

Integración de un Sistema de Visión Artificial a Robots Autónomos

Blanca Estela Rosas Ortega, Candy Sansores Pérez
Universidad del Caribe
050300249@ucaribe.edu.mx

05 de diciembre de 2009

Resumen. En este trabajo se integra una cámara CMUcam3 a un robot KheperaIII, estableciendo la comunicación entre ambos dispositivos a través de sus respectivos puertos seriales, para que el robot sea capaz de ubicar objetos en el entorno en el que se encuentra de forma más precisa que con los otros sensores que posee, ultrasónico e infrarrojo.

Palabras clave: Visión artificial, robótica de comportamiento, sistemas embebidos, compilador cruzado.

1 Introducción

El Laboratorio de Simulación de Sistemas Complejos, en las instalaciones de la Universidad del Caribe, cuenta con robots KheperaIII [1] utilizados en la investigación para simulación de inteligencia colectiva. Estos robots cuentan con 2 tipos de sensores:

- **Ultrasónicos:** generan ondas de sonido de alta frecuencia y evalúan el eco que es recibido de vuelta por el sensor. Los sensores calculan el intervalo de tiempo entre el envío de la señal y la recepción del eco para determinar la distancia a un objeto. Algunas ventajas de estos sensores son: bajo costo, tienen mayor alcance que los sensores infrarrojos, representan la información de forma compacta y rápida, etc. Entre sus desventajas se puede mencionar: la velocidad del sonido es variable ya que influye la densidad del aire y la concentración de polvo en el ambiente, la señal puede verse afectada por ruido eléctrico y reflexiones del sonido hacia distintos lugares que no son el receptor, etc [2].
- **Infrarrojos:** emiten una ráfaga de señales luminosas infrarrojas las cuales, al rebotar contra un objeto cercano, se reciben por otro componente. Al ser recibidas el sistema

detecta la proximidad, de acuerdo a la cantidad de luz que regresa. Entre sus ventajas se puede mencionar: indican la distancia a la que se encuentra un objeto y el material del que está hecho, precisión, etc. Algunas de sus desventajas son: debido a que la señal es direccional, es difícil detectar obstáculos por debajo y por encima de esta [2].

A pesar de las ventajas de este tipo de sensores, en la práctica se han detectado algunas desventajas que afectan la autonomía de estos robots, pues aún con esos sensores no son capaces de evitar todos los obstáculos que se presentan en su camino.

Actualmente los robots tienen la posibilidad de obtener mayor información del ambiente en el que se desenvuelven con el uso de cámaras, las cuales permiten el procesamiento, análisis y explicación de cualquier tipo de información espacial a través de imágenes obtenidas del entorno [3].

Las soluciones existentes para dotar al robot KheperaIII con un sistema de visión, como la cámara KheperaIII Wireless [4], no son óptimas para la autonomía de éstos, ya que los robots dependen de un servidor centralizado que interpreta las imágenes obtenidas con la cámara, que en su mayoría son inalámbricas, restando así independencia al robot, además de que la red utilizada se satura en caso de trabajar con una mayor cantidad de ellos. La necesidad de utilizar un servidor centralizado para el procesamiento de imágenes adquiridas por el robot viene de la poca capacidad de procesamiento que, en su mayoría, presentan éstos.

Por lo anteriormente explicado, se propone integrar al robot KheperaIII una cámara CMUcam3, la cual tiene la capacidad de manipular imágenes, para permitir al robot interpretar localmente las imágenes pre procesadas que reciba y actuar de acuerdo a ello, resolviendo así el problema de autonomía y extendiendo su capacidad de procesamiento; además, con la cámara, se pretende dar la oportunidad de integrar al robot algoritmos de navegación, identificación de patrones, procesamiento de imágenes, etc., y así mejorar su precisión al evaluar su entorno.

2 Sistemas embebidos: KheperaIII y CMUcam3

Para llevar a cabo el desarrollo del módulo de visión artificial propuesto es necesario conocer los aspectos teóricos y tecnológicos acerca del robot KheperaIII y la cámara CMUcam3, que son los componentes a utilizar en este proyecto. También se requiere conocer el concepto de Sistemas embebidos y Compilación cruzada para el desarrollo

del software que permita integrar ambos componentes y, posteriormente, proponer un algoritmo de navegación para probar el sistema desarrollado. En las siguientes líneas se hace una revisión de ellos.

2.1 Sistema embebido

Un sistema embebido es un sistema informático con un propósito específico, según el equipo en el que se encuentra integrado [5].

Puesto que los sistemas embebidos se pueden fabricar en cantidades pequeñas, suelen usar un procesador relativamente pequeño y una memoria pequeña para reducir los costos.

Como ejemplos de sistemas embebidos se puede mencionar la arquitectura del robot KheperaIII y la cámara CMUcam3, los cuales no son la excepción en contar con una memoria de tamaño reducido y capacidad de procesamiento limitada, por lo que es necesario desarrollar y construir las aplicaciones para estos componentes en una PC. Los sistemas embebidos requieren de un entorno de compilación cruzada que genere el código para su propia arquitectura desde una PC. En seguida se explica el funcionamiento de un compilador cruzado:

2.2 Compilación cruzada

Los compiladores cruzados permiten desarrollar software en una plataforma (el anfitrión/host) cuando realmente este software se está construyendo para un sistema alternativo (el objetivo/target) [6]. La computadora objetivo no necesita estar disponible: todo lo que se requiere es un compilador que sepa como escribir código máquina para la plataforma objetivo.

Los principales componentes de un compilador son:

- **Parser:** el parser convierte el lenguaje del código fuente en lenguaje ensamblador. El parser necesita conocer el lenguaje ensamblador del objetivo para hacer la conversión de un formato a otro.
- **Assembler:** el assembler convierte el código en lenguaje ensamblador a los bytecode que la CPU ejecuta.

- **Linker:** el linker combina los códigos objeto individuales, que genera el assembler, en una aplicación ejecutable. El linker necesita cual es el mecanismo de encapsulamiento que utiliza el sistema objetivo
- **Librería C estándar:** las funciones de C proporcionadas por una librería central de C. Esta librería, en combinación con el linker y el código fuente, es usada para producir el ejecutable final, suponiendo que las funciones de la librería de C son usadas en la aplicación.

En particular, en este proyecto de investigación, nos centraremos en dos dispositivos cuyos sistemas embebidos requieren combinarse para proporcionar una funcionalidad y servicio integrado.

2.3 KheperaIII

El primero de los dispositivos antes mencionados es el robot KheperaIII, utilizado para experimentos y demostraciones tales como:

- Navegación
- Inteligencia artificial
- Sistemas multiagente
- Control
- Comportamiento colectivo...

Este robot puede igualar el rendimiento de robots mucho más grandes gracias a las funciones disponibles en su plataforma, como su poder de cómputo escalable usando el sistema KoreBotLE, un conjunto de sensores de largo y de corto alcance para detección de objetos, un sistema de batería intercambiable para optimizar su autonomía, y una excepcional unidad de odometría [7].

La base del robot puede usarse con o sin la tarjeta KoreBotLE. Usando el KoreBot, ofrece un sistema operativo Linux estándar para el desarrollo de aplicaciones autónomas de manera rápida y fácil.

La base del robot incluye 11 sensores infrarrojos para detección de obstáculos; además cuenta con 5 sensores ultrasónicos para detección de objetos a largo alcance. Se le puede agregar un par de sensores infrarrojos de piso, para seguimiento de líneas y detección de bordes de mesas. También tiene conexiones para tarjetas de expansión de: soporte WiFi, Bluetooth, memoria externa, etc.

2.4 CMUcam 3

El segundo dispositivo con un sistema embebido es el CMUcam3, un sensor de visión completamente programable basado en ARM7TDMI. El procesador principal es el NXP LPC2106, conectado a un módulo CMOS de la cámara Omnivision. Se puede desarrollar código en C para el CMUcam3, usando un puerto del toolchain de GNU junto con un sistema de librerías abiertas y programas de ejemplo. Se pueden descargar ejecutables a la cámara usando el puerto serial [8]. Algunas de las funcionalidades de CMUcam3 abarcan:

- Robótica
- Vigilancia
- Redes de sensores
- Educación
- Juguetes interactivos
- Reconocimiento y seguimiento de objetos
- Servo control programable...

2.5 Algoritmo de navegación

Finalmente, una vez que la cámara esté acoplada al robot, será necesario probar su eficacia a través de algún algoritmo de navegación. Un algoritmo de navegación permite la identificación y el reconocimiento de las características o de los objetos distintivos del entorno, en el que se encuentra situado el robot, y que se conocen a priori o se extraen dinámicamente [9]. En otras palabras, estos algoritmos posibilitan a los robots a navegar en un entorno dinámico y desconocido, evitando obstáculos o siguiendo algún patrón en particular.

2.6 Módulos para KheperaIII

Existen varias soluciones que brindan al robot KheperaIII la capacidad de percibir su entorno a través de una cámara, sin embargo, a pesar de su funcionalidad y ventajas, no cumplen con todos los requisitos de autonomía y procesamiento necesarios para el robot. Entre estas soluciones se pueden mencionar las siguientes:

2.6.1 KheperaIII wireless

La cámara KheperaIII wireless [4] está diseñada para transmitir sonido y video desde el robot KheperaIII hacia un access point o router (802.11b/g). Imágenes y sonido pueden entonces ser procesados en una computadora remota, antes de enviar instrucciones de vuelta al robot usando una tarjeta KoreWifi. Es muy fácil conectar y usar la cámara. Se puede acceder al video y sonido con un simple navegador o con un software dado. La orientación de la cámara puede ser cambiada, de acuerdo a las necesidades del experimento.

2.6.2 KoreUSBCam

La cámara KoreUSBCam [10] es un módulo diseñado para capturar imágenes y video con el Korebot o el robot KheperaIII. Estas imágenes y video pueden ser procesadas en el Korebot o en una computadora remota con el KoreWifi.

3. Desarrollo y Resultados

3.1 Entorno de desarrollo y compilación de aplicaciones para los sistemas embebidos

El primer paso fue instalar, en una PC con sistema operativo Fedora 10, el entorno necesario para el desarrollo y compilación cruzada de las aplicaciones del robot KheperaIII, así como de la cámara CMUcam3. Para ello fue necesario seguir los manuales de instalación [11] [12] en los sitios web del robot y de la cámara, respectivamente.

3.2 Conexión y funcionamiento de los puertos seriales de los sistemas embebidos

El segundo paso fue hacer pruebas para comprobar la conexión y el funcionamiento de los puertos seriales TTL de ambos dispositivos, ya que por medio de éstos se establece la comunicación entre el robot y la cámara.

El robot KheperaIII cuenta con 3 puertos seriales: 1 estándar y 2 TTL en la tarjeta KoreBot. El puerto serial estándar (ttyS0) y uno de los puertos seriales TTL (ttyS1) son un acceso directo a la terminal del sistema operativo del robot. Por lo tanto, el puerto

serial TTL ttyS2 es el único libre para enviar información desde el robot a la cámara y viceversa (Figura 1).

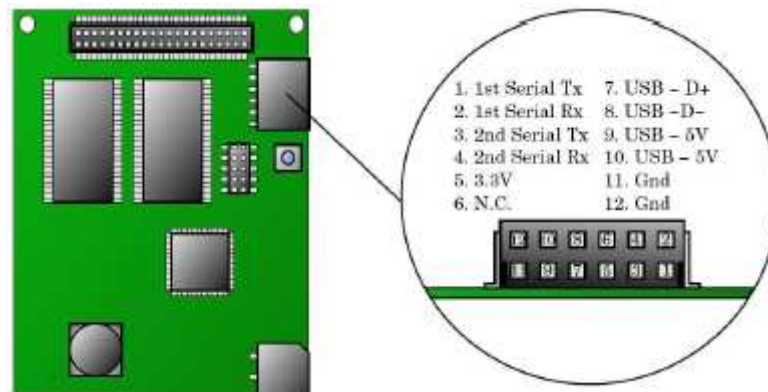


Figura 1. Puertos seriales TTL ttyS1 y ttyS2 en la tarjeta KoreBot

La cámara CMUcam3 cuenta con 2 puertos seriales: 1 estándar y 1 TTL (Figura 2). En este caso, solo se verificó la configuración necesaria, a nivel hardware, para usar el puerto serial TTL.

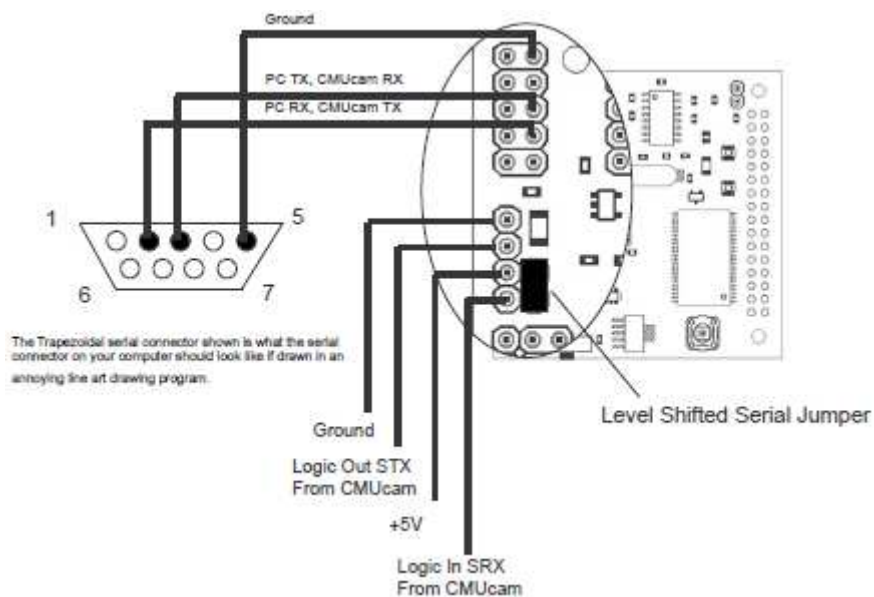


Figura 2. Puertos seriales en la CMUcam3

3.3 Interfaz física y lógica entre los sistemas embebidos

Una vez dominado el funcionamiento de los puertos seriales de ambos dispositivos, se procedió a realizar la interfaz física que los comunicaría. Se buscaron conectores que fueran compatibles con los puertos del robot y la cámara para hacer el cable.

Con la interfaz física lista, el siguiente paso fue realizar la interfaz lógica. Para ello se buscó una librería que permitiera manipular el puerto serial desde una aplicación. La

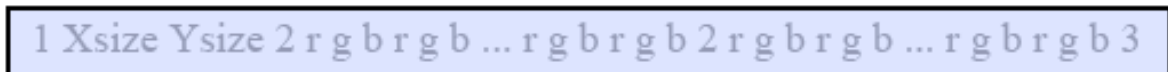
librería *libserial* [13] cuenta con las siguientes funciones, necesarias para la configuración del puerto serial, así como envío y recepción de datos a través de éste:

- `int SerDevOpen (const char *sSerDevName, int nBaudRate, int nByteSize, int cParity, int nStopBits, bool_t bRtsCts, bool_t bXonXoff)`: Abre y configura el puerto serial para comunicación.
- `int SerDevClose (int fd)`: Cierra el puerto serial.
- `int SerDevGetc (int fd, uint_t usec)`: Obtiene un caracter del puerto serial.
- `int SerDevPutc (int fd, byte_t byte, uint_t usec)`: Pone un caracter en el puerto serial.
- `ssize_t SerDevRead (int fd, byte_t *buffer, size_t count, uint_t usec)`: Lee desde el puerto serial.
- `ssize_t SerDevWrite (int fd, byte_t *buffer, size_t count, uint_t usec)`: Escribe en el puerto serial.

Esta librería puede ser compilada para diferentes arquitecturas, incluida la del robot KheperaIII. Se compiló para la PC con Fedora 10 y el robot, por lo que fue necesario descargar la librería *libnr* [14], que es requerida por la librería *libserial*.

La librería *libserial* se utilizó para desarrollar una aplicación que envía comandos desde una PC a la cámara, a través del puerto serial. Esta aplicación se hizo con la finalidad de analizar el tipo de datos que devuelve la cámara, y después realizar la aplicación que se ejecuta desde el robot y controla a la misma.

La aplicación envía comandos [15] a la cámara para que tome una imagen y después la guarde en formato jpg. La cámara responde a través del puerto serial de la siguiente forma:



```
1 Xsize Ysize 2 r g b r g b ... r g b r g b 2 r g b r g b ... r g b r g b 3
```

Figura 3. Formato de paquete F

donde:

1 = New frame

2 = New row

3 = End of frame

De este vector de datos se extrajeron únicamente los valores X y Y, para conocer las dimensiones de la imagen tomada, y los valores rgb para formar la imagen. Una vez convertidos los datos a formato jpg se obtuvo lo siguiente:

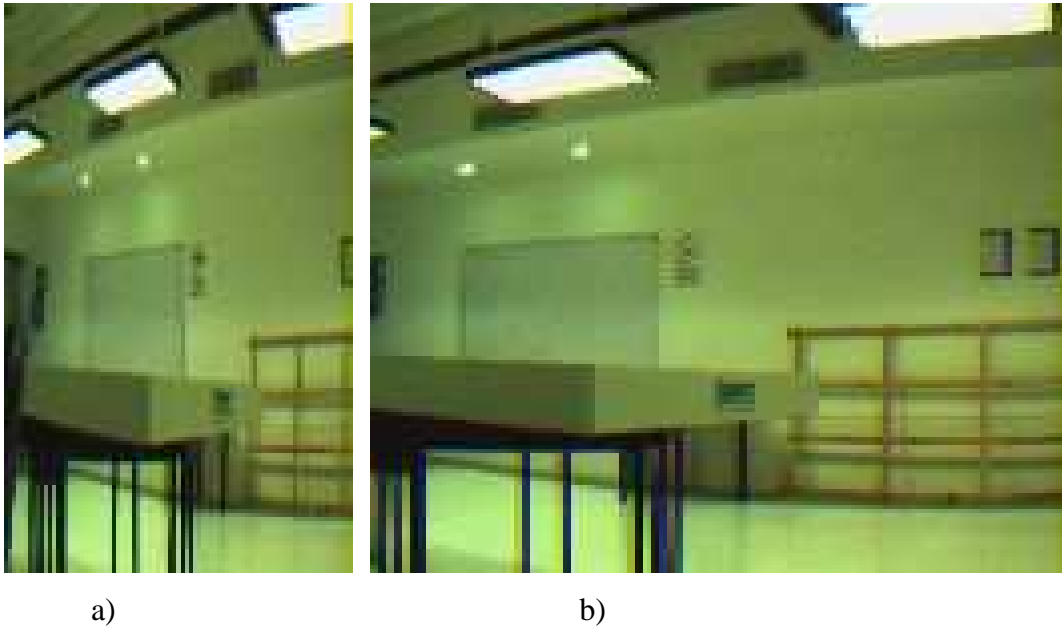


Figura 4. a) Imagen de 87x143 pixeles (tamaño estándar). b) Imagen de 174x142 pixeles (resolución mejorada)

La imagen a), mostrada en la Figura 4, tiene la resolución estándar de la cámara. Si se desea aumentar esta resolución para tener una imagen de mejor calidad, se debe duplicar cada columna de pixeles, obteniendo la imagen b) de la Figura 4.

Una vez comprobado el funcionamiento de los comandos de la cámara, así como del tipo de datos que devuelve, se realizó y compiló la aplicación para el robot.

3.4 Plataforma de la cámara

Para que la cámara esté fija sobre el robot se fabricó un soporte hecho con clips para papel (Figura 5).

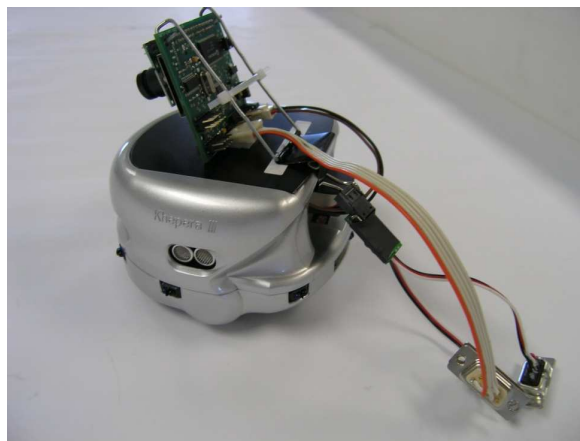


Figura 5. Plataforma de la cámara

3.5 Algoritmo de navegación

Establecida la comunicación física y lógica entre el robot y la cámara, se implementó un algoritmo de navegación, el cuál da la posibilidad al robot de ser más preciso al evaluar su entorno, usando la cámara en combinación con los sensores infrarrojos y ultrasónicos.

Para obtener información de la cámara se utilizó el algoritmo de navegación Polly, que permite al robot encontrar espacios libres en un área con obstáculos, usando visión a muy baja resolución, asumiendo que los pixeles de la parte baja de la imagen (la más cercana al robot) son el ejemplo de un área libre [16]. Este algoritmo da como resultado un histograma (Figura 6) en forma de vector, con una longitud igual a la del número de pixeles de ancho de la imagen.



Figura 6. Conversión de una imagen a histograma de espacio libre con el algoritmo Polly

En la Figura 6, la imagen izquierda es la original, la de en medio es la filtrada y con bordes detectados, y la final es el histograma. El histograma se forma pintando de un mismo color cada columna de pixeles, desde la parte baja de la imagen, hasta que se encuentra un borde. Así, se hace navegar al robot hacia los picos del histograma.

La información obtenida con los sensores de proximidad se utiliza para que el robot funcione como un vehículo de Braitenberg. La suma de los valores devueltos por los sensores influyen directamente en la velocidad de las ruedas del robot: mientras un objeto esté más cerca, mayor será ese valor y el robot se moverá más rápido, con una rueda en sentido positivo y la otra en sentido negativo para poder girar.

La unión del algoritmo Polly y el vehículo de Braitenberg se hizo escalando los valores del histograma a los de los sensores de proximidad, para que así pudieran trabajar en conjunto. El histograma se divide a la mitad, verticalmente, y entonces se conoce si el robot debería moverse a la derecha o la izquierda. Para que la suma de valores de cada mitad del histograma coincida con la de los sensores de proximidad, fue necesario invertirlo (Figura 7), así se obtuvo un valor mayor cuando el obstáculo estaba más

cerca, y uno menor cuando el obstáculo se encontraba más lejos; de esa forma los valores resultantes se pudieron sumar a la velocidad de las ruedas directamente.

El nuevo algoritmo de navegación logró que el robot esquivara obstáculos fuera del alcance de los sensores de proximidad, pero las sombras y brillos en la superficie por la que se desplaza los confunde con obstáculos. Por lo anterior, es necesario hacer mejoras al algoritmo.

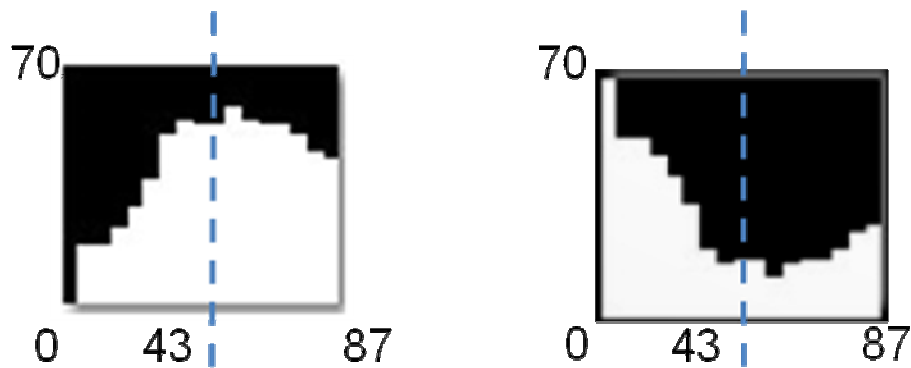


Figura 7. Histograma original e histograma invertido

Referencias

1. “KheperaIII” en:
<http://www.k-team.com/kteam/index.php?site=1&rub=22&page=197&version=EN>
2. http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/amador_g_ja/portada.html
3. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/garcia_b_ci/portada.html
4. “Cámara KheperaIII Wireless” en:
[http://www.k-](http://www.k-team.com/kteam/index.php?site=1&rub=3&upPage=230&page=197&version=EN)
[team.com/kteam/index.php?site=1&rub=3&upPage=230&page=197&version=EN](http://www.k-team.com/kteam/index.php?site=1&rub=3&upPage=230&page=197&version=EN)
5. “Sistema embebido” en: http://es.wikipedia.org/wiki/Sistema_integrado
6. “Compilación cruzada” en: <http://www.ibm.com/developerworks/edu/1-dw-1-cross-i.html>
7. “Manual Khepera III” en:
http://ftp.k-team.com/KheperaIII/Kh3.Robot.UserManual.2.2_booklet.pdf
8. “CMUcam3” en: <http://www.cmucam.org/wiki/Documentation>
9. HU, Huosheng and GU, Dongbing, *Landmark-based navigation of industrial mobile robots*, The Industrial Robot 27, no. 6, 2000.

10. “KoreUSBCam” en:
<http://www.k-team.com/kteam/index.php?site=1&rub=3&upPage=233&page=197&version=EN>
11. “Development for Korebot with kernel 2.6 and the Cross-compiler from x86-linux to arm-linux, full version” en:
http://ftp.k-team.com/korebot/toolchain-2.6-betaV0.1/full_toolchain/development_toolchain_full_version_2.6_readme.txt
12. “CMUcam3 Linux Quick Start Guide” en: <http://www.cmucam.org/wiki/Linux-Quick-Start>
13. “libserial” en: <http://www.roadnarrows.com/customer/libserial>
14. “libnr” en: <http://www.roadnarrows.com/customer/>
15. “Manual CMUcam2” en:
http://www.cmucam.org/attachment/wiki/Documentation/CMUcam2_manual.pdf
16. “Polly” en: <http://www.cmucam.org/wiki/polly>