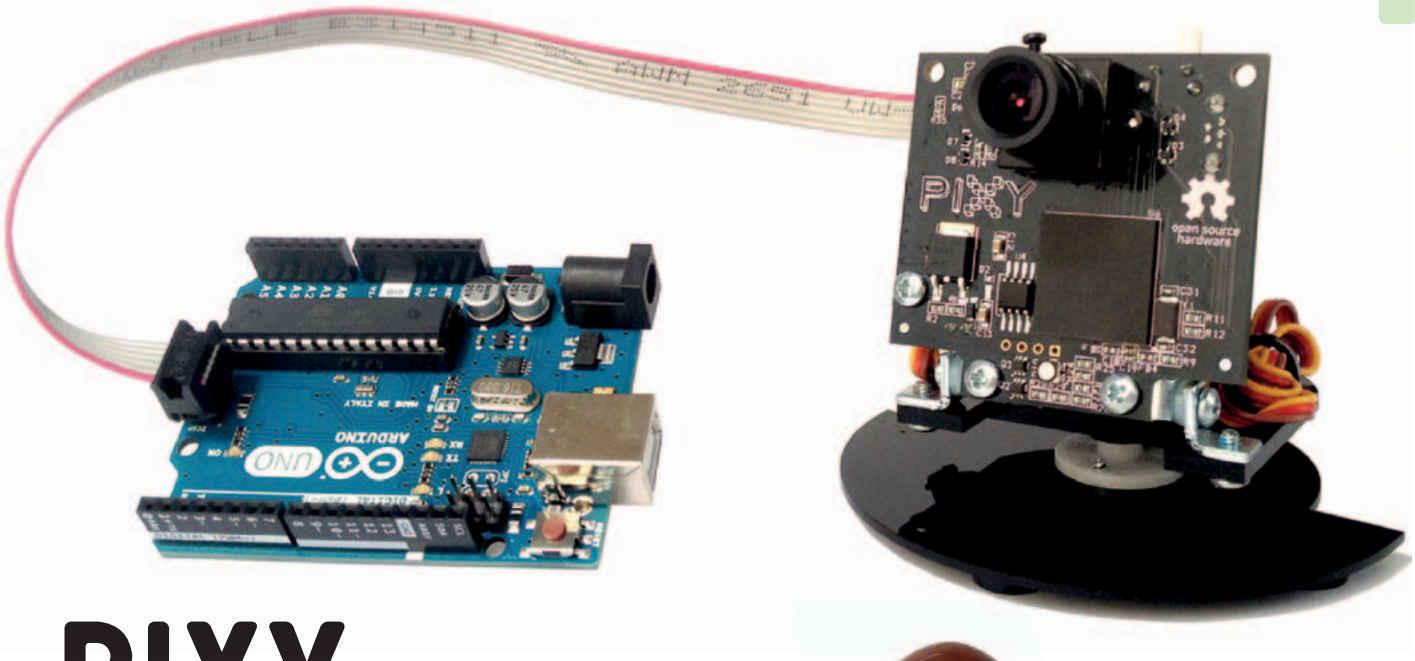


PIXY: LA CÁMARA QUE DISTINGUE

open source hardware

Primer contacto con la cámara capaz de reconocer el color de los objetos y localizar su posición.

ALAN PIPITONE

Hasta hace poco tiempo era verdaderamente difícil desarrollar aplicaciones capaces de explotar algoritmos de visión artificial, sobre todo utilizando sistemas con recursos hardware limitados. Afortunadamente, las cosas han cambiado: además de los distintos frameworks open source, que ayudan a los desarrolladores a implementar fácilmente incluso los algoritmos más complejos, se están difundiendo periféricos completos tanto de la parte hardware, que se ocupa de adquirir las imágenes provenientes del mundo externo, como de la parte software, capaz de analizar e interpretar tales imágenes. Con tales periféricos es posible crear aplicaciones dotadas de visión artificial de manera verdaderamente simple. Basta utilizar estos dispositivos junto con un microcontrolador

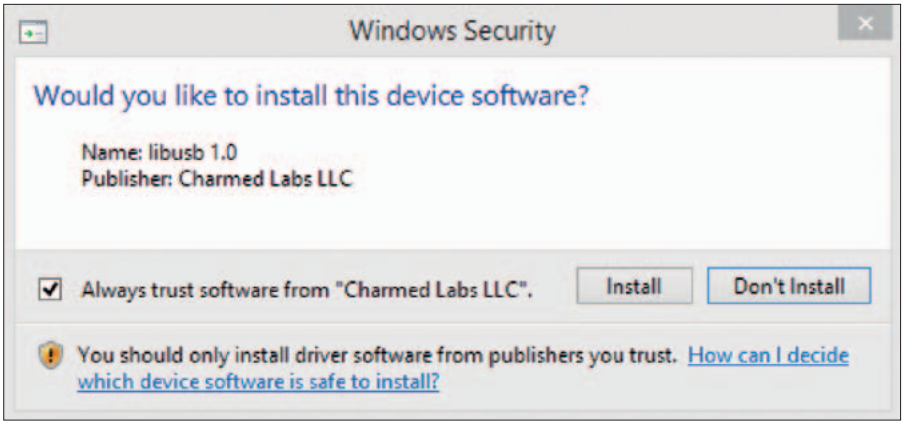


***Fig. 1** - Instalación del software PixyMon.*

o con una tarjeta de creación de prototipos, como por ejemplo Arduino. Estos periféricos se ocupan de toda la parte de cálculo, dejando libre los recursos hardware del dispositivo al cual se conectan, intercambiando con el solo las informaciones esenciales. Así cualquiera, de manera extremadamente simple, podrá realizar aplicaciones capaces de observar y analizar el ambiente circundante.

En este artículo nos ocuparemos de examinar el funcionamiento de la cámara Pixy, dispositivo open source capaz de localizar la posición de los objetos y comunicar las coordenadas a Arduino o a otros microcontroladores.

INSTALAR LOS COMPONENTES NECESARIOS

En primer lugar, tenemos que tener una cámara Pixy; es posible adquirirla en Nueva Electrónica en la página web *www.nuevaelectronica.com* (el código del producto es 7300-PIXYCAM).

En este artículo utilizaremos también el modulo pan/tilt, accesorio que permite a las cámaras Pixy seguir los movimientos de los objetos; esto no es obligatorio ya que podrías construirlo tu mismo, aun así aconsejamos adquirirlo ya listo. También el modulo pan/tilt se puede adquirir en Nueva Electrónica.

Una vez recuperado todo el material hardware, hay que descargar el software de gestión, necesario tanto para preparar fácilmente los objetos que la cámara Pixy debe reconocer, como para efectuar el debug de la cámara misma. El software se llama PixyMon y se puede descargar de la página web *http://cmucam.org/projects/cmucam5/files*. Si tenéis un MAC deberéis descargar el archivo *PixyMon_mac-0.1.49.dmg*, abrirlo y arrastrar la aplicación PixyMon en la carpeta Applications. Si en su lugar tenéis un PC Windows, deberéis descargar el archivo *pixymon_windows-0.1.49.exe*, abrir el ejecutable, iniciar el proceso de instalación y finalmente esperar que aparezca la pantalla mostrada en la **Fig. 1**.

Una vez que aparezca la pantalla, debéis continuar la instalación pulsando el botón "Install".

Si tenéis un PC con sistema operativo Linux deberéis sin embargo descargaros las fuentes y compilarlas; el proceso entero se muestra en la guía oficial disponible en *http://cmucam.org/*

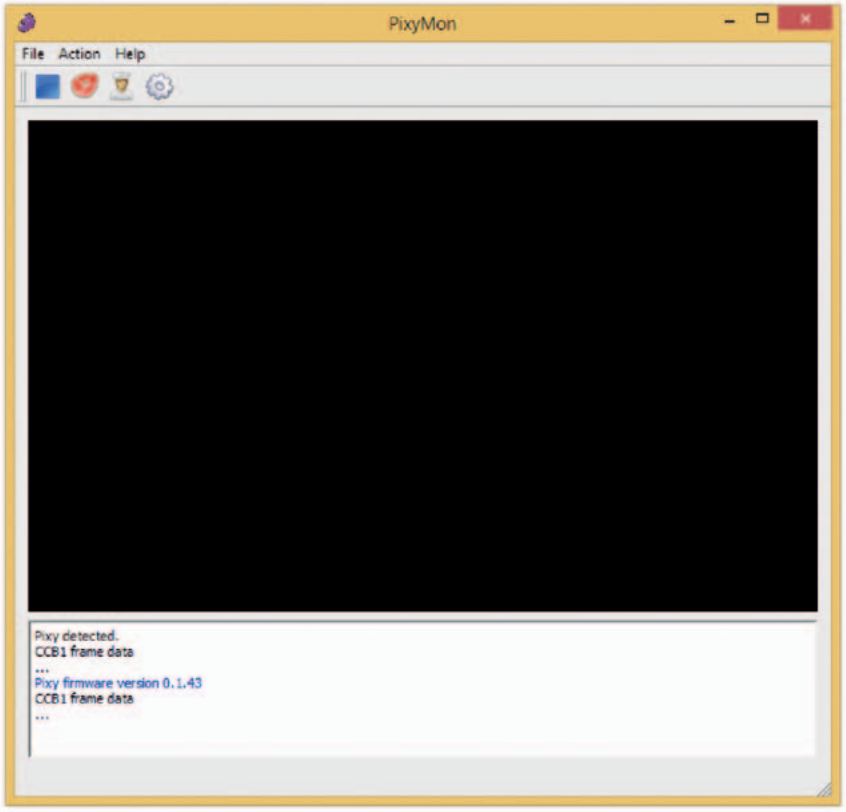


***Fig. 2** - Pantalla principal del programa PixyMon.*

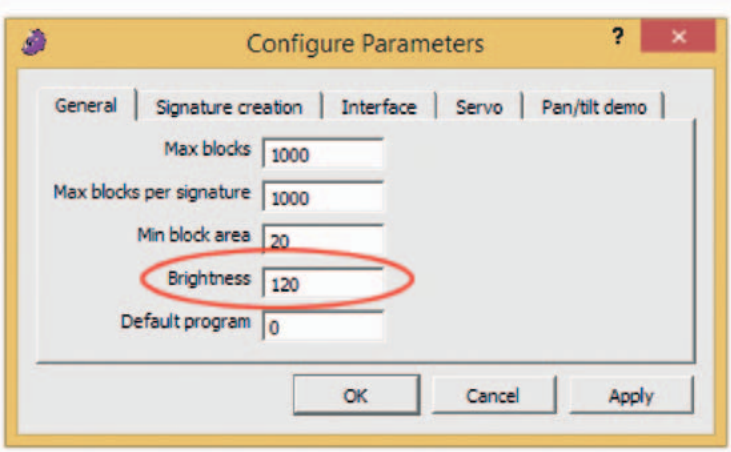


Fig. 3 *Configuración de la luminosidad.*

projects/cmucam5/wiki/ Installing_PixyMon_on_Linux.

PIXYMON: EL SOFTWARE DE GESTION

Terminado el proceso de instalación del software, podemos conectar la cámara Pixy a nuestro PC. En la parte posterior de Pixy está presente un conector de tipo miniUSB; consigamos por tanto el cable oportuno y conectemos la cámara a nuestro ordenador. Si en el ordenador tenemos instalado un sistema operativo Windows, debemos esperar que el mismo detecte la conexión de la cámara; una vez detectada, el dispositivo se instalará de manera automática. Con un MAC, sin embargo, bastará simplemente conectar la cámara al puerto USB, sin tener que esperar ningún tiempo.
Terminado el proceso de conexión es posible abrir el programa PixyMon, del cual podemos ver su pantalla principal en la **Fig. 2**.
Para hacer que la pantalla negra sea sustituida por las imágenes del flujo de video, bastará hacer clic sobre el icono en forma de filete, situado arriba a la izquierda. Si las imágenes resultan demasiado oscuras es posible modificar la luminosidad de Pixy. Desde el menú *Configure Parameters*, accesible en el icono en forma de engranaje, podemos modificar el parámetro *Brightness*, como se puede ver en la **Fig. 3**. Incrementando el valor este parámetro las imágenes se verán más claras; al contrario serán más oscuras. Pixy es capaz de aumentar o disminuir la luminosidad de manera automática, aun siendo el objeto a reconocer demasiado oscuro o demasiado claro se deberá ajustar manualmente el valor de la luminosidad.
En este punto podemos enseñar a Pixy el objeto a reconocer; el dispositivo reconocerá el objeto en base a su color, por lo tanto es importante que este sea el más homogéneo posible. Realizada esta premisa, podemos iniciar la fase de aprendizaje. Debemos situar el objeto a reconocer delante del objetivo de la cámara, después de esto, desde el menú *Action*, debemos seleccionar "Set Signature 1…". Una vez realizado, teniendo pulsado el botón izquierdo del ratón, debemos seleccionar el cuerpo del objeto, como se puede ver en la **Fig. 4**. Soltando el botón del ratón, la cámara memorizará el color del

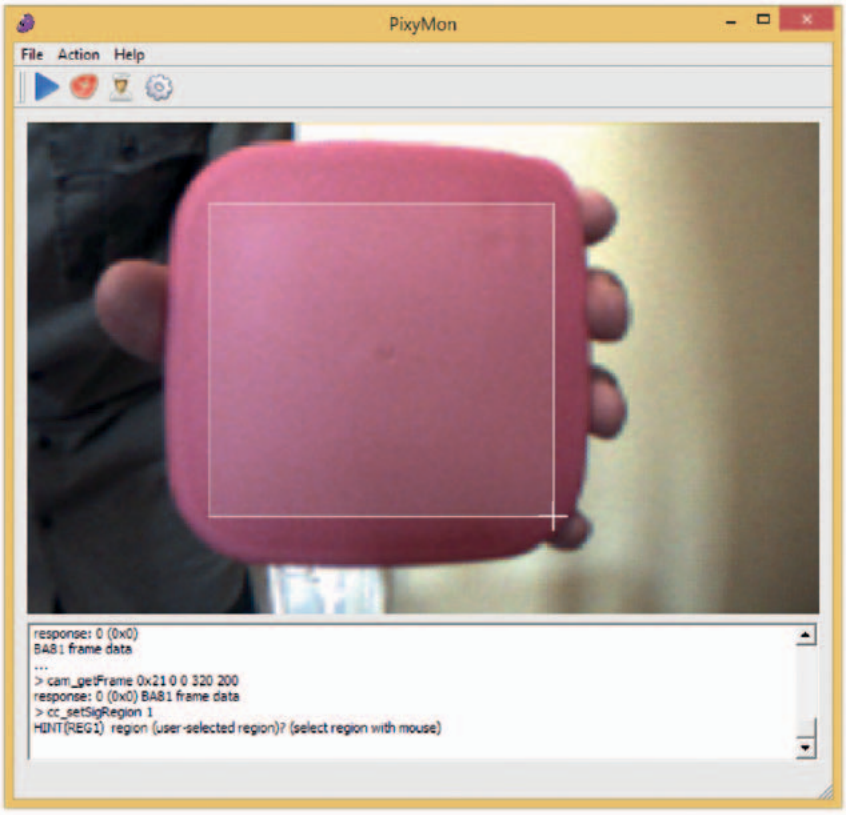


Fig. 4 *- Fase de aprendizaje del color de un objeto.*

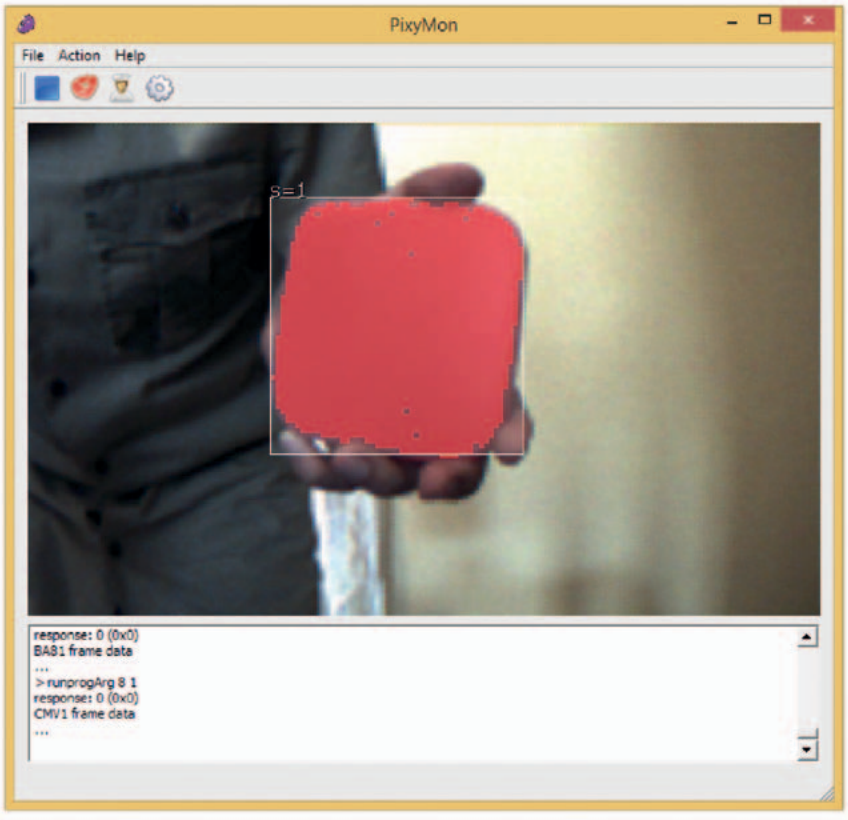


***Fig. 5** - Reconocimiento del objeto.*

objeto como se puede ver en la **Fig. 5**.
También es posible memorizar un objeto sin utilizar PixyMon: pulsando el botón situado en la parte superior de la cámara, respetando la secuencia oportuna, es posible memorizar el objeto situado delante del objetivo. Sin embargo este método es más complejo y lo veremos cuando conectemos la cámara con Arduino.
Independientemente del modo utilizado, Pixy es capaz de memorizar hasta siete objetos diferentes.
La cámara las guardara en memoria mientras no sean eliminadas o sobrescritas, de hecho desconectar Pixy del PC o de su fuente de alimentación no conlleva la eliminación de los objetos capturados.
El software de gestión PixyMon nos permite también testear el sistema pan/tilt; para hacer esto, es necesario memorizar al menos un objeto y luego, desde el menú *Action*, bastará con pulsar sobre "Run pan/tilt demo". Veremos entonces el servo del módulo pan/tilt mover la cámara con el fin de encuadrar el objeto memorizado anteriormente.
Es posible que durante la demo del sistema pan/tilt la cámara se desconecte del PC varias veces. Esto ocurre si se tiene un cable USB de mala calidad o si nuestro PC tiene una gestión de los puertos USB no demasiado óptima.

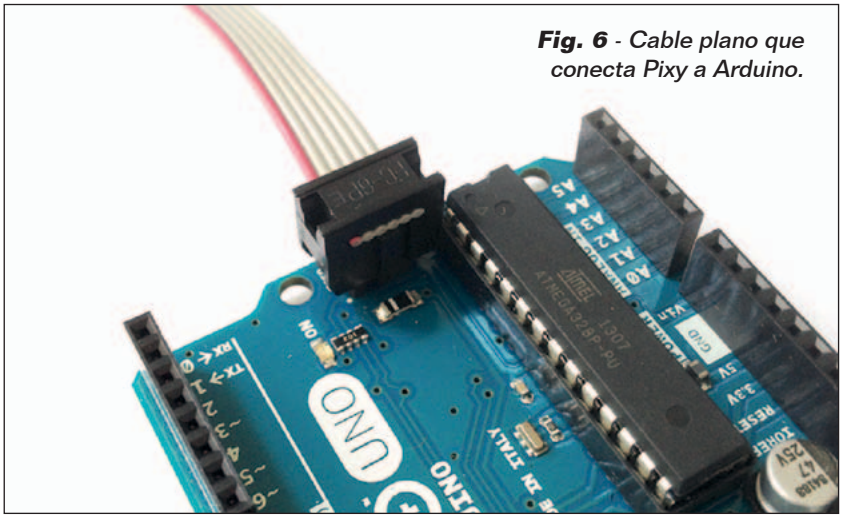


***Fig. 6** - Cable plano que conecta Pixy a Arduino.*

CONECTAR ARDUINO

Conectar Pixy a Arduino es bastante simple: basta con el cable plano incluido en el paquete de la cámara, conectar el conector más grueso a la parte de atrás de Pixy (el conector impide equivocarse de posición) y finalmente insertar el conector más pequeño en la conexión ICSP de Arduino. Prestar atención a la orientación del cable rojo, visible en la **Fig. 6**.
Ahora podemos conectar Arduino a nuestro PC y cargar nuestros primeros sketchs. Es posible descargar las librerías y los sketch de ejemplo desde el link *http://cmucam.org/attachments/download/1065/arduino_pixy-0.1.5.zip*.
Para importar las librerías es necesario abrir el IDE de Arduino, y después, desde el menú *Sketch*, escribir el comando "Im-

Listado 1

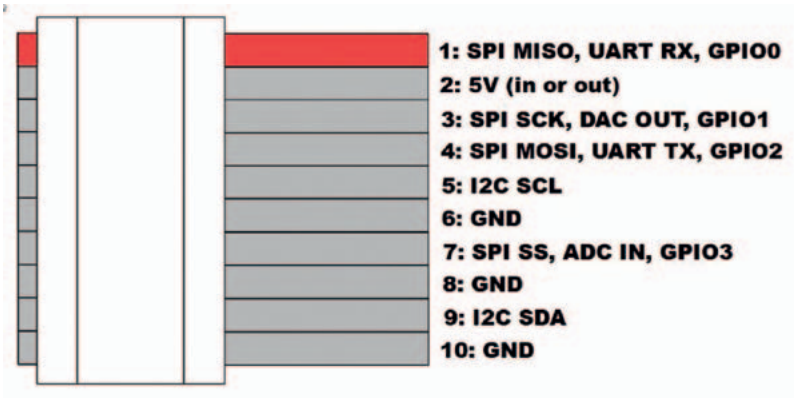
```
//
// begin license header
//
// This file is part of Pixy CMUcam5 or "Pixy" for short
//
// All Pixy source code is provided under the terms of the
// GNU General Public License v2 (http://www.gnu.org/licenses/gpl-2.0.html).
// Those wishing to use Pixy source code, software and/or
// technologies under different licensing terms should contact us at
// cmucam@cs.cmu.edu. Such licensing terms are available for
// all portions of the Pixy codebase presented here.
//
// end license header
//

/*
   06.04.2014 v0.1.3 John Leimon
      + Now using pixy.init() to initialize Pixy in setup().
*/

#include <SPI.h>
#include <Pixy.h>

Pixy pixy;

void setup()
{

   Serial.begin(9600);
   Serial.print("Starting...\n");

   pixy.init();
}

void loop()
{
   static int i = 0;
   int j;
   uint16_t blocks;
   char buf[32];

   blocks = pixy.getBlocks();

   if (blocks)
   {
      i++;

      if (i%50==0)
      {
         sprintf(buf, "Detected %d:\n", blocks);
         Serial.print(buf);
         for (j=0; j<blocks; j++)
         {
            sprintf(buf, " block %d: ", j);
            Serial.print(buf);
            pixy.blocks[j].print();
         }
      }
   }
}
```

port Library…", después "Add Library…" y finalmente seleccionar la carpeta que contiene los archivos del .zip descargado. Esta operación cargará también los sketchs de ejemplo. Para ver el código es suficiente con pulsar sobre el *File*, después sobre *Examples* y finalmente en el submenú *Pixy*.

PRIMER SKETCH: HELLO WORLD

En total los sketchs de ejemplo son cuatro. El primero se llama hello_world, y el código fuente se puede ver en el **Listado 1**.
En él, como primera cosa se realiza la importación de las librerías necesarias para establecer la correcta comunicación entre Arduino y la cámara. Posteriormente se crea un objeto de tipo Pixy, que después se inicializa en la función setup.
En esta función se establece también la comunicación serie con el PC, así podremos ver los distintos mensajes de debug.
El método **getBlocks()**, llamado en el cuerpo de la función loop, sirve para contar cuantos objetos memorizados se encuentran delante del objetivo de la cámara.
El método retorna un número entero, cuyo valor viene dado por el número total de objetos reconocidos. Por cada objeto identificado es posible recuperar una serie de datos, como por ejemplo sus coordenadas y las dimensiones en pixel de la imagen memorizada relacionada con él.
La secuencia "for" siguiente sirve justamente a eso.

***Fig. 7** - Esquema del conector plano de Pixy.*



Listado 2

```
//
// begin license header
//
// This file is part of Pixy CMUcam5 or "Pixy" for short
//
// All Pixy source code is provided under the terms of the
// GNU General Public License v2 (http://www.gnu.org/licenses/gpl-2.0.html).
// Those wishing to use Pixy source code, software and/or
// technologies under different licensing terms should contact us at
// cmucam@cs.cmu.edu. Such licensing terms are available for
// all portions of the Pixy codebase presented here.
//
// end license header
//

#include <SPI.h>
#include <Pixy.h>

#define X_CENTER  160L
#define Y_CENTER  100L
#define RCS_MIN_POS        0L
#define RCS_MAX_POS        1000L
#define RCS_CENTER_POS     ((RCS_MAX_POS-RCS_MIN_POS)/2)

class ServoLoop
{
public:
   ServoLoop(int32_t pgain, int32_t dgain);

   void update(int32_t error);

   int32_t m_pos;
   int32_t m_prevError;
   int32_t m_pgain;
   int32_t m_dgain;
};

ServoLoop panLoop(500, 800);
ServoLoop tiltLoop(700, 900);

ServoLoop::ServoLoop(int32_t pgain, int32_t dgain)
{
   m_pos = RCS_CENTER_POS;
   m_pgain = pgain;
   m_dgain = dgain;
   m_prevError = 0x80000000L;
}

void ServoLoop::update(int32_t error)
{
   long int vel;
   char buf[32];
   if  (m_prevError!=0x80000000)
   {
      vel = (error*m_pgain + (error - m_prevError)*m_dgain)>>10;
      //sprintf(buf, "%ld\n", vel);
      //Serial.print(buf);
      m_pos += vel;
      if (m_pos>RCS_MAX_POS)
        m_pos = RCS_MAX_POS;
      else if (m_pos<RCS_MIN_POS)
        m_pos = RCS_MIN_POS;

      //cprintf("%d %d %d\n", m_axis, m_pos, vel);
   }
   m_prevError = error;
}

Pixy pixy;

void setup()
```

(Continua)

Cada objeto encontrado es automáticamente guardado en el array blocks del objeto pixy; la instrucción **pixy.blocks[j].print()** nos permite por tanto acceder al objeto identificado e imprimir sus datos correspondientes.
Además del método "print" cada objeto tiene también los siguientes atributos: **x**, **y**, **width**, **height** y **signature**. Este último contiene el número del objeto identificado, que varía de 1 a 7 en base al orden en el cual el objeto ha sido memorizado en la cámara.

SEGUNDO SKETCH: PAN/TILT

El sketch llamado "pantilt" permite controlar el modulo pan/tilt directamente desde Arduino. El código fuente del sketch se puede ver en el **Listado 2**.
Después de la importación de las librerías y la definición de algunas constantes, encontramos el código de la clase **ServoLoop**. Para entender el objetivo de tal clase debemos analizar el código del método **update**.
Este método sirve para mantener el objeto encontrado siempre en el centro del objetivo. En el código de la función podemos encontrar un algoritmo de cálculo similar al de un sistema PID. En particular se utiliza el argumento "error" para calcular el valor a asignar a los servo, valor que se guarda después en la variable m_pos. El argumento error contiene la diferencia entre la posición del objeto localizado y el centro de la cámara. En el cuerpo de la función "loop" podemos ver que tal diferencia se calcula en dos partes, una de las cuales está contenida en la variable **panError**, mientras la otra esta guardada en la variable **tiltError**.
Estos valores se utilizan después por dos instancias de la clase ServoLoop, **panLoop** y **tiltLoop**,

que a través del método propio **update** se encargan de actualizar el valor a asignar a los dos servo. Sucesivamente la función **pixy.setServos** establece físicamente la posición de los dos servo.
Este sketch funciona perfectamente solo si se dispone del sistema pan/tilt oficial, es decir, el que es posible adquirir ya listo para usarse.
Si utilizas otro tipo de servo, debes hacer alguna prueba práctica para comprobar la compatibilidad y muy probablemente cambiar algunos parámetros, como por ejemplo el valor de varias ganancias del sistema PID o la posición máxima que los servo pueden asumir.

LOS ULTIMOS SKETCH

En los sketchs que acabamos de ver, la comunicación entre Arduino y Pixy se realiza a través del bus SPI. Sin embargo es posible utilizar también buses diferentes, en particular el I²C y la UART: los últimos dos sketch de ejemplo muestran justamente como utilizar estos. Pero antes

Listado 2 (sigue)

```
{
   Serial.begin(9600);
   pixy.init();
}

void loop()
{
   static int i = 0;
   int j;
   uint16_t blocks;
   char buf[32];
   int32_t panError, tiltError;

   blocks = pixy.getBlocks();

   if (blocks)
   {
      panError = X_CENTER-pixy.blocks[0].x;
      tiltError = pixy.blocks[0].y-Y_CENTER;

      panLoop.update(panError);
      tiltLoop.update(tiltError);

      pixy.setServos(panLoop.m_pos, tiltLoop.m_pos);

      i++;

      if (i%50==0)
      {
        sprintf(buf, "Detected %d:\n", blocks);
        Serial.print(buf);
        for (j=0; j<blocks; j++)
        {
          sprintf(buf, " block %d: ", j);
          Serial.print(buf);
          pixy.blocks[j].print();
        }
      }
   }
}
```

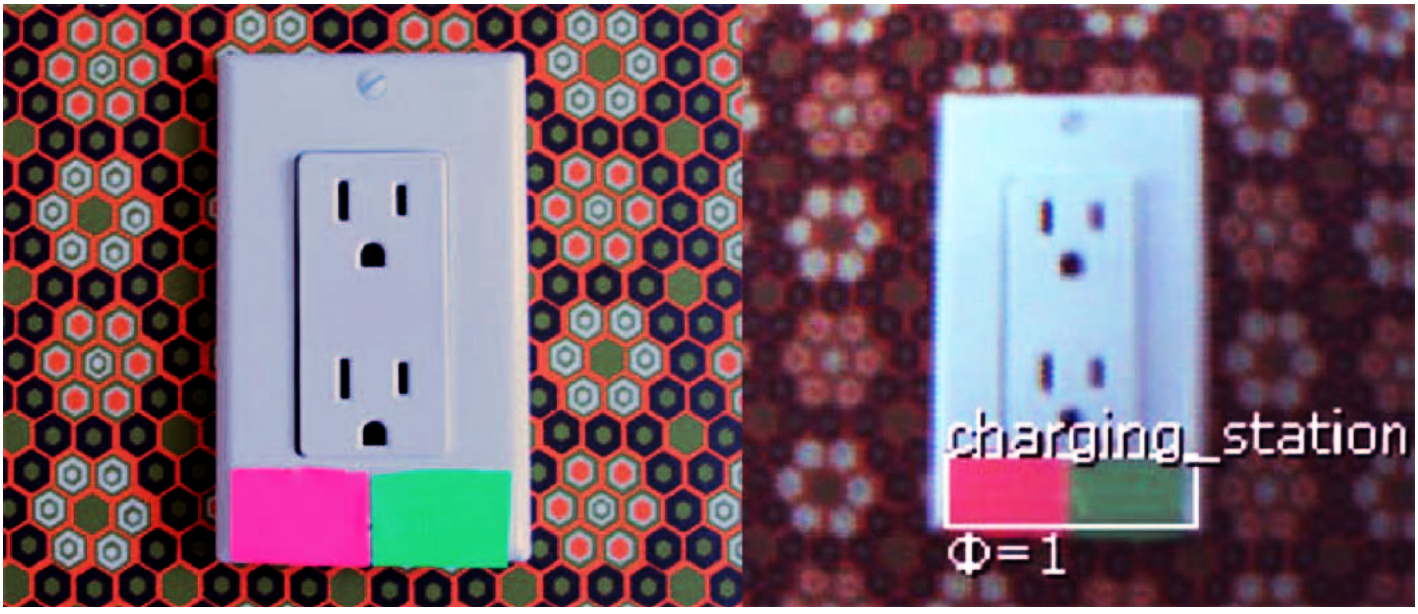


***Fig. 8** - Color Code formado por dos colores.*

Listado 3

```
// begin license header
//
// This file is part of Pixy CMUcam5 or "Pixy" for short
//
// All Pixy source code is provided under the terms of the
// GNU General Public License v2 (http://www.gnu.org/licenses/gpl-2.0.html).
// Those wishing to use Pixy source code, software and/or
// technologies under different licensing terms should contact us at
// cmucam@cs.cmu.edu. Such licensing terms are available for
// all portions of the Pixy codebase presented here.
//
// end license header
//

#include <Wire.h>
#include <PixyI2C.h>


PixyI2C pixy;


void setup()
{
  Serial.begin(9600);
  Serial.print("Starting...\n");

  pixy.init();
}


void loop()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[32];

  blocks = pixy.getBlocks();

  if (blocks)
  {
    i++;

    if (i%50==0)
    {
      sprintf(buf, "Detected %d:\n", blocks);
      Serial.print(buf);
      for (j=0; j<blocks; j++)
      {
        sprintf(buf, " block %d: ", j);
        Serial.print(buf);
        pixy.blocks[j].print();
      }
    }
  }
}
```

de utilizar los dos buses debemos modificar la conexión entre Arduino y Pixy, porque el cable plano suministrado con la cámara no conecta con todos los pines de Pixy, solo aquellos necesarios para la comunicación vía SPI y pocos más. En la **Fig. 7** podemos observar el esquema del conector para cable plano situado en la parte trasera de la cámara.

Para utilizar el bus I²C es necesario conectar el SDA de Arduino al pin 9 de Pixy, el pin SCL al pin 5 de Pixy y finalmente el pin GND de Arduino al pin 6, 8 o 10 de la cámara. El dispositivo tiene montadas las dos resistencias de pull-up necesarias para el correcto funcionamiento del bus I²C, por lo tanto no debemos preocuparnos de ellas. En cuanto a la conexión UART, es suficiente conectar el pin TX de Arduino al pin 1 de Pixy, el pin RX al pin 4 de Pixy y finalmente el pin GND de Arduino al pin 6, 8 o 10 de la cámara.

Terminada la descripción de la conexión entre Arduino y Pixy, podemos analizar los dos sketch restantes. En el **Listado 3** se puede ver el código fuente del sketch llamado i2c, mientras en el **Listado 4** se puede ver el código del sketch llamado uart. El código fuente de ambos listados es muy parecido al del **Listado 1**. Lo que cambia es la importación de las librerías, que varían dependiendo del bus utilizado, y la definición del objeto Pixy, el cual en el **Listado 3** se define con la instrucción **PixyI2C pixy**, mientras en el **Listado 4** es definido a través de **PixyUART pixy**.

RECONOCIMIENTO

Ahora podemos ver como reconocer nuevos objetos sin la ayuda de un PC. Cuando la cámara es alimentada, el LED RGB situado en la parte frontal empieza a parpadear: apenas que el LED se apague es necesario pulsar al menos un segundo el pulsador situado en la parte superior de Pixy. El LED en este momento se encenderá nuevamente y tomara un color blanco, después cambiará a rojo y sucesivamente cambiara a otros colores. Cuando cambie a rojo deberemos soltar el pulsador A y el LED asumirá el color del pixel central de la imagen que el objetivo está encuadrando. El diodo RGB puede utilizarse como *feedback* para entender si el objeto a reconocer esta al centro de la cámara o no.

Cuando el *feedback* nos convence, podemos guardar el objeto en la

memoria de Pixy. Para hacerlo es suficiente con pulsar y soltar el pulsador de la cámara. Si Pixy determina que las características del objeto son buenas (en concreto la saturación del color) el LED realizará un breve parpadeo y el proceso podrá considerarse finalizado. Sin embargo si el objeto no es idóneo para ser guardado, entonces el diodo simplemente se apagara y deberemos rehacer el proceso.

CONCLUSIONES

Como hemos observado es posible crear, de manera bastante simple, sistemas embebidos capaces de reconocer los objetos y localizar su posición.
Sin embargo, la fuerza de Pixy no se limita solo a lo que hemos visto hasta ahora.
El nuevo firmware de Pixy y la nueva versión de PixyMon ofrecen otras funciones muy interesantes. Un ejemplo es la posibilidad de memorizar los "Color Codes", distintas combinaciones de colores que forman una única *signature*. En este modo es posible crear distintos marcadores unívocos, como se puede ver en la **Fig. 8**.
Esto aumenta notablemente el potencial ofrecido por el dispositivo.
No hemos hablado del tema en este artículo ya que, tanto el nuevo firmware como la nueva versión de Pixymon están aún en fase beta.
En próximos artículos afrontaremos distintos proyectos interesantes, profundizando en las distintas funciones avanzadas que la cámara pone a nuestra disposición, así que permaneced atentos.

(189025)

Listado 4

```
// begin license header
//
// This file is part of Pixy CMUcam5 or "Pixy" for short
//
// All Pixy source code is provided under the terms of the
// GNU General Public License v2 (http://www.gnu.org/licenses/gpl-2.0.html).
// Those wishing to use Pixy source code, software and/or
// technologies under different licensing terms should contact us at
// cmucam@cs.cmu.edu. Such licensing terms are available for
// all portions of the Pixy codebase presented here.
//
// end license header
//

#include "PixyUART.h"

PixyUART pixy;

void setup()
{
  Serial.begin(9600);
  Serial.print("Starting...\n");

  pixy.init();
}

void loop()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[32];

  blocks = pixy.getBlocks();

  if (blocks)
  {
    i++;

    if (i%50==0)
    {
      sprintf(buf, "Detected %d:\n", blocks);
      Serial.print(buf);
      for (j=0; j<blocks; j++)
      {
        sprintf(buf, " block %d: ", j);
        Serial.print(buf);
        pixy.blocks[j].print();
      }
    }
  }
}
```

el MATERIAL

La cámara Pixy (cod. PIXYCAM) está disponible al precio de 65,00 Euros. El modulo Pan/Tilt para esta cámara (cod. PIXYPANTILT) también está disponible al precio de 36,00 Euros. La tarjeta Arduino Rev.3 (cod. ARDUINOUNOREV3) está disponible por 24,50 Euros.

Precios IVA incluido sin gastos de envío.
Puede hacer su pedido en:
www.nuevaelectronica.com
pedidos@nuevaelectronica.com