Volver al índice de tutoriales

# Arduino IDE: Programa la placa Hello Button + LED

Este tutorial explica como programar el microcontrolador ATtiny44A de la placa Hello Button + LED que fabricamos en la práctica de Diseño de Circuitos usando el software Arduino (version 1.0.3). El ATtiny44A es un microcontrolador pequeño y barato muy útil para trabajar con programas sencillos.

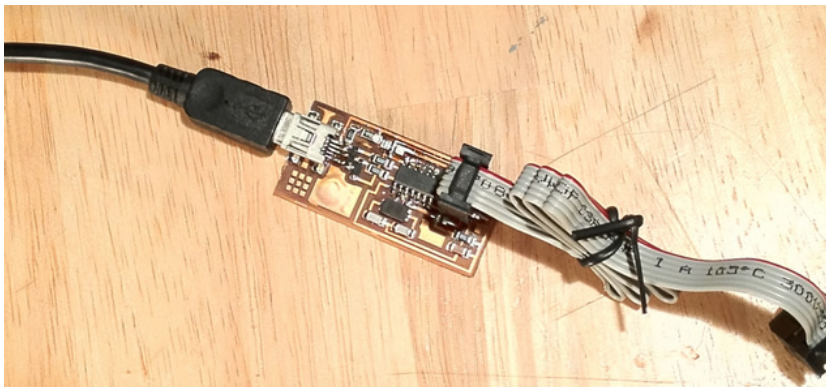
*David Mellis escribió las instrucciones originales:* http://highlowtech.org/?p=1695 *Éstas han sido revisadas, escritas de nuevo y ampliadas aquí para ser utilizadas* para la placa Hello Button + LED.

## *Índice:*


- Materiales y herramientas
- Descargar el software
- Pines del microcontrolador ATtiny 44A
- Añadir soporte para el ATtiny a Arduino
- Conectar las placas
- Configurar el ATtiny para trabajar a 20 MHz
- Seleccionar tu programador
- Si estás usando Ubuntu 12.04 LTS (como configurar avrdude como administrador desde Arduino IDE)
- Programar el Bootloader (todos los sistemas operativos)
- Modificar el ejemplo "Button"
- Programar el ATtiny
- Ejemplo de código
- Intenta modificar el código
- Funciones de Arduino soportadas / Limitaciones
- A continuación - Programa la placa en C para Echo Keyboard
- A continuación - ¡Construye un Fabduino!


## Materiales y herramientas

Necesitarás un programador ISP, usa el FabISP que fabricamos en clase.



## *Descargar el software*

Descarga el software de Arduino

Necesitarás el software de Arduino. Puedes descargar Arduino desde la página web de Arduino.

Aquí puedes encontrar las instrucciones de instalación.

Descarga los archivos para arduino del ATtiny

Attiny.zip

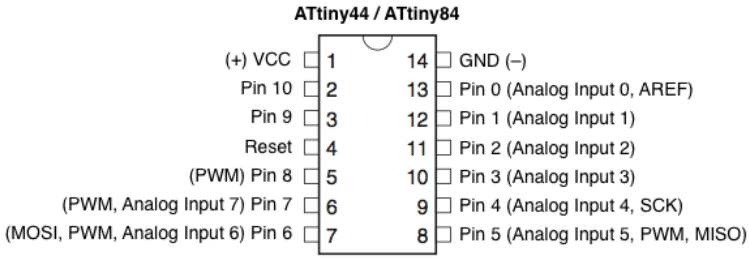
Instala los drivers para el cable FTDI

Descarga los drivers para tu sistema operativo aquí: http://www.ftdichip.com/Drivers/VCP.htm

## *Pines del microcontrolador ATtiny 44A*

Fijaté en que la numeración de los pines del microcontrolador es diferente a la numeración de Arduino. El número del pin del microcontrolador está indicado en el dibujo del microcontrolador que aparece a continuación. El pin correspondiente en el Arduino está indicado en la parte exterior del dibujo, a la derecha o a la izquierda.

Aquí se encuentra el enlace a la ficha técnica.



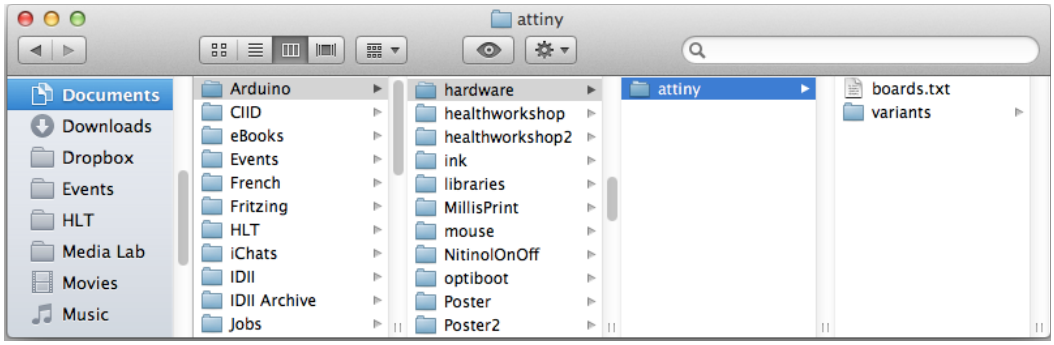

Equivalencia de los pines del ATtiny 44A y los pines de Arduino

| Número de pin ATtiny 44A | Número de pin correspondiente en Arduino | Detalles |
|---|---|---|
| 1 | No number, no access? | VCC (+) |
| 2 | Pin 10 | |
| 3 | Pin 9 | |
| 4 | No number, no access? | Reset |
| 5 | Pin 8 | PWM |
| 6 | Pin 7 | PWM, Analog Input 7 |
| 7 | Pin 6 | MOSI, PWM, Analog Input 6 |
| 8 | Pin 5 | Analog Input 5, PWM, MISO |
| 9 | Pin 4 | Analog Input 4, SCK |
| 10 | Pin 3 | Analog Input 3 |
| 11 | Pin 2 | Analog Input 2 |
| 12 | Pin 1 | Analog Input 1 |
| 13 | Pin 0 | Analog Input 0, AREF |
| 14 | No number, no access? | GND (-) |

## *Añadir soporte para el ATtiny a Arduino*

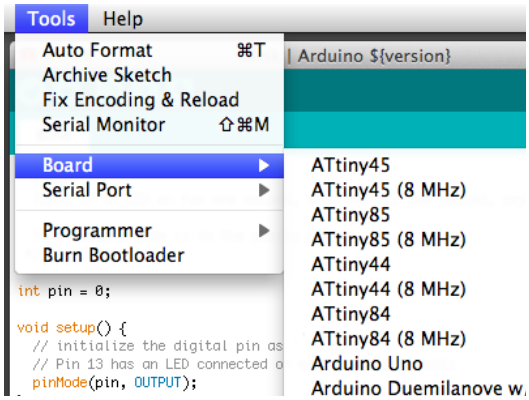
Descarga e instala los archivos de la placa

- Descarga lor archivos de la placa (de forma que el IDE de Arduino pueda interactuar con el ATtiny44A).
- Descomprímelo.
- Localiza la carpeta del Sketchbook del Arduino:
  - En el **software Arduino, en: > menú "Arduino" > preferencias**
- Busca la carpeta de Arduino “sketchbook”.
- Crea una nueva subcarpeta llamada "hardware" en la carpeta “sketchbook”.
- Copia la carpeta attiny extraída del archivo zip en la carpeta "hardware".

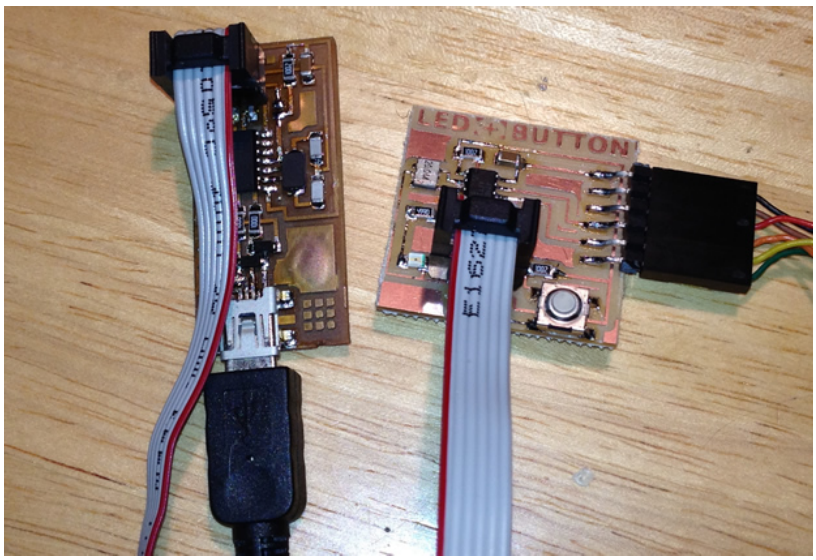



Reinicia

- Reinicia el Arduino IDE.
- Deberías ver el acceso al ATtiny en Tools > Board menu.

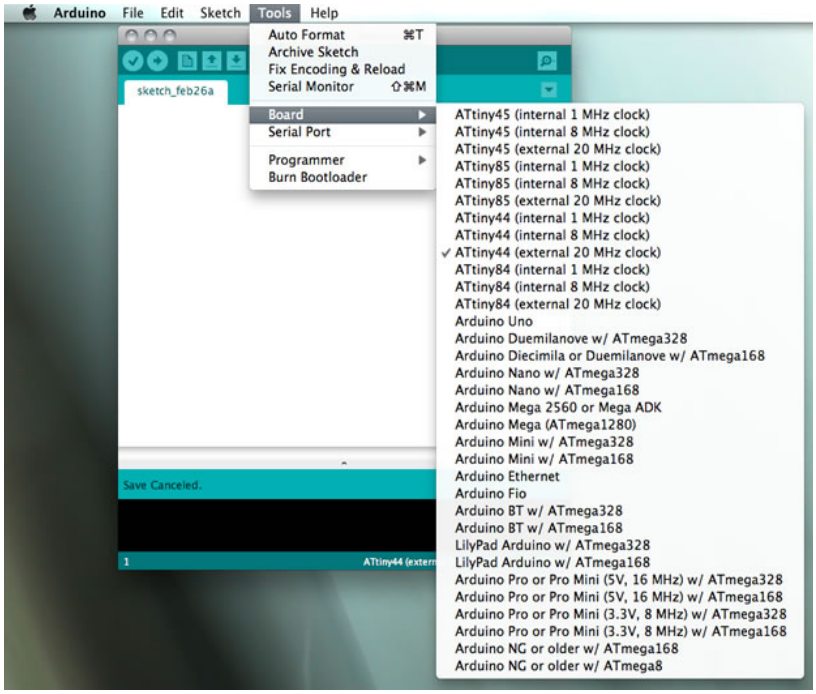



### *Conectar las placas*



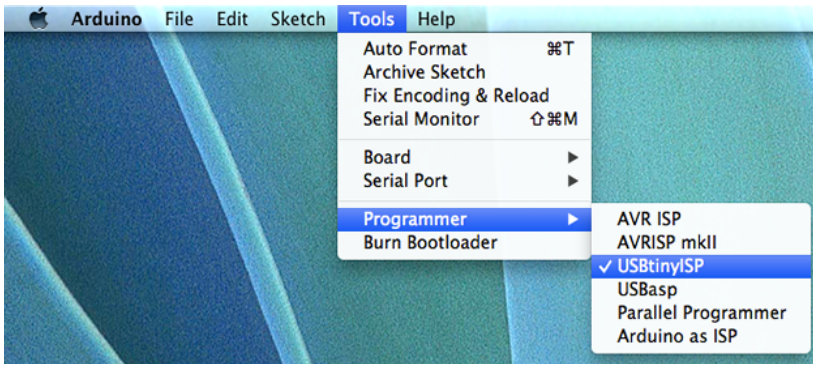

- Necesitarás alimentar el ATtiny y conectarlo a tu programador (FabISP).
- Esto significa conectar MISO, MOSI, SCK, RESET, VCC y GND del programador a los pines correspondientes del ATtiny.
- Conecta el programador al conector ISP de la placa LED + Button
- También necesitarás alimentar la placa LED + Button con un cable FTDI.

### *Configurar el ATtiny para trabajar a 20 MHz*




- Por defecto, el ATtiny trabaja a 1 MHz (la configuración usada por el ATtiny44 sin modificar)
- Necesitas dar un paso más y configurar el microcontrolador para que trabaje a 20 MHz – ya que tenemos un resonador de 20 MHZ en la placa Hello Button + LED.
- Selecciona el "**ATtiny44 (reloj externo de 20 MHz)**" del menú de la placa.

### *Selecciona tu programador*




- Selecciona el elemento apropiado de las herramientas > menú Board (fija el puerto serie para tu placa de Arduino).
- Selecciona el elemento apropiado de las **herramientas > menú Programmer > USBtinyISP**.

### *Si estás usando Ubuntu 12.04 LTS*

**Los usuarios de Mac y Windows no tienen que hacer esto.**

Avrdude necesita ejecutarse como administrador para enviar el código de Arduino a la placa. Necesitarás ejecutar avrdude como administrador.

**Si no lo haces, tendrás un mensaje de error como el que aparece a continuación:**

```
avrdude:  error: usbtiny_transmit: error sending control message:  Operation not permitted
avrdude:  initialization failed, rc=-1
              Double check connections and try again, or use -F to override
              this check.

avrdude:  error: usbtiny_transit: error sending control message:  Operation not permitted
```

Para habilitar el acceso como administrador:

Instala uisp (ausente en Ubuntu 12.04 LTS)

```
sudo apt-get install uisp
```

Para habilitar el acceso como administrador:

Necesitamos crear un archivo llamado 10-usbtinyisp.rules en el directorio **/etc/udev/rules.d.**

Escribe:

```
cd ~/Desktop
```

A continuación:

```
nano 10-usbtinyisp.rules
```

Se abrirá un archivo en blanco en un editor de texto.

Introduce el siguiente texto:

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="1781", SYSFS{idProduct}=="0c9f", GROUP="adm", MODE="0666"
SUBSYSTEM=="usb", ATTR{idVendor}=="1781", ATTR{idProduct}=="0c9f", GROUP="adm", MODE="0666"
SUBSYSTEM=="usb", ATTR{idVendor}=="1781", ATTR{idProduct}=="0c9f", GROUP="plugdev", MODE="0666"
SUBSYSTEM=="usb", ATTR{product}=="usbtiny", ATTR{idProduct}=="0c9f", ATTRS{idVendor}=="1781", MODE="0660", GROUP=
```

- Guarda el archivo usando: CONTROL + O.
- A continuación, presiona la tecla ENTER.
- Cierra el archivo usando: CONTROL + X

Mueve el archivo desde el escritorio a **/etc/udev/rules.d** escribiendo:

```
sudo mv 10-usbtinyisp.rules /etc/udev/rules.d
```

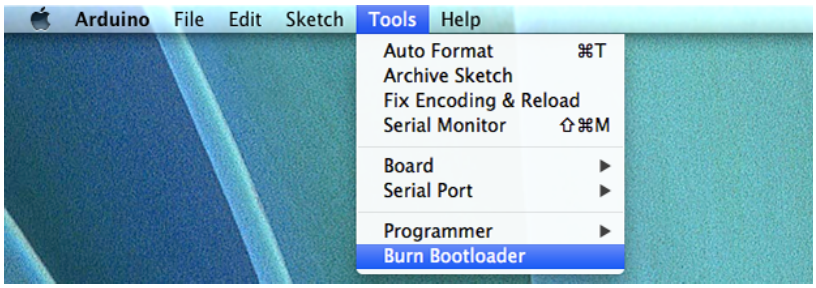
Entonces escribe:

```
sudo service udev restart
```

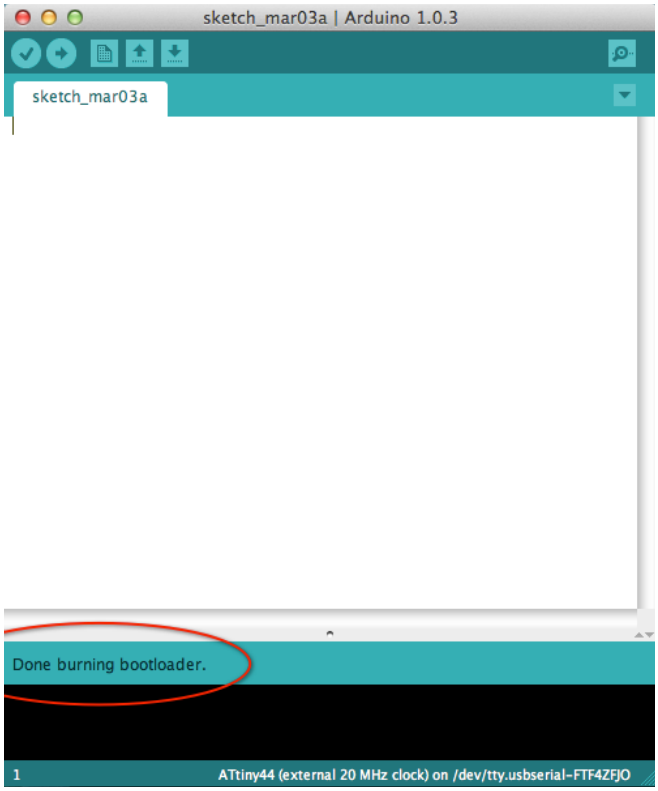
Desconecta todos los dispositivos USB. A continuación, conéctalos de nuevo.

Ahora deberías ser capaz de programar el Bootloader.

### *Programa el Bootloader (todos los sistemas operativos)*




- Selecciona el comando **"Burn Bootloader"** desde el menú Tools.
- Esto configura los fuses del microcontrolador para que trabaje a 20 MHz.
- Fíjate que:
  - los fuses mantendrán su valor hasta que decidas cambiarlos de manera explícita, **de manera que sólo tienes que realizar este paso una vez en cada microcontrolador.**
  - ésto no ha cargado un programa en la placa; necesitarás cargar nuevos programas usando un programador externo.




## *Modificar el ejemplo "Button"*

A continuación, podemos usar el IDE de Arduino para cargar un programa en el ATtiny.

### Modificar el Código

- Abre el sketch "Button" del menú "Examples".
- **File > Examples > Digital > Button**
- Cambia los números de los pines para que se correspondan con los pines usados para el LED y el botón de tu placa Hello Button + LED.

### Localiza este fragmento de código:

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin
```

### Cámbialo para igualarlos con los pins de tu placa Hello Button + LED:

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 3;     // the number of the pushbutton pin
const int ledPin =  7;      // the number of the LED pin
```

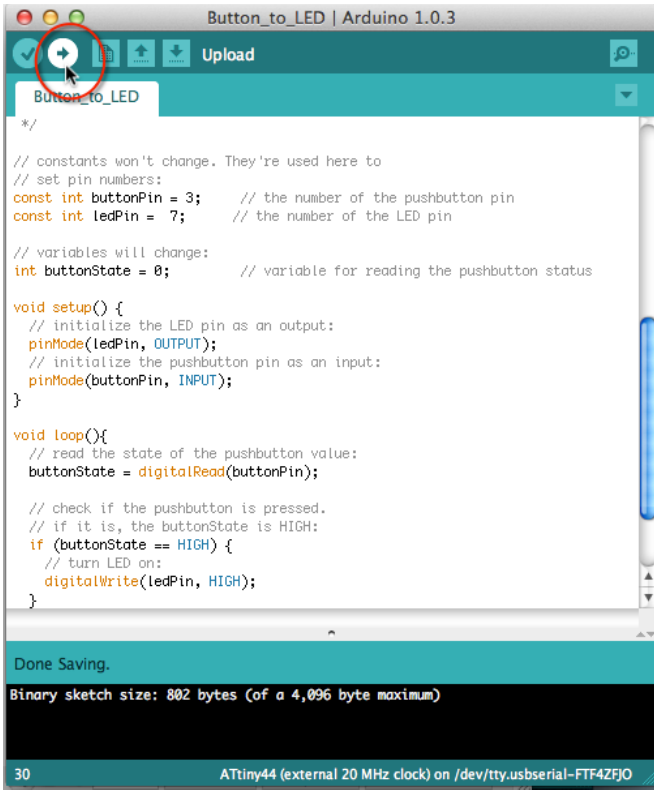
## *Programar el ATtiny*

### Conecta las placas

Aseguraté de que tu placa está conectada al ordenador a través del cable FTDI y de que tu FabISP está conectado a los 6 conectores de la placa Hello Button + LED.

### Carga el sketch

Para cargar el sketch a la placa Hello Button + LED, selecciona el botón con la flecha en el Arduino IDE (se muestra abajo)




Deberías ver "Done uploading" en el software Arduino y ningún mensaje de error.

**El LED debería brillar en la placa. Al presionar en el botón, el LED debería apagarse.**

### *Ejemplo de Código*

Este es un ejemplo de código con comentarios explicativos de lo que va haciendo el código. Copia y pega el código que aparece a continuación.

```
 /*
 LED Off Until Button Pressed

 Blinks a light emitting diode(LED) connected to digital
 pin 7, when pressing a pushbutton attached to pin 3.


 The circuit:
 * LED attached from pin 7 to ground
 * pushbutton attached to pin 3 from +5V
 * 10K resistor attached to pin 3 to +5V
 * 10K resistor pulls pin 3 and the button to HIGH by default

 created 2005
 by DojoDave
 modified 30 Aug 2011
 by Tom Igoe
 modified for Hello Button + LED Board - 19 Mar 2012
 by Anna Kaziunas France

 */

// constants won't change.
// They're used here to set pin numbers:
const int buttonPin = 3;     // the number of the pushbutton pin
const int ledPin =  7;      // the number of the LED pin

// initialize variables:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // read the state of the pin the pushbutton is connected to:
  buttonState = digitalRead(buttonPin);
```

```
  // is the push button pressed?
  // if not pressed - the button state is HIGH
  // the pull up resistor the button / pin 3 makes the button state HIGH by default.
  if (buttonState == HIGH) {
    // turn LED off (LED is off by default)
    digitalWrite(ledPin, LOW);
  }
  //otherwise.....
  // button is pressed
  else {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
}
```

### *Intenta modificar el código*

Modifica el código para que el LED parpadee cuando presionas el botón. Pista - usa la función delay(). Ver el ejemplo "Blink" para más detalles.

El ejemplo "Blink" viene con el Arduino IDE - Está en: File > examples > basics > blink

### *Funciones de Arduino soportadas / Limitaciones*

Debido a las limitaciones de memoria de este microcontrolador, no soporta todas las funciones de Arduino.

Las funciones de Arduino que soporta son los siguientes:

- pinMode()
- digitalWrite()
- digitalRead()
- analogRead()
- analogWrite()
- shiftOut()
- pulseIn()
- millis()
- micros()
- delay()
- delayMicroseconds()
- SoftwareSerial (ha sido actualizado en Arduino 1.0.3)

## A continuación - Programa la placa en C para Echo Keyboard

Para más detalles sobre cómo hacer esto, ver:
**Programa la placa en C para Echo Back Keyboard**

## A continuación - ¡Construye un Fabduino!

Par más detalles sobre cómo fabricar un Arduino fabable con más memoria y funcionalidad, ver:
**Fabrica un Arduino para tu Proyecto Final, INGLES.**

Creative Commons License All content © 2013 Anna Kaziunas France (except where otherwise noted) Some rights reserved.
Licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
Traducción: Covadonga Lorenzo y Epifanio Lorenzo.