Implementación de un agente inteligente BDI sobre hardware reconfigurable

Pontificia Universidad Javeriana

Facultad de Ingeniería

Departamento Ingeniería Electrónica

Héctor Andrés Hernández Valderrama

Directora: Alejandra González P.H.D

Trabajo de grado para optar por el título de Ingeniero Electrónico

Bogotá, Noviembre 2016

TABLA DE CONTENIDO

1. INTRODUCCIÓN

La automatización de procesos y tareas, se ha consolidado como uno de los pilares del desarrollo tecnológico desde que la electrónica se erige como la ciencia que busca entender y aplicar los principios electromagnéticos desde su descubrimiento a finales del siglo XIX.

Con este propósito han surgido un sin número de enfoques que buscan especializar las implementaciones tecnológicas a medida que estás evolucionan, para mejorar la calidad de vida de los seres humanos que hacemos uso de estas. Encontramos automatismo en complejos procesos de fabricación industrial, en los medios de transporte, comunicación y hasta en tareas diarias como programar una alarma despertador. La complejidad o simplicidad de su configuración depende en gran medida de su funcionalidad y del usuario objetivo que vaya a hacer uso de esta, buscando siempre que la interacción se haga de la forma más natural y sencilla posible. Sin embargo, por más fácil que se intente implementar la interfaz, siempre requerirá algún tipo de conocimiento técnico para garantizar que su funcionamiento se desarrolle de la forma esperada: incluso programar el reloj despertador, debe tener en cuenta en su configuración, el día y rango horario (AM/PM) que se requeriere. A raíz de los requerimientos técnicos que implican este tipo de interacción con los procesos de automatización, surge un enfoque que intenta hacer cada vez más natural la ejecución de tareas por parte de sistemas automáticos, buscando incluso que estas se desarrollen de forma tan natural para los humanos que sean apenas perceptibles. En esta línea se espera que los sistemas solucionen problemas y tomen decisiones emulando la forma en que lo hace un ser humano. La inteligencia artificial surge como una rama en busca de este objetivo, estudiando y aplicando los patrones de decisión que usaría un humano para solucionar un problema específico. Esta solución se desarrolla de la mejor forma posible con los recursos existentes y teniendo en cuenta una serie de prioridades en la ejecución de un plan de acción.

Surgen en la década de los noventa, diferentes aproximaciones para generar un modelo adecuado que se acerque a la emulación del comportamiento humano en ese aspecto y así como la teoría de agentes, sucesora del paradigma de objetos, estructura bases para la solución de problemas por parte de sistemas con un grado de autonomía (sin interacción humana directa) y racionalidad, el paradigma BDI se constituye como un modelo que da respuesta a los requerimientos de sistemas complejos, con un carácter hibrido (reactivo-deliberativo) y bio-inspirado, que distribuye el proceso en 3 pilares fundamentales que agrupan los factores más influyentes en el proceso, haciendo énfasis en las prioridades que tiene en cuenta el discernimiento humano,:

1. Estado Actual, tanto de sí mismo como del ambiente que lo rodea; *Beliefs* (Creencias).
2. Qué se desea conseguir o lograr, cuales son los objetivos y que tan importantes son uno con respecto a otro: *Desires* (Deseos).
3. Cómo y cuándo puedo conseguir dichos objetivos, con los recursos de los que dispongo: *Intetions* (Intensiones).

En términos de implementación, se pueden considerar diversas plataformas tecnológicas, lo cual implica que la escogencia de la tecnología adecuada para un desarrollo en específico, es en sí un reto. Los procesadores embebidos, con una unidad central de procesamiento, complementados con memorias y controladores de periféricos, hacen que el software sea el dominio indicado para materializar estos objetivos por medio de procesos de descripción a través de lenguajes de programación. La tecnología pone a disposición una gran cantidad de recursos que facilitan la interacción con usuarios y medios externos y hace que su implementación llegue a ser intuitiva y por esta razón se han erigido como los dispositivos básicos para implementaciones multipropósito. Sin embargo, estas herramientas entregan un número tan elevado de posibles recursos para implementación, distribuidos en funcionalidades y aplicaciones dedicadas, especializadas y específicas, que es prácticamente imposible diseñar un sistema que haga uso de todas los recursos que se ponen a disposición del diseñador en plataformas como los microcontroladores. El "desperdicio" de recursos, lleva inevitablemente a que el rendimiento de las soluciones implementadas se vea limitado a las verdaderas capacidades de los componentes usados y no a las requeridas o esperadas para la solución en específico.

Teniendo esto en cuenta, vale la pena revisar las características de las plataformas para implementación en Hardware, las cuales poseen una cantidad específica de recursos informáticos, que pueden ser estructurados según las necesidades propias de la funcionalidad esperada. Incluso gracias a las tecnologías existentes, es posible hacer uso de las mismas capacidades base de forma repetida, para cubrir diferentes funcionalidades según los requerimientos de diseño. Estos elementos configurables por el usuario, pueden procesar información a grandes velocidades, adicionalmente cuentan con considerables cantidades de componentes lógicos que soportan protocolos estándar de comunicación. Por su capacidad de ser programados directamente por el usuario, reciben su nombre por sus siglas en inglés: FPGA (Field Programable Gate Array).

El presente trabajo de grado busca evidenciar las ventajas de la implementación del modelo de agentes tomando la estructura planteada por el paradigma BDI sobre una plataforma FPGA, tomando como base los estudios, resultados e implementaciones realizadas en trabajos precedentes, sobre plataformas de microcontroladores en configuración por software.

Con este fin, se toma una de las aplicaciones actualmente estudiadas por el grupo de investigación SIRP que busca la implementación del concepto "Teatro Robótico", y se diseña un sistema que permite la ejecución de una serie de instrucciones, realizando un proceso básico de toma de decisiones teniendo en cuenta características simples del mundo que lo rodea y una priorización de metas previamente estructurada. Esto, en términos prácticos permitirá a un usuario indicarle al agente un guion a recitar, el cual será ejecutado basado en una estructura jerárquica de prioridades y una identificación propia del tipo de agente. Una vez diseñada su funcionalidad por medio de diagramas de flujo, se realiza su estructuración por medio de lenguajes HDL (Hardware Description Language), retomando el original AHPL (A hardware Programming Language) planteado por Frederick J Hill y Gerald R. Peterson en 1987, para un primer acercamiento con el dominio hardware, definiendo unidades de almacenamiento, control secuencial y lógica combinatoria, finalizando con una implementación VHDL.

Para la comprobación de los desarrollos, se elige un fabricante líder en tecnología FPGA: Altera, y por medio de sus herramientas de descripción, simulación y configuración, se realizan diferentes ciclos de simulación y pruebas para corroborar el funcionamiento esperado de la implementación. Es importante resaltar que el alcance del proyecto, contempla el agente como módulo de procesamiento y la generación de señales hacia módulos independientes que manejarán la integración con periféricos externos. En sí, abarca la implementación de la inteligencia que a futuro podrá ser usada para reemplazar el *core* de procesamiento en proyectos paralelos, que están siendo desarrollados por estudiantes de la facultad de ingeniería, departamentos de electrónica y de sistemas, quienes realizaron la implementación de un prototipo robótico modular que podría recibir las señales e interpretarlas para convertirlas en acciones de facto.

2. MARCO TEÓRICO

Los agentes surgen desde el inicio de la rama de estudio conocida como inteligencia artificial, en la cual se pretende dar un grado cada vez mayor de autonomía a la tecnología desarrollada, para actuar de forma similar a la que lo haría un ser humano. El grado de discernimiento que debe tener cualquier desarrollo considerado inteligente es en sí un punto difícil de definir, pero existen diferentes aproximaciones que permiten reconocer o conceptualizar cuando un desarrollo es inteligente

2.1 TEORÍA DE AGENTES E INTELIGENCIA ARTIFICIAL

. Coloquialmente el término "inteligencia artificial" se aplica cuando una máquina imita las funciones "cognitivas" que los humanos asocian con otras mentes humanas, como por ejemplo: "aprender" y "resolver problemas" [RUSSELL, 2009] Actualmente, la inteligencia artificial está involucrada en ramas como la Economía, astronomía, ingeniería y medicina.

En su publicación: *"Artificial Intelligence a Modern Aproach",* Stuart Russell y Peter Norvig identifican los siguientes tipos de inteligencia artificial:

- Sistemas que piensan como humanos: intentan emular el pensamiento humano. Pretenden automatizar actividades relacionadas al pensamiento humano como la resolución de problemas, la toma de decisiones y el aprendizaje.
- Sistemas que actúan como humanos: intentan actuar o imitar el comportamiento humano. En este enfoque la robótica cubre el estudio de cómo lograr que los computadores realicen tareas que hasta el momento los humanos realizan mejor.
- Sistemas que piensan racionalmente: con lógica imitan o emulan el pensamiento lógico racional del ser humano.
- Sistemas que actúan racionalmente: intentan emular de forma racional el comportamiento humano, teniendo conductas inteligentes. En esta rama encontramos a los Agentes inteligentes.

El concepto de agente, que deriva del latín "agere" (hacer), describe una abstracción de software, un programa que actúa para un usuario (humano o no), una idea o concepto similar al de los métodos, funciones y objetos, planteada en la programación orientada a objetos. El concepto provee una forma conveniente y poderosa de describir una compleja entidad de software, que es capaz de actuar con algún grado de autonomía, para cumplir tareas en representación o con los mismos objetivos de personas. A diferencia de los objetos (definidos por métodos y atributos), un agente es definido por su propio comportamiento. [Diccionario de Informática, 2010]. Aunque no existe una única definición universalmente aceptada, un agente por defecto debe poder percibir su entorno por medio de "sensores" y realizar modificaciones sobre el por medio de "actuadores" basado en la evaluación de técnicas de resolución de problemas. Cualquier proceso computacional dirigido por un objetivo, capaz de interaccionar con su entorno de forma flexible y robusta puede considerarse como un agente, el cual basado en estos principios, tendrá las siguientes características [Iglesias 1998]:

- Autonomía: capacidad de actuar sin intervención humana directa o de otros agentes.
- Sociabilidad: capacidad de interaccionar con otros agentes, utilizando como medio algún lenguaje de comunicación entre agentes.
- Reactividad: un agente está inmerso en un determinado entorno (habitat), del que percibe estímulos y ante los que debe reaccionar en un tiempo preestablecido.
- Iniciativa: un agente no sólo debe reaccionar a los cambios que se produzcan en su entorno, sino que incluye un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe de satisfacer.
- Movilidad: habilidad de trasladarse en una red de comunicación informática.

Las actuaciones de los agentes deben buscar la consecución de uno o más objetivos, en función de los cuales ejecutará las acciones adecuadas para alcanzarlo. De esta forma todo agente debe tener un objetivo o una función, sin embargo la forma en que se consigue dicho objetivo diferencia a un agente inteligente o racional, el cual debe maximizar su rendimiento, según la secuencia de percepciones que ha observado hasta el momento. De esta forma podríamos definir que los agentes inteligentes son agentes software que pueden funcionar fiablemente en un entorno rápidamente cambiante e impredecible. [Wooldridge 1995]

"If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action" - Principio de Racionalidad (Allen Newell (1982). The knowledge level. Artificial Intelligence 18: 87-127)

El comportamiento del agente inteligente para conseguir sus objetivos, se presenta como una serie de pasos: realizar monitoreo de eventos, razonar sobre los mismos, inferir posibles acciones, y finalmente ejecutar la acción. Teniendo estos conceptos en cuenta, Wooldridge y Jennings [Wooldridge 1995], definen la existencia de 2 "tipos" de agentes según las propiedades que lo caracterizan. Esta noción identifica un agente débil que cumple con las 5 características anteriormente enunciadas, y un agente fuerte que además de cumplir con estas, también posee una o más de las siguientes:

- Nociones mentales: Tiene creencias, deseos e intenciones.
- Racionalidad: Realiza acciones a fin de mejorar sus objetivos.

- Veracidad: No es capaz de comunicar información falsa a propósito.
- Adaptabilidad: Es capaz de aprender de su experiencia.

2.2 ARQUITECTURA DE AGENTES

Es claro entonces que aunque no existe un consenso universal al respecto, se tiene una amplia gama de cualidades que nos permiten estructurar una implementación basada en agentes, para desarrollar sistemas con inteligencia artificial, que busque conseguir un objetivo u objetivos específicos para cubrir una funcionalidad dada. La metodología particular que se debe seguir para lograr una satisfactoria implementación basada en la teoría de agentes, se conoce como arquitectura de agentes, en la cual se especifica cómo el agente puede ser descompuesto en un conjunto de módulos componentes y la forma en que estos interactúan. El conjunto total de módulos y sus interacciones deben proveer una respuesta a la pregunta de cómo el dato monitoreado, y el estado interno del agente determinan las acciones y estados internos futuros [Wooldridge 1995].

En este sentido Wooldridge y Jennings proponen una clasificación para las posibles formas que se pueden estructurar las arquitecturas para la implementación de agentes, basados en el eje de acción del mismo. Los autores identifican así 3 categorías:

- Arquitecturas deliberativas: contienen un mundo representado explícitamente y un modelo lógico del mismo, en el cual las decisiones (por ejemplo acerca de las acciones a realizar) son hechas por medio de un razonamiento lógico, basado en concordancia con patrones y manipulación simbólica.
- Arquitecturas reactivas: no incluye ningún modelo central del mundo y no utiliza razonamiento simbólico complejo.
- Arquitecturas híbridas: une los enfoques deliberativos y reactivos, separando las capas de razonamiento y mapeo de acciones directas.

Wooldridge y Jennings, sugieren entonces, que el enfoque adecuado para la construcción de un agente, no debe buscar un enfoque completamente deliberativo, ni completamente, es decir un enfoque hibrido, buscando construir un agente compuesto de dos subsistemas: uno deliberativo, que contenga un módulo simbólico del mundo, desarrolle planes y tome decisiones de la manera propuesta por la inteligencia artificial simbólica; y uno reactivo, que sea capaz de reaccionar a eventos que ocurren en el ambiente sin necesidad de un razonamiento complejo. En este enfoque suele dársele al componente reactivo, cierto grado de precedencia sobre el deliberativo, buscando proveer una pronta respuesta a eventos importantes en el entorno.

En este tipo de arquitectura, los sistemas de control del agente se organizan de tal forma que las capas más altas procesan información de mayor nivel de abstracción y manejar los objetivos a largo plazo, permitiendo que las capas inferiores puedan procesar datos de entrada y manejar directamente los actuadores de salida.

Un acercamiento del modelo deliberativo deseado para el desarrollo del presente trabajo, está basado en la arquitectura BDI, el cual complementado con una capa reactiva de arquitectura composicional, logran cubrir el enfoque Hibrido propuesto por Wooldridge y Jennings, dentro de la implementación del agente inteligente.

2.3 BDI

La arquitectura BDI, ve al sistema como un agente racional con ciertas actitudes mentales presentes en los seres humanos, haciendo énfasis en las creencias, deseos e intenciones, representando cada una respectivamente, los estados de información (*Beliefs*), motivación (*Desires*) y deliberativo (*Intentions*) del agente. Estas actitudes mentales definen el comportamiento del sistema y son de vital importancia para conseguir el mejor desempeño posible cuando la deliberación está sujeta a recursos limitados.

En este tipo de arquitectura, las creencias de un agente representan el conocimiento del mismo, con respecto a si mismo, de su historia y su entorno. Información que el agente asume como cierta y debe tener en cuenta al momento discernir para tomar una decisión. Los deseos por su parte proveen al agente la motivación para actuar, sus metas a cumplir. Los objetivos que constituyen los deseos pueden llegar a ser contradictorios, forzando a que el sistema pueda elegir qué objetivo alcanzar primero. En este punto aparecen las intenciones que pueden ser consideradas como un conjunto de planes para lograr los objetivos que constituyen los deseos. Como se puede observar, BDI lleva a cabo procesos de deliberación complejos que lo hacen un esquema altamente deliberativo. [Wooldridge 1995]

Por otra parte la arquitectura BDI, se relaciona con un esquema reactivo en las que todas las funcionalidades son diseñadas como una serie de componentes estructurados jerárquicamente, que interactúan, basados en tareas. Las tareas se caracterizan en términos de sus entradas, sus salidas y su relación con otras tareas. La interacción y cooperación entre componentes y usuarios se especifica en términos de intercambio y secuencialidad de información, y dependencias de control. Los componentes en sí pueden ser de cualquier complejidad y pueden realizar cualquier función de dominio. [A. Georgeff]

En esta arquitectura, se definen 5 elementos a ser descritos y modelados explícitamente:

- Descomposición de tareas: Se puede especificar un conjunto de sub tareas por cada tarea en una jerarquía;
- Intercambio de información: se especifica como vínculos de información entre componentes. Cada vinculo de información, relaciona la salida de un componente con la entrada de otro;
- Secuenciamiento de tareas: se modela explícitamente dentro de los componentes, como conocimiento de control de tarea. Este incluye no solo el conocimiento de la tarea a ser activada, cuando y como, sino también conocimiento sobre información de control asociado con la activación de la tarea y el porcentaje de esfuerzo que puede permitirse para lograr un objetivo dado.
- Delegación de sub tareas: el proceso de decisión de que componente puede desarrollar una tarea de la mejor manera. Aun cuando la tarea sea modelada como un todo en el proceso de diseño, la delegación puede permitir un mejor uso de recursos para llevar a cabo eficientemente la tarea en general.
- Estructuras de conocimiento: conocimiento claro del significado de los conceptos utilizados. Los conceptos se requieren para identificar objetos distinguibles en un dominio, pero además para expresar los métodos y estrategias empleadas para realizar una tarea.

Basado en estas definiciones y teniendo en cuenta el objetivo de lograr una arquitectura que permita aprovechas de la mejor manera los esquemas deliberativo y reactivo, el modelo BDI está basado en un análisis de las tareas desarrolladas por un agente. Dicho análisis, resulta en una composición (jerárquica) de tareas, que es la base para un modelo composicional. En esa, el modelo establece las siguientes tareas necesarias para el agente:

- Control de sus propios procesos.
- Cumplimiento de sus tareas propias.
- Manejo de su interacción con el mundo.
- Manejo de su comunicación con otros agentes.
- Mantenimiento de información sobre el mundo.
- Mantenimiento de información sobre otros agentes.

En la arquitectura BDI, cada una de las tareas anteriores, es refinada descomponiéndolas en los 3 ejes propios BDI: Creencias, Deseos e Intenciones del Agente. Entonces la jerarquía de tareas presentada en el modelo genérico se extiende agregando estos 3 componentes haciendo que dentro del "control de sus propios procesos" mencionada en el modelo genérico, se determinen además las creencias, deseos e intenciones, como sub tareas del proceso.

Se puede notar entonces que existe una gran diversidad de ideas en cuanto a la forma de modelar la arquitectura de un agente. A partir de la esencia de cada una se pretende analizar aquella o aquellas que contemplan las características presentes en la definición de agente que se ha adoptado.

En una arquitectura para agentes reactivos, el interés se centra en el modelado de la reactividad, la autonomía y la interacción con el ambiente, a través de sensores y actuadores, dejando de lado las características de inteligencia a partir de una representación interna de los conocimientos. Por tanto con una arquitectura de este tipo podría ser imposible capturar todas las características que se consideran de importancia para el modelado de agentes.

Algo más interesante se presenta en la arquitectura para agentes deliberativos, en cuyas diferentes propuestas, además de poder modelar las características de interacción con el ambiente y la autonomía, aparecen el componente pro-actividad, y una característica muy importante: las nociones mentales. Asociados con este último aparecen la racionalidad y el aprendizaje.

Con una arquitectura para agentes híbridos se puede integrar todas las características de ambas arquitecturas (reactiva y deliberativa). El resultado es una arquitectura más específica, que modela en forma explícita ciertas características como: descomposición de tareas, delegación de tareas, estructuras mentales, etc. La arquitectura BDI, abarca de manera más exhaustiva las características presentes en la definición de Wooldridge y Jennings.

Una vez conocidos los aspectos a tener en cuenta para la implementación de un agente inteligente BDI, se entra en las definiciones necesarias para su implementación. En este sentido encontrámos un gran número de herramientas que permiten llevar a cabo esta tarea. Sin embargo siendo el objetivo del presente trabajo, demostrar las ventajas de realizar dicha implementación en el dominio Hardware, se hace uso de lenguajes de descripción de hardware HDL en el proceso de diseño, implementación y configuración del agente.

3. OBJETIVO DEL PROYECTO

Mediante la implementación en dominio hardware de un agente inteligente basado en la arquitectura BDI, se espera consolidar el conocimiento adquirido por diversos trabajos y estudios previos en este aspecto, llevados a cabo por el grupo de investigación SIRP de la Universidad Javeriana. Con esto en mente, se evaluaron diferentes opciones que podrían favorecer el objetivo de implementar un agente BDI únicamente en dominio Hardware, teniendo en cuenta las limitaciones de tiempo y presupuesto que implican su investigación, diseño y desarrollo. En este sentido, el alcance del proyecto, cubre la implementación del módulo de razonamiento y aplicación de las capacidades del paradigma BDI, abarcando su diseño estructural, el modelo para, y su implementación como tal, la verificación y validación del diseño estructurado en el contexto de una aplicación del grupo SIRP, las evaluaciones de viabilidad para incluir o descartar modelos que puedan resultar útiles en futuros proyectos y en este mismo sentido, definiciones de integración con otros agentes o módulos de interacción externos. El proyecto no pretende generar un agente completamente funcional, ni su implementación en una plataforma o desarrollo ya existente. Aunque se plantea un protocolo de pruebas en el cual por medio de circuitos básicos se pueda evidenciar el comportamiento del agente, los montajes circuitales resultantes, no hacen parte como tal del diseño y se estructuran como una ayuda netamente académica.

3.1 AGENTE ACTOR

Como parte de las funcionalidades estudiadas para la implementación del agente BDI, se encontró la posibilidad de enfocar este esfuerzo para favorecer una rama de estudio del grupo de investigación, que busca la implementación de sistemas para desarrollar el concepto de teatro robótico. Este, tiene como objetivo identificar y analizar todas las posibilidades que brinda la tecnología, para desarrollar sistemas capaces de ejecutar una serie de instrucciones teniendo en cuenta parámetros de inteligencia artificial, como el conocimiento de su entorno y de sí mismo, para llevar a cabo dichas instrucciones de una manera "personalizada" según parámetros previamente establecidos por un usuario.

Se plantea entonces la implementación de un agente inteligente cuya principal funcionalidad, será ejecutar una serie de instrucciones (guion), teniendo en cuenta los siguientes parámetros que conformarán el grupo de Creencias iniciales que tendrá el agente sobre sí mismo y su entorno:

- Parámetro de identidad, que lo caracterice como individuo. Se plantean opciones de personalidades, optimistas, pesimistas o balanceadas y una característica de género: hombre o mujer.
- Estado actual de sí mismo dentro de su entorno: la posición inicial en la que se encuentra al no tener un registro previo de si mismo.
- Estado actual del entorno en el que se encuentra, mediante límites para realizar su desplazamiento.

Cada una de estas características, tendrá una incidencia en el proceso deliberativo que deberá llevar a cabo, durante la ejecución de cada una de las instrucciones que el usuario le indique y debe llevar por

parte del agente, una respuesta característica para cada una. Es decir, teniendo en cuenta sus parámetros perceptivos, la respuesta puede cambiar, acorde a los mismos.

Se define entonces un grupo de 7 capacidades que busca cubrir las funcionalidades más relevantes identificadas para un agente inteligente en un ámbito de teatro robótico:

- Dos capacidades de configuración en línea y establecimiento de parámetros por medio de las cuales el usuario podrá interactuar con el agente, para indicarle información importante sobre si mismo y sobre sus objetivos a cumplir. Estas son la descripción inicial de si mismo y de su entorno y la capacidad de entregarle un guion de instrucciones.
- Una capacidad de ejecución automática que le permitirá recitar el guion entregado previamente por el usuario.
- Dos capacidades con inferencia netamente reactiva para manejar los actuadores propuestos de emisión de luz y sonido.
- Dos capacidades con inferencia deliberativa que requerirán una etapa de discernimiento. Se logra de esta manera, estructurar la arquitectura hibrida buscada, con un balance entre procesos deliberativos y reactivos, entre los cuales se segmentan las tareas a ejecutar.

El agente actor tendrá la capacidad de recitar y expresar mediante estas capacidades, una serie definida de instrucciones, que en conjunto se asemejan claramente con la interpretación de escenas por parte de un actor, basado en un guion.

3.1.1 ENFOQUE REACTIVO:

Esperando tener la posibilidad por parte del usuario de solicitar la ejecución de capacidades básicas que cualquier sistema reactivo podría ejecutar, se implementan dos procesos básicos que no tienen una deliberación directa por parte del agente, y que pueden ser llevados a cabo como transferencias de registros únicamente:

- ENCENDER LED: Permite al agente indicar en sus actuadores los parámetros solicitados por un usuario para el encendido de LED’s en diferentes colores y por diferentes periodos de tiempo.

- EMITIR SONIDO: Permite al agente indicar en sus actuadores los parámetros solicitados por un usuario para la emisión de un sonido en diferentes tonalidades y por diferentes periodos de tiempo.

3.1.2 ENFOQUE DELIBERATIVO:

Complementando el enfoque del agente propuesto, con procesos deliberativos que tengan en cuenta las características propias de un agente inteligente, se estructuran 2 capacidades basadas directamente en el estado actual del agente en cuanto a la posición en la que se encuentra, el nivel de carga de su fuente de poder y un parámetro de identidad indicado por el usuario. El proceso deliberativo se ejecuta de forma concurrente dentro del agente y se toma la decisión basado en una jerarquización de prioridades:

- DESPLAZAMIENTO HACIA UNA POSICIÓN: permite al agente moverse desde su posición actual (De la cual tiene conocimiento) hasta una coordenada indicada por el usuario, evaluando si

dicha coordenada se encuentra dentro del espacio en el que se le es permitido moverse, y esquivando agentes externos que puedan encontrarse en el camino.

- EXPRESAR EMOCIÓN: permite al agente expresar una emoción especifica dentro de una serie de posibilidades definidas para tal fin, o expresar su "estado emocional" actual, el cual define según su nivel de batería y un parámetro de identidad parametrizado por el usuario. La emoción definida se expresa finalmente como un conjunto de órdenes en los actuadores, para emitir un patrón de luz y sonido previamente establecido.

3.1.3 CAPACIDADES DE CONFIGURACIÓN

Con el fin de dar a conocer al agente los parámetros base que guiarán sus procesos deliberativos, se estructuran 3 instrucciones que configuran en la capa de almacenamiento dicha información:

- CONFIGURACIÓN DEL MUNDO: permite al usuario indicar al agente 3 parámetros de configuración inicial que le dan el conocimiento inicial de si mismo y su entorno. Deben ser las primeras instrucciones dadas al agente para garantizar que pueda cumplir con las demás capacidades de la forma esperada. Estas son:
    - Los límites que tendrán sus movimientos, teniendo en cuenta que el entorno en el que se encuentra, se define como una matriz de X posiciones en el eje horizontal y Y posiciones en el eje vertical.
    - La posición inicial del agente, en el momento en que se realiza la parametrización.
    - Su parámetro de "identidad", elegido entre 3 opciones emocionales y un indicador de género: Masculino, Femenino.
- ENTREGA DEL GUION: Permite almacenar en memoria, una seria de instrucciones con un numero de **"escena"** definido, para una posterior ejecución secuencial.

3.1.4 CAPACIDAD DE EJECUCIÓN AUTOMÁTICA

Finamente se le entrega al agente la capacidad de realizar el control de sus acciones, basado en una escena previamente configurada en su memoria, mediante la entrega del guion. Aunque el proceso es ejecutado de forma secuencial, sigue manteniendo el carácter deliberativo en cuanto evalúa siempre su estado actual y toma la decisión de continuar o no con la ejecución del guion basado en este estado.

- RECITAR: mediante el número de escena que el usuario desea que el agente interprete, se le indica al agente que inicie con la ejecución secuencial del grupo de instrucciones almacenadas para dicha escena.

Bajo este grupo de funcionalidades, se estructura un grupo de componentes lógicos y de almacenamiento a ser implementados en una plataforma de desarrollo FPGA, elegida por su facilidad de manejo, alto respaldo técnico y profesional para su configuración, confiabilidad y robustez, teniendo en cuenta incluso su pequeño tamaño para una futura implementación robótica. En este sentido se elige una de las tarjeta de desarrollo del fabricante Altera, que disponen la FPGA Cyclone 2 junto a un conjunto de pines que facilitan la interacción con los puertos entrada y salida y punto de alimentación DC a 5V, cuyas especificaciones técnicas y diagrama de pines se entregan como parte del presente trabajo en el ANEXO 1.

4. DESARROLLO

Con el fin de cubrir las funcionalidades propuestas de interpretación para el modelo BDI en un contexto de teatro robótico, el agente inteligente se estructura con diferentes componentes que manejan el flujo de información de entrada, salida y el procesamiento de la misma internamente.

4.1 ENFOQUE BDI

Una vez identificadas y reconocidas todas las características que debe tener el agente actor y aunque para su diseño desde un inicio se tuvo en cuenta el enfoque BDI, es valioso realizar un acercamiento a detalle para identificar como el grupo de capacidades planteadas, cubre efectivamente este tipo de arquitectura. Para este efecto, se toman las nociones identificadas para un modelo de agente BDI, descritas a fondo por Alejandra González en su tesis doctoral "DISEÑO DE SISTEMAS EMBEBIDOS COMPLEJOS A PARTIR DE AGENTES BDI HÍBRIDOS CON MIGRACIÓN DE DOMINIO", investigación base del presente trabajo. Recordando los pilares de este paradigma, que se enuncian y describen a detalle en el mencionado documento, sabemos que al referirnos a creencias podemos generalizar hablando de un "estado de información" del agente, en el caso de los deseos, un "estado de motivación" y finalmente las intenciones como un "estado deliberativo" o estado proactivo [Dragoni, 2008]. Se tienen entonces las respectivas correspondencias encontradas en la implementación del agente.

4.1.1 BELIEFS

Desde el punto de vista teórico, las creencias son elementos del estado mental de un agente. La percepción del medio ambiente a través de sensores y la comunicación con el exterior a través de canales, hace de las creencias componentes dinámicos de la arquitectura BDI. Un agente posee creencias que corresponden a sentencias del mundo y de sí mismo. Desde el punto de vista del propio agente, dichas sentencias son verdaderas aún si estas son falsas en la realidad; es decir, no hay discriminación del valor de verdad [Thangarajah, 2012]. Durante el ciclo de vida de un agente, las creencias son complementadas y modificadas por varios eventos relacionados con la activación recurrente de la secuencia de percepción

Llevado a nuestro agente actor, estas creencias se evidencian de tres formas:

1. Parametrizaciones iniciales realizadas por el usuario sobre el agente, generando un primer grupo de conocimiento fáctico de sí mismo y del mundo que lo rodea. Específicamente hablando, mediante la capacidad WORLD se da al agente el conocimiento sobre:
    i. Los límites de su entorno
    ii. Su Posición Actual en el mundo
    iii. Su Identidad
2. Parámetros generados por los sensores externos a través de SENBUS, que le indican al agente el estado actual del entorno, y que una vez almacenados en el registro Reg_SEN, le permiten identificar en su proceso de discernimiento, la presencia de agentes externos en su periferia inmediata y el estado actual de su única fuente de energía de la cual depende su supervivencia.
3. El conocimiento de su posición mediante lógica aritmética simple, teniendo en cuenta posición actual y final dentro del mundo. Esto es, mediante la sentencia MOVE, el agente identifica si su movimiento es posible, y una vez realizado el movimiento respectivo, actualiza su creencia con respecto a su nueva posición.

4.1.2 DESIRES

Un agente situado, entrará en un ciclo de atención para construir el mundo y recibir estímulos del exterior que modifiquen sus creencias y generen un proceso de decisión-acción en búsqueda del cumplimiento de un objetivo general. Los objetivos pueden ser de diferente naturaleza y por lo tanto las acciones difieren entre sí. El modelo BDI trabajado, contempla una catalogación de objetivos a los que debe enfrentarse el agente en su proceso deliberativo, ilustrado en la figura 30. [Gonzalez 2012]

Figura 30. Jerarquía para la construcción de Objetivos

En términos del agente actor, pueden asignarse la relevancia que tienen para si mismo sus capacidades dentro de esta pirámide como:

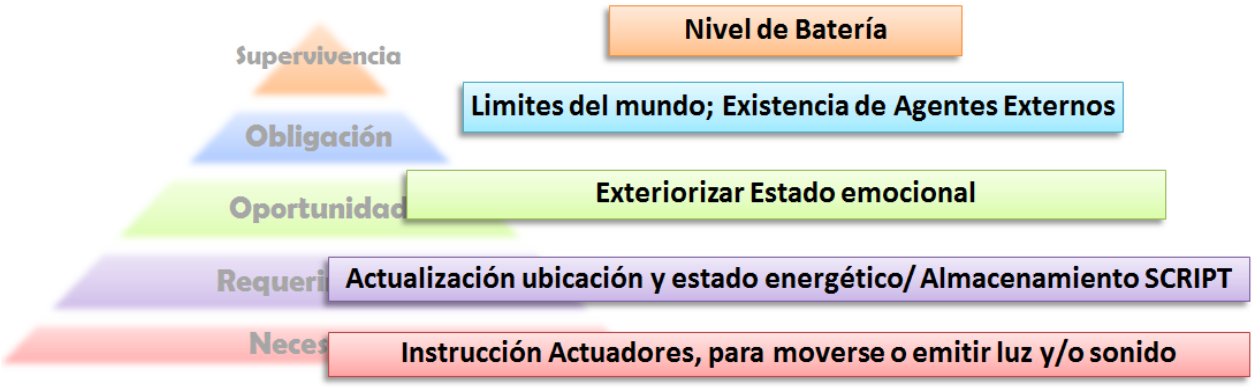

Figura 31. Jerarquía Objetivos Agente Actor

De esta forma la prioridad de supervivencia una vez actualice el estado de los sensores, siempre será disponer de un nivel de batería suficiente para realizar cualquier otra acción. Esto se traduce en tener un nivel de batería superior a 10%. De no ser así el agente entrará en EMOTION RESET sin eliminar los registros ni la memoria almacenada para permitir que su batería sea cargada.

Superando este nivel, los límites impuestos por el usuario para el tamaño de su entorno (WORLD) y la imposibilidad de ocupar el mismo espacio que el de otro agente externo (MOVE), dictan las obligaciones que a tener en cuenta durante el discernimiento de capacidades que puedan llegar a violarlas.

Siendo un agente diseñado para actuar, la exteriorización de sus emociones (EMOTION) se plantea como su objetivo principal, el cual debe buscar cumplir valiéndose de sus demás capacidades.

Debido a que para cumplir con los niveles jerárquicos superiores es necesario mantener un conocimiento de las modificaciones de su entorno y de sí mismo (Reg_STAT, Reg_SEN), es necesario permitir actualizar esta información conforme pasa el tiempo para lograr un funcionamiento adecuado (SENBUS). Finalmente las instrucciones con enfoque reactivo (LED, SOUND), posibilitan todas los demás niveles.

4.1.3 INTENTIONS

La etapa de acción comienza con la asignación de un rol al agente, el cual actuará reactivamente o deliberativamente siguiendo planes preestablecidos para cada acción. En el agente actor, esta etapa se caracteriza por 2 partes:

1. La generación de órdenes en el registro de salida Reg_CNTOUT y su posterior comunicación a los actuadores que se encargan de ejecutar las ordenes.
2. La emisión de señales de control al usuario para indicar que se encuentra listo para la recepción de nuevas órdenes.

4.2 DEFINICIÓN ARQUITECTURA

Como un primer punto a tener en cuenta, se decide trabajar con buses paralelos en esta primera aproximación, esperando que en trabajos futuros se pueda extender a comunicación serial, teniendo en cuenta que la información trabajada es del orden de bytes y que cada instrucción tomará como mínimo 1s en ejecución para que pueda ser perceptible. Adicionalmente es importante resaltar el enfoque de comportamiento modular, que permita en una eventual aplicación práctica, el manejo independiente de entradas y salidas, fuera del módulo de razonamiento.

La arquitectura propuesta para el desarrollo del agente actor, se basa en el paradigma BDI para agentes inteligentes y tiene en cuenta para la definición de los módulos que la componen, las características, funcionalidades y capacidades de integración buscadas para el agente como un todo.

Como se observa en la Figura 1, se contemplan los siguientes componentes:

- BUSES: agrupación de señales en paralelo que comunican entradas y salidas con los registros internos. De diferentes tamaños según los registros que se estén trabajando. INBUS, SENBUS, MXYBUS, LEDBUS, SONBUS.
- SEÑALES: Conexión unitaria entre el exterior y el interior usadas como banderas de enable para iniciar el desarrollo de procesos. M_ready, led_ready, son_ready, agent_ready, in_ready y sex.
- REGISTROS: Unidad de almacenamiento básica que permite persistir bits de información. Reg_STAT, Reg_D, Reg_TM, Reg_DIRTM, Reg_CNTOUT, Reg_CREC, Reg_CDIR.
- MEMORIAS: Unidad de almacenamiento "masivo" que permite persistir altas cantidades de información. Funciona como un arreglo de registros. MEM y DIR.
- CU: Unidad de Control que maneja la secuencialidad y habilitación de los demás componentes para permitir el flujo de la información de la forma esperada.
- DECODER: Centro de procesamiento de la información donde se realizan las actividades deliberativas y reactivas correspondientes según las capacidades solicitadas por el usuario.

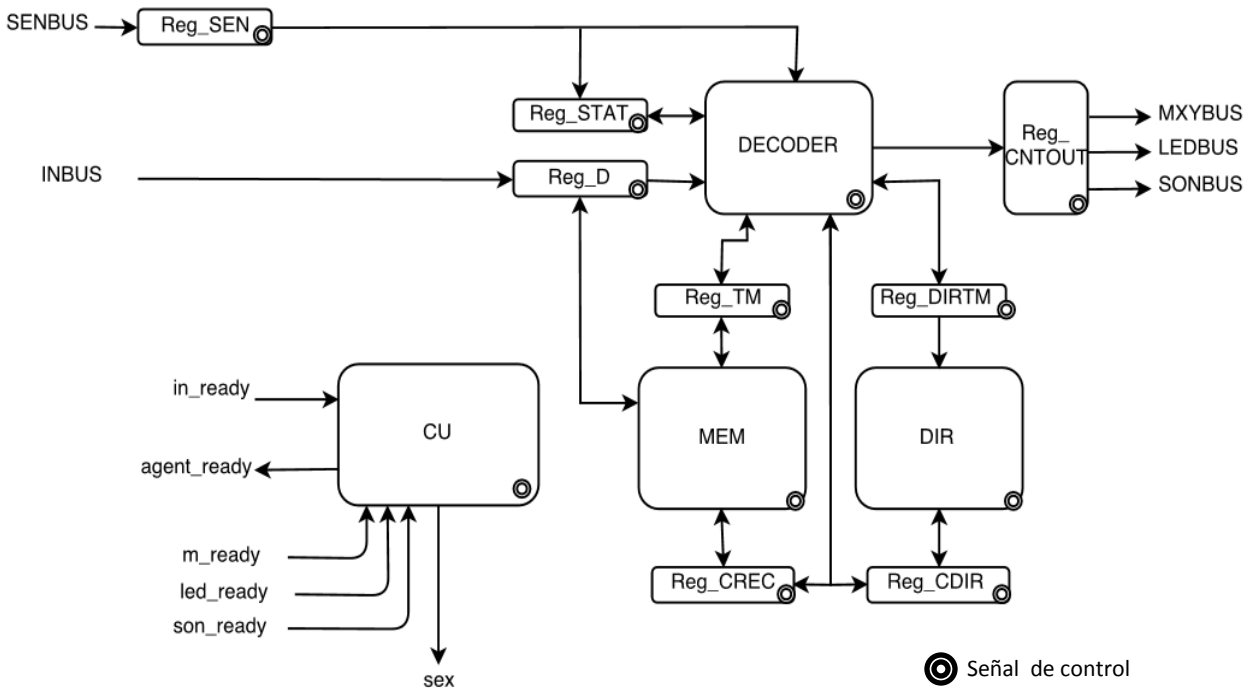

Figura 1. Diagrama de componentes

El flujo de datos, se estructura de tal forma que la información ingrese al agente por medio de los buses de entrada SENBUS e INBUS, y las 4 señales de control provenientes de los actuadores (m_ready, led_ready, son_ready) y el usuario (in_ready), sea manejada por una unidad de control lógico secuencial, procesada por un decodificador que interpretará cada instrucción y ejecutará las acciones requeridas, para finalmente almacenar en un registro de salida las órdenes para los actuadores según lo defina el procesamiento, quienes las recibirán por medio de buses dedicados para cada uno.

En este orden de ideas se definen los componentes, para cubrir el flujo de información.

4.2.1 ENTRADAS

- INBUS. Bus paralelo de 12 posiciones (3 Bytes), que ingresa el código de instrucción y los dos operandos para su interpretación.
- SENBUS: Bus paralelo de 6 posiciones (6 Bits), que ingresa el nivel de batería y la ubicación de un agente externo de haber detectado alguno
- In_ready: Señal de 1 Bit para leer INBUS, que indica si exsite una instrucción en INBUS para ser leida por el agente,
- M_ready: Señal de 1 Bit para indicar que es posible escribir MXYBUS.
- Led_ready: Señal de 1 Bit para indicar que es posible escribir LEDBUS.
- Son_ready: Señal de 1 Bit para indicar que es posible escribir SONBUS.

4.2.2 SALIDAS

- MXYBUS: Bus paralelo de 8 posiciones (1 Byte), que indica al actuador la dirección en la cual debe efectuar el movimiento.

- LEDBUS: Bus paralelo de 8 posiciones (1 Byte) que indica al actuador, el color y tiempo deseado para la emisión de luz.
- SONBUS: Bus paralelo de 8 posiciones (1 Byte) que indica al actuador, el tono y tiempo deseado para la emisión de sonido
- agent_ready: Señal de 1 Bit para el usuario que indica que el agente se encuentra esperando una nueva instrucción.
- sex: Señal de 1 Bit para los actuadores que complementan la personalización del agente en la expresión de sus emociones.

4.2.3 REGISTROS:

- Reg_Sen: registro que se encarga de almacenar el contenido de SENBUS. Én la figura 2, PS indica la posición de un agente externo en la periferia del agente de haber sido detectado y BATT, el nivel de bateria disponible.

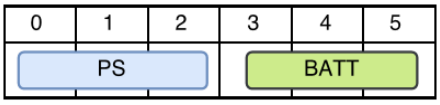

figura 2. Reg_Sen

- Reg_STAT: Registro de estado. Se encarga de almacenar la información relevante sobre el estado actual del agente como batería, posición actual, estado emocional y banderas de ejecución. En la figura 3, X0 y Y0 son las coordenadas actuales del agente. R corresponde a la bandera para indicar que el agente se encuentra recitando, por lo que las instrucciones provienen de la memoria y no del exterior. G corresponde a la bandera para controlar el flujo de información hacia la memoria si el agente se encuentra almacenando las instrucciones de un guion.MX XR YR son banderas usadas para el proceso de movilidad del agente. RS indica que el agente fue ordenado a ser reiniciado y W, un contador para identificar que los parámetros iniciales fueron configurados.

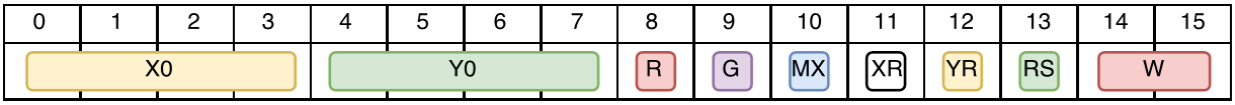

Imagen 3. Reg_STAT

- Reg_D: Registro de decodificación. Almacena la instrucción a ser interpretada por el decodificador, proveniente de INBUS, MEM o el mismo DECODER según sea el caso. En la figura 4, OPCODE corresponde al codigo de instrucción y OPER1 y OPER 2 a los operadores caracterisitcos de cada capacidad.

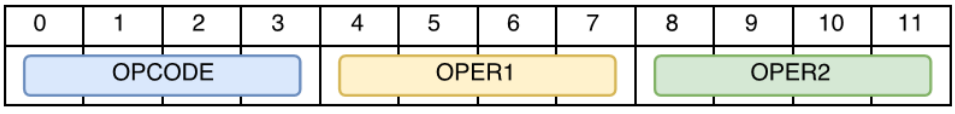

Figura 4. Reg_D

- Reg_TM: Registro que almacena un dato de Memoria requerido para la decodificación de alguna instrucción. En la figura 5, XMAX y YMAX corresponde a las coordenadas máximas de la matriz del mundo e ID al parametro de idenditidad otorgado al agente.

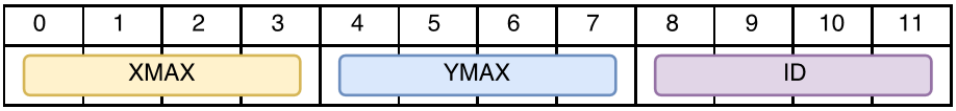

Figura 5. Reg_TM

- Reg_CREC: Registro de control para el acceso a la Memoria MEM para la ejecución de la capacidad RECITAR. En la figura 6, RADDW lleva el registro de la dirección en memoria en la cual se debe escribir la siguiente instrucción y RADDR de la cual se debe leer.

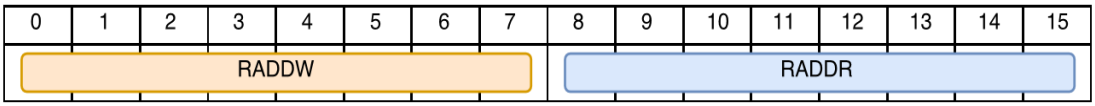

Figura 6. Reg_CREC

- Reg_CDIR: Registro de control para el acceso a la memoria DIR para la búsqueda de una escena solicitada. En la figura 7, DADDW lleva el registro de la dirección en memoria en la cual se debe escribir la siguiente instrucción y DADDR de la cual se debe leer.

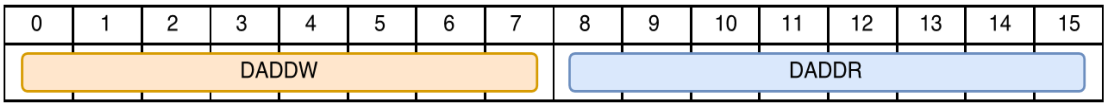

Figura 7. Reg_CDIR

- Reg_CNTOUT: Registro de salida. Almacena la instrucción que será entregada a los actuadores. En la figura 8, OPER 1 y OPER 2 son caracteristicos del actuador al cual se desee enviar la instucción.

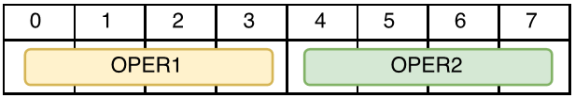

Figura 8. Reg_CNTOUT

4.2.4 MEMORIAS

- MEM: Arreglo de 256 x 12 bits, en el que se almacena la información del mundo en la dirección 0 y las instrucciones para cada una de las escenas definidas por el usuario a partir de la dirección 1 en adelante. Es de vital importancia resaltar que la información del mundo se almacena en la memoria y no en un registro externo, pues es información estática que solamente puede ser configurada una vez por el usuario. Se genera una eventual funcionalidad de RESET que será explicada más adelante. Su estructura se observa en la figura 9.

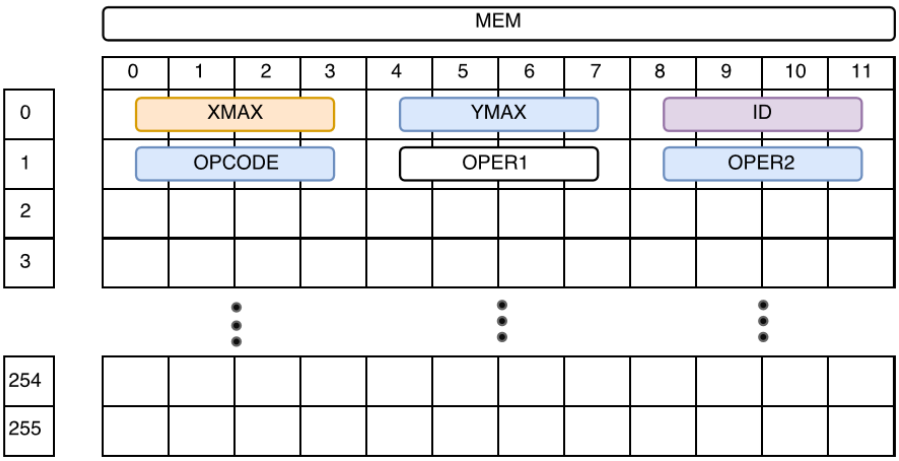

Figura 9. Memoria MEM

- DIR: Arreglo de 128 x 16 bits, en el que se almacena la dirección en memoria de las escenas almacenadas por el usuario. Su estructura se observa en la figura 10.

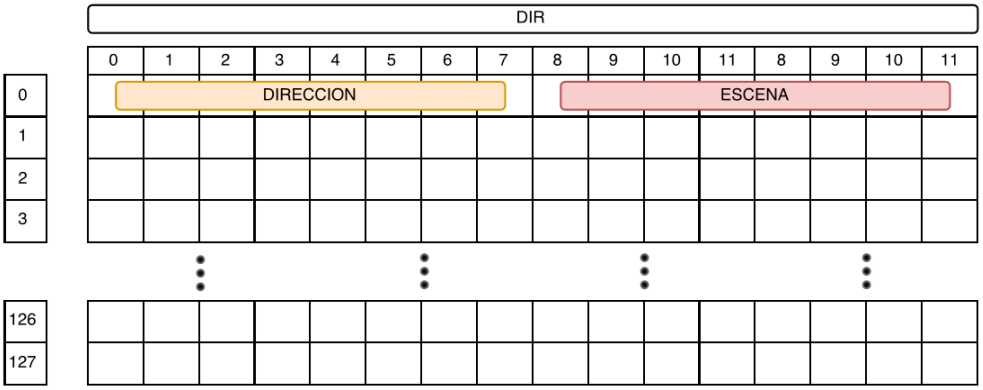

Imagen 10. Memoria DIR

4.3 MANEJO DE DATOS

Se muestra a continuación el detalle de como fluye y es manejada la información de entrada dentro del agente.

4.2.1 ENTRADA

Es de vital importancia reconocer que los datos con los cuales el agente realizará el proceso deliberativo, pueden provenir del usuario o de los sensores. Esta información debe estar estructurada según un protocolo definido para el agente, organizando la información en cada BUS de entrada de la siguiente forma:

INBUS estructurado como se muestra en la figura 1. La información proveniente es almacenada únicamente en Reg_D (Imagen 4), cuando la CU lo defina.

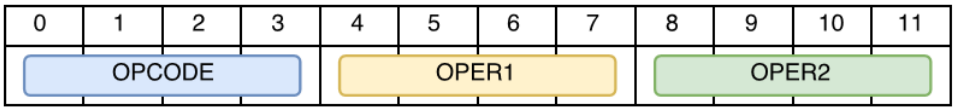

Figura 11. Estructura INBUS

- OPCODE: Código que determina la capacidad que el usuario requiere que el agente ejecute.

OPCODE	OC			
	0	1	2	3
N/A	0	0	0	0
WORLD	0	0	0	1
MOVER	1	0	0	1
EMOCION	1	0	1	0
ENCENDER LED	1	0	1	1
EMITIR SONIDO	1	1	0	0
RECITAR	1	1	0	1
GUION	1	1	1	0
N/A	1	1	1	1

Tabla 1. OPCODE's

- OPER1: Operador determinado para definir el detalle de cada una de las funcionalidades que el usuario puede requerir del agente. Depende de cada capacidad.
- OPER2: Segundo operador, para definir el detalle de cada una de las funcionalidades que el usuario puede requerir del agente. Depende de cada capacidad.

SENBUS: la información de entrada, es almacenada únicamente en Reg_SEN (Figura 2) cuando la CU lo defina. Está compuesta por:

- PS: Perimetral Sensors, indica si alguno de los sensores propuestos para la implementación, detecta la presencia de un agente externo en alguna de las posiciones perimetrales contiguas al agente. Esto es al norte (N), sur (S), este(E) u oeste(W) del agente, como se ilustra en la figura 12 y tabla 2.

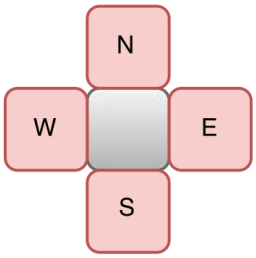

Figura 12. Perimetral Sensor

PS			PERIMETRAL SENSOR
0	0	0	N/A
0	0	1	N
0	1	0	S
0	1	1	E
1	0	0	W
1	0	1	N/A
1	1	0	N/A
1	1	1	N/A

Tabla 2. Valores PS

- BATT: Nivel de batería entregado en un código de 3 bits según un determinado nivel de carga según se caracteriza en la Tabla 3.

BATT			Battery (%)
0	0	0	1 - 10
0	0	1	10 - 20
0	1	0	20 - 30
0	1	1	30 - 40
1	0	0	40 -50
1	0	1	60 - 80
1	1	0	90 - 80
1	1	1	100 - 90

Tabla 3. Valores Battery

Se requiere adicionalmente el uso de banderas que indiquen al agente cuando puede interactuar con la información en un BUS específico, controlando así el flujo de información hacia y desde el agente. Estas señales son:

- In_ready: señal que indica al agente que el usuario ha ingresado una nueva instrucción en INBUS y puede leer la información.
- M_ready: señal que indica al agente que los actuadores de motores están listos para recibir una nueva orden.
- Led_ready: señal que indica al agente que los actuadores de encendido de LED's están listos para recibir una nueva orden.
- Son_ready: señal que indica al agente que los actuadores de sonido están listos para recibir una nueva orden.

4.3.1 SALIDA

La información que entrega el agente, puede estar dirigida al usuario o a los actuadores. En este último caso puede ser enviada a través de uno de los 3 buses dedicados para cada uno de los destinos según la operación lo requiera.

- MXYBUS: bus paralelo de 8 bits (1 Byte) que contiene la información para el actuador de motores. Su estructura se muestra en la figura 13.

Figura 13. MXYBUS

- LEDBUS: bus paralelo de 8 bits (1 Byte) que contiene la información para el actuador de encendido de LED's. Su estructura se muestra en la Figura 14 y su contenido se explica a detalle en 4.2.3.4.

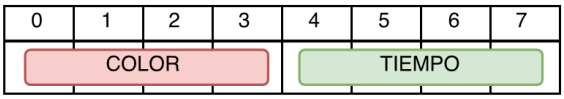

Imagen 14. LEDBUS

- SONBUS: bus paralelo de 8 bits (1 Byte) que contiene la información para el actuador de Sonido. Su estructura se muestra en la figura 15 y su contenido se explica a detalle en 4.2.3.5.

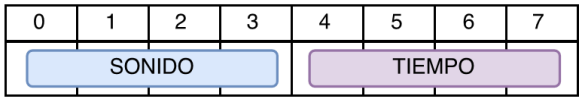

Figura 15. SONBUS

- agent_ready: señal de 1 Bit que indica al usuario si el agente está preparado para recibir una nueva instrucción.
- gender: señal de 1 Bit que indica a los actuadores de LED y SONIDO la identidad de género configurada por el usuario para el agente.

4.3.2 PROCESAMIENTO INTERNO

Una vez la información ha sido recibida por el agente, se da inicio a los procesos de decodificación, razonamiento y actuación del agente según haya sido solicitado por el usuario. Estos procesos involucran la integración y participación de diferentes registros y según la capacidad solicitada, de la ejecución de los procesos deliberativos o reactivos antes mencionados. Se prosigue entonces a definir cada uno de las capacidades del agente, con un enfoque técnico y funcional que permita observar la implementación de la arquitectura BDI planteada, estableciendo los parámetros de funcionamiento de cada una y las acciones que el agente tomará.

4.3.2.1WORLD (OPCODE 0001)

La capacidad WORLD permite al usuario ingresar los parámetros que definen en un primer momento las características de su entorno y su identidad. Esta instrucción debe ser ejecutada 3 veces, para configurar la información de su identidad, límites del mundo y posición inicial, todas de vital importancia para permitirle al agente la correcta ejecución de sus demás capacidades. Es importante observar que el diseño del agente se basa en que las características de su entorno son las siguientes:

- El mundo se define como una matriz de máximo 16 x 16 posiciones, en la cual el agente ocupará una única coordenada en todo momento.
- El agente no podrá moverse fuera de los limites parametrizados por el usuario.
- Pueden existir agentes externos dentro de dicha matriz, los cuales en el alcance del actual trabajo, no serán caracterizados.
- El agente podrá reconocer la existencia de un agente externo por sus sensores perimetrales.
- Dentro del mundo, la posición de coordenadas 0,0 siempre corresponderá a un punto de carga al cual el agente se dirigirá automáticamente, una vez identifique que su batería llego al punto crítico 0-10%. (4.2.1 BATT).

Una vez inicie su operación, el agente dispone de un contador W (STAT [14,15]) con valor inicial 0, que reconocerá las primeras 3 instrucciones WORLD para almacenar los operandos de entrada en un registro y posición específicas:

- W = 0: almacena OPER1 y OPER 2 en los bits 0 a 7 de MEM en la Dirección 0. XMAX y YMAX hacen referencia al tamaño máximo de la matriz que caracteriza el entorno; Como se mencionó

anteriormente, y técnicamente hablando por el tamaño de los operandos, el máximo valor será 16 o 1111 binario. El proceso se ilustra en la figura 16.

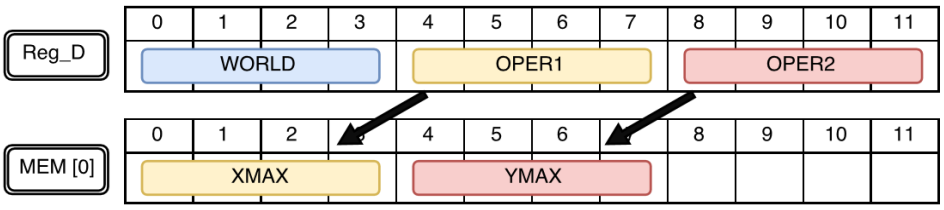

Figura 16. WORLD MAX

- W = 1: almacena OPER1 en los bits 8 a 11 de MEM en la Dirección 0. ID hace referencia a la identidad especifica del agente el cual lo caracterizará para la expresión de emociones. El proceso se ilustra en la figura 16 y los posibles valores se observan en la tabla 4.

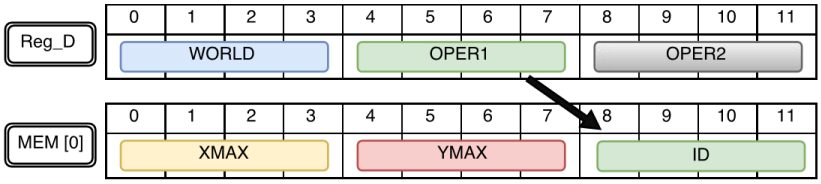

Figura 17. ID

ID			Identity
0	0	0	N/A
0	0	1	Balanceado (H)
0	1	0	Alegre (H)
0	1	1	Depresivo (H)
1	0	0	Balanceado (M)
1	0	1	Alegre(M)
1	1	0	Depresivo (M)
1	1	1	N/A

Tabla 4. Valores ID

- W = 2: almacena OPER1 y OPER2 en los bits 0 a 7 de STAT. X0 y Y0 hacen referencia a la posición inicial del agente. De esta forma el usuario puede dar inicio a la operación del agente en cualquier posición dentro de la matriz. El proceso se ilustra en la figura 18.

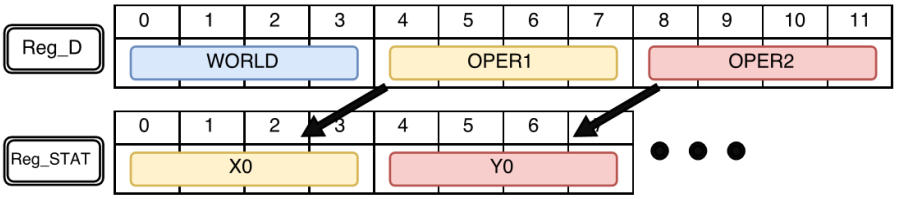

Figura 18. W0 / Y0

- W = 3: el agente expresa ERROR. El proceso se ilustra en la figura 19.

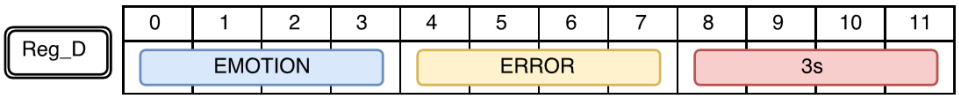

Figura 19. ERROR WORLD

Aunque los parámetros deben ser ingresados en este orden, pueden ser ingresados con instrucciones intermedias; claramente, esto puede generar comportamientos erráticos, en específico para las capacidades deliberativas, al no tener la información requerida para su procesamiento.

4.3.2.2MOVE (OPCODE 1001)

Teniendo en cuenta las características del mundo mencionadas en 4.2.3.1, la capacidad MOVE permite al agente realizar el desplazamiento de su posición actual a la posición determinada por las coordenadas ingresadas en los operandos OPER1 y OPER2, como se ilustra en la figura 20, donde MOVE corresponde al OPCODE respectivo para esta característica, y XF y YF a las coordenadas hacia las cuales el usuario requiere que el agente se mueva.

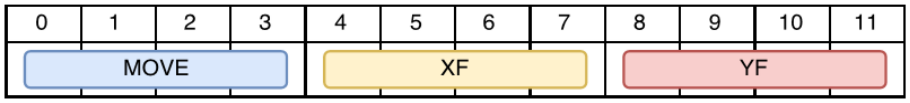

Figura 20. Instrucción MOVE

El agente tendrá la intención de realizar un movimiento diagonal de a una posición, esto es: un movimiento en X, un movimiento en Y, un movimiento en X y así sucesivamente, hasta que encuentre la primera coordenada y luego realizará un movimiento recto hasta llegar a su destino. En caso de encontrar un agente externo en su trayectoria, realizará un movimiento de carácter evasivo para superar el obstáculo, comprendido por un movimiento en la dirección contraria (si era +, se moverá en – y viceversa) en la otra coordenada a la cual se encontró el obstáculo: es decir, si el movimiento requerido era en X, se realizará un movimiento en -Y, y luego un movimiento en la coordenada en la que se encontró el agente externo, en nuestro ejemplo, X. El detalle de este proceso ilustra en la figura 21 y se encuentra detallado en el ANEXO 5.

Con la finalidad de no añadir complejidad innecesaria al diseño del agente, se especifican que los agentes externos no podrán estar situados de forma contigua uno del otro en ninguna posición de su periferia.

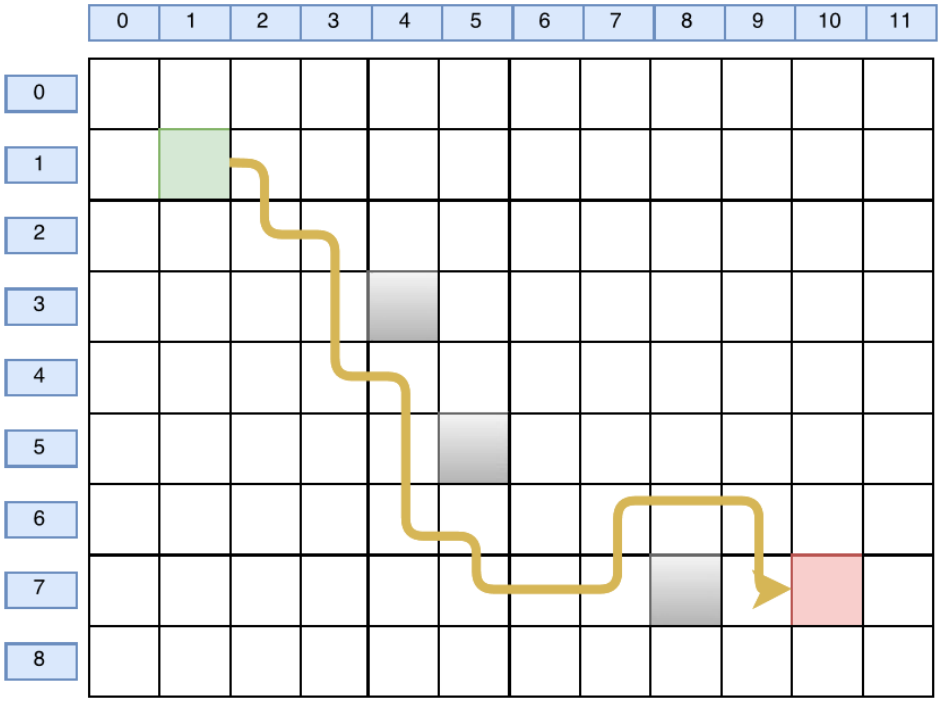

Figura 21. Ejemplo de ejecución MOVE

Esta capacidad tiene un enfoque deliberativo, dado que el agente debe evaluar sus creencias del mundo y de sí mismo para determinar si es posible llevar a cabo la instrucción, requiriendo para esto la priorización de los objetivos a cumplir con el fin de escoger el movimiento indicado para llegar a su destino como se ilustra en la imagen 22 y se define a continuación.

1. El destino se encuentra dentro de los límites del mundo, XMAX/YMAX?
   - Conciencia del mundo
2. El destino es diferente de mi posición actual? Debo moverme? XF!=X0 / YF!=Y0.
   - Conciencia de sí mismo
3. Si debo moverme, en qué dirección puedo hacerlo según mi entorno?
   - Conciencia del mundo
4. Puedo Realizar mi movimiento?

Imagen 22. Jerarquía de DESEOS planteada para MOVE

Adicionalmente aunque la prioridad del movimiento es hacia el destino, si el proceso deliberativo determina que es necesario hacerlo en dirección contraria, podrá ejecutarlo aun siendo contraproducente (aumentar la cantidad de movimientos necesarios) con la finalidad de cumplir su objetivo final: llegar al destino.

4.3.2.3EMOTION (OPCODE 1010)

La capacidad EMOTION, permite al agente por medio de sus actuadores LED y SOUND, exteriorizar una de 7 posibles emociones que se traducen en una Combinación especifica de color y sonido por un periodo determinado de tiempo o siguiendo uno de los patrones previamente establecidos. Cabe aclarar que aunque el agente indicará a los actuadores los parámetros para efectuar cada acción, estos deben estar en capacidad de interpretar dichas instrucciones. Esta lógica combinatoria simple, fue excluida del agente pues es inherente a la funcionalidad del actuador mismo, como un módulo independiente, más que a la del agente. El proceso de transferencia de información se ilustra en la Figura 23, donde "lógica EMOTION" se refiere a un proceso que identifica el color y sonido respectivo para cada una de las emociones que puede expresar el agente como se observa en la Tabla 5.

.

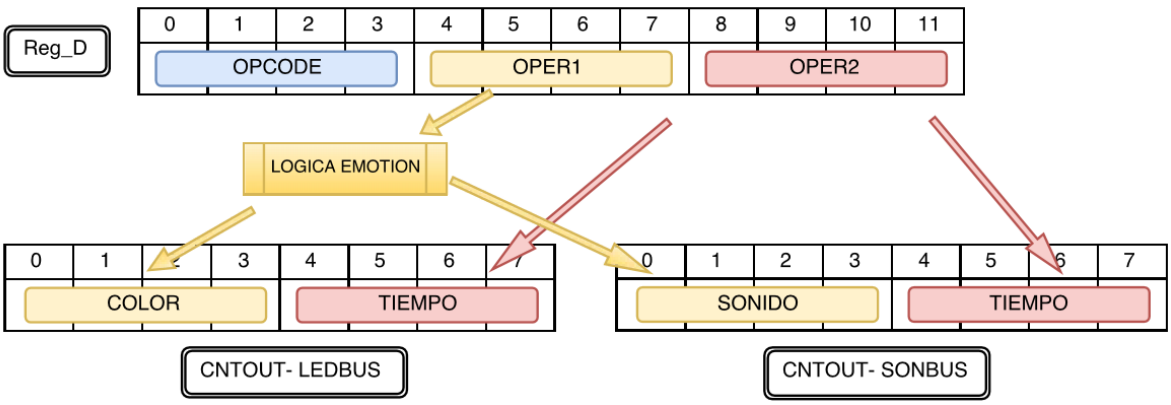

Figura 23. Instrucción EMOTION

	EMOCION	OC				OPER1				OPER2			
						0	1	2	3	0	1	2	3
	Real/EMO	1	0	1	0	0	0	0	0	T0	T1	T2	T3
ESTADO EMO	Explendido	1	0	1	0	0	0	0	1	T0	T1	T2	T3
	Feliz	1	0	1	0	0	0	1	0	T0	T1	T2	T3
	Tranquilo	1	0	1	0	0	0	1	1	T0	T1	T2	T3
	Preocupado	1	0	1	0	0	1	0	0	T0	T1	T2	T3
	Triste	1	0	1	0	0	1	0	1	T0	T1	T2	T3
	Deprimido	1	0	1	0	0	1	1	0	T0	T1	T2	T3
ACTUACION	Sorprendido	1	0	1	0	0	1	1	1	T0	T1	T2	T3
	Emocionado	1	0	1	0	1	0	0	0	T0	T1	T2	T3
	Enojado	1	0	1	0	1	0	0	1	T0	T1	T2	T3
	N/A	1	0	1	0	1	0	1	0	T0	T1	T2	T3
	N/A	1	0	1	0	1	0	1	1	T0	T1	T2	T3
	N/A	1	0	1	0	1	1	0	0	T0	T1	T2	T3
	N/A	1	0	1	0	1	1	0	1	T0	T1	T2	T3
	ERROR	1	0	1	0	1	1	1	0	T0	T1	T2	T3
	RESET	1	0	1	0	1	1	1	1	T0	T1	T2	T3

Tabla 5. EMOTION

a. Expresar una emoción específica. Actuación.

En este caso, el agente usará un proceso de lógica combinatoria con un enfoque reactivo, que generará en el registro de salida Reg_CNTOUT, las instrucciones específicas hacia los actuadores para expresar la emoción solicitada por el tiempo indicado. La característica de luz y sonido se observa en la tabla 6.

b. Expresar la emoción que muestre el estado actual del agente, según su percepción de sí mismo.

- Conciencia de sí mismo

	OPER1				**Color**	**Sonido**
Explendido	0	0	0	1	Verde	B
Feliz	0	0	1	0	Azul	C
Tranquilo	0	0	1	1	Rosa	D
Preocupado	0	1	0	0	Amarillo	E
Triste	0	1	0	1	Naranja	F
Deprimido	0	1	1	0	Morado	G
Sorprendido	0	1	1	1	Amarillo	E
Emocionado	1	0	0	0	Azul	C
Enojado	1	0	0	1	Rojo	H
EMO2	1	0	1	0	Amarillo	E
EMO3	1	0	1	1	Naranja	F
EMO4	1	1	0	0	Morado	G
EMO5	1	1	0	1	Rojo	H

Tabla 6. ACT

Esta capacidad, infiere un proceso deliberativo para expresar según su estado actual, una expresión específica, basado además en su propia identidad. Esto es, según el nivel de batería actual, identificará un estado emocional, el cual expresará durante el tiempo establecido, de una forma específica según su propia identidad. Basado en las tablas 3 y 4, se estructura la tabla 7 que describe las emociones mencionadas.

Estados									
ID	Estado	Color	Sonido	Batería					Tipo ID
				Max	Min	Batt			
ID 1	Explendido	1	1	100	90	1	1	1	Balanceado
	Feliz	2	2	90	60	1	1	0	
						1	0	1	
	Tranquilo	3	3	60	40	1	0	0	
	Preocupado	4	4	40	20	0	1	1	
						0	1	0	
	Triste	5	5	20	10	0	0	1	
	Deprimido	6	6	10	1	0	0	0	
ID 2	Explendido	1	1	100	80	1	1	1	Alegre
						1	1	0	
	Feliz	2	2	80	40	1	0	1	
						1	0	0	
	Tranquilo	3	3	40	30	0	1	1	
	Preocupado	4	4	30	20	0	1	0	
	Triste	5	5	20	10	0	0	1	
	Deprimido	6	6	10	1	0	0	0	
ID3	Explendido	1	1	100	90	1	1	1	Depresiva
	Feliz	2	2	90	80	1	1	0	
	Tranquilo	3	3	80	60	1	0	1	
	Preocupado	4	4	60	40	1	0	0	
	Triste	5	5	40	20	0	1	1	
						0	1	0	
	Deprimido	6	6	1	20	0	0	1	
						0	0	0	

Tabla 7. Tabla de estados emocionales

c. Expresar la ocurrencia de un ERROR

Según la ejecución específica de una capacidad, el agente podría encontrarse eventualmente en situaciones que no tengan sentido para sus capacidades técnicas. En este caso, la emoción ERROR, permite exteriorizar dicho escenario, de una forma característica para notificar al usuario, acorde a la "personalidad de Actor" que se quiere con la implementación del Agente. En este escenario se desea exteriorizar el color rojo por un periodo de 3 segudos, como se ilustra en la figura 24.

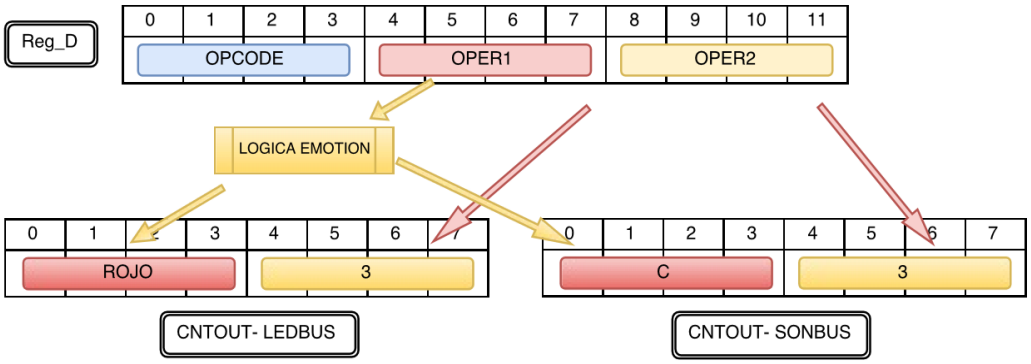

Figura 24. Emoción ERROR

d. Generar un proceso de REINICIO.

Con esta capacidad en específico, se pretende dar al usuario la posibilidad de entregar un nuevo grupo de instrucciones al agente, en términos prácticos: un nuevo guion, permitiendo de esta forma liberar los recursos de memoria usados hasta el momento, para no comprometer los recursos de forma innecesaria. Teniendo en cuenta que el objetivo del agente actor es mostrar a un público humano una serie de emociones, el tamaño asignado a memoria, permite una ejecución de instrucciones del orden de minutos, lo suficientemente acorde para ser comprendida.

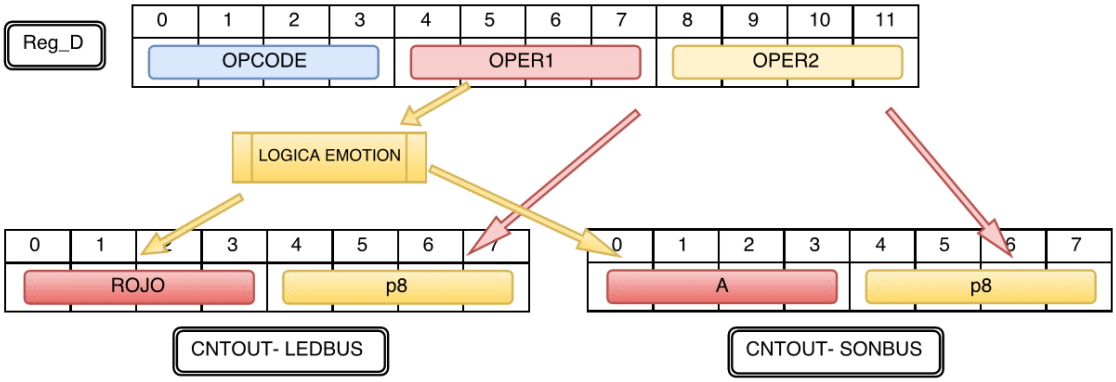

Figura 25. Emoción RESET

Al ingresar esta instrucción el agente ejecutará las siguientes instrucciones de forma secuencial:

a. Indicará a los actuadores LED Y SONIDO el color rojo y el sonido C con un patrón de tiempo p8, como se ilustra en la figura 25.
b. Almacenará en el registro Reg_D la capacidad MOVE con coordenadas (0,0).
c. Levantará la bandera indicando que está en proceso de RESET en el registro STAT, para que una vez finalice el desplazamiento, vuelva a su estado inicial, borrando el contenido de sus registros y de las memorias.

4.3.2.4LED (OPCODE 1011)

La capacidad LED permite indicar a los actuadores un color y un tiempo de encendido deseado como se ilustra en la figura 26. Es una capacidad netamente reactiva por lo que solamente requiere la transferencia de la instrucción de un registro a otro. De nuevo, es importante tener en cuenta que aunque el agente indicará al actuador los parámetros para realizar el encendido del led, este debe estar en capacidad de interpretar dichas instrucciones. Esta lógica combinatoria simple, fue excluida del agente pues es inherente a la funcionalidad del actuador como un módulo independiente, más que a la del agente

			OPER1 COLOR						OPER2 T				
			0	1	2	3			0	1	2	3	
		Color Fijo	0	0	0	0	Negro	Tiempo Fijo	0	0	0	0	0.5
			0	0	0	1	Verde		0	0	0	1	1
			0	0	1	0	Azul		0	0	1	0	3
			0	0	1	1	Rosa		0	0	1	1	5
			0	1	0	0	Amarillo		0	1	0	0	10
			0	1	0	1	Naranja		0	1	0	1	15
			0	1	1	0	Morado		0	1	1	0	30
			0	1	1	1	Rojo		0	1	1	1	60
H: AZUL	M: ROSA	Intensidad / Degradé Color Personal	1	0	0	0	1	Patron de tiempos	1	0	0	0	p1
			1	0	0	1	2		1	0	0	1	p2
			1	0	1	0	3		1	0	1	0	p3
			1	0	1	1	4		1	0	1	1	p4
			1	1	0	0	5		1	1	0	0	p5
			1	1	0	1	6		1	1	0	1	p6
			1	1	1	0	7		1	1	1	0	p7
			1	1	1	1	Blanco		1	1	1	1	p8

Figura 26. Capacidad LED

4.3.2.5SOUND (OPCODE 1100)

La capacidad SOUND permite indicar al actuador de sonido, un código de sonido y un tiempo de emisión deseado como se observa en la figura 27. Es una capacidad netamente reactiva por lo que solamente requiere la transferencia de la instrucción de un registro a otro. De nuevo, es importante tener en cuenta que aunque el agente indicará al actuador los parámetros para realizar la emisión del sonido, este debe estar en capacidad de interpretar dichas instrucciones. Esta lógica combinatoria simple, fue excluida del agente pues es inherente a la funcionalidad del actuador como un módulo independiente, más que a la del agente

			OPER1						OPER2				
			SONIDO						T				
			0	1	2	3			0	1	2	3	
		Sonido Fijo	0	0	0	0	A	Tiempo Fijo	0	0	0	0	0.5
			0	0	0	1	B		0	0	0	1	1
			0	0	1	0	C		0	0	1	0	3
			0	0	1	1	D		0	0	1	1	5
			0	1	0	0	E		0	1	0	0	10
			0	1	0	1	F		0	1	0	1	15
			0	1	1	0	G		0	1	1	0	30
			0	1	1	1	H		0	1	1	1	60
H: GRAVE	M: AGUDO	Intensidad / Degradé Sonido Personal	1	0	0	0	1	Patron de tiempos	1	0	0	0	p1
			1	0	0	1	2		1	0	0	1	p2
			1	0	1	0	3		1	0	1	0	p3
			1	0	1	1	4		1	0	1	1	p4
			1	1	0	0	5		1	1	0	0	p5
			1	1	0	1	6		1	1	0	1	p6
			1	1	1	0	7		1	1	1	0	p7
			1	1	1	1	8		1	1	1	1	p8

Figura 27. Capacidad SOUND

4.3.2.6 ACT (OPCODE 1101)

La capacidad ACT, permite al usuario indicarle al agente que requiere ejecutar una de las escenas almacenadas previamente en MEM mediante la capacidad SCRIPT descrita en 4.2.3.7. El agente realizará una búsqueda en la Memoria DIR del número de escena solicitado, para identificar la dirección en memoria desde la cual inician las instrucciones almacenadas para la escena solicitada. Si no encuentra el número de escena requerido, ejecutará EMOTION ERROR y esperará la siguiente instrucción del usuario. En caso de encontrarla, llevará a cabo los siguientes pasos, ilustrados también en la figura 28:

b. Levantará la bandera de Actuación en el registro Reg_STAT[8]
c. Mantendrá la señal agent_ready en bajo para indicar al usuario que no recibirá nuevas instrucciones.
d. El registro de decodificación Reg_D tomará las instrucciones directamente de la memoria MEM y no del INBUS.
e. Realizará la ejecución de las capacidades almacenadas en memoria, hasta encontrar una instrucción ACT. Esta indicará que la escena finalizó.
f. Una vez finalizada, se bajará la bandera en Reg_STAT[8], se activara agent_ready y se esperará una nueva instrucción por parte del usuario.

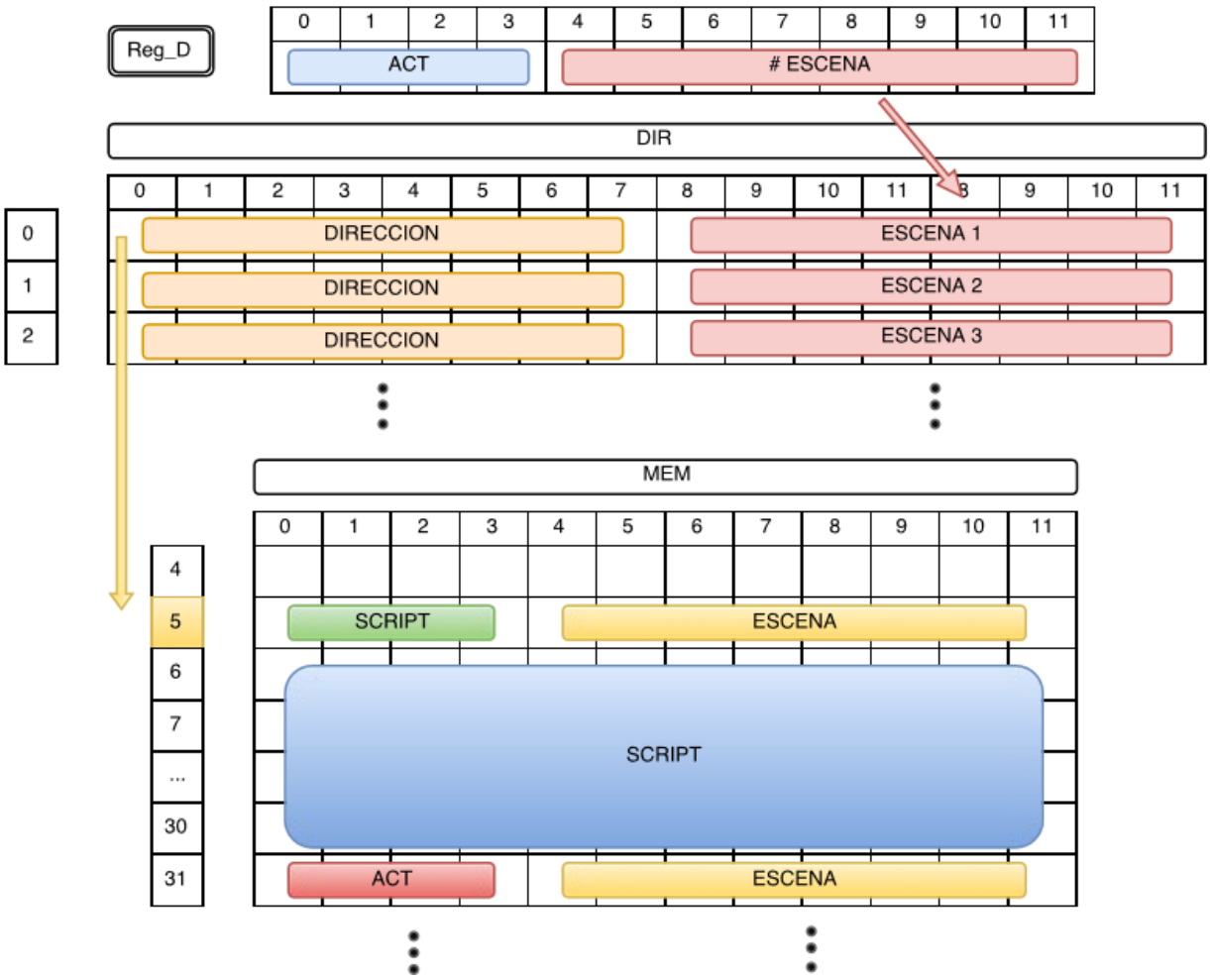

Figura 28. Capacidad ACT

4.3.2.7SCRIPT (OPCODE 1110)

Finalmente, esta capacidad permite al usuario almacenar en memoria una serie de instrucciones para su posterior ejecución, haciendo analogía a como un actor recibe un guion. Una vez el agente recibe esta instrucción realiza los siguientes procedimientos ilustrados en la figura 29:

a. Almacena en la siguiente posición de memoria libre de MEM, el número de la escena que le es indicado dentro con los operadores OPER1 y OPER2. Adicionalmente almacena en la memoria DIR, la dirección y el número de escena, para su posterior búsqueda.

b. Levanta la bandera en Reg_STAT[9] indicando al decodificador que las instrucciones que reciba desde INBUS desde ese momento en adelante, deben ser almacenadas en la memoria MEM y no ejecutadas.

c. Al recibir una nueva instrucción de SCRIPT, bajará la bandera de Reg_STAT[9] y almacenará en la siguiente posición de MEM la instrucción ACT con el número de escena, para cuando el agente la esté ejecutando, reconozca que este es el fin de la misma.

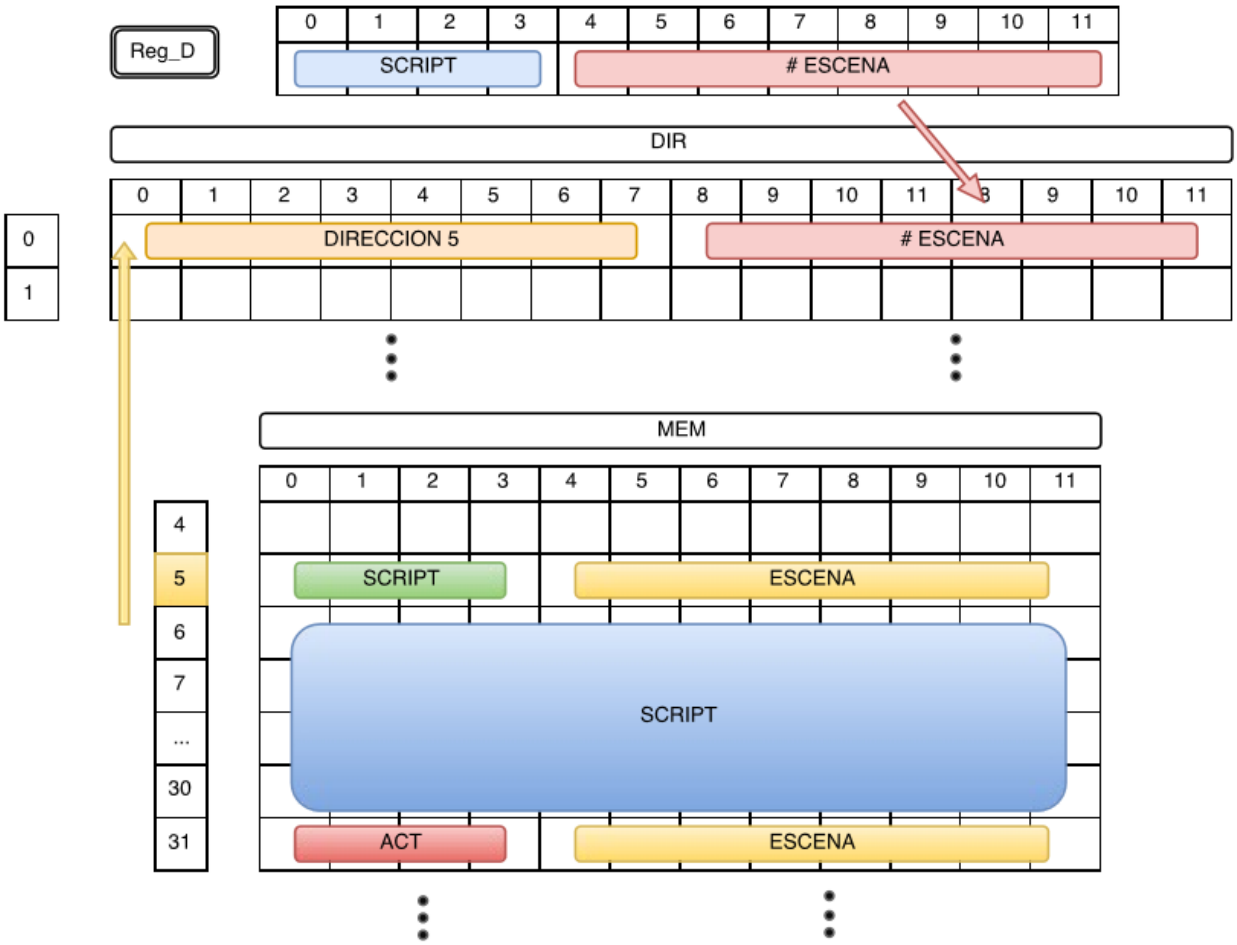

Figura 29. Capacidad SCRIPT

4.4 IMPLEMENTACIÓN

Como parte del proceso para implementar la arquitectura y el flujo de datos propuesto, se realizan 2 acercamientos de diseño para identificar, cuales son los pasos que el agente debe seguir para cubrir las funcionalidades propuestas. Se modela entonces por medio de diagramas de flujo, los pasos a nivel de capa transferencia de registros, identificando todos los posibles caminos en los que viajaría la información desde el usuario hasta los actuadores para cada una de las capacidades planteadas. Se identifican 2 procesos principales para simplificar la diagramación:

- MAIN: En el cual se encuentra la asignación inicial de registros, el proceso de "stand by" mientras se recibe la señal por parte del usuario para iniciar la lectura de instrucciones, y el proceso de recepción desde INBUS. (ANEXO 2)
- DECODER: Lectura de sensores, identificación del código de capacidad a ejecutar y direccionamiento al flujo propio de cada instrucción. Se muestra como parte del trabajo el diagrama principal de deliberación en la Figura 30 y los propios de cada instrucción se tienen como ANEXOS del 3 al 7. Se realiza un mapeo de cada uno de los pasos del AHPL con los puntos del diagrama de flujo para vincular la información, relacionando su contenido, esperando facilitar su entendimiento.

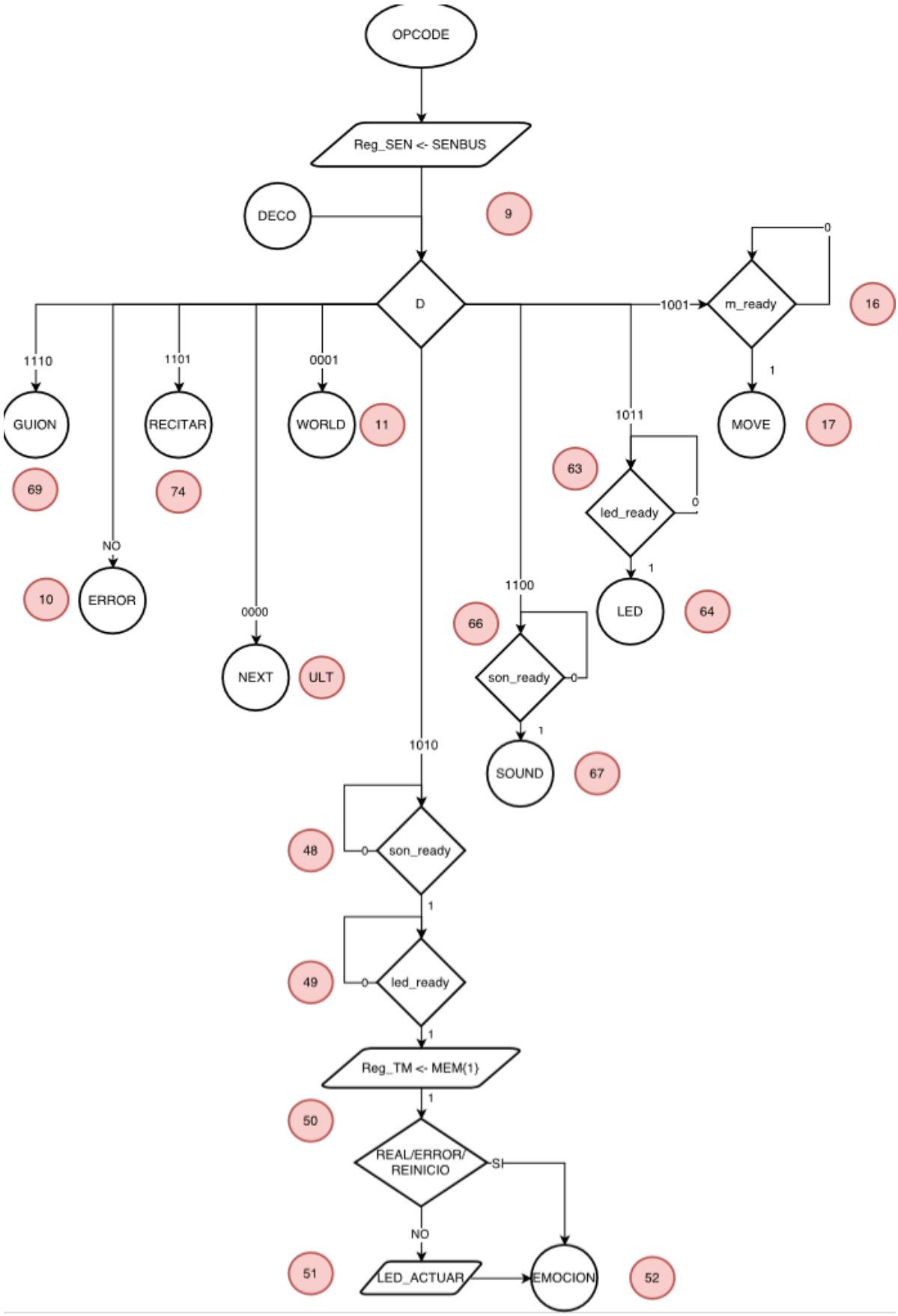

Figura 30. Diagrama de flujo OPCODE

Una vez identificados todos los caminos, se procede a hacer una descripción con lenguaje AHPL, de cada una de las transferencias requeridas y las condiciones que llevan desde y hacia cada una de ellas. De esta forma, se modelan los estados que tendrá la máquina de estados a ser implementada finalmente en VHDL sobre la FPGA y se identifican todas las posibles conexiones entre estos. (ANEXO 8)

Finalmente basado en el diseño AHPL, se realiza la implementación del agente en lenguaje VHDL sobre la Interfaz de programación del fabricante Altera: QUARTUS II. Este proceso aunque bastante facilitado gracias al trabajo realizado para el diseño del agente y sus capacidades, debe tener en cuenta las capacidades propias de la tarjeta elegida, buenas prácticas y técnicas de codificación, para lograr una adecuada asignación de recursos, esperando alcanzar el mejor desempeño posible de la plataforma. (ANEXO 9)

5. PROTOCOLO DE PRUEBAS

La validación de los desarrollos, se hace por medio de dos interfaces: una herramienta de simulación y una plataforma física básica para permitir la entrada y salida de señales a la FPGA Cyclone II. La herramienta provista por QUARTUS II para efectuar análisis de señales en tiempo: Vector Waveform Editor, la cual permite parametrizar los valores de las señales y buses de entrada, y observar los cambios en las señales y buses de salida sobre una línea de tiempo. De esta forma se observa si las señales hacia los actuadores están siendo modificadas de la forma esperada según los parámetros de creencias y las instrucciones entregadas al agente.

Es importante tener en cuenta que para posibilitar las mediciones por medio de esta herramienta, fue necesario realizar los siguientes desarrollos adicionales sobre la implementación original:

- Se instancian 3 buses de salida para pruebas no especificados dentro de la arquitectura del agente: CONTADOR, TEST1 y TEST 2, para posibilitar la visualización del estado en el que se encuentran la ejecución secuencial de la máquina de estados dentro del agente y la información que está siendo intercambiada por los registros en diferentes puntos del proceso de ejecución del agente. Este proceso afecta las mediciones añadiendo componentes adicionales a la estructura del agente, pero son requeridos para evaluar el correcto comportamiento del mismo a nivel de simulación.
- Se realiza la simulación con una velocidad de reloj de 50MHz, para posibilitar la visualización de las iteraciones en los valores de las señales. Esto debido a que el proceso de simulación se entrega en orden de nanos y micro segundos, mientras el comportamiento esperado en el agente implementado sería del orden de milisegundos y segundos.

Es de resaltar que se trabaja con lógica negada para todas las señales y buses de entrada, con el fin de evitar que valores aleatorios no deseados puedan generar comportamientos erraticos, en la eventual construcción del agente físico.

Se construye además un circuito básico para efectos prácticos de observar la implementación, que se observa en las imágenes 31 y 32. Este permite por medio de interruptores, generar las señales de entrada a la FPGA, tanto del in_BUS como de in_ready, led_ready, son_ready, m_ready y reset, y observar las salidas de las señales agent_ready, gender, la frecuencia de cambio del reloj con el que está trabajando y los buses LEDBUS, SONBUS, MXYBUS y contador por medio de LED’s.

Se realiza la grabación de un corto video que muestra el funcionamiento de la plataforma, adicionado la dirección web donde puede ser accedido.

Figura 31. Plataforma Validación desarrollos

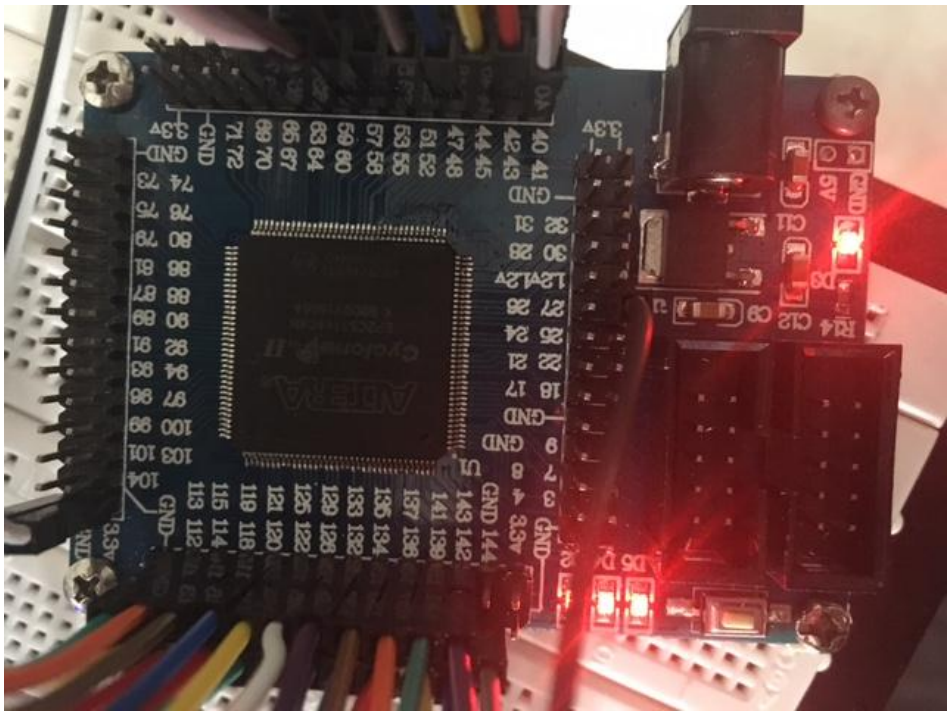

Figura 32. Tarjeta de desarrollo Altera Cyclone II

6. SIMULACION

Para dar un primer acercamiento al funcionamiento de la implementación realizada, se hace uso de la herramienta Vector Waveform que permite realizar diagramas de tiempo para observar los cambios en las señales de salida, según se modifiquen las señales de entrada. Para poder realizar este proceso, es necesario trabajar con una frecuencia mucho mayor a la planteada para la implementación física, pues la herramienta de simulación entrega resultados en ordenes de nano y micro segundos. Aunque se realizaron una serie de mediciones, se adjuntan las imágenes que mejor muestran la correcta iteración de estados, la cual a su vez valida la correcta transferencia de información dentro del código, pues es necesaria esta condición para que los cambios de estado sucedan en el orden esperado. Para observar la información almacenada en las memorias DIR y MEM, se ordena mediante la instrucción SCRIPT el almacenamiento de 2 ordenes específicas, y se evidencia luego su ejecución con la instrucción ACT.

6.1.1 WORLD

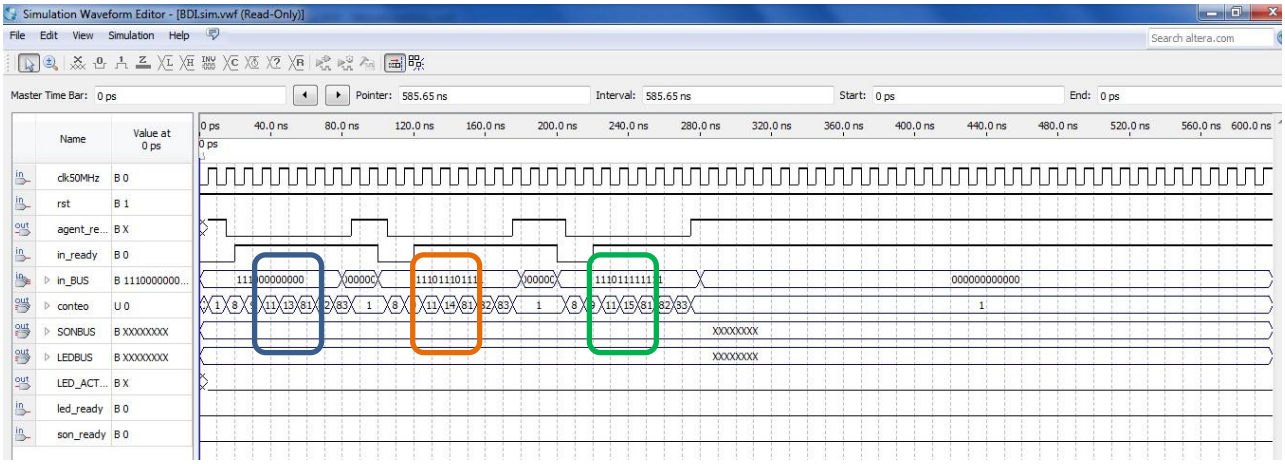

Figura 33. Simulación WORLD

Se observa en la Figura 33, con ayuda de la señal de apoyo conteo, el paso por los estados 13, 14 y 15, en los cuales secuencialmente se almacena la información correspondiente a las creencias del agente.

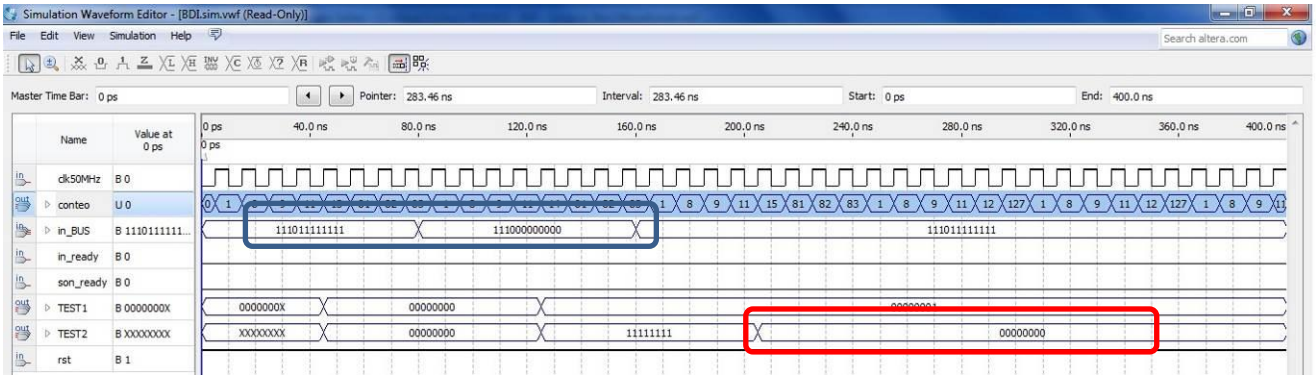

Figura 34. Simulación WORLD (2)

En la Figura 34, además del paso por los estados 13, 14 y 15 secuencialmente, se observa con la ayuda de las señales de prueba TEST2, la señal que está siendo ingresada a la memoria MEM.

6.1.2 LED

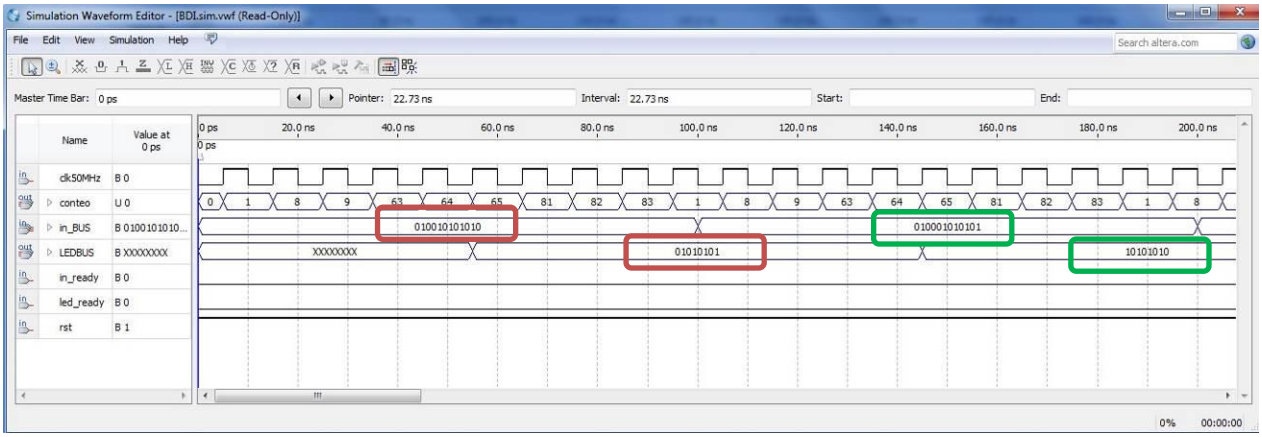

Figura 35. Simulación LED

En la figura 35, se observa la transferencia de las señales de entrada in_BUS a las señales de salida LEDBUS y la correcta secuenciación de los estados con ayuda la señal de apoyo contador.

6.1.3 SOUND

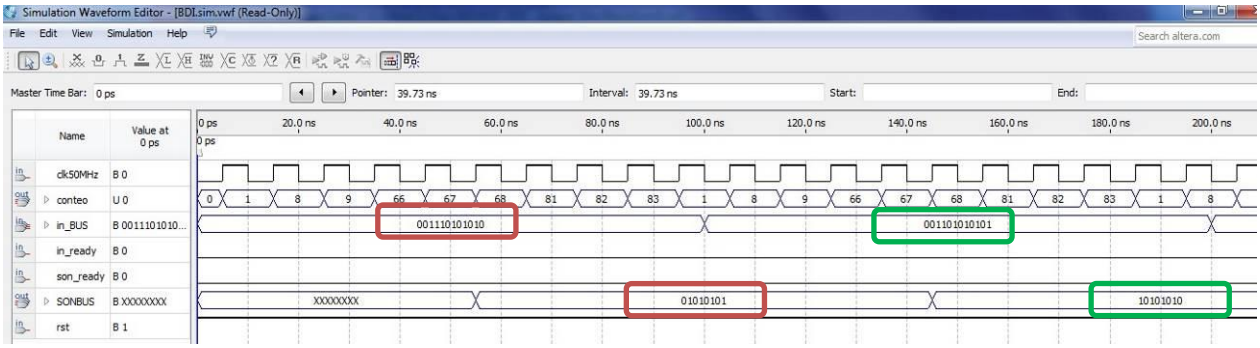

Figura 36. Simulación SOUND

En la figura 36, se observa la transferencia de las señales de entrada in_BUS a las señales de salida SONBUS y la correcta secuenciación de los estados con ayuda la señal de apoyo contador.

6.1.4 SCRIPT

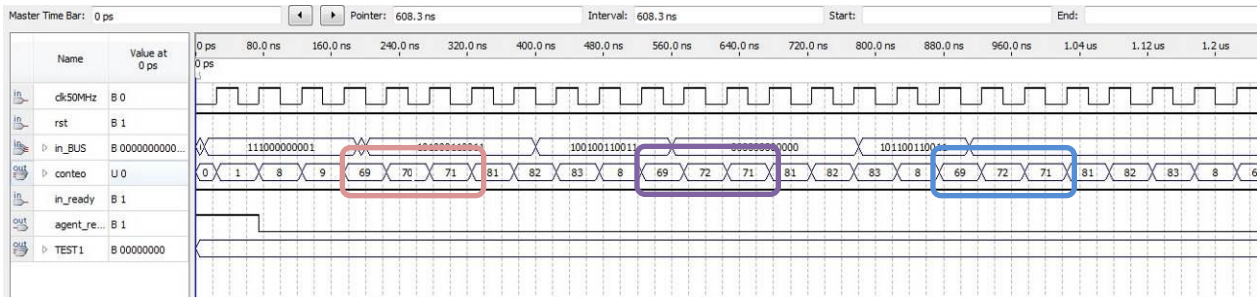

Figura 37. Simulación SCRIPT

En la figura 37, se evidencia la correcta secuenciación de los estados con ayuda la señal de apoyo contador. En este se evidencia el correcto funcionamiento de los condicionales para determinar el estado siguiente, manteniéndose dentro del ciclo de almacenamiento de instrucciones descrito por 69, 70, 71, sin pasar a la ejecución de instrucciones.

6.1.5 MOVE

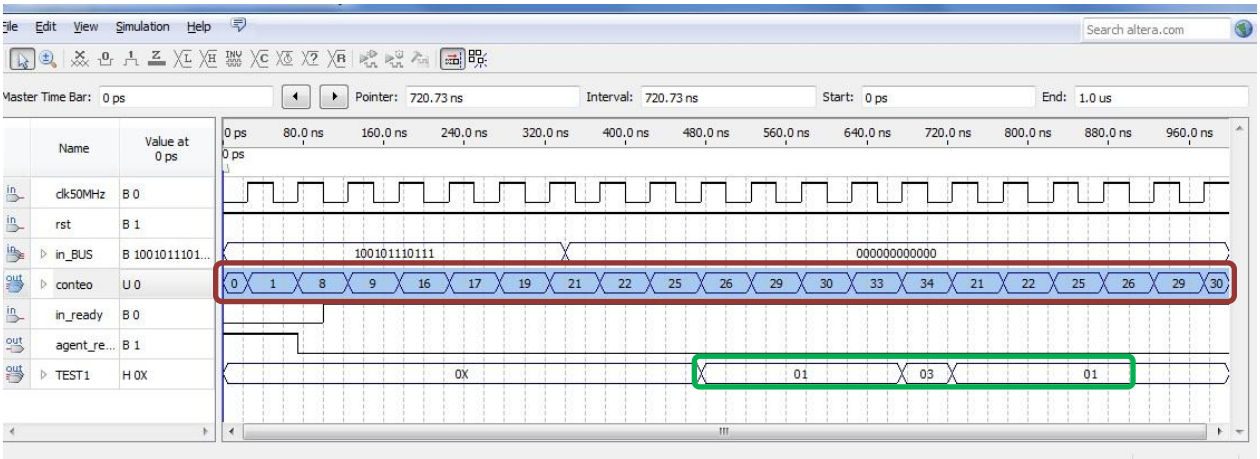

Figura 38. Simulación MOVE simple sin obstáculos

En la figura 38 se observa la correcta secuencia de estados para realizar un movimiento hacia una coordenada (7,7). Debido a las limitaciones por la imagen, solo se observan los primeros dos movimientos, determinados por los estados 16, 17, 19, validación de máximos, 21, 22, 25, 26, movimiento en X, 29, 30, 33, 34, movimiento en Y, nuevamente a 21 para moverse en X y finalmente 29 para un nuevo movimiento en Y.

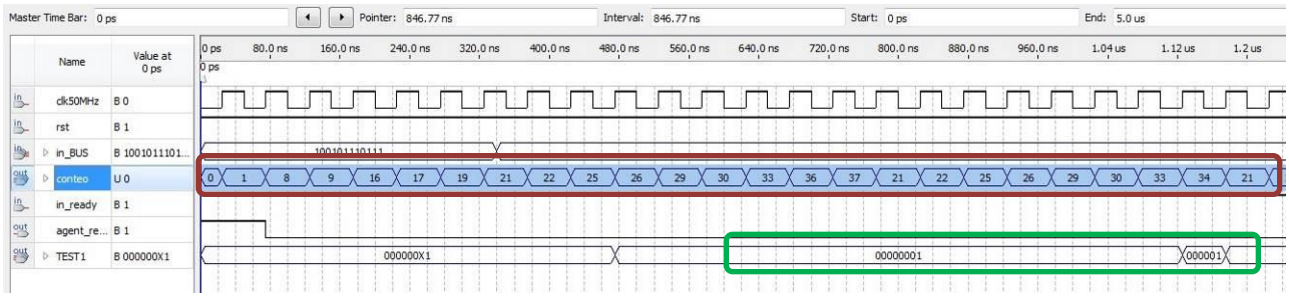

Figura 39. Simulación MOVE simple con obstáculo

En la figura 39, se observa la secuenciación de estados, esta vez teniendo un salto del 33 al 36, indicando que no fue posible realizar un movimiento en Y por lo que después de validar 37, vuelve a 21 para moverse nuevamente en X, y finalmente en estado 34, logra moverse en Y.

6.1.6 ACT

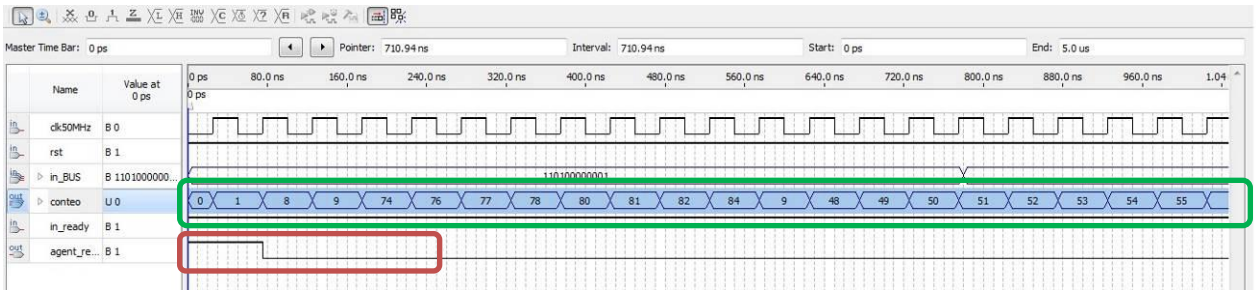

Figura 40. Simulación ACT

Concluyendo, se observa en la figura 38, el secuenciamiento de estados para ejecutar las instrucciones almacenadas para la escena 1 observando la instrucción correspondiente en in_BUS como1001 para el OPCODE correspondiente a ACT y 00000001 como el número de escena que se solicita interpretar.

Efectivamente el agente pasa al estado 74 correspondiente a ACT para levantar la bandera correspondiente, realizar la búsqueda de la dirección en memoria de la escena 1 en los estados 77 y 78, y finalizando con la ejecución de la instrucción EMOTION realizando el salto al estado 48 y los subsecuentes.

7. ANALISIS DE RESULTADOS

Por medio de las herramientas proporcionadas por el fabricante Altera, uno de los líderes mundiales en el área de implementación hardware en ámbitos profesionales y académicos, se logran identificar puntos clave que validan la consecución de los objetivos propuestos:

- Se completó el diseño efectivo de un agente con capacidades deliberativas características de un agente inteligente, teniendo en cuenta el conocimiento que tiene de sí mismo y de su entorno, observando el resultado de dichas deliberaciones en las señales de salida que se entregaba a los actuadores para realizar modificaciones que le permitieran alcanzar los objetivos planteados mediante las capacidades solicitadas.
- El modelo de componentes y arquitectura planteada, fue estructurado de tal forma que pueda ser complementado en futuros proyectos. Esto sería posible realizando relativamente pocas modificaciones sobre el tamaño de los registros para permitir un mayor número de capacidades al agente, un mayor número de interacciones con el mundo externo y de esta forma aumentar la información con la cual puede tomar decisiones para actuar. Una eventual implementación más compleja del agente, podrá y es requerida para evidenciar de mejor manera las ventajas de la implementación de hardware sobre software.
- Se implementó de manera satisfactoria, la estructura del agente diseñado sobre una plataforma Hardware FPGA Altera Cyclone II.
- Se corroboró el funcionamiento por medio de las herramientas de simulación provistas por el fabricante y ayudado por una plataforma de exteriorización básica que aunque no hacía parte del objeto del trabajo, permitió evidenciar su desarrollo.
- Se incluyó la capacidad de percibir situaciones que pueden modificar las creencias del agente en tiempo real, mediante la capacidad de movimiento basada en la lectura de los sensores perimetrales que habilitaban o restringían el mismo.
- Finalmente los requerimientos identificados para una eventual hibridación con agentes BDI en dominio software, tienen que ver en gran medida con los inconvenientes encontrados para la implementación en hardware que podrían ser resueltos de mejor manera en el dominio software:
    - La lógica de los actuadores, se decidió mantenerla externa al agente. Aunque puede tener alta o poca complejidad, en su mayor parte es reactiva y podría implementarse más fácilmente en el dominio software, tratando cada actuador como un agente BDI capaz de interpretar las instrucciones del agente actor y cuyo objetivo principal sea llevarlas a cabo.
    - El protocolo de comunicación con el usuario, se manejó de forma paralela para no añadir complejidad a la implementación del agente de forma innecesaria. Sin embargo es claro

que en una aplicación práctica, es más eficiente realizar comunicaciones con protocolos seriales, en especial si se busca la complejización del agente para manejar un mayor número de señales e información. Sin embargo, se observa la simplificación de los problemas de diseño del agente, usando paralelismo de señales, por lo que una lógica de conversión Serie- Paralelo, podría ser una implementación útil en el dominio Software, manteniendo el protocolo entre agentes de naturaleza Hardware.

Aun teniendo estos puntos en cuenta, se evidencia la posibilidad de aplicar modificaciones en el enfoque con el que se abordó el proyecto. Específicamente, es valioso resaltar que:

1. Aunque los resultados de las simulaciones generan las salidas esperadas, las pruebas sobre la plataforma FPGA Cyclone II, demuestran un comportamiento errático cuando los puertos de entrada o salida son asignados en bloques específicos de la tarjeta. Esta condición que aunque no es advertida, o por lo menos, no de una manera evidente en la documentación del fabricante, generó dificultades a la hora de corroborar los desarrollos sobre la plataforma median el circuito implementado para tal fin. Finalmente pudo ser aislado el bloque que generaba los inconvenientes de funcionamiento, y los bancos de puertos entrada/salida restantes, fueron suficientes para realizar las validaciones requeridas.
2. Las diversas formas de implementación en hardware por medio de VHDL, no permiten tener una absoluta certeza durante el desarrollo del proyecto, de estar siendo implementado de la manera más óptima posible. Es necesario un conocimiento avanzado de este lenguaje descriptivo de hardware para garantizar que las implementaciones sean interpretadas por la herramienta de una forma que permita optimizar el uso de los recursos de la tarjeta de desarrollo.
3. Al no hacer parte de los objetivos del proyecto, la plataforma de pruebas implementada para observar el comportamiento de la tarjeta de desarrollo, no tiene la robustez ideal para evidenciar el funcionamiento de los desarrollos y se evidencia sensible a valores parasito en los puertos de entrada y salida. Esto generó en repetidas ocasiones comportamientos no esperados, que fueron resueltos con reinicios del proceso y con nuevas cargas sobre la tarjeta, mostrando que dichas señales, pudieron inducir formas de programación no esperadas sobre la FPGA.
4. Aunque fue suficiente para cubrir los objetivos del proyecto, el tiempo se tornó en un factor en contra para la profundización sobre técnicas de implementación sobre FPGA basadas en VHDL. La mayor parte del tiempo de trabajo, se invirtió en la definición de las funcionalidades, la arquitectura y el diseño del agente para cubrir los objetivos planeados. Esto generó que el tiempo de implementación y prueba no fuera el esperado al plantearse el proyecto.
5. Aunque el tiempo para implementación y prueba jugo en contra durante el cierre del proyecto, quedo en evidencia que el esfuerzo invertido en el diseño de la arquitectura del agente, no fue en

vano pues permitió realizar la implementación de forma ágil, siguiendo rigurosamente el diseño planteado.

6. Queda en evidencia la gran importancia del lenguaje de descripción de transferencias de registros AHPL, para permitir una implementación fluida de los desarrollos. Aunque la documentación encontrada al respecto en general es muy poca comparada con los lenguajes de descripción de Hardware modernos VHDL y Verilog, el libro origen de AHPL fue una guía clave para lograr la implementación del diseño planteado y abordar la implementación en hardware de forma correcta.

Sobre los objetivos del proyecto:

Es claro que la dedicación de recursos que se logra con implementaciones Hardware, permite una distribución eficiente de tareas durante el tiempo de ejecución del proceso. Esto fue evidenciado en la tesis origen del presente trabajo y se logró corroborar observando la concurrencia de procesos que evidencian algunos de los flujos de simulación realizados. Sin embargo, es posible que dada la funcionalidad elegida de "teatro robótico", y su objetivo de mostrar a seres humanos su comportamiento, no sean tan evidentes los beneficios de la implementación hardware sobre el software, en gran parte por el nivel de desarrollo básico que se contempló y que la exteriorización debe ser realizada en periodos de tiempo perceptibles al ojo y entendimiento humano. Se insta a aumentar la complejidad del desarrollo aquí propuesto, para aumentar las capacidades y concurrencia del agente para evidencia de una mejor manera las bondades de la implementación Hardware.

8. CONCLUSIONES Y RECOMENDACIONES

La implementación en Hardware, trae consigo desafíos en todas las fases que comprende el desarrollo de un proyecto tecnológico de cualquier característica. Desde el diseño de los componentes, hasta la comprobación de los resultados esperados, la implementación Hardware exige un alto nivel de entendimiento de los sistemas a un nivel mucho más bajo que el dominio Software.

Quedó en evidencia la importancia de abordar este tipo de implementaciones, con un conocimiento claro de los componentes y valores de la capa lógica e incluso de la capa circuital, mencionada en el marco teórico; esto es, una base fuerte del funcionamiento de los componentes lógicos flip flops y compuertas lógicas, de sus valores binarios y las herramientas de abstracción para el manejo de dichos componentes, como las tablas de verdad y los diagramas de karnaugh. A partir de allí es imprescindible reconocer el lenguaje AHPL como el poderoso vehículo que permite plasmar las necesidades funcionales, sobre una capa de transferencia de registros y los diagramas de flujo para dar un entendimiento funcional sobre lo esperado del desarrollo. Aunque actualmente se reconoce más ampliamente los lenguajes de descripción de hardware VHDL y Verilog, sigue siendo imperativo contar en las fases de diseño con una forma eficiente de plasmar los requerimientos funcionales, pues al ser estos últimos ciertamente más limitados y complejos que los lenguajes de programación en software, requiere un mayor nivel de abstracción en su diseño para obtener los resultados esperados. No es para nada aconsejable, abordar VHDL sin una idea clara del diseño y de su funcionamiento en capa de transferencia de registros. No en implementaciones Hardware.

Finalmente, el mayor logro obtenido con el presente trabajo, fue demostrar las acertadas características que posee el enfoque BDI para abordar problemas sobre inteligencia artificial. Su forma de plantear los problemas mediante los 3 pilares Creencias, Deseos e Intensiones, permite estructurar de una forma lógica los requerimientos funcionales para observar la tecnología desde una perspectiva antropológica que facilite su interpretación. Seguir trabajando y hacer énfasis en una de las ramas de estudio de IA con diversos trabajos como el presente, nos permitirá acercarnos cada vez más a un entendimiento lo cada vez más práctico de implementación de inteligencia artificial. Se plantea un siguiente paso en el teatro robótico, aumentando la complejidad del agente aquí expuesto para manejar cada vez un número mayor de información y capacidades, y una implementación física de los actuadores propuestos para observar eventualmente, un agente actor con la capacidad de expresar sus “emociones” por medio, por ahora, de movimientos, sonidos y luces.

9. BIBLIOGRAFIA

1. Russell y Norvig, 2009, p. 2.
2. GONZALEZ, Alejandra. "DISEÑO DE SISTEMAS EMBEBIDOS COMPLEJOS A PARTIR DE AGENTES BDI HÍBRIDOS CON MIGRACIÓN DE DOMINIO" Julio 2014
3. [Wooldridge 1995] Wooldridge, M. and Jennings, N. R. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2):115–152, 1995.
4. Stuart Russell and Peter Norvig, Artificial Intelligence: A modern Arpoach, 1995.
5. HILL, Frederick. PETERSON, Gerald. DIGITAL SYSTEMS: HARDWARE ORGANIZATION AND DESIGN. Third Edition.
6. http://www.desarrolloweb.com/articulos/499.php
7. [Iglesias 1998] Iglesias Fernández, C. A.: Definición de una Metodología para el Desarrollo de Sistemas Multiagente. Tesis Doctoral. Departamento de Sistemas Telemáticos. Universidad Politécnica de Madrid. Enero 1998.
8. Yan MENG. Agent-based reconfigurable architecture for real time object tracking. 2009.
9. Anand Rao and Michael P. Georgeff, BDI Agents: From Theory to Practice
10. 9 Dominguez Carlos, Hassan Houcine, Crespo Alfns and Albaladejo José, Multicore and FPGA implementations of emotional-based agents architectures, 2014.
11. http://www.kramirez.net/SMA_Maestria/Material/Presentaciones/ArquitecturasSMA.pdf
12. Naji Hamid, Reconfigurable Parallel Data Flow Architecture, 2010.
13. Mehdi Dastani and Bas R. Steunebrink, Operational Semantics for BDI Modules in Multi-Agent Programming
14. http://quartushelp.altera.com/15.0/mergedProjects/hdl/vhdl/vhdl_pro_state_machines.htm
15. http://www.fdi.ucm.es/profesor/jpavon/doctorado/sma.pdf
16. Michal Cap, Mehdi Dastani and Maaike Harber, Belief/Goal Sharing BDI Modules.
17. Dragoni, AF. "Mental states as multi-context systems." ANNALS OF MATHEMATICS AND ARTIFICIAL INTELLIGENCE 54, nº. 4 (Diciembre 2008): 265-292
18. Thangarajah, L. Padgham, and J. Harland, "Representation and Reasoning for Goals in BDI Agents," in Proceedings of the Twenty-fifth Australasian Conference on Computer Science - Volume 4, Darlinghurst, Australia, Australia, 2012, pp. 259–265.
19. Carrillo Medina José Luis, ANÁLISIS DE METODOLOGÍAS PARA DESARROLLO DE AGENTES, CASO PRÁCTICO: BUSCADOR INTELIGENTE PARA UNA INTRANET, Riobamba, Ecuador, 2015
20. https://www.academia.edu/3130264/Introducci%C3%B3n_a_los_Agentes_Inteligentes_y_Sistemas_Multi-Agente
21. http://web.engr.oregonstate.edu/~traylor/ece474/vhdl_lectures/essential_vhdl_pdfs/essential_vhdl107-127.pdf
22. https://www.youtube.com/user/cafajar1

10. LISTA DE ANEXOS