3. Nivel de Inteligencia

3.1. Introducción

Atendiendo a los niveles de la torrebot definidos en la introducción, este es el grado más alto que puede alcanzar un robot individualmente. En este nivel se realiza el control orientado hacia una tarea concreta, con unos objetivos claros, y es probablemente el nivel más complejo.

En este capitulo se realiza una breve introducción a los distintos tipos de arquitecturas de control que se han ensayado sobre robots a lo largo de su historia, y se profundizará en la elegida para este proyecto y su integración sobre el sistema realizado.

3.2. Breve historia de las Arquitecturas de Control en Robótica

Desde un principio, el principal objetivo de la comunidad científica que se dedicó al campo de la robótica móvil era el de construir ingenios que, a partir de la información que percibían de sus sensores de ultrasonidos, infrarrojos, odométricos[1] y cámaras de vídeo, fuesen capaces de moverse por el entorno y realizar una serie de tareas de forma autónoma. Un robot móvil debe ser capaz de procesar la información sensorial suministrada por los sensores y adoptar una acción que le lleve a un estado definido como meta.

Realizando un repaso histórico, se puede asegurar que gran parte del entusiasmo inicial por la Inteligencia Artificial en sus orígenes fue debido al éxito con que se resolvieron ciertos problemas, como jugar al ajedrez o la resolución automática de teoremas. En la Inteligencia Artificial, que se puede definir como Clásica o Inteligencia Artificial Basada en el Conocimiento, todos los esfuerzos se concentraron en definir una serie de estructuras simbólicas formales y métodos de inferencia que permitiesen un razonamiento sobre estas estructuras. Es decir, se trataba de definir una base de conocimiento y un sistema inteligente (máquina de inferencia) que fuese capaz de interpretar esa información simbólica y obtener resultados (en forma de símbolos) de forma automática. Siguiendo ese tipo de filosofía, los investigadores de la rama de la robótica móvil trataron de conectar los ingenios manipuladores de símbolos a los sensores y actuadores de los robots móviles. En principio, los módulos sensoriales deberían proveer a la base de conocimiento de un conjunto de símbolos que identificasen cada uno de los elementos que estuviesen percibiendo. Uno de los principales objetivos era el de generar un modelo simbólico del mundo (escenario cartesiano) a partir de la información sensorial, de tal forma que la maquina de inferencia pudiera razonar y planificar a-priori los movimientos del robot. Los requerimientos de este tipo de sistemas son de arriba a abajo. Es decir, primero se especificaban el formato de la

[1] Los sensores odométricos miden los desplazamientos del robot en el entorno, son sensores de este tipo los encoders de las ruedas ó sistemas más complejos como los inerciales.

base simbólica y la máquina de inferencia y, más tarde, se desarrollaban los módulos de percepción que deberán proveer los símbolos necesarios: Por ejemplo, el ingeniero de sistemas que estaba desarrollando la base de conocimiento y la máquina de inferencia del robot suponía que el ingeniero de visión artificial realizaría un modulo capaz de transformar las imágenes percibidas en símbolos de ese tipo.

A mediados de los años ochenta, Rod Brooks, en respuesta a la visión tradicional de la Inteligencia Artificial, introdujo una nueva filosofía de diseño aplicada a la robótica móvil que el definió como Arquitecturas Reactivas. Observando que los robots desarrollados hasta entonces eran excesivamente lentos y en absoluto robustos, reemplazó la modularización funcional por módulos generadores de comportamientos. Una jerarquía de procesos se hace cargo del sistema; cada uno de estos procesos era extremadamente simple, y el corto camino entre los sensores y los actuadores (desde un punto de vista computacional) proporcionaba una respuesta en tiempo real del robot ante cambios en el entorno. La implementación del sistema se realizaba de abajo a arriba, de modo que primero se trataba de asegurar la supervivencia del robot, y después se le iban asignando nuevas y más complejas aptitudes. Por otro lado, las Arquitecturas Basadas en Comportamientos, al igual que las arquitecturas reactivas, están distribuidas y se componen de una colección de comportamientos que se ejecutan concurrentemente, sin necesidad de un planificador o arbitro central. En claro contraste con las arquitecturas reactivas, los comportamientos son más poderosos que las reglas puramente reactivas, puesto que incorporan memoria y un preprocesamiento de la señal sensorial.

Recientemente, las llamadas Arquitecturas Híbridas han ganado popularidad. Estas arquitecturas ofrecen un compromiso entre las arquitecturas puramente reactivas y las arquitecturas tradicionales basadas en un planificador. Se componen de un sistema reactivo para ejecutar tareas de bajo nivel, y de un planificador para establecer tareas de más alto nivel. Estas arquitecturas separan el sistema en dos o más partes independientes, que se comunican. En la mayor parte de los casos, los procesos reactivos de bajo nivel se encargan de la integridad del robot en cada instante, mientras que el planificador selecciona toda una serie de acciones que ejecutar en el futuro.

3.3. Control por conductas y la TC-FPGA

Como se apuntaba en la introducción del proyecto, la filosofía del sistema creado está orientado hacia la filosofía del control mediante conductas. En este apartado se profundiza en dicha filosofía a nivel de implementación y como se adecua en el sistema diseñado, así como las ventajas y desventajas frente a la implementación en un µC.

3.3.1. Control por Conductas

La idea básica en la que se sustentan las arquitecturas reactivas es la descomposición del problema en niveles de competencia horizontales. Se construyen niveles del control para cada uno de los niveles de competencia, añadiendo nuevas capas a las ya existentes. Se comienza construyendo un robot completo que logre acometer una competencia de grado cero, y depurando este control se consigue un robot que realiza a la perfección las especificaciones encomendadas a ese nivel. Tal y como se puede observar en la figura 3.1, un nivel cero de competencia sería aquel que asegurase la supervivencia del robot en el entorno. Es decir, un nivel que reaccione a la señal sensorial de tal forma que cuando el robot se encuentra muy cercano a un obstáculo lo evite. Sobre esta capa de control se puede añadir una segunda capa, en la que el objetivo de la misma sea la de vagar por el entorno, sin tener en principio un objetivo claro. De esta forma, el control del robot está estratificado de modo que los niveles inferiores toman el control del robot cuando eventos del entorno así lo deciden. El sistema se puede cortar en cualquiera de los niveles implementados, y los niveles inferiores formarán un sistema de control operativo completo.

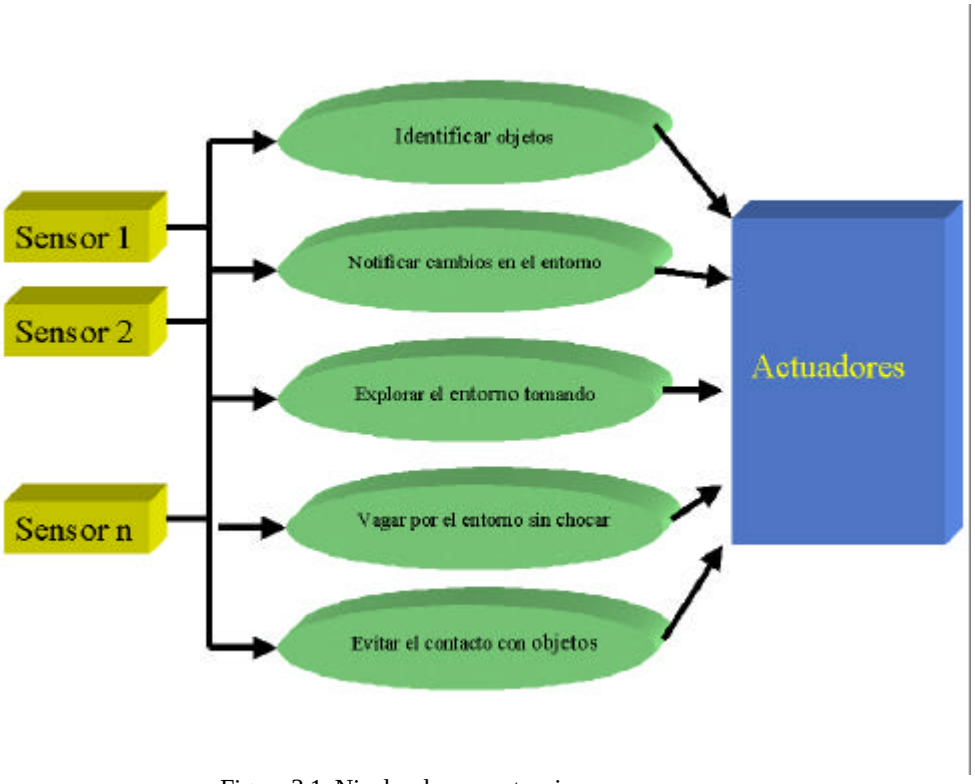

Figura 3.1: Niveles de competencia

Concretamente la arquitectura de subsunción (traducción directa del inglés *Subsumption architecture*), concebida por R. Brooks, utiliza los denominados AFSM[1], que son máquinas de estados finitos a las que se añaden temporizadores,

[1] Augmented Finite State Machine

para construir reglas de control simples. Los AFSMs son muy importantes dentro de este tipo de arquitectura, ya que suponen el ladrillo básico para la construcción de conductas. En la figura 3.2 se puede observar una de estas estructuras.

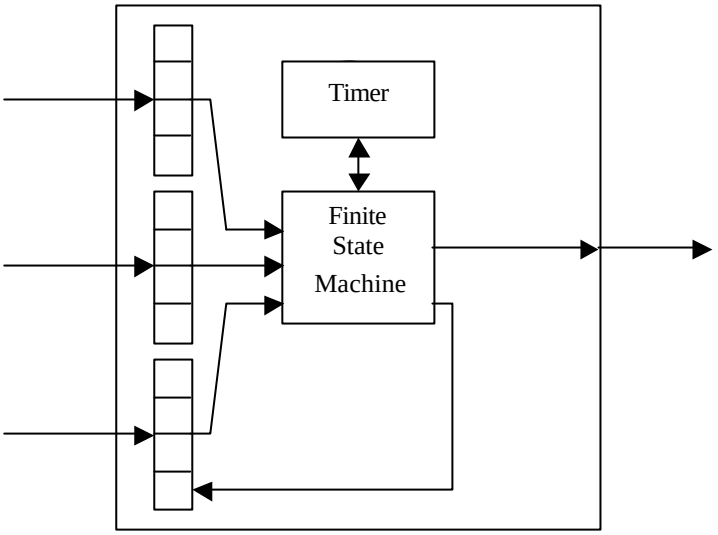

Figura 3.2: AFSM, "Augmented Finite State Machine"

Los AFSMs se comunican mediante un protocolo de paso de mensajes y de supresión e inhibición de mensajes. Mediante la supresión un AFSM puede suprimir las entradas que recibe de otro AFSM durante un periodo de tiempo, mientras que mediante la inhibición se anulan las salidas. La combinación de diversos AFSMs forman comportamientos que son los constituyentes de la arquitectura de subsunción, y que sirven como niveles de abstracción. Los comportamientos son combinados en niveles de competencia, o capas, que se corresponden con cada una de las habilidades del robot (esquivar obstáculos, vagar por el entorno, seguir paredes...etc.).

Mediante esta técnica de ir añadiendo niveles a los ya preexistentes se consigue que cada una de las capas pueda estar trabajando en diferentes metas concurrentemente. Por otro lado, se obvia por completo el problema de la fusión sensorial, ya que cada uno de los módulos extraerá de los sensores la información que le sea necesaria para su objetivo particular: No hay ninguna necesidad de generar un modelo del mundo donde se fusione la información proporcionada por los sensores. La robustez del sistema se garantiza por el desarrollo del diseño del sistema, donde se implementan y depuran los niveles de competencia más bajos antes de que se añada una nueva capa. Finalmente, la arquitectura permite que se vayan añadiendo cada vez más capas de competencia, aumentando por tanto las habilidades del robot.

A continuación se muestra en la figura 3.3 un ejemplo de este tipo de control, concretamente pertenece al robot ALLEN del propio Brooks, este robot contempla 3 niveles básicos de competencia, evitar obstáculos, vagar por el entorno y explorar.

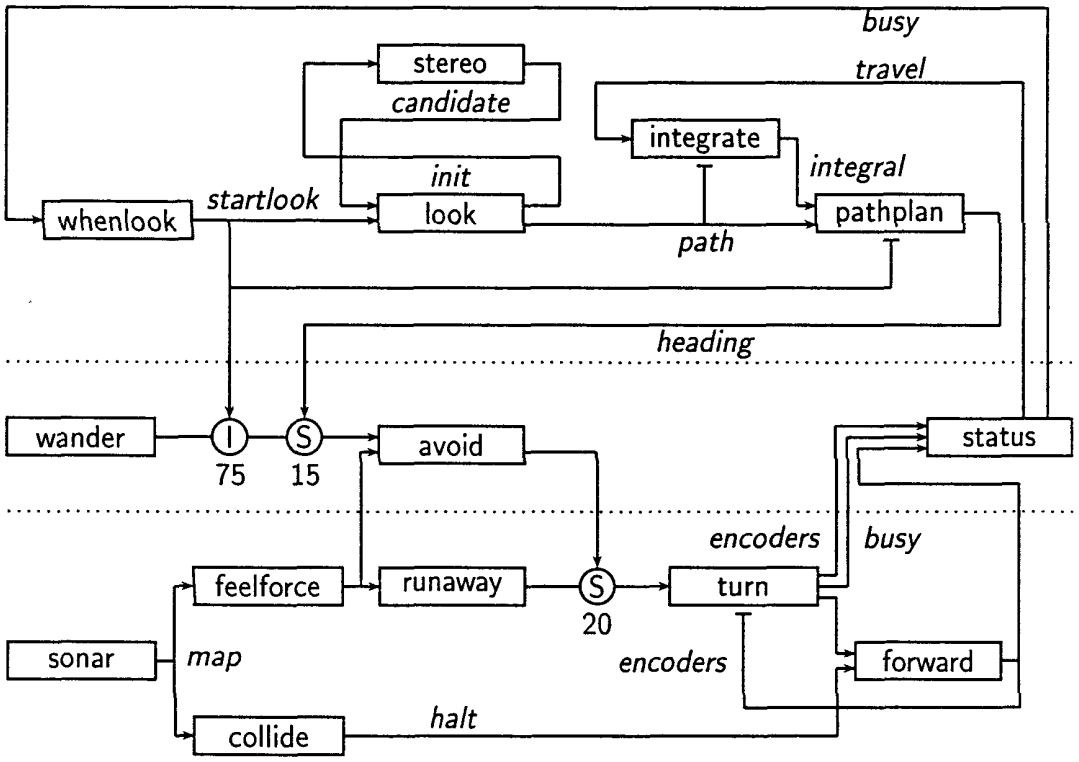

Figura 3.3: Diagrama de control del robot ALLEN, diseñado por Brooks

Las circunferencias con una "S" corresponden a los nodos supresores mientras que los que contienen una "I" son los inhibidores. En ambos casos los mensajes procedentes de la parte superior suprimen ó inhiben a los inferiores. En el siguiente punto se tratará con más detalle su funcionamiento.

Como puede observarse la arquitectura propuesta por Brooks está dividida en niveles de competencia, como ya se ha explicado, y estos niveles acogen a varios AFSMs que implementan las conductas necesarias y su interconexión, de modo que llevan a cabo la competencia de la capa.

La arquitectura que se usa en este proyecto no será la propuesta por Brooks, sino una variante propuesta por Connell y que se explica en el siguiente punto.

3.3.2. Adecuación de las conductas a una FPGA

Desde el punto de vista de la implementación sobre la FPGA no hay ningún factor que incline la balanza a favor de la arquitectura de Connell o la de Brooks, sin embargo se ha optado por la arquitectura de Connell por resultar más estructurada, y como se verá a continuación permite mayor potabilidad de las conductas creadas. A continuación se describe en detalle esta arquitectura.

En la figura 3.4 se muestra un ejemplo de la arquitectura de control de Connell, en ella se observan un conjunto de módulos (cajas), cada una de las cuales

implementa una parte de una conducta. Estas conductas se corresponden con actividades primarias como el evitar obstáculos ó la fototaxis[1] positiva. En este sentido, los módulos son elementos de control completos y autocontenidos, que usan información del entorno obtenida por sus sensores (denominadas **Pn)**, para generar las señales necesarias para los actuadores del robot (denominadas **An**).

Solamente un módulo podrá hacer uso de un actuador en un mismo tiempo, es decir, si varios módulos proporcionan comandos para el manejo de un actuador, solamente uno de ellos tendrá el control del mismo en un instante dado. Es importante ver que excepto a través de la red de supresión, cada módulo es independiente de los demás. No existen canales de comunicación directos entre los módulos ni un mecanismo de comunicación común del tipo pizarra[2]. Esto constituye una ventaja respecto al modelo de Brooks, ya que permite la realización y prueba de cada módulo, creación de una librería y posteriormente no será necesario tener un conocimiento exhaustivo del modo en que el módulo realiza su tarea, bastará con conocer su comportamiento.

De este modo se podrá hacer uso de la tremenda portabilidad de los diseños para la FPGA, ya que en un sistema secuencial como un µC pueden programarse funciones que no tengan problemas al ejecutarse aisladamente, pero creen conflictos de recursos o de tiempo de ejecución con otras. Esto no ocurre en una FPGA, puesto que todos los módulos se ejecutarán en paralelo, y solamente la falta de espacio en la misma impedirá la realización del diseño.

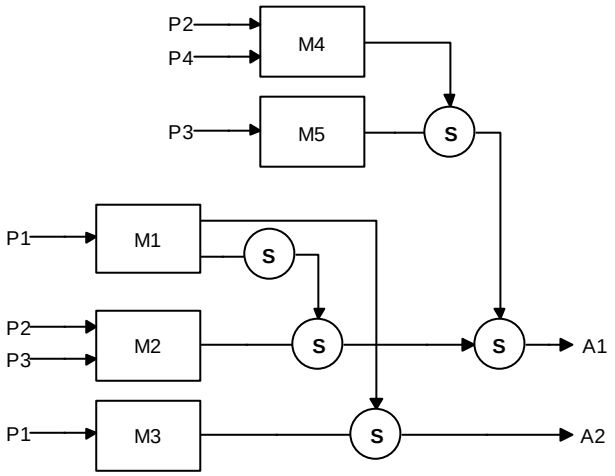

Figura 3.4: Ejemplo de arquitecura de control de Connell

En esta arquitectura las comunicaciones ocurren desde los sensores hasta los módulos, y de los módulos a los nodos supresores. Conceptualmente los mensajes que viajan a través de estos cables de conexión, que en cada instante de tiempo contendrán el estado actual de los sensores o de las ordenes para los actuadores. Sin

[1] Conducta de atracción o repulsión hacia fuentes luminosas dependiendo de si es positiva o negativa respectivamente.

[2] Elemento de programación dedicado a la comunicación entre tareas, en el cual se dejan mensajes para que cualquier otra tarea pueda leerlos.

embargo tanto Brooks como Connell precisaban de un artificio para la emulación de este comportamiento, debido a que la implementación se realizaba sobre varios µCs que se comunicaban entre sí mediante canales serie. Para ello los módulos emitían unos paquetes que contenían las acciones deseadas. Los receptores de estos paquetes mantenían esta información durante un corto espacio de tiempo, pasado el cual dichas acciones se consideraban caducadas. Por ello, los módulos debían emitir periódicamente estos paquetes aunque la acción fuese la misma. De este modo se emulaba el comportamiento continuo del sistema.

En el caso de la FPGA no ocurre este problema, ya que todos los módulos se ejecutan en paralelo de un modo real, estos pueden emitir continuamente sus ordenes hacia los nodos supresores.

Otro punto importante radica en al modo de arbitrar la lucha de los módulos por el control de los actuadores. En el caso de la arquitectura de Brooks existen 2 nodos, los inhibidores y los supresores, pero en el modelo de Connell solamente son necesarios los últimos, pues como se verá más adelante, la implementación de los módulos lo hace innecesario.

En los nodos supresores la salida del módulo dominante (la entrada superior del nodo) sobreescribe sus comandos de control a los del nodo dominado, consiguiendo que las conductas con un nivel de competencia más específico tomen el control sobre las conductas más generales, cuando las circunstancias lo exijan. La implementación de estos nodos tiene dos particularides en el caso de los experimentos de Connell. Por un lado el nodo tiene asociada una constante de tiempo, que es el tiempo en el cual caduca el último mensaje recibido, y por otro, los nodos no son implementados como módulos ni como hardware adicional, sino que son los AFSMs los que reciben los mensajes y suprimen sus salidas haciendo pasar las del módulo superior.

La implementación realizada sobre el sistema basado en la FPGA permite la realización de los nodos supresores como módulos específicos, por tanto no están incluidos en los AFSMs, sino que son elementos hardware programados en la FPGA, que tienen un funcionamiento propio y totalmente continuo (no discreto). El módulo creado es el siguiente:

Nodo supresor

Se parametriza el ancho de bus "**n**", siendo los bits n-1 a 0 quienes proporcionan la señal de control de los actuadores. Mientras que el bit n indica si:

- 1 → Conducta "ACTIVA"
- 0 → Conducta "INACTIVA"

- e1[n..0]➔ Bus de señales procedentes de la conducta que puede ser suprimida.
- e2[n..0]➔ Bus de señales procedentes de la conducta supresora.
- s[n..0]➔ Bus de salida con la conducta más prioritaria de las de entrada.

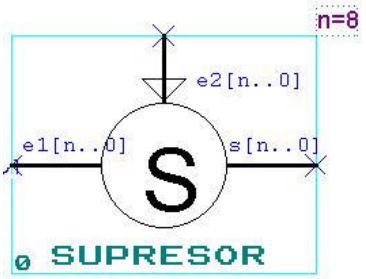

Fig 3.5:Símbolo del módulo supresor.

Otra diferencia significativa en la estructura interna de los módulos es que Brooks define cada módulo mediante las AFSMs. Sin embargo los de Connell (Figura 3.6) pueden considerarse, en un sentido amplio, *como reglas de producción.* La estructura interna de un módulo está formada por dos submódulos: el *predicado de aplicabilidad*, que determina cuando debe activarse el módulo, y la *función de transferencia*, que realiza las operaciones sobre los datos de entrada para obtener la salida.

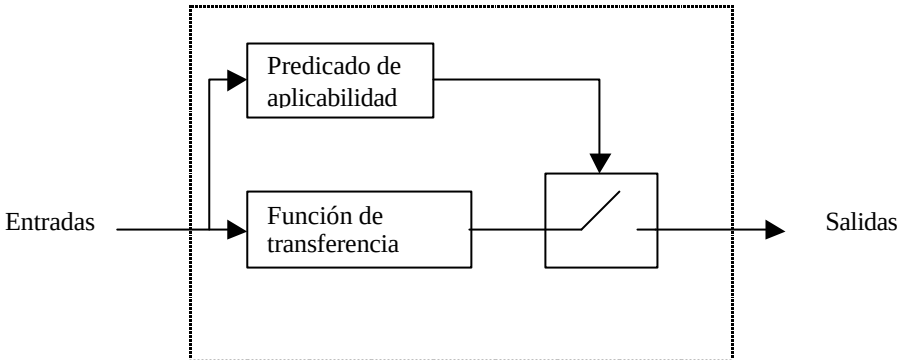

Fig. 3.6: Estructura interna de los tas conductas según Connell

Con este esquema los módulos solamente responden a los eventos que cumplan el predicado de aplicabilidad, en cuyo caso se aplica la función de transferencia a las entradas para generar las salidas.

El predicado de aplicabilidad aun puede descomponerse (fig. 3.7), ya que en general existe una causa para el inicio de la aplicabilidad y otra que indica el final ó conclusión de los objetivos del módulo. Así se dota al predicado de aplicabilidad de una memoria que permite que los módulos puedan responder de forma similar ante situaciones que se mantienen a lo largo del tiempo. En esta memoria se guarda la situación de actividad en la que se encuentra un módulo para alcanzar un objetivo. Una vez que se ha logrado el objetivo, el módulo se desactiva. Por tanto los módulos permanecen en una situación de reposo de la que salen cuando se produce alguna situación en el entorno, volviendo de nuevo a ésta una vez que la situación ha pasado o cuando ha transcurrido un periodo determinado.

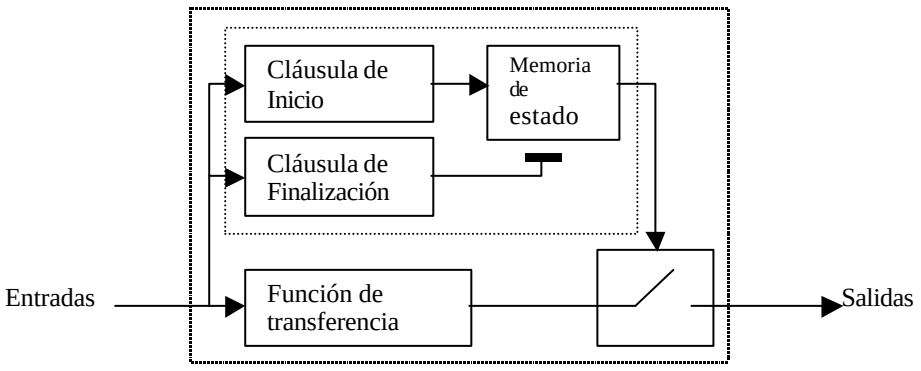

Fig. 3.7 Predicado de aplicabilidad según Connell

La implementación de estos módulos en la FPGA se realiza de forma muy sencilla. Por un lado se debe implementar la función de transferencia, que puede ser unas constantes, filtros digitales, autómatas de estados...etc. que en la mayor parte de los casos es fácil realizarlos en lenguajes como VHDL. Y por otro, se implementa el predicado de aplicabilidad. Este último puede codificarse como un autómata de estados en muchos casos, lo que resulta fácil de implementar mediante lenguajes HDL, mientras que en otros la cláusula de finalización suele ser el transcurso de un periodo de tiempo. Para este último caso se ha previsto de un módulo monoestable que realiza este cometido:

Monoestable

El siguiente módulo ha sido creado para realizar temporizaciones en el diseño. Al introducirlo, se configura el parámetro "**ciclos**" correspondiente al número de ciclos de reloj de durará disparado desde que la señal *DISPARO* vuelva a su estado normal. La señal *ACTIVO* se pondrá a "1" logico cuando la señal *DISPARO* pase a "0" lógico, volviendo *ACTIVO* a "0" lógico pasados los **ciclos** de reloj configurados después de volver *DISPARO* a "1" lógico. Para una mejor comprensión, vea la fig 3.9 que muestra una simulación del módulo con un reloj de periodo 200ns y 10 **ciclos** que durará disparado.

Fig 3.8:Símbolo del módulo monoestable.

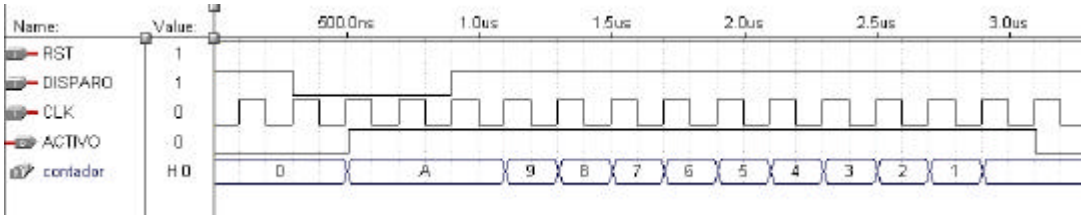

Fig 3.9 Diagrama de tiempos del módulo monoestable.

Esta es básicamente la arquitectura de control mediante conductas de Connell. Con estos conceptos e ideas se pueden desarrollar algoritmos inteligentes sobre este sistema. En el siguiente apartado se analizan los principales beneficios e inconvenientes del uso de una FPGA frente a un μC para la implementación de esta arquitectura.

3.3.3. FPGA vs µC

Sin lugar a dudas el uso de una FPGA para la realización de este proyecto fue una apuesta arriesgada, ya que no existen robots realizados de este modo. Sin embargo ese ha sido uno de los alicientes fundamentales. Como se ha ido viendo a lo largo de este capítulo, su uso aporta ciertas ventajas y se analizará puntualmente estos así como los inconvenientes:

Ventajas

- **Multitarea**

  Los controles reactivos precisan de un sistema multitarea para poder ser implementados de forma correcta, ya que si se realiza un único programa conteniendo a todas las conductas será muy difícil añadir nuevos niveles de competencia. Sin embargo en la FPGA ocurre todo lo contrario, ya que todas las conductas se realizan en paralelo.

- **Portabilidad de los módulos**

  Todos los módulos que se generan al programar la FPGA son encapsulados automáticamente al compilarlos, formando un componente. Este componente tiene una presencia gráfica con entradas y salidas, de modo que puede ser usado en cualquier diseño de forma sencilla, independientemente del lenguaje con el que fue creado. Además cada módulo corre en paralelo con los demás, por lo que nunca podrán dejar de funcionar dos módulos debido a la falta de recursos, como en el caso de los µC.

- **Facilidad de integración**

  Los módulos pueden ser programados en distintos lenguajes (VHDL, AHDL, esquemáticos...) según la conveniencia, pueden ser depurados por separado, y posteriormente integrarse con la garantía de que un mal funcionamiento se deberá a un fallo en el diseño, pero no a un fallo de integración.

- **Diseño visual con Hipervinculaciones**

  Los entornos de desarrollo proporcionados por los fabricantes de FPGAs siempre implementan el modo gráfico, en el cual se pueden realizar esquemáticos. Esto permite la realización de los módulos en lenguajes como VHDL, y la posterior integración del sistema completo en el modo gráfico, apareciendo los distintos módulos como cajas con entradas y salidas.

  Tal y como se plantea este sistema, en el que ya se han programado los nodos supresores, ciertas conductas y recursos para manejar sensores y actuadores, resulta muy sencillo unificar estos componentes en distintas capas de competencias para el manejo de un robot. Además, estos entornos cuentan con hipervinculación entre los componentes gráficos y su implementación. De este modo se puede acceder al código de un módulo simplemente haciendo doble click con el ratón.

Inconvenientes

- **Programación distinta a la habitual**

  Los lenguajes de descripción hardware poseen una sintaxis parecida a los lenguajes como C ANSI. sin embargo no tienen la posibilidad de realizar estructuras de datos, y los bucles son poco aconsejables. Además la metodología de creación de algoritmos es muy distinta a los lenguajes para µC . No obstante tras los primeros fracasos no es demasiado difícil acostumbrase a estos métodos.

- **Dificultad de implementación de algoritmos de alto nivel**

  Al no contar con estructuras de datos y la dificultad en el uso de ciertas sentencias de control como los bucles *for*, resulta engorroso y difícil realizar algoritmos orientados a la creación de mapas y conductas de muy alto nivel. Por otro lado las FPGAs son muy apropiadas para la implementación de redes neuronales y algoritmos genéticos.

- **Poco extendido**

  Como ya se ha comentado en varias ocasiones durante este proyecto, no existen precedentes de robots realizados con FPGAs como núcleo de su sistema de control. Por tanto aun está por probar su verdadera eficacia.

- **Precio**

  Las FPGAs son caras en comparación con los µC que pueden estar a la altura de sus capacidades computacionales. En el caso de este proyecto es utilizada una FPGA de 10.000 puertas, que puede compararse en potencia a un µC de 8 bits, sin embargo su precio puede duplicar y hasta triplicar el de estos.

3.4. Expansión de posibilidades µC + FPGA

Hasta ahora se ha considerado que el núcleo del sistema está formado únicamente por la FPGA, siendo el µC que la acompaña un sistema para gestionar sus configuraciones y prestar ayuda a la hora de depurar los módulos en tiempo real con los sensores y actuadores conectados.

Sin embargo el µC puede acoger aplicaciones de usuario que hagan uso de los módulos de comunicación entre la FPGA y el µC. Mediante este método se

puede solventar el problema más importante del a FPGA, que consistía en la dificultad de implementar algoritmos para la realización de mapas, o en los que se manejen estructuras de datos. Así se puede realizar las conductas de bajo nivel sobre la FPGA, mientras que las conductas especializadas, que requerirán de cálculos matemáticos más complejos, podrán ser implementadas sobre el μC en lenguaje C. La lectura de sensores así como el envío de las ordenes a los actuadores puede realizarse de forma sencilla mediante los módulos de comunicaciones. Por otro lado la res de supresión se debería seguir implementando sobre la FPGA, para que se ejecute en paralelo y sea fácilmente manejable.

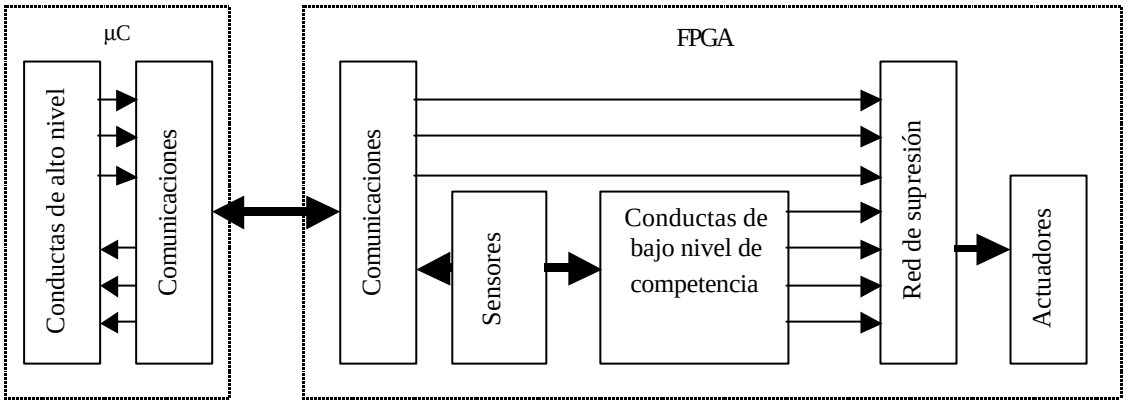

Fig. 3.10

3.5 Bibliografía

1. "*Minimalist Movile Robotics"* J.H.Connell. Academic Press, Boston, 1990.
2. "*Inteligence whitout representation"* R.A. Brooks. Artificial intelligence. 1991.*
3. "*Robots Móviles*" Javier de Lope Asiaín. EUI (UPM). 2000.
4. *"Elephants don't play chess*" R.A. Brooks. Designing Autonomous Robots, páginas 3-15. MIT Press, Cambridge, Massachusetts, 1991.*
5. "*Planificación basada en Percepción Activa para la Navegación de un Robot Movil"* Miguel Schneider Fontán. Instituto de automática industrial, Madrid, 1996.*

* Incluido en el CD