Facultad
de
Ciencias

Manejador de bajo nivel para cámaras con procesamiento de imágenes integrado

Driver for integrated image-processing cameras

Trabajo de Fin de Grado
para acceder al

GRADO EN Ingeniería Informática

Autor: Ángel Pérez Fuentes

Director: Héctor Pérez Tijero

Co-Director: Mario Aldea Rivas

Octubre - 2017

Índice de contenido

Índice de figuras

Agradecimientos

En primer lugar, tengo el deseo de decir que agradezco de todo corazón el apoyo y la confianza que han depositado mi madre, mi padre mi hermana y los demás miembros de mi familia. Todas esas palabras llenas de fe que he recibido por parte de mi familia, han sido el impulso necesario para llegar a este punto de mi vida.

También me gustaría agradecer a los compañeros que empezaron el grado en el mismo momento que yo lo hice. Tanto a los que hoy están conmigo, como los que quedaron atrás, gracias por hacer de cada momento, algo único que siempre recordaré.

Por otro lado, quiero expresar lo afortunado y orgulloso que me siento de seguir conservando las amistades de mi infancia. Gracias por todas las carcajadas, momentos de alegría, momentos de tristeza, situaciones complicadas en las que habéis permanecido a mi lado, sin importar las consecuencias.

Y, por último, pero no por ello menos importante, manifestar mi agradecimiento a las dos personas que han hecho posible mi graduación como Ingeniero Informático, Héctor Pérez y Mario Aldea. No solamente por eso, sino también por su paciencia, sabiduría y por ser dos grandes personas al igual que profesores.

Estas palabras son pocas para mostrar mi gratitud a todas esas personas que quiero mencionar, sin embargo, me gustaría plasmar estos sentimientos para la prosperidad y recordar estos momentos.

Resumen

El principal concepto de este proyecto desarrollado durante el curso 2016-2017, era implementar un manejador de bajo nivel para cámaras con procesamiento de imágenes integrado, estas cámaras son NXTCam-v4 [2] y Pixy CMUcam5 [3], de las cuales hablaremos en capítulos posteriores.

Las cámaras utilizadas pertenecen a distintos fabricantes, así que, como es lógico, existen importantes diferencias a tener en cuenta. Por ello, el estudio por separado de cada una de ellas forma parte del trabajo realizado para conseguir este proyecto.

Otro aspecto a considerar, es el esfuerzo aplicado para conseguir la portabilidad de ambas cámaras en distintos entornos.

Como primera instancia, se desarrolló el proyecto utilizando como entorno un lego ev3-dev [4], el cual es un computador desarrollado por la empresa LEGO [7] en el que utilizaremos como sistema operativo un Debian Linux. Más tarde, comprobado que la funcionalidad de ambas cámaras era la esperada, se siguió el mismo procedimiento para conseguir la implementación en Raspberry Pi [8] que utilizaría MaRTE OS [5] como sistema operativo y un “brick” Pistorms como puente de comunicación I2C [1]. Cabe destacar, como se ha mencionado anteriormente, que se consiguió hacer del proyecto una librería funcional entre estos dos sistemas, manteniendo las mismas características.

Finalmente, dado por concluido el desarrollo del proyecto, se optaría por realizar un demostrador en el que se demuestre de forma práctica el éxito del proyecto.

Abstract

The main concept for this project , developed during the course 2016-2017, it is to know how Works one of the most used communication protocols in embedded systems, called I2C *(Inter-Integrated Circuit).*

The project is based on using tracking objects cameras, making use of controlled real-time environment. The cameras used, belong to different manufacturer, so, as it is logic, it exists important differences to know. By this way, part of the work applied to get this project was acquire knowdeledge of each camera. To be precise, cameras are; NXT Cam (manufacturer Mindsensors) and Pixy Cam (manufacturer Charmedlabs)

Another aspect to consider, it is the effort applied to make the project a portable system in different environments.

In the first instance, the project was developed by using a lego ev3-dev as developing platform, which make use of Debian Linux based operating system. Later, once it was checked the expected behaviour, I followed same way to make the system to work in Raspberry Pi which uses MaRTE OS as operating system and a brick Pistorms as I2C communication bridge. It is important to remark that it was gotten to keep portability mentioned before.

Finally, to finish the project development, it was decided to create a practical demonstration to show the success of the project.

1.- Introducción

1.1 Motivación

La robótica es una ciencia que, con el paso del tiempo, se ha introducido en nuestra vida diaria de una forma sutil, hasta el punto de hallarse en cualquier aspecto de la vida de una persona. El campo de la robótica es tan amplio y a su vez tan importante, que puede localizarse desde máquinas con fines médicos, hasta máquinas que controlen el transporte público de una ciudad, pasando por los dispositivos domésticos que cada día utilizamos, como por ejemplo una lavadora o una alarma de seguridad. Por ello, es muy importante concienciar a las generaciones venideras y brindarles un conocimiento base del tema. Un ejemplo que ha incentivado a muchos jóvenes es el primer campeonato de robótica educativa de Euskadi, denominado Gazterobotika [9].

Uno de los dispositivos más utilizados hoy en día para estos fines es la Raspberry Pi. Ésta adquirió su gran fama por ser un procesador de propósito general cuyo precio era excelente a pesar de ofrecer tantas características. Su bajo coste, ha permitido que el campo de la robótica educativa se desarrolle de una manera fácil y atractiva para las nuevas generaciones. La versatilidad de la Raspberry Pi permite realizar miles de proyectos dependiendo del conocimiento de programación y la imaginación del autor. De este modo, la posibilidad de crear tantos proyectos con un procesador tan barato, es una manera para animar a los estudiantes a conocer el mundo de la robótica.

Lo mismo ocurre con el "*brick*" Lego Mindstorms. Este "brick" es también un mini-computador con muchas funcionalidades disponibles. A diferencia de la Raspberry Pi, éste ofrece una imagen mucho más cercana y atractiva para el usuario, ya que en el kit del dispositivo podemos encontrar piezas con las que podremos montar nuestro proyecto de una manera más intuitiva.

Al ser el campo de la robótica tan amplio, es necesario focalizar la relación de esta ciencia con nuestro proyecto. Nuestro trabajo se centra en el campo de los sistemas empotrados, que consisten en dispositivos con un bajo consumo integrados en ciertos productos, y que sirven para controlar una o varias características de dicho producto, o bien ampliar su funcionalidad.

Los sistemas empotrados están generalmente ligados a lo que denominamos sistemas de tiempo real. Éstos se utilizan para reaccionar ante estímulos del entorno en un periodo de tiempo acotado. Uno de los ejemplos más cercano a nuestro proyecto es el sistema operativo MaRTE OS, desarrollado por la Universidad de Cantabria.

En concreto, en nuestro trabajo nos centraremos en sistemas de detección de objetos, que con el paso del tiempo están incrementando su importancia en conceptos relacionados con la inteligencia artificial. La detección automática de objetos es un tema muy interesante hoy en día ya que nos proporciona una mejor interacción con las máquinas. Por ejemplo, en el mundo de los videojuegos, hace unos años entró en auge la investigación de dispositivos que detecten el movimiento humano, permitiendo al

usuario tener la sensación de interaccionar directamente con el mundo virtual. Uno de los más conocidos es “kinect” [6] desarrollado por la empresa *Microsoft*.

Finalmente, otro aspecto relevante en un sistema empotrado es la comunicación entre dispositivos. La comunicación es necesaria para fines como mantener informado a un sistema. Por ejemplo, un sensor de humo debería ser capaz de comunicarse con la central para indicar que se está produciendo un incendio. Es aquí donde entran en juego los protocolos de comunicación. Uno de los más utilizados hoy en día es el protocolo I2C (Circuito Inter-Integrado).

1.2 Objetivos

El objetivo fundamental de este Trabajo de Fin de Grado es la creación de un manejador de bajo nivel para cámaras con procesamiento de imágenes integrado. En concreto este manejador de bajo nivel será implementado para dos cámaras distintas: NXTCam-v4 y Pixy CMUcam5. Junto con el manejador se implementará también una librería que facilite el uso de la cámara a las aplicaciones de usuario.

Por otro lado, otro de los objetivos relevantes del proyecto es facilitar la portabilidad entre plataformas. Para ello, utilizaremos un sistema operativo Linux de propósito general, así como para un sistema operativo de tiempo real denominado MaRTE OS.

Principalmente se utilizará el computador ev3-dev cuyo sistema operativo está basado en un Debian Linux. Sobre esta plataforma se desarrollará una librería que permita el uso de las cámaras “NXTCam” y “PixyCam”. En el desarrollo para MaRTE OS, el manejador se integrará en una librería para el control de sensores y actuadores ya realizada previamente [10]. Por tanto, en este caso parte de nuestro esfuerzo se centrará en extender esta librería para el uso de las cámaras con procesamiento de imágenes ya integrado.

Finalmente, se implementará un demostrador sobre una de las plataformas utilizadas que muestre la funcionalidad del trabajo desarrollado.

1.3 Metodología

La metodología utilizada para este proyecto fue la iterativa incremental. Ésta ha consistido en concretar reuniones semanalmente con el director y codirector del trabajo para presentar el trabajo realizado. En cada iteración se ha ido mejorando las prestaciones de la librería a partir de las versiones anteriores y se han realizado pruebas en cada iteración que probaban la funcionalidad añadida.

2.- Tecnología y Herramientas utilizadas.

2.1 Lego Mindstorms

Lego Mindstorms EV3 pertenece a la compañía Lego como su propio nombre indica. Esta empresa dedica su negocio a comercializar kits de este estilo, así como la fabricación de piezas de juguete que permitan el montaje de proyectos de una manera más cercana al usuario.

El "brick" programable utiliza el sistema operativo por defecto que está basado en Linux. Sin embargo, existen opciones para utilizar sistemas operativos diferentes, como por ejemplo en nuestro caso un Linux Debian.

Consta de un procesador ARM9 correspondiendo así con la tercera generación en la línea de LEGO Mindstorms.

La siguiente figura muestra la apariencia de este componente junto con los diferentes sensores y actuadores disponibles:

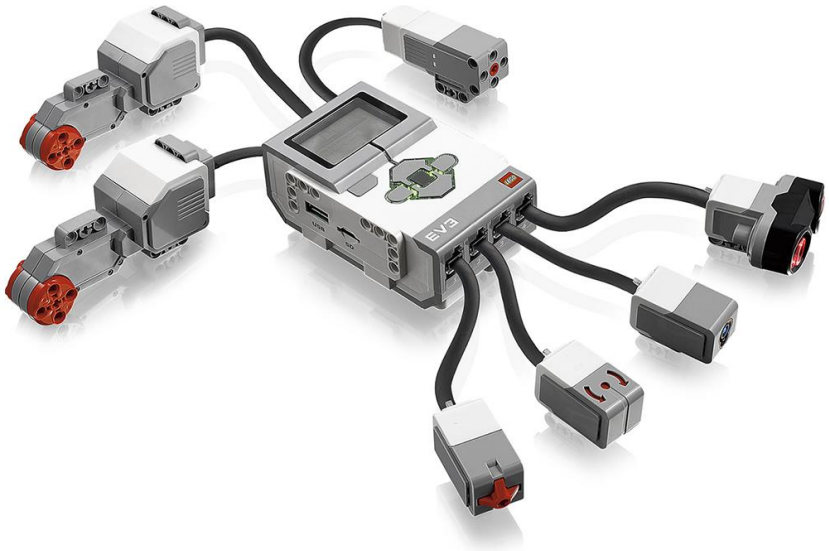


Figura 1: Kit de Lego Mindstorms EV3 [17]

En la figura 1 se observa que el brick se compone de 4 puertos para sensores y otros 4 para motores. Gracias a esto podemos disponer de un sistema realmente complejo, al brindarnos la capacidad de usar tantos periféricos.

Nuestra primera parte del proyecto se centrará en este entorno, para comprobar el funcionamiento de las cámaras explicadas en la introducción y realizar la librería para Linux.

2.2 Adaptador Pistorms Mindtorms

Existen adaptadores en el mercado que permiten a la Raspberry Pi utilizar distintos sensores y actuadores. Uno de estos adaptadores es el brick pistorms. Básicamente da la posibilidad de utilizar la Raspberry Pi como si fuera el brick de Lego (descrito una de las secciones anteriores).

En la siguiente imagen se muestra el adaptador Pistorms:

Figura 2: Adaptador Pistorms Mindstorms [18]

Este adaptador consta de dos bancos de registros (Banco A y Banco B) los cuales son utilizados para conectar diferentes periféricos a través de un cable RJ12. Cada banco dispone de dos entradas para sensores en las partes laterales (donde conectaremos nuestras cámaras) y otros dos para motores o actuadores en la parte superior. También mencionar que la pantalla es táctil, lo cual permite en ciertos casos una interacción con el robot más sencilla.

En el contexto de este proyecto se ha utilizado el adaptador Pistorms para utilizar las cámaras con procesamiento de imágenes integrado en la plataforma Raspberry Pi. El adaptador permite la comunicación entre ambos dispositivos a través del protocolo I2C. Para más información, consultar el Anexo I.

2.3 Raspberry Pi 1 Modelo B+

Raspberry Pi es un procesador de propósito general debido a su versatilidad. El objetivo principal por el que se desarrolló este computador fue para la enseñanza de ciencias en la computación en las escuelas. Existen varias versiones con distintas características y funciones, sin embargo, para nuestro proyecto utilizaremos el número 1 modelo B+ porque es una versión suficiente para desarrollar el proyecto en cuanto a rendimiento. Además, los pines de GPIO son compatibles con el adaptador Pistorms que utilizaremos.

Para conocer los principales elementos de esta versión de Raspberry Pi, se presenta la siguiente imagen:

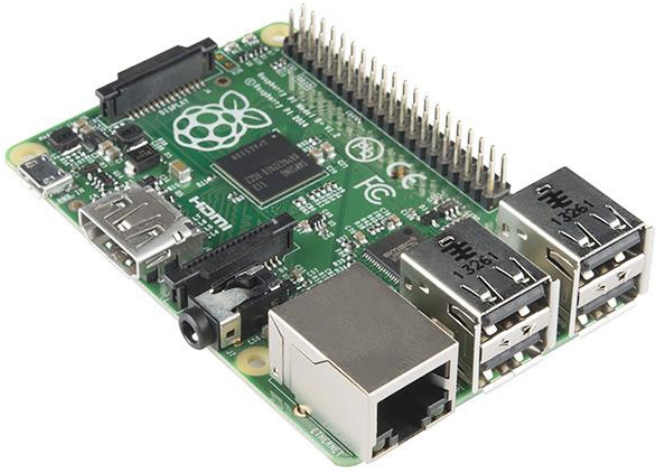


Figura 3: Raspberry Pi 1 Modelo B+ [16]

Este modelo dispone de 40 pines, de los cuales 26 son para el uso GPIO, mientras que los restantes se pueden tomar como toma de tierra o de voltaje. Dispone de una salida HDMI (la cual utilizaremos para controlar el comportamiento del dispositivo). Las principales características de este computador se detallan a continuación:

- Procesador: ARM 1176JZF-S A 700MHz
- RAM: 512 MB compartidos con la GPU
- Almacenamiento: Micro SD
- Conectividad:
    - 10/100 Ethernet (RJ-45)
- Consumo energético: 800 mA

En nuestro proyecto, este dispositivo será utilizado para enviar órdenes I2C hacia las cámaras. Por ultimo, la Raspberry Pi permite el uso de distintos sistemas operativos, como Linux o MaRTE OS, siendo éste ultimo el que será utilizado en esta plataforma.

2.4 MaRTE OS

MaRTE OS es un sistema operativo de tiempo real especialmente para sistemas embebidos. Fue diseñado y creado por la Universidad de Cantabria. Además, brinda la posibilidad de ejecutar aplicaciones "multi-thread" en tiempo real.
Algunas de las características que ofrece el "kernel" de MaRTE OS son:

- Soporte para lenguajes como Ada y C
- Soporte para arquitecturas x86, Raspberry y Linux
- Soporte total para tareas basadas en Ada
- Servicios POSIX.13 como "pthread" o "mutex".

Dentro de este proyecto, este sistema operativo se utilizará en la Raspberry Pi para trabajar con las cámaras.

2.5 NXT Cam v4

NXTCam es una cámara que se utiliza para la detección de objetos. Ésta incluye un motor de procesamiento de imagen en tiempo real, al que se puede acceder a través de un puerto estándar de NXT o EV3. Esta conexión permite obtener información post-procesada de lo que está viendo la cámara. Dicha información contiene las coordenadas de los objetos de interés.

La NXTCam permite realizar el seguimiento de hasta 8 colores de objetos a 30 imágenes por segundo dentro de su campo de visión, una vez actualizado el mapa de colores. Para hacer posible la detección de objetos, es necesario indicar en el mapa de colores cuales son los colores de los objetos de interés. Para ello utilizaremos el programa de configuración NXTCam View.

NXTCam View es una aplicación que sirve para controlar y configurar estáticamente la cámara NXTCam. Aunque la actualización del mapa de colores se puede hacer de forma dinámica, para este proyecto se decidió realizarlo a través de esta aplicación.

Las plataformas que soportan este software son Windows y Linux. Sin embargo, en la versión de Linux la detección de objetos era muy inexacta, así que, para utilizar esta aplicación, se tomó la decisión de utilizar solo la versión de Windows.

La siguiente imagen proporciona una idea de cómo detecta los objetos este software:

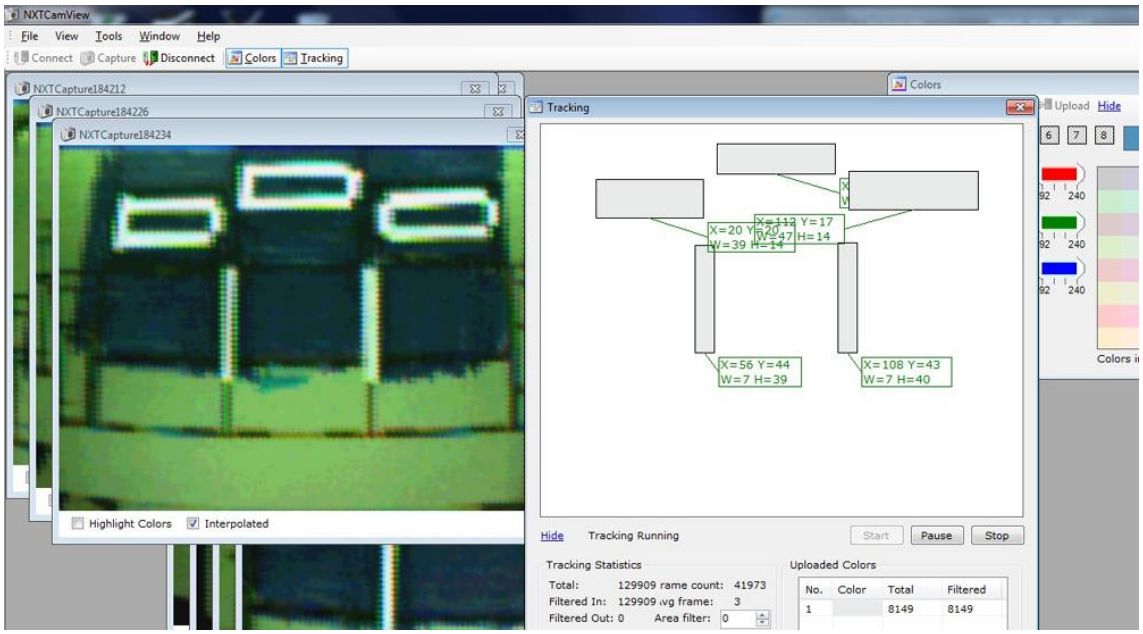


Figura 4: Vista de la aplicación NXTCam View

La cámara permite ajustar las lentes para establecer el foco de imagen. Una vez establecido el seguimiento de la misma, obtendremos imágenes con una resolución de 88 x 144 píxeles.

La siguiente ilustración presenta la apariencia de la cámara:

Figura 5: NXT CAM-v4

Para más información consultar el Anexo II.

2.6 Pixy Cam

Pixy es un pequeño y rápido sistema de visión de bajo coste. Es capaz de recordar hasta 7 colores distintos, lo que significa que puede procesar independientemente múltiples objetos de distinto color dentro del campo de visión de la cámara. El campo de visión es de 650x400 píxeles, además de poder seguir objetos a 50 imágenes por segundo.

Figura 6: Pixy Cam

La figura 6 presenta el diseño de pixy. Como se puede ver, todo el sistema está compactado en un solo chip, lo cual hace de ella una cámara fácil de transportar.

Para la configuración de la cámara utilizaremos el programa Pixy Mon. Éste permite configurar los colores de los objetos de interés, con el fin de hacer un seguimiento y obtener información de ellos como las coordenadas.

Se ha decidido utilizar solamente la versión de Windows por cuestiones de eficiencia y fiabilidad, al igual que con NXT Cam View.

Con el fin de establecer la conexión entre la Raspberry Pi y las Pixy Cam, se utilizará el adaptador Pixy de Mindsensors. Este adaptador ofrece la posibilidad de establecer la conexión a la cámara con un cable RJ12. En nuestro caso era estrictamente necesario hacerlo de esta manera, ya que la cámara va directamente conectada al Pistorms, como se ha hablado en secciones anteriores.

La ventaja más significante de utilizar este adaptador, es que nos permite controlar de la misma manera ambas cámaras, ya que pertenece a mindsensors (mismo fabricante que NXT Cam).

En la siguiente imagen aparece la Pixy con su adaptador:

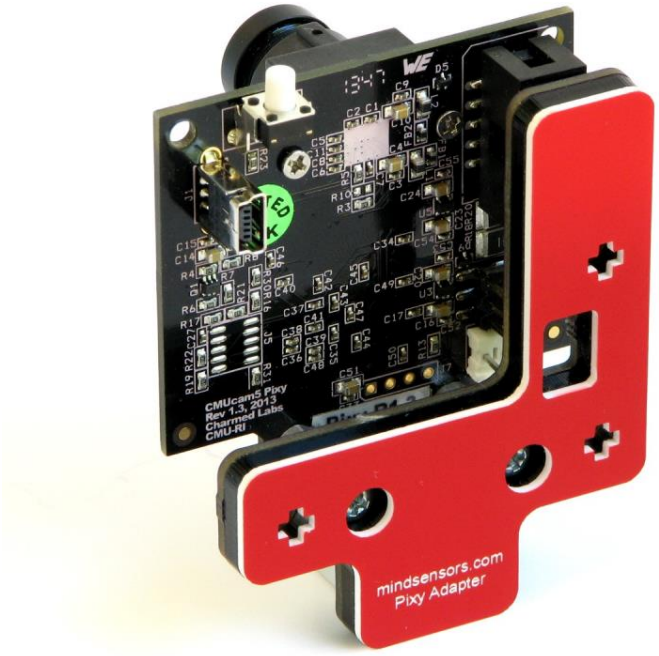


Figura 7: Adaptador Mindstorms para Pixy Cam

Para más información sobre el adaptador consultar el Anexo II

2.7 Protocolo I2C.

Como se explicó en la introducción, I2C es uno de los protocolos más utilizados hoy en día en sistemas embebidos. El protocolo se basa en el concepto Maestro-Esclavo. Este protocolo consiste en que el Maestro (normalmente el procesador) manda peticiones a través del bus I2C, dirigidas a los dispositivos esclavos.

En la imagen que se muestra a continuación, aparece un esquema gráfico de la estructura del protocolo:

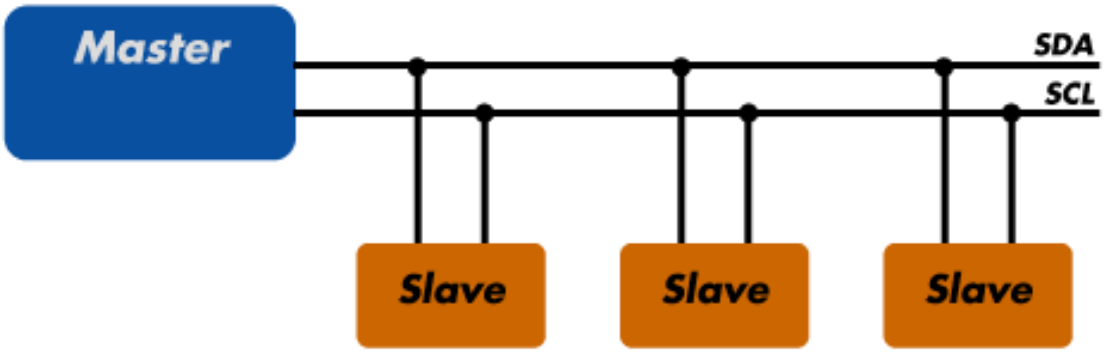


Figura 8: Estructura del protocolo I2C [15]

Como se observa en la figura 9, el bus consta de dos líneas, SDA (Serial Data) y SCL (Serial Clock). La línea SDA es la utilizada para la transmisión de los datos hacia y desde los dispositivos esclavos. Por otro lado, el maestro es el que controla directamente la línea SCL, estableciendo el reloj.

Para que se produzca una comunicación entre el Maestro Esclavo, es necesario que el Maestro inicie la petición I2C con una secuencia de bits y la finalice con lo que se conoce como bits de stop. Esto implica que la línea SCL se levante hasta que lleguen los bits de stop. El envío de datos se producirá siempre y cuando la línea SCL esté activa.

Este protocolo es el que hemos utilizado para comunicarnos entre las cámaras y la plataforma LEGO, y entre las cámaras y la plataforma Raspberry Pi a través del adaptador Pistorms.

2.8 Doxygen.

Doxygen es una herramienta que nos permite la creación de documentación desde lenguajes de programación como C, C++, PHP o Java, entre otros.

Doxygen es capaz de generar documentación en línea (HTML). Esta información es extraída directamente desde el código fuente. Además, una característica muy peculiar es la capacidad de extraer la estructura del código desde archivos fuente sin documentación.

En el Anexo III aparecen ejemplos de documentación extraída por Doxygen desde la librería creada.

3.- Cámaras de detección de objetos

Las cámaras que serán utilizadas en este proyecto se corresponden con NXT Cam v4, diseñada por Mindsensors, y Pixy Cam, diseñada por CharmedLabs. El fin de estas cámaras es realizar un seguimiento de objetos constante, gracias a la detección del color de los mismos.

Es muy importante realizar una configuración previa de las cámaras, para que estas mismas sean capaces de seguir los objetos de interés. La configuración consiste en indicar directamente a la cámara el color de los objetos que quieren ser detectados. Para la configuración de las cámaras existen dos opciones:

- Dinámica: Esta opción consiste en actualizar los objetos de interés de la cámara sin ayuda de un software externo. De esta forma se ofrece una mayor facilidad a la hora de actuar con distintos demostradores. No obstante, esta opción se decidió en un principio no contemplarla y dejarla para proyectos futuros.
- Estática: Es la opción escogida para nuestro Proyecto y consiste en elegir los objetos de interés con el software de configuración de cada cámara. Una vez realizado esto, la cámara ya disponía de los colores que debían ser seguidos, incluso cuando la cámara se encontraba apagada.

También cabe decir que para la plataforma Linux ev3-dev existen drivers de soporte básico que solamente permiten el seguimiento de un objeto en ambas cámaras.

A continuación, se explicarán las funciones de cada una, ya que ambas presentan algunas diferencias que deben ser destacadas.

3.1 Funcionamiento NXT Cam v4

Esta cámara es capaz de detectar hasta 8 objetos dentro de su campo de visión una vez realizada la configuración con NXT Cam View.

La cámara tiene distintos modos de funcionamiento, que permiten acceder a distinta información sobre los objetos detectados. Por ello, existen comandos predefinidos en el hardware de la cámara que sirven para conseguir este fin.

La comunicación con la cámara se realiza a través del protocolo I2C [1]. Para facilitar esta comunicación, la cámara proporciona una serie de registros que permiten enviar o recibir información. La tabla 1 resume los comandos más habituales para el manejo de la cámara. En concreto para esta cámara, disponemos de un registro denominado como “registro de comandos" (número 0x42). Según la documentación, cualquier comando escrito en este registro, será ejecutado.

Tal y como se ilustra en la tabla 1, para activar el seguimiento de la cámara basta con escribir en el registro de comandos el comando “enable tracking” (0x45). Para hacer lo contrario, es decir, finalizar el seguimiento, escribir el comando “disable tracking” (0x44). También existen funciones para indicar cuántos objetos están siendo detectados

(leer el registro 0x42). En este caso, será muy importante haber activado previamente el seguimiento de objetos, por lo contrario, detectará cero objetos.

En la introducción también se habló de que la cámara es capaz de informar sobre las coordenadas de cada objeto relativamente a su campo de visión.

Otra función característica de la NXTCam es que tiene la capacidad de priorizar los objetos detectados según el tamaño o el color. Estas funciones corresponden con los comandos "sort tracked objects by size" y "sort tracked objects by color". Existe otra posibilidad que nos permite no asignar ninguna prioridad, correspondiendo esta misma con la función de "do not sort tracked objects".

Finalmente, este dispositivo es capaz de configurarse bajo dos opciones. La primera, la capacidad de realizar el seguimiento de objetos, componiendo estos mismos como una única figura cuadrilátera. Esto se consigue con el comando "select object tracking mode". Sin embargo, el otro modo permite la composición de objetos en una línea formada también por figuras cuadriláteras. Esta función es muy útil para realizar el seguimiento de objetos de una forma más exacta, cuando la posición de éstos, se corresponda con una dirección en diagonal con respecto al campo de visión de la cámara.

Para una comprensión más sencilla, se adjunta la siguiente tabla que corresponde con las funciones comentadas:

Función	**Acción**
Activar Seguimiento	Escribir 0x45
Desactivar Seguimiento	Escribir 0x44
Objetos Detectados	Leer registro 0x42
Coordenadas Objeto	Leer registros correspondientes al objeto (ANEXO II)
Ordenar Objetos Por Tamaño	Escribir 0x41
Ordenar Objetos Por Color	Escribir 0x55
No Ordenar Objetos	Escribir 0x58
Seguimiento Por Objeto	Escribir 0x42
Seguimiento Lineal	Escribir 0x4C

Tabla 1: Funciones principales de NXTCam

Para conocer en profundidad los comandos o características recién explicados, consultar el Anexo II.

3.2 Funcionamiento Pixy Cam

Pixy utiliza un algoritmo basado en el filtrado de color para detectar objetos. Es capaz de recordar hasta 7 firmas (descripción almacenada en la cámara de un color) de colores distintas, lo que implica poder realizar un seguimiento de hasta 7 colores de objetos (siempre que sean diferentes). La capacidad de realizar el seguimiento de múltiples objetos al mismo tiempo, es una muestra de la potencia de cómputo que tiene esta cámara a pesar de su tamaño.

Una facilidad que ofrece Pixy, es un botón que permite guardar una firma de color sin necesidad de utilizar el software Pixy Mon. Esto brinda una ventaja al usuario, haciendo más fácil su configuración.

Finalmente, otra característica muy útil es lo que se denomina como "color-code". Consiste en guardar dos colores para realizar un seguimiento de los objetos que se compongan de dichos colores almacenados.

La siguiente figura consta de un ejemplo donde se utiliza esta función:

Figura 9: Código de color como estación de carga [11]

En este caso, tenemos el color rosa y verde, de esta manera configuramos el "color-code" número 1. Así nuestro robot, será capaz de reconocer que donde exista esta combinación de colores, significará que es una estación de carga.

PixyCam es capaz de configurarse bajo varios protocolos de comunicación. Éstos pueden ser UART, SPI e I2C. En nuestro caso solamente utilizaremos el I2C ya que es el protocolo que decidió utilizarse desde un principio.

La cámara recoge en una lista cada 20 ms, los objetos que ha detectado. En este caso, el I2C se basa en una política "polling" o encuesta, la cual consiste en preguntar a la cámara cada cierto tiempo sobre los objetos detectados. Dichos objetos se representan por lo que se denomina como "Bloque de Objeto" cuya estructura se muestra en la figura siguiente:

Data label	Size	Description
sync	2 bytes	Synchronization tag, indicates start of object block
checksum	2 bytes	Simple checksum for the rest of the object block data
signature	2 bytes	Color signature of object (1-7) or list of signatures for color code
x_position	2 bytes	X position of object center in image
y_position	2 bytes	Y position of object center in image
size	2 bytes	Size (area) of object in image
angle	2 bytes	Angle of object (optional, only available for color codes)

Figura 10: Estructura del Bloque de Objeto [12]

Pixy Cam también utiliza una variante del protocolo I2C denominada I2C LEGO. El funcionamiento consiste en que cuando la cámara es accedida a través de I2C, el primer byte enviado se denomina como "Query Address". Además, para cierto tipo de consultas, existe la posibilidad de enviar bytes adicionales para especificar la petición. Estos bytes adicionales se conocen como "Optional Query Address". Cualquier dato enviado a través del I2C se codificará como "Little Endian", lo que significa que el byte menos significativo se envía primero. Para más información sobre este protocolo, es recomendable consultar el Anexo IV.

3.3 Comparativa

Como parte de nuestro proyecto, uno de los objetivos era realizar el análisis de cada cámara para después ser capaz de ofrecer una comparativa que mostrara una comparación clara de ambas.

Para una explicación más sencilla y entendible, se adjunta la siguiente tabla:

Característica	**NXT CAM**	**PIXY CAM**
FPS	30	50
Campo de visión	88 x 144 *pixeles*	650 x 400 pixeles
Nº Firmas de color	8	7
Orden de objetos	Tamaño y Color	Ninguno
Modos de seguimiento	Objetos y Lineal	Objetos
Código de Color	NO	SI

Tabla 2: Comparación de características entre las cámaras

Por un lado, NXT CAM presenta funciones útiles como ordenar los objetos detectados o realizar el seguimiento con el modo objeto, o lineal (ya explicados en el apartado 3.1). Aunque estas características no existan en la Pixy, ésta es mucho más potente además de ofrecer una mayor resolución.

Tener mayor resolución te permite obtener de una manera más exacta las coordenadas de cada objeto al disponer de una mayor distribución de pixeles.

Para realizar la comparativa de ambas cámaras, como primera fuente de información se utilizaron la documentación de cada cámara. No obstante, para afianzar los resultados se optó por realizar pequeñas pruebas en cada una de las cámaras para comprobar su eficiencia y utilidad. Dichas pruebas consistieron en realizar operaciones como establecer la comunicación I2C con la cámara, o recoger las coordenadas de los objetos detectados por la misma. El resultado de las mismas concluyó en que la cámara con mejor rendimiento fue la PixyCam ya que esta reconocía los objetos de una forma más exacta.

4.- Desarrollo de la librería

4.1 Entorno de Desarrollo

Cuando se utilizan diferentes plataformas para desarrollar un proyecto, como en este caso una librería funcional, es necesario utilizar lo que se denomina como compilador cruzado. Esto consiste en un compilador capaz de crear código ejecutable para otras plataformas.

Para conocer los componentes del mismo, se muestra la siguiente figura:

Figura 11: Estructura del entorno para la librería versión Linux [11]

Como se observa en la figura 12, un compilador cruzado está compuesto por 3 elementos principales. Un host donde desarrollaremos y compilaremos el código que queremos ejecutar en la otra plataforma, un medio de comunicación con la plataforma donde queremos utilizar el ejecutable y por último la propia plataforma que, en este caso, corresponde con un sistema empotrado.

Es muy importante utilizar sistemas de desarrollo cruzado en proyectos de este tipo porque brinda al usuario una gran facilidad a la hora de desarrollar las aplicaciones. Además, una vez configurado correctamente, la compilación y ejecución de las aplicaciones creadas pueden realizarse automáticamente.

En nuestro proyecto como se ha comentado anteriormente, disponemos de dos plataformas. Por tanto, tendremos dos entornos de desarrollo.

Para la programación con el LEGO ev3-dev, se tomó la decisión de hacerlo bajo el compilador cruzado de ARM. Esto nos permitía la compilación de nuestra librería trabajando así de una manera más cómoda. Además, al utilizar el entorno de programación eclipse, es posible configurar la comunicación con el dispositivo a través de ssh y ejecutar las aplicaciones de manera remota.

Por otro lado, para el desarrollo de la librería en MaRTE OS se necesitará hacer uso de otro tipo de compilador.

4.2 Diseño General

En este apartado, nos centraremos en explicar el diseño de la librería. Además, se explicará el entorno de programación utilizado para desarrollar la misma.

El diseño de la librería se constituye por varios niveles. El nivel más bajo (correspondiente con el nivel "Driver" de la figura 12) de la librería se ocupa de la interacción con las cámaras a través del manejo del protocolo de comunicaciones I2C. Por otro lado, el nivel superior (referenciado en la figura 12 como "Librería de usuario") presenta un diseño diferente para cada una de las cámaras. Dado que el adaptador Pixy Mindsensors permite controlar la PixyCam de forma similar a la NXTCam, (esto es, a través de un conjunto de registros), la complejidad de la librería a nivel de usuario se reduce sensiblemente. Finalmente, como existen diferencias en el manejo de drivers de los dispositivos entre Linux y MaRTE OS, resulta indispensable realizar el manejo de la comunicación I2C de diferente forma para cada entorno. Esto se ilustra en la figura 12.

En la siguiente figura se muestra la estructura deseada de la librería:

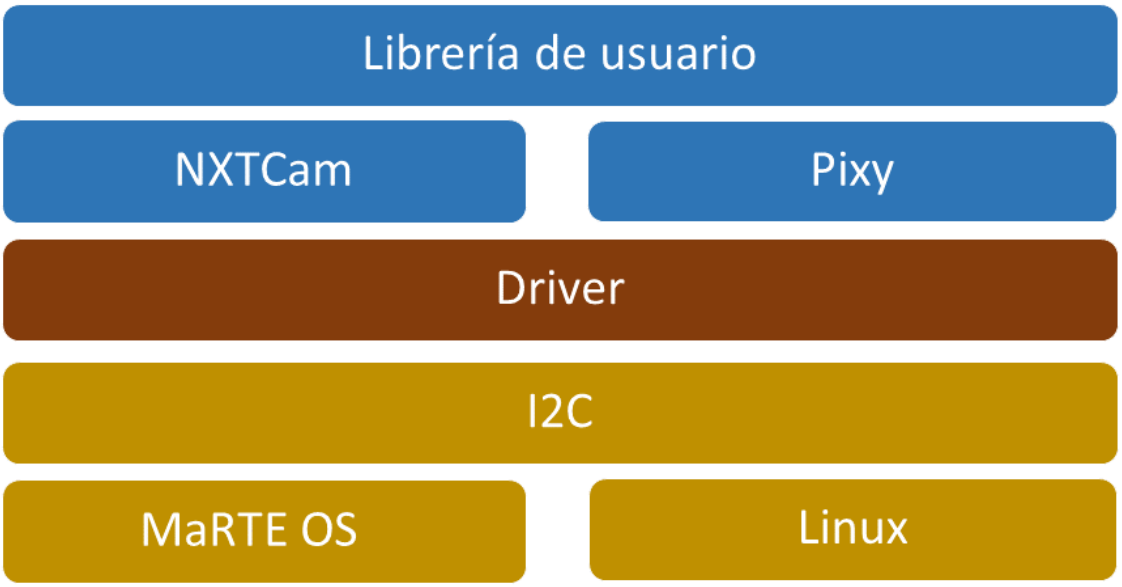


Figura 12: Estructura de la librería

4.3 Descripción del nivel de Usuario

En los siguientes apartados, se explicará el desarrollo llevado a cabo para implementar el uso de las cámaras NXTCam-v4 y Pixy CMUcam5 para los entornos Linux y MaRTE OS.

Como se explicó en la sección anterior, se decidió unificar la librería de usuario para ambas cámaras en una sola y dejar la comunicación I2C para los dos niveles inferiores, diferentes debido al entorno donde se localizan. A continuación, explicaremos en la tabla 3 cada una de las funciones de usuario implementadas y después nos centraremos en cada apartado explicando el modo de comunicación I2C que utiliza.

Función	Descripción

camera_init	Inicializa la comunicación I2C con la cámara
camera_change_address	Cambia la dirección I2C de la cámara
camera_objects_detected	Indica el número de objetos detectados
camera_software_version	Indica la versión software de la cámara
camera_vendor_id	Indica el vendor ID de la cámara
camera_device_id	Indica el device ID de la cámara
camera_start_tracking	Ordena a la cámara activar el seguimiento
camera_stop_tracking	Ordena a la cámara desactivar el seguimiento
camera_reset	Reinicia la cámara
camera_objects_coordinates	Obtiene las coordenadas del objeto deseado
camera_sort_objects	Hace que la cámara ordene los objetos detectados según el modo especificado

Tabla 3: Interfaz de usuario de la librería

Las funciones mayor relevancia se comentan a continuación:

- `camera_init`: Es la encargada de establecer la comunicación con la cámara a través de I2C. Cuando se realiza la llamada a esta función es necesario indicar la dirección de la cámara y el puerto en el que está conectada. Esta función utiliza exclusivamente la función *i2c_init* la cual pertenece a la capa de la librería de comunicaciones I2C.
- `camera_start_tracking`: Esta función se utiliza para indicar a la cámara que comience con el seguimiento de objetos. Para ello, se encarga de escribir el comando "start tracking" en el registro de comandos, llamando directamente a la función *i2c_write* que, al igual que *i2c_init*, pertenece a la capa de la librería de comunicaciones I2C.
- `camera_objects_coordinates`: Se encarga de recoger las coordenadas del objeto especificado haciendo uso de la llamada *i2c_read* que también pertenece a la última capa de la librería encargada de las comunicaciones I2C. Las coordenadas se guardan en una estructura definida con 5 campos, uno para el color del objeto en cuestión, y otros cuatro para las coordenadas del objeto. Además, es necesario indicar el objeto del cual se quiere extraer esta información. Este aspecto se considera una mejora de los drivers de Linux ya que estos solo permiten el seguimiento de un objeto. Sin embargo, con esta mejora es posible realizar el seguimiento de hasta 8 objetos.
- camera_info: Existen 3 funciones que indican información de la cámara. Estas son:
  - `camera_software_version`: Recoge la versión software de la cámara
  - `camera_vendor_id`: Recoge el vendor id
  - `camera_device_id`: Recoge el device id (fabricante)
- `camera_change_address`: Esta función se utiliza para cambiar la dirección que permite manejar la cámara como dispositivo I2C.
- `camera_sort_objects`: Cuando la cámara tiene el seguimiento activado, ésta por defecto ordenará los objetos detectados según el tamaño, siendo el primero el más grande, hasta el último que coincide con el más

pequeño. No obstante, esta función nos permite indicar tres modos de ordenación de los objetos. El primero sería el recién explicado, que ordena los objetos según el tamaño. El segundo nos permite ordenar los objetos según el color de cada uno. Finalmente tenemos también el modo de no ordenar los objetos.

La comunicación con el hardware se realiza a través del protocolo I2C. Este protocolo se implementa de diferente forma según el sistema operativo utilizado.

4.4 Descripción del nivel de Driver

El nivel que corresponde con el driver estará compuesto por tres funciones que serán utilizadas para las funciones que pertenecen al nivel de usuario. A continuación explicamos cada una con detalle:

La función `i2c_read` se utiliza para leer directamente los registros de la cámara una vez esté establecida la comunicación. Esta función recibe el registro que se quiere leer, un "buffer" donde se depositará la lectura y finalmente el número de bytes que se quieren leer (cada registro será 1 byte).

Para realizar la acción de lectura de los registros de la cámara, utilizaremos la función `i2c_write`. Los parámetros de esta función son el registro sobre el que se quiere escribir y el valor que se desea escribir en dicho registro.

Para inicializar la comunicación con la cámara se utilizará la función `i2c_init`. Como parámetros utiliza la dirección I2C de la cámara y el puerto al que está conectada.

4.4.1 Desarrollo para Linux

En este apartado se mostrará cómo se ha implementado cada función de I2C para la plataforma Linux de una forma sencilla, explicando brevemente el uso de otras funciones proporcionadas por la librería i2c-dev.h

Un aspecto importante a destacar del sistema operativo de Linux es el sistema de drivers de POSIX. POSIX es un estándar definido por el IEEE (Instituto de Ingeniería Eléctrica y Electrónica) que proporciona una interfaz de sistema y entorno al sistema operativo. Una propiedad fundamental son las llamadas al sistema. Éstas están encargadas de realizar operaciones que necesiten una autorización del sistema operativo. En nuestro caso, tendremos que hacer uso de estas llamadas al sistema para poder implementar nuestra capa de comunicaciones I2C. En concreto solamente utilizaremos dos de ellas:

- `open(const char *pathname, int flags)`: Nos permite abrir un fichero indicándole la ruta al mismo. Devuelve un identificador denominado como descriptor de fichero, el cual es utilizado para trabajar con dicho fichero.
- `ioctl(int fd, unsigned long request)`: Esta llamada al sistema permite controlar o comunicarse con un driver de dispositivo. Utiliza el descriptor de fichero para saber sobre qué fichero se está estableciendo la comunicación.

Por otro lado, necesitamos una manera de comunicarnos con los registros de la cámara tanto para leer como para escribir. La librería i2c-dev.h es una librería ya incluida en el "kernel" de Linux la cual permite realizar estas operaciones de una manera más cómoda para el usuario. Con el uso de la misma, desarrollaremos nuestras propias funciones de comunicación I2C. Las principales funciones que utilizaremos de esa librería serán:

- `i2c_smbus_read_byte_data(int file, int reg)`
- `i2c_smbus_write_byte_data(int file, int reg, int value)`

Dichas funciones serán utilizadas para crear las funciones *i2c_read* e *i2c_write.* Por otro lado, la función de *i2c_init*, que es la que inicia la comunicación I2C con la cámara, hará uso de la llamada al sistema **"open"**. La idea al utilizar esta función es tener un fichero que sirva de puente de comunicaciones entre la cámara y el Lego Mindstorms. Una vez hecha la llamada a **"open"** utilizaremos la llamada al sistema **"ioctl"** que será la encargada de concretar la comunicación I2C como se ha explicado anteriormente.

4.4.2 Desarrollo para MaRTE OS

Para este apartado, al igual que el anterior, se hará un resumen de como se ha implementado la librería para el sistema operativo MaRTE OS. Existirán pequeñas diferencias con respecto a la versión de Linux que deben ser destacadas para comprender el diseño de la librería.

En primer lugar, utilizaremos una librería de apoyo distinta. En este caso será la librería bcm2835 la cual ofrece funciones de lectura y escritura a través de I2C. Las funciones *i2c_read* y *i2c_write* utilizarán respectivamente las siguientes funciones:

- `bcm2835_i2c_read_register_rs(char* regaddr, char* buf, uint32_t len);`
- `bcm2835_i2c_write(const char * buf, uint32_t len);`

Por otro lado, para la función *i2c_init* existen varios aspectos que son necesarios comentar:

El primero, es que fue necesario adaptar la velocidad de comunicación I2C con el fin de que las cámaras tuvieran el tiempo necesario para actuar. Ésto lo conseguimos llamando a la función **bcm2835_i2c_setClockDivider**. El motivo de uso de dicha función se debe a que se encontró un "bug" de la Raspberry que se conoce como "clock stretching" [14] el cual hacía que el reloj de la misma no se mantuviera en sincronía con el de las cámaras.

Lo segundo, es que MaRTE OS es un sistema operativo que carece de sistema de ficheros, así que no fue posible utilizar la función "open" como se hacía en la versión de Linux. La solución a ello es el uso de la función **bcm2835_setSlaveAddress** a la cual indicamos la dirección de la cámara.

También como librería para controlar el Pistorms, se utilizó una ya existente la cual nos permite configurar varios aspectos del Adaptador Pistorms. En esta librería se implementará el módulo que permite el uso de las cámaras.

La organización de la librería con la implementación del nuevo módulo para las cámaras se presenta en la figura 13:

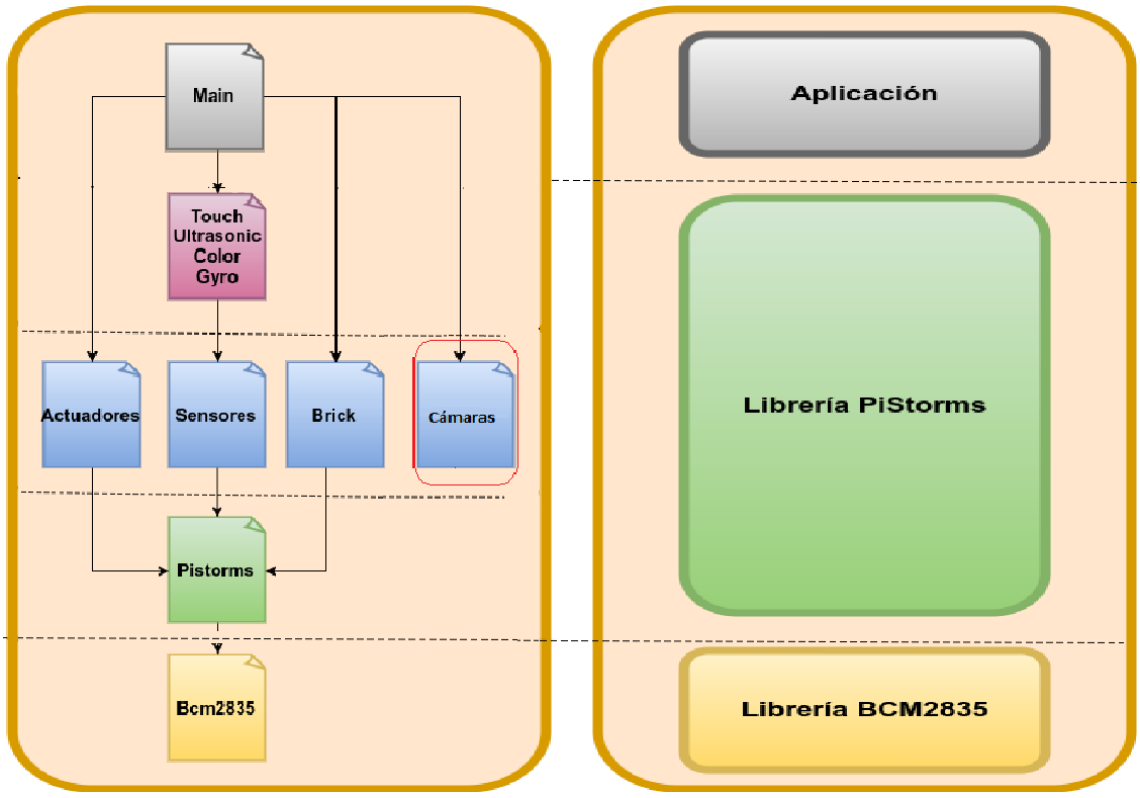


Figura 13: Diseño librería Pistorms actualizada [13]

Finalmente, también se ha de comentar que, puesto que utilizábamos el Adaptador Pistorms como puente de comunicaciones I2C, era necesario indicarle en que puerto teníamos conectada la cámara. Gracias a la librería de Pistorms para Raspberry Pi se pudo utilizar la función **pistorms_port_set_type_sensor** que permitía establecer el puerto indicado en modo I2C, y como consecuencia, permitir la comunicación con la cámara a través de dicho puerto.

5.- Pruebas

En este apartado se desarrollarán las diferentes pruebas realizadas desde el comienzo hasta el final del proyecto. Las pruebas que se han realizado eran según las distintas versiones de nuestra librería. Cada versión integraba funcionalidad adicional. A continuación, se explicará para cada plataforma las pruebas realizadas. En ambos casos se siguió la siguiente estructura de pruebas:

- Comprobar que la comunicación con las cámaras era correcta.
- Asegurarse que las funciones de las cámaras proporcionaban resultados esperados.

5.1 Pruebas Sobre la Plataforma Lego EV3:

Para comprobar la ejecución de programas en lenguaje C, se creó un programa que pintaba simplemente un mensaje por pantalla.

Una de las pruebas sobre la comunicación con las cámaras era tratar de leer la versión software o el "vendor id" de cada una. De esta manera afianzábamos que la comunicación con las cámaras a través de I2C era la correcta. Para realizar esta prueba, se escribió un pequeño código en el que establecíamos la dirección de la cámara en cuestión como dispositivo I2C y así ser capaces de leer información de la misma. El siguiente paso consistía en llamar a una de las funciones de lectura que nos proporcionaba la librería i2c-dev.h. Al utilizar eclipse, ejecutábamos el programa en el EV3 de forma remota. Viendo que la lectura era la esperada, se dio por concluida la parte de comunicación I2C.

También era necesario asegurarse que las funciones como informar sobre el número de objetos detectados se hacía de forma correcta. El ejemplo consistía en que, una vez establecida la comunicación con la cámara, debíamos escribir en el registro de comandos, el comando que activa el seguimiento. Acto seguido, leer el registro que recoge el número de objetos detectados en cada momento. Al ejecutar el programa, se puso la cámara enfocando a una superficie blanca en la que se situaba una bola de color rojo. Como era de esperar, la ejecución del programa indicaba que los objetos detectados era uno.

Para completar las pruebas sobre las cámaras, también fue importante recoger las coordenadas de cada objeto. Como se observó que el resultado fue también el esperado, se dio por concluida la comprobación de las cámaras, por lo que se comenzó a escribir la librería para Linux.

5.2 Pruebas Raspberry Pi y Adaptador Pistorms

Por otro lado, para este entorno se realizaron las mismas pruebas que en el apartado anterior. No obstante, el hecho de tener un dispositivo que haga de puente de comunicaciones I2C como es el adaptador Pistorms, implicaba realizar el procedimiento de comunicación de forma diferente.

Para ello, se intentó leer directamente el “vendor id” del adaptador. Como la prueba fue exitosa, la primera parte de la comunicación se dio por concluida.

Para afianzar que la parte de comunicaciones se comprobó que todos los comandos I2C fueran reenviados a la cámara a través del adaptador de Pistorms. Así que el procedimiento consistía en establecer el modo I2C al puerto conectado a la cámara, además de llamar a una función que establecía la dirección de la cámara como dispositivo esclavo. El resultado fue que después de realizar estos dos pasos, la comunicación I2C con la cámara era la esperada.

Una vez comprobado esto anterior, se volvieron a realizar las mismas pruebas de funcionamiento de la cámara que en el apartado anterior.

6.- Demostrador

6.1 Descripción General

Para considerar el trabajo realizado como una librería funcional y fiable se diseñó y programó un demostrador con el que se ofreciese una idea del funcionamiento de las cámaras.

Dicho demostrador está diseñado para que el sistema en completo sea capaz de recoger las bolas de color lanzadas por un usuario. La cámara notificará en cada momento la posición del objeto de interés, cuyo color ha sido configurado previamente en la cámara. Una vez la bola entre en el campo de visión de la cámara, ésta deberá recoger las coordenadas de la bola para que el motor de lego mueva el centro de la portería a la posición indicada por la cámara.

Para el demostrador se utilizarán la Pixy Cam y un motor de Lego. Dichos elementos se conectarán utilizando el adaptador Pistorms. Es decir, tanto las cámaras como el motor de Lego utilizado se conectarán al adaptador con un cable RJ12.

6.2 Estructura del Demostrador

La estructura del demostrador está únicamente compuesta por piezas del kit de LEGO, las cámaras y la Raspbery Pi con su Adaptador Pistorms. Estará compuesta por dos plataformas la primera es una cinta trasportadora en la cual encajaremos una portería que irá moviéndose conforme lo haga la cinta. Después tenemos una grúa que es la encargada de sostener la cámara obteniendo así una vista de planta del campo de juego. Es importante destacar que según la posición de la cámara solo nos preocuparemos de recoger la coordenada Y. El rango de pixeles de la coordenada Y, es de 0 a 100. En cambio, el rango de posición de la cinta es de 0 a 557. Por tanto, para encajar la portería con la posición obtenida de la cámara, fue necesario establecer cuántos pixeles de la cinta correspondía con cada pixel de la cámara. En este caso fue sencillo, ya que el rango de pixeles de la cámara es de 0 a 100, así que hacienda una regla de tres, obteníamos que cada pixel de la cámara corresponde con 5,57 posiciones de la cinta.

La cinta trasportadora está conectada a un motor LEGO, el cual utilizaremos de actuador para que la portería sea móvil.

Por otro lado, la grúa se creó desde cero con piezas LEGO. Previamente se estimó cual era la altura más adecuada para que la cámara detectase objetos a tiempo.

A continuación, en la figura 12 se muestra la organización del demostrador:

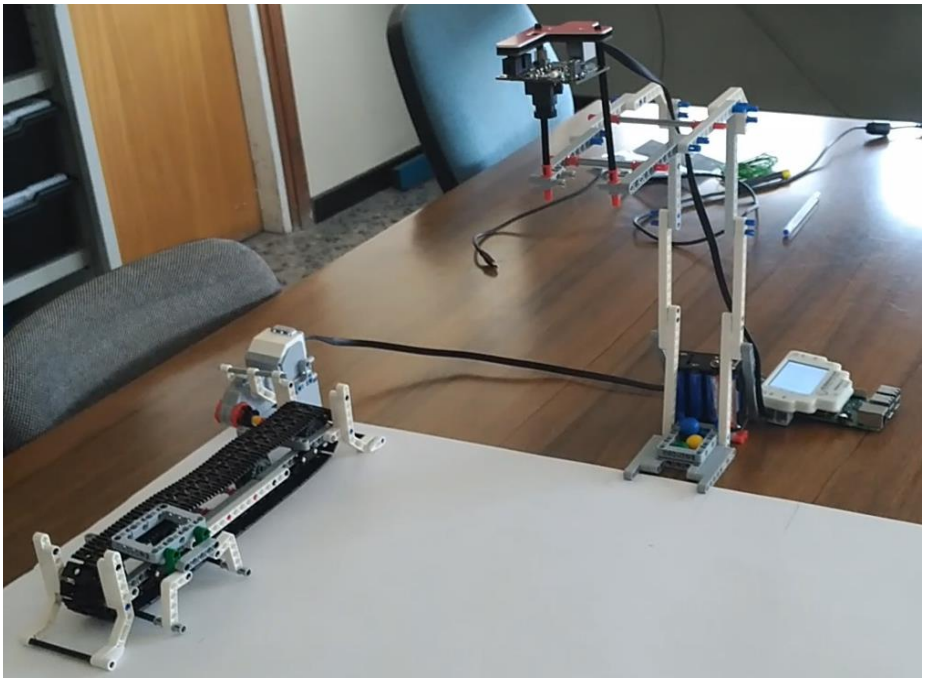


Figura 14: Demostrador utilizando piezas y motor de Lego

6.3 Descripción de la aplicación

Para explicar el funcionamiento explicado en el primer apartado de este capítulo, se implementará un pseudocódigo que ayudará a entender la aplicación.

```
//Iniciamos comunicación

1: Iniciar bcm y bus I2C
2: Reseteamos todos los parámetros del motor
3: Establecer la velocidad deseada para el motor.

//Iniciamos la cámara como dispositivo I2C.

4: Indicamos la dirección de la cámara y el puerto del Adaptador Pistorms.
5: Ordenamos a la cámara que comience con el seguimiento.
6: En un bucle infinito hacemos:
        7: Indicamos a la cámara como dispositivo activo.
        8: Realizamos 5 medidas de las coordenadas si detectamos algún objeto
        9: Si hemos recogido una o varias medidas con coordenadas > 0
                10: Realizamos la media según el número de medidas recogidas.
                11: Traducimos las coordenadas obtenidas por la cámara a la cinta.
                12: Movemos el motor.
        13: En caso de no haber cogido ninguna medida, no comunicamos ninguna
        posición
        14: Esperamos 40 milisegundos para volver a enviar otra medida
```

Figura 15: Pseudocódigo del demostrador

En el punto 4 del pseudocódigo anterior es necesario llamar a la función camera_init para establecer la comunicación con la cámara.

Cuando accedemos al motor perdemos la comunicación con la cámara, por ello en el punto 7 del pseudocódigo del demostrador es necesario volver a activar la comunicación con la cámara.

Por otro lado, es necesario realizar las 5 medidas con la cámara ya que el seguimiento de la misma indica posiciones con un cierto rango de error. Aunque el objeto estuviera quiero, por condiciones de luz, la cámara podría detectar posiciones inexactas con un margen de error de 2 unidades. Realizando 5 medidas para después recoger la media nos acercamos a la posición real del objeto.

7.- Conclusiones y Líneas Futuras

7.1 Conclusiones

En resumen, este trabajo se ha involucrado en desarrollar e implementar una librería para el manejo de cámaras de procesamiento de imágenes integrado para sistemas Linux, además de ampliar la funcionalidad de otra librería ya existente para Raspberry Pi.

La librería desarrollada para Linux presenta una importante diferencia respecto a la ya existente para Linux, la cual solo permitía el seguimiento de un objeto. Como mejora se implementó la posibilidad de realizar el seguimiento de hasta 8 objetos.

En este trabajo también se han utilizado dos cámaras, PixyCam y NXTCam. Por ello parte de nuestro trabajo consistió en desarrollar una interfaz genérica de ambas cámaras.

Además de esto, el diseño de la librería se realizó con el fin de guardar una portabilidad entre las plataformas Linux y MaRTE OS. No obstante, el modo de comunicación perteneciente al nivel inferior de la librería se realizará de manera diferente en cada plataforma.

Personalmente, realizar este trabajo me ha servido para aprender de primera mano muchas de las características que ofrecen los sistemas empotrados. Además, me ha ayudado a desarrollar mi capacidad para enfrentarme a problemas y solucionarlos de una manera metódica y eficiente.

7.2 Líneas Futuras

Utilizar las cámaras directamente en la Raspberry Pi, sin tener que hacer uso del adaptador Pistorms, lo cual requeriría un trabajo a bajo nivel (por ejemplo, adaptando el cableado físico entre la Raspberry Pi y las cámaras).

Por otro lado, otra opción sería extender la librería de este trabajo para integrar nuevos buses de comunicaciones, ya que algunas cámaras utilizan otros protocolos (como la Pixy Cam y su comunicación mediante USB).

Finalmente, sería posible implementar la configuración dinámica para ambas cámaras, añadiendo la funcionalidad en la misma librería.

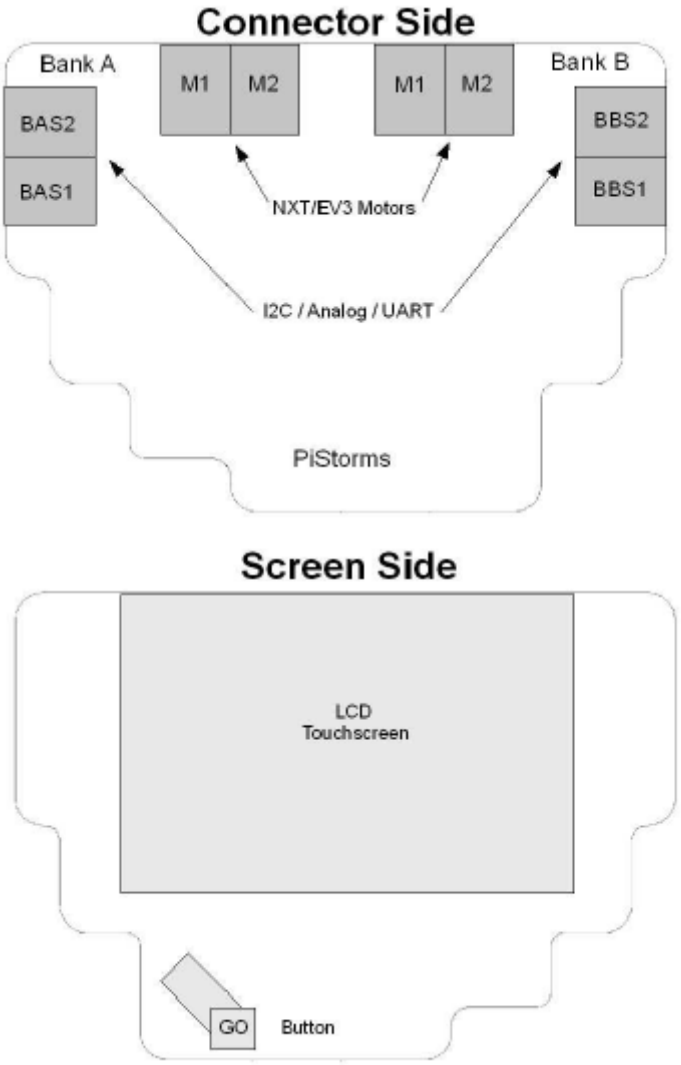
8.- Bibliografía

[1]: Introducción al I2C bus, 2012
http://www.diarioelectronicohoy.com/blog/introduccion-al-i2c-bus
[2]: NXT Cam v4, 2017
http://www.mindsensors.com/ev3-and-nxt/14-vision-subsystem-camera-for-nxt-or-ev3-nxtcam-v4
[3]: Pixy Cam, 2017
http://charmedlabs.com/default/pixy-cmucam5/
[4]: LEGO Group, Sistema operativo ev3dev
http://www.ev3dev.org/
[5]: Universidad de Cantabria, Sitio web Sistema Operativo MaRTE OS, 2000-2017
http://marte.unican.es/
[6]: Kinect para Xbox 360,
https://es.wikipedia.org/wiki/Kinect
[7]: Lego Group, Mindstorms, ev3
https://www.lego.com/es-es/mindstorms/products/mindstorms-ev3-31313
[8]: Raspberry Pi, Web site
https://www.raspberrypi.org/
[9]: Campeonato de Robótica, Robotica educative
http://www.diariovasco.com/gipuzkoa/robotica-crea-cantera-20170924004529-ntvo.html
[10]: Libreria Carlos
[11]: Imagen “Color Code” PixyCam http://cmucam.org/projects/cmucam5
[12]: Imagen estructura de bloque de objeto
https://www.kickstarter.com/projects/254449872/pixy-cmucam5-a-fast-easy-to-use-vision-sensor?lang=es
[13]: Imagen extraída del proyecto que desarrolló la librería para Pistorms
[14]: Bug Clock Stretching Raspberry Pi
http://www.advamation.com/knowhow/raspberrypi/rpi-i2c-bug.html
[15]: Imagen de estructura del protocolo I2C
http://dignal.com/wp-content/uploads/2015/03/i2c-diagram.png
[16]: Imagen de Raspberry Pi 1 Modelo B+
https://electronilab.co/wp-content/uploads/2014/07/Raspberry-Pi-Modelo-B+_Electronilab_01.jpg
[17]: Imagen Kit Lego Mindstorms
http://k12lab.com/sites/default/files/styles/original_as_jpg/public/topics/featured_image/8358291640_5cfe5bc561_h%20-%201200x675.png?itok=uHYc1Qd-

9 ANEXOS

ANEXO I:

Estructura del adaptador Pistorms



ANEXO II:

Tabla de comandos para NXTCam y Adaptador Mindstoms para PixyCam

Commands		Action
ASCII	Hex	
A	0x41	Sort tracked objects by size
B	0x42	Select Object tracking mode
C	0x43	Write to image sensor Registers Use CAUTION when using command C since this can stop NXTCam from working properly. In case this happens, please power off your NXTCam and power it on again.
D	0x44	Disable Tracking
E	0x45	Enable Tracking
G	0x47	Get the Color map from NXTCam Engine
H	0x48	Read data from the image sensor Registers
I	0x49	Illumination on (Future)
L	0x4C	Select Line tracking mode
N	0x4E	Set ADPA mode ON (setting stored in NVRAM)
O	0x4F	Set ADPA mode Off (default) (setting stored in NVRAM)
P	0x50	Ping NXTCam Engine
R	0x52	Reset NXTCam Engine
S	0x53	Send the color map to NXTCam Engine
T	0x54	Illumination Off
U	0x55	Sort tracked objects by color
V	0x56	Get NXTCam Engine firmware version No, Read resulting string at 0x42 (12 bytes).
X	0x58	Do not Sort tracked objects
J	0x4A	Lock tracking buffer Tracking is going on continuously, and while you read
		buffer may be updated by the tracking engine. You can issue this command to Lock the buffer updates. After issuing this command allow 25 milliseconds for any updates in progress to finish. Then read the tracking information.
K	0x4B	Unlock tracking buffer. If you had locked the buffer before reading, ensure to unlock it after you are done reading.

Tabla de registros de NXTCam y Adaptador Mindstoms para PixyCam

Register	Read	Write	Comments
0x00-0x07	Software version – *(Vn.nn)*	-	
0x08-0x0f	Vendor Id – *mndsnsrs*	-	
0x10-0x17	Device ID – *NXTCam*	-	
0x41	-	Command	This register is command register. A command written here will be executed.
0x42	Number of objects detected	-	Shows how many objects are being tracked. Zero indicates that there are no objects being tracked.
0x43	1st object color	-	This is the first object color as per the sorting method selected.
0x44[1]	1st object - X upper left		Upper left X coordinate of first object
0x45	1st object - Y upper left		Upper left Y coordinate of first object
0x46	1st object - X lower right		Lower right X coordinate of first object
0x47[2]	1st object - Y lower right		Lower right Y coordinate of first object
0x48	2nd object color		
0x49-0x4C	2nd object co-ordinates		
0x4D	3rd object color		
0x4E-0x51	3rd object co-ordinates		
0x52	4th object color		
0x53-0x56	4th object co-ordinates		
0x57	5th object color		

ANEXO III:

Documentación generada por la herramienta Doxygen

marte_pistorms_camera_i2c.h File Reference

Go to the source code of this file.

Functions

int i2c_read (int file, char reg, char *data, int bytes)

int i2c_write (int file, char reg, int value)

int i2c_init (int address, int portNumber)

Function Documentation

◆ i2c_init()

```
int i2c_init ( int  address,
               int  portNumber
             )
```

◆ i2c_read()

```
int i2c_read ( int    file,
               char   reg,
               char * data,
               int    bytes
             )
```

◆ i2c_write()

```
int i2c_write ( int   file,
                char  reg,
                int   value
              )
```

Generated by doxygen 1.8.13

marte_pistorms_camera.h File Reference

Go to the source code of this file.

Data Structures

struct object

Macros

#define COMMAND_REGISTER 65
#define OBJECTS_DETECTED_REGISTER 66
#define SOFTWARE_VERSION_REGISTER_BASE 0
#define SOFTWARE_VERSION_REGISTER_TOP 7
#define VENDOR_ID_REGISTER_BASE 8
#define VENDOR_ID_REGISTER_TOP 15
#define DEVICE_ID_REGISTER_BASE 16
#define DEVICE_ID_REGISTER_TOP 23
#define OBJECT_BASE 67
#define STOP_TRACK_COMMAND 68
#define START_TRACK_COMMAND 69
#define RESET_COMMAND 82
#define SORT_SIZE_COMMAND 65
#define SORT_COLOR_COMMAND 85
#define NO_SORT_COMMAND 88

Typedefs

typedef struct object object_properties_t

Functions

int pistorms_sensor_camera_configure (int connector_id)
int camera_init (int address, int port)
void camera_set_as_active_device ()
void camera_change_address (int newAddress)
int camera_objects_detected ()
void camera_software_version (char *buffer)
void camera_vendor_id (char *buffer)
void camera_device_id (char *buffer)
int camera_start_tracking ()
int camera_stop_tracking ()
int camera_reset ()
int camera_object_coordinates (int object, object_properties_t *data)

ANEXO IV:

Protocolo I2C LEGO para PixyCam

General Query

This is query makes up what is known to as "General Mode" in the Pixy LEGO documentation.

Query Address	Optional Query Data	Pixy Response
0x50	none	6 bytes that describe the largest detected block within all signatures, including color coded blocks. If no object is detected, all bytes will be 0.

Byte	Description
0, 1	16-bit value that describes the signature of the largest block. A value of 1 thru 7 corresponds to signatures 1 thru 7. A value greater than 7 is a color code encoded in octal (base-8).
2	X value of center of largest detected block, ranging between 0 and 255. An x value of 255 is the far right-side of Pixy's image.
3	Y value of center of largest detected block, ranging between 0 and 199. A value of 199 is the far bottom-side of Pixy's image.
4	Width of largest block, ranging between 1 and 255. A width of 255 is the full width of Pixy's image.
5	Height of largest block, ranging between 1 and 200. A height of 200 is the full height of Pixy's image.

Signature Query

This query simply asks Pixy to return the largest detected block of the specified signature. The specified signature is the Query Address - 0x50. For example, a Query Address of 0x51 would request the largest detected block of signature 1.

Query Address	Optional Query Data	Pixy Response
0x51 thru 0x57	none	5 bytes that describe the largest detected block within all signatures 1 thru 7. If no object is detected, all bytes will be 0.

0	Number of blocks that match the specified signature.
1	X value of center of largest detected block, ranging between 0 and 255. An x value of 255 is the far right-side of Pixy's image.
2	Y value of center of largest detected block, ranging between 0 and 199. A value of 199 is the far bottom-side of Pixy's image.
3	Width of largest block, ranging between 1 and 255. A width of 255 is the full width of Pixy's image.
4	Height of largest block, ranging between 1 and 200. A height of 200 is the full height of Pixy's image.

Color Code Query

This query asks Pixy to return largest block that matches the specified color code.

Query Address	Optional Query Data	Pixy Response
0x58	16-bit value (2 bytes) that specify the color code value encoded in octal (base-8).	6 bytes that describe the largest block that matches the specified color code. If no detected object matches the specified color code, all bytes will be 0.

Byte	Description
0	Number of blocks that match the specified color code.
1	X value of center of largest detected block, ranging between 0 and 255. An x value of 255 is the far right-side of Pixy's image.
2	Y value of center of largest detected block, ranging between 0 and 199. A value of 199 is the far bottom-side of Pixy's image.
3	Width of largest block, ranging between 1 and 255. A width of 255 is the full width of Pixy's image.
4	Height of largest block, ranging between 1 and 200. A height of 200 is the full height of Pixy's image.

5	Angle of largest color code block, ranging between 0 and 255. A value of 255 corresponds to an angle of 360 degrees.

Angle Query

A separate angle query is provided for when a General Query results in a color coded block. In this case, the missing angle information can be queried here. This query only works after first making a General Query.

Query Address	Optional Query Data	Pixy Response
0x60	none	1 byte which corresponds to the angle of largest color coded block returned by a previous General Query, The value ranges between 0 and 255. A value of 255 corresponds to an angle of 360 degrees.