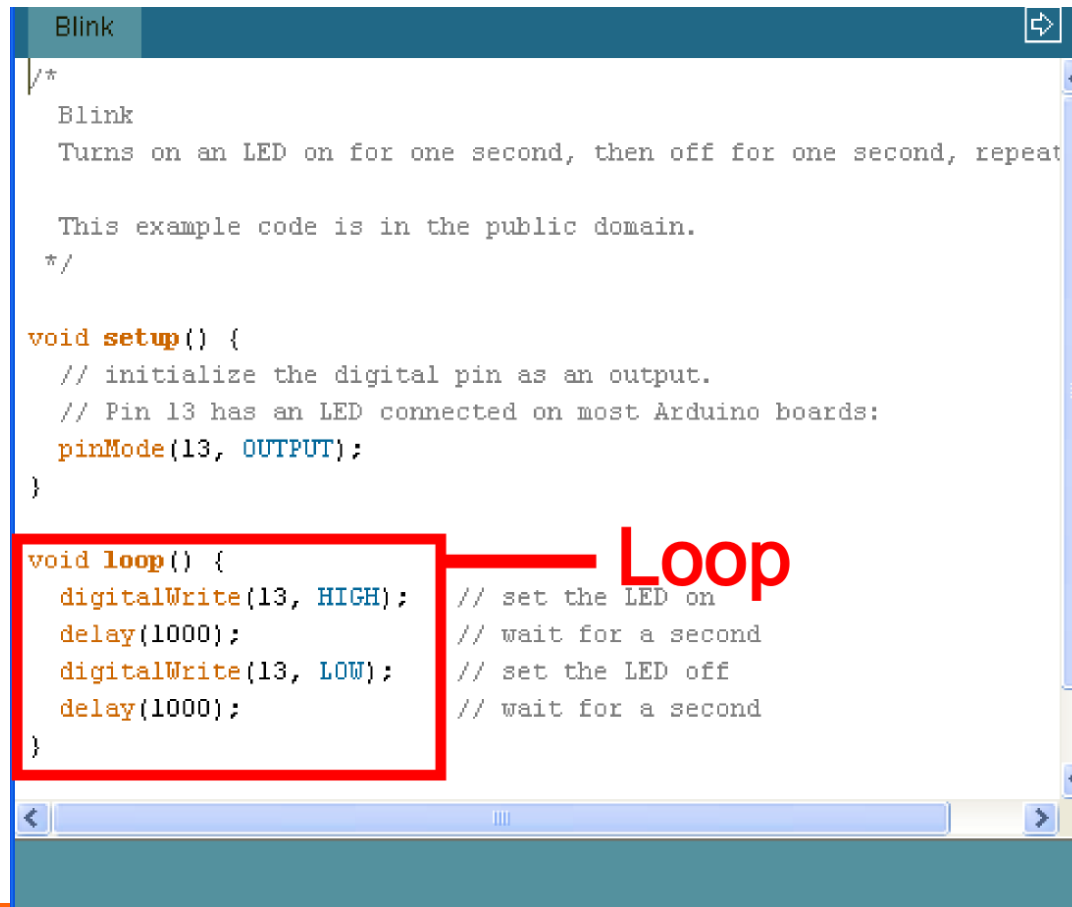
— Else, optional

Bucles comunes

- loop
- For
- while

Repetición básica de la placa

void loop () { }



```
Blink

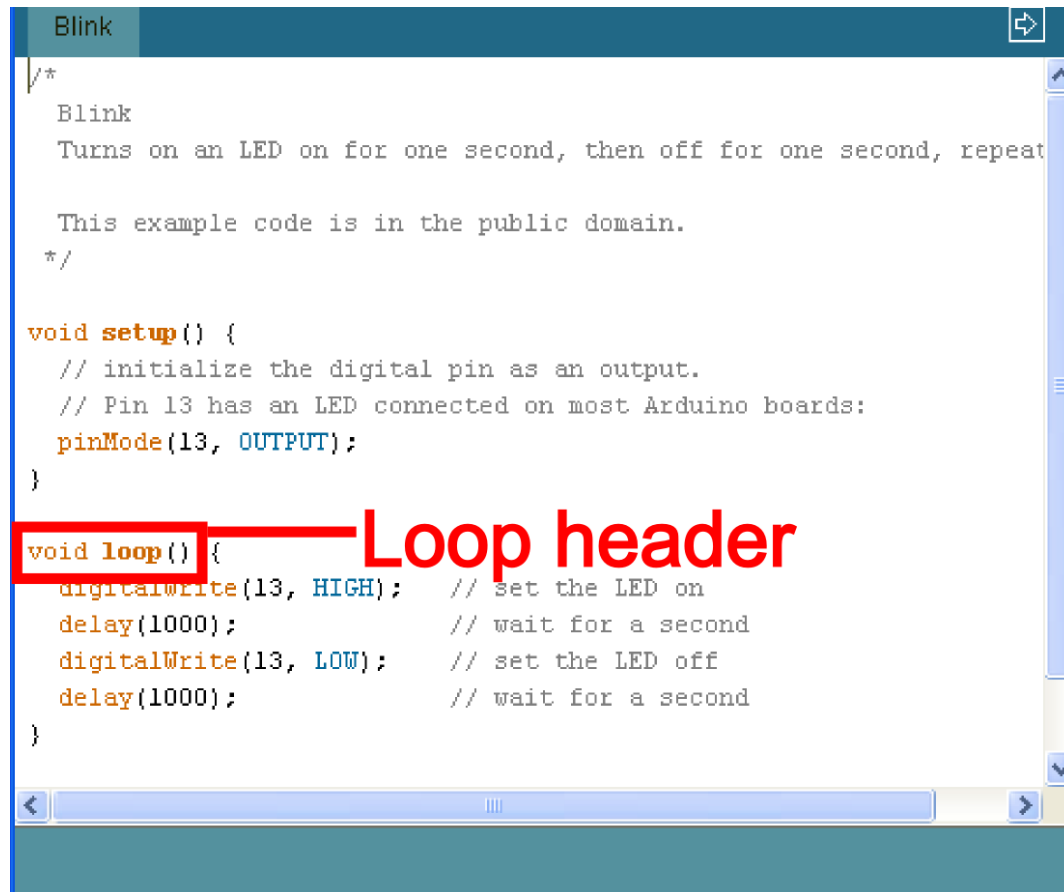
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeat
 *
 * This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);             // wait for a second
}
```

Repetición básica

void loop () { }



```
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeat
 *
 * This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}
```

Bucle básico

void loop () {}

```
Blink
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeat

  This example code is in the public domain.
  */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);             // wait for a second
}
```

Loop body
between curly
brackets

Bucle for

```
for (int count = 0; count<10; count++)  
{  
//cuerpo del bucle  
}
```

```
void setup()  
{  
    //Set each pin connected to an LED to output mode (pulling high  
    for(int i = 0; i < 8; i++){  
        pinMode(ledPins[i],OUTPUT); //we use this to set each LED p  
    }  
    //the code this replaces is
```

— For loop

```
/* (commented code will not run)  
 * these are the lines replaced by the for loop above they do ex  
 * same thing the one above just uses less typing
```

```
pinMode(ledPins[0],OUTPUT);
```

```
pinMode(ledPins[1],OUTPUT);
```

```
pinMode(ledPins[2],OUTPUT);
```

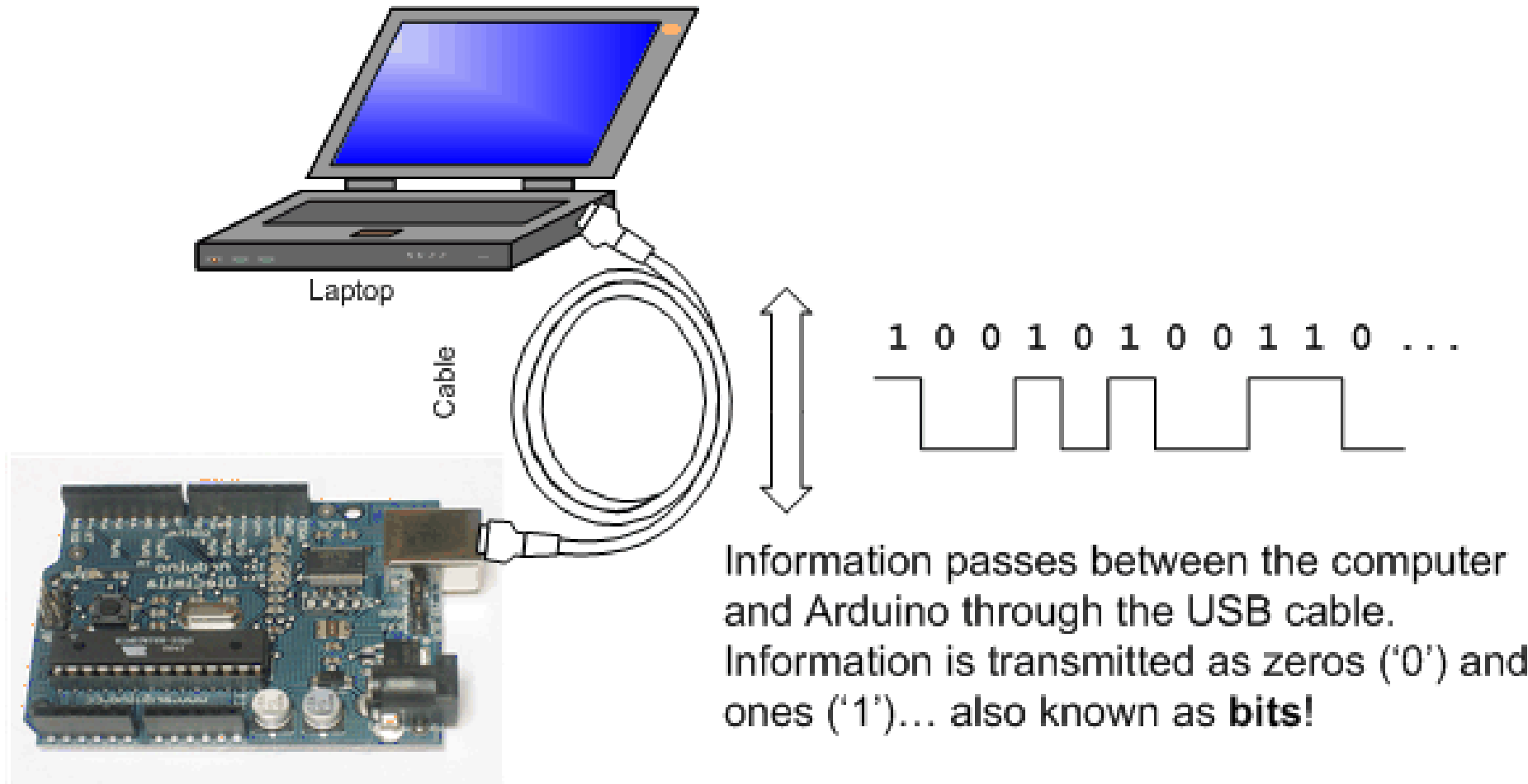
Bucle while

```
while ( count < 10 )  
{  
//cuerpo del bucle  
}
```

Bucle while

```
while ( digitalRead(buttonPin)==1 )  
{  
//la condición puede ser también el  
valor de una entrada  
}
```

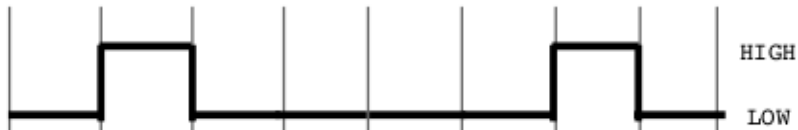

Comunicación serie



Serial Communications

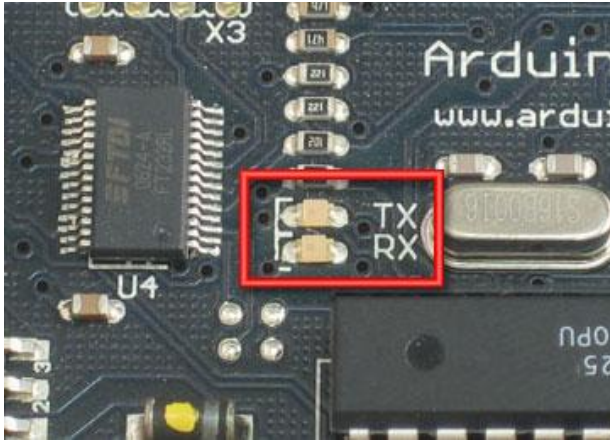
- “Serial” because data is broken down into bits, each sent one after the other down a single wire.

- The single ASCII character ‘B’ is sent as:

‘B’ = 0 1 0 0 0 0 1 0
= L H L L L L H L
= 

- Toggle a pin to send data, just like blinking an LED
- You could implement sending serial data with `digitalWrite()` and `delay()`
- A single data wire needed to send data. One other to receive.

Comunicación serie

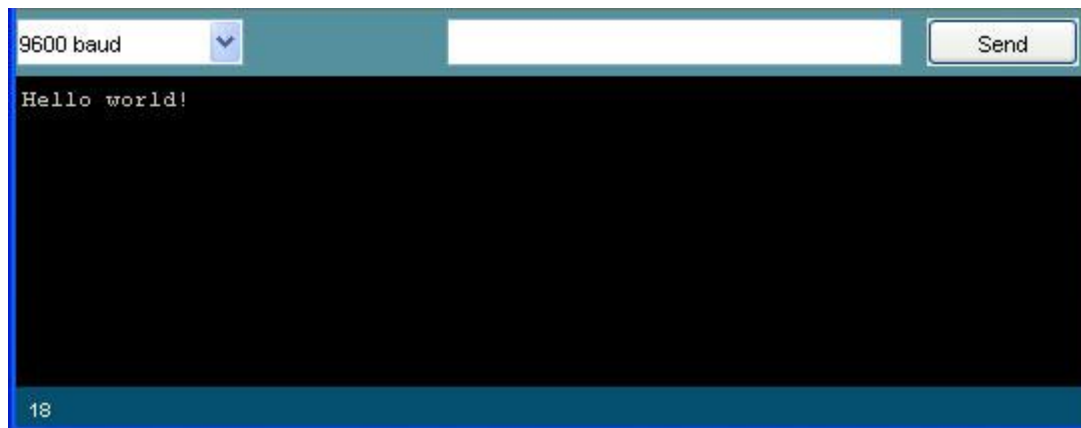


- ***Al compilar se obtiene una serie de bits***
- ***Los bits se envían en serie por el puerto USB a Arduino desde el PC***
- Dos LEDs junto al conector USB parpadean
 - **RX** Arduino recibiendo
 - **TX** Arduino transmitiendo

Programación con comunicación serie

```
/*  
 * Hello World!  
 * From www.ladyada.net  
 * It shows how to send data to the computer  
 */  
  
void setup()                                // run once, when the sketch  
starts  
{  
    Serial.begin(9600);                     // set up Serial library at 9600  
bps  
    Serial.println("Hello world!");         // prints hello with a line  
break  
}  
  
void loop()                                // run over and over
```

Abrir el monitor serie y luego cargar el programa



Ejemplos de cosas que se pueden hacer en el monitor serie

- *Mover `Serial.println("Hello world!");` a `loop()`*
- *Añadir en `setup()`:*

```
int a = 5;
```

```
int b = 10;
```

```
Serial.print("a + b = ");
```

```
Serial.println(a + b);
```

- *O poner en su lugar este código:*

```
int val = 33;
```

```
Serial.print(val);
```

```
Serial.print(val, DEC);
```

```
Serial.print(val, BIN);
```

-