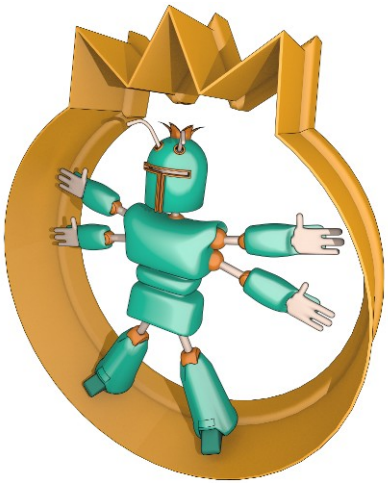
Granabot Badge

Guía de Programación



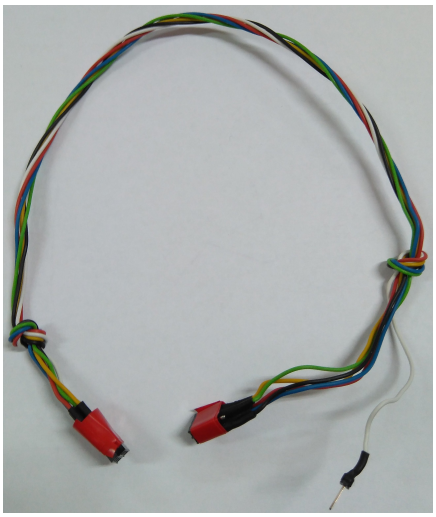
Indice

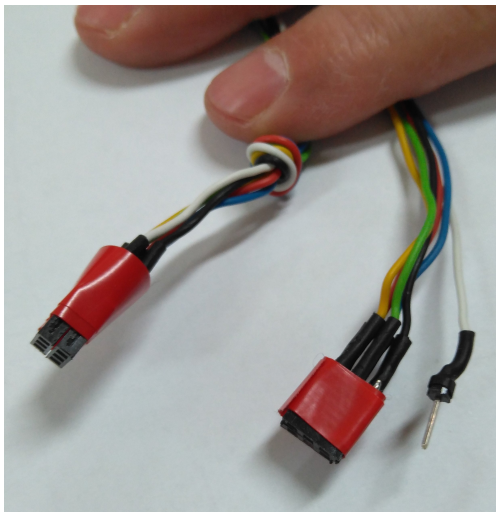
Introducción

Los materiales que necesitaremos, además del propio badge, son una placa Arduino UNO o similar, que hará las funciones de programador, un cable ICSP, que deberemos confeccionar nosotros mismos, y un cable USB, para conectar la placa Arduino al ordenador. En cuanto a los programas y librerías necesarias, usaremos el propio IDE de Arduino, descargable desde www.arduino.cc, y las librerías CapacitiveSensor, originalmente escrita por Paul Bagder y mantenida actualmente por Paul Sttofregen, y Adafruit_NeoPixel, de Adafruit. Ambas librerías las podemos instalar usando el gestor de librerías incluido en el IDE de Arduino.

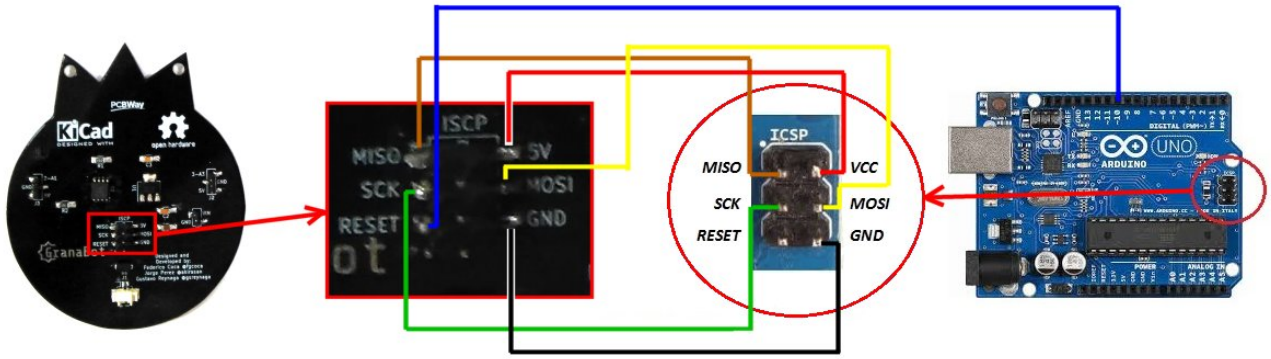
El cable ICSP

Nos va a permitir programar el ATTiny85 del badge a través del conector ICSP (In Circuit Serial Programming, programación serie en circuito). Consiste en un mazo o manguera de 6 cables. En uno de los extremos tiene un conector de 2x3 pines hembra, en nuestro caso construido a partir de dos tiras de 1x3 pines hembra unidas entre sí. En el otro extremo se encuentra otro conector similar al anterior pero en el que solo se usan 5 de los 6 pines. Al cable que en este extremo queda libre se le añade un pin macho.

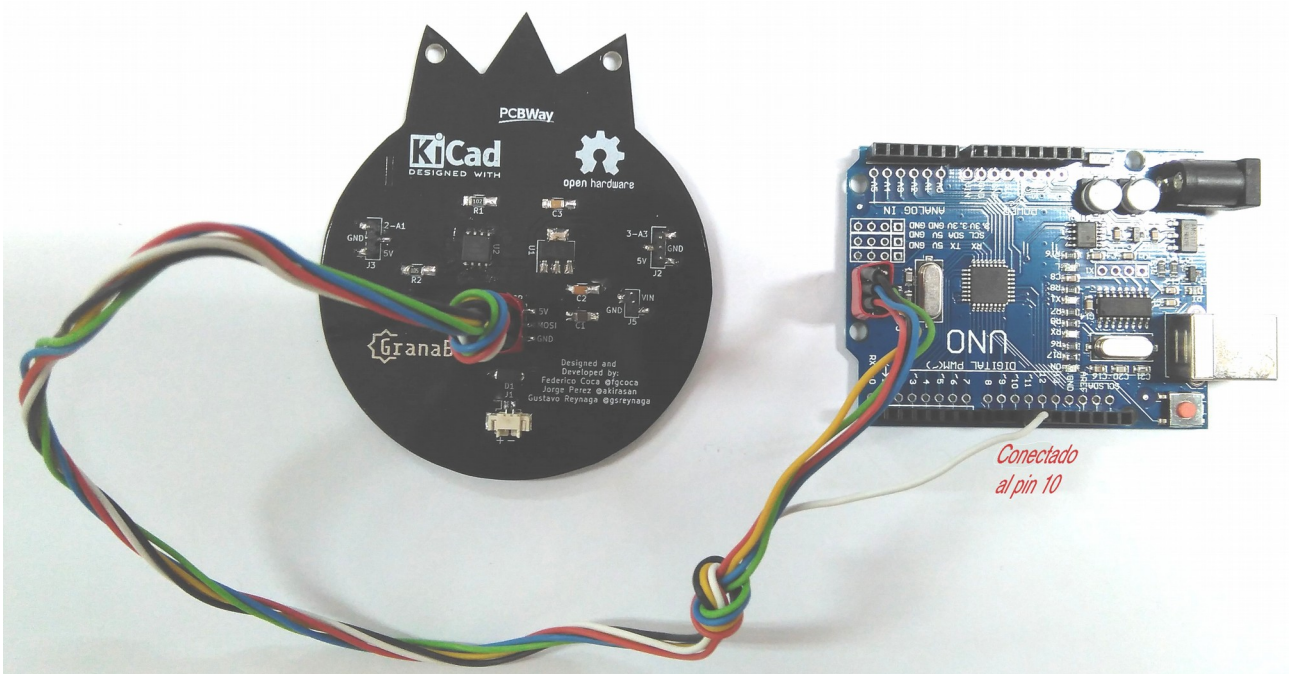




El conexionado entre el Badge y Arduino es el siguiente:

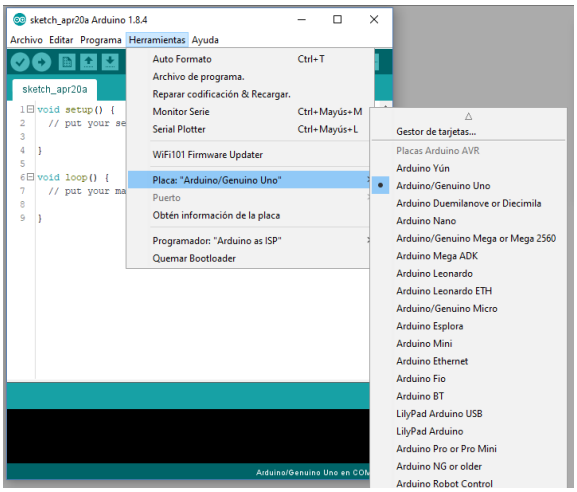


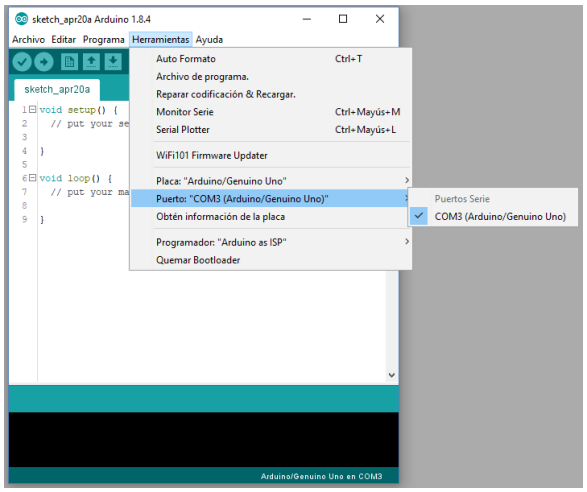
Una imagen real de este conexionado dará una mejor idea del mismo:



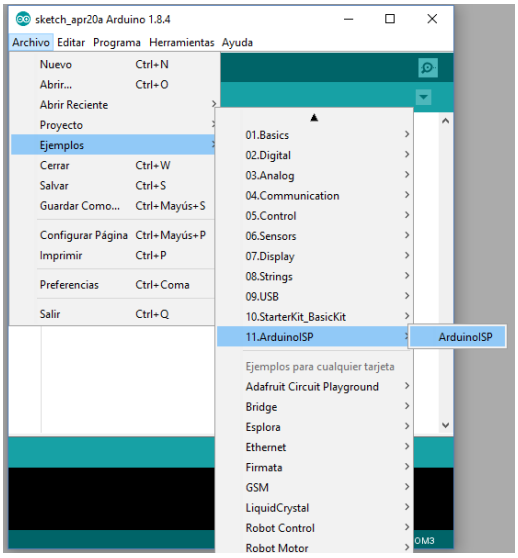
Preparación de Arduino para usarlo como programador

Para poder programar el badge usando un Arduino (UNO en este caso) deberemos realizar un proceso que empieza por conectar el Arduino al ordenador y arrancar el IDE. Los ajustes de dicho IDE deberán ser los habituales, es decir, la placa seleccionada debe ser Arduino UNO y el puerto el adecuado:





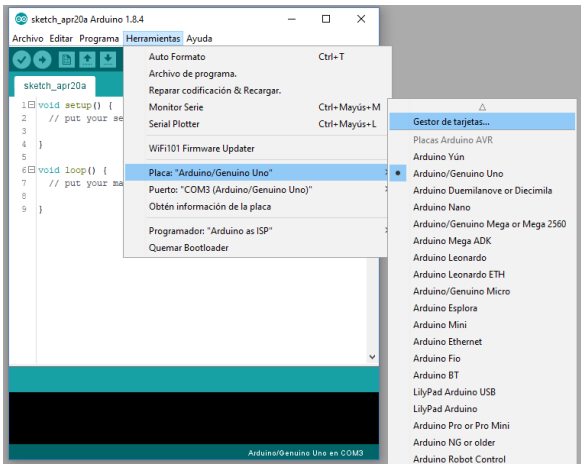
Seguidamente abriremos el ejemplo ArduinoISP:



Cargaremos este sketch en Arduino UNO sin modificar nada del mismo. Tras esto nuestro Arduino se habrá convertido en un programador ICSP.

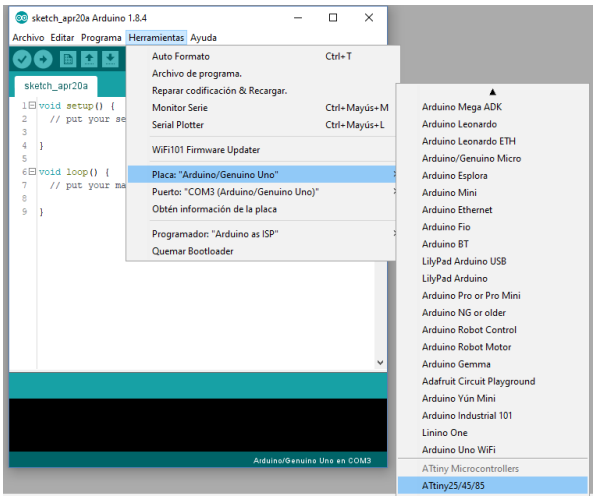
Preparar el IDE para trabajar con el ATTiny85

El ATTiny85 no forma parte de las “placas” que el IDE de Arduino es capaz de programar. Sin embargo, gracias a David A. Mellis que ha desarrollado los cores necesarios, podemos hacer que el IDE reconozca nuestro ATTiny85 para poder programarlo como si de una placa Arduino más se tratara. Para lograr esto deberemos empezar por abrir el gestor de tarjetas:

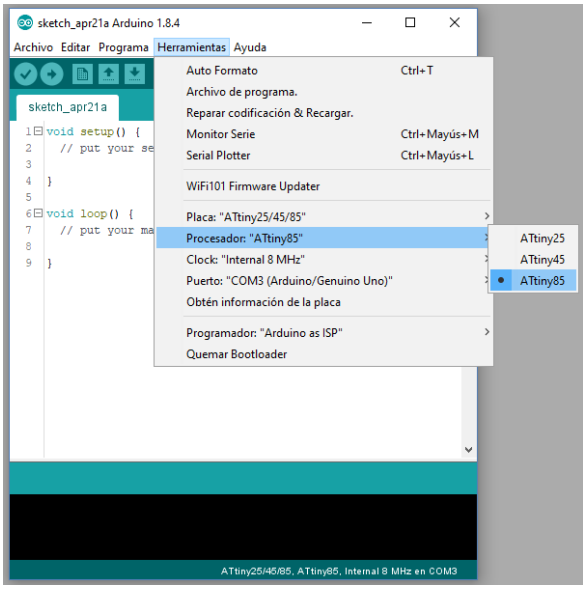


En dicho gestor buscaremos la tarjeta **attiny**:

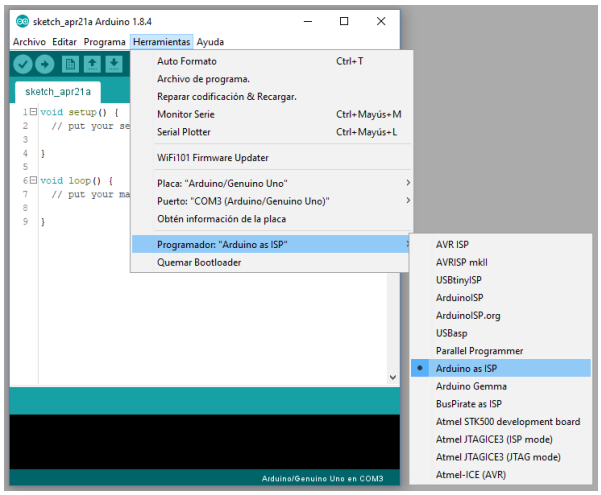
Seleccionaremos la tarjeta **attiny** by **David A. Mellis** y la instalaremos. A continuación seleccionaremos la tarjeta ATTiny85:



Tras hacer esto debemos seleccionar el ATTiny85:

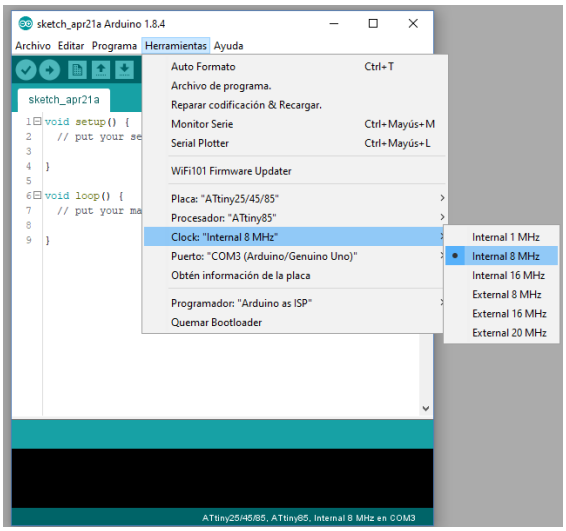


Por último deberemos indicarle al IDE que queremos usar el Ardunio UNO como programador del badge:

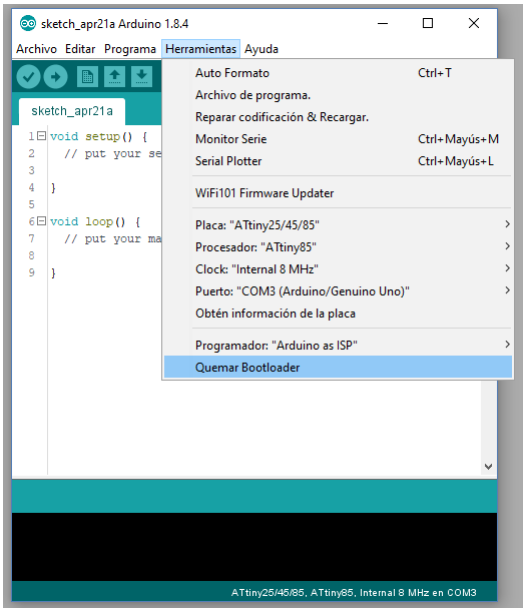


Cambiar los fuses del ATTiny85

El ATTiny85 viene de fábrica con sus fuses programados de tal forma que está configurado para trabajar con el reloj interno a 1MHz. Con esta configuración por defecto el badge no puede usar la librería Adafruit Neopixel. Es por esto que en GranaBot suministramos el ATTiny85 del badge configurado para que use el reloj interno a 8MHz. Si se tiene necesidad de cambiar esto sólo hay que empezar por seleccionar la opción adecuada:



Y proceder a programar los fuses:



Uso básico de la librería CapacitiveSensor

Es la librería que podemos usar para leer el sensor capacitivo incluido en el badge. Esta librería trabaja con un objeto de tipo CapacitiveSensor. Dicho objeto viene a ser una “digitalización” del sensor físico existente en el badge. Es decir, es una especie de representación digital abstracta, existente en el interior de Arduino, y que se corresponde con el sensor físico, real, disponible en la placa (Muy al estilo de Tron, una película estadounidense de 1982, del género ciencia ficción escrita y dirigida por Steven Lisberger).

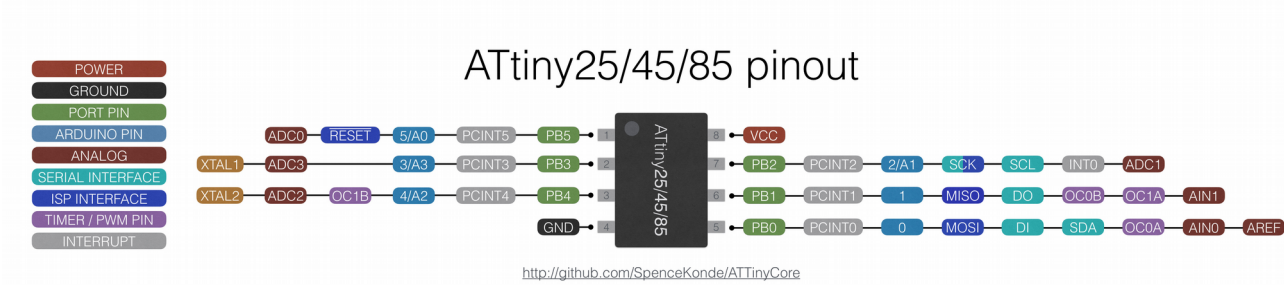
El primer paso para trabajar con esta librería es incluirla como fichero de cabecera con la directiva #include :

#include <CapacitiveSensor.h>

Hecho esto deberemos continuar por crear una instancia de la clase CapacitiveSensor. Esta instancia es el objeto que representa al sensor físico del badge, tal como se ha comentado más arriba:

CapacitiveSensor mi_sensor = CapacitiveSensor (pin 1, pin2);

El objeto creado se llama en este caso mi_sensor y, debido al hardware del badge, el pin 1 deberá ser el pin PB0 y pin 2 el pin PB1.



Para la lectura del sensor usaremos la función CapacitiveSensor(muestras). Esta función devuelve el valor de la capacidad que tiene el sensor (devolverá un valor bajo si no se está tocando el sensor) en unidades arbitrarias. El parámetro muestras específicamente cuantas muestras sucesivas se toman para determinar la capacidad del sensor durante la medida. El resultado devuelto será una media de todos los valores muestreados. Por tanto, para realizar una medida de la capacidad del sensor procederemos de la siguiente forma:

long medida_sensor = mi_sensor.CapacitiveSensor (10);

De esta forma en la variable medida_sensor se almacena el valor medio resultado de 10 medidas sucesivas de la capacidad del sensor. Si se toca el sensor este valor será más o menos elevado. Si no se toca será un valor bajo.

Uso básico de la librería Adafruit_NeoPixel

Usaremos esta librería para el control de los LEDs RGB incluidos en el badge. Esta librería solo funciona con frecuencias de reloj del micro-controlador iguales o mayores de 8MHz. Como siempre, para poder usar las funciones disponibles en la librería deberemos empezar por incluir la misma en nuestro sketch:

#include <Adafruit_NeoPixel.h>

También, como en el caso descrito anteriormente, será necesario instanciar la clase Adafruit_NeoPixel para crear el objeto que representa a la "tira de LEDs" incluida en el badge:

Adafruit_NeoPixel mis_LEDs = Adafruit_NeoPixel (8, PB4, NEO_GRB + NEO_KHZ800);

El 8 se debe a que nuestro badge tiene 8 LEDs RGB. PB4 es el pin del ATTiny encargado de controlar los LEDs RGB. El resto… bueno, es lo que hay que poner en el caso que nos ocupa (ver https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use o sketch de ejemplo strandtest.ino incluido al instalar la librería para más detalles).

Una vez creada nuestra "imagen virtual" de los LEDs del badge en el interior del ATTiny85 ya podemos empezar a controlarlos mediante las funciones que nos proporciona la librería. Habrá que empezar por ajustar el brillo máximo de los LEDs e inicializar el objeto creado. Normalmente esto se hace dentro del setup del sketch:

mis_LEDs.setBrightness(brillo);

mis_LEDs.begin();

El valor dado al brillo máximo tiene que ser de tipo byte, o sea, entre 0 y 255.

Para controlar el color de cada uno de los LEDs del badge usaremos la siguiente función:

mis_LEDs.setPixelColor(n, rojo, verde, azul);

Esta función ajusta el color del LED número n (en nuestro caso n es un número entero entre 0 y 7). Los colores, rojo, verde y azul, son de tipo byte. El efecto de esta función no se refleja inmediatamente en los LEDs. Para que ello ocurra deberemos mandar la nueva configuración de color de los LEDs a los mismos empleando la función siguiente:

mis_LEDs.show();

Test de funcionamiento

En los enlaces siguientes podemos ver el video de funcionamiento del badge con el firmware grabado por defecto en el mismo.

Enlace al video en Youtube –> https://youtu.be/pwC2fTl5S1I

Enlace al video en Github