Revista Tekhnê
Julio - Diciembre 2016, Vol. 13, No. 2, pp. 13– 18

ISSN 1692-8407

Diseño de plataformas robóticas diferenciales conectadas en topología Mesh para tecnología Zigbee en entornos cooperativos

Differential wheeled robotic platforms design in Mesh topology for connected Zigbee technology in cooperative environments

Jhon J. Benitez N.
Universidad Distrital Francisco José de Caldas
jjbenitezn@correo.udistrital.edu.co

Lisbeth J. Parra C.
Teshboss SAS
ljparrac@correo.udistrital.edu.co

Holman Montiel A.
Universidad Distrital Francisco José de Caldas
hmontiela@udistrital.edu.co

Este artículo muestra el desarrollo de un sistema de navegación basado en robótica tipo enjambre con un computador y tres plataformas diferenciales. El computador actúa como coordinador y las plataformas como routers haciendo uso del protocolo de comunicación ZigBee. Por medio de los módulos X-bee serie 2, un sensor ultrasónico y un odómetro, se estructura un esquema de control cooperativo con una interfaz gráfica configurable para el control de individual de los robots. El sistema se validó sobre prototipos reales demostrando un alto desempeño y versatilidad de uso.

Palabras clave: Comunicación, datos, mesh, robótica de enjambres

This paper shows the development of a navigation system based on swarm robotic with a computer and three differential wheeled platforms. The computer acts as a coordinator and platforms as routers using the ZigBee communication protocol. Through the X-bee series 2 modules, an ultrasonic sensor and an odometer, we structure a cooperative control scheme with a configurable graphical interface for the individual control of the robots. The system was validated on real prototypes demonstrating high performance and versatility of use.

Keywords: Communication, data, mesh, swarm robotics

Tipología del artículo: Investigación

Fecha recepción del manuscrito: Noviembre 4, 2016
Fecha aceptación del manuscrito: Diciembre 5, 2016

Investigación financiada por: Universidad Distrital Francisco José de Caldas.

Edición digital: http://revistas.udistrital.edu.co/ojs/index.php/tekhne/issue/view/798
Cómo citar: Benitez, J., Parra, L. y Montiel, H. (2016). *Diseño de plataformas robóticas diferenciales conectadas en topología Mesh para tecnología Zigbee en entornos cooperativos*. Revista Tekhnê, 13(2), 13-18.

Introducción

La robotica tipo enjambre se basa en la coordinación y desarrollo de un trabajo conjunto entre robots simples para llegar a un fin concreto cooperativo sin tener que realizar tareas largas. De esta manera se logra mejorar los tiempos de respuesta en la ejecución del proceso establecido. Se toma como ejemplo principal el comportamiento de algunos insectos, como las hormigas, o la función desarrollada por las células en los seres vivos. En la actualidad, esta área de investigación tipo enjambres ha tomado gran fuerza debido a las ventajas obtenidas respecto a la ejecución y simplicidad en los dispositivos (Sempere y Mireia, 2014).

Teniendo en cuenta las ventajas de la robotica tipo enjambre, se diseñaron tres plataformas roboticas diferenciales con dos sensores de navegación (odómetro y ultrasonido). Estas plataformas están conectadas a una red tipo Mesh, basada en el protocolo Zigbee, usando módulos XBEE S2 (Martinez S. y Delgado, 2012). Esta estructura permite la creación de un entorno cooperativo.

Se diseñó una interfaz gráfica que permite controlar las plataformas que conforman el sistema siguiendo tres pasos: en primer lugar, escoger la plataforma que se desea mover. En segundo lugar, ingresar la dirección del movimiento o la lectura del sensor, y por último, asignar la distancia a recorrer. Adicionalmente, la interfaz permite visualizar un mensaje de respuesta por parte del sistema.

Cada plataforma tiene integrada una tarjeta Arduino UNO que se encarga de la recepción e interpretación de la trama de datos con el fin de generar el movimiento definido para la plataforma. Sin embargo, el movimiento se ejecutará si y solo si el sensor de ultrasonido no detecta obstáculos en la distancia suministrada. En la Fig. 1 se muestra un diagrama del funcionamiento del sistema.

Metodología

Diseño de la topología Mesh

En el diseño de la topología tipo Mesh se implementaron cuatro módulos XBEE S2 configurados en modo AT (comunicación AT, configuración transparente, sin codificación), para la ejecución del proyecto, y en modo API (comunicación API, transmisión empaquetada que permite identificar el emisor, configurar y monitorear remotamente (Digi International, 2009)), para la visualización de la red. Tres de estos módulos actúan como *Routers* y el restante tiene el rol de Coordinator. Todos estos tienen configuración tipo *broadcast* permitiendo la comunicación total entre todos los dispositivos. En la Fig. 2 se detalla la estructura de comunicación de la topología tipo Mesh en el software X-CTU.

Broadcast

Los módulos XBEE permiten tener una conexión punto a punto o multipunto. En este caso se realizó una configuración multipunto o *broadcast* que consiste en transmitir datos desde un elemento de la red a todos los dispositivos conectados a esta de forma simultánea y sin necesidad de hacer el envío uno a uno. Esta configuración se realizó en el programa X-CTU que, a su vez, permitió verificar el funcionamiento correcto de la red (Mayalarp, Limpaswadpaisarn, Poombansao, y Kittipiyakul, 2010).

En la tabla 1 se muestra la versión de *firmware* utilizada para cada uno de los módulos con su respectiva dirección.

Tabla 1
Configuración de red para los módulos.

ESTRUCTURA DE FIRMWARE				
Nodo Identificador	Firmware	Dirección de red	Número Serial en Alto	Número Serial en bajo
COORDINADOR	20A7	1996	0013A200	40E429AC
ROUTER 1	22A7	1996	0013A200	40F49EEC
ROUTER 2	22A7	1996	0013A200	40AEBAB1
ROUTER 3	22A7	1996	0013A200	40AEBA9C

Como se observa en la tabla 1, el Coordinador tiene la versión de *Firmware* 20A7 y los *Routers* cuentan con la versión 22A7. Estas son las últimas versiones disponibles a la fecha del software. Se muestran también las direcciones físicas de cada uno de los dispositivos y la dirección de la red o PAN ID, la cual es la misma en todos los módulos para poder lograr una comunicación transparente y eficaz en la red.

En las Figs. 3, 4, 5 y 6 se evidencia la apropiada configuración de *broadcast* y de direccionamiento de cada módulo.

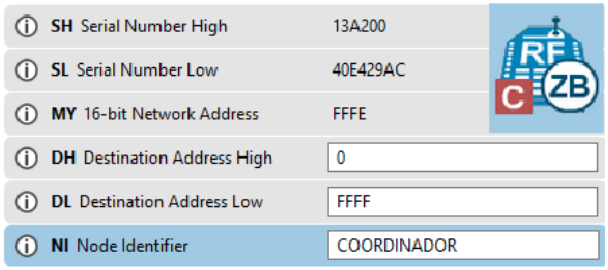

Figura 3. Direccionamiento y configuración de *broadcast* para módulo Coordinador. Tomada de X-CTU.

En las figuras se corrobora la información anteriormente expuesta en la tabla 1, y se muestra la dirección de destinatario. En esta última se debe digitalizar la dirección física del dispositivo con el que se desea comunicar, y así realizar una conexión punto a punto. Para realizar una conexión tipo *broadcast* se escoge como dirección en alto (*Destination Address High*) 0000 y como dirección en bajo (*Destination Address Low*) $FFFF$, ya que, de esta manera,

ISSN 1692-8407

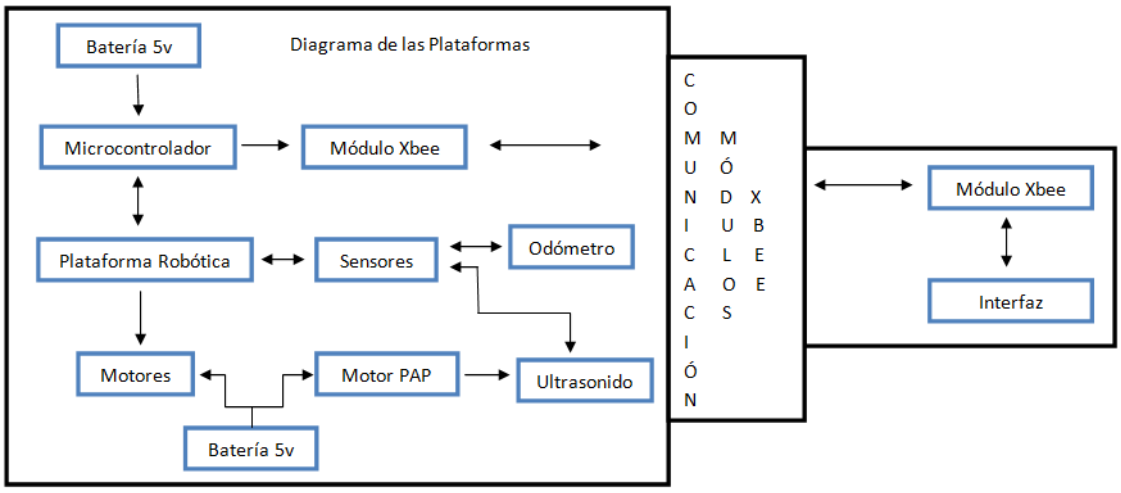

Figura 1. Diagrama de funcionamiento de la plataforma robotica.

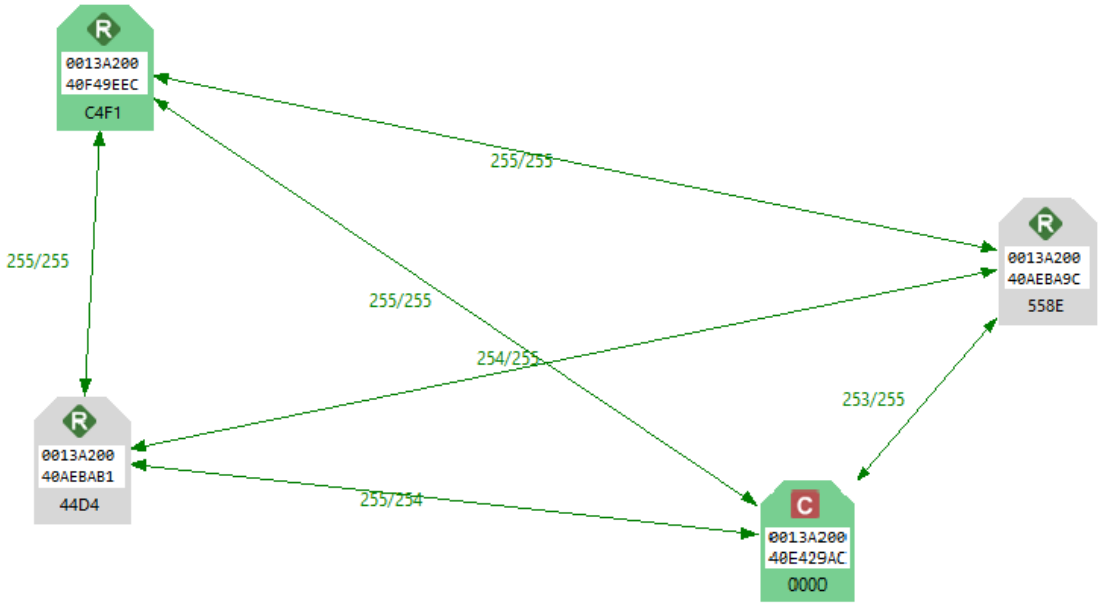

Figura 2. Red de comunicación.

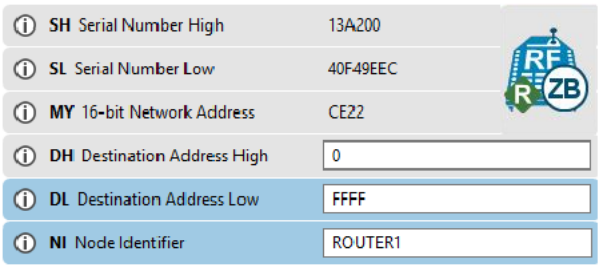

Figura 4. Direccionamiento y configuración de *broadcast* para módulo Router1. Tomada de X-CTU.

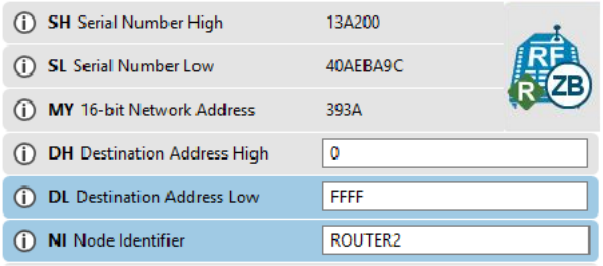

Figura 5. Direccionamiento y configuración de *broadcast* para módulo Router2. Tomada de X-CTU.

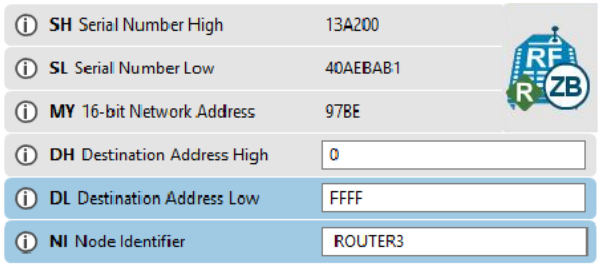

Figura 6. Direccionamiento y configuración de *broadcast* para módulo Router3. Tomada de X-CTU.

el módulo identifica que la comunicación es para todos los dispositivos de la red.

Adquisición de datos

Para la adquisición de datos se utilizó un sensor de ultrasonido HC-SR04, el cual permite detectar obstáculos a una distancia de entre 4 y 300 cm, con un ángulo de proyección de 40 grados en la dirección en la que se encuentra. Su rango de funcionamiento de muestra en la Fig. 7.

Para garantizar que los robots no colisionen y que puedan realizar un movimiento óptimo, el sensor ultrasónico cambia su posición inicial gracias a un motor paso a paso (PAP) que genera movimientos entre 17 y 20 grados por pulso. Esto garantiza que el sensor cubra un mayor ángulo de visión en su posible desplazamiento.

Con el fin de validar el funcionamiento de la plataforma, se diseñó un programa en la tarjeta Arduino UNO para activar el sensor y hacer que este emita un sonido a una frecuencia no audible. La tarjeta recibe un pulso en el momento en el que el sensor detecta de vuelta el sonido y determina así el tiempo entre el envío y el retorno del sonido. Este tiempo se utiliza con el fin de obtener la distancia entre el sensor y el obstáculo más cercano (ecu. 1) .

$$distancia = \frac{tiempo}{58\frac{s}{cm}} \tag{1}$$

También se implementó un odómetro por medio de un encoder rotatorio en cada llanta, el cual está compuesto por un emisor infrarrojo, un sensor infrarrojo y un disco con ranuras. El sensor infrarrojo ve interrumpida la luz emitida por el emisor, a causa del constante movimiento del disco con ranuras. La tarjeta Arduino cuenta los pulsos producidos por dichas interrupciones, dato con el cual es posible establecer la cantidad de vueltas de la llanta. Se hace uso de la ecu. 2 para calcular la distancia recorrida por la plataforma.

$$distancia_{robot} = vueltas_{llanta} \times 22\,cm \tag{2}$$

Implementación de módulos XBee

A través de una conexión Rx/Tx entre el módulo y la Arduino UNO, se logró el correcto funcionamiento de la red. Para realizar el envío de datos, se diseñó un protocolo de empaquetamiento, el cual tiene como objetivo controlar la recepción de datos en la tarjeta y de esta manera poder entender cada uno de los caracteres recibidos.

Empaquetamiento. El empaquetamiento diseñado consiste en iniciar la cadena de datos con el símbolo @ y finalizarla con el símbolo #. Esto, para garantizar que la trama llegue completa y que no se reciba información no deseada. En la Fig. 8 se puede observar el modo idóneo de enviar el mensaje.

Figura 8. Modo de empaquetamiento.

Como se observa, la cadena de datos está compuesta por 8 bytes. A continuación se describe la función específica de cada byte.

- @: Byte de Inicio.
- P1: Bytes de selección de plataforma (P1, P2 ó P3).
- A: Bytes de selección de dirección o lectura de sensor (A, R, D, I, S).
- 100: Bytes de distancia a recorrer en cm.
- #: Byte de finalización.

Interfaz

El desarrollo de la interfaz se realizó en JAVA. El IDE seleccionado fue NetBeans versión 8.2. La comunicación serial realizada fue permitida gracias a la librería Giovynet Driver 2.0, la cual se define como: *un marco de trabajo o framework que posibilita el uso de lenguaje Java para crear aplicaciones que se comuniquen con circuitos externos al PC*. Como se pudo observar en el proyecto, esta librería permitió la transmisión bidireccional de datos de forma limpia y eficaz con las plataformas robóticas. A continuación, se muestran los pasos realizados para el diseño de la interfaz.

1. Instalación de la librería Giovynet Driver 2.0.
2. Validación de puertos COM permitidos.
3. Implementación de botones, cajas de texto, etiquetas, etc.
4. Creación del código para la transmisión de datos.
5. Creación del código para la recepción de datos.
6. Visualización de los datos recibidos.

En la Fig. 9 se muestra la interfaz diseñada, la cual está compuesta por:

- 2 *combobox* que facilitan la selección de la plataforma requerida junto con la acción que se desea ejecutar.

ISSN 1692-8407

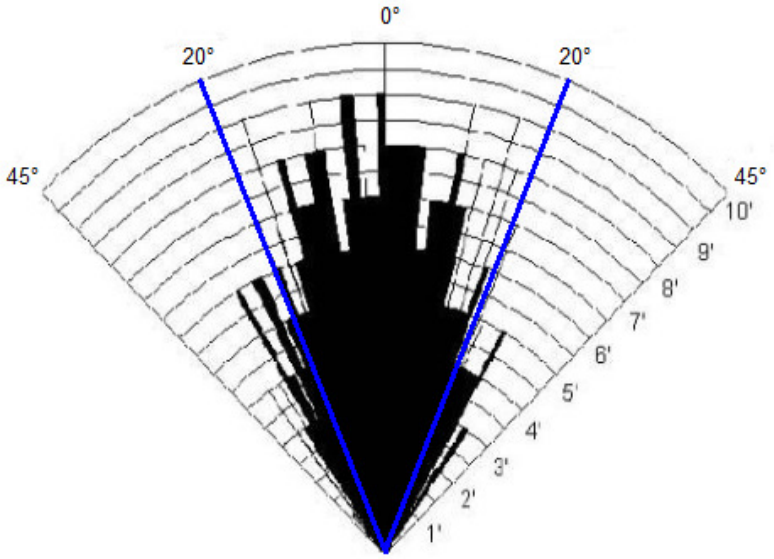

Figura 7. Lóbulo radiado sensor HC-SR04, suministrada por el fabricante, editada por los autores (Allnet, 2014).

• 4 *textbox* de los cuales uno permite realizar la digitación de la distancia a recorrer en centímetros y los otros tres muestran la respuesta obtenida por la plataforma seleccionada.

• 1 *button* que da inicio a la transmisión y a la recepción de datos.

• 7 *labels* que proporcionan una ayuda para el usuario con el fin de garantizar el buen manejo de la interfaz.

Figura 9. Interfaz gráfica en Java.

Pruebas y resultados

En el presente documento se mostró el desarrollo de un sistema de navegación que tiene como función principal analizar un espacio determinado en el momento en que el usuario lo desee, a través de un mensaje transmitido por los módulos XBee S2 que se encuentran conectados en una red tipo Mesh por medio de la topología Zigbee. Esta red de comunicación se basa en una configuración tipo *broadcast* que permite la transmisión de datos a todos los dispositivos de la red. A continuación, se muestran las diferentes pruebas y sucesos que permitieron el buen funcionamiento de los dispositivos.

Al realizar las pruebas de comunicación con los robots se obtuvieron diferentes tiempos de respuesta dependiendo de la distancia entre los módulos y los objetos de interferencia encontrados. El rango aproximado de respuesta fue de 9 a 11 segundos por robot. Este retardo se debe al movimiento que debe realizar el motor PAP para que el sensor tenga mayor rango de visión, como se muestra en la Fig. 10.

Como se puede apreciar en la Fig. 10, el retardo entre los pulsos del motor es de 500 ms. Para realizar un giro de 180 grados es necesario generar 10 pulsos en el motor, esto conlleva a que el retardo total de este movimiento sea la multiplicación entre el retardo y la cantidad de pulsos, para, de esta forma, generar un retardo total de 5 segundos. Para el retorno a la posición inicial del motor PAP se implementó el código que se muestra en la Fig. 11.

Se observa en la Fig. 11 que el retardo entre los pulsos del motor es de 100 ms. Para realizar el retorno es necesario generar 10 pulsos en el motor, por lo que al realizar el mismo proceso anterior, se obtiene un retardo final de 1 segundo para este movimiento. El proceso de la adquisición de datos toma un tiempo estimado de 6 segundos, y el proceso de transmisión y recepción de estos varía entre 2 y 5 segundos,

```
void derecha(){
  for (a=0 ; a<3 ; a++){
  switch (a) {
      case 0:
    digitalWrite(bobina0,HIGH); delay(500);
    digitalWrite(bobina0,LOW);    b=b+1;
    ultrasonido();
    break;
      case 1:
    digitalWrite(bobina1,HIGH); delay(500);
    digitalWrite(bobina1,LOW);    b=b+1;
    ultrasonido();
    break;
      case 2:
    digitalWrite(bobina2,HIGH); delay(1000);
    digitalWrite(bobina2,LOW);    b=b+1;
    ultrasonido();
    break;
  }
  while(b==10){
    a=8; b=11; e=1;
    mySerial.println(prueba);
    digitalWrite(bobina2,LOW);
    digitalWrite(bobina1,LOW);
    digitalWrite(bobina0,LOW);
   }
```

Figura 10. Fragmento de código usado para el movimiento del motor PAP.

```
void izquierda(){
 for (c=0 ; c<3 ; c++){
 switch (c) {
     case 0:
   digitalWrite(bobina2,HIGH);  delay(100);
   digitalWrite(bobina2,LOW);   d=d+1;
   break;
     case 1:
   digitalWrite(bobina1,HIGH);   delay(100);
   digitalWrite(bobina1,LOW);    d=d+1;
   break;
     case 2:
   digitalWrite(bobina0,HIGH);   delay(100);
   digitalWrite(bobina0,LOW);    d=d+1;
   break;
 }
  while(d==10){
    c=0;
    d=0;
    f=1;
   digitalWrite(bobina2,LOW);
   digitalWrite(bobina1,LOW);
   digitalWrite(bobina0,LOW);
   Serial.print('\n');
  }
 }
```

Figura 11. Fragmento de código usado para el retorno del motor PAP.

dependiendo de la distancia y los objetos que interfieren en la línea de transmisión de datos.

Para la validación de la comunicación se realizaron envíos de mensajes de prueba en los diferentes módulos como se muestra en la Fig. 12.

```
70 72 75 65 62 61 20
72 6F 75 74 65 72 20 31
72 6F 75 74 65 72 20 32
```

Figura 12. Cadena de prueba transmitida.

Se consideran en la Fig. 12 tres cadenas en hexadecimal. La cadena azul hace referencia a la transmisión de datos realizada por el coordinador de la red, y las cadenas en color rojo hacen referencia a los datos recibidos por el mismo. Con esto se muestra la comunicación bidireccional y limpia a través de la red.

Conclusiones

Se creó un algoritmo como alternativa de solución para recibir, desempaquetar y comparar la información transmitida por los módulos. Esto se realiza con el fin de determinar el momento en el que se debe realizar la lectura del sensor ultrasónico, y seguido de esto, generar un movimiento en la plataforma. A través del uso de la interfaz gráfica, desarrollada en Java, se puede realizar el control del sistema de navegación, permitiendo escoger la dirección y la distancia a recorrer por los robots. Se puede evidenciar en la topología Mesh que al conectar una gran cantidad de dispositivos se permite una comunicación con mayor alcance. En caso de presentar alguna anomalía en el entorno, se procede a buscar canales recurrentes para la transmisión de datos.

Referencias

Allnet. (2014). *Dyp serial ultrasonic sensor.* On line.

Digi International. (2009). *XBee/XBee-PRO RF Modules.* On line. 11001 Bren Road East, Minnetonka, MN 55343. (Product Manual v1.xEx - 802.15.4 Protocol)

Martinez S., F. H., y Delgado, J. (2012, January). Wireless visual sensor network robots-based for the emulation of collective behavior. *Revista Tecnura*, *16*(31), 10-18. (ISSN 0123-921X)

Mayalarp, V., Limpaswadpaisarn, N., Poombansao, T., y Kittipiyakul, S. (2010). Wireless mesh networking with xbee. En *2nd ecti-conference on application research and development (ecti-card 2010)* (p. 1-5).

Sempere, T., y Mireia, L. (2014). *Agentes y enjambres artificiales: modelado y comportamientos para sistemas de enjambre roboticos*. Tesis Doctoral no publicada, Universidad de Alicante. Departamento de Ciencia de la Computación e Inteligencia Artificial.