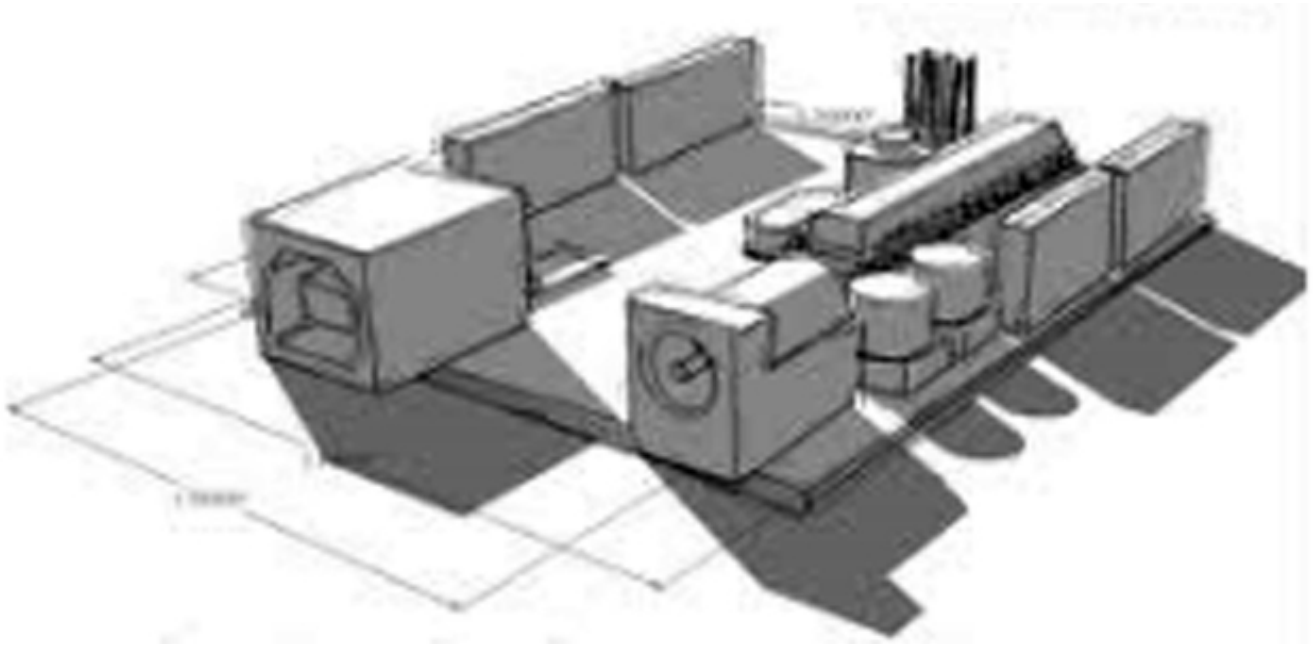
There are no translations available.

Descubre herramientas de trabajo con las que podremos iniciar a los alumnos en el campo denominado genéricamente de la “computación física”, es decir, el aprendizaje de la programación orientada a interactuar con dispositivos electrónicos sencillos.

Capítulo1
1) Introducción

En mayo del 2006 participé en un taller en Madrid para profesores de Secundaria en el que David Cuartielles nos mostró las posibilidades de la tarjeta Arduino para el mundo de la educación. En aquel momento, Arduino era casi un prototipo de una gran idea: crear una plataforma de open-hardware de fácil programación y destinada a crear dispositivos electrónicos con los que interactuar. Dicho de otro modo, comprando o construyendo una tarjeta Arduino (ensamblando sus componentes), y descargando en el ordenador la última versión de su software de programación desde *www.arduino.cc/es*, cualquiera puede disponer en el taller de Tecnología de un dispositivo barato, de uso sencillo y rápidos resultados, con el que introducir a los alumnos de Secundaria en la electrónica, la automática o la robótica.



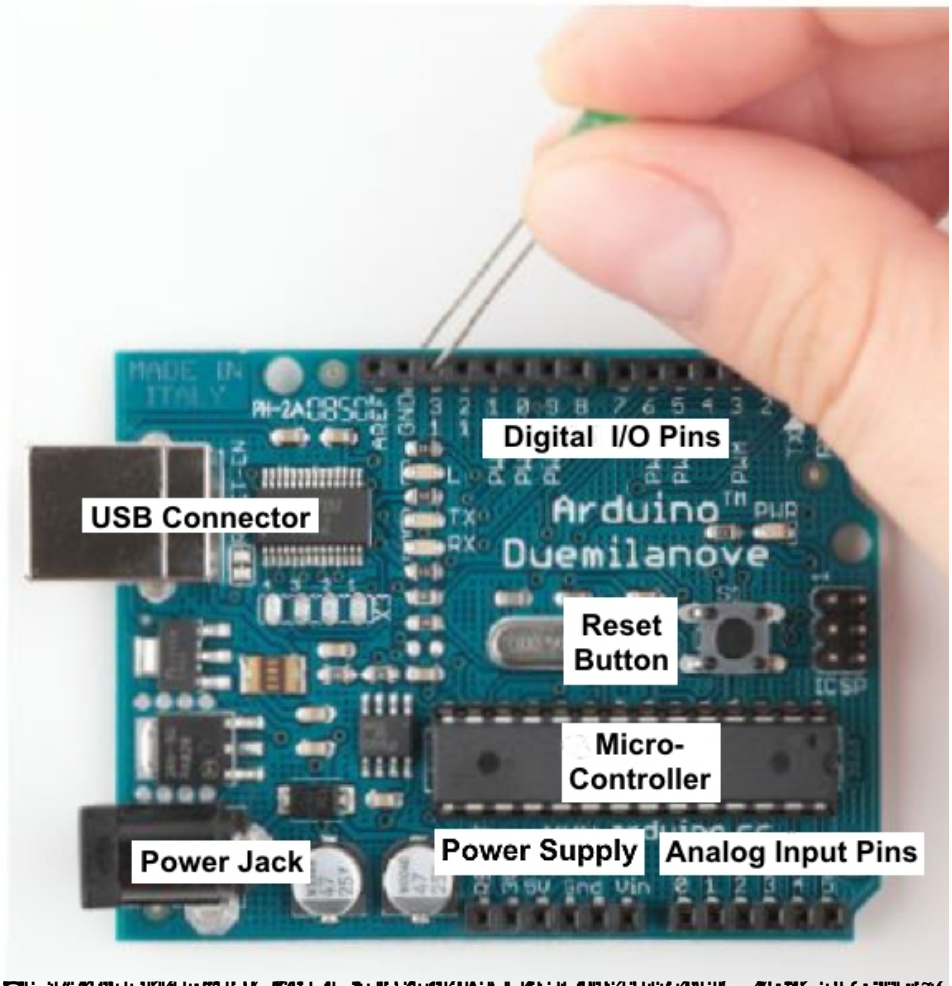
(imagen del grupolinda.org)

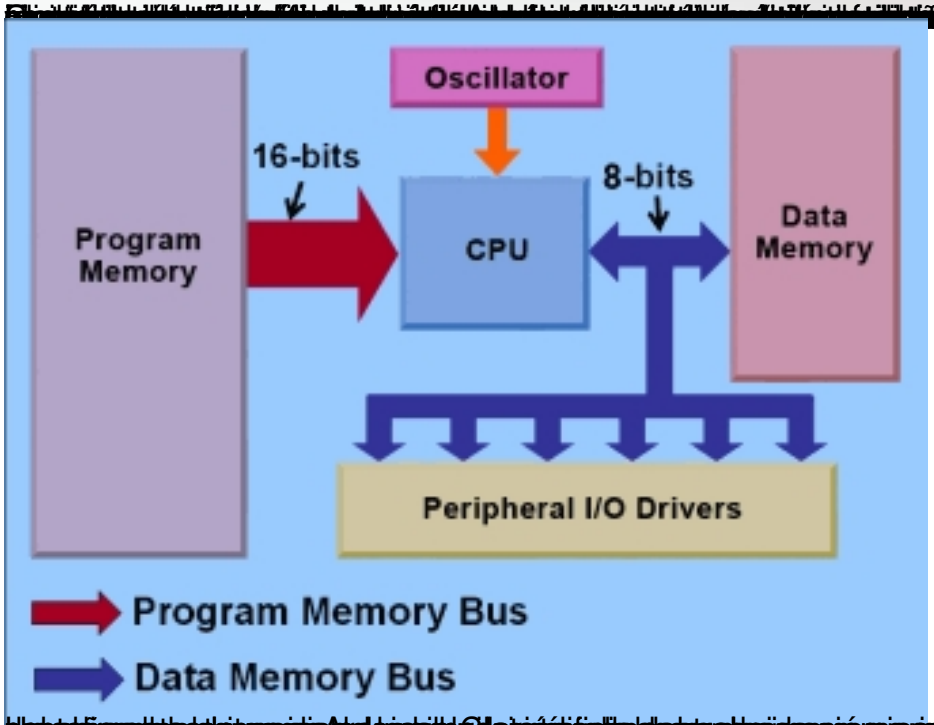
Sólo la excelente página web citada anteriormente, ya contiene una gran cantidad de información en castellano sobre la descripción de la tarjeta, su puesta en funcionamiento, dónde comprarla, cómo complementarla, así como tutoriales y recursos externos para iniciarse en su manejo. Los propios alumnos pueden ser convenientemente guiados a través de ella, de modo que desarrollen estrategias de autoaprendizaje para experimentar con la tarjeta.

Si además tecleamos Arduino en cualquier buscador, comprobaremos que existe en la web una extensa y activa comunidad internacional de usuarios dispuestos a intercambiar experiencias y programas, de todos los niveles de complejidad, realizados con Arduino en cualquiera de sus versiones comerciales (también existen desarrollos libres del diagrama electrónico base) o artesanales (como "*paperduino*"). Casi podemos afirmar que estamos ante el standard de las tarjetas programables con microprocesadores AVR Atmega, ya que tradicionalmente en la experimentación educativa se utilizan los microprocesadores PIC, mucho más difíciles de programar y para los que apenas existen soluciones de hardware abierto.

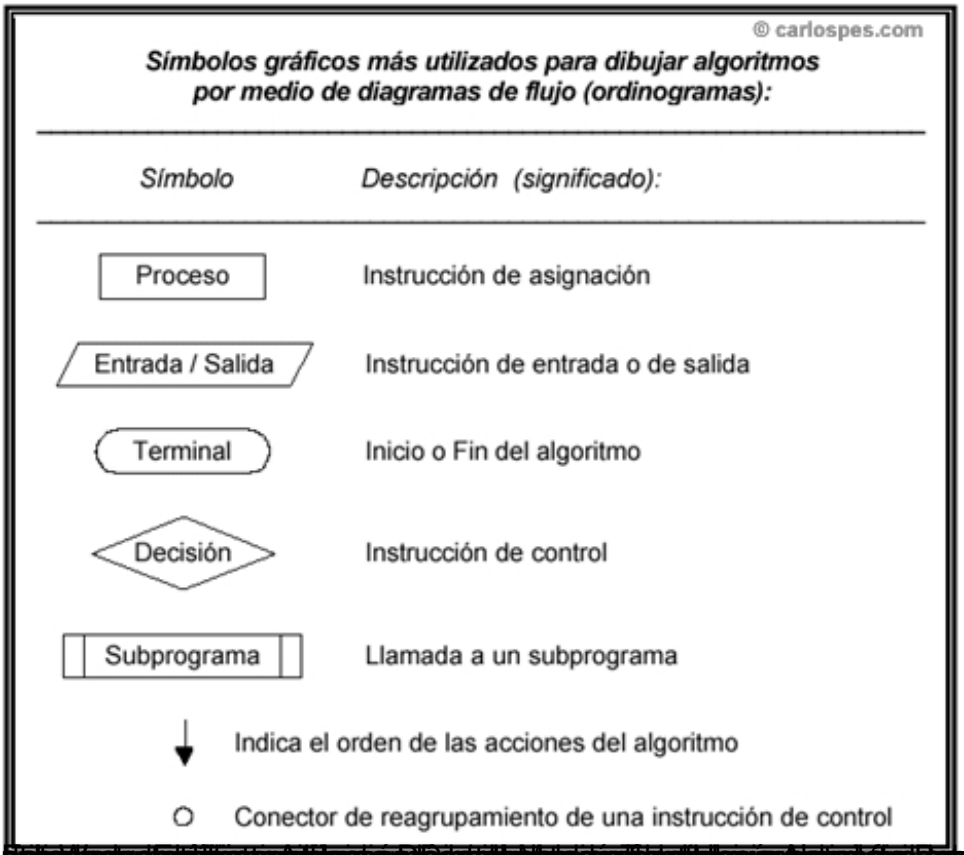
2) ¿Cómo es la tarjeta Arduino?

Al observar por primera vez una Arduino, posiblemente los elementos que más llaman la atención son los numerosos pines o agujeros de conexión hembra que tiene a ambos lados, además de la conexión USB (con la que podemos programar y activar la electrónica de la tarjeta), así como una entrada de alimentación (admite hasta 30V) con la que podemos independizar el microprocesador Atmega respecto del ordenador, una vez esté cargado nuestro programa. Volviendo a los pines hembra, hay que destacar que disponemos, a un lado, de 14 entradas o salidas digitales (se especificaría esta característica a través del software),así como la posibilidad de crear 6 salidas analógicas con los pines digitales 3, 5, 6, 9,10, 11, indicados como PWM (lo que hará Arduino es modular por pulsos),y al otro lado, de 6 entradas analógicas junto con pines de conexión a masa (GND) y a distintos valores de voltaje (necesarios para completar el circuito electrónico de los sensores externos con los que trabajemos).


Digital I/O Pins
USB Connector
Arduino
Duemilanove
Reset Button
Micro-Controller
Power Jack
Power Supply
Analog Input Pins


Oscillator
16-bits
8-bits
Program Memory
CPU
Data Memory
Peripheral I/O Drivers
Program Memory Bus
Data Memory Bus

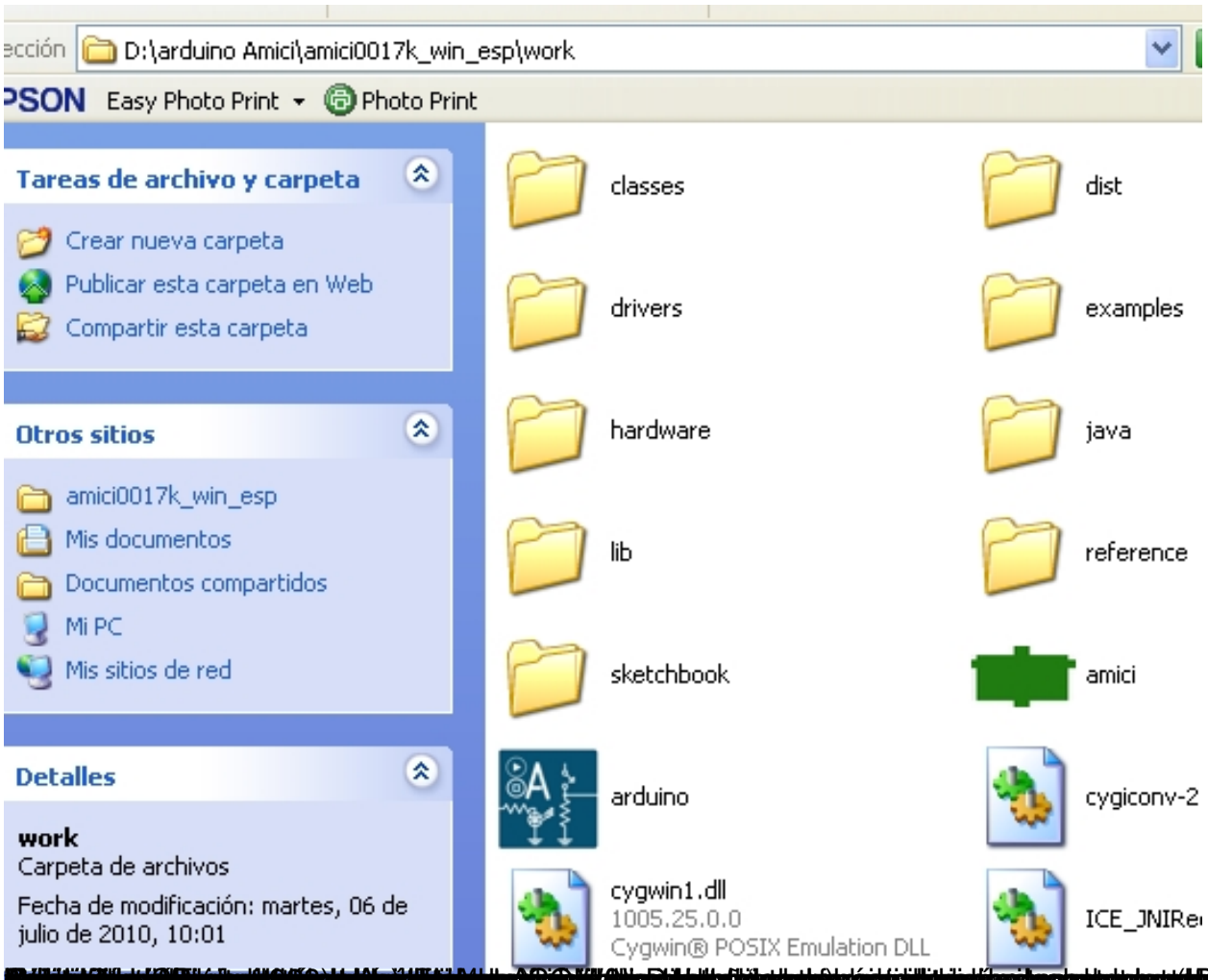
```
int Variable3l=500;
int Pinl3 = l3;

 void setup(){
pinMode(Pinl3, OUTPUT);

}
void loop(){
digitalWrite(Pinl3, HIGH);
delay(Variable3l);
digitalWrite(Pinl3, LOW);
delay(Variable3l);

}
```

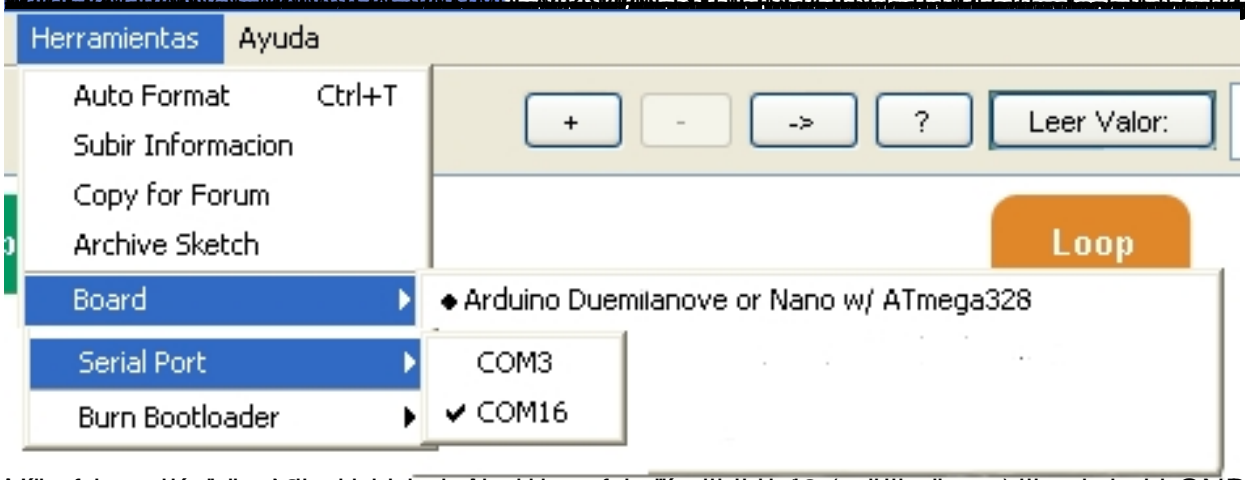


3) Primeros pasos

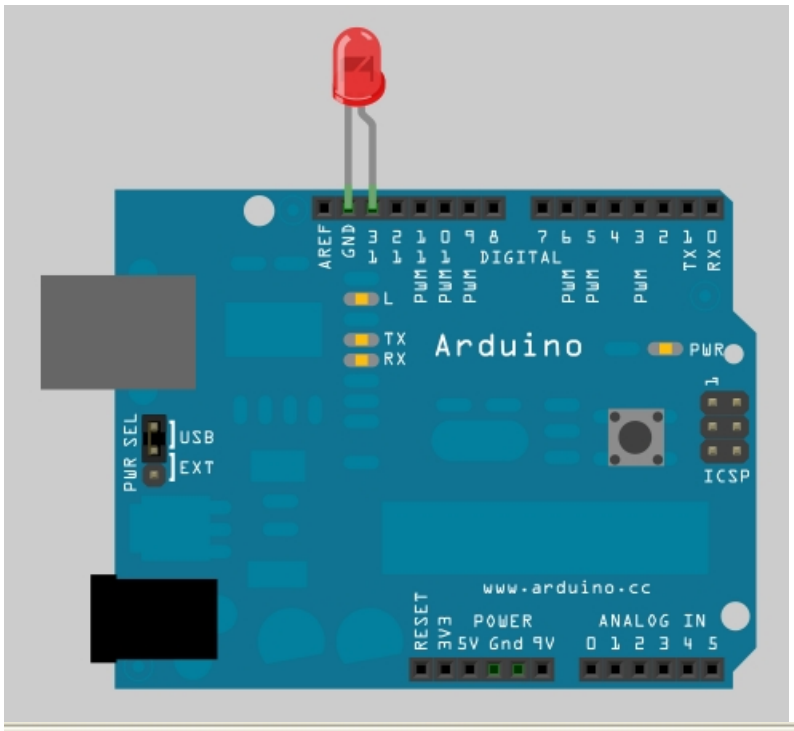
Aunque desde septiembre está disponible una versión beta de Amici0019, voy a desarrollar mi exposición utilizando la versión 0017, traducida al castellano y con la que he experimentado con mis alumnos durante el curso pasado; dejo para próximos capítulos la experimentación con la versión más reciente, disponible por el momento, sólo en inglés o en alemán.

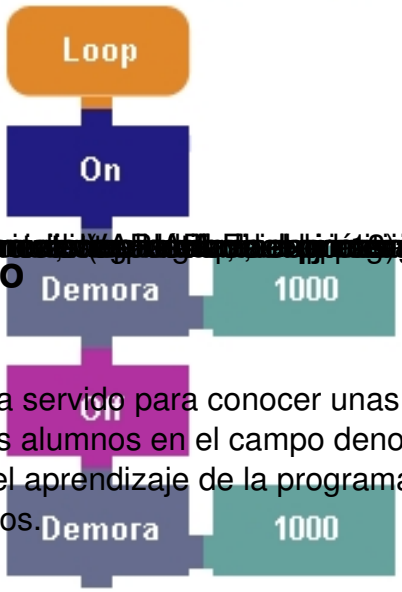
Lo primero que debemos hacer es descargar Amici para Duemilanove en nuestro disco duro (http://dimeb.informatik.uni-bremen.de/eduwear/category/development-software/ *)* y extraer la carpeta comprimida de nombre
amici0017k_win_esp .
Si es la primera vez que vamos a conectar al ordenador la tarjeta Duemilanove, se necesita cargar un driver para que sea reconocida como nuevo hardware, el cual se encuentra en la subcarpetas
drivers/ FTDI USB Drivers
de la carpeta amici0017k.


D:\arduino Amici\amici0017k_win_esp\work
Easy Photo Print
Photo Print
Tareas de archivo y carpeta
Crear nueva carpeta
Publicar esta carpeta en Web
Compartir esta carpeta
Otros sitios
amici0017k_win_esp
Mis documentos
Documentos compartidos
Mi PC
Mis sitios de red
Detalles
work
Carpeta de archivos
Fecha de modificación: martes, 06 de julio de 2010, 10:01
classes
dist
drivers
examples
hardware
java
lib
reference
sketchbook
amici
arduino
cygiconv-2
cygwin1.dll
1005.25.0.0
Cygwin® POSIX Emulation DLL


Herramientas
Ayuda
Auto Format Ctrl+T
Subir Informacion
Copy for Forum
Archive Sketch
Board
Serial Port
Burn Bootloader
Arduino Duemilanove or Nano w/ ATmega328
COM3
COM16
+
->
?
Leer Valor:
Loop

apero





4. Conclusión y anexo

El desarrollo de este capítulo ha servido para conocer unas herramientas de trabajo con las que podremos iniciar a nuestros alumnos en el campo denominado genéricamente de la “computación física”, es decir, el aprendizaje de la programación orientada a interactuar con dispositivos electrónicos sencillos.

En la página siguiente se anexa un cuadro resumen que explica las funciones incluidas en cada uno de los bloques visuales de programación.

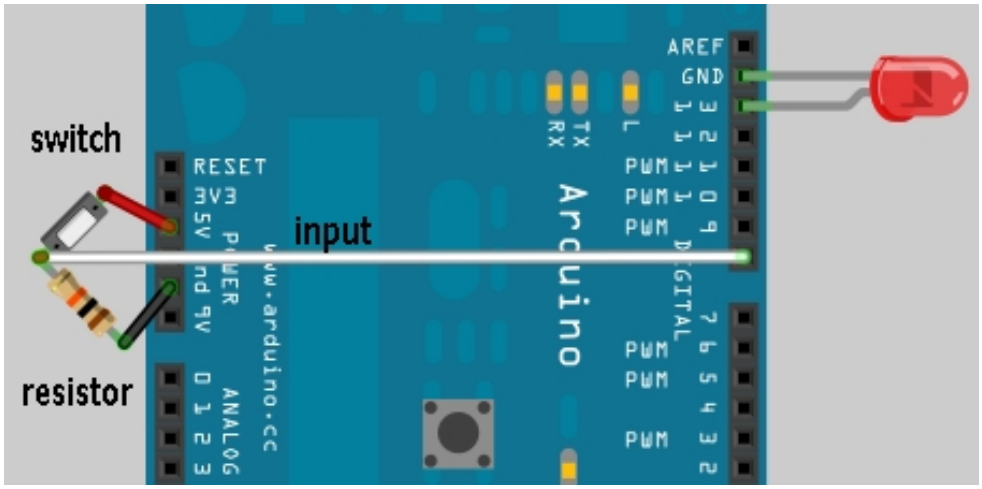

ArduinoBlocks
Archivo
Editar
Bosquejo
Herramientas
Ayuda
+
-
->
?
Leer Valor
On
conecta un actuador
inicia el programa principal
Loop
On por
crea una intermitencia, de tiempo variable, en un actuador
If
comprueba si cambia el valor detectado en un sensor y activa un bloque (
Esperar hast
activa un bloque "mientras que" se mantenga un valor fijado para el sens
Off
desconecta un actuador (led, motor, zumbador)
Sonido
crea un sonido (tono), durante un tiempo, en un piezoeléctrico
Leer Valor
escribe en la ventana el valor detectado en un sensor
Demora
indica que un bloque se activará durante un tiempo variable (a su derecha
Repetir
permite la repetición de una parte del programa un numero variable de ve
Variable
número necesario para determinar los tiempos o las repeticiones
Metodo
indica que un subprograma se ejecutará a partir de un momento dado
Metodo
define el nombre e inicio de un subprograma, con los bloques que contiene
Informacion
recolecta una serie de valores detectados por un sensor y los almacena en
Contador
cuenta las detecciones realizadas por un sensor y monitoriza su valor
papelera que permite eliminar de la pantalla los bloques en desuso

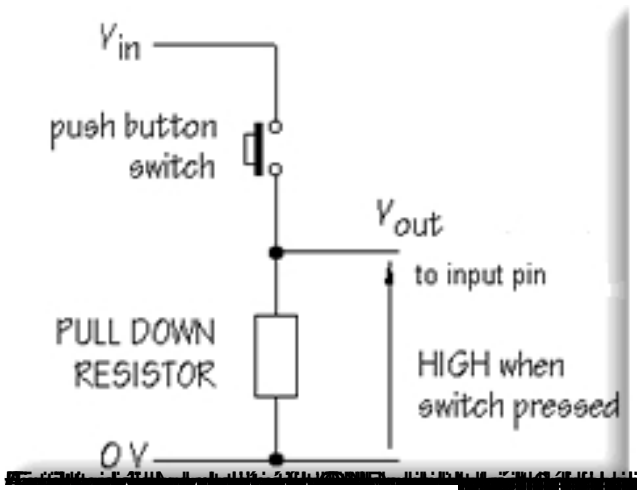
Capítulo 2
1) Introducción

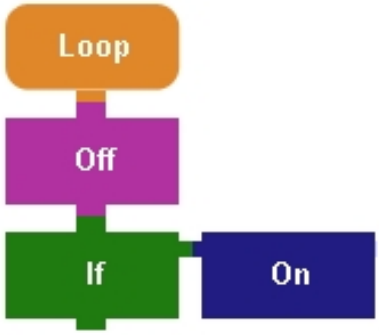
En el primer capítulo de este monográfico hemos conocido el entorno Arduino, su hardware y software, de modo que hemos probado a crear intermitencias en un diodo led. A partir de ahora vamos a incluir en nuestros ejercicios de programación, secuencias luminosas combinadas con la creación de sonidos en un zumbador piezoeléctrico, siendo su bucle normal de funcionamiento interrumpido mediante sensores conectados en las entradas de la tarjeta Arduino.

2) Conexionado de entradas digitales.

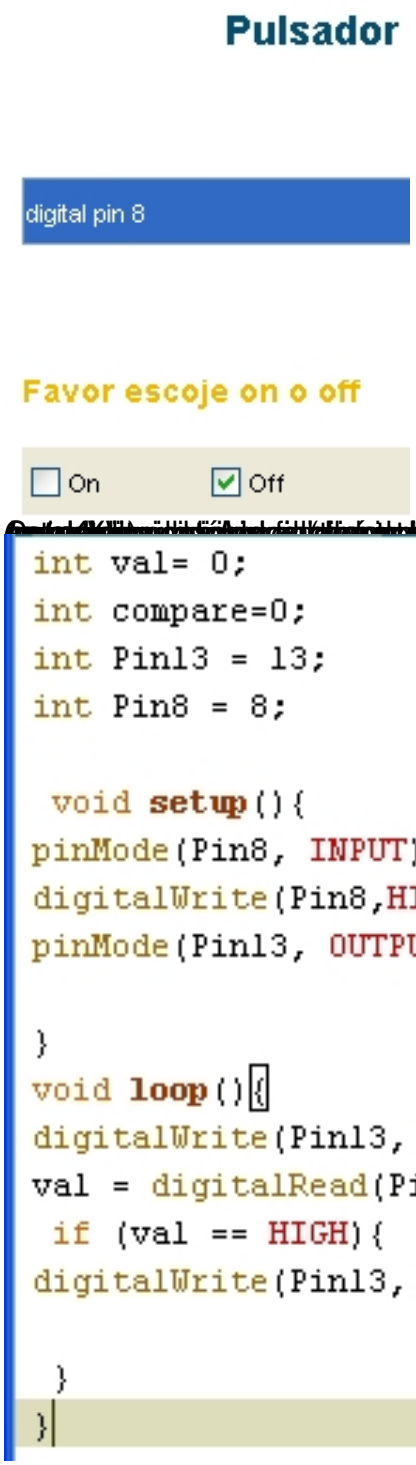
Para que un sensor digital o analógico pueda ser detectado en alguno de los pines correspondientes de la tarjeta, antes hay que establecer un circuito de alimentación desde 5V a GND (masa) que atraviese dicho sensor y que además produzca una caída de tensión o una débil corriente en una resistencia que se intercala entre el sensor y el pin de entrada de la tarjeta, formando todo ello un típico esquema electrónico denominado “pull-down”, necesario para proteger la tarjeta.




Vin
push button switch
Vout
to input pin
PULL DOWN RESISTOR
HIGH when switch pressed
0 V


Loop
Off
If
On


Loop
Off
On

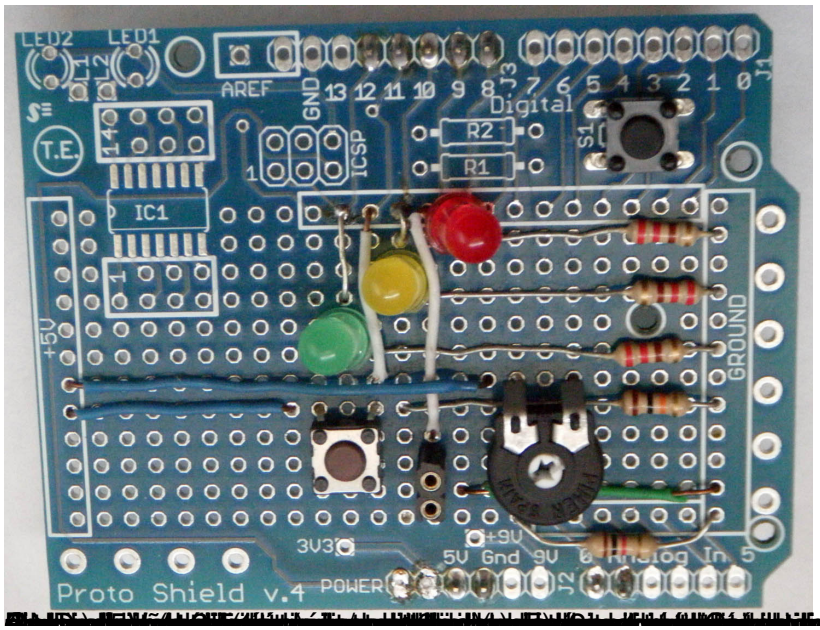


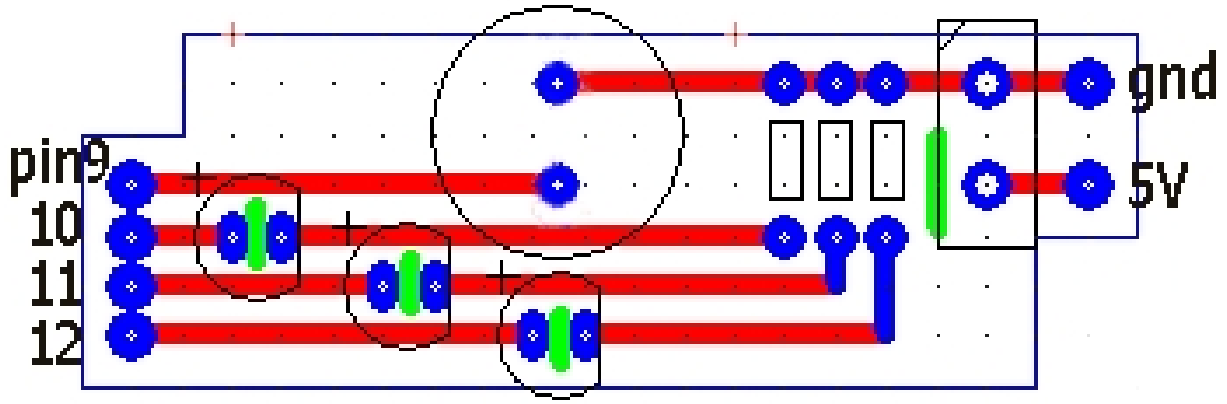

```
int val= 0;
int compare=0;
int Pinl3 = l3;
int Pin8 = 8;

 void setup(){
pinMode(Pin8, INPUT);
digitalWrite(Pin8,HIGH);
pinMode(Pinl3, OUTPUT);

}
void loop(){
digitalWrite(Pinl3, LOW);
val = digitalRead(Pin8);
 if (val == HIGH){
digitalWrite(Pinl3, HIGH);

 }
}
```

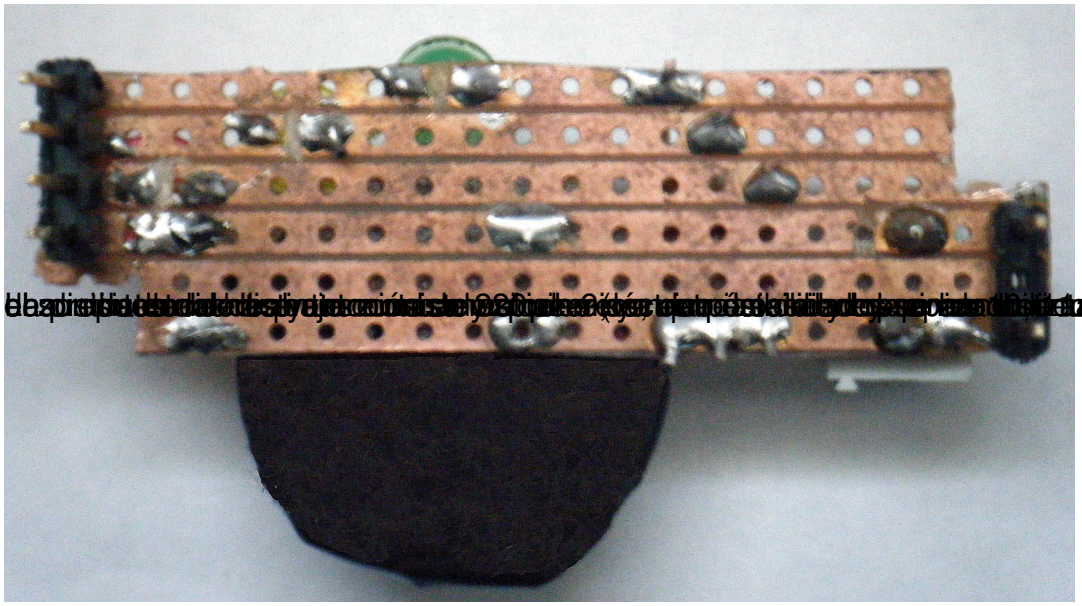
```
int val= 0;
int compare=0;
int Pin13 = 13;
int Pin8 = 8;

 void setup(){
pinMode(Pin13, OUTPUT);
pinMode(Pin8, INPUT);

}
void loop(){
digitalWrite(Pin13, LOW);
while(digitalRead(Pin8)==LOW){
delay(10);
}
digitalWrite(Pin13, HIGH);

}
```

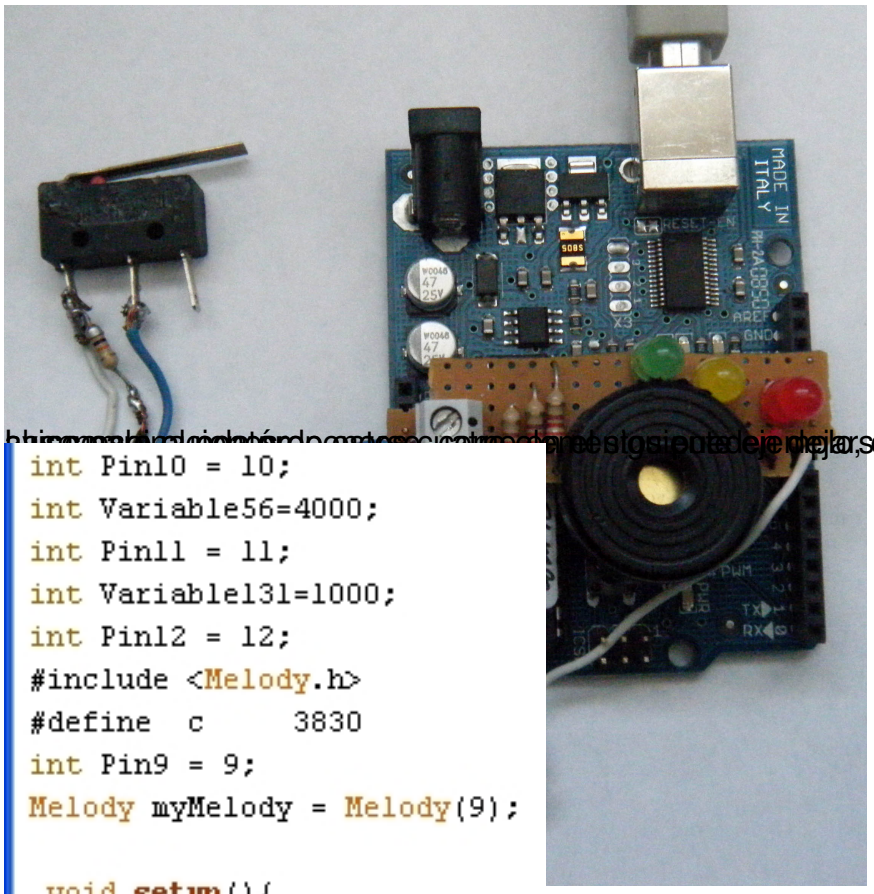
3) Combinando dispositivos de luz y sonido en una placa.

La forma más extendida de conectar con distintos dispositivos (leds, pulsadores, resistencias,...) simultáneamente a los diferentes pines de la tarjeta Arduino, es pinchando los componentes electrónicos y cables de colores en una placa de prototipos, creando los caminos eléctricos adecuados. Yo utilizo esta estrategia para hacer pruebas rápidas de programas, pero para evitar la confusión que a veces genera el exceso de cableado sobre la protoboard, con los alumnos prefiero que construyan pequeñas placas electrónicas (de gradual complejidad) soldadas, y que después las pinchen en los pines de alimentación, entradas o salidas de Arduino. El conjunto placa-tarjeta queda más sólido, se recoge mejor el material tras cada tiempo de taller (de modo que se pueden flexibilizar las sesiones de trabajo) y se combina la actividad manual con la de programación. Existen también en el mercado placas standard de prototipado para Arduino, pero su uso encarece la actividad y además su posible reciclado, en mi opinión, entretiene en exceso (la imagen siguiente es un posible ejemplo de utilización).

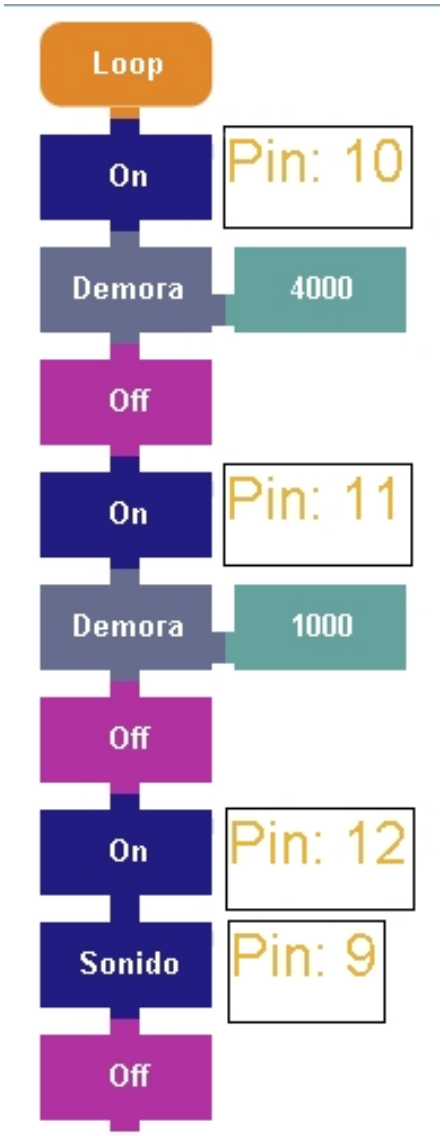

LED2
LED1
AREF
GND
Digital
ICSP
R2
R1
S1
IC1
+5V
GROUND
3V3
+9V
5V Gnd 9V
Proto Shield v.4
POWER


pin9
10
11
12
gnd
5V






```
int Pin10 = 10;
int Variable56=4000;
int Pin11 = 11;
int Variable131=1000;
int Pin12 = 12;
#include <Melody.h>
#define  c     3830
int Pin9 = 9;
Melody myMelody = Melody(9);

 void setup(){
pinMode(Pin12, OUTPUT);
pinMode(Pin11, OUTPUT);
pinMode(Pin10, OUTPUT);
myMelody.init();
}
void loop(){
analogWrite(Pin10,255);
delay(Variable56);
analogWrite(Pin10,0);
analogWrite(Pin11,255);
delay(Variable131);
analogWrite(Pin11,0);
digitalWrite(Pin12, HIGH);
myMelody.playTone(c,1000);
digitalWrite(Pin12, LOW);
}
```


Loop
On
Pin: 10
Demora
4000
Off
On
Pin: 11
Demora
1000
Off
On
Pin: 12
Sonido
Pin: 9
Off

```
int Pinl2 = l2;
int Variable56=l000;
int Pinll = ll;
int Pinl0 = l0;
int Pin8 = 8;

 void setup(){
pinMode(Pinl2, OUTPUT);
pinMode(Pinll, OUTPUT);
pinMode(Pinl0, OUTPUT);
pinMode(Pin8, INPUT);
}
void loop(){
digitalWrite(Pinl2, HIGH);
delay(Variable56);
analogWrite(Pinll,255);
delay(Variable56);
analogWrite(Pinl0,255);
delay(Variable56);
while(digitalRead(Pin8)==LOW){
delay(l0);
}
analogWrite(Pinl0,0);
analogWrite(Pinll,0);
digitalWrite(Pinl2, LOW);
delay(Variable56);
}
```

4) Conclusión

Las ventajas de la programación con bloques gráficos en Amici evita las dificultades de la escritura en lenguaje C (o mejor dicho, en Processing) para usuarios inexpertos, pero sin renunciar a dar el paso de analizar como queda compilada la solución en la consola de Arduino, lo que facilita futuras experiencias más complejas que las mostradas en este monográfico, las cuales por otro lado, están disponibles en número casi ilimitado en la Red.

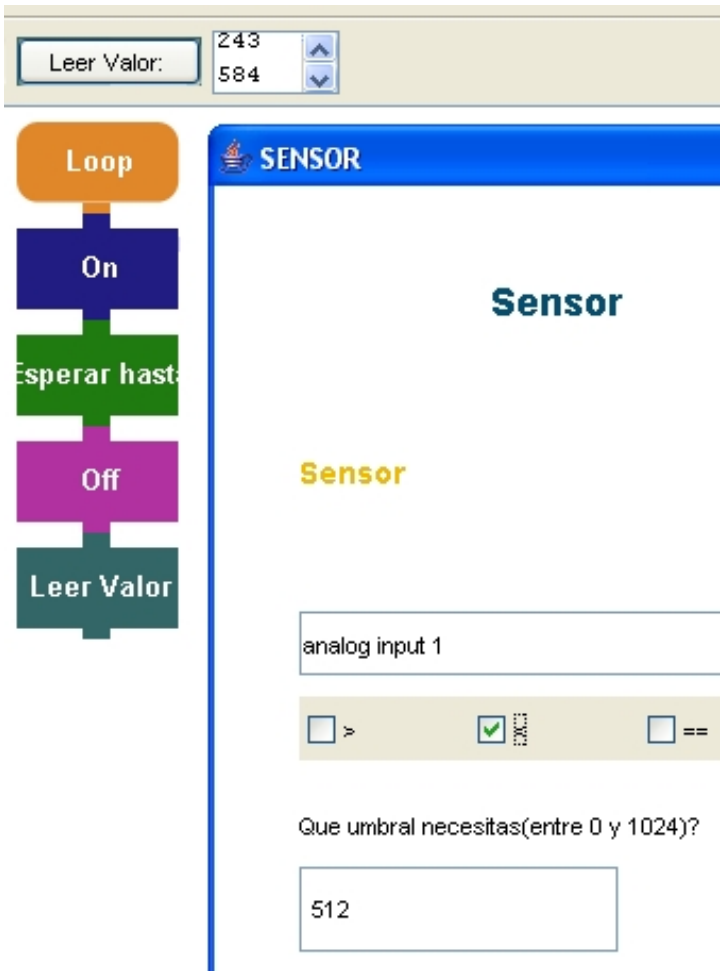
Capítulo 3
1) Introducción

Superada la experimentación con entradas digitales para crear programas en nuestra placa, ahora conviene conocer las posibilidades de la tarjeta Arduino para interpretar señales analógicas de entrada, utilizando los sensores electrónicos más utilizados con nuestros alumnos.

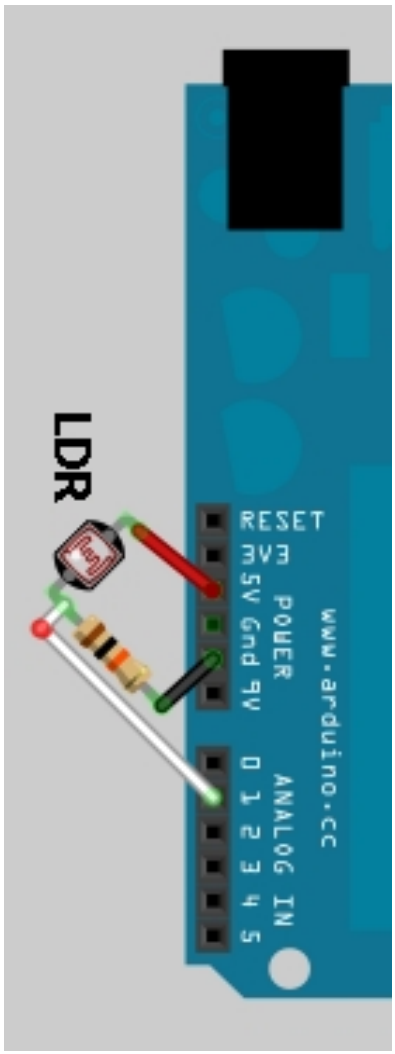
2) Conexionado de entradas analógicas.

La ventaja que nos proporcionan los sensores analógicos frente a los digitales estriba en la posibilidad de regular su holgura de detección, evitando el "todo o nada" (0V ó 5V de tensión en el pin de entrada) característico de los sensores (y de las salidas) digitales. Con un sensor de luz resistivo, por ejemplo, podremos decidir a nuestro gusto cuál es el umbral de iluminación con el que actuarán los elementos conectados a las salidas de la tarjeta Arduino. Dicho de otro modo, un sensor analógico estará constantemente cambiando el valor de tensión entre sus terminales con el tiempo, por lo que necesitará un tratamiento diferente, en comparación con el sensor digital, por parte del microprocesador.

Arduino incorpora 6 patillas o pines denominados *Analog In.* Mediante la configuración "pull-down" del sensor (ver capítulo anterior), las pequeñas variaciones de voltaje que se produzcan en el pin de entrada correspondiente de la tarjeta, serán traducidas por el software en un número comprendido entre 0 (= 0 voltios) y 1023 (= 5 voltios), es decir, el umbral de tensión que queramos que perciba la tarjeta como límite, se fijará escribiendo un número menor que 1024, junto con la función anal ogRead()
. También es importante saber que la pantalla de Amici dispone de una pequeña ventana en su parte superior donde, tras pulsar el botón
"Leer Valor",
podremos monitorizar la lectura del sensor analógico.


Leer Valor:
243
584
Loop
On
Esperar hast
Off
Leer Valor
SENSOR
Sensor
Sensor
analog input 1
>
==
Que umbral necesitas(entre 0 y 1024)?
512

```
int Pinl3 = 13;
int Pinl = 1;

 void setup(){
pinMode(Pinl3, OUTPUT);
pinMode(Pinl, INPUT);
//Serialmonitor
Serial.begin(9600);

}
void loop(){
digitalWrite(Pinl3, HIGH);
while(analogRead(Pinl)>=512){
delay(10);
}
digitalWrite(Pinl3, LOW);
int analogValue;
analogValue = analogRead(1);
Serial.println(analogValue);
delay(10);

}
```

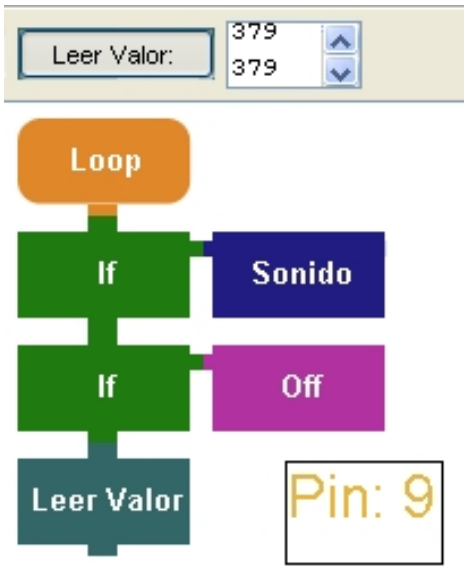

LDR
RESET
3V3
5V
Gnd
Vin
POWER
ANALOG IN
0 1 2 3 4 5
www.arduino.cc

3. Experimentando con sensores analógicos

La resistencia variable con la luz es posiblemente el sensor analógico más conocido y utilizado por nuestros alumnos en el taller de Tecnología, pero existen otros que también pueden estar a nuestro alcance para crear programas con Amici, como por ejemplo, una resistencia variable con la temperatura (NTC), un potenciómetro, un sensor de infrarrojos de corto alcance (CNY70) o de medio alcance (Sharp GP2D12).

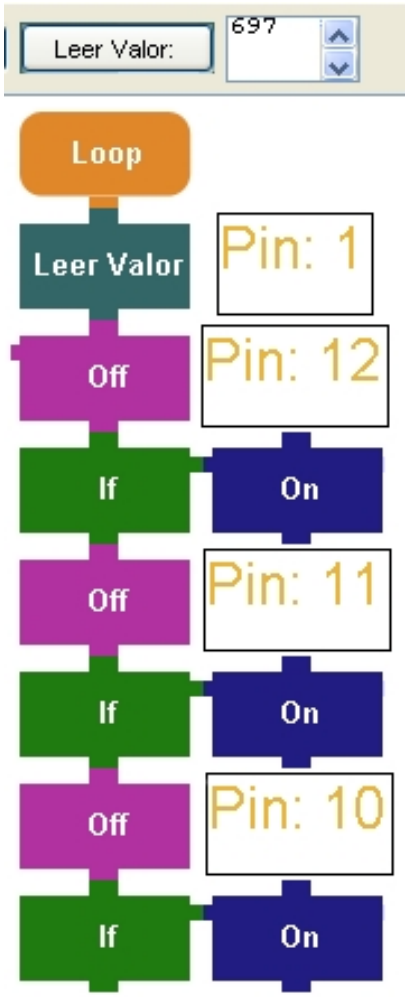
Para realizar el siguiente ejemplo, he pinchado una resistencia variable con la temperatura de coeficiente negativo, es decir, que disminuya el valor de su resistencia conforme aumente la cantidad de calor que almacene. El inconveniente de estas resistencias es que, al contrario que el resto de los sensores analógicos que utilizaré, tienen una respuesta lenta, lo que debe aprovecharse con algunos alumnos para que tengan tiempo de analizar su funcionamiento, a través de una programación elemental. El esquema de cableado para conectarlo a la tarjeta Arduino es similar al caso de la LDR, aunque en este caso, voy a utilizar una resistencia de 4,7 Kohmios entre el terminal del sensor y masa (Gnd).

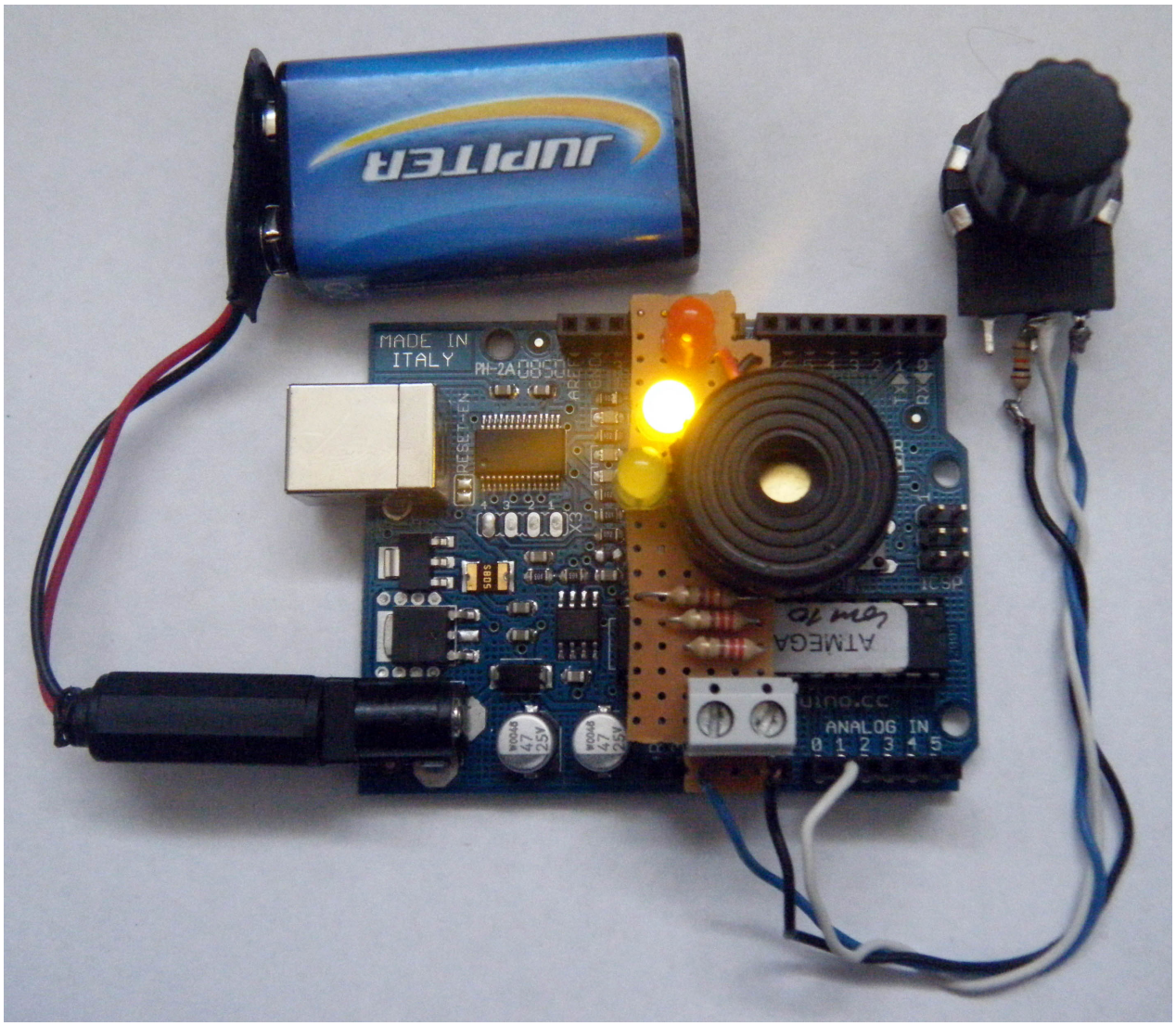
El programa que desarrollo a continuación es muy sencillo e incorpora un ejemplo de cómo utilizar dos bloques IF para crear una doble condición, utilizando la entrada analógica: si la señal detectada en el pin es superior a un valor fijado en el programa, sucederá una acción, y si es inferior, sucederá la contraria. En mi caso, aprovecho la presencia del zumbador piezoeléctrico de mi tarjeta y hago que suene (con una breve intermitencia) tras calentar la resistencia variable poniéndola en contacto, durante algunos segundos, con la punta de mi soldador, previamente calentado. El sonido continuará durante el tiempo que, una vez retirado el calor, la resistencia se enfríe lo suficiente para que la señal en la entrada analógica (pin 1), disminuya en valor numérico por debajo del umbral indicado (en mi ejemplo vale 500), hecho que observaremos atentamente en la ventana de LEER VALOR.



```
void loop(){
val = analogRead(Pin1);
compare=500 ;
if (val > compare){
myMelody.playTone(c,50);
delay(100);
} else { }

val = analogRead(Pin1);
compare=500 ;
if (val < compare){
analogWrite(Pin9,0);
} else { }

int analogValue;
analogValue = analogRead(1);
Serial.println(analogValue);
delay(10);
}
```

```
void loop(){
analogWrite(Pin10,0);
val = analogRead(Pin1);
compare=600;
if (val == compare){
analogWrite(Pin10,255);
} else {
 }
analogWrite(Pin11,0);
val = analogRead(Pin1);
compare=800;
if (val == compare){
analogWrite(Pin11,255);
} else {
 }
digitalWrite(Pin12, LOW);
val = analogRead(Pin1);
compare=1000;
if (val == compare){
digitalWrite(Pin12, HIGH);
} else {
 }
int analogValue;
analogValue = analogRead(1);
Serial.println(analogValue);
delay(10);
}
```


Leer Valor:
697
Loop
Leer Valor
Pin: 1
Off
Pin: 12
If
On
Off
Pin: 11
If
On
Off
Pin: 10
If
On

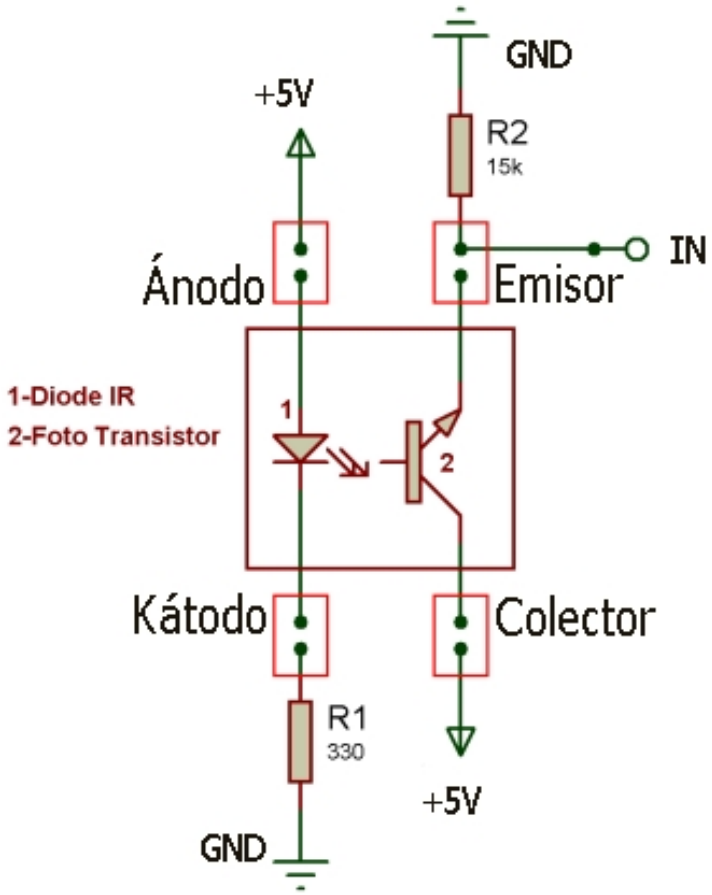


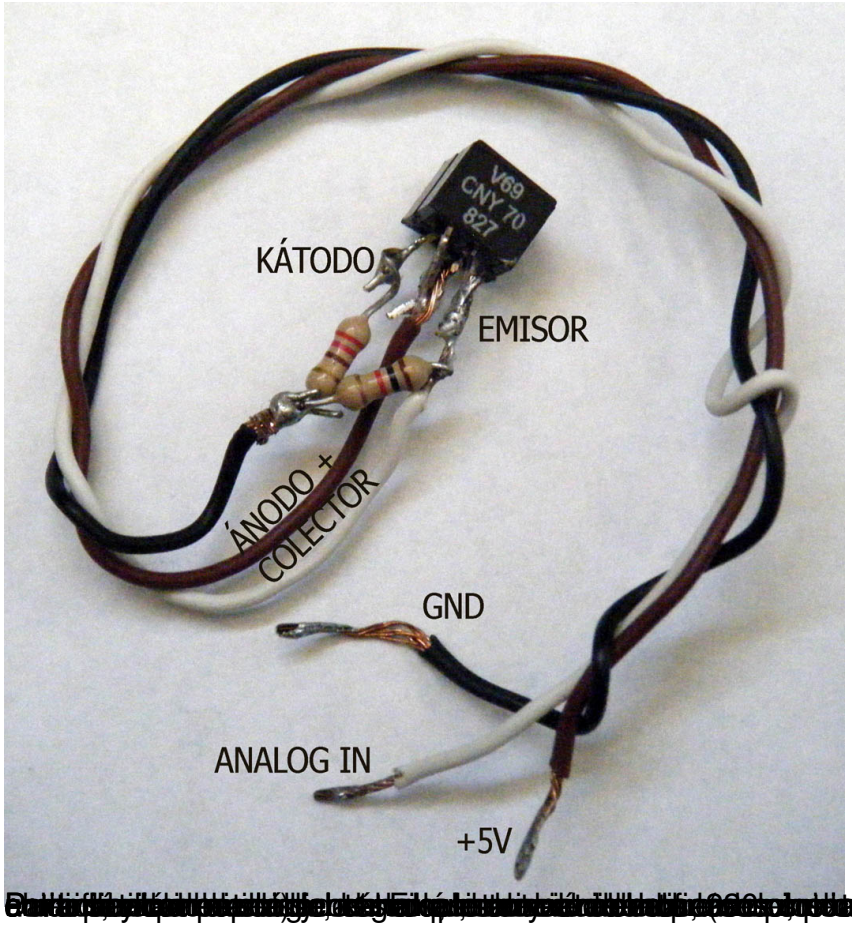
4) Conexionado de sensores de infrarrojos

La utilización de este tipo de sensores analógicos supone un pequeño salto cualitativo en el taller de Tecnología, respecto al tipo de dispositivos que allí normalmente se emplean y que hemos descrito hasta ahora. Sin embargo el detector denominado CNY70 ya aparece en circuitos ejemplificados en distintos libros de texto de 4º ESO, no resulta caro (algo menos de un euro) y es de gran fiabilidad; quizás su mayor inconveniente resida en la conexión correcta de sus cuatro patillas (dos para el diodo emisor de infrarrojos y dos para el fototransistor que actúa como receptor) a los tres cables que deben pincharse en la tarjeta.

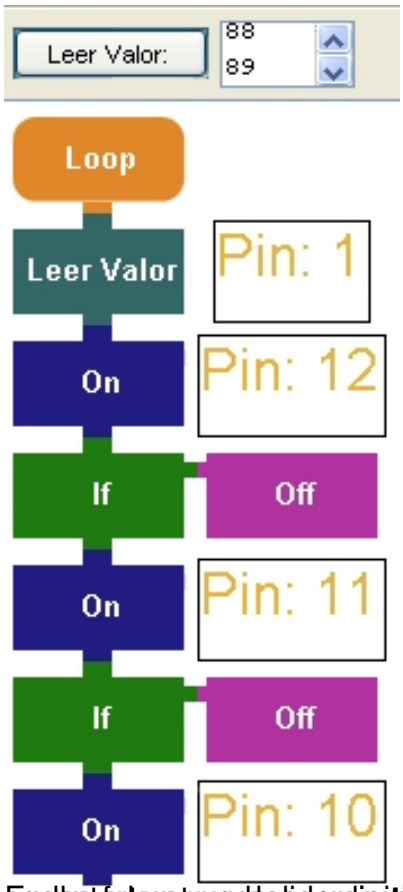
Normalmente se utilizan pareados en pequeños robots rastreadores que distinguen entre una línea negra y un fondo blanco para variar su trayectoria, pero yo en mi ejemplo lo voy a utilizar además para detectar el color, entendido éste como una tonalidad intermedia entre negro y blanco; el encendido simultáneo de uno, dos o tres diodos leds en mi tarjeta, determinará en este mismo orden, cuál es el grado de reflexión de la luz (casi nula para el negro, media para el

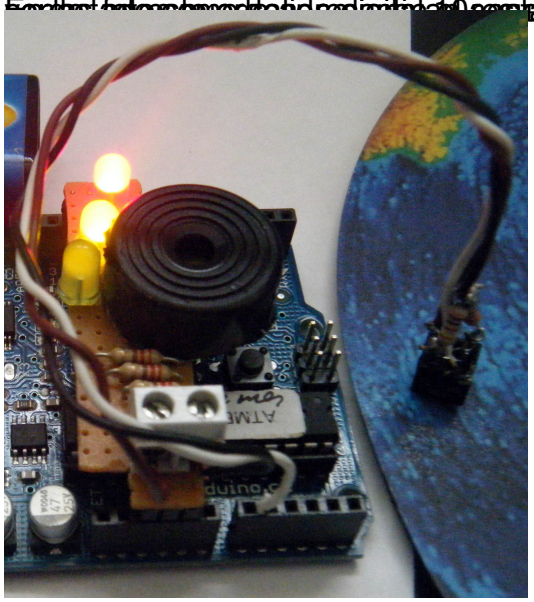
color y máxima para el blanco) en la superficie sobre la que aproximemos el sensor de infrarrojos.

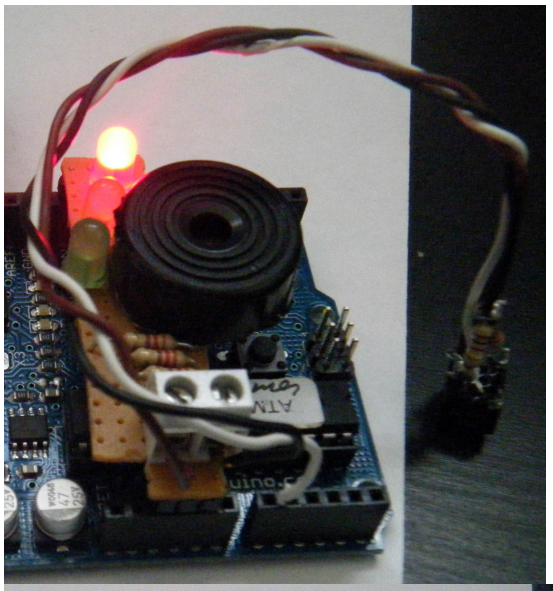


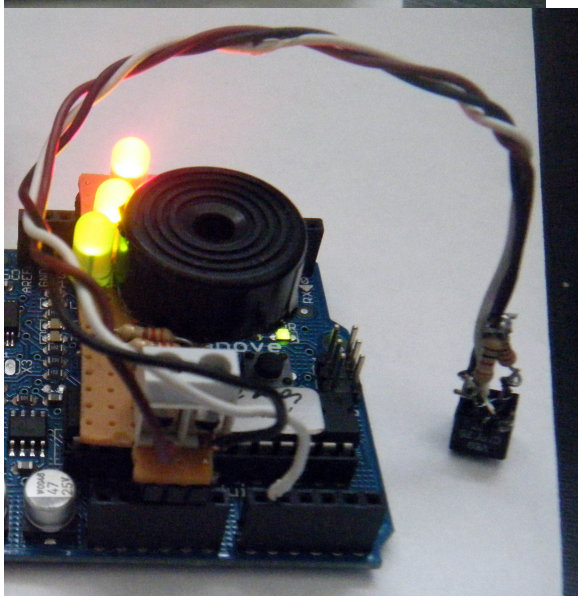

V69
CNY 70
827
KÁTODO
EMISOR
ÁNODO +
COLECTOR
GND
ANALOG IN
+5V

```
void loop(){
int analogValue;
analogValue = analogRead(1)
Serial.println(analogValue)
delay(10);

digitalWrite(Pin12, HIGH);
val = analogRead(Pin1);
compare=100;
if (val < compare){
digitalWrite(Pin12, LOW);
} else {
 }

analogWrite(Pin11,255);
val = analogRead(Pin1);
compare=20;
if (val < compare){
analogWrite(Pin11,0);
} else {
 }

analogWrite(Pin10,255);
}
```


Leer Valor:
88
89
Loop
Leer Valor
Pin: 1
On
Pin: 12
If
Off
On
Pin: 11
If
Off
On
Pin: 10







5) Conclusión

En este capítulo se han ampliado las posibilidades de interacción de nuestra placa conectada en la tarjeta Arduino con el mundo exterior y sus variables físicas, si bien empieza a echarse en falta la posibilidad de conectar, además de una variedad de dispositivos de entrada, algún nuevo actuador en una salida, como puede ser un pequeño motor de los que se disponen habitualmente en el taller de Tecnología, experimentación que abordaré en los próximos capítulos.

Capítulo 4

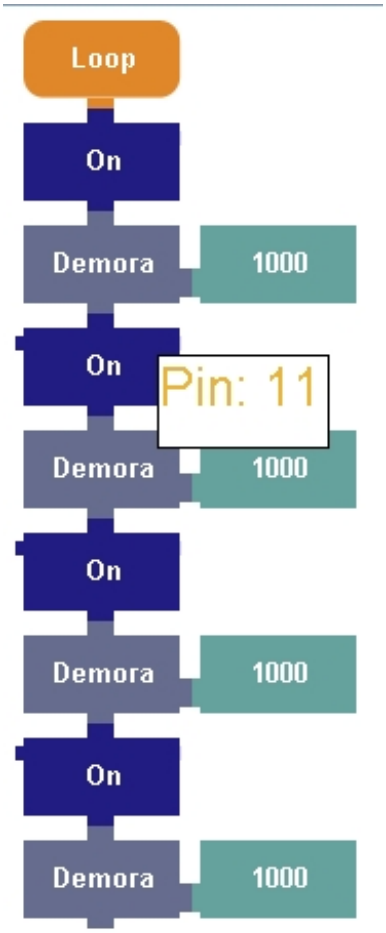
1) Introducción

Una posibilidad que proporciona la tarjeta Arduino, como ya comenté en el primervcapítulo, es la de crear señales de salida analógicas, es decir, variables en el tiempo y en su valor de tensión, desde 0 a 5V. El programa Amici permite programar estas señales de tipo PWM (*pulse width modulation*
) en los pines 9, 10 y 11, asignando su tensión de salida gradual con números del 0 al 255, lo que probaré a continuación.

2) Iluminación variable de un diodo led

La utilización del bloque SONIDO para que suenen distintas notas musicales a través del piezoeléctrico conectado al pin digital 9 de nuestra placa, es una forma sofisticada de aprovechar una salida analógica de la tarjeta Arduino, en este caso, con una estrategia difícil de explicar a nuestros alumnos, ya que los tonos se calculan con el número inverso a la frecuencia de cada nota musical, y así aparece explicitado en el editor de texto de Arduino, a través de la librería denominada por el programa <Melody.h >.

```
int val= 0;
int compare=0;
int Pinll = ll;
int Variable56=1000;

 void setup(){
pinMode(Pinll, OUTPUT);

}
void loop(){
analogWrite(Pinll,5);
delay(Variable56);
analogWrite(Pinll,50);
delay(Variable56);
analogWrite(Pinll,100);
delay(Variable56);
analogWrite(Pinll,200);
delay(Variable56);

}
```




```
#include <Melody.h>
#define  c     3830
#define  d     3400
#define  e     3038
#define  f     2864
#define  g     2550
#define  a     2272
#define  b     2028
#define  C     1912
int Pin9 = 9;
Melody myMelody = Melody(9);
```

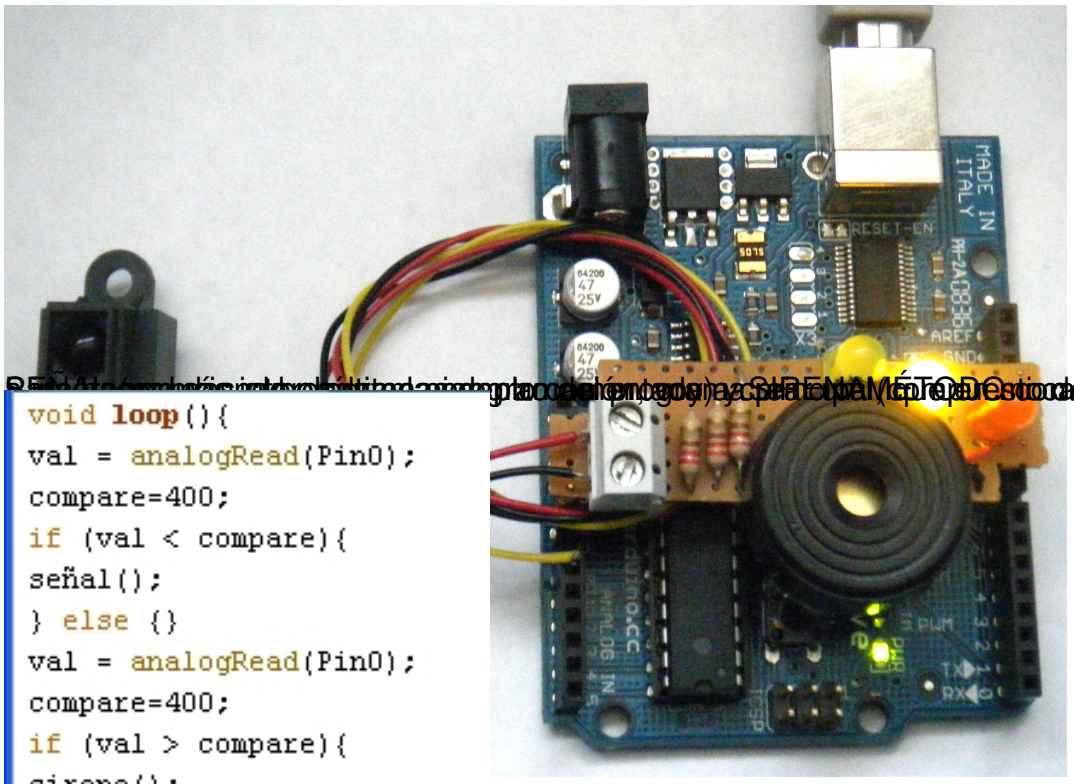
3) Creación de un detector de presencia utilizando el bloque METODO

Las alarmas y detectores de presencia son dispositivos de control muy utilizados en nuestra vida cotidiana para crear sistemas antirrobo, la apertura automática de puertas de garaje o de comercios, activar escaleras mecánicas, barreras de paso de vehículos o la cinta corredera en la caja de un supermercado. Los sensores más utilizados en estos ejemplos son los detectores

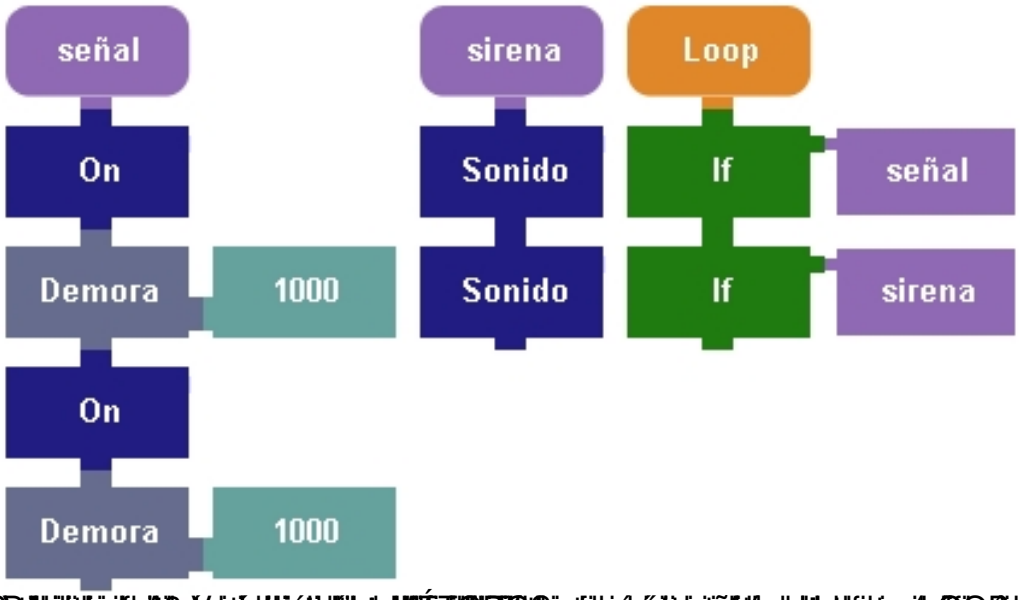
de presión (que actúan como un pulsador) y los conjuntos emisor-receptor de infrarrojos.

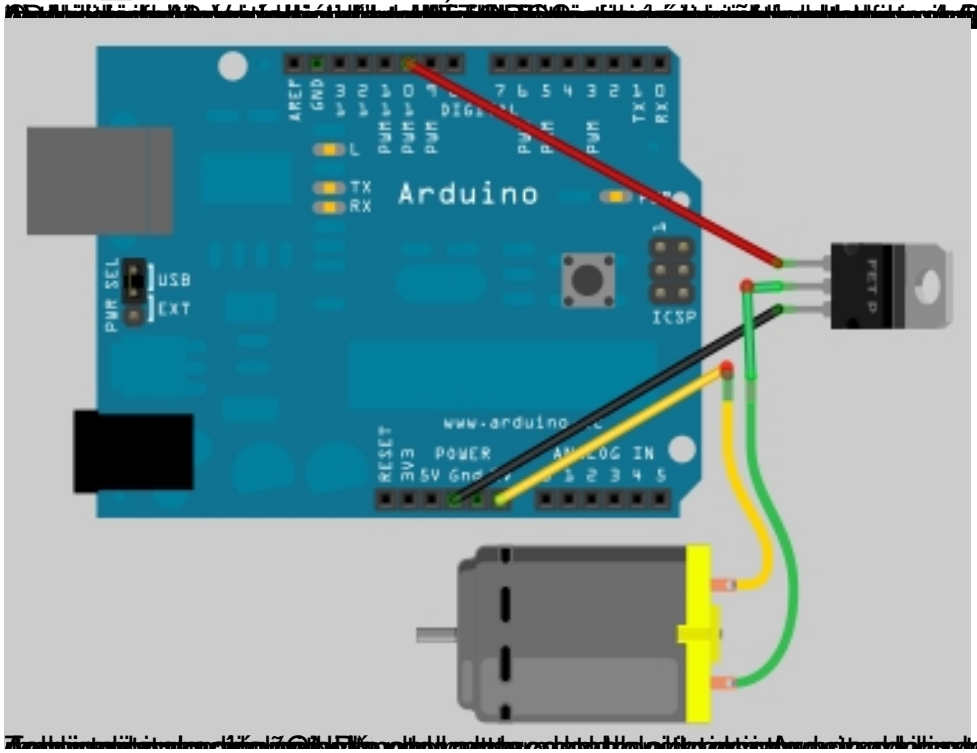
Voy a programar una alarma acústica-luminosa utilizando mi placa, para generar simultáneamente, una melodía y la variación de luz de un diodo led, cuando aproxime mi mano a un sensor de infrarrojos.

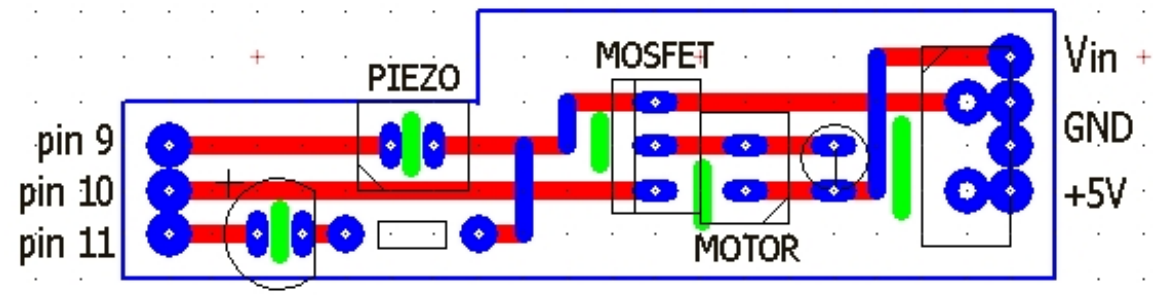
Si conecto un detector tipo Sharp GP2D12 en la entrada analógica 0, puedo asignar (con la pestaña “LEER VALOR” de Amici) un valor umbral de distancia que superado, produzca el sonido de alarma o, por contra, la variación de brillo en el led como señalización del sistema. Este sensor es más sofisticado y caro (unos 15 euros) que los que he utilizado hasta ahora, pero me parecía interesante mostrarlo en un ejemplo, ya que su conexionado es muy sencillo y funciona como un potenciómetro que varía su valor resistivo al situarse un elemento frente a él, a mayor o menor distancia (desde 10 a 80 cm); la luz infrarroja de medio alcance procedente del emisor rebotará en el objeto detectado y llegará una señal de luz al receptor (ambos formando una única pieza). Simplemente dispone de tres cables que conectar: negro en GND, rojo en + 5V y amarillo en al pin de entrada analógica elegido.

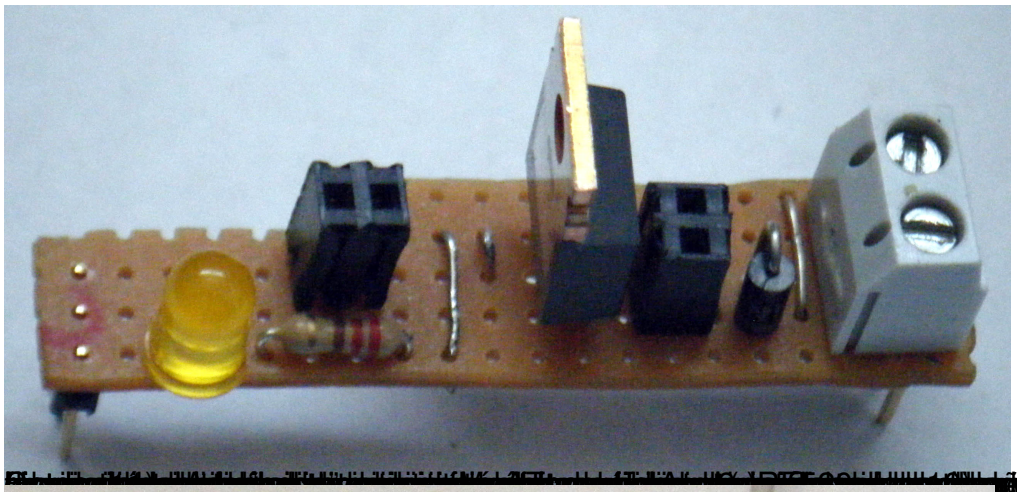


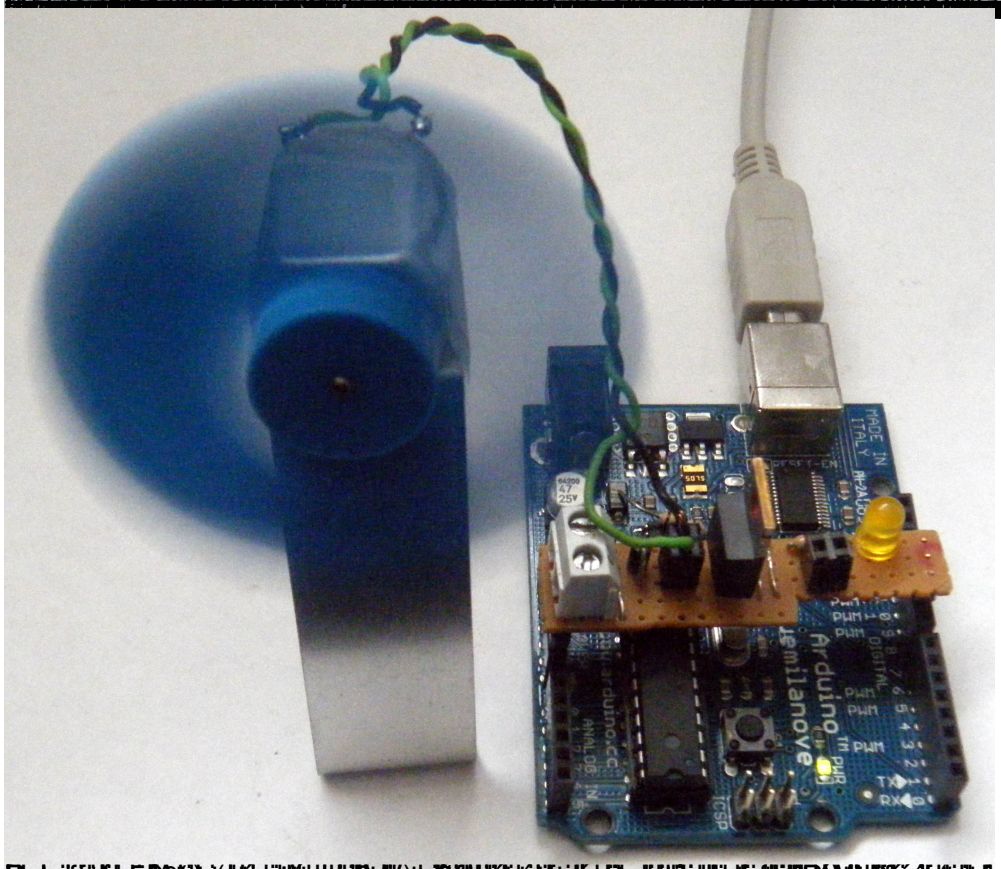

```
void loop(){
val = analogRead(Pin0);
compare=400;
if (val < compare){
señal();
} else {}
val = analogRead(Pin0);
compare=400;
if (val > compare){
sirena();
} else {}
}

void señal(){
analogWrite(Pin11,64);
delay(Variable53);
analogWrite(Pin11,192);
delay(Variable103);
}

void sirena(){
myMelody.playTone(c,2000);
delay(100);
myMelody.playTone(g,2000);
delay(100);
}
```


señal
On
Demora
1000
On
Demora
1000
sirena
Sonido
Sonido
Loop
If
señal
If
sirena


Arduino
USB
EXT
PWR SEL
ICSP
POWER
ANALOG IN


PIEZO
MOSFET
MOTOR
pin 9
pin 10
pin 11
Vin
GND
+5V

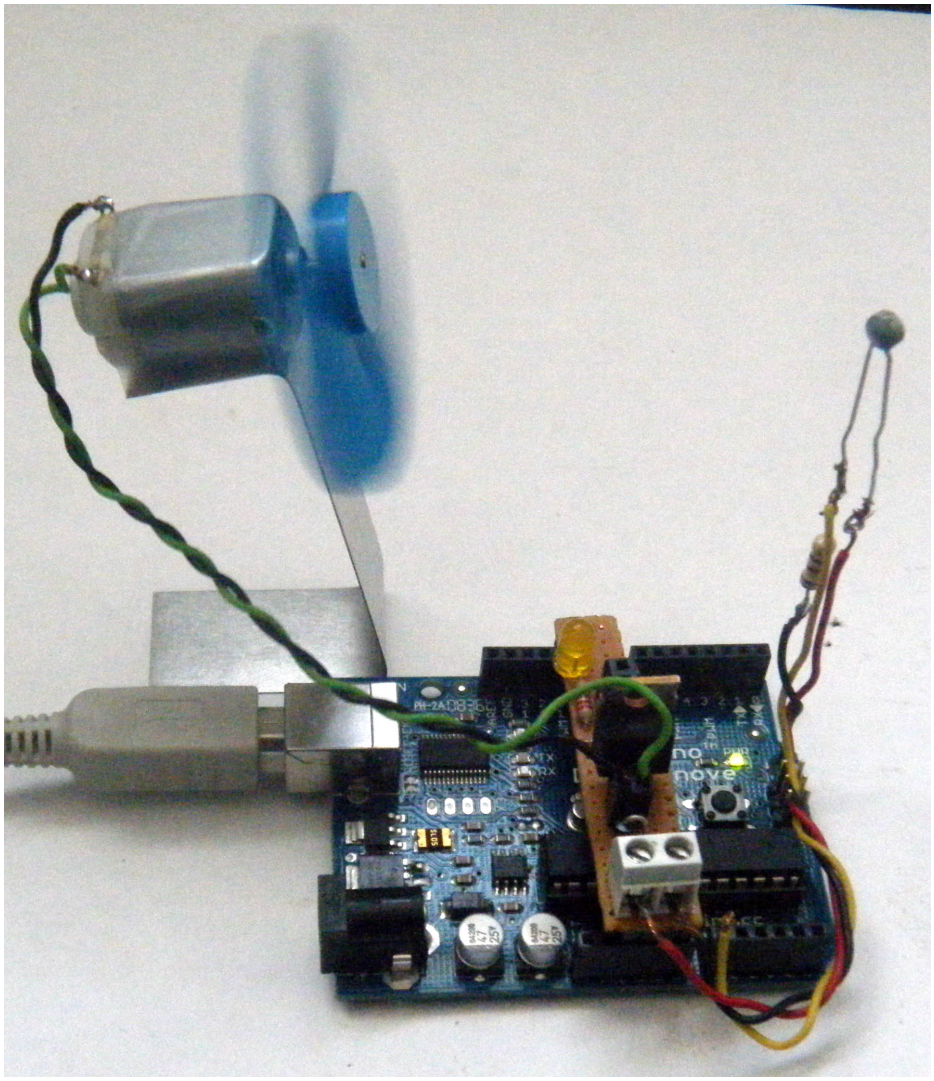




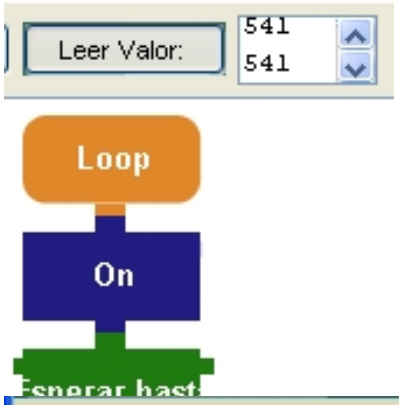
5) Programación de un ventilador con termostato

El dispositivo de control de mayor presencia actualmente en nuestras viviendas, tanto incorporado en distintos electrodomésticos (horno, frigorífico, radiadores, etc.) como presente en las modernas instalaciones de calefacción, es el termostato; de hecho, es el ejemplo que utiliza la mayoría de libros de texto de Secundaria para explicar el funcionamiento de un bucle de control en lazo cerrado o con realimentación.

Podemos nosotros construirnos un sensor de temperatura (ver capítulo 3) y conectarlo a nuestra placa para motor, ya que ésta incorpora una clema, aprovechando los pines de 5 V y GND de la tarjeta Arduino, que permite establecer una corriente a través de la resistencia variable NTC (o de cualquier otro sensor). Ahora la idea es enfrentar el pequeño motor con aspas al sensor, para provocar con su calentamiento (acercando el soldador o presionando con nuestros dedos la NTC) y refrescamiento (con el aire generado por las aspas), un encendido y apagado contínuo del motor, en un bucle sin fin.



El reto en la programación será, tras conectar el cable de datos amarillo en el pin analógico 0, determinar el valor umbral que, monitorizado a través del cable USB y mediante la pestaña “LEER VALOR” de la consola de Amici, provoque las transiciones de giro o parada en el motor, espaciando dichos estados en el tiempo según calentemos más o menos la resistencia.




```
int Pin10 = 10;
int Pin0 = 0;

 void setup(){
pinMode(Pin10, OUTPUT);
pinMode(Pin0, INPUT);
Serial.begin(9600);

}
void loop(){
analogWrite(Pin10,255);
while(analogRead(Pin0)>=550){
delay(10);
}
analogWrite(Pin10,0);
int analogValue;
analogValue = analogRead(0);
Serial.println(analogValue);
delay(10);

}
```

6) Conclusión

La placa que he utilizado para activar un motor con la tarjeta Arduino abre la posibilidad el control de los pequeños proyectos que construimos con nuestros alumnos en el taller y que programamos con tarjetas comerciales tipo Enconor. También podríamos modificar el funcionamiento de pequeños juguetes con motor,

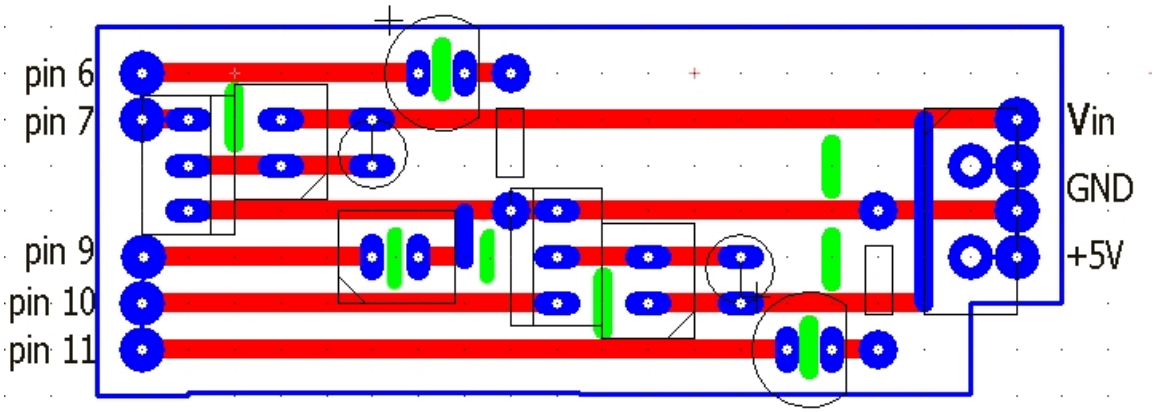
luz o sonido; por último, duplicando la presencia del transistor en una placa, tendremos la oportunidad programar un dispositivo móvil sencillo.

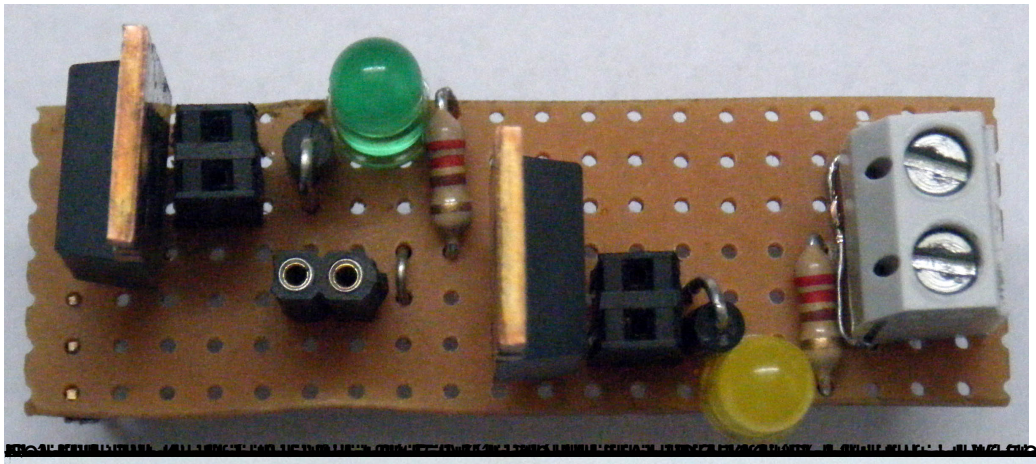
Capítulo 5
1) Introducción

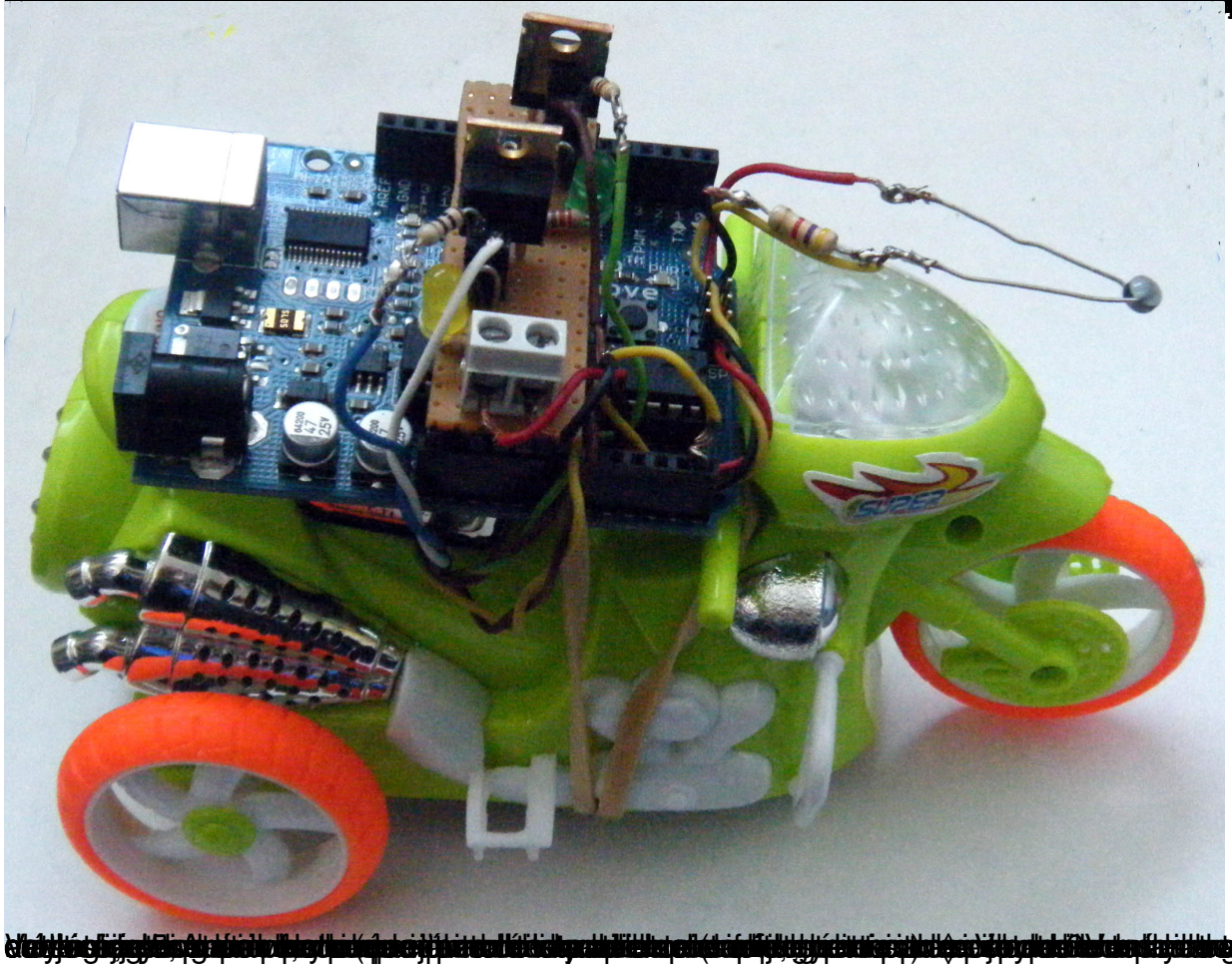
En este capítulo, la propuesta final consistirá en crear un dispositivo móvil sencillo dotado con dos pequeños motores, aunque sólo giren en un sentido; pero antes experimentaré a insertar Arduino en algún juguete baratos, con luz y movimiento, modificando su funcionamiento.

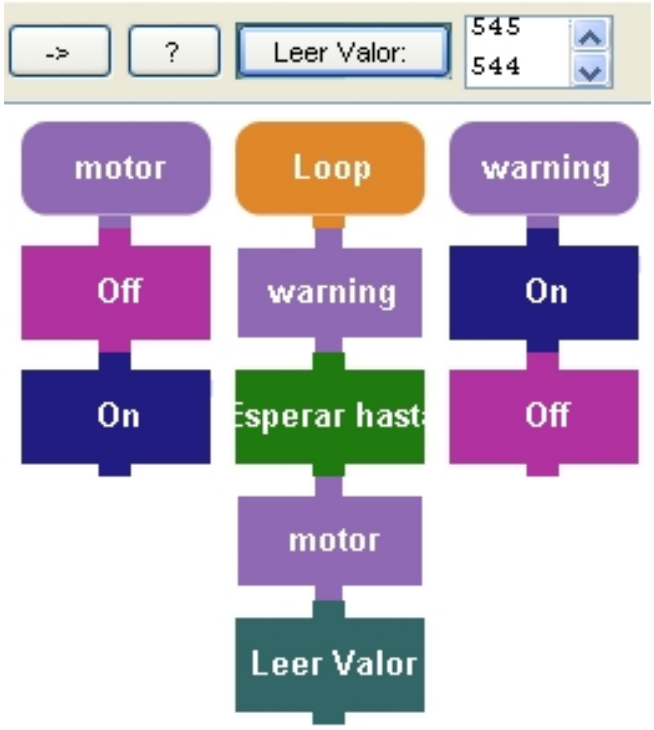
2) Diseño de una placa con dos transistores y programación de un juguete

Voy a comenzar diseñando una placa con dos transistores, basada en la ampliación de la que utilizamos en el capítulo anterior y con sus características: presencia del pin 9 para crear sonidos con Amici, posibilidad de activar dos diodos led (que se encenderán con los pines de salida 6 y 11) y presencia de una clema (pinchada sobre los pines Gnd y 5 V) para la alimentación de los sensores.

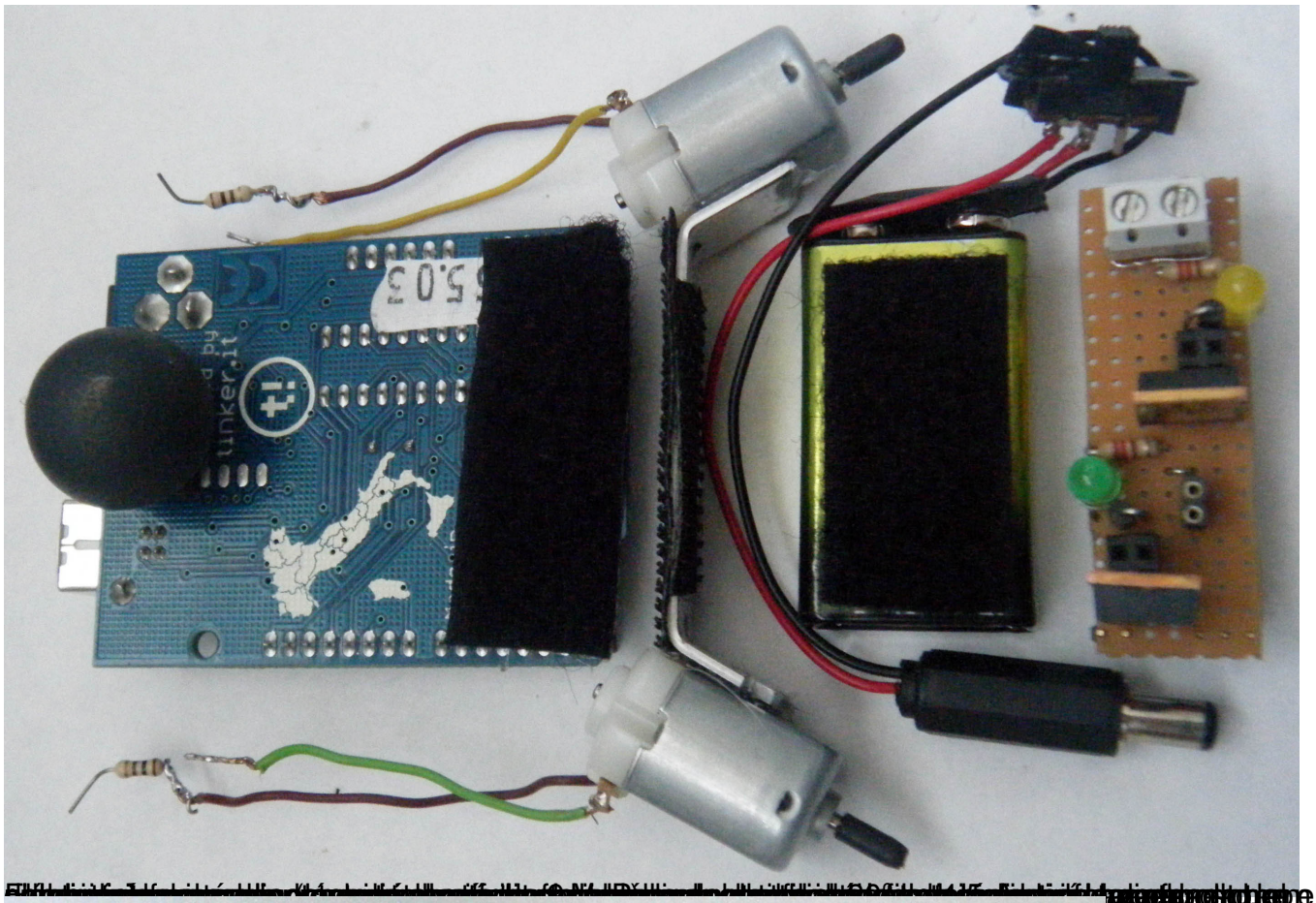

pin 6
pin 7
pin 9
pin 10
pin 11
Vin
GND
+5V

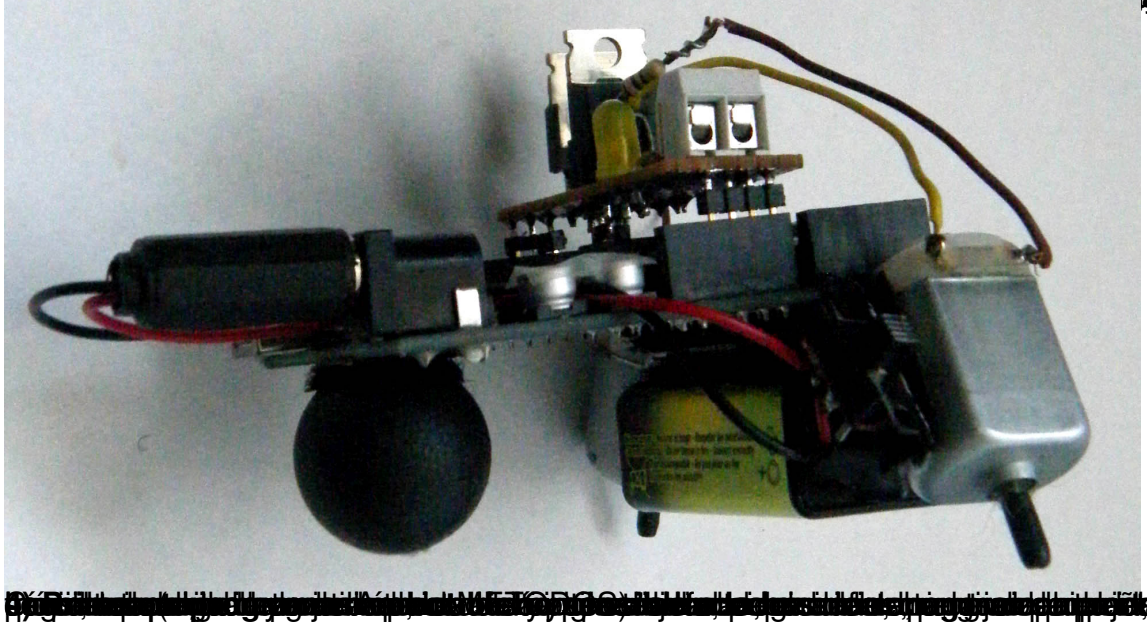


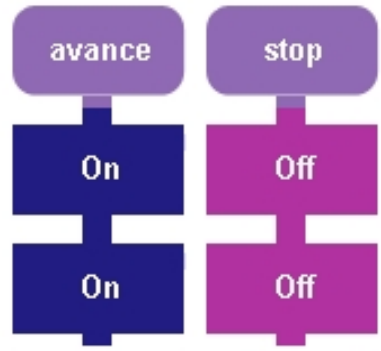


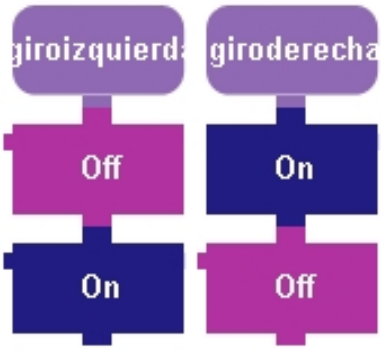


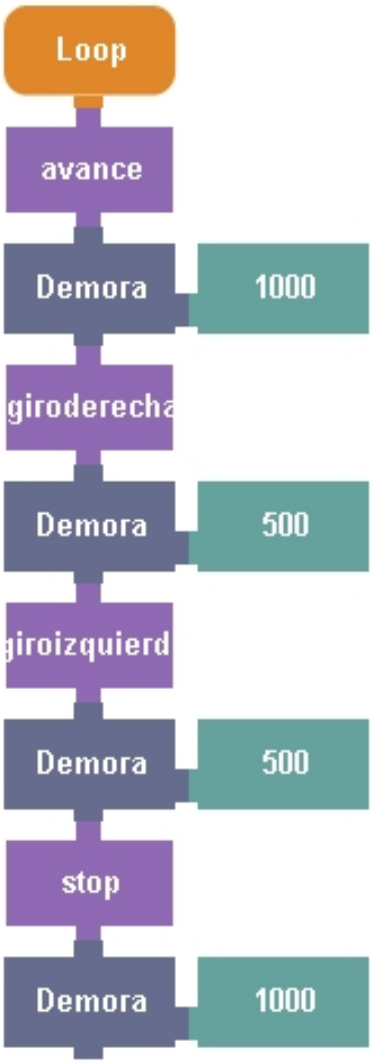

```
void loop(){
warning();while(analogRead(Pin5)<=550){
delay(10);
}
motor();int analogValue;
analogValue = analogRead(Pin5);
Serial.println(analogValue);
delay(10);
}

void motor(){
digitalWrite(Pin7, LOW);
analogWrite(Pin10,255);
}

void warning(){
digitalWrite(Pin7, HIGH);
analogWrite(Pin10,0);
}
```


tinker.it

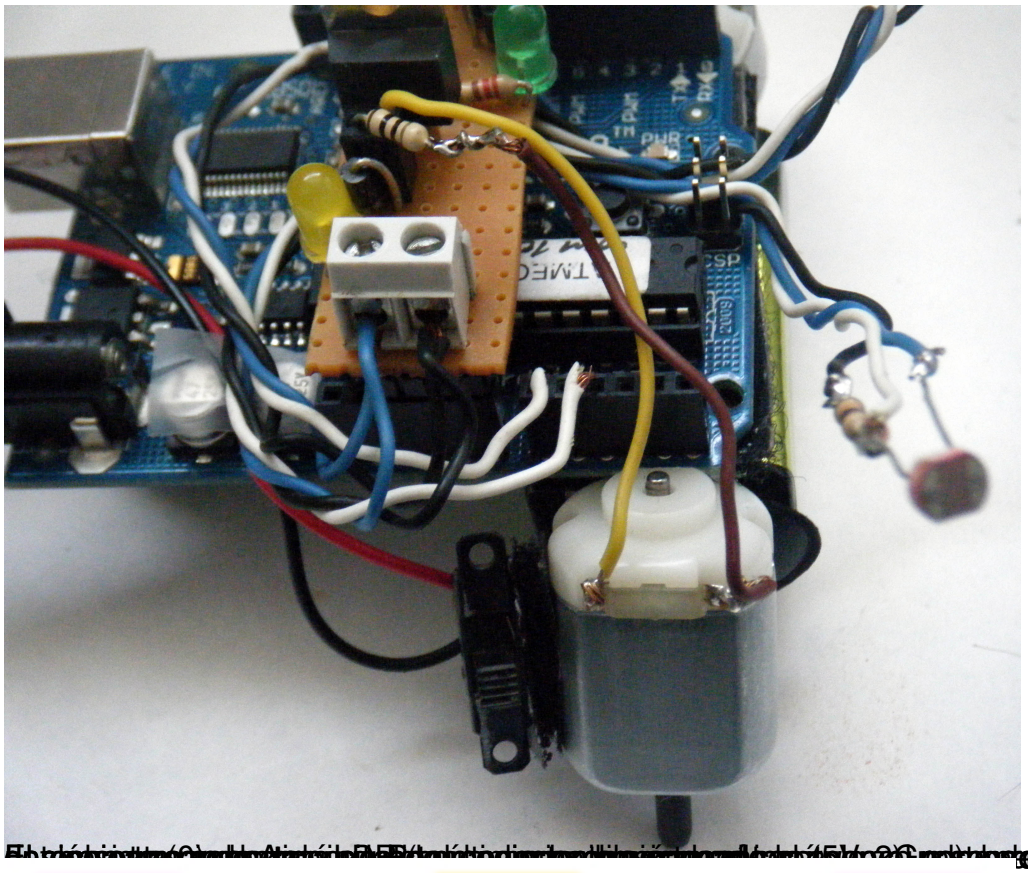


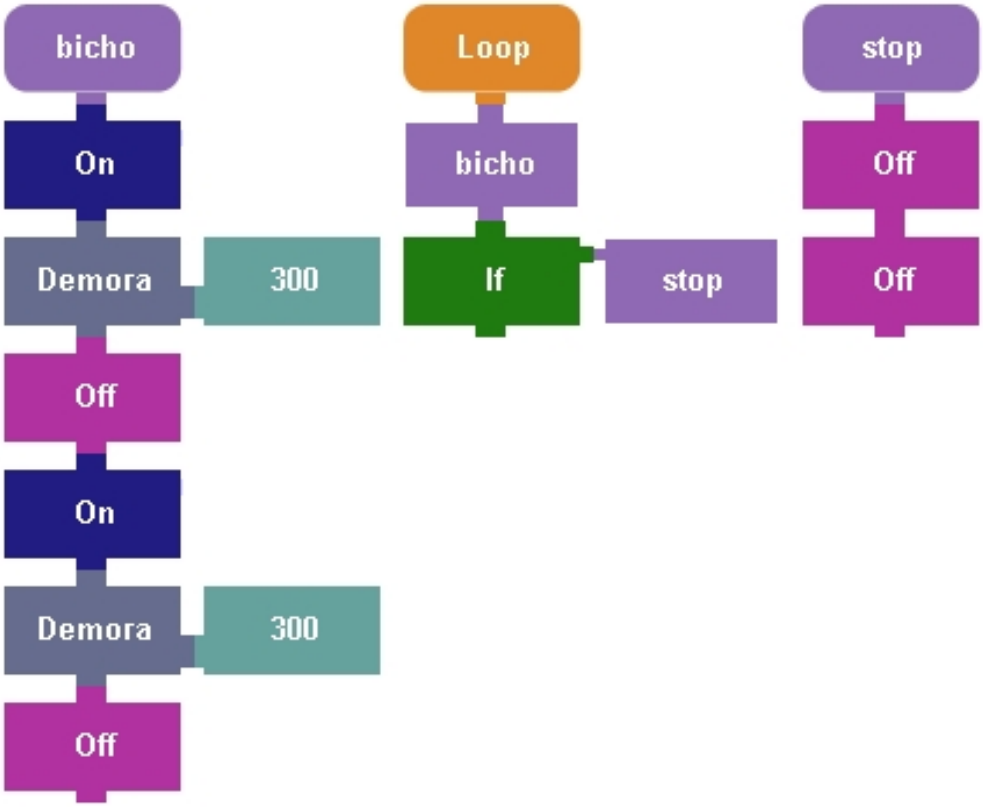

avance
stop
On
Off
On
Off


giroizquierd
giroderecha
Off
On
On
Off


Loop
avance
Demora
1000
giroderecha
Demora
500
giroizquierd
Demora
500
stop
Demora
1000

```
void loop(){
avance();delay(Variable56);
giroderecha();delay(Variable106);
giroizquierda();delay(Variable106);
stop();delay(Variable56);
}

void avance(){
digitalWrite(Pin7, HIGH);
analogWrite(Pin10,255);
}

void stop(){
digitalWrite(Pin7, LOW);
analogWrite(Pin10,0);
}

void giroizquierda(){
digitalWrite(Pin7, LOW);
analogWrite(Pin11,255);
}

void giroderecha(){
digitalWrite(Pin7, HIGH);
analogWrite(Pin10,0);
}
```

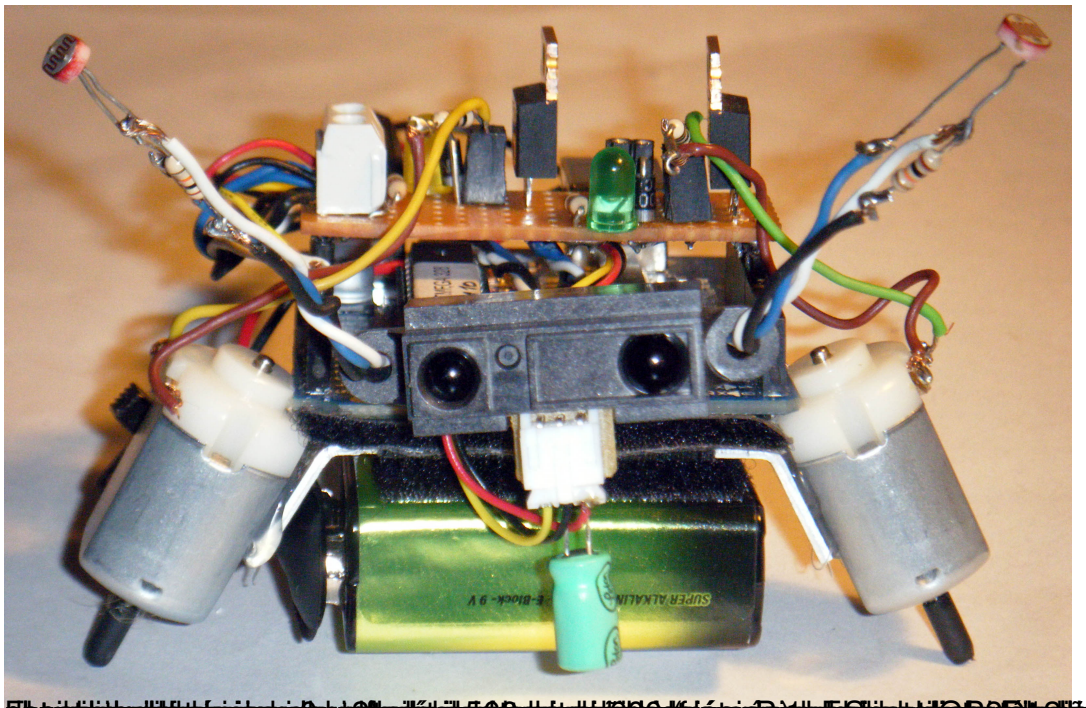


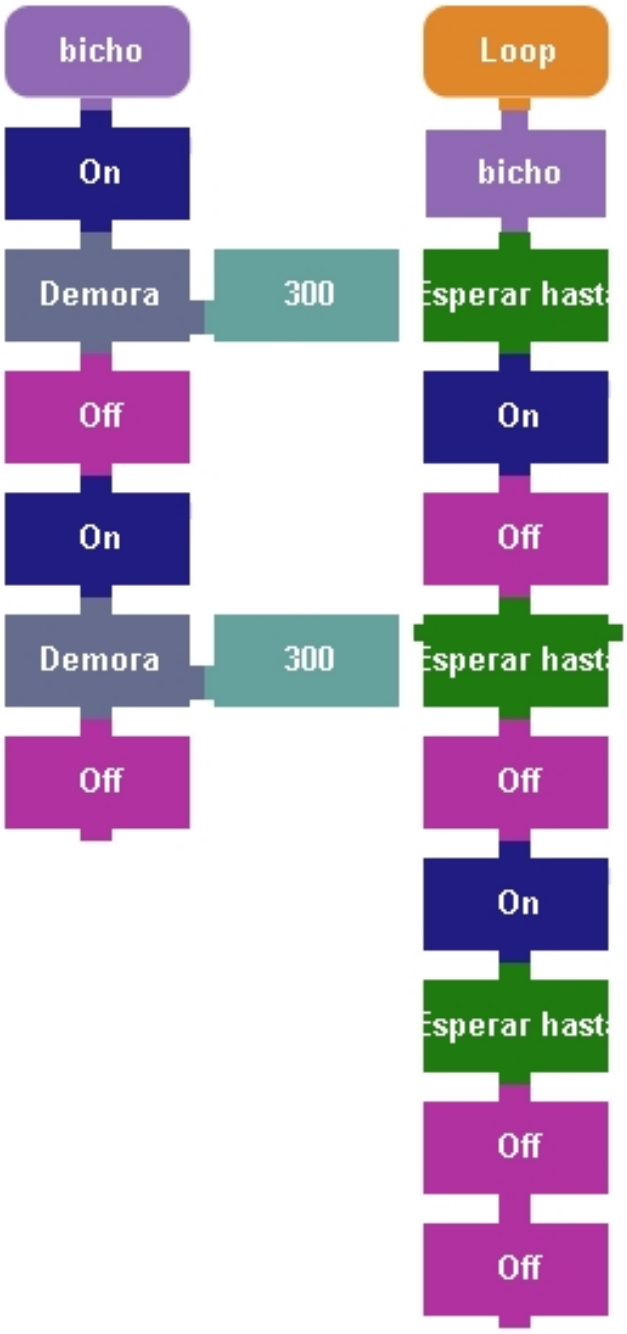

bicho
On
Demora
300
Off
On
Demora
300
Off
Loop
bicho
If
stop
stop
Off
Off

```
void setup(){
pinMode(Pin10, OUTPUT);
pinMode(Pin7, OUTPUT);
pinMode(Pin0, INPUT);
}

void loop(){
bicho();val = analogRead(Pin0);
compare=500;
if (val < compare){stop();}
else {}
}

void bicho(){
digitalWrite(Pin7, HIGH);
delay(Variable54);
digitalWrite(Pin7, LOW);
analogWrite(Pin10,255);
delay(Variable54);
analogWrite(Pin10,0);
}

void stop(){
digitalWrite(Pin7, LOW);
analogWrite(Pin10,0);
}
```

5) Robot-bicho al completo

Duplicando la presencia del sensor de luz, puedo dotar al robot-bicho de un aspecto más atractivo, al mostrar las resistencias como si fuesen sus antenas. Ahora la idea es que pueda guiar su trayectoria con la mano: al separar suficientemente las LDR, puedo ampliar el programa del apartado anterior al tener en cuenta las dos entradas digitales. He decidido cambiar el condicional IF por el de ESPERAR HASTA, de modo que el móvil avanzará recto, hasta que con mi mano cree la suficiente sombra para detener el motor que está en el mismo lado que el sensor sombreado y el robot-bicho girará apenas unos segundos, precisamente, hacia ese lado. Repitiendo el proceso varias veces, en ambas antenas, se comprobará que la dirección marcada por la mano y su sombra sobre la LDR correspondiente, será la que determine la trayectoria del móvil.


SUPER ALKALINE
E-Block - 9 V


bicho
On
Demora
300
Off
On
Demora
300
Off
Loop
bicho
Esperar hast
On
Off
Esperar hast
Off
On
Esperar hast
Off
Off

```
void loop(){
bicho();while(analogRead(Pin0)<=500){
delay(10);
}
digitalWrite(Pin7, HIGH);
analogWrite(Pin10,0);
while(analogRead(Pin2)<=500){
delay(10);
}
digitalWrite(Pin7, LOW);
analogWrite(Pin10,255);
while(analogRead(Pin1)<=600){
delay(10);
}
digitalWrite(Pin7, LOW);
analogWrite(Pin10,0);
}

void bicho()
{
digitalWrite(Pin7, HIGH);
delay(Variable54);
digitalWrite(Pin7, LOW);
analogWrite(Pin10,255);
delay(Variable129);
analogWrite(Pin10,0);
}
```

6) Conclusión

Construir un "ardobicho" como el propuesto en este capítulo, es un proceso relativamente sencillo (a excepción quizás de la placa electrónica con los mosfet, donde se necesitan ciertas destrezas añadidas) y permite seguir proponiendo soluciones de programación con distintos sensores y activando simultáneamente los leds de la placa o el piezoeléctrico.