UNIVERSIDAD AUSTRAL DE CHILE
CAMPUS PUERTO MONTT
ESCUELA DE INGENIERIA EN COMPUTACION

INVESTIGACIÓN Y DESARROLLO SISTEMA PROTOTIPO DE ASISTENCIA DOMÓTICA PARA PERSONAS CON MOVILIDAD LIMITADA

Seminario de
Titulación para optar
al título de Ingeniero en
Computación.

PROFESOR PATROCINANTE:
Sra. Claudia Zil Bontes.

PROFESOR COPATROCINANTE:
Srta. Carola Ríos Leal.

MAURICIO RAMIRO HERIQUEZ SCHOTT

PUERTO MONTT - CHILE
2005

1954

Universidad Austral de Chile
Escuela de Ingeniería en Computación

Los Pinos s/n, Balneario Pelluco
Campus Puerto Montt
Puerto Montt · Chile
Casilla 1327 · Fono: 56 65 260990
Fax: 56 65 277156
Email: ecomputa@uach.cl
www.uach.cl

Puerto Montt, 19 de abril de 2005

COMUNICACIÓN INTERNA N° 064

DE : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA DE INGENIERIA EN COMPUTACION

A : Dr. Enzo Crovetto Espinosa – **DIRECTOR CAMPUS PUERTO MONTT**
Sra. Cristina Barriga – **REGISTRO ACADEMICO**
Sra. Alba Vásquez - **ENCARGADA DE TITULACIÓN CAMPUS PUERTO MONTT**

C.c : Mauricio Henríquez Schott
Claudia Zil Bontes
Carola Ríos Leal
Moisés Coronado Delgado

MOTIVO:

Informar a usted, las calificaciones obtenidas por el alumno de Ingeniería en Computación ***Sr. Mauricio Ramiro Henríquez Schott*** Rut 13.319.538-6, en su informe de Titulación "***Investigación y Desarrollo Sistema Prototipo de Asistencia Domótica para personas con Movilidad Limitada***"

Prof. Claudia Zil Bontes	7.0
Prof. Carola Ríos Leal	7.0
Prof. Moisés Coronado Delgado	7.0
Promedio Seminario	**7.0**

Sin otro particular, le saluda atentamente,

SANDRA RUIZ AGUILAR
DIRECTORA

SRA/mva

PUERTO MONTT, 18 de abril del 2005

De : Sra. Claudia Zil Bontes
PROFESORA PATROCINANTE

A : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA INGENIERÍA EN COMPUTACIÓN

MOTIVO:

Informar a Usted, sobre seminario de titulación "**INVESTIGACIÓN Y DESARROLLO SISTEMA PROTOTIPO DE ASISTENCIA DOMÓTICA PARA PERSONAS CON MOVILIDAD LIMITADA"** del alumno **MAURICIO RAMIRO HERIQUEZ SCHOTT.**

NOTA: 70.

JUSTIFICACION:

- Producto y desarrollo innovador.
- Uso de metodología, diagramación y elementos de desarrollo completos y bien aplicados en entorno.
- Sistema integral, tanto de hardware y software, que usan muchos conocidos de momento de desarrollo

OTRAS OBSERVACIONES:

CLAUDIA ZIL BONTES
PROFESORA PATROCINANTE

PUERTO MONTT, 8 - 04 - 2005

De : Srta. Carola Ríos Leal
PROFESORA CO-PATROCINANTE

A : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA INGENIERÍA EN COMPUTACIÓN

MOTIVO:

Informar a Usted, sobre seminario de titulación **"INVESTIGACIÓN Y DESARROLLO SISTEMA PROTOTIPO DE ASISTENCIA DOMÓTICA PARA PERSONAS CON MOVILIDAD LIMITADA"** del alumno **MAURICIO RAMIRO HERIQUEZ SCHOTT.**

NOTA: 7.0

JUSTIFICACION:

Es un proyecto innovador desarrollado de una forma impecable aplicando adecuadamente distintas metodologías. La presentación es excelente.

OTRAS OBSERVACIONES:

CAROLA RIOS LEAL
PROFESORA CO-PATROCINANTE

PUERTO MONTT, 19 - abril - 2005

De : Sr. Moisés Coronado Delgado
PROFESOR INFORMANTE

A : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA INGENIERÍA EN COMPUTACIÓN

MOTIVO:

Informar a Usted, sobre seminario de titulación "**INVESTIGACIÓN Y DESARROLLO SISTEMA PROTOTIPO DE ASISTENCIA DOMÓTICA PARA PERSONAS CON MOVILIDAD LIMITADA**" del alumno **MAURICIO RAMIRO HERIQUEZ SCHOTT.**

NOTA: 7.0

JUSTIFICACION:

Muy buen Trabajo

OTRAS OBSERVACIONES:

MOISÉS CORONADO DELGADO
PROFESOR INFORMANTE

Agradecimientos

No puedo dejar pasar esta oportunidad para agradecer a todas aquellas personas que me han ayudado y confiado en mí de una u otra forma. A lo largo de mi vida a mi Madre y mis amigos, muy especialmente a aquellos que hicieron posible que mí transito por la universidad se desarrollara de forma más sencilla: Rodrigo Espinosa y Alex Segovia.

Y también por supuesto, a todos aquellos que me apoyaron de forma directa o indirecta en el desarrollo de este proyecto y que creyeron que era posible de conseguir, muy especialmente a aquellos que se involucraron activamente: Juan Carlos Gallardo, Mario Guerrero, Martín Sandoval, Fabián Oyarzún, Kevin Mellot, Marcus Müller y a mi profesora patrocinante y amiga Claudia Zil.

A todos ellos, de corazón, muchas gracias por todo.

A mi Madre, sin su paciencia y dedicación, no estaría escribiendo esto.

ÍNDICE

Síntesis en castellano

Síntesis en inglés

1. Introducción 01

2. Objetivos 04

2.1 Objetivo general 04

2.2 Objetivos específicos 04

3. Planteamiento del Problema 06

3.1 Antecedentes 06

3.2 Justificación 13

3.2.1 Situación sin Proyecto 13

3.2.2 Situación con Proyecto 14

3.3 Delimitación 14

4. Metodología 18

5. Recursos 21

5.1 Hardware 21

5.2 Software 22

6. Electrónica 26

6.1 Análisis 26

6.1.1 Circuito para manejo de señales infrarrojas “irTrans” 26

6.1.2 Interfaz de realidad virtual “NewGlove” 31

6.1.3 Circuito de control eléctrico y mecánico 33

6.2 Diseño 43

6.2.1 Diseño circuito irTrans 43

6.2.2 Diseño circuito guante realidad virtual NewGlove 46

6.2.3 Diseño circuito de control eléctrico y mecánico 49

6.3 Implementación 63

6.3.1 Implementación irTrans 64
6.3.2 Implementación interfaz realidad virtual guante NewGlove 64
6.3.3 Implementación circuito control eléctrico y mecánico 65
6.4 Pruebas 72
7. Mecánica 73
7.1 Análisis 74
7.1.1 Ecuaciones para cálculo de engranajes rectos y cremallera .. 75
7.1.2 Calculo de engranaje recto 76
7.1.3 Calculo de la cremallera o “guía” 73
7.2 Diseño 78
7.3 Implementación 81
7.4 Pruebas 89
8. Software 90
8.1 Análisis 90
8.2 Diseño 102
8.3 Implementación 108
8.4 Pruebas 155
9. Base de Datos 161
9.1 Modelo Conceptual BDSAD (“Base de Datos Sistema de Asistencia Domótica”) 161
9.1.1 Entidades 161
9.1.2 Relaciones 172
9.1.3 Identificar y asociar atributos con una entidad o relación 185
9.1.4 Determinar dominios de atributos 208
9.1.5 Determinar claves candidatas y primarias de los atributos .. 208
9.1.6 Identificar tipo de entidades de superclase y subclase 210
9.1.7 Diagrama Modelo Conceptual BDSAD 213
9.2 Modelo Lógico BDSAD (“Base de Datos Sistema de Asistencia Domótica”) 215

9.2.1 Mapeo del modelo conceptual a lógico 215
9.2.1.1 Relaciones “Muchos – A – Muchos” 215
9.2.1.2 Relaciones “Complejas” 218
9.2.1.3 Relaciones “Recursivas” 222
9.2.1.4 Relaciones con “Atributos” 223
9.2.1.5 Optimización de Relaciones 225
9.2.2 Derivar relaciones del modelo de datos lógico 229
9.2.3 Validar el modelo utilizando normalización 237
9.2.3.1 Primera Forma Normal (1FN) 237
9.2.3.2 Segunda Forma Normal (2FN) 237
9.2.3.3 Tercera Forma Normal (3FN) 242
9.2.3.4 Forma Normal Boyce-Codd (FNBC) 247
9.2.3.5 Otras Formas Normales 247
9.2.4 Transacciones de Usuario 249
9.2.5 Diagrama ER Final 256
9.2.6 Definir restricciones de integridad 258
9.2.6.1 Integridad Referencial 258
9.2.6.2 Restricciones de la Empresa 263
9.3 Construir y validar el modelo de datos lógico global 265
9.4 Traducir el modelo lógico global para el DBMS especificado 265
9.4.1 Diseñar las relaciones bases para el DBMS especificado 265
9.4.2 Diseñar las restricciones de la empresa para el DBMS especificado 266
9.4.3 Implementación 267
9.5 Diseñar representación física 292
9.5.1 Analizar transacciones 292
9.5.2 Elegir organización de archivos 292
9.5.3 Elegir índices secundarios 293
9.5.4 Introducción de redundancia controlada 293

9.5.5 Estimar los requerimientos de espacio en disco 295
9.6 Diseñar mecanismos de seguridad 296
9.6.1 Diseñar vistas de usuarios 296
9.6.2 Diseñar reglas de acceso 296
9.7 Monitoreo y refinamiento del sistema operacional 297
10. Conclusiones y/o Recomendaciones 298
11. Bibliografía 300

Anexo Digital (CD)

(Cada directorio posee un archivo con información sobre el contenido)

X:\Documentos\Electronica

X:\Documentos\Mecanica

X:\Documentos\BDSAD

X:\Documentos\DomoSoft

X:\Fuentes\BDSAD

X:\Fuentes\DomoSoft

X:\Fuentes\Electronica\NewGlove

X:\Fuentes\Electronica\irTrans

X:\Software Necesarios

Tablas

1. Grupo de colaboradores. 12
2. Hardware utilizado. 21
3. Software utilizado. 23
4. Listado de Claves Primarias y Candidatas 208

Diagramas

1. Esquema principal del sistema. 02
2. Arquitectura sistema irTrans. 30
3. Diagrama en bloques tarjeta principal circuito control eléctrico / mecánico.. 36
4. Diagrama en bloques tarjeta control motor PAP 38
5. Diagrama en bloques tarjeta de control motor DC 39
6. Diagrama en bloques tarjeta de control solenoide....40
7. Diagrama en bloques tarjeta de control salida 220V 41
8. Diagrama en bloques tarjeta de control cerradura eléctrica 42
9. Diseño electrónico dispositivo irTrans....44
10. Diseño electrónico guante realidad virtual NewGlove. 47
11. Diseño tarjeta principal circuito eléctrico mecánico. 50
12. Diseño circuito control de motores PAP. 52
13. Diseño circuito control Motores DC....54
14. Diseño circuito control salidas 220V. 56
15. Diseño circuito control solenoides. 58
16. Diseño circuito control de cerraduras eléctricas. 60
17. Diseño circuito retroalimentación. 41
18. Diseño engranaje recto para montar al eje del motor. 78
19. Diseño cremallera o guía para montar al marco de la puerta o ventana. 79
20. Diseño de puerta y ventana modelo, utilizadas durante las pruebas. .80
21. Modelo de Casos de Uso. 92
22. Modelo de Clases 103
23. Especialización Dispositivos....210
24. Generalización Parámetros 212
25. Modelo Conceptual BDSAD....214

26. Relación Muchos – A – Muchos “GuanteRV – A – TipoPatronesGuenteRV” 216
27. Relación Muchos – A – Muchos “TipoBuses – A – PuertosLPT” 217
28. Relación Compleja “TeclasAmbienteRV” 218
29. Relación Compleja “MouseAmbienteRV” 219
30. Relación Compleja “PatronesGuanteRV” 220
31. Relación Compleja “Ejecutar” 221
32. Relación Recursiva “AccionesAgrupanAcciones” 222
33. Relación con Atributos “CalibrarGuanteRV” 223
34. Relación con Atributos “SistemaTieneUsuarios” 224
35. Optimización relación “CalibrarGuanteRV” 225
36. Optimización “PatronesGuanteRV” 226
37. Optimización “DatosBuseLPT” 227
38. Optimización “Dispositivos” 228
39. Transacción configurar y calibrar GuanteRV 249
40. Transacción configurar AmbienteRV 250
41. Transacción configurar sintetizador y reconocedor de voz 251
42. Transacción manipular señales IR 252
43. Transacción configurar servidor irTrans 253
44. Transacción manipulación bytes puerto LPT 254
45. Transacción dispositivos y acciones asociadas 255
46. Transacción ejecutar acciones 256
47. Diagrama Entidad-Relación final BDSAD 257
48. Redundancia controlada tabla “PatronesGuanteRV” 294

Figuras

1. **Implementación dispositivo irTrans 64**
2. **Implementación interfaz de realidad virtual, guante NewGlove 65**
3. **Implementación tarjeta control motor PAP 66**
4. **Implementación tarjeta etapa de potencia para tarjeta control motor PAP 66**
5. **Implementación tarjeta control motor DC 67**
6. **Implementación tarjeta control salida 220V 68**
7. **Implementación tarjeta control solenoide 69**
8. **Implementación tarjeta control cerradura eléctrica 70**
9. **Implementación tarjeta principal circuito eléctrico mecánico 71**
10. **Tarjeta principal con tarjetas de expansión domóticas 72**
11. **Bendix utilizado como engranaje recto 82**
12. **Volante de motor como guía para el recorrido de la puerta o ventana 82**
13. **Motor de alza vidrio eléctrico 83**
14. **Solenoide de dos posiciones, utilizado como pestillo de ventana 84**
15. **Puerta y Ventana modelo 85**
16. **Puerta modelo con sistema motorizado montado 86**
17. **Ventana modelo con sistema motorizado montado 87**
18. **Cerradura eléctrica junto a su transformador, utilizados en conjunto con la puerta modelo 87**
19. **Sistema mecánico motorizado, para giro de cámara de vigilancia 88**
20. **Ventana configuración dispositivos irTrans conectados al sistema . 138**
21. **Ventana recepción señales infrarrojas 139**
22. **Ventana para envío de una señal infrarroja previamente almacenada 140**

23. Ventana de calibración y pruebas del guante de realidad virtual NewGlove 141
24. Ventana para calibración y pruebas del ambiente virtual 142
25. Ventana configuración y pruebas de la interfaz de reconocimiento y síntesis de voz 144
26. Ventana asociación acciones a clases y métodos expuestos 146
27. Ventana árbol de ejecución acción dispositivo 147
28. Ventana control mediante ambiente virtual 149
29. Ventana control mediante reconocimiento de voz 150
30. Ventana para monitoreo y control de fuentes de video 151
31. Ventana para control y monitoreo del puerto LPT 152
32. Usuario interactuando con el ambiente virtual mediante interfaz de realidad virtual guante NewGlove 156
33. Usuario interactuando con el sistema de reconocimiento de voz 157
34. Envío y recepción de señal infrarroja 158
35. Envío de señal IR para control de dispositivo 159
36. Pruebas de software y componentes mecánicos 160

Síntesis

El uso de sillas de ruedas, muletas, andadores u otro tipo de aparatos para la asistencia en el desplazamiento, hacen que las personas que presentan discapacidades, ya sea de forma permanente o transitoria, exhiban diferentes tipos de dificultades respecto al quehacer cotidiano al interior de su residencia o lugar de trabajo. Por lo general, este tipo de dificultades existe debido a la disposición de los distintos aparatos con los cuales se debe interactuar y que no consideran su utilización por parte de personas con este tipo de limitaciones.

Teniendo presente este contexto se analizó, diseñó e implementó un sistema prototipo de asistencia Domótica, el cual permite el control y automatización de distintos aspectos al interior del entorno de un usuario que presenta limitaciones motoras.

Si bien, los dispositivos y aparatos con los cuales se debe interactuar al interior de un ambiente, como el de residencia o lugar de trabajo, son muchos e innumerables, este proyecto se concentró en dar una solución práctica a aquellos que se consideran de uso común y frecuente. Debido a lo anterior, el sistema cuenta con un dispositivo para manejo de señales infrarrojas, el cual al ser conectado a una computadora, permite la lectura y emisión de señales de este tipo, permitiendo el control de todo tipo de aparatos que funcionen mediante estas señales como: Televisores, Reproductores de DVD y Música, ciertos equipos de iluminación, etc.

Además, el accionar mecánico de diversos aparatos como: puertas y ventanas, cerraduras y cerrojos, tomas de corriente, etcétera, hizo tremendamente necesario el desarrollo de un circuito de control eléctrico y mecánico capaz de automatizar este tipo de mecanismos.

Puesto que las distintas discapacidades presentan dificultades y limitaciones diferentes, es que esta implementación cuenta con dos interfaces alternativas al teclado y ratón tradicional para el control del sistema, y por ende, del entorno de la persona. La primera es una interfaz de realidad virtual controlable mediante el Teclado/Ratón, o a través de un guante de realidad virtual configurable a las necesidades y limitaciones particulares del usuario. La segunda es una interfaz de reconocimiento de voz que permite al usuario dictar comandos verbales para la ejecución de diferentes acciones en su entorno.

La construcción del software y la base de datos se hizo fundamental para aglutinar y coordinar todos los componentes del sistema, el software se desarrolló mediante una orientación a objetos y metodología R.U.P, y la base de datos siguió las técnicas propuestas por Connolly.

Si bien este proyecto se enmarca dentro del contexto de una investigación, los resultados de ésta son considerables, ya que se logró cumplir con todos los objetivos propuestos así como también dar forma a un sistema completamente funcional y de “Arquitectura Abierta” que permitirá nuevos desarrollos futuros en esta área.

Abstract

The use of wheel chairs, crutches, walking frames, and other types of appliances used to help the walking movement, make people that show disability; in a permanent or transitory way, show certain kinds of difficulties regarding their everyday life indoors. Generally, these kinds of difficulties exist due to the availability of different appliances only designed for people that do not show any kind of physical limitation.

Taking into account this background, a prototype system of Domotic assistance was analysed, designed, and implemented in order to automate certain aspects inside the environment of a user with motor limitations.

Although devices and appliances that a user must use indoors, (inside his/her residence, or at his/her work place) are many and countless, this project focused on giving a practical solution to those that are considered of a frequent and common use. Considering the latter, the system has a device used in the control of infrared signals which, by being connected to a computer, allow reading, and emission of them. This fact allows control over any kind of appliances that use this type of signals, such as: TV sets, DVD and music players, some illumination devices, etc.

Also, the mechanic manipulation of different appliances such as: doors, windows, locking and bolts, electric sockets, etc, made absolutely necessary the

development of an electric and mechanic control circuit able to automate these mechanisms.

Since different types of disabilities show difficulties and different kinds of limitations, this implementation has two interfaces for the control of the system (alternative for the traditional keyboard and mouse) that may help a person in his/her life indoors. The first is a virtual reality interface manageable through the Keyboard/Mouse, or through a dataglove that can be configured according to the personal needs of the user. The second is a voice recognition interface that allows the user to generate verbal orders in order to activate any of the appliances indoors.

The software development and database were fundamental to coordinate and put together all the components of the system. The software was developed through an object orientation and R.U.P methodology. The database followed the techniques proposed by Connolly.

Although this project takes part within an investigation context, the results obtained are considerable for further analysis. All the objectives proposed were aimed, and it was possible to create a functional system of “Open Architecture” that will allow new future developments in this area.

1. Introducción

En Chile el número de personas discapacitadas corresponde al 12.9% de la población del país [FONADIS.2005], es decir que 2.068.072 personas de un total de 16.031.565, presentan algún tipo de discapacidad permanente. Si a esto, además, se le suma el número de personas que, producto de la edad, accidentes o enfermedades, presentan algún tipo de discapacidad transitoria, como fracturas, reposo médico, etc., se puede concluir que un grupo importante de la población presenta dificultades de diverso tipo. Si bien, cada discapacidad plantea una serie de retos distintos en el quehacer cotidiano a la persona que la padece, sin duda alguna que una de las problemáticas más recurrentes, sobre todo en los casos de personas afectadas con discapacidades asociadas al desplazamiento, es la de la interacción del individuo con su entorno. Esto se debe principalmente a la poca o difícil movilidad asociada con el reposo, fracturas o vejez, o al uso de aparatos como sillas de ruedas, muletas, andadores, etc.

Normalmente, el ambiente cerrado con el que más frecuentemente interactúan las personas, tanto discapacitadas como sanas, es el de su vivienda. Hoy en día, generalmente, estos entornos están pensados para ser

utilizados por personas que no presentan dificultades al desplazarse. En el quehacer diario, una persona saludable interactúa con los objetos de su residencia de manera natural y simple, puesto que el medio en el que se encuentra fue ideado y construido para ser utilizado de esta forma. Sin embargo, para las personas discapacitadas con dificultades motoras, el simple hecho de encender el interruptor de la luz, o alcanzar el control remoto de algún dispositivo, puede tornarse en una tarea complicada.

Desde hace algunos años, en distintas partes del mundo, se ha estado desarrollando un área de la informática, que en conjunto con otras disciplinas tales como mecánica, electrónica, telefonía, etc., se aboca a la automatización de residencias o “Domótica”. Si bien esta tecnología podría ayudar a desenvolverse de mejor forma en sus hogares a las personas discapacitadas, en la actualidad estos sistemas se centran básicamente en aumentar o mejorar el confort de personas que por lo general no presentan dificultades serias de desplazamiento.

Además, la mayoría de estos sistemas o “Kits de Domótica”, existentes hoy en el mercado, son difíciles de instalar, poseen una serie de requerimientos previos a la instalación, se encuentran en tempranas etapas de desarrollo o incluso muchos de ellos son solamente prototipos para investigación. A esto, se le debe agregar el alto costo de adquisición como de instalación que

usualmente está asociado a estos sistemas. También se debe considerar que los sistemas actuales de domótica, al usar protocolos específicamente creados para ellos como X-10, EIB o Lonworks, por lo general obligan a utilizar ciertos dispositivos, como lámparas, tomas de corriente, TV, VCR, DVD, etc., que cumplan con las características propias del protocolo de comunicación que se va a emplear, lo cual se traduce en un aumento del costo de implementación de este tipo de soluciones.

De acuerdo a lo anterior, y para que la domótica pueda presentarse como una ayuda real para personas discapacitadas, es que se debe investigar y desarrollar un sistema de automatización pensado y enfocado en personas que posean movilidad limitada y que requieran un medio como éste para lograr un desenvolvimiento más natural dentro del ambiente de su residencia.

2. Objetivos

2.1 Objetivo general

Diseñar y construir un sistema de Domótica que permita mejorar la calidad de vida de personas con limitaciones severas de movimientos (enfermedades musculares, accidentes, vejez, reposo médico, etc.), ampliando su rango de control sobre distintos aspectos de su vivienda y mejorando su independencia.

2.2 Objetivos específicos

- Lograr desarrollar un sistema de automatización de casas de costo accesible y de instalación lo más sencillo posible.

- Poseer distintas interfaces de control para este sistema, que permitan adecuarse de forma separada o en conjunto a las necesidades particulares de cada persona (reconocimiento de voz, dispositivos de realidad virtual, etc.).

- Adaptar distintos componentes existentes hoy en el mercado o de relativamente fácil construcción, con el fin de integrarlos entre sí para lograr este nuevo objetivo de asistencia domótica.

- Utilizar, dentro de lo posible, la mayor cantidad de aparatos comúnmente disponibles en el hogar, evitando la adquisición de dispositivos especiales.

- Trabajar en un ambiente de arquitectura abierta, en donde tanto el software como el hardware (circuitos electrónicos, esquemas mecánicos, etc.), puedan ser liberados, permitiendo su duplicación por parte de personas que lo requieran y de igual forma continuar con su desarrollo y mejoras a futuro.

3. Planteamiento del Problema

3.1 Antecedentes

La definición de discapacidad es [I.N.E.1999]:

"Toda limitación grave que afecta en forma permanente al que la padece en cualquier actividad. Tiene su origen en una deficiencia. Se considera permanente si dura uno o más años.

La deficiencia se define como cualquier pérdida o anomalía de un órgano o de la función propia de éste. Ejemplos: ausencia de una mano, ceguera, sordera, retraso mental."

Se hace claro que dentro de este grupo de la población es frecuente encontrarse con dificultades de interacción con los objetos del hogar, como consecuencia directa o indirecta de la condición particular del individuo. Es así, que para las personas con alguna discapacidad motora, que deben utilizar aparatos para la asistencia en el desplazamiento como, sillas de ruedas, muletas, andadores, etc., el accionar cotidiano de la vivienda puede tornarse en una labor complicada. Lo anterior también es válido para las personas afectadas por discapacidades transitorias, que al tener que guardar reposo médico o debido a fracturas ven restringido su porcentaje de movilidad.

Labores sencillas, como presionar el interruptor de una luz, abrir una puerta o ventana, utilizar una toma de corriente o alcanzar un control remoto, pueden transformarse en tareas complicadas cuando se utiliza una silla de

ruedas o andador o si se debe permanecer inmóvil debido al reposo médico o a fracturas. Esto ocurre principalmente producto de la posición del objeto o dispositivo en la vivienda (interruptores de luz, tomas de corriente, etc.), o a la interacción mecánica que se debe realizar para su manipulación (puertas, ventanas, etc.), donde no se considera la utilización de éstos por personas con limitaciones de movilidad.

En la actualidad, la domótica, permite el control y automatización de una serie de dispositivos y acciones tanto al interior como en el exterior del hogar, sin embargo, su principal enfoque es el de brindar un mayor confort al usuario de estos sistemas, abarcando como público objetivo solo a personas que no presentan dificultades motoras. La domótica es un área de reciente desarrollo y por lo tanto existen una serie de casos y situaciones que los sistemas de este tipo no abordan, ejemplo de esto es la manipulación de los kits de domótica por personas de movilidad restringida. Debido a esto último, es que recientemente, en diciembre de 2003, la empresa “CasaDomo” invitó formalmente al sector tecnológico a comprometerse con las personas para las que la tecnología es realmente crucial, y poder de esta forma lograr resolver los problemas cotidianos desde el enfoque de personas discapacitadas, por lo tanto, no se conocen otros esfuerzos serios o iniciativas anteriores que ahonden en este problema.

Ante esta situación, la solución que aquí se plantea, es la de investigar, diseñar y construir un sistema de domótica básico, de bajo costo y de relativamente fácil instalación, que se enfoque en su utilización por parte de personas con movilidad limitada y que aborde las tareas frecuentes pero fundamentales que se encuentran en la interacción con la residencia. Tal como se mencionó, este sistema debe orientarse a resolver los problemas cotidianos que pudiesen presentársele a una persona discapacitada, tales como: abrir una puerta, encender el televisor o una luz, etc. Para que este sistema pueda ser utilizado por un grupo amplio de personas discapacitadas y debido a que éstas presentan distintas restricciones de movilidad, es que el sistema debe proveer una serie de interfaces para su utilización, así como un control total desde un lugar central de la vivienda, como puede ser una cama o sillón.

Para lograr esto, es esencial que el sistema incluya una serie de dispositivos electrónicos, mecánicos, hardware de computadora y el software necesario para controlar todo. El diagrama Nº 1 muestra los diferentes componentes y su labor dentro del sistema propuesto.

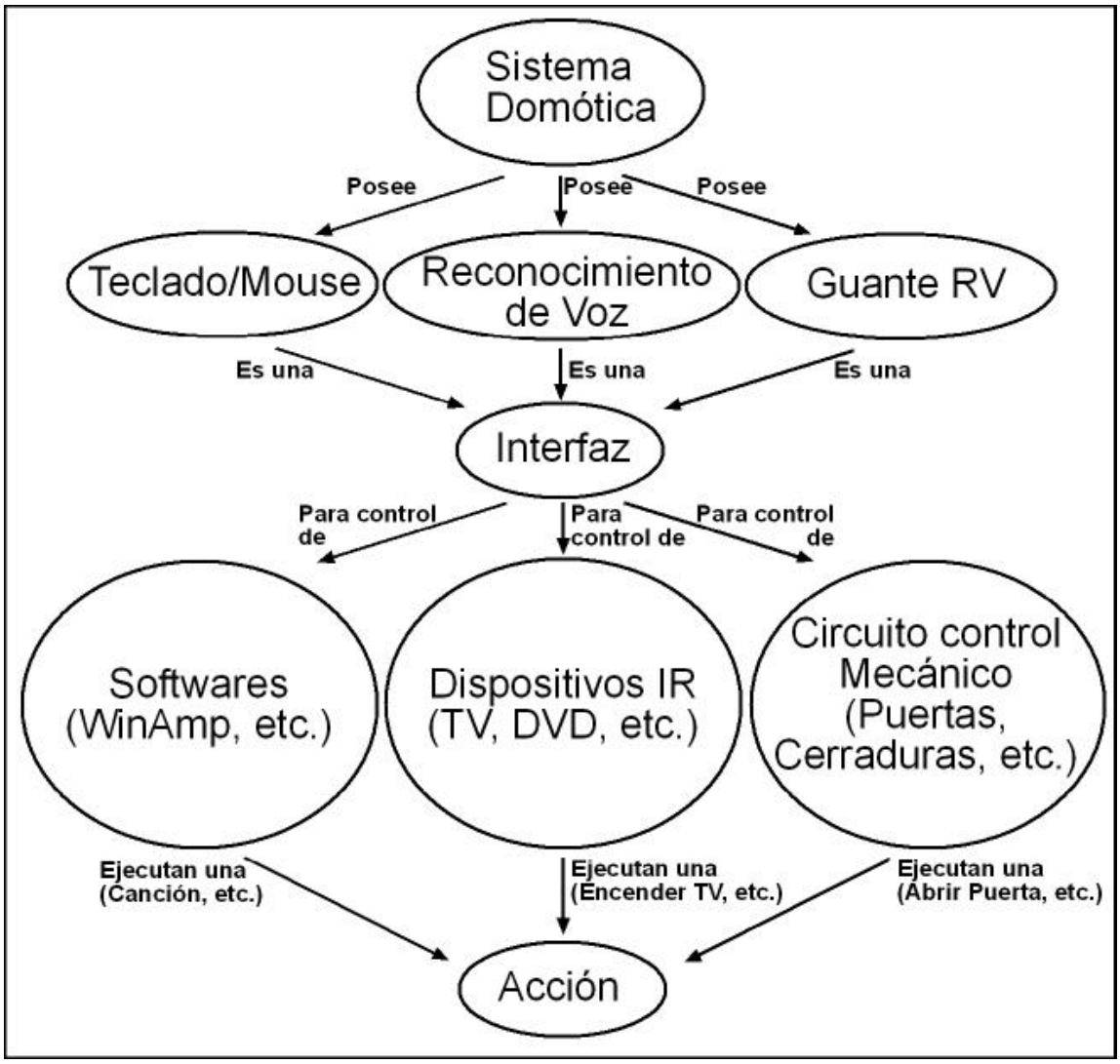

Diagrama Nº 1: Esquema principal del sistema.

Tal como se muestra en el diagrama Nº 1, existen una serie de componentes que interactúan entre sí para la ejecución de una acción determinada, la definición en concreto de cada componente y su labor es la siguiente:

- **“Sistema Domótica”:** Sistema completo, que abarca todos los componentes empleados.

- **“Teclado/Mouse”:** Interfaz tradicional para control de la computadora y por ende del sistema.

- **“Reconocimiento de Voz”:** Interfaz de control mediante comandos verbales. Esta interfaz es útil en los casos en los cuales es imposible utilizar el teclado/mouse, o el guante RV.

- **“Guante RV”:** Dispositivo de realidad virtual que se presenta como alternativa para aquellas personas que no puedan utilizar el teclado/mouse o el reconocimiento de voz.

- **“Software”:** Cualquier software que sea manipulable mediante llamadas a funciones de la API de Windows.

- **“Dispositivos IR”:** Dispositivo para control de señales infrarrojas, el cual permite la manipulación de aparatos que posean controles remotos de este tipo (TV, VCR, DVD, etc.)

- **“Circuito control Mecánico”:** Circuito que permite la manipulación de una serie de acciones mecánicas como motores, interruptores, etc., los cuales pueden ser utilizados para el manejo de puertas, ventanas, cerraduras, etc.

Si bien una iniciativa de este tipo, debido a las diversas disciplinas empleadas, requiere de la colaboración de expertos en distintas áreas, para este proyecto en particular no se creó un equipo formal de desarrollo. Solo se recurrió a colaboradores puntuales, en diversas partes del mundo, para la fabricación de los distintos componentes electrónicos y mecánicos que se utilizarán. La tabla Nº 1 muestra a las personas involucradas y su participación:

Tabla Nº 1: Grupo de colaboradores.

Nombre	Colaboración	País
Juan Carlos Gallardo	Diseño y construcción de circuitos electrónicos para control de partes mecánicas.	Chile
Mario Guerrero	Diseño y construcción de partes mecánicas.	Chile
Kevin Mellot	Diseño y construcción “Guante de Realidad Virtual”.	Estados Unidos
Marcus Müller	Diseño y construcción de circuito de control de dispositivos IR.	Alemania

La participación de este grupo de personas, permite que el sistema se desarrolle en un ambiente de arquitectura abierta, en donde cada colaborador libera las especificaciones de su aporte de manera de poder ser utilizado por algún otro miembro del grupo y permitir de esta forma mejoras a futuro, lo cual es uno de los objetivos de esta investigación.

El alumno participa en las etapas de análisis, diseño, implementación y pruebas funcionales del software necesario para el desarrollo de este sistema, así como en la conceptualización y requerimientos de los distintos componentes, tanto electrónicos como mecánicos, que se emplearán para este

proyecto. El alumno no desarrolla el motor de reconocimiento de voz (SAPI V4) y algunos controladores de dispositivos.

3.2 Justificación

3.2.1 Situación sin Proyecto

Sin la realización de este proyecto, es probable que los sistemas de domótica actuales tarden algún tiempo en madurar lo suficiente como para presentarse como una alternativa de ayuda real para las personas discapacitadas, haciendo que los problemas cotidianos que éstas presentan, al interactuar en el ambiente de su casa, queden sin resolver.

Tal como se mencionó anteriormente, los sistemas actuales de este tipo, por lo general, no permiten la utilización de los artefactos que ya se encuentran en la residencia, y por lo tanto se deben adquirir nuevos, que cumplan con las especificaciones y requerimientos del protocolo que se desea emplear. Sin duda esto último aumenta los costos asociados cuando se consideran este tipo de soluciones.

3.2.2 Situación con Proyecto

El beneficio de este sistema es que permitirá a la persona discapacitada, aumentar su grado de autonomía y reducir la necesidad de supervisión constante, mejorando de esta forma su calidad de vida.

Además, el sistema será de relativamente bajo costo, permitirá la utilización, dentro de lo posible, de los dispositivos que ya se encuentran en el hogar, lo cual reducirá los requerimientos de nueva infraestructura y se plantea como un sistema de “Arquitectura Abierta”, lo que permitirá su duplicación y utilización parcial o total por cualquier persona que lo necesite.

3.3 Delimitación

- Puesto que el sistema propuesto se enmarca dentro de una investigación, es que no se incluyen pruebas formales de sistema, ya que el resultado sólo es un software “Prototipo”, y sólo se realizarán pruebas funcionales. Esto debido principalmente a que este software no pasará por la etapa de implantación y el tiempo empleado en resolver problemas puntuales, que no pongan en riesgo la funcionalidad, pueden retrasar toda la investigación en general.

- Si bien, el guante de realidad virtual fue concebido para ser utilizado por un amplio número de personas con distintas restricciones de movilidad, es posible que en casos extremos o particulares, este dispositivo no pueda ser empleado de manera satisfactoria debido a limitaciones propias a la electrónica de este.

- Respecto a la interfaz de reconocimiento de voz, se debe decir que puede no ser utilizable en caso de personas que presenten ciertos timbres o temblores en la voz producto de la vejez o enfermedad.

- El dispositivo electrónico para la manipulación de señales infrarrojas, en teoría debiera funcionar con cualquier aparato que utilice este tipo de señal. Sin embargo en casos especiales, como las señales empleadas por empresas locales, podrían no ser compatibles con este dispositivo. Además, existe una restricción asociada al retorno de información sobre algunos estados en los aparatos que utilicen señales de infrarrojo, esto se explica de mejor forma a través de un ejemplo: “Usualmente los televisores y otros dispositivos utilizan una misma señal de infrarrojo para conmutar un estado, como ocurre con el encendido/apagado y algunos otros. Si el usuario, a través del sistema, enciende el televisor, pero físicamente alguien lo apaga, el sistema seguirá informando que el

televisor se encuentra encendido y la persona deberá corregir manualmente el estado del aparato dentro del sistema". Este fenómeno se suscita ya que no existe una comunicación bidireccional entre el sistema y este tipo de aparatos, y aunque es posible encontrar alternativas para evitarlo, estas son difíciles de implementar y pueden aumentar excesivamente los costos asociados al proyecto, lo cual se debe tratar de evitar sobre todo considerando que ésta es una dificultad menor.

- Tal como se mencionó anteriormente, cabe destacar, que el desarrollo de este sistema se enmarca dentro del contexto de una investigación, por lo cual, sólo se desarrollarán las funcionalidades necesarias para demostrar que los conceptos empleados son válidos y que es posible la incorporación y manipulación de los distintos componentes antes mencionados. Por lo tanto, no presentará todas las características que se encuentran habitualmente en un software comercial, como administración de seguridad, configuraciones asociadas a la personalización del software (imágenes de fondo, colores, fuentes y tamaños de letra, etc.), así como tampoco algunas operaciones comunes con la base de datos (updates, deletes, etc.), que no aporten nada significativo para la demostración de algún punto, ya que para efectos del desarrollo de las funcionalidades principales de la investigación, estas

operaciones se pueden realizar directamente en el DBMS. Sin embargo el modelo de datos entregado será lo más completo posible, lo cual permitirá continuar con el desarrollo de nuevas versiones prototipo de este sistema.

- Finalmente, y a pesar de que este sistema ha sido pensado para ser utilizado por un amplio número de personas con distintos tipos de discapacidades, en algunos casos extremos, como es el caso de personas ciegas sordomudas, este sistema podría no ajustarse a sus necesidades particulares, debido principalmente al importante número de restricciones que éstas presentan.

4. Metodología

Puesto que uno de los objetivos de este proyecto es que pueda seguir evolucionando con el tiempo, se escogió desarrollar la parte del software, mediante programación orientada a objetos, lo que permitirá una mejor comprensión del código, mayor facilidad de corrección y mantención, y la posibilidad de permitir versiones de este para distintas plataformas. De acuerdo a esto, entonces, se escogió R.U.P (*"Rational Unified Process"* o "Proceso Unificado de Rational") como metodología de desarrollo para el sistema, ya que es la que más se enfoca al desarrollo orientado a objetos.

R.U.P (también conocido como "Proceso Unificado"), sus actividades y su ciclo de vida, se definen como ["El Proceso Unificado de Desarrollo de Software" 2000]:

R.U.P: El Proceso Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

La vida del Proceso Unificado: El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes.

Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición. Cada fase se subdivide a su vez en iteraciones.

Fases dentro de un ciclo: Cada ciclo se desarrolla a lo largo del tiempo. Este tiempo, a su vez, se divide en cuatro fases. A través de una secuencia de modelos, los implicados visualizan lo que esta sucediendo en esas fases. Dentro de cada fase, los directores o los desarrolladores pueden descomponer adicionalmente el trabajo en iteraciones con sus incrementos resultantes. Cada fase termina con un hito. Cada hito se determina por la disponibilidad de un conjunto de artefactos: es decir, ciertos modelos o documentos han sido desarrollados hasta alcanzar un estado predefinido.

De acuerdo a lo anterior, el “Proceso Unificado” se presenta como una alternativa ideal para el desarrollo de un software orientado a la investigación como este, ya que la vida del sistema se puede dividir en ciclos, y este seminario de titulación sería el primero de ellos, para en un futuro incorporar nuevos ciclos que agreguen o mejoren características del software, además, estos ciclos se subdividen en las fases de Inicio, Elaboración, Construcción y Transición, los cuales representarán la construcción de una o más clases dentro del esquema general, cada una de estas clases engloban una serie de casos de uso partiendo desde los mas críticos hasta los mas específicos, y además, permitiendo que cada fase se subdivida a su vez en la cantidad de iteraciones necesarias para lograr los objetivos. Esto es de mucha utilidad para sistemas de este tipo, ya que de esta forma, la metodología permite que se puedan considerar inicialmente, para cada clase, solo los casos de uso más críticos que son importantes de desarrollar para probar la viabilidad de algún concepto en particular, iterando una o mas veces de acuerdo a las necesidades del

problema y luego abordar otro ciclo que incluya nuevas problemáticas a resolver.

5. Recursos

5.1 Hardware

La elección del hardware de computadora apropiado se realizó tomando en cuenta los requerimientos mínimos para la correcta ejecución de las herramientas de desarrollo, así como también la disponibilidad de estos equipos por parte del alumno. El hardware electrónico adicional se escogió de acuerdo a diversos factores, como precio, utilidad, diseño libre, etc. La tabla Nº 3 muestra el hardware empleado y su justificación principal.

Tabla Nº 2: Hardware utilizado.

Nombre	Descripción	Justificación	Provee
Computador para pruebas	Intel Pentium II 350Mhz, memoria de 160MB RAM, Disco duro de 20Gb, Tarjeta de sonido SoundBlaster AWE64, Unidad de CD-RW AOpen JustLink 20x10x40, Tarjeta de video ATI All-In-Wonder 128 Pro, Micrófono unidireccional	Computador existente con capacidad suficiente para ejecutar el sistema propuesto.	Alumno
Computador para desarrollo	Intel Pentium IV 1,7Ghz, Memoria de 384MB RAM, Disco duro de 45Gb, Disco duro 2,5Gb, Tarjeta de sonido SoundBlaster Audigy Live 32, Unidad de CD-ROM LG 52X,	Computador disponible con capacidad suficiente para llevar a cabo el desarrollo de este	Alumno

	Unidad de CD-RW LG 52x24x52, Unidad de DVD Samsung 16X, Tarjeta de video ATI All-In-Wonder Rodeon 7500, Micrófono unidireccional, Cámara Digital/Web USB Genius SE402.	proyecto	
irTrans	Dispositivo electrónico para la manipulación y control de señales infrarrojas.	Junto con este dispositivo, se entregan todos los esquemas electrónicos, listado de componentes y software necesario para su duplicación.	Marcus Müller
NewGlove	Guante de Realidad Virtual.	Junto con este dispositivo, se entregan todos los esquemas electrónicos, listado de componentes y software necesario para su duplicación.	Kevin Mellot

5.2 Software

Al igual que lo hecho con el hardware, se presenta el software utilizado en el desarrollo y el adecuado para un correcto funcionamiento del sistema, justificando su elección.

Tabla Nº 3: Software utilizado.

Nombre	Descripción	Justificación
Windows XP Professional	Sistema operativo empleado.	Sistema operativo estable que permite la ejecución de las herramientas de desarrollo empleadas.
Rational Rose XDE DevPlus	Utilizado en el proceso de diseño (RUP con UML) y construcción de la estructura general de la aplicación.	Software que se integra con Visual Studio .NET 2003 y que permite el desarrollo mediante la metodología escogida.
Sybase ASE (Adaptive Server Enterpriese) 12.5	DBMS empleado para la manipulación de los datos	DBMS que se integra con Rational Rose XDE DevPlus y también es utilizable por Visual Basic .NET.
Visual Basic .NET	Entorno de desarrollo y lenguaje de programación totalmente orientado a objetos con el cual se construirá la aplicación.	Lenguaje provisto por Visual Estudio .NET 2003, que permite un desarrollo orientado a objetos y que además presenta la posibilidad a futuro de adaptación para otras plataformas mediante el proyecto “Mono” *.
Microsoft SAPI (Speech Application Program Interface) V4	Software para el reconocimiento y síntesis de voz que se utilizado para la interfaz de voz del sistema.	Puesto que Microsoft se aboca actualmente a SAPI V5, esta versión ha sido liberada para usos no comerciales.
OpenGl (Mediante la implementación de TAO **)	Librerías graficas con las cuales se construirán los modelos 3D que se utilizarán en la aplicación para interactuar con el sistema.	Estas librerías, además de ser gratuitas, son multiplataforma, lo cual permite una más fácil traducción del código al cambiar de plataforma.

Nombre	Descripción	Justificación
OpenAl (Mediante la implementación de TAO)	Librerías para manejo de audio con las cuales se manipularán los sonidos que se utilizarán en la aplicación para interactuar con el sistema.	Estas librerías, además de ser gratuitas, son multiplataforma, lo cual permite una más fácil traducción del código al cambiar de plataforma.
CircuitMaker	Software utilizado para el diseño y análisis de circuitos electrónicos.	Software conocido y utilizado por los desarrolladores de los circuitos electrónicos.
Protel DXP	Software utilizado para el diseño y simulación de circuitos electrónicos.	Software conocido y utilizado por los desarrolladores de los circuitos electrónicos.

(*) "Mono": El proyecto "Mono" (http://www.mono-project.com) es una iniciativa de la organización "Ximian" (http://www.ximian.com), el cual pretende crear un "Framework", gratuito, que permita compilar aplicaciones creadas en lenguajes ".NET" en plataforma Linux y viceversa. Actualmente el compilador de C# para Linux se encuentra en la etapas finales de desarrollo y se esta comenzando a trabajar en el compilador para Visual Basic .NET, lo cual podría permitir crear una versión a futuro de este sistema para la plataforma Linux.

(**) "TAO": "TAO" (http://www.taoframework.com) es la implementación de las librerías que permiten la utilización de OpenGL y OpenAL en lenguajes .NET. Estas librerías son de distribución gratuita y son compatibles con el proyecto "Mono".

En las siguientes secciones, se explica en detalle los distintos componentes que fueron necesarios para el desarrollo de esta investigación, organizándolos de la siguiente manera:

En el punto 6 se presentará todo lo relacionado con la electrónica empleada en esta investigación, abarcando el dispositivo para control de señales infrarrojas “irTrans”, la interfaz de realidad virtual guante “NewGlove”, y finalmente el circuito de control Eléctrico/Mecánico.

El punto 7 abordará lo competente a la mecánica necesaria para la automatización de distintos componentes como: Puerta y Ventana modelo, cerraduras y cerrojos, sistema motorizado de giro para cámara de vigilancia, etc.

A continuación, en el punto 8, se tratará todo lo concerniente al desarrollo he implementación del software que involucra el sistema, destacando las secciones mas importantes de éste, y basando esta documentación en lo propuesto por la metodología R.U.P.

Finalmente en el punto 9, se abordará el desarrollo de la base de datos utilizada por el sistema, tomando como metodología para este propósito, la propuesta por Connolly.

6. Electrónica

En un proyecto de este tipo se hace indispensable el uso y desarrollo de distintos componentes electrónicos que, en conjunto con el software, permitirán el control y manipulación de distintos aspectos de la residencia, así como también proporcionarán nuevos tipos de interfaces de control para el sistema.

6.1 Análisis

Para este sistema se analizaron tres subsistemas electrónicos que proveerán de las funcionalidades básicas de control y retroalimentación que se pretenden lograr en este proyecto. Su análisis y justificación se explican a continuación:

6.1.1 Circuito para manejo de señales infrarrojas “irTrans”

Dentro del ambiente de la residencia, es común que el individuo interactúe con distintos dispositivos que cuentan con mandos infrarrojos, como televisores, reproductores de DVD, equipos de sonido, etc. Debido a lo anterior, se consideró la incorporación dentro del sistema, de un circuito electrónico que permitiera la manipulación de estas señales infrarrojas en conjunto con la

computadora, para de esta forma automatizar este tipo de control desde un lugar centralizado. Esto corresponde al control graficado en el Diagrama Nº 1 “Dispositivos IR”.

Si bien existen una serie de circuitos disponibles en internet, que realizan este tipo de funciones, por lo general se limitan a recibir señales infrarrojas para luego, mediante el software adecuado, realizar alguna acción dentro de la computadora a modo de “control remoto” para el PC. En este caso se necesitaba que además de recibir y almacenar la señal leída dentro de la computadora, el circuito fuera capaz posteriormente de enviar nuevamente esta señal para reproducir el efecto del control remoto original del dispositivo, abarcando el rango más amplio de señales posibles, permitiendo de esta forma el almacenamiento de múltiples señales infrarrojas correspondientes a distintos dispositivos de distintas marcas y modelos y su posterior control mediante estas señales.

Debido a lo anterior, y luego de una exhaustiva búsqueda y análisis de distintas alternativas, tanto comerciales como abiertas, es que se decidió por la utilización del dispositivo electrónico “irTrans”, construido en Alemania por Marcus Müller. Este dispositivo, además de ser de arquitectura abierta, cuenta con las siguientes características que lo hacen ideal para un proyecto de este tipo:

- irTrans permite el uso de la computadora como si fuera un control remoto universal, permitiendo la manipulación de las señales propias de un televisor, DVD, equipos de audio, etc.

- irTrans es capaz de recibir y almacenar dentro de la computadora cualquier señal infrarroja emitida desde un control remoto convencional como los que se incluyen en la mayoría de los equipos de audio y video del mercado, para luego poder emitir estas señales cuando se desee, con lo cual la persona no necesitará adquirir nuevo equipamiento (Televisores, Reproductores de DVD, etc.) que se adapte un protocolo de control como X-10 o Lonworks.

- Permite la transmisión de señales IR (infrarrojas) a través de diferentes habitaciones mediante un bus serial de dos cables provisto por el mismo dispositivo irTrans.

- irTrans permite tener un control centralizado en la computadora de todos los dispositivos que empleen señales infrarrojas dentro del hogar.

- Permite el control de la computadora a través de un control remoto.

- A pesar de que existen algunas soluciones que permiten integrar a la computadora con sistemas IR, la mayoría de éstas, tienen problemas como: no presentan un hardware estándar para su construcción o evolución, además, por lo general, solo cuentan con la capacidad de recibir señales infrarrojas pero no transmitirlas, y junto con esto, los sistemas IR que basan la codificación de la señal mediante la computadora, presentan problemas cuando el PC es demasiado lento o esta sobrecargado, ya que los sistemas operativos actuales (Windows, Linux), no están optimizados para trabajos en tiempo real, por lo tanto se presentan problemas para leer señales con un pulso menor a 100µs o en el envío de señales a partir de los 40khz. Por lo anterior, el dispositivo irTrans, cuenta con el microcontrolador Atmel Mega8515, que es capaz de realizar estas operaciones en tiempo real y de esta forma abarcar un porcentaje mayor de señales infrarrojas utilizadas actualmente.

A continuación se presenta el diagrama de la arquitectura del dispositivo irTrans:

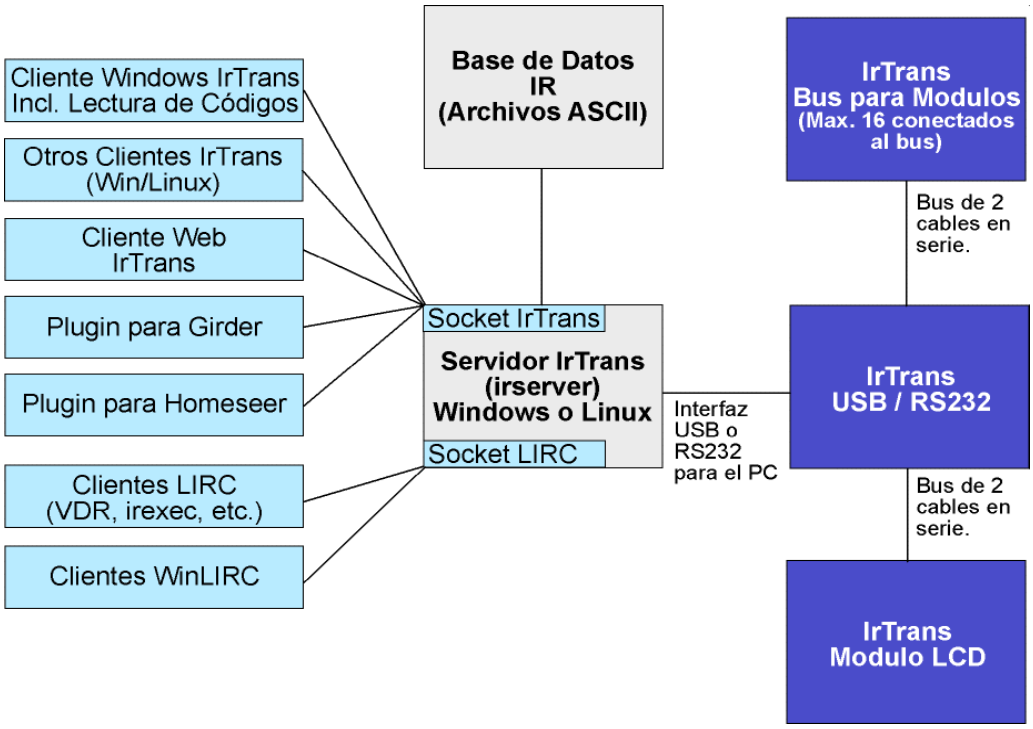

Diagrama Nº 2: Arquitectura sistema irTrans.

Tal como se muestra en el diagrama, el dispositivo irTrans funciona como un servidor que maneja los códigos infrarrojos capturados en una base de datos de archivos ASCII. Los clientes para este servidor, pueden conectarse mediante un socket convencional, o a través de un socket LIRC (“Linux InfraRed Control”, o “Control de Linux mediante Infrarrojos”, por sus siglas en ingles).

En este caso, la aplicación se deberá conectar como cliente a este servidor mediante un socket convencional, además, se deberán mantener sincronizados los datos de la base de datos de archivos ASCII, con los de la BD del sistema.

6.1.2 Interfaz de realidad virtual “NewGlove”

Dentro del campo de la electrónica de este proyecto, se consideró la construcción de un tipo de interfaz para el software alternativa al teclado/mouse tradicional, correspondiente a lo graficado en Diagrama Nº 1 como “Guante RV”. Esta interfaz, debía enfocarse principalmente en su utilización por parte de personas con restricciones severas de movilidad, que le impidiesen el uso del teclado o mouse convencionales. Se pensó entonces, en la construcción de una interfaz llamada “DataGlove” o “Guante de Datos”, que es la típica interfaz de realidad virtual en forma de guante que la persona podría utilizar para transmitir movimientos normales o leves de la mano a la computadora, para que esta los traduzca a acciones en el ambiente.

Luego de una investigación al respecto, se logró precisar que este tipo de interfaces son usualmente de tecnología propietaria y de un costo bastante elevado, por lo que se optó por buscar a alguien con los conocimientos técnicos necesarios para construir este tipo de dispositivo. También a partir de esta investigación se logró encontrar información acerca de un “DataGlove” de nombre “PowerGlove”, que a modo de mando de juegos, fue distribuido por Nintendo para su primera consola de entretenimiento, y que debido a un mal soporte por parte de los desarrolladores de juegos de esa época, fue retirada del mercado, pero que actualmente, se puede encontrar a precios bastante

económicos en tiendas de remate en internet y que con ciertas modificaciones electrónicas, puede ser utilizada en conjunto con un PC.

De esta forma y luego de contactarse con Kevin Mellot, quien, en Estados Unidos, se dedica a la implementación y pruebas de diversos dispositivos de realidad virtual, se construyó esta nueva interfaz de realidad virtual con el nombre de “NewGlove”, la que cuenta con las siguientes características:

- Comunicación serial con la computadora a modo de ráfagas de datos, con lo cual se evita la utilización de controladores especiales y permite la lectura directa de datos en el puerto serial del PC en cualquier momento.

- Dos grados de libertad, o 2-DOF (“Two Degrees of Freedom”, o “Dos Grados de Libertad” por sus siglas en ingles), uno para giro de la muñeca y el otro para inclinación de la misma.

- Detección del porcentaje de flexión y/o extensión para los dedos: pulgar, índice, anular y medio.

- Dos entradas de datos adicionales para ser utilizadas en conjunto con los movimientos del dedo meñique, codo u hombro, o para la

implementación a futuro de interfaces de retroalimentación para realidad virtual, conocidas como “Interfaces Hapticas”

6.1.3 Circuito de control eléctrico y mecánico.

Dentro del quehacer cotidiano al interior del hogar, tanto de una persona sana como de una discapacitada, es usual que el individuo deba interactuar con una serie de componentes que requieren de acciones mecánicas para su utilización, como ocurre en el caso de puertas y ventanas, tomas de corriente, interruptores de luz, cerraduras, etc. Este tipo de labores pueden tornarse altamente complejas al usar aparatos para la asistencia en el desplazamiento (sillas de ruedas, andadores, muletas, etc.) o debido a la escasa movilidad producto de la vejez, enfermedades, fracturas, reposo médico, etc. Esto se debe principalmente a que la disposición de estos componentes dentro del hogar, por lo general, no contemplan su utilización por parte de personas con movilidad restringida, con lo cual se hace necesaria la implementación de un circuito electrónico que cumpla con ciertas características que satisfagan esta problemática y permitan la automatización y control de estos componentes, esto se refleja en el Diagrama Nº 1 como “Circuito Control Mecánico”.

Para lograr esto último se contó con la colaboración de Juan Carlos Gallardo, quien construyó un circuito controlable por computadora para este propósito con las siguientes características:

- Control del circuito vía puerto paralelo, para una fácil comunicación y control del mismo y evitar el uso y desarrollo de controladores especiales.

- Control de 7 motores de corriente continua (Motores DC), para ser utilizados principalmente en puertas, ventana, camas posicionables y otros componentes de este tipo al interior de la residencia.

- Control de 7 motores paso a paso (Motores PAP), utilizados principalmente en robótica, y que permiten un control mas preciso sobre la posición del eje del motor en un momento en particular, los que pueden ser utilizados para motorizar cámaras de vigilancia en distinto puntos del hogar.

- Control sobre dos cerraduras eléctricas, para ser utilizadas en puertas o portones principales.

- Control de 7 solenoides (dispositivos electrónico-mecánicos de dos posiciones), para ser utilizados como cerraduras de ventanas, puertas de muebles, etc.

- Control de 8 tomas de corriente alterna (utilizada en la red eléctrica del hogar), para controlar lámparas, calentadores, hervidores de aguas, ventiladores, etc.

- Capacidad de retroalimentación para mantener actualizada la información sobre el estado de sus componentes (motores, solenoides, cerraduras, etc.).

- Modularidad, lo que permite que la persona desarrolle solo las partes del circuito que le son necesarias para satisfacer su problemática en particular.

Como parte del análisis y de acuerdo a las características que aquí se presentan, se desarrollaron los siguientes diagramas de bloques que explican el funcionamiento de las distintas partes que componen el circuito de control electrónico / mecánico.

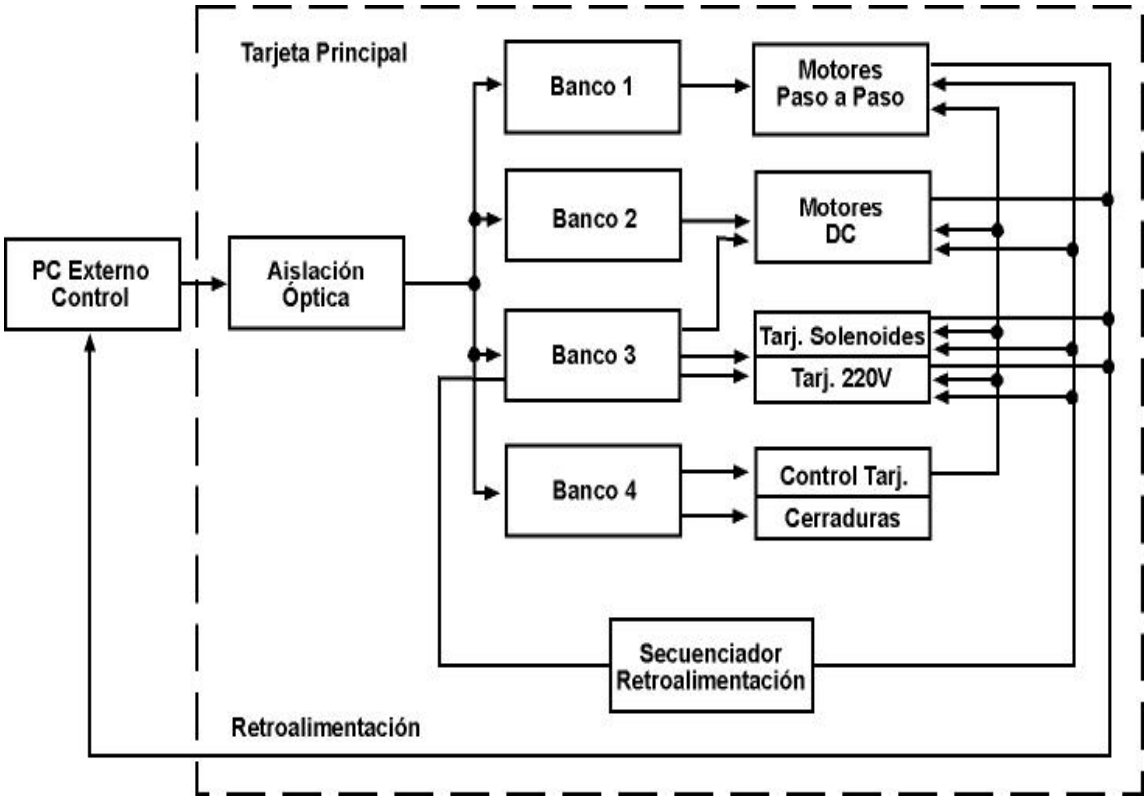

Diagrama Nº 3: Diagrama en bloques tarjeta principal circuito control eléctrico / mecánico.

Tal como se muestra en el diagrama de bloques, se ideó, que la mejor forma de cumplir con los requerimientos en cuanto a modularidad y capacidades de expansión, era la de descomponer el circuito en una placa principal capaz de controlar vía el puerto paralelo de la computadora distintos "bancos de control", en donde a cada uno de estos bancos se podrán conectar las distintas "tarjetas de expansión domóticas" capaces de controlar distintos

componentes. La mejor forma de explicarlo, es haciendo la analogía con un PC de escritorio, en el cual existe una “Placa Principal” o “Tarjeta Madre”, a la cual se le pueden añadir distintas capacidades conectando distintas tarjetas de expansión como MODEM, sonido, red, video, etc, a alguno de sus puertos PCI o ISA disponibles, permitiendo que el PC se adapte a las necesidades y presupuestos del usuario. De igual forma, lo mismo ocurre con la placa principal del circuito de control electrónico / mecánico a la cual se le podrán implementar y conectar las tarjetas de expansión domóticas que el usuario requiera de acuerdo a su presupuesto y necesidades.

Tal como se muestra también en el diagrama, el circuito principal contará con la capacidad de retroalimentación, pudiendo de esta forma, recibir información de retorno desde las distintas tarjetas adicionales, para de esta forma mantener actualizada la información respecto a cada uno de los componentes conectados.

Tal como se mencionó, existirán entonces distintas tarjetas de expansión que agregarán diversas capacidades de control a la placa principal. A continuación, se presentan los diagramas en bloques para cada una de ellas:

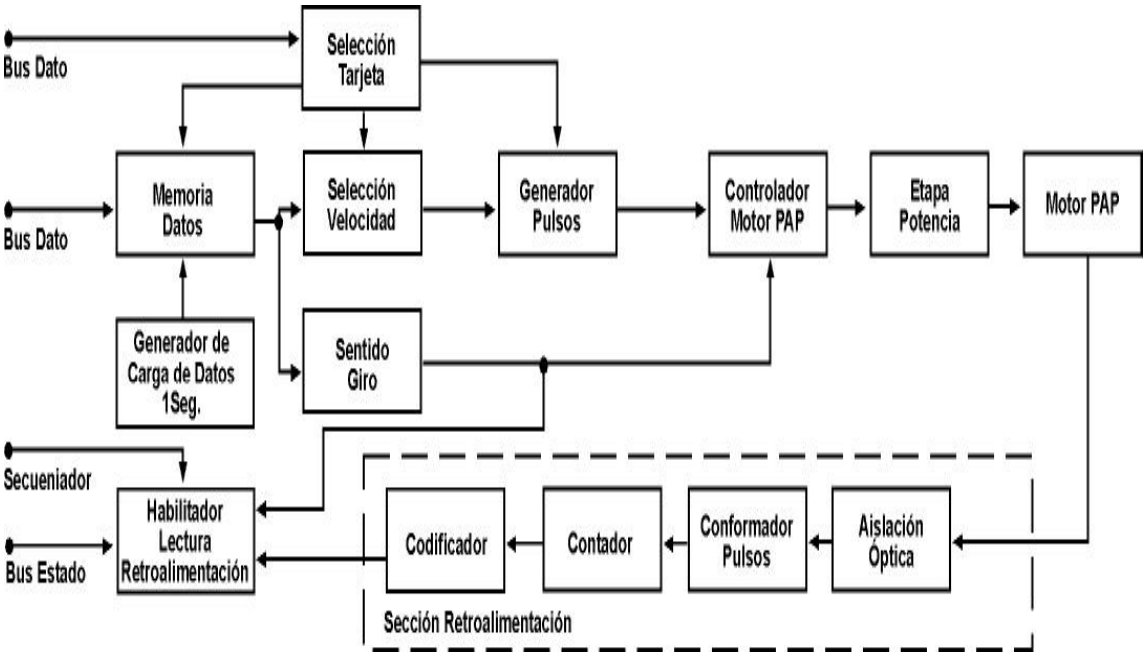

Diagrama Nº 4: Diagrama en bloques tarjeta control motor PAP.

Esta tarjeta será la encargada de agregar la capacidad de control sobre un motor paso a paso. En el diagrama 4 se pueden apreciar las distintas características de esta tarjeta, tales como selección de velocidad, sentido del giro, conformación de pulsos para retroalimentación, etc.

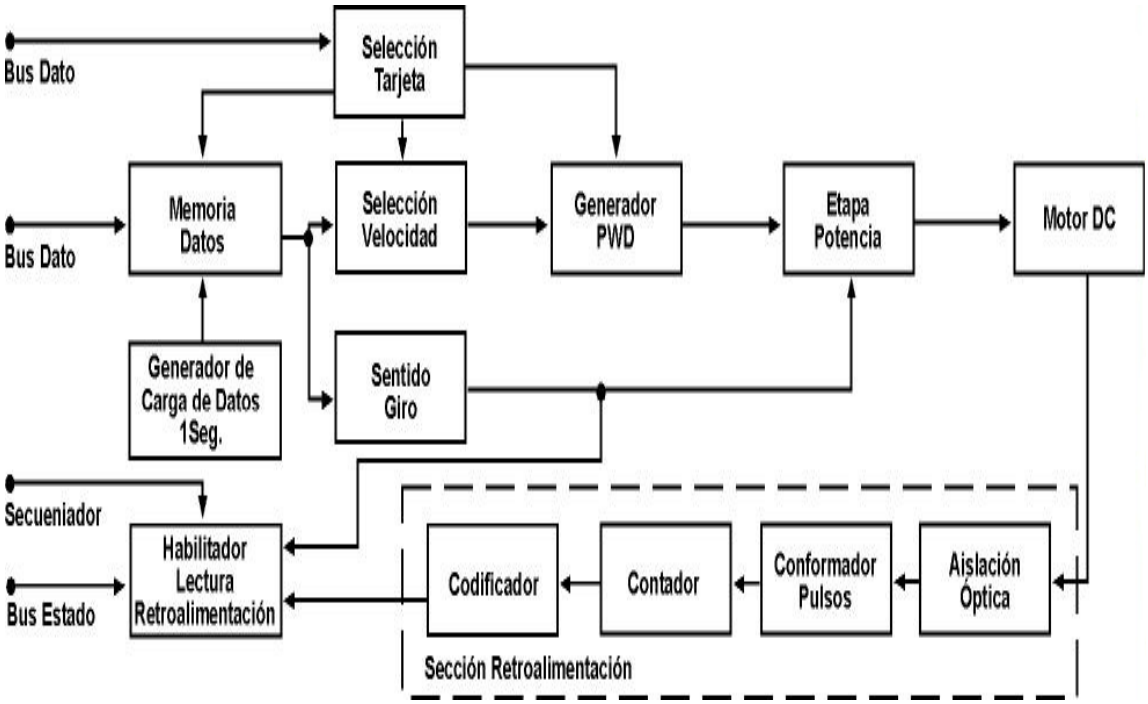

Diagrama Nº 5: Diagrama en bloques tarjeta de control motor DC.

De forma similar al diagrama anterior, esta tarjeta proporcionará control en este caso para motores de corriente continua normales. Las características de control son iguales a las anteriormente planteadas.

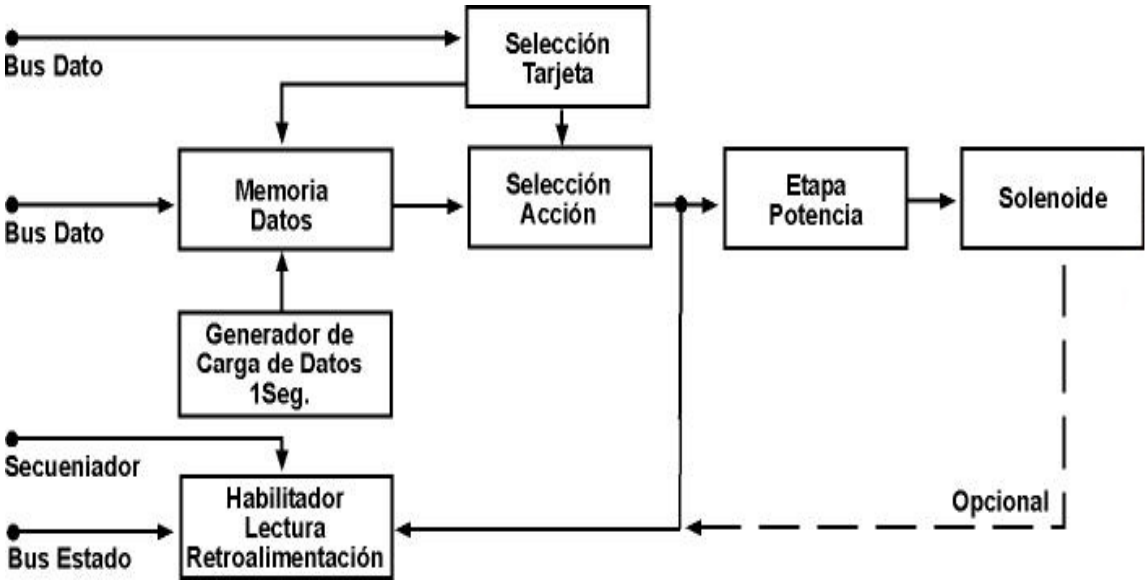

Diagrama Nº 6: Diagrama en bloques tarjeta de control solenoide.

Este diagrama muestra la forma en la cual se podrá incorporar control sobre solenoides, este tipo de control es algo más sencillo que el de motores. Si bien, al igual que en los diagramas anteriores, se conserva la etapa de “selección de tarjeta”, a esto solo se le agrega el control sobre la acción que se desea, es decir, que el solenoide se retraiga o extienda, además de la capacidad opcional de retroalimentación para obtener el estado de éste.

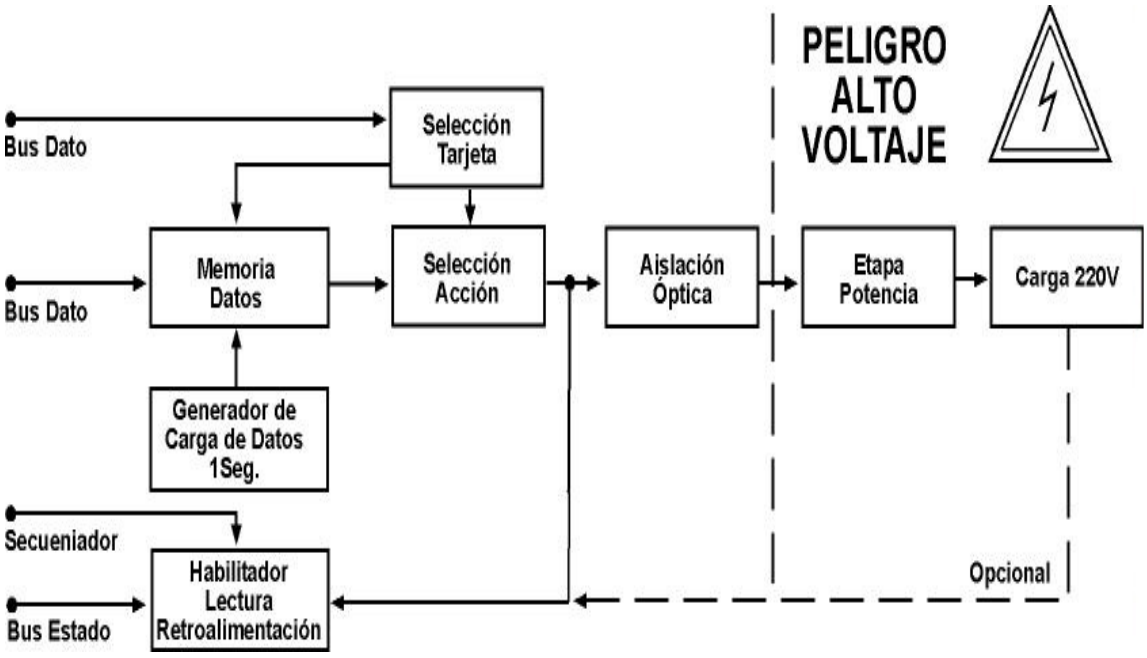

Diagrama Nº 7: Diagrama en bloques tarjeta de control salida 220V.

Esta tarjeta añadirá la capacidad de control sobre tomas de corriente de 220V, como las que se encuentran habitualmente en los hogares, para de esta forma controlar el encendido o apagado (selección de acción) de cualquier aparato conectado a una de ellas. También contará con la capacidad opcional de retroalimentación para censar el estado de la tarjeta en todo momento.

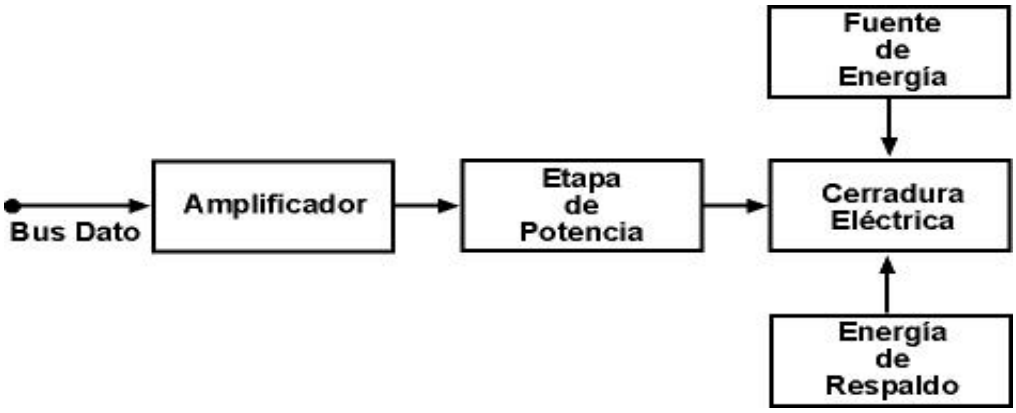

Diagrama Nº 8: Diagrama en bloques tarjeta de control cerradura eléctrica.

Finalmente y para completar los requerimientos antes planteados, este diagrama, muestra la forma en la cual esta tarjeta agregará la capacidad de control sobre una cerradura eléctrica conectada a ésta. El control sobre este tipo de dispositivos, se reduce a la generación de un pulso por parte del bus de datos, el cual es amplificado para gatillar el accionar de la cerradura, este tipo de control, por su sencillez, no requiere de capacidad de retroalimentación.

6.2 Diseño

Puesto que dentro de la electrónica de los circuitos empleados para este proyecto, lo más importante, desde el punto de vista del diseño, son los diagramas de circuito, es que se incluyen éstos a continuación, junto con una descripción general de los mismos.

La información más detallada respecto a sus partes y piezas, y el funcionamiento de las misma, etc, se incluyen en el anexo digital (X:\Documentos\Electronica), en donde tambien se podran encontrar los esquemas electronicos desarrollados en “CircuitMaker” y “EagleCAD”, para una mejor apreciación de los mismos.

6.2.1 Diseño circuito irTrans

A continuación se presenta el diagrama electrónico para el circuito de control de señales infrarrojas irTrans.

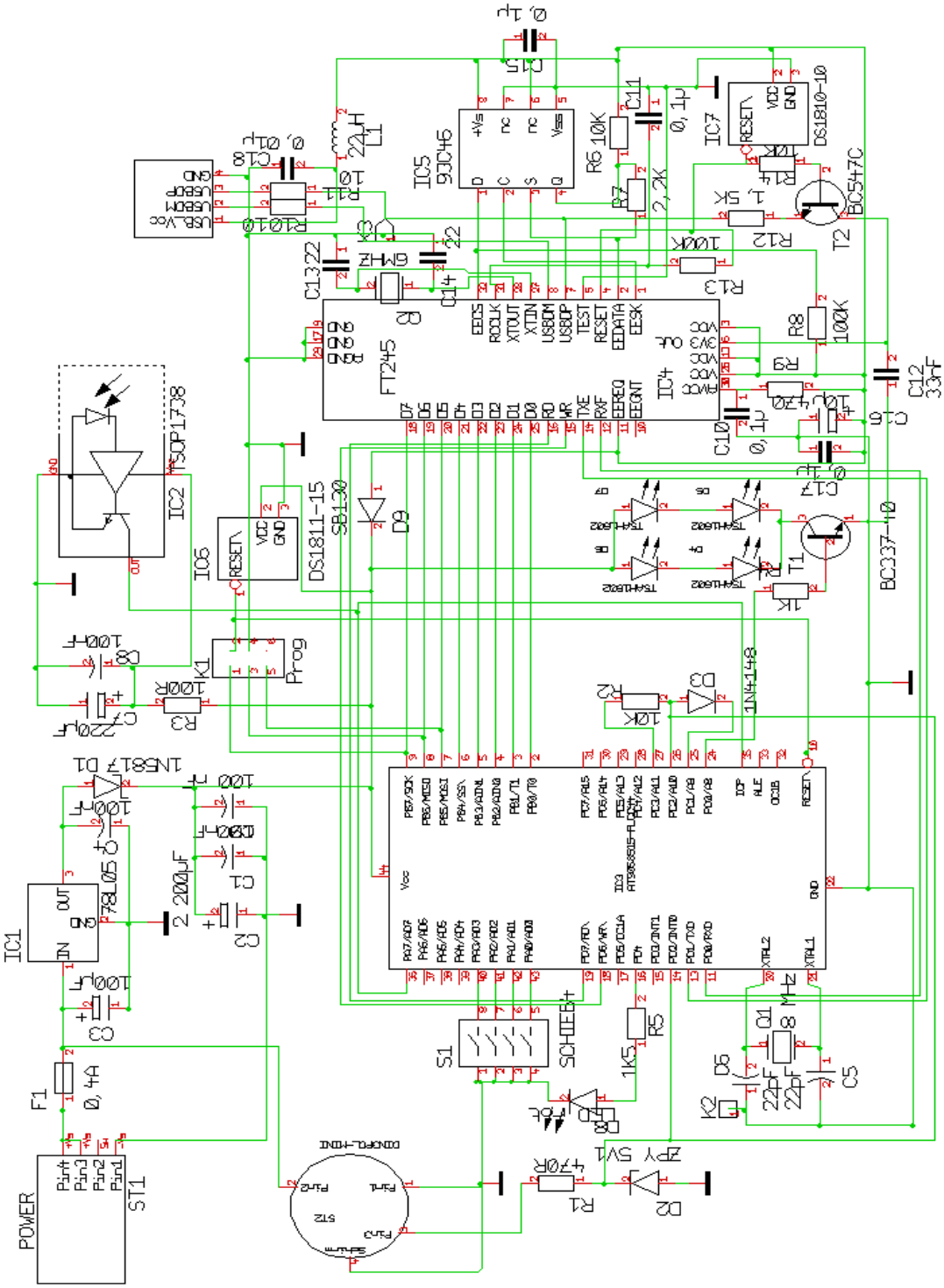

Diagrama Nº 9: Diseño electrónico dispositivo irTrans.

En el diagrama anterior se muestra el diseño electrónico del dispositivo irTrans, que se compone de los siguientes elementos:

- Microcontrolador Atmel Mega8515 con velocidad de reloj de 8Mhz y memoria Flash ROM de 8K.

- 4 LEDs de transmisión IR con un rango de entre 15 – 20m.

- Receptor IR de tipo TSOP 1138.

- Bus serial de 2 cables para conectar hasta 16 transreceptores en diferentes habitaciones. Señales IR pueden ser emitidas o recibidas en cualquier habitación. Debido a que todos los transreceptores poseen una dirección única, puede especificarse cual de estos enviará cada comando IR.

- El receptor IR puede recibir señales portadoras entre los 30 – 50Khz, abarcando el 98% de los sistemas IR usados actualmente.

- El transmisor IR puede generar pulsos de entre 15 – 200Khz y 455Khz, lo cual es suficiente para los actuales sistemas IR.

- La fuente de alimentación puede ser provista por el mismo PC, o mediante una fuente de poder alternativa para alimentar a todos los demás dispositivos irTrans conectados al bus.

Información adicional respecto al listado de partes y piezas, su función en particular, diagramas para tarjetas PCB y la posición de los componentes en esta, así como también el Firmware que debe ser cargado en el microcontrolador, son adjuntos en anexo digital (X:\Fuentes\Electronica).

6.2.2 Diseño circuito guante realidad virtual NewGlove

El siguiente diagrama presenta el diseño del circuito electrónico de la interfaz de realidad virtual NewGlove.

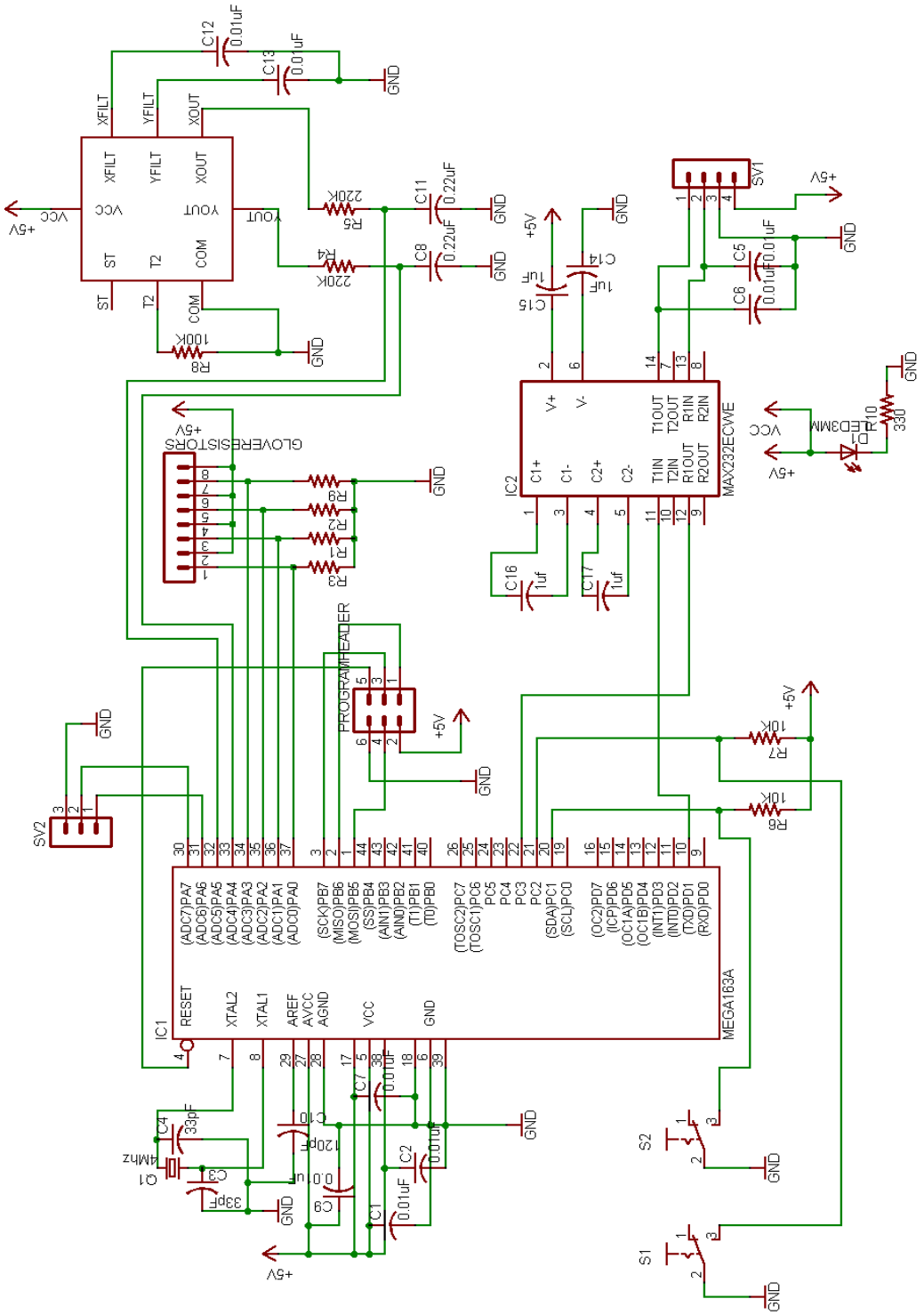

Diagrama Nº 10: Diseño electrónico guante realidad virtual NewGlove.

El diagrama anterior muestra los distintos componentes empleados en la realización de este circuito, tales como:

- Microcontrolador ATmega163L.

- Acelerómetro ADXL202, que permite la detección del giro y la inclinación.

- Controlador MAX232, utilizado en la comunicación serial con la computadora.

- Velocidad de reloj de 4Mhz.

Básicamente, el circuito de la interfaz NewGlove, es un convertidor A/D (Análogo/Digital), con un acelerómetro para detección de giro e inclinación y sensores de flexión para determinar la posición de los dedos, la comunicación se realiza vía un conector DB9 al puerto serial, a una velocidad de 19200 baudios, con 8 bit de datos, sin paridad y 1 bit de parada, (19200, 8, N, 1). Además, los sensores de flexión son capaces de discriminar hasta 128 posiciones distintas, aunque los dedos de la mano humana solo son capaces de generar entre 60 a 70 posiciones.

Información adicional respecto al listado de partes y piezas, su función en particular, diagramas para tarjetas PCB y la posición de los componentes en esta, así como también el Firmware que debe ser cargado en el microcontrolador, son adjuntos en anexo digital (X:\Fuentes\Electronica).

6.2.3 Diseño circuito de control eléctrico y mecánico

El circuito de control eléctrico y mecánico, fue concebido pensando en la modularidad de este, es decir, existe un circuito base capaz de albergar en bancos separados, distintos circuitos de expansión que proveen de distintas funcionalidades (Control de: Motores DC, Motores PAP, Solenoides, Cerraduras, Tomas de corriente, etc.).

El siguiente diagrama presenta el circuito de control principal.

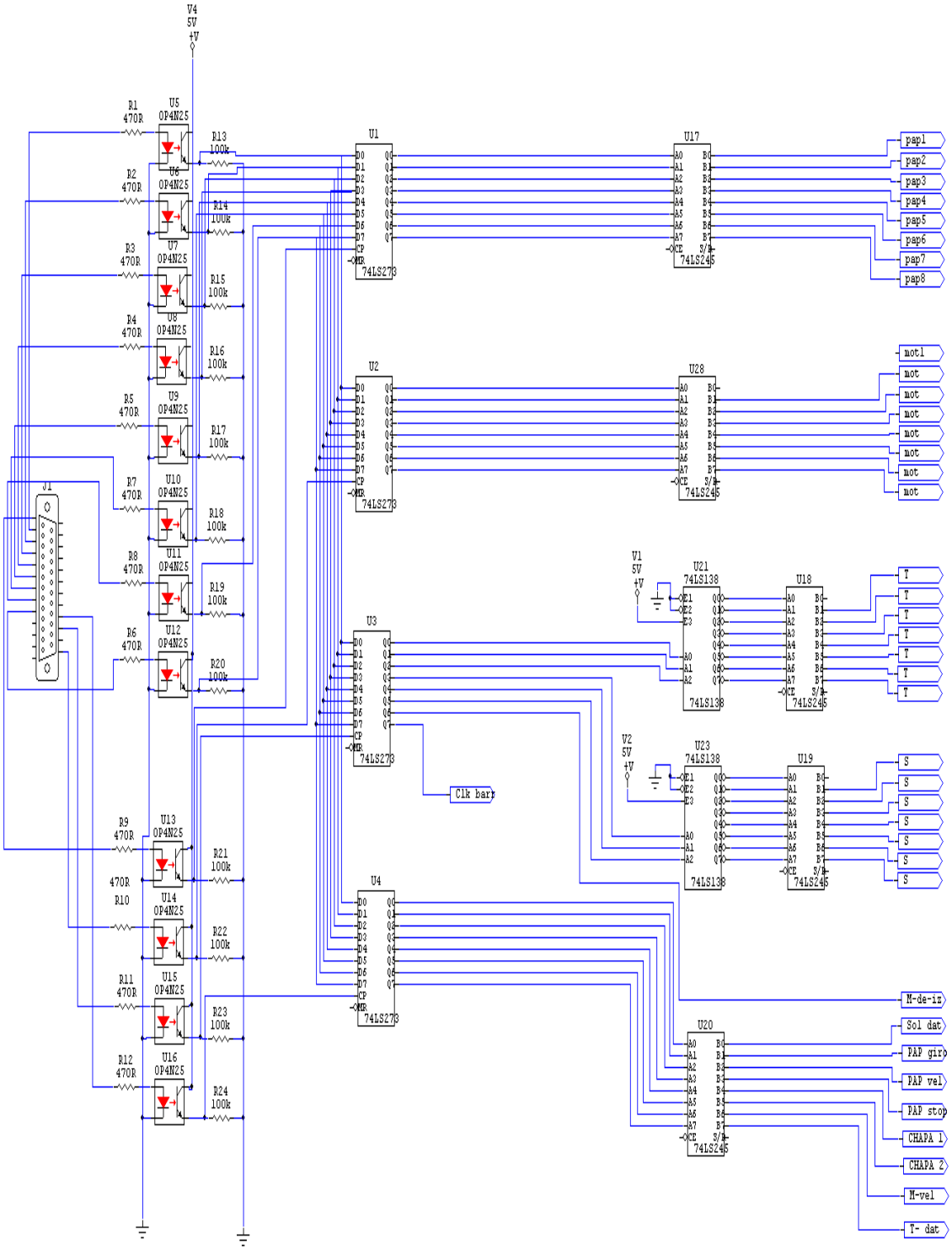

Diagrama Nº 11: Diseño tarjeta principal circuito eléctrico mecánico.

En el diagrama anterior se muestra la electrónica que compone la placa principal del circuito de control eléctrico y mecánico. Básicamente, este circuito es un expansor del bus de datos del puerto paralelo de la computadora con memorias de tipo LATCH para mantener la información en conjunto con las señales del bus de control del puerto y que a su vez, permite recibir retroalimentación a través del bus de estatus del mismo, para de esta forma, lograr un control digital sobre un número importante de otros circuitos.

Tal como se mencionó anteriormente, este circuito posee una serie de bancos, en los cuales se pueden conectar una serie de otros circuitos para control de motores DC o PAP, solenoide, cerraduras eléctricas, etc.

A continuación se presenta el diseño electrónico del circuito para control de motores PAP.

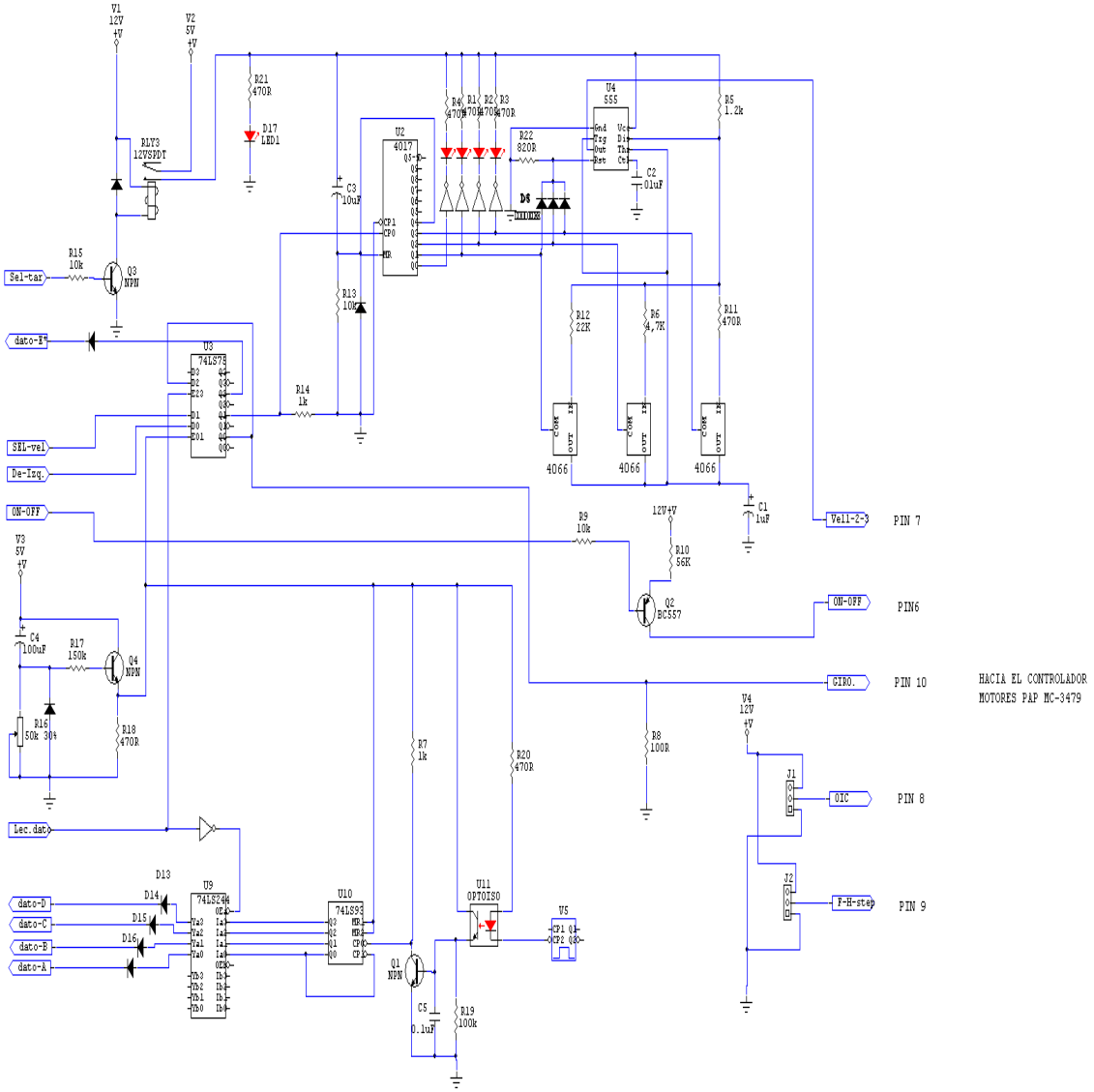

Diagrama Nº 12: Diseño circuito control motor PAP.

Este circuito funciona mediante un circuito integrado controlador para motores paso a paso tipo MC-3479, el cual recibe un pulso que determina la habilitación del integrado, un pulso que le indica el sentido de giro al motor, y un último pulso para indicar el inicio o la detención del motor en un momento determinado. Además, el circuito cuenta con un selector de 3 velocidades (baja, media y alta) mediante generación de pulsos.

El siguiente diagrama muestra el diseño electrónico del circuito para control de motores normales de corriente continua (Motores DC).

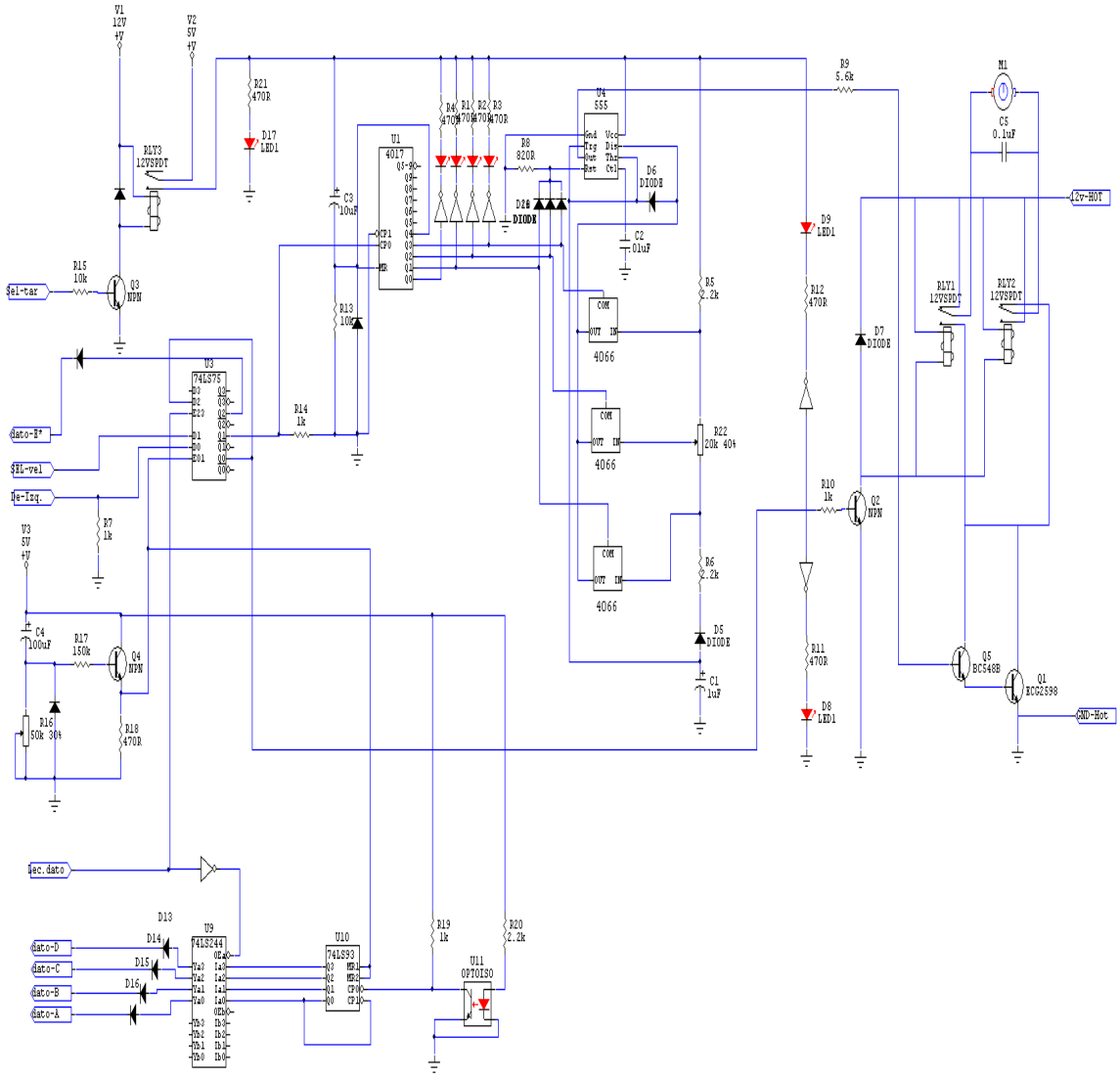

Diagrama Nº 13: Diseño circuito control de motor DC.

Este circuito cuenta, al igual que el circuito de control de motores PAP, de un selector de 3 velocidades mediante pulso, además de la selección de giro del motor. El circuito funciona mediante relés que actúan como interruptores para el control del motor.

El siguiente diagrama muestra el diseño del circuito para control de salidas 220V.

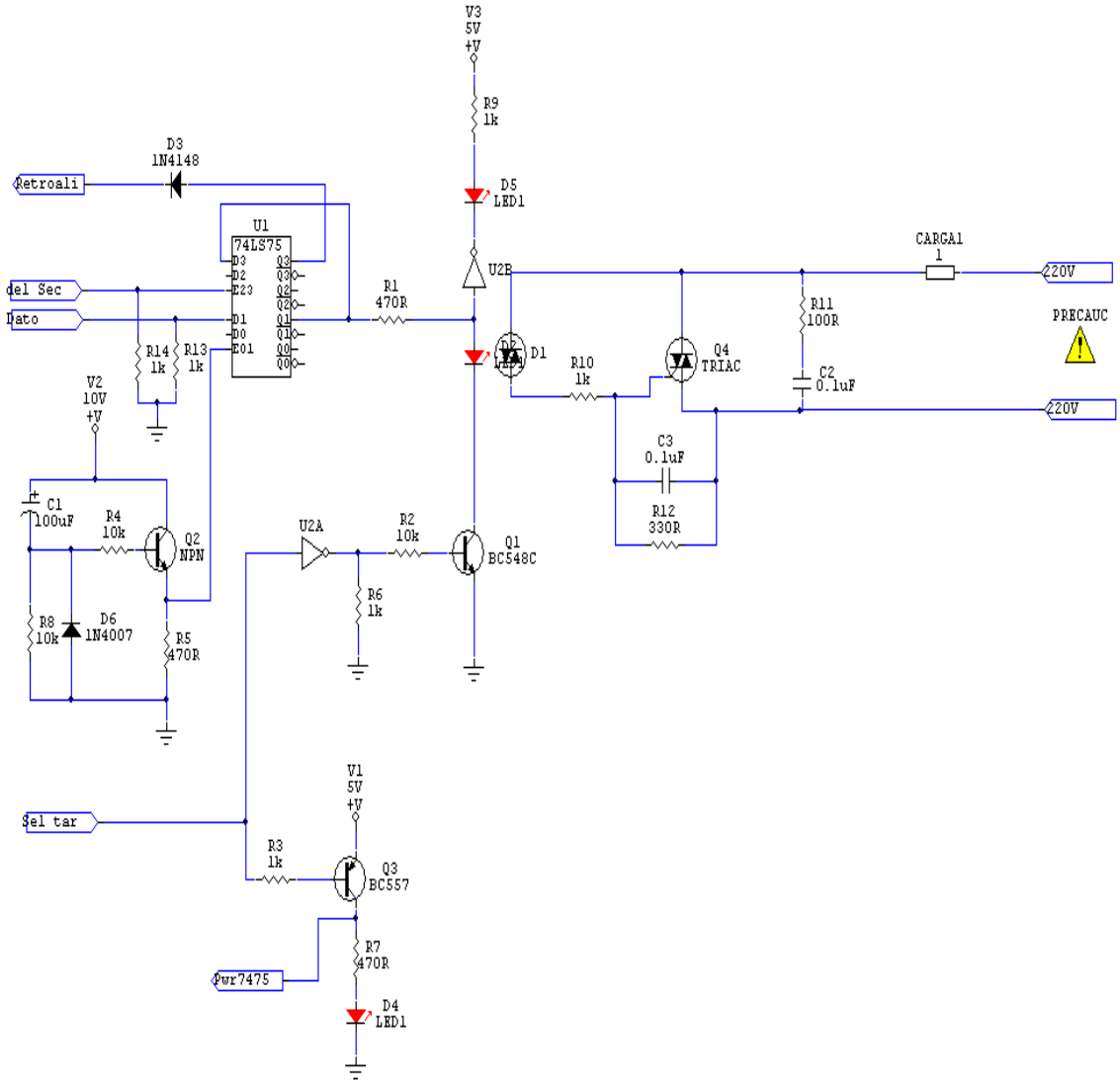

Diagrama Nº 14: Diseño circuito control salidas 220V.

Este circuito permite, mediante pulsos, el control de encendido/apagado de tomas de corriente de 220V correspondientes a la red eléctrica del hogar, lo que permite el control sobre lámparas, ventiladores, calefactores, hervidores de agua, etc.

A continuación, se presenta el diseño electrónico del circuito para control de solenoides.

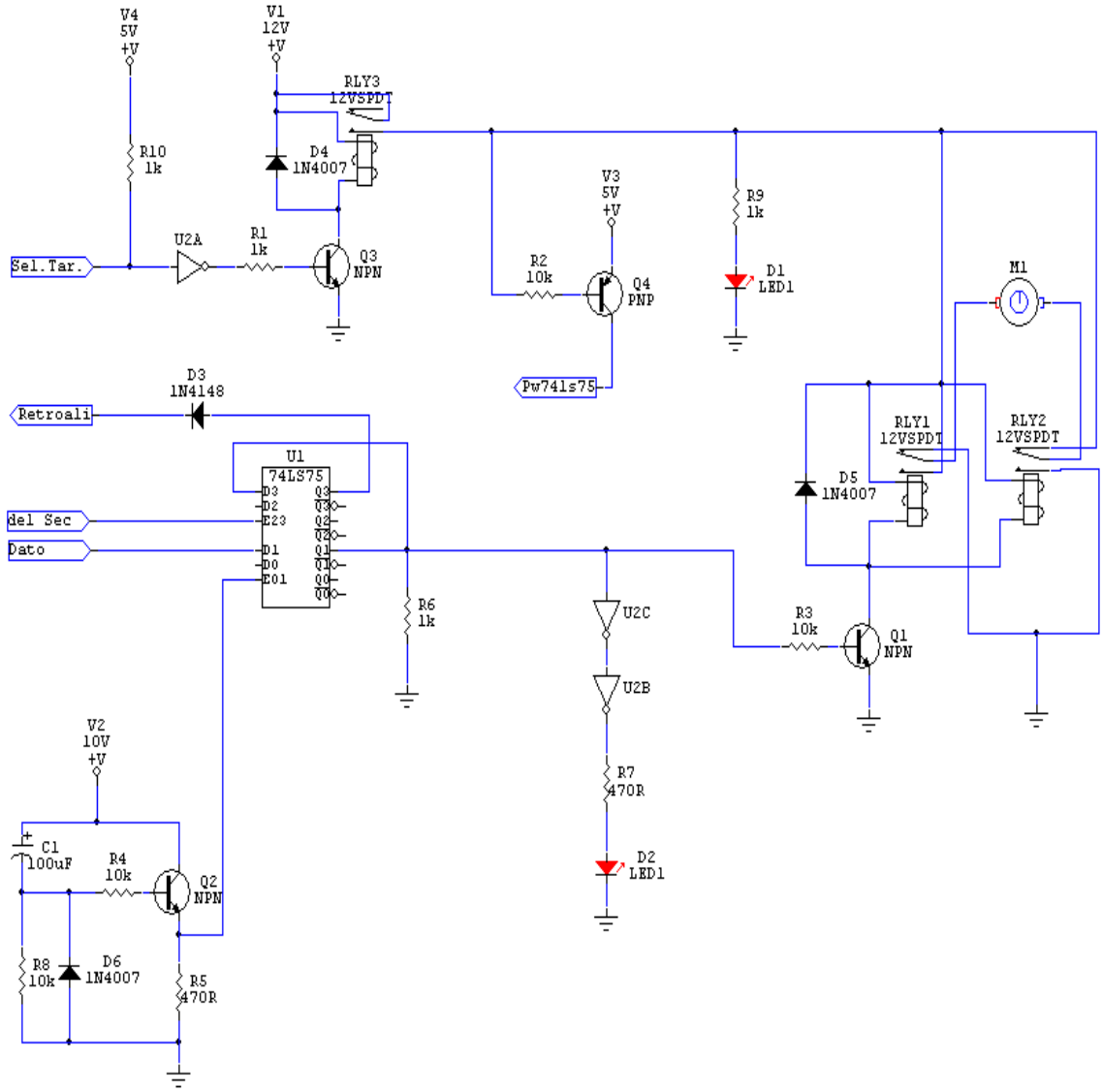

Diagrama Nº 15: Diseño circuito control solenoide.

Este circuito permite, mediante pulsos, controlar el estado de un solenoide en un momento en particular.

A continuación, se presenta el diseño electrónico del circuito para control de cerraduras eléctricas.

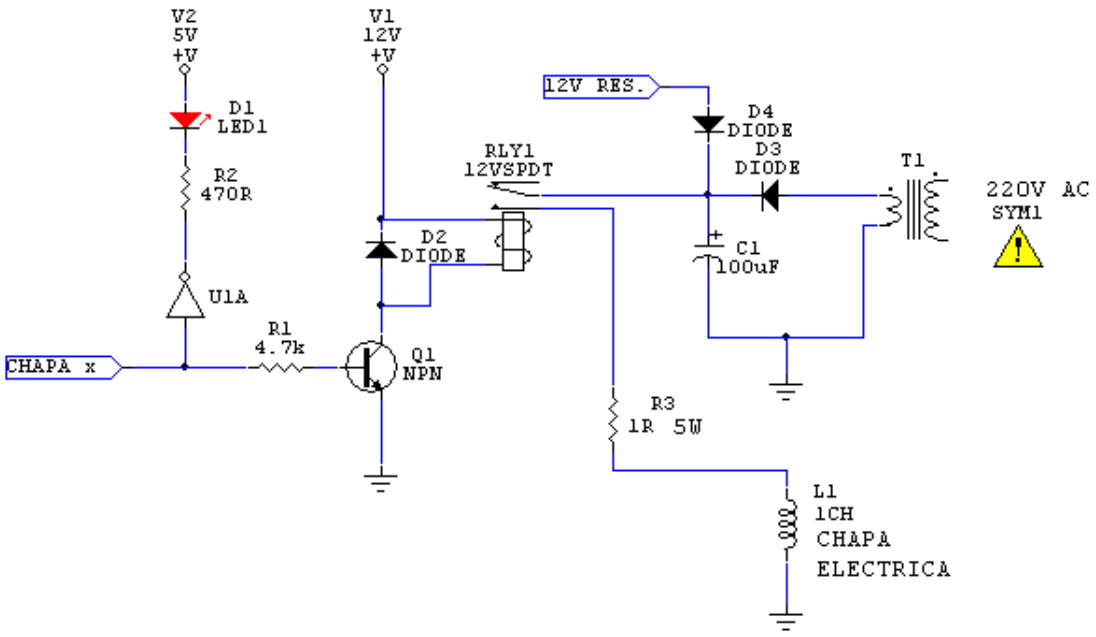

Diagrama Nº 16: Diseño circuito control cerradura eléctrica.

Este circuito permite, mediante pulsos, controlar el estado de una cerradura eléctrica, para ser utilizado en conjunto con, por ejemplo, un circuito de control de motores DC en puertas y/o ventanas.

A continuación, se presenta el diseño electrónico del circuito de retroalimentación.

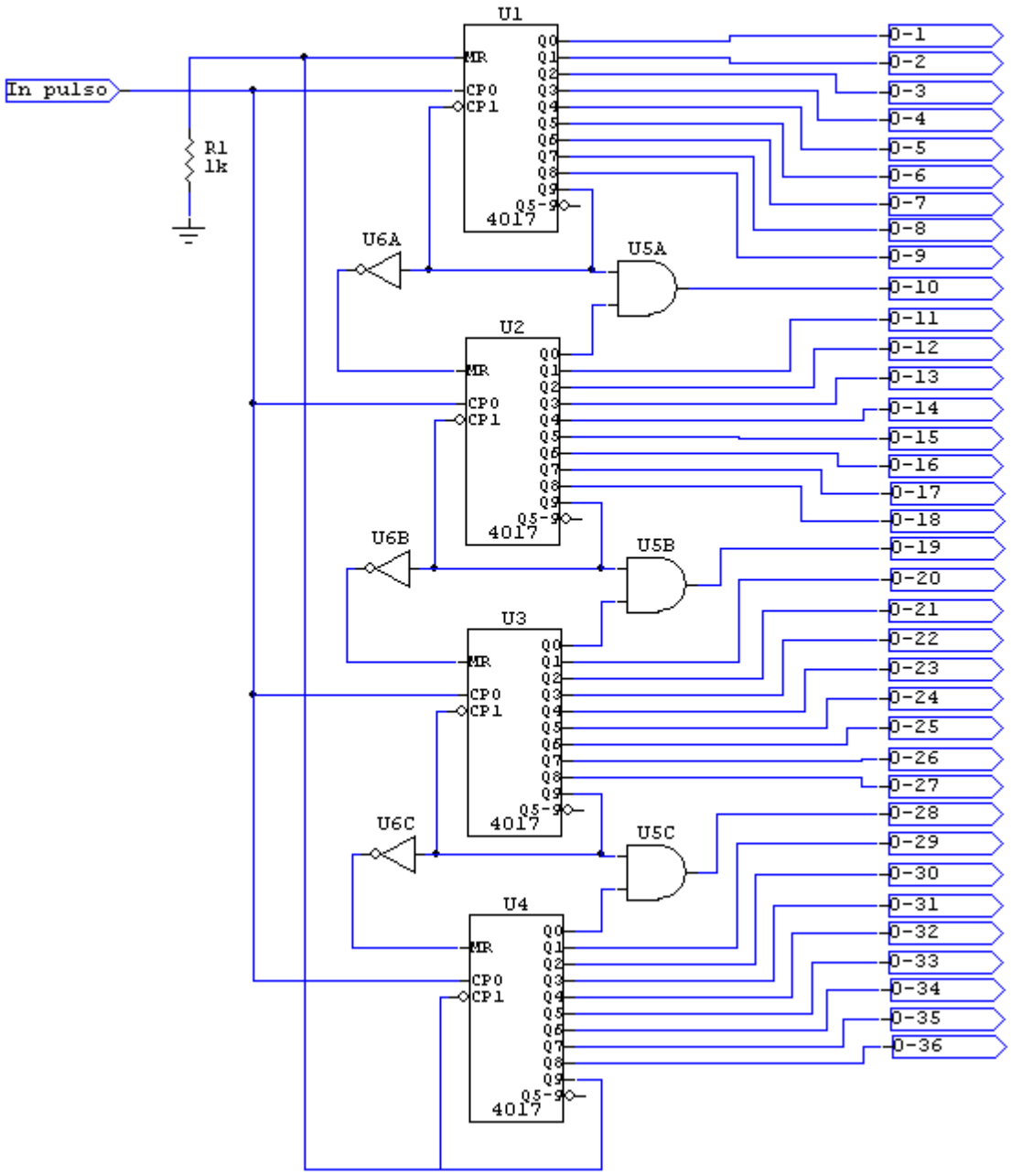

Diagrama Nº 17: Diseño circuito retroalimentación.

Este circuito es el encargado de censar el estado de los distintos componentes conectados a las tarjetas de expansión, manteniendo de esta forma siempre actualizados los valores de retorno de los componentes y determinar si el componente (motor, solenoide, etc.) ha respondido correctamente al comando dado, o si ha sido manipulado de forma externa al sistema.

Tal como se mostró en los circuitos anteriores, la estructura general del circuito principal es modular, lo que permite que la persona construya solo los circuitos especializados que le sirvan para solucionar una problemática en particular, disminuyendo los costos de implementación asociados a su construcción, y luego conectándolos en los bancos apropiados en el circuito de control principal.

De igual forma que en los circuitos anteriores, la información adicional respecto al listado de partes y piezas, su función en particular, diagramas para tarjetas PCB y la posición de los componentes en ésta, son adjuntos en el anexo digital (X:\Fuentes\Electronica).

6.3 Implementación

La implementación de cada uno de los circuitos fue realizada por las personas mencionadas anteriormente como colaboradores en estas áreas: Marcus Müller circuito irTrans, Kevin Mellot guante realidad virtual NewGlove, y Juan Carlos Gallardo circuito control eléctrico y mecánico así como sus tarjetas de expansión.

6.3.1 Implementación irTrans

La siguiente figura muestra la implementación del dispositivo para control y manejo de señales infrarrojas irTrans.

Figura Nº 1: Implementación dispositivo irTrans.

6.3.2 Implementación interfaz realidad virtual guante NewGlove.

La siguiente figura muestra la implementación de la interfaz de realidad virtual, guante NewGlove.

Figura Nº 2: Implementación interfaz de realidad virtual, guante NewGlove.

6.3.3 Implementación circuito control eléctrico y mecánico

A continuación, se presentan la implementación de las distintas tarjetas que componen el circuito de control eléctrico y mecánico.

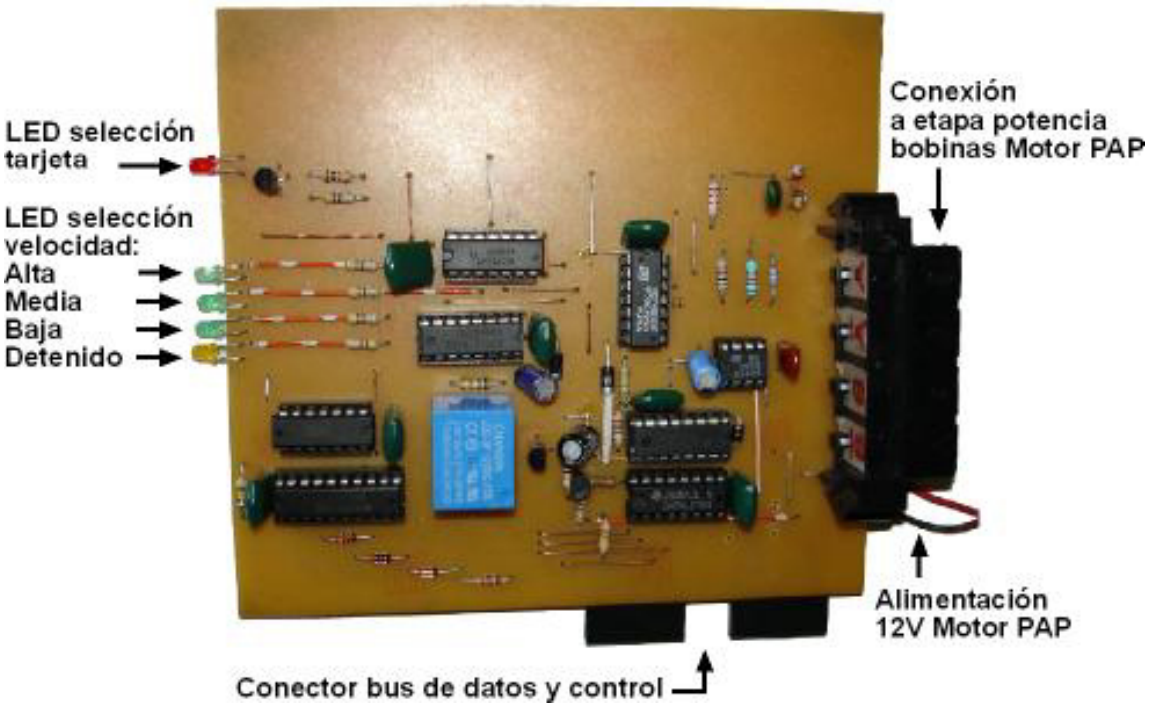

Figura Nº 3: Implementación tarjeta control motor PAP.

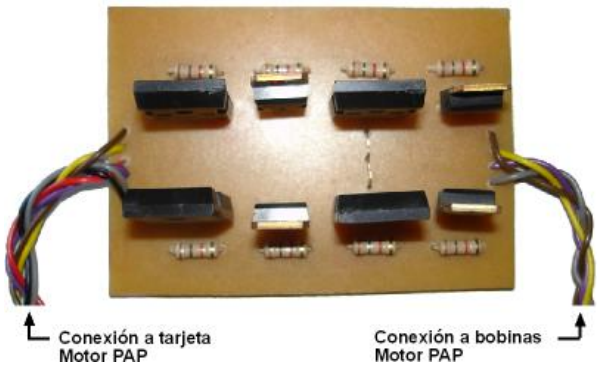

Figura Nº 4: Implementación tarjeta etapa de potencia para tarjeta control motor PAP.

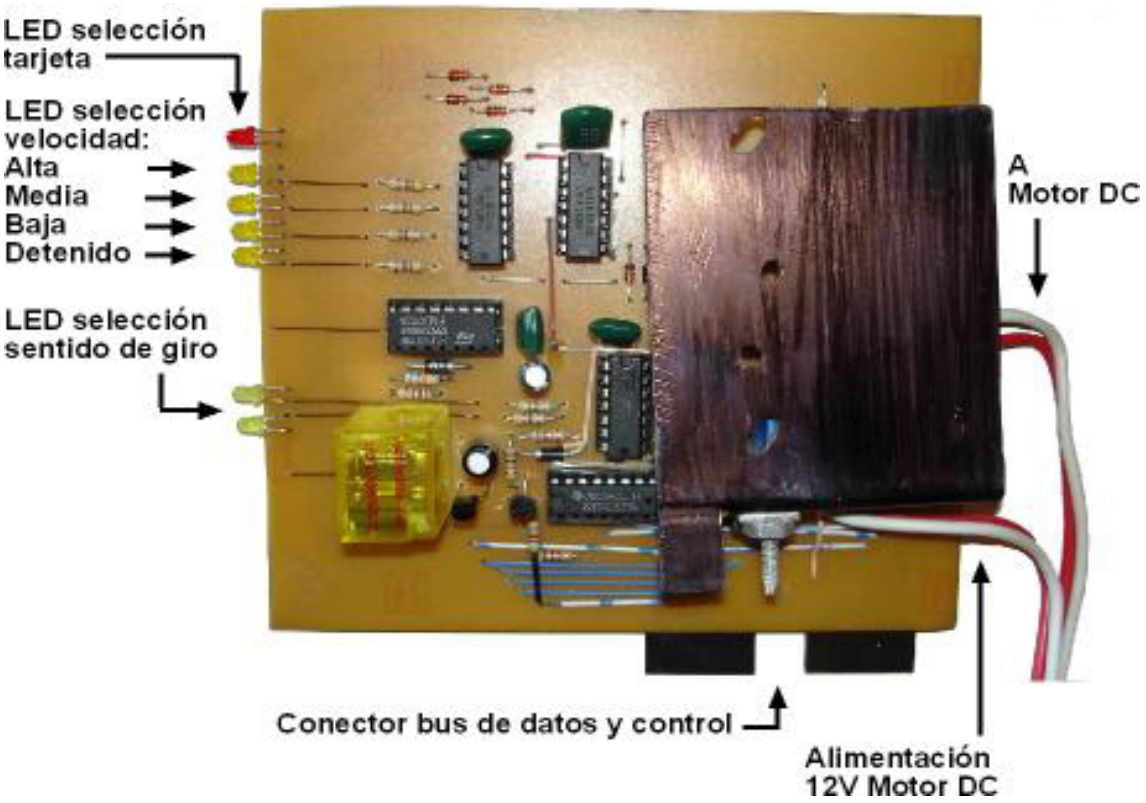

Figura Nº 5: Implementación tarjeta control motor DC.

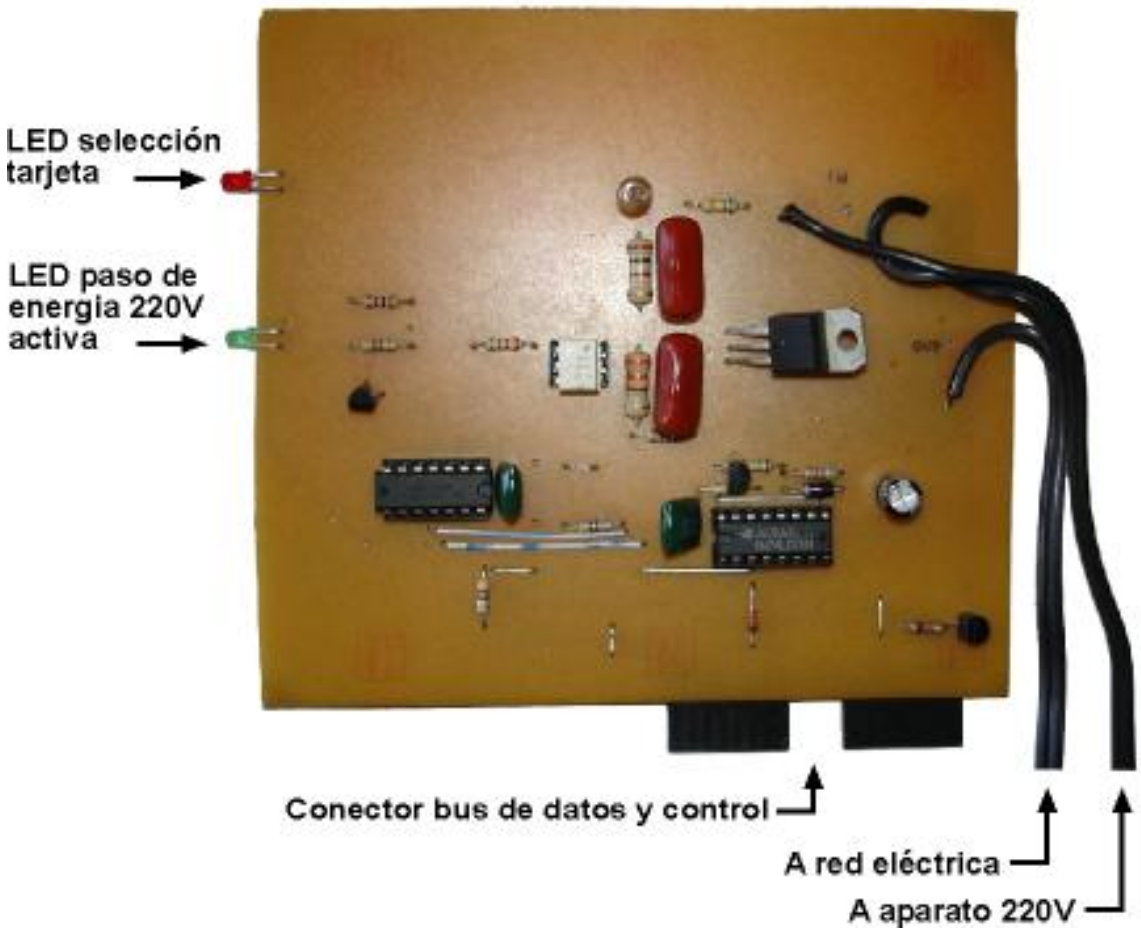

Figura Nº 6: Implementación tarjeta control salida 220V.

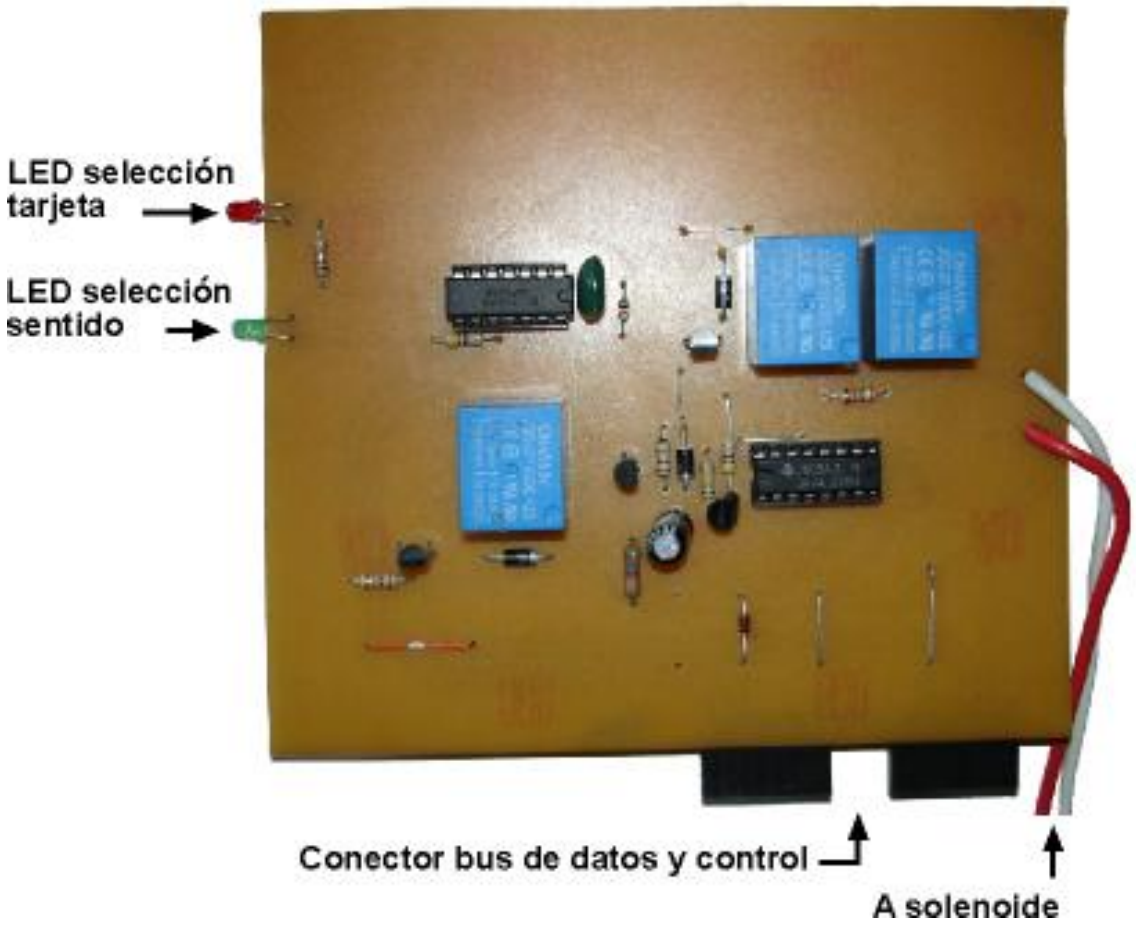

Figura Nº 7: Implementación tarjeta control solenoide.

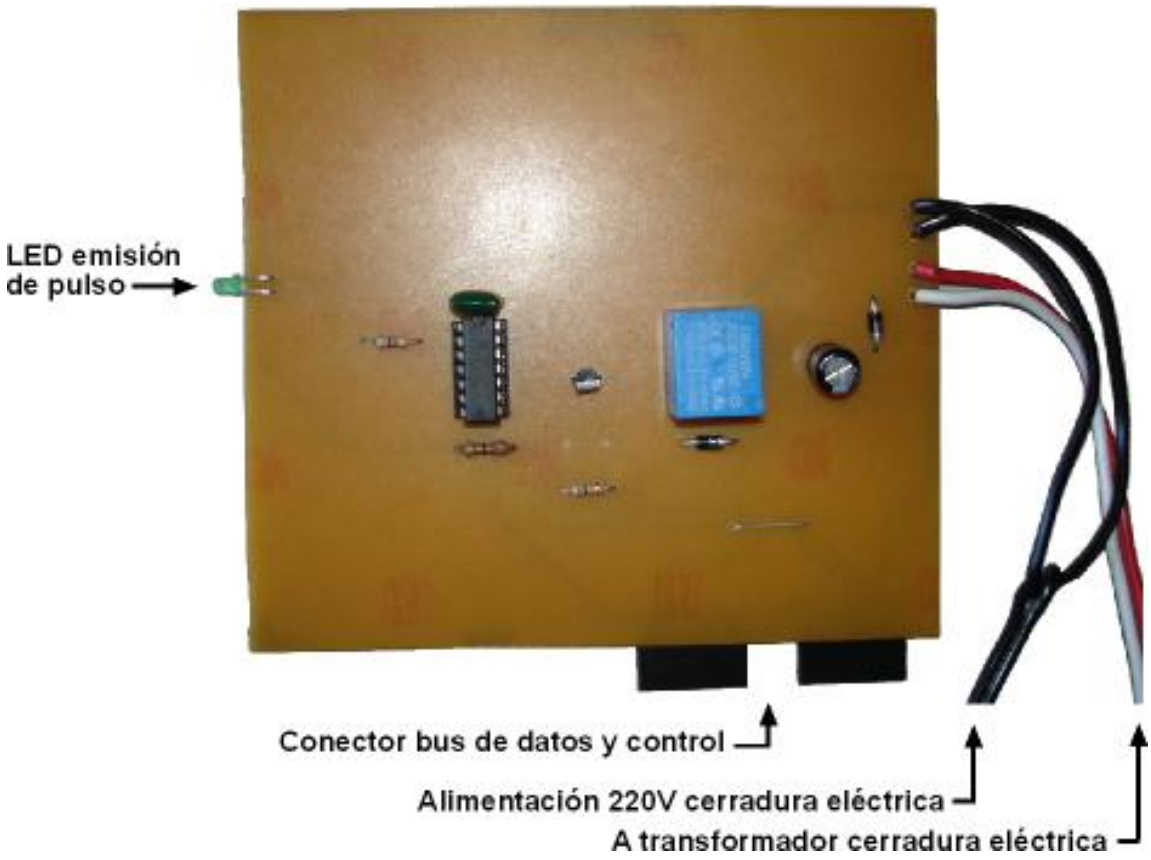

Figura Nº 8: Implementación tarjeta control cerradura eléctrica.

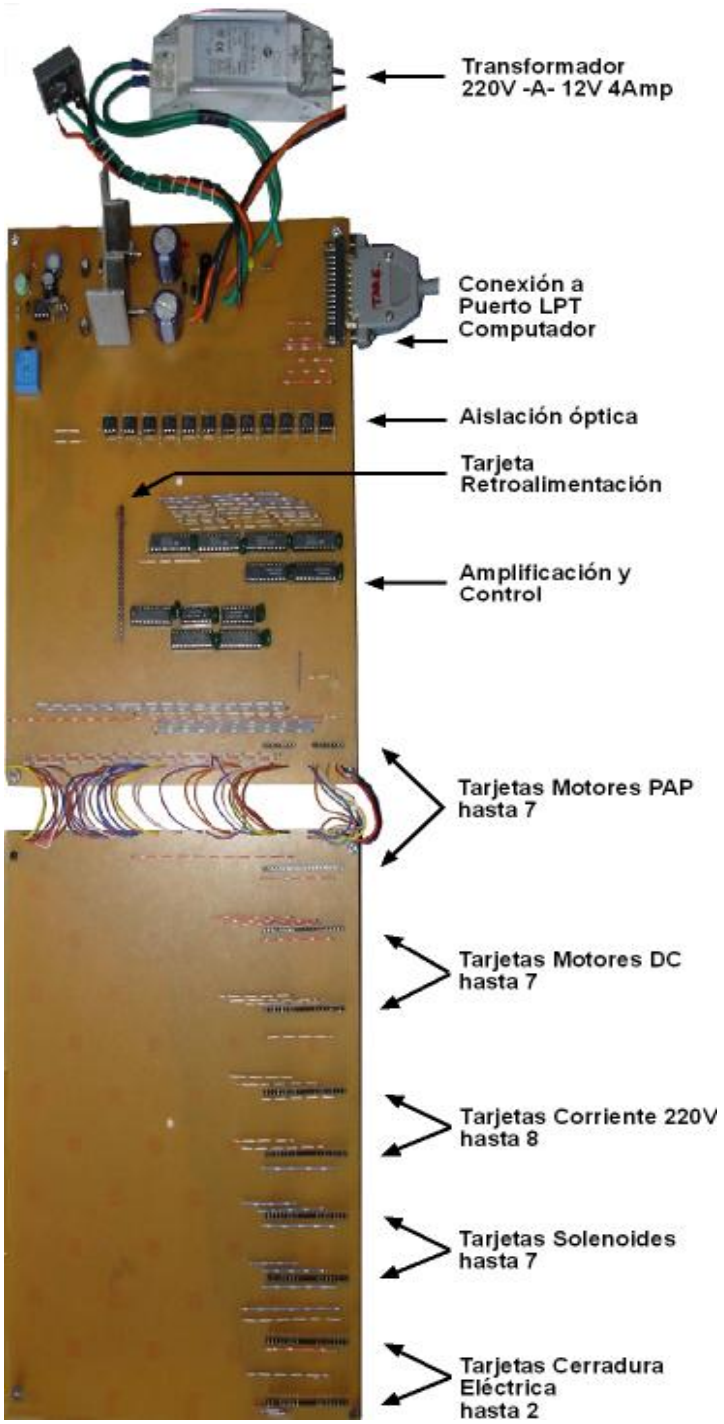

Figura Nº 9: Implementación tarjeta principal circuito eléctrico mecánico.

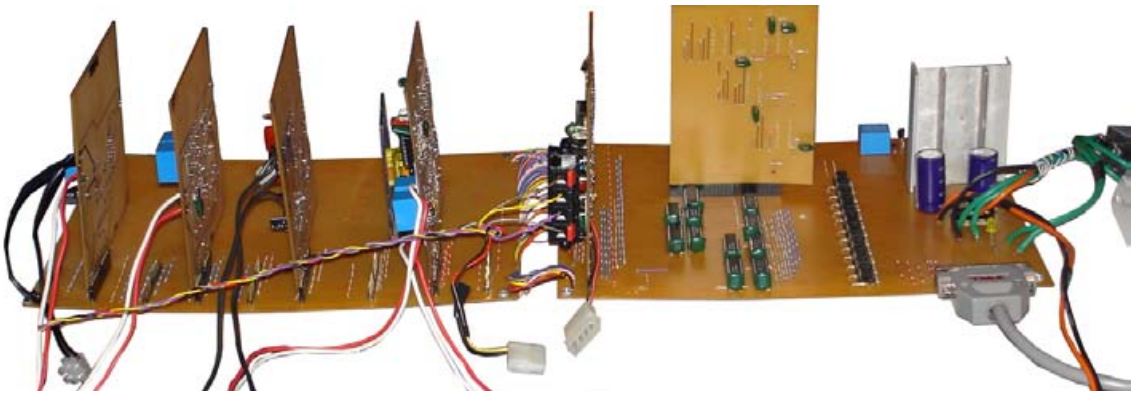

Figura Nº 10: Tarjeta principal con tarjetas de expansión domóticas.

6.4 Pruebas

Las pruebas de los circuitos electrónicos se incluyen dentro de la sección de software, por lo cual no se ahondará respecto a este tema en esta sección.

7. Mecánica

El accionar mecánico que se requiere al manipular componentes del hogar como puertas o ventanas, puede tornarse en una labor compleja para las personas que utilizan aparatos para la asistencia en el desplazamiento como sillas de ruedas, andadores muletas, etc., o que ven su movilidad restringida producto de la vejez, reposo médico, fracturas, etc. Por esto, se deben automatizar estos procesos para lograr manipularlos de forma sencilla desde un lugar centralizado, y controlados por la electrónica y el software implementados para estas labores.

Dentro de los aspectos considerados a automatizar, se encuentra el de los procesos de abrir o cerrar una puerta o ventana, puesto que estas son algunas de las acciones que más frecuentemente las personas realizan al interactuar con su residencia y que, además, requieren el diseño de partes mecánicas para su automatización.

7.1 Análisis

La automatización del proceso de abrir o cerrar una puerta o ventana, implica que estas acciones deben ser realizadas por un sistema que considere la apertura o cierre de un pestillo (solenoide) o cerradura eléctrica, y el posterior movimiento de la puerta o ventana, mediante un motor que gire en uno u otro sentido según el comando seleccionado. Para esto se pensó en un motor de alza vidrio eléctrico para automóviles, ya que estos se encuentran relativamente fácil y cuentan con la potencia necesaria para llevar a cabo esta acción. Debido a esto, se analizó que la mejor opción era la de incorporar un sistema de cremallera (engranaje dentado que sirve como guía) fijo al marco de la puerta o ventana. De esta forma, el motor queda fijo a la puerta o ventana con un engranaje recto o cilíndrico montado o acoplado al eje del motor, así cuando el motor gire en uno u otro sentido seguirá el trayecto trazado por la cremallera, esto es posible ya que una puerta o ventana dibuja una semicircunferencia al abrirse o cerrarse.

Los cálculos que se utilizaron para especificar las medidas del engranaje recto y la cremallera son los siguientes.

7.1.1 Ecuaciones para cálculo de engranajes rectos y cremallera

Las formulas matemáticas que se utilizaron para el cálculo del engranaje recto y la cremallera son las siguientes:

- *M* = Módulo (grosor de la fresadora u otra herramienta con la que se vaya a cortar la pieza).
- *Dp* = Diámetro primitivo del diente.
- *De* = Diámetro exterior del diente.
- *Di* = Diámetro interior del diente.
- H = Altura del diente.
- *Pc* = Paso circunferencial.
- *e* = Espesor del diente.

De acuerdo con esto se tiene que:

$$M = Dp / Z$$
$$Dp = M * Z$$
$$Dp = De - 2 * M$$
$$De = Dp + 2 * M$$
$$Di = M * (Z - 2.314)$$
$$Di = De - 2 * H$$
$$H = 2.157 * M$$
$$M = Pc / \pi$$
$$e = Pc / 2$$
$$e = (M * \pi) / 2$$

Los números que aparecen en las fórmulas, son constantes numéricas utilizadas por las ecuaciones.

7.1.2 Cálculo de engranaje recto

De acuerdo con las formulas anteriores se realizaron los siguientes cálculos para la obtención de los valores para el engranaje recto:

$$De = 25mm$$
$$Z = 9$$
$$M = 2.5$$

$$Dp = 2.5 * 9$$
$$Dp = 22.5mm$$

$$Di = (Z - 2.314) * M$$
$$Di = (9 - 2.314) * 2.5$$
$$Di = 16.71$$
$$Di \approx 17mm$$

$$H = 2.157 * M$$
$$H = 2.157 * 2.5$$
$$H = 5.39$$
$$H \approx 5mm$$

$$e = (M * \pi) / 2$$
$$e = (2.5 * \pi) / 2$$
$$e = 3.927$$
$$e \approx 4mm$$

7.1.3 Cálculo de la cremallera o “guía”

Se realizaron los siguientes cálculos para la obtención de los valores para la guía o cremallera:

$$M = 2.5$$
$$Z = 168$$
$$De = 360\,mm$$

$$Dp = De - 2 * M$$
$$Dp = 360 - 2 * 2.5$$
$$Dp = 355\,mm$$

$$H = 2.157 * M$$
$$H = 2.154 * 2.5$$
$$H = 5.39$$
$$H \approx 5mm$$

$$Di = De - 2 * H$$
$$Di = 360 - 10$$
$$Di = 350\,mm$$

$$e = (M * \pi) / 2$$
$$e = (2.5 * 3.14) / 2$$
$$e = 3.92$$
$$e \approx 4mm$$

El cálculo aquí realizado es para una guía de 360º, de los cuales solo se utilizaran 90º, ya que esta cantidad en grados es la dibujada por el recorrido de una puerta o ventana al abrirse o cerrarse.

7.2 Diseño

Para el diseño de las piezas previamente analizadas y calculadas, así como las medidas para los modelos de puerta y ventana utilizados durante las pruebas, se utilizó la herramienta de diseño asistido por computadora AutoCAD, con la cual se generaron los diagramas que a continuación se presentan.

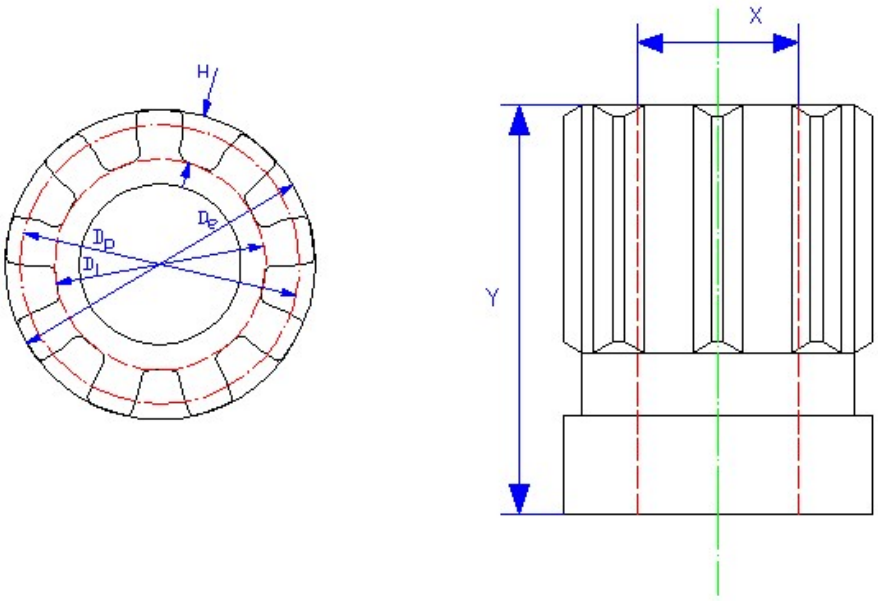

Diagrama Nº 18: Diseño engranaje recto para montar al eje del motor.

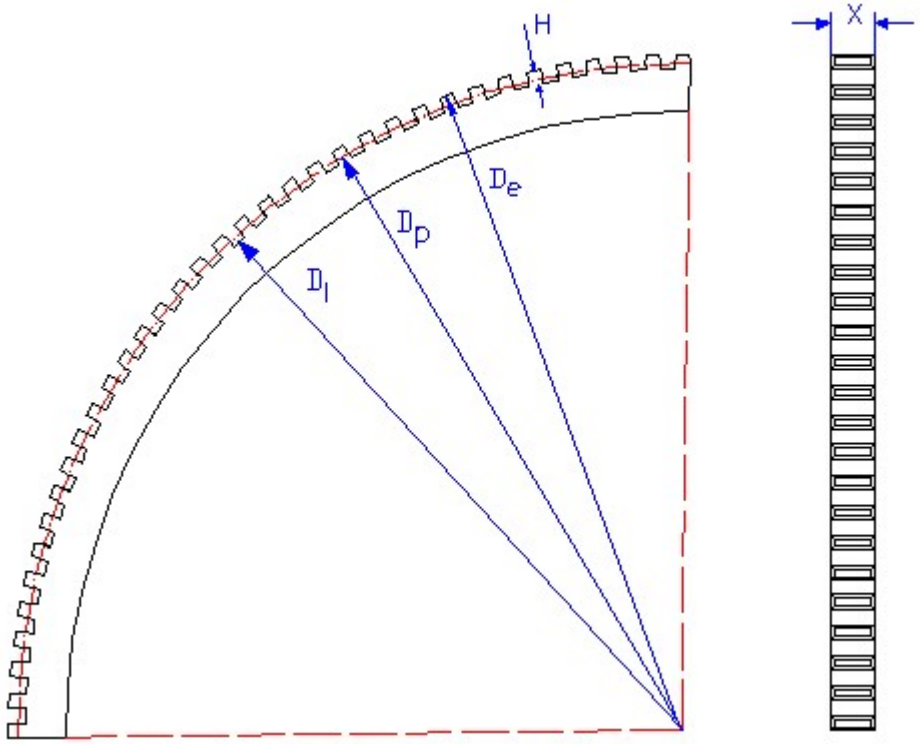

Diagrama Nº 19: Diseño cremallera o guía para montar al marco de la puerta o ventana.

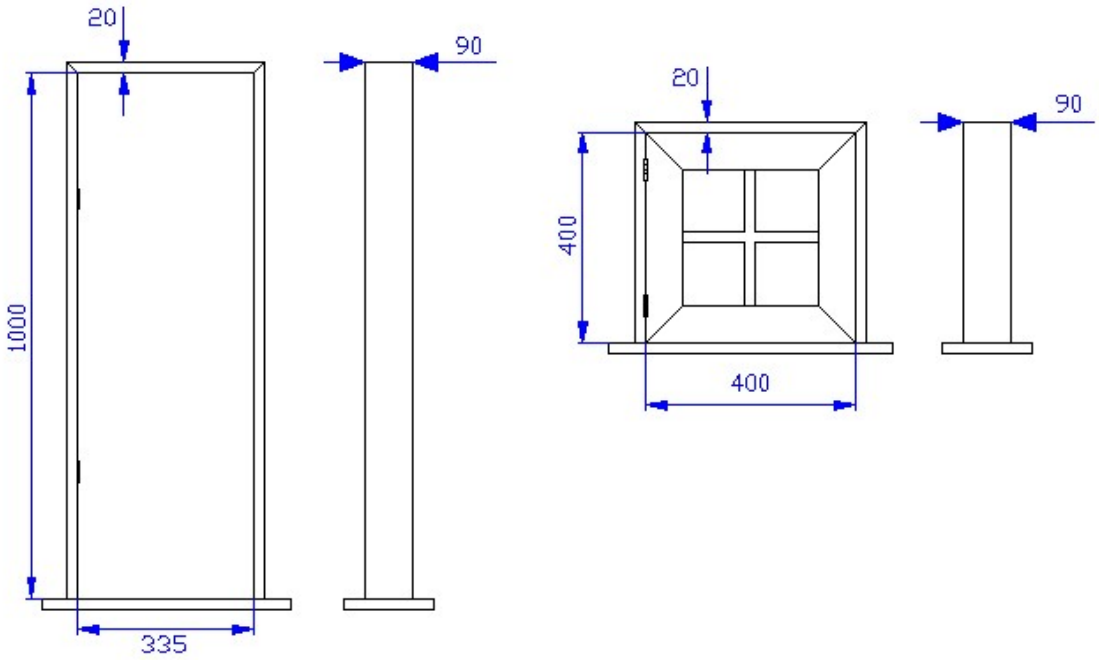

Diagrama Nº 20: Diseño de puerta y ventana modelo, utilizadas durante las pruebas.

No se incluyen diagramas de bloques o de ensambles puesto que, como se verá en la sección de implementación mecánica, el armado de las piezas se reduce a tan solo hacer coincidir el eje de giro de la puerta o ventana, con el centro de la circunferencia descrita por la cremallera, para luego solo fijar el motor en la posición correcta.

7.3 Implementación

El engranaje recto se implementó mediante un “Bendix” (engranaje del eje del motor de arranque de un automóvil), y el engranaje guía mediante el volante del motor de un vehiculo. Tal como se mencionó anteriormente el motor utilizado es el del alza vidrio eléctrico de un automóvil, el pestillo para la ventana, se implementó mediante un solenoide mecánico provisto por el cierre centralizado de un automóvil (seguro de la puerta del auto), y la cerradura de la puerta con una convencional cerradura eléctrica.

La implementación de los componentes mecánicos anteriormente diseñados se muestra a continuación en las siguientes figuras.

Figura Nº 11: Bendix utilizado como engranaje recto.

Figura Nº 12: Volante de Motor como guía para el recorrido de la puerta o ventana.

Figura Nº 13: Motor de alza vidrio eléctrico.

Este tipo de motores, por su potencia y versatilidad, no solo puede ser utilizado en puertas y ventanas, si no más bien en una serie de otras soluciones, que pueden comprender desde muebles, portones eléctricos o inclusive hasta camas ajustables, etc.

A continuación, se presenta el solenoide que en este caso será utilizado como cerrojo para una ventana modelo. Sin embargo, este tipo de dispositivo mecánico, puede ser utilizado para ayudar a resolver un sin numero de problemáticas, tales como abrir o cerrar una llave de agua o gas, elevar o descender distintos aparatos, etc.

Figura Nº 14: Solenoide de dos posiciones, utilizado como cerrojo de ventana.

Figura Nº 15: Puerta y Ventana modelo.

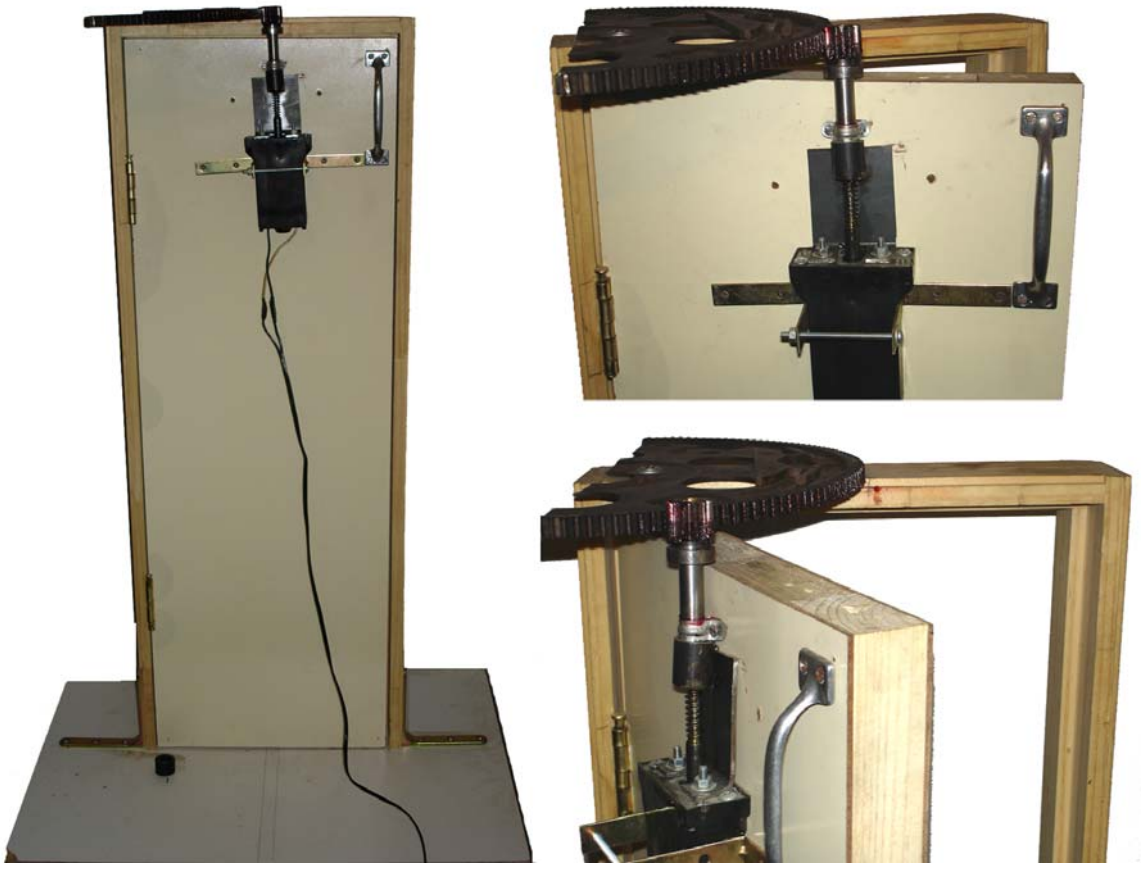

Figura Nº 16: Puerta modelo con sistema motorizado montado.

Figura Nº 17: Ventana modelo con sistema motorizado montado.

A continuación, se presentan algunas imágenes de dispositivos que fueron utilizados de forma directa o indirecta junto con las partes mecánicas aquí expuestas.

Figura Nº 18: Cerradura eléctrica junto a su transformador, utilizados en conjunto con la puerta modelo.

Figura Nº 19: Sistema mecánico motorizado, para giro de cámara de vigilancia.

Tal como se muestra en la figura Nº 19, se implemento un sencillo sistema para motorizar una común cámara Web, y de esta forma implementar un básico sistema de vigilancia, así como también probar la precisión del control de motores paso a paso.

Además de las figuras que se presentaron, se utilizaron otros diversos aparatos mecánicos y eléctricos para probar las distintas funcionalidades, tanto del sistema mecánico, así como también del sistema electrónico de control de éstos, algunos de éstos fueron: Lámparas, ampolletas, calentadores, etc.

7.4 Pruebas

Las pruebas mecánicas se incluyen dentro de la sección de software, por lo cual no se ahondará respecto a este tema en esta sección.

8. Software

Para el desarrollo de un proyecto de este tipo, la implementación del software, se convierte en una pieza fundamental, que aglutina a todos los demás componentes (electrónicos, mecánicos, etc.) para unificarlos bajo un solo sistema.

Tal como se mencionó anteriormente, el desarrollo del software, fue guiado por la metodología R.U.P, lo que permite un desarrollo del software orientado a objetos, lo cual, además, posibilita que a futuro se continúe con su evolución y mejoramiento de forma dinámica. Si bien R.U.P genera modelos iterativos durante todo el ciclo de vida del software, a continuación se presentan los modelos finales obtenidos según esta metodología, para este primer ciclo del software.

8.1 Análisis

Dentro del análisis en el desarrollo del software, la captura de requerimientos se torna una labor fundamental a la hora de encontrar las características que el software debe cumplir para completar con éxito los objetivos propuestos. Para esto, R.U.P, proporciona el “Diagrama de Casos de Uso”, el cual representa de forma gráfica (mediante diagramación U.M.L.) cuales son las acciones que el software debe contemplar, así como los

“Actores” involucrados en la acción. La incorporación de nuevos casos de uso y nuevos actores, posibilitan la expansión, evolución y mejoramiento continuo del software a través del tiempo.

Los distintos elementos de un diagrama de casos de uso, están sujetos a las definiciones entregadas por la metodología R.U.P, y también en este caso a las herramientas de diagramación UML proporcionadas por Rational Rose XDE. Luego del diagrama del modelo de casos de uso, se presenta una explicación mas en detalle para cada caso de uso expuesto.

A continuación se presenta el diagrama de casos de uso correspondiente al ciclo actual de desarrollo del software.

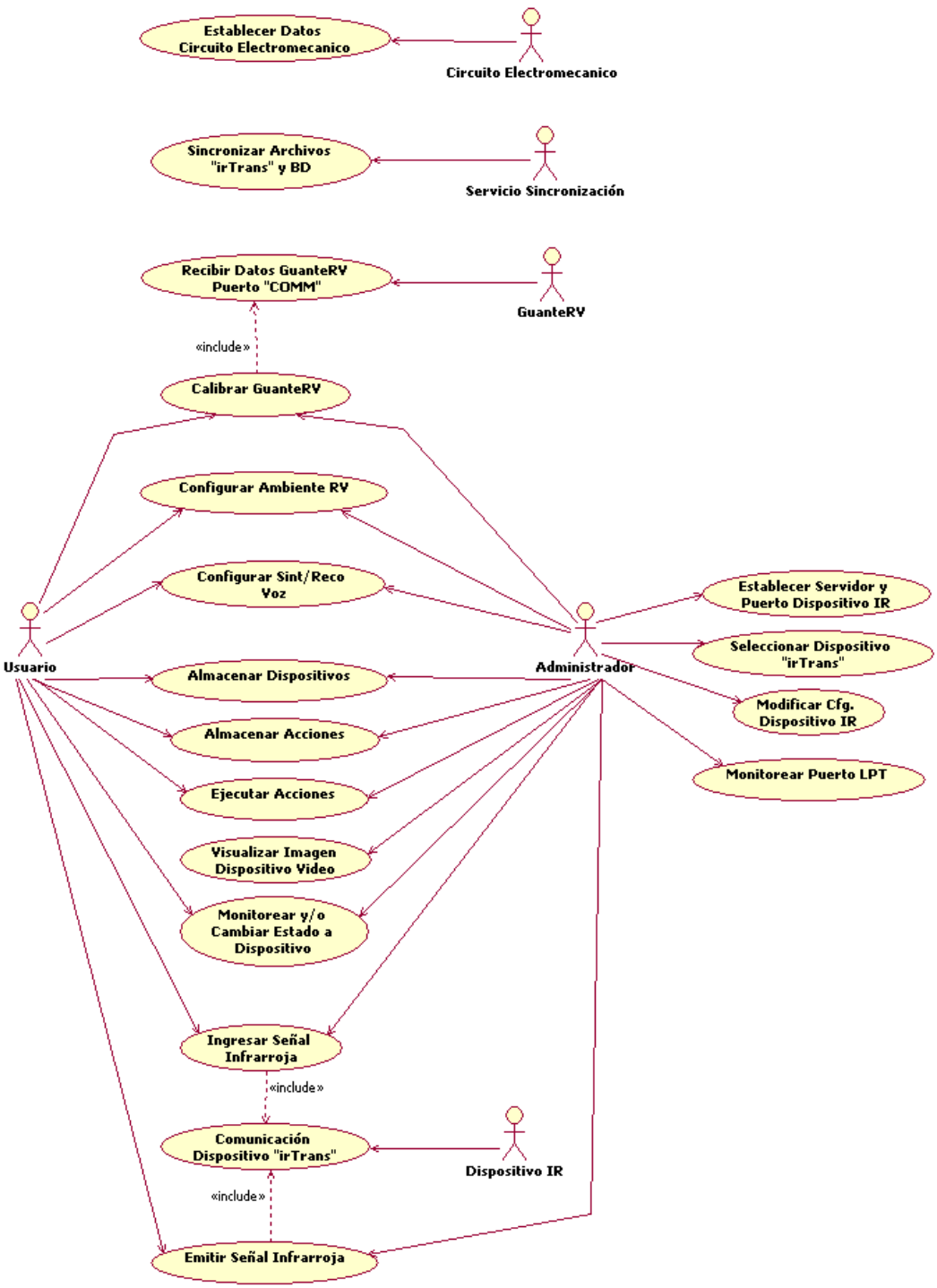

Diagrama Nº 21: Modelo de Casos de Uso.

El siguiente, es el análisis de cada caso de uso en particular dentro del modelo:

CASO DE USO: Establecer Datos Circuito Electromecánico.	
ACTOR (ES): Circuito Electromecánico.	
CURSO NORMAL	
1) La aplicación envía una señal de control al circuito Electromecánico, para manipular algún aparato conectado a este (Motor, Cerradura, etc.)	
2) El circuito responde al comando y activa la electrónica necesaria.	
3) El circuito envía la información de retroalimentación necesaria para que la aplicación determine el resultado de la ejecución.	

CASO DE USO: Sincronización archivos “irTrans” y BD.	
ACTOR (ES): Servicio de Sincronización.	
CURSO NORMAL	**ALTERNATIVAS**
1) El servicio de sincronización detecta la modificación de uno de los archivos que componen la base de datos ASCII del servidor “irTrans”.	
2) El servicio se conecta a la base de datos de la aplicación.	2.1) Si no se puede establecer la conexión con la BD, se debe general un registro “log” con información de lo ocurrido.
3) El servicio actualiza las tablas necesarias dentro de la BD.	
4) El servicio se desconecta de la BD.	

CASO DE USO: Recibir Datos GuanteRV Puerto “COMM”.	
ACTOR (ES): GuanteRV.	
CURSO NORMAL	
1) El hardware del guante de realidad virtual, envía datos respecto a la posición de la mano y dedos del usuario al puerto COMM de la computadora.	
2) La aplicación recibe la información del guante a través del puerto COMM y actualiza la representación tridimensional de la mano del usuario.	

CASO DE USO: Calibrar GuanteRV.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) El usuario o el administrador solicitan la calibración del guante de realidad virtual.	
2) Se establece inicialmente una calibración por defecto.	2.1) Si el usuario está a gusto con la calibración por defecto, se almacenan los valores de la calibración para el usuario actual.
3) Se seleccionan los valores de movilidad tanto de la mano como de los dedos del usuario, según las necesidades.	
3) Se almacenan los valores de la calibración para el usuario actual.	

CASO DE USO: Configurar AmbienteRV.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) El usuario o el administrador solicitan la configuración del ambiente virtual para un usuario en particular.	
2) Se presenta una configuración por defecto.	2.1) Si el usuario está a gusto con la configuración por defecto, se almacenan los valores de la configuración para el usuario actual.
3) Se establece el comportamiento de los objetos dentro del ambiente, así como también los patrones de movimientos del GuanteRV, teclas, y acciones del ratón, que realizaran los movimientos dentro del ambiente.	
3) Se almacenan los valores de la configuración para el usuario actual.	

CASO DE USO: Configurar SintRecoVoz.
ACTOR (ES): Usuario, Administrador.
CURSO NORMAL
1) El usuario o el administrador solicitan la configuración del sintetizador y reconocedor de voz para un usuario en particular.
2) Se establece el sexo y velocidad de la voz sintetizada, así como también los modos de reconocimiento de voz disponibles.
3) Se almacenan los valores de la configuración para el usuario actual.

CASO DE USO: Almacenar Dispositivos.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	
1) El usuario o el administrador ingresan un dispositivo acorde a sus necesidades, y que represente un artefacto que se desee controlar.	
2) Se almacenan los valores del dispositivo en particular.	

CASO DE USO: Almacenar Acciones.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) Se selecciona el dispositivo al cual se le desea asociar una acción.	
2) Se ingresa la información pertinente a la acción.	
3) Se asocia la acción con un objeto y método expuesto por el sistema.	3.1) Se asocia la acción con alguna otra acción perteneciente al dispositivo actual o a algún otro dispositivo, de manera de permitir la agrupación de acciones, en acciones más complejas y globales.

CASO DE USO: Ejecutar Acciones.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	
1) Se selecciona el dispositivo para el cual se desea ejecutar una acción.	
2) Se selecciona la acción del dispositivo que se desea ejecutar.	
3) Se ejecuta la acción propiamente tal.	

CASO DE USO: Visualizar Imagen Dispositivo Video.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) Se selecciona el dispositivo de video conectado al sistema.	
2) Se monitorea la fuente de video.	2.1) Opcionalmente, se puede cambiar la configuración respecto al tamaño y formato de video, así como también almacenar imágenes estáticas de la fuente de video monitoreada.
3) Se cierra el monitoreo de la fuente.	

CASO DE USO: Monitorear y/o Cambiar Estado a Dispositivo.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) Se selecciona el dispositivo definido por el usuario que se desea monitorear.	
2) Se verifica el estado actual del dispositivo seleccionado.	2.1) Opcionalmente, se puede alterar manualmente el estado del dispositivo, para que refleje su estado real.
3) Se finaliza el monitoreo del dispositivo.	

CASO DE USO: Ingresar Señal Infrarroja.	
ACTOR (ES): Usuario, Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) La aplicación envía la orden de “lectura” al dispositivo “irTrans”.	1.1) Si “irTrans” responde estar ocupado, se espera o cancela la acción.
2) Se le da un nombre a la señal que se almacenara.	
3) Se presiona el botón del control remoto original, apuntando al dispositivo “irTrans”.	2.1) Si no se lee la señal enviada se puede repetir o cancelar la acción.
4) Se almacena el código IR recibido.	

CASO DE USO: Comunicación Dispositivo “irTrans”.	
ACTOR (ES): Dispositivo IR.	
CURSO NORMAL	**ALTERNATIVAS**
1) El dispositivo “irTrans”, mediante su programa servidor, abre un socket de conexión.	1.1) Si no es posible abrir un socket en el puerto por defecto, se debe especificar otro número de puerto disponible.
2) Los clientes se conectan según sus necesidades a este socket, tanto para recibir como para enviar comandos al servidor “irTrans”, y de esta forma controlar el dispositivo “irTrans.	

CASO DE USO: Emitir Señal Infrarroja.
ACTOR (ES): Usuario, Administrador.
CURSO NORMAL
1) Se selecciona el “remoto” para el cual se desea enviar una señal infrarroja.
2) Se selecciona la señal o “comando” a enviar.
2) Se envía la señal.

CASO DE USO: Establecer Servidor y Puerto Dispositivo IR.	
ACTOR (ES): Administrador.	
CURSO NORMAL	
1) El administrador establece o modifica el equipo que actuara como servidor para el dispositivo “irTrans”.	
2) El administrador establece o modifica el puerto al cual se conectarán los clientes al servidor “irTrans”.	

CASO DE USO: Seleccionar Dispositivo “irTrans”.	
ACTOR (ES): Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) Puesto que el servidor “irTrans”, puede manejar hasta 16 dispositivos “irTrans” conectados a su bus, el administrador puede seleccionar de la lista cual será usado por defecto.	1.1) Si solo existe un dispositivo “irTrans”, se utilizará ese por defecto.
2) Se almacenan las modificaciones hechas a la configuración.	

CASO DE USO: Modificar Cfg. Dispositivo IR.	
ACTOR (ES): Administrador.	
CURSO NORMAL	**ALTERNATIVAS**
1) Se selecciona de acuerdo a las necesidades de las señales infrarrojas que se desean almacenar, los modos en los cuales debe trabajar el dispositivo “irTrans”, para una correcta lectura de estas.	1.1) Si el dispositivo “irTrans” informa estar ocupado, se puede esperar o cancelar la acción.
2) Se envía la orden al dispositivo “irTrans” para guarde la nueva configuración en su memoria EPROM.	

CASO DE USO: Monitorear Puerto LPT.	
ACTOR (ES): Administrador.	
CURSO NORMAL	
1) Si se presenta algún problema, el administrador, puede monitorear directamente los valores actuales del puerto LPT.	
2) Se corrigen los valores y se finaliza el monitoreo.	

8.2 Diseño

Durante la etapa iterativa del diseño de la aplicación en la metodología R.U.P, se generan una serie de diagramas que van siendo modificados, adaptados y ampliados a medida que el software se construye, haciendo de estos diagramas, un modo gráfico y dinámico de entender la estructura y comportamiento de la aplicación.

Sin duda que dos de los diagramas más importantes dentro del diseño del software, son el “Modelo de Clases”, que identifica las entidades o clases que componen la aplicación, así como la relación entre ellas, y el modelo “Entidad-Relación” que será el encargado de dar forma y estructura a la base de datos de la aplicación.

Si bien, existen otros diagramas importantes como los “Diagramas de Interacción” y los “Diagramas de Secuencias”, estos no serán incluidos, puesto que brindaron su ayuda durante el proceso de desarrollo, y actualmente no representan fielmente el estado actual del sistema.

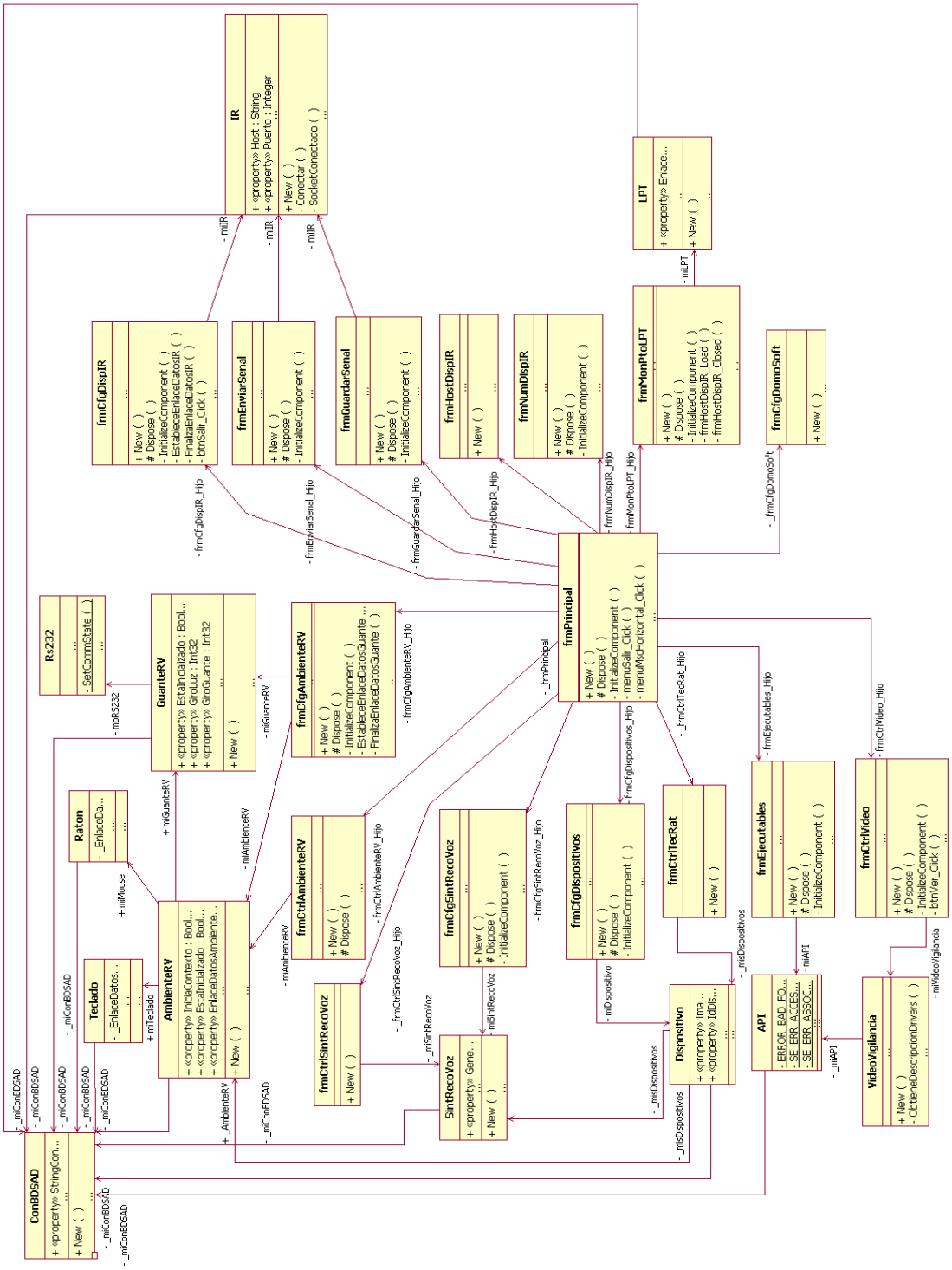

Diagrama Nº 22: Modelo de Clases.

Se presenta a continuación, una descripción de cada una de las clases del modelo. Para realizar esta descripción, se utilizarán las tarjetas “CRC” (“Clase-Responsabilidad-Colaboración”), propuesta por la diagramación UML para estos propósitos. No se describirán las clases que comienzan con el prefijo “frm”, ya que corresponden solo a clases de interfaces graficas de usuario o “GUI”, las cuales tienen como propósito solamente desplegar la información de las clases que componen la aplicación, y que por lo tanto, podrían ser implementadas de distinta manera, o incluso en formato de programa de consola.

Responsabilidad	ConBDSAD	Colaboración
- Proporcionar la disponibilidad de conexión a la Base de Datos para las demás clases.		

Responsabilidad	Teclado	Colaboración
- Mantener la información de las teclas presentes en el sistema y sus correspondientes códigos ASCII.		ConBDSAD

Responsabilidad	Raton	Colaboración
- Mantener la información de las acciones del ratón disponibles en el sistema y sus correspondientes códigos asignados.		ConBDSAD

Responsabilidad	Rs232	Colaboración
- Permitir la conexión y recepción de datos a través del puerto serial.		

Responsabilidad	GuanteRV	Colaboración
- Mantiene la información del guante de realidad virtual utilizado por el usuario.		ConBDSAD
- Permitir la calibración del rango de movimientos de los dedos y del giro e inclinación de la muñeca.		
- Permitir la calibración de los patrones de movimientos que serán detectados por la clase para la ejecución de acciones.		
- Mantener actualizada la representación tridimensional de la mano del usuario.		Rs232

Responsabilidad	AmbienteRV	Colaboración
- Mantener la configuración del ambiente virtual para cada usuario.		ConBDSAD
- Permitir establecer las teclas que efectuaran acciones dentro del ambiente.		Teclado
- Permitir establecer las acciones del ratón que efectuarán acciones dentro del ambiente.		Raton
- Permitir establecer los patrones de movimientos del GuanteRV que efectuarán acciones dentro del ambiente.		GuanteRV
- Ejecutar las acciones seleccionadas por el usuario para cada dispositivo, según se requiera.		Dispositivo

Responsabilidad	SintRecoVoz	Colaboración
- Mantener la configuración del sintetizador y reconocedor de voz para cada usuario.		ConBDSAD
- Ejecutar las acciones seleccionadas por el usuario para cada dispositivo, según se requiera.		Dispositivo

Responsabilidad	Dispositivo	Colaboración
- Establecer la información definida por el usuario respecto a los dispositivos y sus acciones asociadas.		ConBDSAD
- Asociar las acciones de los dispositivos a clases y métodos expuestos por el sistema.		

Responsabilidad	API	Colaboración
- Hacer disponibles las funciones de la API de Windows que se requieran, para ser utilizadas según sea necesario por las demás clases del sistema.		

Responsabilidad	VideoVigilancia	Colaboración
- Poner a disposición del sistema, las distintas fuentes de video conectadas al computador.		API
- Modificar el formato de la fuente de video.		API
- Capturar imágenes fijas de la fuente de video.		API

Responsabilidad	IR	Colaboración
- Establecer conexión mediante socket con el servidor “irTrans”.		
- Enviar comandos y órdenes al servidor “irTrans”.		
- Enviar señales infrarrojas.		
- Permitir la recepción y almacenamiento de las señales infrarrojas.		
- Modificar el modo de funcionamiento del dispositivo “irTrans”.		
- Establecer el dispositivo “irTrans” por defecto para la aplicación.		ConBDSAD
- Establecer el equipo servidor o “Host” y su puerto para el dispositivo “irTrans”.		ConBDSAD

Responsabilidad	LPT	Colaboración
- Mantener la información respecto a los puertos y buses de estos disponibles en el sistema.		ConBDSAD
- Permitir el envío de bytes a través del puerto LPT seleccionado.		
- Permitir la recepción de bytes a través del puerto LPT seleccionado.		
- Permitir un monitoreo de los bytes que entran y salen por el puerto LPT seleccionado.		

8.3 Implementación

La implementación de esta aplicación, planteó una serie de retos debido principalmente a las características con las cuales debía cumplir. Esto es:

- Control a bajo nivel del puerto LPT: Control de circuito Eléctrico/Mecánico.

- Comunicación serial: Comunicación con guante de realidad virtual “NewGlove”.

- Gráficas OpenGl: Ambiente virtual.

- Sonido OpenAl: Ambiente Virtual.

- Servicios de sincronización entre archivos y base de datos: Archivos servidor “irTrans”.

- Comunicación socket: Comunicación con servidor “irTrans” y software cliente.

- Reconocimiento de voz: Interfaz de usuario mediante voz.

Favorablemente y aunque no todas estas características se encuentran disponibles de forma nativa en el Framework .NET de Microsoft, fue posible desarrollar todos los requerimientos del sistema, tanto con características internas del Framework .NET, como con la ayuda de librerías externas para diversos propósitos.

En consecuencia, la implementación del software, se desarrollo en Visual Basic .NET, que al presentarse como un lenguaje totalmente renovado y 100% orientado a objetos, sumado a la potencia y portabilidad del Framework .NET V1.1, hicieron que la aplicación cumpliera con los objetivos propuestos para el desarrollo, permitiendo también, la posibilidad de desarrollos futuros tanto para plataforma Windows como para el sistema operativo Linux mediante el proyecto “Mono” (Mono: Framework libre y multiplataforma, desarrollado por Novell/Ximian y compatible con Microsoft Framework .NET).

Ademas, las fuentes del software se distribuirán como “OpenSource” bajo los términos de la licencia “G.P.L” (“General Public License”, o “Licencia Publica General” por sus siglas en ingles), lo cual permite la colaboración libre a futuro de desarrolladores en todo el mundo para mejorar y expandir el sistema.

A continuación se presentan las características principales de las clases que componen la aplicación, la documentación completa del código de las clases, se encuentra incluida en forma de comentarios dentro de la implementación de estas y se adjuntan en “X:\Fuentes\DomoSoft”:

Clase “ConBDSAD”:

Esta clase encapsula el código necesario para la conexión con la base de datos, de esta forma, las demás clases del sistema pueden definir objetos de este tipo y conectarse a la BD cuando lo requieran.

Para realizar la conexión, se agrega como referencia al proyecto la librería “AseClient” (“Sybase.Data.AseClient.dll”) que se incluye con la instalación del servidor “Sybase ASE 12.5” y sus herramientas cliente. Esta librería, actúa como cliente ADO .NET y provee el espacio de nombre “Sybase.Data.AseClient”, el cual fue utilizado en esta clase.

La clase se encarga de mantener los valores necesarios para una correcta conexión con el servidor de base de datos “Sybase ASE”, y solo requiere, para la creación de un objeto de este tipo, que se entreguen los datos del nombre de usuario y contraseña que se ingresaron al iniciar la aplicación, al constructor de la clase. Esto se realiza de la siguiente forma:

```
…
Imports Sybase.Data.AseClient
…
Public Class ConBDSAD2
    …
    Private _NomBD As String = "BDSAD"
    Private _NomServidor As String = "NOTEBOOK"
    Private _PtoServidor As String = "5001"
    Private _AseConBDSAD As AseConnection
    …
      Public Sub New(ByVal idUsuario As String, ByVal Contrasena As
      String)
        _idUsuario = idUsuario
        _Contrasena = Contrasena
End Sub 'New
```

La conexión se establece mediante la función “Conectar”, y de esta forma, esta función devuelve un objeto de tipo “AseConnection”. A continuación se muestra el extracto del código que realiza esto:

```
Public Function Conectar() As AseConnection
        …
        Try
            _AseConBDSAD = New AseConnection(_strConexionUsado)
            Return _AseConBDSAD

        Catch Ex As AseException
            MessageBox.Show("Ha ocurrido un error en la aplicación." &
            " Descripción del error:" & Ex.Message & " Fuente del
            error:" & Ex.Source, "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error, MessageBoxDefaultButton.Button1,
            MessageBoxOptions.DefaultDesktopOnly)
            _AseConBDSAD.Close()
        End Try
End Function 'Conectar
```

Clase “Rs232”:

Debido a que el Framework .NET se presenta como una nueva arquitectura, es que ya no se utiliza de forma nativa la tecnología “COM” (también conocida como “ActiveX”), esto implica que el objeto “COMM” utilizado en las versiones previas de Visual Basic para establecer comunicación serial, solo puede ser utilizado mediante el servicio “Interop” provisto por .NET, el cual provee compatibilidad con este tipo de objetos. Sin embargo, la utilización de “Interop” presenta como desventaja tiempos de ejecución más largos y la posibilidad de presentar problemas de incompatibilidad o errores difíciles de

controlar, debido a que el modelo de memoria utilizado en la arquitectura “ActiveX” (memoria no manejada), es distinta de la que presenta .NET (memoria manejada). Por esto se decidió utilizar la clase “Rs232” escrita por Corrado Cavalli, la cual, se puede descargar gratuitamente desde Internet. La documentación y descripción de esta clase, se encuentra en forma de comentarios al interior de la misma.

Clase “GuanteRV”:

Esta clase se encarga de todo lo relacionado con la interfaz de realidad virtual “NewGlove”, y por lo tanto utiliza un objeto de tipo “Rs232” para comunicarse con el hardware del guante.

Puesto que esta clase debe tomar los datos enviados por el guante de realidad virtual y recibidos por el puerto serie, para luego hacer una representación grafica tridimensional de la mano del usuario, es que se agregaron referencias en el proyecto a las librerías “Tao.OpenGl” y “Tao.Platform.Windows”, las cuales encapsulan los métodos y funciones OpenGl y un control contenedor que permite renderizaje OpenGl respectivamente.

Esta clase implementa el procedimiento delegado “pUICommEventUpdate” del evento de recepción de datos del puerto serie

“moRS232_CommEvent“, para gatillar la actualización de la representación tridimensional de la manos, esto se realiza de la siguiente forma:

```
Private Sub moRS232_CommEvent(ByVal source As Rs232, ByVal Mask As
Rs232.EventMasks) Handles moRS232.CommEvent
      …
      Dim oUIUpdater As New CommEventUpdate(AddressOf
      pUICommEventUpdate)
      Dim args(1) As Object
      args(0) = source
      args(1) = Mask
      oUIUpdater.Invoke(source, Mask)
      …
End Sub
```

El método “Invoke” está disponible para cualquier objeto definido por el programador o por el propio lenguaje, y se utiliza para llamar a un método del mismo objeto por su nombre, es decir en vez de escribir el nombre del método y poner entre paréntesis los parámetros que espera, se puede pasar como un string que contenga el nombre del método y en un arreglo de objetos pasar los parámetros que el método espera. En este caso, y al tratarse de un evento, el objeto “oUIUpdater”, utiliza el método “Invoke” para gatillar una llamada al delegado “CommEventUpdate”, pero el delegado solo especifíca la firma del tipo y cantidad de parámetros que el método recibe, el método que efectivamente es ejecutado al gatillarse el evento, está definido luego de la instrucción “AddressOf”, por lo tanto el método a ejecutar es “pUICommEventUpdate”.

Este método es el que efectivamente ejecuta la actualización de los valores del dibujo:

```
Private Sub pUICommEventUpdate(ByVal source As Rs232, ByVal mask As
Rs232.EventMasks)
        If (mask And Rs232.EventMasks.RxChar) > 0 Then
            gActualizacionGuante()
        End If
End Sub
```

El método “gActualizacionGuante”, lee el buffer del puerto serial de la siguiente forma:

```
Private Sub gActualizacionGuante()
      …
      Dim Buffer() As Byte
      Try
            Buffer = moRS232.InputStream
            …
            ActualizaDatos()
            FiltroDatos()
            …
        Catch Ex As Exception
            'Llamada
            …
        End Try
End Sub
```

Luego se reordenan los valores de un arreglo que mantiene los datos por separado, y tal como se muestra hacia el final del método, se realiza una llamada a “FiltraDatos”, el cual se encarga de realizar los cálculos y validaciones necesarias antes de llamar a la actualización de las distintas secciones del dibujo (dedos, palma, etc.)

El dibujo en concreto de la mano se realiza en el manejador del evento “Paint” del control grafico del dibujo, el cual es entregado como parámetro en el constructor de la clase.

```
Private Sub ControlDespliegue_Paint(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.PaintEventArgs)
      Me.DibujaGuante()
      …
End Sub
```

Luego de gatillarse el evento (señal del sistema operativo, actualización de valores, etc.), se realiza la llamada a “DibujaGuante”, el cual ejecuta las instrucciones OpenGl y métodos necesarios para la representación virtual.

```
Public Sub DibujaGuante()
      If Me._IniciaContexto = True Then
            Dim Posicion As Single() = {0, 0, 1.5, 1}
            …
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT Or
            Gl.GL_DEPTH_BUFFER_BIT)
            'Luz movible
            Gl.glPushMatrix()
            Gl.glTranslatef(0, 0, -5)
            Gl.glPushMatrix()
            Gl.glRotated(_GiroLuz, 1, 0, 0)
            Gl.glRotated(0, 1, 0, 0)
            Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_POSITION, Posicion)
            Gl.glTranslated(0, 0, 2.5)
            Gl.glDisable(Gl.GL_LIGHTING)
            …
            EstructuraMano()
            …
End Sub
```

El método “EstructuraMano” finalmente realiza la representación grafica de la mano del usuario según los datos actuales.

```
Private Sub EstructuraMano()
      …
      'Mano
      Gl.glPushMatrix()
      'Posicion y escala inicial
      Gl.glTranslatef(_PosXGuanteRV, _PosYGuanteRV, _PosZGuanteRV)
      Gl.glScalef(_EscalaXGuanteRV, _EscalaYGuanteRV, _EscalaZGuanteRV)
      …
      Gl.glTranslatef(0, -0.5, -7)
      Gl.glRotatef(_GiroGuante, 0, 0, 1)
      Gl.glRotatef(_InclinacionGuante, 1, 0, 0)
      …
      Gl.glPushMatrix()
      Gl.glScalef(3.0#, 2, 0.5)
      Gl.glCallList(_mPalma)
      Gl.glPopMatrix()
      …
      Gl.glPushMatrix()
      Gl.glTranslatef(1.0#, 1.25, 0.0#)
      iCuadroClave = vIndice(_iClave, PRIMER_DEDO)
      Indice(iCuadroClave)
      Gl.glPopMatrix()
      …
End Sub
```

Tal como se puede apreciar en el extracto de código, las instrucciones y funciones OpenGl, se presentan precedidas por “Gl”, esto es debido a que se importó al comienzo de la clase el espacio de nombres “TAO.OpenGl”, lo cual hace posible este tipo de codificación.

Esta clase pone a disposición de la aplicación todos las propiedades necesarias para una correcta configuración de un objeto de este tipo, esto se realiza creando una tabla virtual, la cual mantiene los valores actualizados, y

luego la clase proporciona funciones de tipo “Binding”, las cuales pueden proporcionar enlaces a distintas propiedades de los objetos gráficos de interfaz de usuario (cuadros de texto, etiquetas, etc.). Esto se realiza de esta forma:

```
Public Function EnlaceDatoValMaxMedio() As Binding
        EnlaceDatoValMaxMedio = New Binding("Text", _SDtblGuanteRV,
        Me._NomTblColVMaxMedio)
        Return EnlaceDatoValMaxMedio
End Function
```

En este ejemplo, la función “EnlaceDatoValMaxMedio”, proporciona un enlace a la propiedad “Text” de un cuadro de texto (u otro objeto grafico) cualquiera, que la agregue a sus enlaces de datos mediante “DataBindings.Add” (pudiendo utilizarse otras propiedades como: Checked, Visible, etc.). De esta forma el cuadro de texto que realice esto, tendrá constantemente actualizado el valor contenido en el set de datos “_SdtblGuanteRV”, en su columna “_NomTblColVMaxMedio”, en este caso el valor máximo configurado por el usuario para el dedo medio.

Este tipo de enlaces es utilizado tanto por los valores de la configuración del usuario, así como también por los datos correspondientes a la tabla virtual que almacena los datos recibidos en tiempo real a través del puerto serie, y de la misma forma, también por las demás clases del sistema con sus valores respectivos.

Pero también ésta y las demás clases del sistema, ponen a disposición, distintas propiedades para una correcta configuración del objeto. Un ejemplo de cómo se realiza esto se presenta a continuación:

```
Public Property ActDibPulgar() As Boolean
        Get
            Return _ActDibPulgar
        End Get
        Set(ByVal Value As Boolean)
            _ActDibPulgar = Value
            If Me._EnlaceDatosGuanteRVPosible = True Then
                  _SDtblGuanteRV.Tables.Item(_NomTblCalibrarGuanteRV).R
                  ows(0).Item(_NomColActDibPulgar) =
                  Math.Abs(CType(Me._ActDibPulgar, Integer))
            End If
        End Set
End Property
```

De esta forma, las interfaces de usuario, gráficas o de consola, pueden utilizar los métodos de enlace “Binding” o las propiedades según lo requieran.

Para anunciar la detección de un patrón de movimientos, la clase “GuanteRV”, define el evento “EventoDeteccionPatron” y su correspondiente delegado de la siguiente manera:

```
Public Event EventoDeteccionPatron As ManejadorEventoDeteccionPatron
Public Delegate Sub ManejadorEventoDeteccionPatron(ByVal TipoMovimiento
As PATRONMOVIMIENTO)
```

De esta forma, cuando el algoritmo de detección de patrones de movimiento detecta uno, el evento es gatillado de la siguiente manera:

```
Private Sub ReconocePatronFlexPulgar(ByVal Dato As Integer, ByVal
Sentido As SENTIDOMOVIMIENTO)
        If (Sentido = _SentidoFlexPulgar) Then
            _SwRecPatFlexPulgar = True
            If _ContGrabFlexPulgar = 0 Then
                _GrabPatFlexPulgar(_ContGrabFlexPulgar) = Dato
            End If
            _ContGrabFlexPulgar += 1
        ElseIf Sentido <> _SentidoFlexPulgar Then
            _ContGrabFlexPulgar = 1
            _GrabPatFlexPulgar(_ContGrabFlexPulgar) = Dato
            Dim CalRng, RngSup, RngInf, AuxRng As Single
            CalRng = _GrabPatFlexPulgar(0) -
            _GrabPatFlexPulgar(_ContGrabFlexPulgar)
            RngSup = _RngFlexPulgar + _ErrFlexPulgar
            RngInf = _RngFlexPulgar - _ErrFlexPulgar
            If RngInf > RngSup Then
                AuxRng = RngSup
                RngSup = RngInf
                RngInf = AuxRng
            End If
            If (CalRng <= RngSup) And (CalRng >= RngInf) Then
                If _SwRecPatFlexPulgar = True Then
                    If _RngFlexPulgar > 0 Then
                        TablaDesplieguePatron.Rows(0).Item(0) =
                        "Patron: Flexión pulgar"
                        If sw = True Then
                              RaiseEvent
                              EventoDeteccionPatron(PATRONMOVIMIENTO.FL
                              ECTAR_PULGAR)
                            _Tpr.Enabled = True
                        End If

                    End If
                End If
            End If
            _ContGrabFlexPulgar = 0
            _SwRecPatFlexPulgar = False
        End If
End Sub
```

En este caso, se presentan los cálculos necesarios para la detección del patrón de movimiento correspondiente a la flexión del dedo pulgar. De reconocerse el patrón, se gatilla el evento mediante la instrucción “RaiseEvent”,

la cual lo gatilla entregándole como parámetro el patrón reconocido, y de esta forma poder usar esta información para el control del ambiente virtual, etc.

Clase “AmbienteRV”:

Esta clase, se encarga de la representación virtual del ambiente en donde se controlarán los dispositivos y sus acciones asociadas. La representación gráfica de esto, la realiza al igual que la clase anterior, mediante OpenGl.

Es interesante notar en esta clase el manejo del evento de detección de patrones movimiento de la clase “GuanteRV”, ya que el ambiente virtual incluye dentro de esta la representación de la mano del usuario dentro del ambiente. Esto se realiza de la siguiente forma:

```
…
AddHandler Me.miGuanteRV.EventoDeteccionPatron, AddressOf
GuanteRV_ReconocimientoPatron
…

Private Sub GuanteRV_ReconocimientoPatron(ByVal TipoPatron As
GuanteRV.PATRONMOVIMIENTO)
      …
End Sub
```

La utilización de la instrucción “AddHandler”, permite lo que se conoce como “Enlace Tardío”, lo que permite que se enlace el método que manejará el evento solo cuando se requiera, y de esta manera no mantener o eliminar este

enlace (“RemoveHandler”) si el usuario por ejemplo, no utiliza el guante de realidad virtual como interfaz para el sistema.

Luego el método “GuanteRV_ReconocimientoPatron”, se encarga de realizar la acción necesaria dentro del ambiente, según la configuración del usuario y el tipo de patrón reconocido, recibido como parámetro en el método.

El ambiente virtual se presenta como un par de tarjeteros, uno vertical y otro horizontal, el primero muestra los dispositivos configurados en el sistema, y el segundo, las acciones asociadas a cada unos de estos dispositivos, entonces, por ejemplo, en el tarjetero vertical podemos tener dispositivos como “Televisor” o “Puerta”, y en el horizontal las acciones para cada uno de estos: “Encender, Apagar” para “Televisor” y “Abrir, Cerrar”, para “Puerta”. Entonces, el usuario puede mover, mediante el teclado, mouse o guante de realidad virtual, estos tarjeteros para seleccionar tanto el dispositivo como la acción que desea ejecutar. Cuando se produce el movimiento de alguno de estos tarjeteros, la clase realiza una animación, que muestra la nueva tarjeta que contiene ya sea un dispositivo o acción. La ejecución de esta animación, puede producir el bloqueo de la interfaz grafica mientras se está ejecutando, para lo cual el método que ejecuta esta animación, fue puesto en un “Hilo” de ejecución diferente al de la interfaz gráfica para que no se produzca este fenómeno. Esto se realizó de la siguiente manera:

```
Private Sub GirarAbajo()
        If _ControlGiroAbajo = True Then
            _ControlGiroAbajo = False
            …
            Al.alSourcePlay(_Fuente(SONIDOS.SONIDO_ABAJO))
            …
            Dim _HiloAbajo As New Thread(AddressOf DibujoGirarAbajo)
            _HiloAbajo.Start()
        End If
End Sub
```

Tal como se puede apreciar, la utilización de hilos en .NET, es muy sencilla de realizar, solo basta definir una variable de tipo “Hilo” (Thread), y especificar mediante “AddressOf”, el método que se debe ejecutar en este hilo separado. Luego solo hace falta iniciar la ejecución del hilo mediante el método “Start”.

Dentro de este mismo procedimiento, cabe destacar la utilización de la librería “Tao.OpenAl”, la cual implementa la utilización de “OpenAl” en lenguajes .NET. De esta forma, se puede utilizar OpenAl para la reproducción de sonidos envolventes, ya que tal como OpenGl se encarga de los gráficos tridimensionales, OpenAl se encarga de que los sonidos se reproduzcan de forma envolvente o tridimensional. Para poder realizar esto, se deben implementar las inicializaciones respectivas de la librería, para luego poder reproducir el sonido que se desea tal como se muestra en la línea “Al.alSourcePlay(_Fuente(SONIDOS.SONIDO_ABAJO))”, lo cual reproducirá el sonido configurado por el usuario cuando el tarjetero vertical se mueva hacia abajo.

Clase “IR”:

Esta clase, se encarga de todo lo relacionado con el manejo de señales infrarrojas, para lo cual le es imprescindible implementar una comunicación mediante “Socket” con el servidor de dispositivos “irTrans”. Anteriormente, los lenguajes de programación de Microsoft, utilizaban para esto el objeto ActiveX “WinSock”, pero debido a que corresponde a la arquitectura antigua de componentes, ahora .NET ofrece la clase “Socket”, por lo tanto la clase “IR”, importa el espacio de nombres “System.Net.Sockets”, y la implementación en el código se realizó de la siguiente manera:

```
Private Sub Conectar()
        Try
            _Cliente = New Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp)
            …
            _Cliente.BeginConnect(EPRemoto, AddressOf SocketConectado,
            _Cliente)
            …
End Sub


Private Sub SocketConectado(ByVal ar As IAsyncResult)
        Try
            If _Cliente.Connected = False Then
                …
                RaiseEvent SocketError("Conexion rechazada.")
                Exit Sub
            End If
            _Cliente.BeginReceive(_Estado.Buffer, 0,
            _Estado.TamanoBuffer, 0, AddressOf SocketArrivoDatos,
            _Estado)
            …
            RaiseEvent ConexionEstablecida()
        Catch
            RaiseEvent SocketError(Err.Description)
        End Try
End Sub
```

```
Private Sub SocketArrivoDatos(ByVal ar As IAsyncResult)
      …
      SyncLock _ObjetoBloqueo
      …
            Try
            …
            'Se realiza lo necesario según los valores recibidos
            Catch
                  RaiseEvent SocketError(Err.Description)
                  Exit Sub
            End Try
      …
      End SyncLock
End Sub
```

Tal como se puede apreciar en el extracto de código, el método “Conectar”, intenta establecer una conexión de tipo “Socket” según la familia de direcciones y el puerto que se requiere. Luego que el socket está conectado (método “SocketConectado”), se gatilla la ejecución del método que manejará el arribo de datos, en este caso “SocketArrivoDatos”, el cual realizará lo necesario según los datos que se reciban desde el servidor “irTrans”.

En este último método, cabe destacar la utilización de la instrucción “SyncLock”, la cual permite el bloqueo de una sección de código. Esto fue necesario, ya que la comunicación con el socket de “irTrans”, se realiza de forma asíncrona, lo cual quiere decir que el servidor “irTrans” podría enviar nuevos datos aunque el procesamiento de los datos anteriormente recibidos no hubiese sido totalmente completado. De esta forma, el software se encarga de poner en la cola de ejecución cada arribo de datos, y solo se produce la ejecución de la sección de código cuando se termina de procesar la anterior.

Finalmente, para el envío de información, mediante el socket, al servidor “irTrans”, se implementaron los siguientes métodos:

```
Private Sub EnvioDatos(ByVal Datos() As Byte)
        Try
            Dim ByteDatos As Byte() = Datos
            _Cliente.BeginSend(ByteDatos, 0, ByteDatos.Length, 0,
            AddressOf SocketEnvioTerminado, _Cliente)
        Catch
            RaiseEvent SocketError(Err.Description)
            Exit Sub
        End Try
End Sub
```

```
Private Sub SocketEnvioTerminado(ByVal ar As IAsyncResult)
        Try
            _Cliente = CType(ar.AsyncState, Socket)
            Dim BytesEnviados As Integer = _Cliente.EndSend(ar)
            RaiseEvent EnvioCompletado(BytesEnviados)
        Catch
            RaiseEvent SocketError(Err.Description)
            Exit Sub
        End Try
End Sub
```

Estos métodos, se encargan del envío de los datos al servidor “irTrans” y en caso de no producirse errores, gatillar el evento “EnvioCompleto”, el cual anuncia que el envío de datos fue finalizado correctamente.

Clase “LPT:

Esta clase, se encarga del manejo de los bytes que entran y son enviados mediante el puerto LPT, para el control del circuito Eléctrico / Mecánico. Para esto, se necesitó de una librería externa que permitiera el control del puerto paralelo a bajo nivel, además, esta librería no podía ser cualquiera, ya que la mayoría de las librerías que permiten este accionar, están desarrolladas directamente en assembler, y debido a que los sistemas operativos modernos de Microsoft, como Windows XP, no permiten la ejecución de este tipo de instrucciones. Por lo tanto se utilizó la librería “inpout32.dll”, la cual adquiere permisos de tipo “controlador de núcleo” para permitir la ejecución de instrucciones assembler y de esta forma lograr este tipo de control en el puerto.

De esta manera, los métodos que realizan el envío de bytes a los buses de control y de datos son los siguientes:

```
Public Sub EnviarByteRegDatos(ByVal Valor As Integer)
        If Valor < 0 Or Valor > 255 Then
            Exit Sub
        End If
        Out(_DireccionLPT, Valor)
        Me.Monitorear()
End Sub

Public Sub EnviarByteRegControl(ByVal Valor As Integer)
        If Valor < 0 Or Valor > 63 Then
            Exit Sub
        End If
        Out(_DireccionLPT + 2, Valor)
        Me.Monitorear()
End Sub
```

Como se puede apreciar, ambos métodos, utilizan la función “Out” implementada en la librería “inpout32.dll”, esta función solo requiere que se le pasen como parámetros la dirección del bus de la LPT al cual se le quiere enviar el valor, y el valor mismo que se desea enviar. En el caso del bus de datos, la dirección del bus, es la misma que la del mismo puerto, y en el caso del bus de control, es la dirección del puerto más dos.

Para la lectura de valores, se implementó un método de monitoreo del puerto como se muestra a continuación:

```
Private Sub Monitorear()
      …
      'Lee el registro (BUS) de datos
      _Registro = Inp(_DireccionLPT)
      …
      'Lee el registro (BUS) de estado
      _Registro = Inp(_DireccionLPT + 1)
      …
      'Lee el registro (BUS) de control
      _Registro = Inp(_DireccionLPT + 2)
      …
End Sub
```

Tal como se puede apreciar en el extracto, se deben realizar tres llamadas a la función “Inp”, implementada en la librería “inpout32.dll”, para obtener en cada una de ellas, los datos correspondientes a cada uno de los buses del puerto LPT. Cabe señalar que a cada lectura, se debe realizar el tratamiento adecuado al byte entregado, ya que el puerto LPT contiene diferentes bits dentro de los bytes que están implementados de forma inversa, es decir que presentan un cero cuando almacenan un uno lógico y viceversa.

Clase “SintRecoVoz”:

Esta clase, se encarga de manejar todo lo concerniente a la síntesis y reconocimiento de voz, a modo de interfaz para la aplicación. Para esto, se utilizó SAPI V4 (“Speech Aplication Program Interfaze”, o “Interfaz para Programas de Aplicación de Habla”, por sus siglas en ingles). Esta versión de SAPI se distribuye gratuitamente debido a que Microsoft se enfoca actualmente en su versión 5, la cual no fue posible utilizar debido a la carencia de soporte para el idioma español, lo cual era requerido para esta aplicación. La desventaja de utilizar SAPI V4, es que los componentes desarrollados para los programadores, así como su tecnología en si, están implementadas mediante el enfoque de ActiveX, el cual difiere del nuevo modelo de objetos presentado por Framework .NET. Favorablemente, es posible utilizar componentes antiguos mediante el servicio “Interop” incluido en .NET. Para esto, el Framework genera versiones .NET especiales de las librerías ActiveX, en este caso “AxInterop.ACTIVEVOICEPROJECTLib.dll” y “AxInterop.HSRLib.dll”, las cuales fueron incorporadas como referencias al proyecto, y se encargan de la síntesis y del reconocimiento de la voz respectivamente.

Luego de esto, ya es posible definir las variables necesarias, de la siguiente manera:

```
Private _SintVozSistema As AxACTIVEVOICEPROJECTLib.AxDirectSS
Private _RecoVozSistema As AxHSRLib.AxVcommand
```

Luego de esto, ya es posible implementar el método que realizará la síntesis de palabras habladas de la siguiente manera:

```
Public Sub Decir(ByVal Frase As String)
        If _HabilitarSintesisVoz = True Then
            Try
                _SintVozSistema.CurrentMode = _GeneroVoz
                _SintVozSistema.Speed = _VelocidadVoz
                _SintVozSistema.Speak(Frase)
            Catch Ex As Exception
                  …
            End Try
        End If
End Sub
```

La implementación del método que realizaría el reconocimiento de comandos, se realizó en el manejador del evento “CommandRecognizeEvent”, tal como lo muestra el siguiente extracto de código.

```
Private Sub RecoVozSistema_CommandRecognize(ByVal sender As Object,
ByVal e As AxHSRLib._VcommandEvents_CommandRecognizeEvent)
      …
      If UCase(e.command) =
      TablaComandoSistema.Rows(0).Item(_NomColIdSistema) Then
      _TimerReco.Enabled = False
      _SwEstadoReco = 1 'Listo a reconocer el Dispositivo
      …
End Sub
```

Este evento, recibe como parámetro el comando reconocido en forma de un texto tal como se puede apreciar en la instrucción “e.command”, para luego realizar las operaciones requeridas, según el comando reconocido.

Tal como se mencionó, este es solo un extracto del código implementado en el evento de reconocimiento de comandos, ya que se desarrolló un algoritmo que define una sintaxis de reconocimiento que sigue la siguiente forma:

Sistema → Dispositivo → Acción

Entonces, el primer comando que se debe reconocer es el del propio nombre del sistema o computadora en la cual se esta ejecutando la aplicación. Esto sirve para personificar al sistema y que este solo atienda cuando se estén refiriendo a él para darle una orden. La orden en particular dada, esta definida por los comandos “Dispositivo” y “Acción” hablados por el usuario. Además, entran en juego una serie de temporizadores, que permiten el reconocimiento del comando completo durante un periodo de tiempo configurable, lo cual posibilita que el sistema no tome palabras pronunciadas en distintos espacios de tiempo, y que por ende ejecuten acciones no deseadas.

Tanto los dispositivos como sus acciones asociadas, son definidos por el usuario según sus necesidades, y almacenadas en la base de datos en las tablas correspondientes.

Clase “API”:

Esta clase se encarga de ejecutar todo lo que tenga relación con la API misma de Windows. Para poder ejecutar las funciones implementadas en las diferentes librerías del sistema operativo Windows, se deben realizar las definiciones correspondientes, de la siguiente manera:

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias
"ShellExecuteA" (ByVal hwnd As Integer, ByVal lpOperation As String,
ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory
As String, ByVal nShowCmd As Integer) As Integer
```

Para luego ya poder utilizar las funciones declaradas de la siguiente manera:

```
Public Sub EjecutaPrograma(ByVal Manejador As Form, ByVal Programa As
String, Optional ByVal Parametros As String = "", Optional ByVal
DirectorioTrabajo As String = "", Optional ByVal ModoVentanaPrograma As
ModoVentana = API.ModoVentana.SW_SHOWNORMAL)

        Me.ShellExecute(Manejador.Handle.ToInt32, "Open", Programa,
        Parametros, DirectorioTrabajo, ModoVentanaPrograma)
End Sub
```

Clase “VideoVigilancia”:

Esta clase se implementó con el propósito de poder desplegar en una ventana de la aplicación, una fuente de video conectada al sistema mediante ya sea una tarjeta de captura o una cámara USB, y de esta forma, permitir que el

usuario pueda realizar un monitoreo de estas fuentes de video desde un lugar centralizado.

Este tipo de codificación no es trivial y requirió de innumerables llamadas a la API de Windows y específicamente a los manejadores de VFW (“Video For Windows”, o “Video Para Windows”, por sus siglas en ingles).

Un extracto del código donde se puede apreciar este tipo de llamadas es el siguiente:

```
Public Function Ver() As Boolean
        Try
            _Manejador = Nothing
            _Manejador = New Form

            …
            Me.ObtieneDescripcionDrivers()
            _lwndC = Nothing
            _lwndC = Me.CreaVentanaCaptura
            _Manejador.FormBorderStyle = FormBorderStyle.Sizable
            _Manejador.Height = 240
            _Manejador.Width = 320
            _Manejador.ControlBox = True
            _Manejador.Text = "Camara"
            _Manejador.Show()
            …
            'Conecta la ventana de captura con el controlador
            _ConexionControladorVentana = Me.capDriverConnect(_lwndC,
            0)
            …
            Me.capPreviewScale(_lwndC, True)
            'Configura la velocidad de la vista previa ("Preview Rate")
            Me.capPreviewRate(_lwndC, _FPS)
            Comienza la visualización de la imagen desde la fuente de
            video
            Me.capPreview(_lwndC, True)
            …
End Function
```

Esta función muestra como se inicializa el controlador de video, para luego ser asociado al puntero de una nueva ventana, luego de esto se puede definir la velocidad del muestreo de video y la iniciación de la imagen.

Clase “Dispositivos”:

Esta clase es la encargada de administrar todos los dispositivos y acciones definidos por el usuario, pero además, también se encarga de ejecutar los métodos internos de la aplicación que el usuario a asociado a cada acción definida. Por ejemplo: Un usuario, puede definir el dispositivo “Televisor” y una acción “Encender” para este. Por si solos, este dispositivo y acción, no representan nada más que algo tangible para el usuario, pero la aplicación sigue sin saber que debe hacer cuando se seleccione. Por lo tanto, la aplicación expone una serie de clases y métodos para que el usuario los asocie a las acciones que él define, por lo tanto la acción “Encender” para el dispositivo “Televisor”, podría hacer referencia a la clase del sistema “IR” y a su método “EnviarSeñal(<Alguna Señal Infrarroja>)”.

De esta forma el sistema se convierte en una aplicación altamente configurable y adaptable a las distintos requerimientos de las personas que lo utilizan.

Estas asociaciones, se almacenan en la base de datos en forma de texto, es decir, existe la tabla “Ejecutar”, la cual almacena la información

correspondiente a que la acción “Encender” del dispositivo “Televisor”, debe ejecutar la clase “IR” y su método “EnviarSeñal(<Alguna Señal Infrarroja>), y por lo tanto la clase “Dispositivos” debe poder ejecutar este método contenido en un string .

Para realizar esto, se podría haber implementado un bloque de tipo “Select Case”, el cual detectaría que método es el que contiene el string para luego invocarlo, pero esto habría hecho que la aplicación careciera de la flexibilidad necesaria. Por lo tanto, se recurrió al empleo de “Reflection”, el cual es provisto por .NET, y permite tanto la utilización de una clase, así como la ejecución de uno de sus métodos en particular mediante los nombres de la clase y el método contenidos en cadenas de texto, tal como se muestra en el siguiente extracto de código:

```
Public Sub Ejecutar(ByVal IdDispositivo As String, ByVal IdAccion As
String)
      'Se obtiene el tipo de la clase desde el "Assembly" de la
      aplicacion
      ClaseObj =
      InfoDomosoft.EncuentraTipoDeObjeto(CType(_SDtblEjecutar.Tables(_N
      omTblEjecutar).Rows(I).Item(_NomColIdClaseObjTblEjecutarDirecto),
      String))
      …
      'Se obtiene el miembro de la clase desde el "Assembly"
      'de la aplicacion
      Miembro = InfoDomosoft.EncuentraMiembro(ClaseObj,
      CType(_SDtblEjecutar.Tables(_NomTblEjecutar).Rows(I).Item(_NomCol
      IdMetodoTblEjecutarDirecto), String))
      …
      ClaseObj.InvokeMember(Miembro.Name, Banderas, Nothing, Nothing,
      Parametros)
      …
End Sub
```

De esta forma “Reflection”, permite que se importe el “Assembly” mismo de la aplicación, para definir un tipo de objeto de acuerdo al nombre entregado en una cadena de texto, esto también se realiza para el método en cuestión, y luego de que se tienen ambos, se puede definir un arreglo de objetos, el cual contendrá los parámetros requeridos por el método. Finalmente el método “InvokeMember” propio de cada clase implementada por el programador o existente en el lenguaje, permite que se ejecute uno de sus métodos miembros mediante un texto que contenga su nombre. Las banderas entregadas como parámetros, dependerán del tipo de método al que se este haciendo referencia (estático, de instancia, etc.).

El código necesario para encontrar el tipo de objeto y el método dentro del “Assembly” del ejecutable del sistema a partir de un texto, se encuentra documentado dentro de la misma clase “InfoDomosoft”.

Este tipo de codificación, permite que la aplicación sea altamente expandible y adaptable a nuevos requerimientos, mediante la implementación y exposición de nuevos tipos de objetos y métodos que cumplan con las nuevas necesidades.

Clase “ServSincIRBDSAD”:

Tal como se mencionó anteriormente, el manejo de las señales infrarrojas es administrado por el servidor de dispositivos “irTrans”, este servicio de nombre “irServer”, es incluido con el dispositivo “irTrans” desarrollado por Marcus Müller. Lamentablemente el servicio “irServer”, maneja la información de “Controles Remotos” y sus señales infrarrojas en archivos planos de texto (uno por cada control remoto), por lo cual se requirió de la creación un servicio adicional, que monitorizase el directorio en donde se encuentran estos archivos y de esta forma, cada vez que se creara o manipulara uno de estos archivos, realizará las acciones necesarias para sincronizar los datos con la BD.

Anteriormente, desarrollar servicios para plataformas Windows NT y posterior, era por lo general una tarea ardua, que requería codificación no estándar en lenguajes rígidos como “C”, e incluso con herramientas de desarrollo específicas para este propósito. En la actualidad, esta labor se ha simplificado en gran medida, ya que .NET proporciona un tipo especial de proyecto en el cual se pueden implementar este tipo de servicios. Por lo tanto se creó en un proyecto separado la clase “ServSincIRBDSAD”, la cual se encarga de las actualizaciones necesarias para mantener los datos de los archivos y la base de datos correctamente sincronizados.

A continuación, se presenta un extracto del código que realiza esta función:

```
Protected Overrides Sub OnStart(ByVal args() As String)
      …
      _RutaServicio = System.Windows.Forms.Application.ExecutablePath
      _RutaServicio = _RutaServicio.Substring(0,   Len(_RutaServicio) -
      _LargoNomServicio)
      …
      'Segundo argumento es el directorio contenedor de los
      'archivos de "Remotos" de servidor IR que se debe monitorear
      …
      IniciarMonitoreo(Configuration.ConfigurationSettings.AppSettings(
      "Ruta").ToString)
      …
End Sub
```

El método “OnStart”, es ejecutado cuando se inicia el servicio, y luego de establecer algunas variables de configuración, llama al método “IniciarMonitoreo”, el cual mediante la librería del sistema “System.IO.FileSystemWatcher”, monitorear cualquier cambio en el directorio que se esta vigilando (creación, eliminación o modificación de archivos).

Luego que el proyecto para el servicio es compilado, solo se requiere instalarlo con el siguiente comando:

C:> InstallUtil "C:\Ruta\Al\Ejecutable\Del\Servicio\ServSincIRBDSAD.exe"

Puesto a que en .NET todo es una clase, las interfases graficas de usuario fueron implementadas en distintas clases que exponen los diferentes datos de cada una de las demás clases funcionales del sistema. Para mostrar esto de mejor forma, a continuación se presentan una serie de capturas de

pantalla funcionales de la aplicación, en las cuales, se muestran sus principales características.

Figura Nº 20: Ventana configuración dispositivos irTrans conectados al sistema.

En la pantalla de la figura Nº 20, se pueden configurar los modos de envío y recepción de las señales infrarrojas para los dispositivos irTrans conectados al sistema.

Figura Nº 21: Ventana recepción señales infrarrojas.

En esta captura de pantalla de la aplicación (Figura Nº 21), se puede apreciar la recepción codificada de una señal infrarroja enviada por el control remoto original del dispositivo así como también el nombre del dispositivo y la acción a la que corresponde.

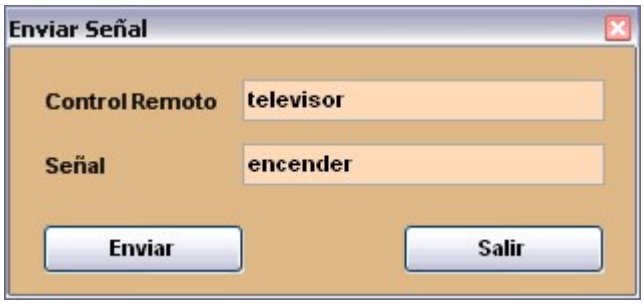

Figura Nº 22: Ventana para envío de una señal infrarroja previamente almacenada.

Esta ventana (Figura Nº 22), permite el envío de una señal infrarroja previamente almacenada, permitiendo el control de dispositivo que respondan a este tipo de señales.

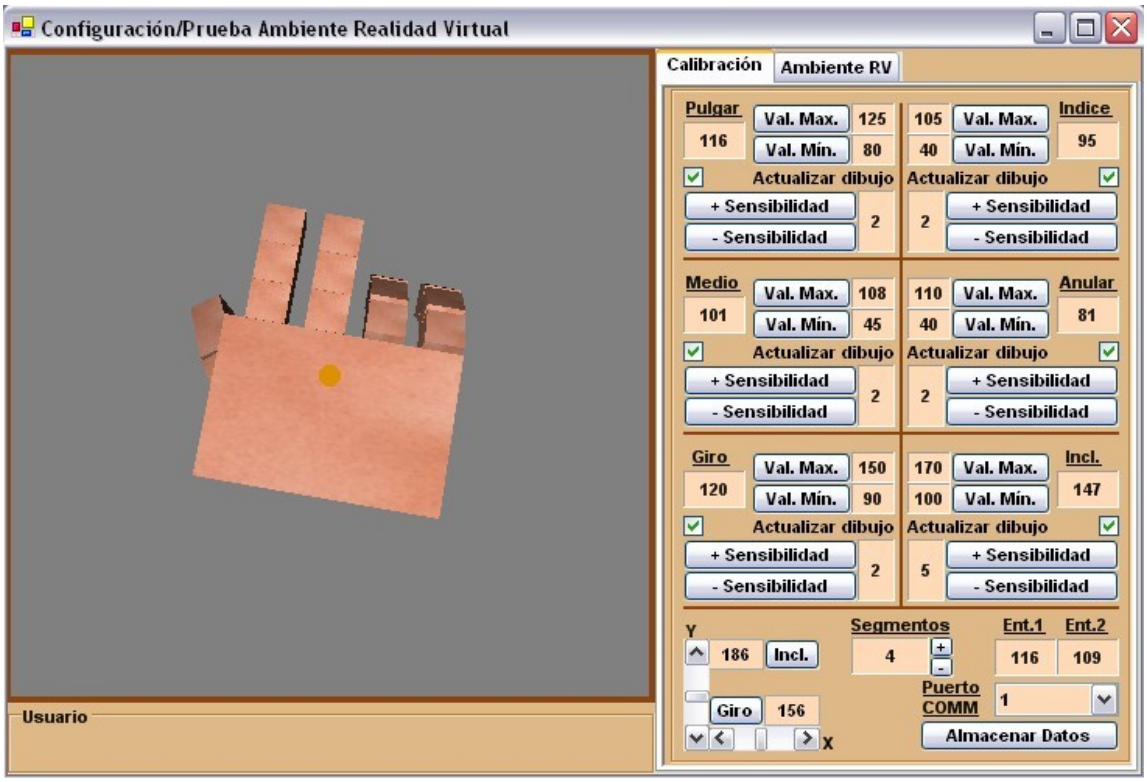

Figura Nº 23: Ventana de calibración y pruebas del guante de realidad virtual NewGlove.

Esta ventana (Figura Nº 23), permite la calibración y pruebas de la interfaz de realidad virtual NewGlove para que se ajuste a las necesidades particulares del usuario. Ademas muestra los datos provenientes del guante en tiempo real y permite la asignación de estos valores a la configuración del usuario actual.

Figura Nº 24: Ventana para calibración y pruebas del ambiente virtual.

En esta ventana (Figura Nº 24), se puede configurar y probar el ambiente virtual en conjunto con el guante de realidad virtual. El ambiente cuenta con dos “tarjeteros” que representan dispositivos y las acciones asociadas a estos, el tarjetero vertical representa a los dispositivos y el horizontal a las acciones. Además, en esta ventana se pueden asignar los patrones de movimiento (flexión/extensión del índice u otro dedo, inclinación o giro de la muñeca) a una

acción especifica en el ambiente (giros en uno u otro sentido de los tarjeteros), es decir, por ejemplo: la inclinación de la muñeca puede asignarse al movimiento hacia abajo del tarjetero vertical, para que cuando el usuario ejecute este patrón de movimiento, el ambiente responda con la acción deseada y de esta forma pueda escogerse un dispositivo y una acción de entre las ingresadas al sistema (Puerta→Abrir, Puerta→Cerrar, Televisor→Encender, Winamp→Cargar, etc.). Cualquier patrón de movimientos en cualquier rango y con un porcentaje de error configurable puede ser asignado a cualquier acción dentro del ambiente, adaptándose, de esta forma, de mejor manera a las necesidades particulares del usuario. Además, también se pueden asignar acciones del mouse o teclas para que ejecuten las mismas acciones dentro del ambiente. Cuando el usuario configura un patrón de movimiento para el ambiente, las barras de colores le indican el porcentaje de movimiento que deberá efectuar posteriormente para que el patrón sea reconocido adecuadamente, incluso el usuario puede utilizar el mismo patrón de movimiento, por ejemplo: flexión del índice, un número distinto de veces para ejecutar distintas acciones dentro del ambiente, por ejemplo: flexionar una vez el dedo índice, puede mover el tarjetero vertical hacia abajo, y flexionar el mismo dedo índice dos veces para mover el tarjetero vertical hacia arriba, con lo cual con un número limitado de movimientos, se pueden ejecutar todas las acciones necesarias para el control total del ambiente virtual.

Figura Nº 25: Ventana configuración y pruebas de la interfaz de reconocimiento y síntesis de voz.

En esta ventana (Figura Nº 25) se puede configurar, probar y utilizar la interfaz de reconocimiento y síntesis de voz del sistema, el personaje que se muestra toma el nombre de la máquina como propio (en este caso “HAL”) para solo responder a este. Esto se debe a que el algoritmo de reconocimiento de voz implementado toma la siguiente sintaxis:

Sistema → Dispositivo → Acciones → Parámetros (opcional)

Además, se puede especificar el sexo de la voz que se desea utilizar para la síntesis de voz, así como la velocidad de ésta, etc.

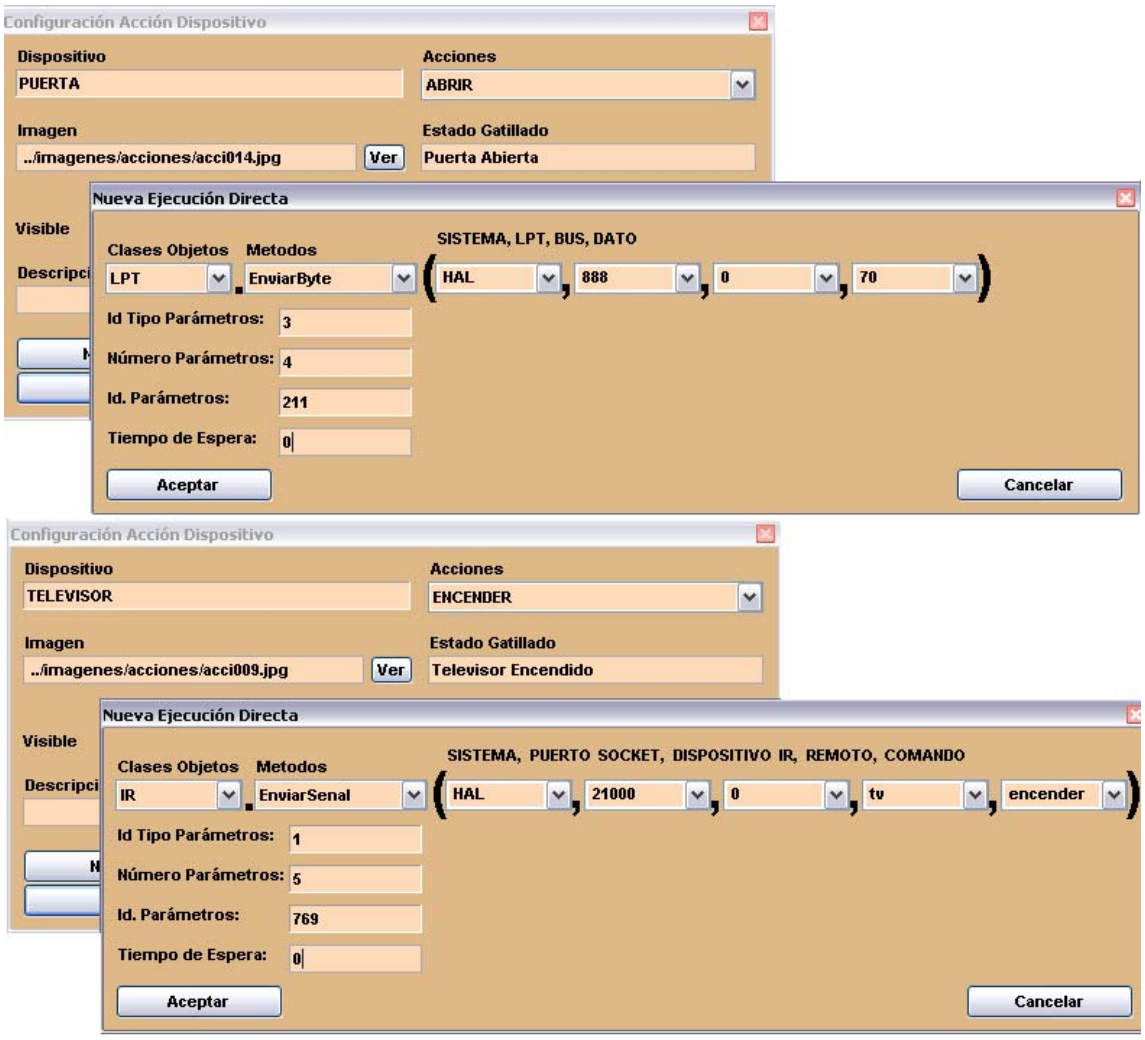

Figura Nº 26: Ventana asociación acciones a clases y métodos expuestos.

En las ventanas que se muestran en la figura Nº 26, se puede apreciar como dos acciones definidas por el usuario para dos dispositivos diferentes, son asociadas a métodos expuestos por el sistema. En el primer caso, la acción

“Abrir”, del dispositivo “Puerta”, se está asociando con el método “EnviarByte”, de la clase “LPT”. En el segundo par de ventanas, podemos ver como la acción “Encender”, perteneciente al dispositivo definido por el usuario “Televisor”, se asocia con el método “EnviarSenal”, de la clase “IR”.

Todas las acciones definidas por el usuario o el administrador, pueden estar asociadas a una o más de estas acciones “directas”, pero tambien pueden asociarse a otras acciones pertenecientes a otros dispositivos, tal como se muestra a continuación en la figura Nº 27.

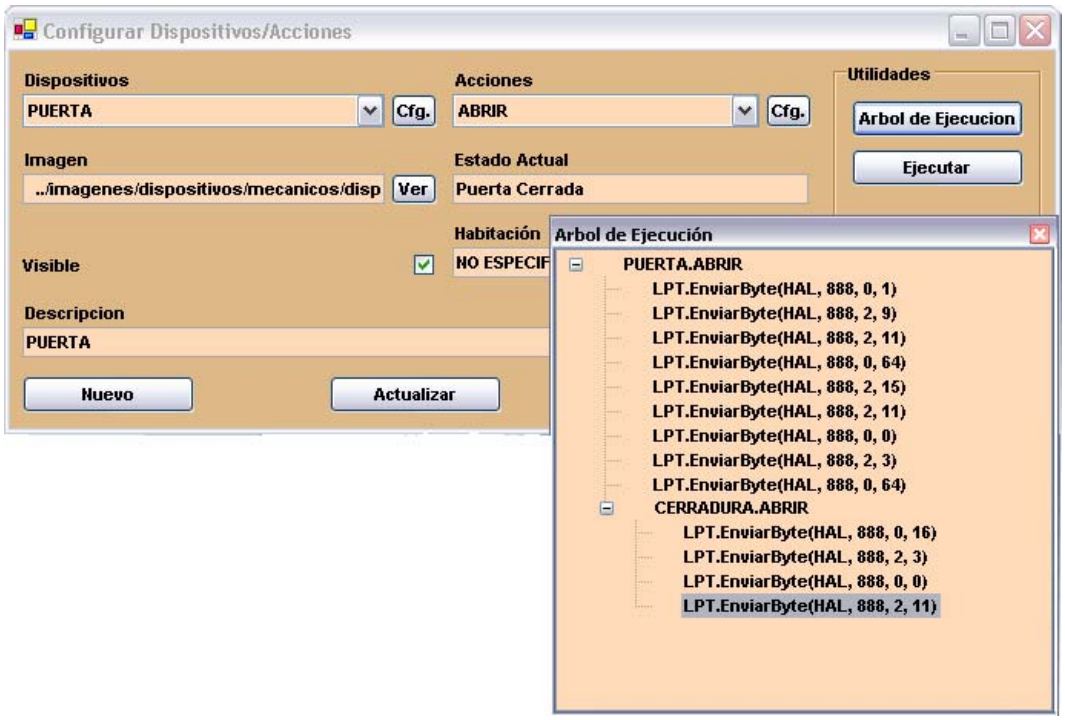

Figura Nº 27: Ventana árbol de ejecución acción dispositivo.

Tal como se mencionó, en la figura Nº 27, se muestra como la acción “Abrir” del dispositivo “Puerta”, ejecuta “directamente” varias veces el metodo expuesto “EnviarByte” para controla el motor montado en la puerta a modelo a través del circuito Eléctrico / Mecánico, pero tambien de forma “indirecta”, controla el funcionamiento de la cerradura electrica, al agrupar la acción “Abrir”, del dispositivo “Cerradura”.

El sistema no pone restricciones respecto al número o tipo de asociaciones que se pueden hacer para una acción en particular, pudiendo ser estas de cualquier tipo y pertenecientes a cualquier dispositivo previamente definido.

Si bien las interfaces gráficas presentadas en las figuras Nº 26 y 27 pueden no ser las más amigables, se debe tener en consideración que se desarrollaron según las necesidades de la investigación y pensando en el usuario experto o administrador, quienes seíian los que habitualmente realizarían este tipo de configuraciones, considerando también, que el usuario tradicional, interactuará con los dispositivos mediante el ambiente virtual o el sistema de reconocimiento de voz. Sin embargo, no es descartable la idea de desarrollar o modificar a futuro nuevas interfaces para manejar los dispositivos, permitiendo por ejemplo agregar dispositivos mediante el ambiente virtual o comandos verbales, así como tambien nuevas interfaces en donde el usuario defina esquemas gráficos de control, etc.

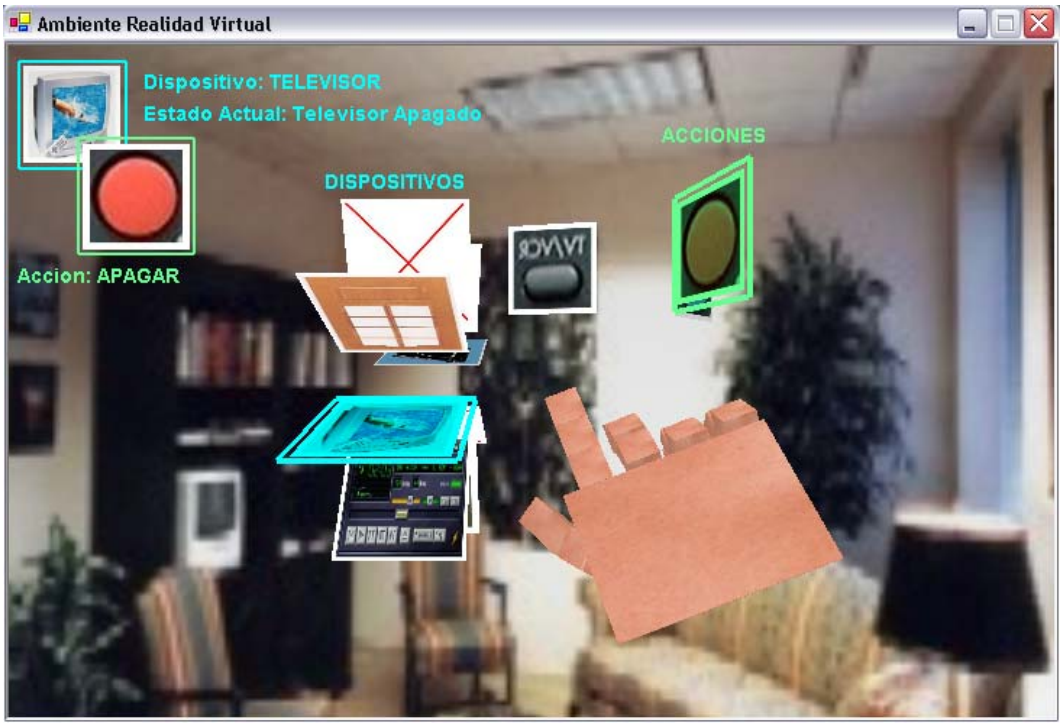

Figura Nº 28: Ventana control mediante ambiente virtual.

En la ventana de la figura Nº 28, podemos observar el ambiente virtual de control, a diferencia del ambiente de pruebas y configuración presentado en la figura Nº 24, en este se puede ordenar al sistema que efectivamente ejecute cierta acción, asi como también monitorear o modificar el estado actual de los diferentes dispositivos. Este ambiente puede ser controlado mediante los patrones de movimientos de la mano del usuario previamente configurados, así como también por las teclas y acciones del ratón que se hallan definido para estos efectos.

Figura Nº 29: Ventana control mediante reconocimiento de voz.

En la figura Nº 29, se muetra el panel de control por voz, que a diferencia del presentado para configuración y pruebas, este puede ejecutar las acciones dictadas verbalmente por el usuario, siguiendo el mismo patron sintáctico antes explicado (Sistema→Dispositivo→Acción).

Figura Nº 30: Ventana para monitoreo y control de fuentes de video.

En esta ventana (Figura Nº 30), se puede realizar un monitoreo de las fuentes de vídeo conectadas al sistema (WebCams, Tarjetas de Captura con entradas de video, Sintonizadores de Televisión, etc.). El usuario puede cambiar la fuente de video o tener varias de estas abiertas simultáneamente, configurando también el formato y tamaño de visualización, además, el usuario puede tomar una captura estática desde la fuente de video hacia el portapapeles del sistema, para luego, mediante un programa de edición de imágenes (Photo Editor, etc.), almacenarla para su posterior análisis.

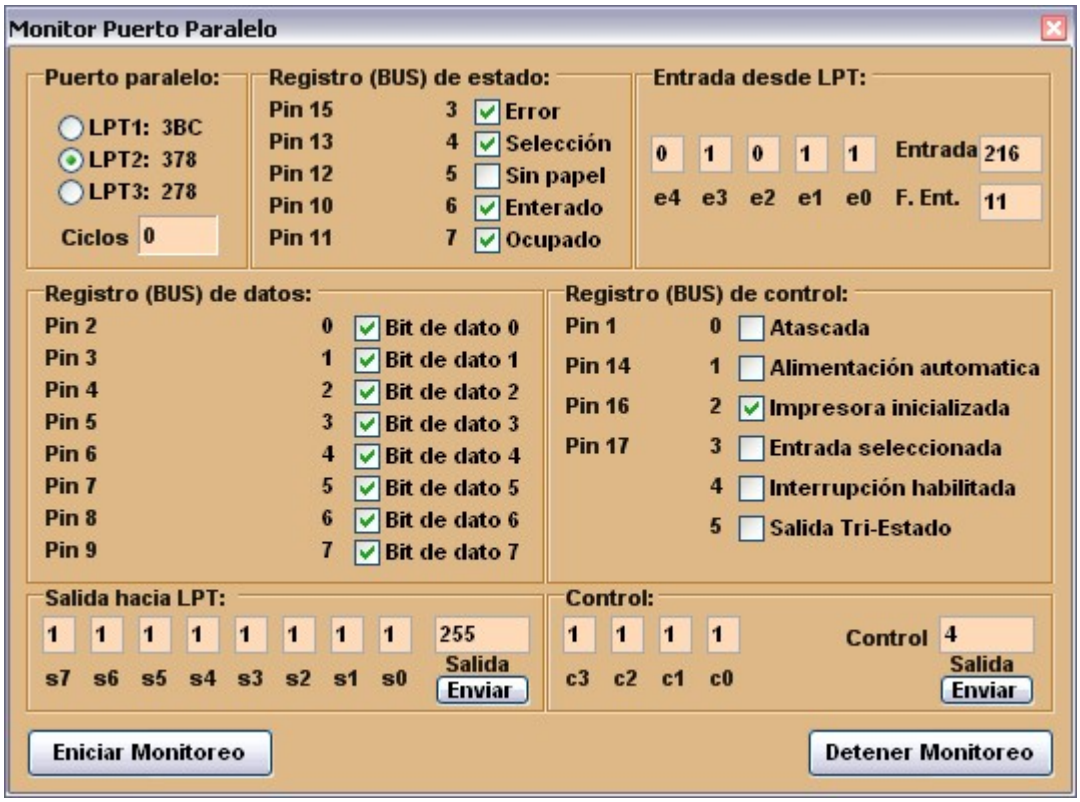

Figura Nº 31: Ventana para control y monitoreo del puerto LPT.

En esta ventana (Figura Nº 31), se puede acceder a nivel de bit al puerto paralelo de la computadora, para de esta forma, probar y enviar los pulsos TTL de control para el circuito Eléctrico-Mecánico, así como también monitorear el bus de estado, en el cual este circuito devuelve información de retroalimentación al sistema.

Antes de finalizar con la sección de “Implementación”, es importante destacar que al ser la aplicación un software 100% orientado a objetos y la base de datos corresponder a un modelo relacional normal, se presentan problemas a la hora de asociar clases con tablas. Esto se debe a que ambas metodologías se abocan a paradigmas distintos, esto es:

“Orientación a Objetos”: Paradigma de modelado de objetos reales y tangibles en código de programación.

“Modelo Relacional”: Paradigma matemático de teoría de conjuntos, que pretende una correcta e eficiente manera de administrar la información.

Esto es un problema conocido y documentado, que se conoce como “Object-Relational Impedance Mistmatch”, lo que se puede traducir como “Desigualdad de Impedancia entre Relacional y Objetos”. Esto se produce principalmente porque los modelos relacionales tradicionales, no permiten el almacenamiento y persistencia de objetos residentes en memoria dentro de sus tablas. Esto podría evitarse utilizando una base de datos orientada a objetos, pero lamentablemente estas aún se encuentran en etapas de desarrollo, en donde los estándares necesarios para su correcta implementación difieren entre los distintos fabricantes.

Sin embargo, una forma de lidiar con este problema, es el de utilizar una técnica conocida como “mapeo de objetos a tablas”, en la cual cada clase encapsula la ejecución de los distintos procedimientos almacenados o funciones de la base de datos para mantener los datos de los objetos, y de igual manera, sincronizar los datos de las tablas correspondientes con las propiedades y miembros de la clase. Lamentablemente esta solución con frecuencia provoca que mucho del código escrito solo es necesario para mantener el mapeo correspondiente entre la clase y las una o más tablas que deben almacenar los datos para cada una de estas clases en particular.

Finalmente, si bien siempre es posible mejorar la eficiencia y rendimiento del código de un programa, para esta aplicación, se completaron mas de 20.000 líneas de código, escritas por un solo programador, en poco más de un año de desarrollo. Es difícil plasmar en tan solo un documento todo cuanto se tuvo que implementar, los distintos algoritmos para resolver una serie de problemáticas, el código de las interfases graficas que muestran los datos de configuración de los usuarios, el código de mapeo necesario para las relaciones entre clases y tablas, etc.

Por lo tanto, lo que se ha mostrado aquí, es tan solo un extracto de todo lo que se desarrolló, dejando gran parte de esto documentado en el mismo código de implementación de las clases, en donde se podrá comprender mejor, puesto a que se encuentra enmarcado en el mismo contexto en donde se a desarrollando.

8.4 Pruebas

Debido a la naturaleza iterativa de la metodología R.U.P, es que cada una de estas iteraciones, produce una versión del software o “Producto” que completa, de forma incremental, los casos de uso detectados en cada una de las fases. Por esto, a cada iteración de la aplicación le corresponden pruebas funcionales como parte de los hitos propuestos en la planificación.

Las capturas de pantalla incluidas en la sección “Implementación”, del software, muestran, además, a la aplicación funcionando en su estado actual, en conjunto con los dispositivos con los cuales interactúa (irTrans, NewGlove, Síntesis y Reconocimiento de Voz, Acciones y Dispositivos, etc.).

A continuación, se presentan algunas imágenes de la aplicación y el sistema funcionando en su ambiente de pruebas.

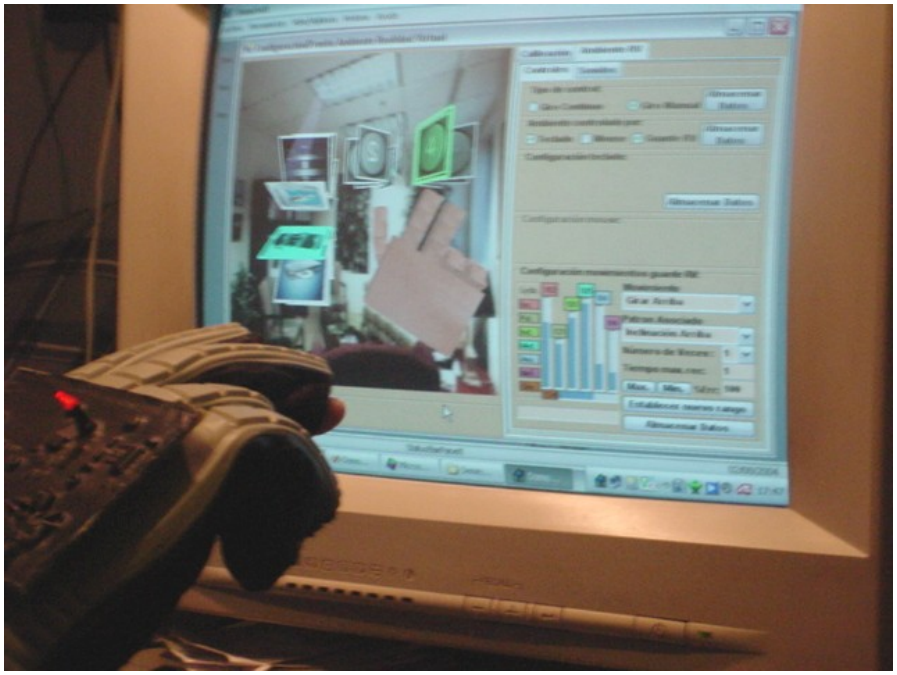

Figura Nº 32: Usuario interactuando con el ambiente virtual mediante interfaz de realidad virtual guante NewGlove.

En la figura anterior (Figura Nº 32), se presenta a un usuario interactuando con el ambiente virtual mediante la interfaz guante NewGlove.

Figura Nº 33: Usuario interactuando con el sistema de reconocimiento de voz.

En esta imagen (Figura Nº 33), se muestra a un usuario interactuando con el sistema de reconocimiento y síntesis de voz.

Figura Nº 34: Envío y recepción de señal infrarroja.

En esta imagen (Figura Nº 34), podemos ver en funcionamiento el sistema de envío y recepción de una señal infrarroja, la señal es enviada por el control remoto de un dispositivo, en este caso el de un televisor (imagen izquierda), es recibida por el transreceptor irTrans y comunicada al software para su almacenamiento y uso posterior (imagen derecha).

Es posible visualizar la luz infrarroja proveniente del control remoto debido a que se utilizó una cámara digital para tomar la imagen.

Figura Nº 35: Envío de señal IR para control de dispositivo.

En esta imagen (Figura Nº 35), podemos ver el envío de una señal IR seleccionada en la aplicación y enviada mediante el transreceptor irTrans (imagen izquierda), provocando que el dispositivo, en este caso un televisor (imagen derecha), al cual corresponda la señal, responda, ejecutando la acción correspondiente al comando seleccionado.

En esta imagen, también podemos apreciar la potencia de salida de la señal IR emitida por el dispositivo irTrans.

La accion que se efectuó en la figura anterior (Figura Nº 35), puede ser ejecutada de igual forma mediante la interfaz de realidad virtual y el ambiente virtual, o a través de la interfaz de reconocimiento de voz.

Figura Nº 36: Pruebas de software y componentes mecánicos.

En la figura Nº 36, se muestra como el software controla los movimientos de apertura y cierre de la puerta modelo empleada en la fase de pruebas de esta investigación.

9. Base de Datos

La implementación y documentación que a continuación se presenta, está basada en la metodología propuesta por Connolly para bases de datos.

NOTA:

Para una mejor visualización de los esquemas aquí presentados, en los anexos corresponientes se incluyen los diagramas desarrollados en “PowerDesigner”.

9.1 Modelo Conceptual BDSAD (“Base de Datos Sistema de Asistencia Domótica”)

Para el modelo conceptual de esta base de datos, solo se necesitó una vista de usuario, ya que al enmarcarse dentro de una investigación, todos los aspectos relativos al modelo de datos se desarrollaron de forma conjunta.

9.1.1 Entidades

Del análisis de los casos de uso, se desprendieron las siguientes entidades para el modelo conceptual de la base de datos:

NOMBRE DE ENTIDAD : Sistema
DESCRIPCIÓN : Almacena la información sobre los sistemas o computadoras en donde se esta ejecutando el software.
ALIASES : SISTEMA
OCURRENCIA
- Un sistema normalmente puede tener hasta 3 puertos LPT, aunque es posible expandirlos a más.
- Un sistema puede tener varios servidores de dispositivos IR.
- Un sistema puede tener muchos usuarios validos registrados.
- Un sistema puede exponer muchas clases de objetos.
- Un sistema puede conectar hasta 4 guantes RV a sus puertos seriales, aunque es posible expandirlos a más.

NOMBRE DE ENTIDAD : Usuarios
DESCRIPCIÓN : Almacena información sobre los usuarios del sistema.
ALIASES : USUARIOS
OCURRENCIA
- Un usuario puede o no tener configuración para el sintetizador y reconocedor de voz.
- Un usuario puede o no tener configuración para el ambiente de realidad virtual.
- Un usuario puede calibrar cero o más guantes RV en algún sistema valido.
- Un usuario puede definir uno o más dispositivos en para algún sistema en particular.

NOMBRE DE ENTIDAD : GuanteRV
DESCRIPCIÓN : Almacena los datos para los guantes de realidad virtual “NewGlove” conectados al sistema.
ALIASES : GUANTERV
OCURRENCIA
- Un guante de realidad virtual puede realizar todos o algunos de los patrones de movimientos disponibles.
- Un guante RV puede calibrar o no algunos de sus patrones para ser utilizados por un usuario del sistema.

NOMBRE DE ENTIDAD : TipoPatronesGuanteRV
DESCRIPCIÓN : Tipos de patrones de movimiento realizable por un guante de realidad virtual “NewGlove”.
ALIASES : TIPOPATRONESGUANTERV
OCURRENCIA
- Un guante de realidad virtual puede realizar algunos o ninguno de los tipos de patrones de movimiento.

NOMBRE DE ENTIDAD : TipoMovimientos
DESCRIPCIÓN : Tipos de movimientos realizados por el ambiente de realidad virtual.
ALIASES : TIPOMOVIMIENTOS
OCURRENCIA
- Un ambiente de realidad virtual pude tener muchos tipos de movimientos.

NOMBRE DE ENTIDAD : TipoControl
DESCRIPCIÓN : Tipos de control sobre el ambiente virtual; Giro manual o giro automático.
ALIASES : TIPOCONTROL
OCURRENCIA
- El ambiente de realidad virtual puede tener un solo tipo de control a la vez.
- Muchos tipos de movimientos del ambiente virtual pueden corresponder a un tipo de control.

NOMBRE DE ENTIDAD : AmbienteRV
DESCRIPCIÓN : Almacena los valores de configuración del usuario para el ambiente de realidad virtual.
ALIASES : AMBIENTERV
OCURRENCIA
- Un usuario puede o no tener configuración para el ambiente de realidad virtual.
- Un ambiente de realidad virtual puede tener o no teclas que controlan los movimientos del ambiente, para un usuario en particular.
- Un ambiente de realidad virtual puede tener o no acciones del Mouse que controlan los movimientos del ambiente, para un usuario en particular.
- Un ambiente de realidad virtual puede tener o no patrones de movimientos del guante RV, que controlan los movimientos del ambiente, para un usuario en particular.

NOMBRE DE ENTIDAD : Mouse
DESCRIPCIÓN : Almacena la información sobre las acciones que puede realizar el Mouse para controlar el ambiente de realidad virtual.
ALIASES : MOUSE
OCURRENCIA - Las acciones del Mouse pueden ejecutar movimientos en uno o más ambientes virtuales.

NOMBRE DE ENTIDAD : Teclado
DESCRIPCIÓN : Almacena la información sobre las teclas que pueden ser utilizadas para generar movimientos en el ambiente virtual.
ALIASES : TECLADO
OCURRENCIA
- Las acciones del teclado pueden ejecutar movimientos en uno o más ambientes virtuales.

NOMBRE DE ENTIDAD : SintRecoVoz
DESCRIPCIÓN : Almacena la información de configuración del sintetizador y el reconocedor de voz para un usuario en particular.
ALIASES : SINTRECOVOZ
OCURRENCIA
- Un usuario puede o no tener una configuración para el sintetizador y reconocedor de voz.

NOMBRE DE ENTIDAD : ServDispositivosIR
DESCRIPCIÓN : Almacena información sobre los servidores “irTrans” conectados al sistema.
ALIASES : SERVDISPOSITIVOSIR
OCURRENCIA
- Un servidor para dispositivos IR, “irTrans”, puede tener entre 0 y 15 dispositivos “irTrans” conectados.

NOMBRE DE ENTIDAD : DispositivosIR
DESCRIPCIÓN : Almacena los datos para los dispositivos de manejo de señales infrarrojas (IR) conectados al bus "irTrans" en el sistema.
ALIASES : DISPOSITIVOSIR
OCURRENCIA
- Un dispositivo IR, "irTrans", puede tener definidos muchos controles remotos.

NOMBRE DE ENTIDAD : Comandos
DESCRIPCIÓN : Almacena los "Comandos" infrarrojos manejados por el servidor de dispositivos ir "irTrans"
ALIASES : COMANDOS
OCURRENCIA
- Un control remoto, puede definir muchos comandos.

NOMBRE DE ENTIDAD : Remotos
DESCRIPCIÓN : Almacena la información sobre los "Remotos" (controles remotos de dispositivos con mandos IR) definidos por el usuario y manejados por el servidor "irTrans".
ALIASES : REMOTOS
OCURRENCIA
- Un dispositivo IR, "irTrans", puede tener definidos muchos controles remotos.

NOMBRE DE ENTIDAD : Habitaciones
DESCRIPCIÓN : Almacena los datos sobre habitaciones.
ALIASES : HABITACIONES
OCURRENCIA
- En una habitación, pueden encontrarse varios dispositivos definidos por el usuario.
- En una habitación pueden encontrarse muchos dispositivos IR.

NOMBRE DE ENTIDAD : Ejecutables
DESCRIPCIÓN : Almacena los datos para los programas "Ejecutables" que puede cargar y controlar el sistema.
ALIASES : EJECUTABLES
OCURRENCIA - Un sistema puede cargar y manejar muchos programas ejecutables.

NOMBRE DE ENTIDAD : Parametros
DESCRIPCIÓN : Almacena los identificadores generalizados para los distintos parámetros empleados por los métodos.
ALIASES : PARÁMETROS
OCURRENCIA - Una combinación de parámetros pueden ser utilizados por muchos métodos.

NOMBRE DE ENTIDAD : TipoParametros
DESCRIPCIÓN : Almacena los identificadores generalizados para los distintos parámetros empleados por los métodos.
ALIASES : TIPOPARÁMETROS
OCURRENCIA
- Un tipo de parámetros puede estar asociado a muchos métodos.
- Un tipo de parámetro puede estar asociado a muchos parámetros.

NOMBRE DE ENTIDAD : PuertosLPT
DESCRIPCIÓN : Almacena la información sobre puertos LPT disponibles en el sistema.
ALIASES : PUERTOSLPT
OCURRENCIA - Un puerto LPT puede tener hasta 3 buses: Datos = 0, Estado = 1 y Control = 2.

NOMBRE DE ENTIDAD : TipoBuses
DESCRIPCIÓN : Almacena información sobre los tipos de buses del puerto LPT (datos, estado, control).
ALIASES : TIPOBUSES
OCURRENCIA - A cada bus del puerto LPT le corresponde solo un tipo de bus.

NOMBRE DE ENTIDAD : Sentidos
DESCRIPCIÓN : Almacena información sobre sentidos de dirección.
ALIASES : SENTIDOS
OCURRENCIA - Cada tipo de bus del puerto LPT, le corresponde un sentido: Entrada, Salida o Bidireccional.

NOMBRE DE ENTIDAD : DatosBusLPT
DESCRIPCIÓN : Almacena los valores numéricos, que representan los bytes que pueden ser enviados o recibidos por los distintos buses del puerto LPT.
ALIASES : DATOSBUSLPT
OCURRENCIA
- Cada bus de un puerto LPT puede enviar o recibir, de acuerdo a su tipo, un conjunto de valores enteros entre 0 y 255, que representan los bytes que ingresan o salen por cada uno de estos buses.

NOMBRE DE ENTIDAD : Metodos
DESCRIPCIÓN : Almacena los nombres de “Métodos” asociados a “Clases” del software que están expuestos al usuario, para ser asociados con pares “Dispositivos.Acción” definidos por este último.
ALIASES : METODOS
OCURRENCIA - Cada clase de objeto dentro de un sistema en particular, puede exponer muchos métodos para ser utilizados por el usuario.

NOMBRE DE ENTIDAD : ClasesObj
DESCRIPCIÓN : Almacena los nombres de “Clases” del software que están expuestas al usuario, para ser asociados con pares “Dispositivos.Acción” definidos por este último.
ALIASES : CLASESOBJ
OCURRENCIA - Cada sistema puede exponer una o más clases internas, para ser utilizadas por el usuario.

NOMBRE DE ENTIDAD : Acciones
DESCRIPCIÓN : Almacena las acciones definidas por el usuario para cada dispositivo.
ALIASES : ACCIONES
OCURRENCIA - Un usuario en un sistema en particular, puede asociar cero o más acciones para un dispositivo definido.

NOMBRE DE ENTIDAD : Dispositivos
DESCRIPCIÓN : Almacena los datos para los dispositivos definidos por el usuario.
ALIASES : DISPOSITIVOS
OCURRENCIA - Un usuario en un sistema en particular, puede definir cero o más dispositivos.

NOMBRE DE ENTIDAD : DispElecMeca
DESCRIPCIÓN : Especialización de la tabla “Dispositivos” que almacena los datos para los dispositivos electrónicos / mecánicos.
ALIASES : DISPELECMECA
OCURRENCIA - Cero o más dispositivos definidos por el usuario para un sistema en particular, pueden especializarse como dispositivos electrónicos / mecánicos.

NOMBRE DE ENTIDAD : DispEMMotoresDC
DESCRIPCIÓN : Especialización de la sub-especialización “DispElecMeca” que almacena los datos para los dispositivos electrónicos / mecánicos de tipo “Motores DC”.
ALIASES : DISPEMMOTORESDC
OCURRENCIA - Cero o más dispositivos electrónicos / mecánicos definidos por el usuario para un sistema en particular, pueden sub-especializarse como dispositivo “Motor de Corriente Continua”.

NOMBRE DE ENTIDAD : DispEMMotoresPAP
DESCRIPCIÓN : Especialización de la sub-especialización “DispElecMeca” que almacena los datos para los dispositivos electrónicos / mecánicos de tipo “Motores Paso A Paso”.
ALIASES : DISPEMMOTORESPAP
OCURRENCIA - Cero o más dispositivos electrónicos / mecánicos definidos por el usuario para un sistema en particular, pueden sub-especializarse como dispositivo “Solenoide”.

NOMBRE DE ENTIDAD : DispEMSolenoides

DESCRIPCIÓN : Especialización de la sub-especialización “DispElecMeca” que almacena los datos para los dispositivos electrónicos / mecánicos de tipo “Solenoides”.

ALIASES : DISPEMSOLENOIDES

OCURRENCIA

- Cero o más dispositivos electrónicos / mecánicos definidos por el usuario para un sistema en particular, pueden sub-especializarse como dispositivo “Motor Paso a Paso”.

9.1.2 Relaciones

Las entidades anteriormente presentadas, exhiben las siguientes relaciones:

TIPO DE ENTIDAD	: GuanteRV
TIPO DE RELACIÓN	: GuanteRVTiposPatrones
DESCRIPCIÓN	- Un GuanteRV puede tener n patrones de movimientos, y un patron de movimiento pude estar asociado a n GuantesRV.
TIPO ENTIDAD	: TipoPatronesGuanteRV
CARDINALIDAD	: n, n
EXISTENCIA (PARTICIPACIÓN): (O:M)	: M: M

TIPO DE ENTIDAD	: Sistema
TIPO DE RELACIÓN	: SistemaTieneGuantesRV
DESCRIPCIÓN	: Un sistema puede tener conectados n guantes de realidad virtual, y cada GuanteRV puede estar conectado a un solo sistema a la vez.
TIPO ENTIDAD	: GuanteRV
CARDINALIDAD	: 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M)	: M: M

TIPO DE ENTIDAD : TipoControl
TIPO DE RELACIÓN : TipoControlTieneMovimientos
DESCRIPCIÓN : Un tipo de control sobre el ambiente virtual tiene asociados n movimientos del ambiente, y cada tipo de movimiento del AmbienteRV se relaciona con un solo tipo de control.
TIPO ENTIDAD : TipoMovimientos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Usuarios
TIPO DE RELACIÓN : UsuarioTieneAmbienteRV
DESCRIPCIÓN : Un usuario solo puede tener una configuración sobre el ambiente virtual, y cada instancia de configuración del AmbienteRV puede pertenecer a un solo usuario.
TIPO ENTIDAD : AmbienteRV
CARDINALIDAD : 1, 1
EXISTENCIA (PARTICIPACIÓN): (O:M) : O: M

TIPO DE ENTIDAD : TipoControl
TIPO DE RELACIÓN : TipoControlSobreAmbiente
DESCRIPCIÓN : Un tipo de control puede estar asociado a n configuraciones del ambiente virtual, y a cada instancia de configuración del ambiente virtual le corresponde solo un tipo de control.
TIPO ENTIDAD : AmbienteRV
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Usuarios
TIPO DE RELACIÓN : CfgUsuarioSintRecoVoz
DESCRIPCIÓN : Un usuario puede tener una configuración sobre el sintetizador y reconocedor de voz, y cada instancia del sintetizador y reconocedor, puede pertenecer a un solo usuario.
TIPO ENTIDAD : SintRecoVoz
CARDINALIDAD : 1, 1
EXISTENCIA (PARTICIPACIÓN): (O:M) : O: M

TIPO DE ENTIDAD : ServDispositivosIR
TIPO DE RELACIÓN : SerDispIRTieneDispIR
DESCRIPCIÓN : N dispositivos para manejo de infrarrojos pueden estar conectados a un servidor de dispositivos, y un dispositivo IR puede estar conectado a un solo servidor de dispositivos a la vez.
TIPO ENTIDAD : DispositivosIR
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Remotos
TIPO DE RELACIÓN : RemotosTienenComandos
DESCRIPCIÓN : Un control remoto puede tener n comandos IR, y cada comando (señal IR) esta asociada a un solo control remoto.
TIPO ENTIDAD : Comandos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Habitaciones
TIPO DE RELACIÓN : HabitacionesTienenDispIR
DESCRIPCIÓN : En una habitación se pueden encontrar n dispositivos para manejo de señales infrarrojas, y cada dispositivo IR puede o no estar asociado a solo una habitación.
TIPO ENTIDAD : DispositivosIR
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : O: M

TIPO DE ENTIDAD : DispositivosIR
TIPO DE RELACIÓN : DispIRTienenRemotos
DESCRIPCIÓN : Cada dispositivo para manejo de infrarrojos puede tener definidos n controles remotos, y cada control remoto está asociado a solo un dispositivo IR.
TIPO ENTIDAD : Remotos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Sistema
TIPO DE RELACIÓN : SistemaTieneDispIR
DESCRIPCIÓN : Un sistema puede tener n servidores de dispositos IR, y cada servidor de dispositivos IR puede estar conectado a solo un sistema.
TIPO ENTIDAD : ServDispositivosIR
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Sistema
TIPO DE RELACIÓN : SistemaTieneProgramas
DESCRIPCIÓN : Un sistema puede cargar y controlar n programas ejecutables, y cada ejecutable es controlado por un solo sistema.
TIPO ENTIDAD : Ejecutables
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : TipoParametros
TIPO DE RELACIÓN : ParametrosTienenTipo
DESCRIPCIÓN : Un tipo de parámetros puede tener n instancias de parámetros asociada, y cada instancia de parámetros le corresponde un solo tipo.
TIPO ENTIDAD : Parámetros
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : PuertosLPT
TIPO DE RELACIÓN : BusesLPT
DESCRIPCIÓN : Un puerto LPT puede tener n tipo de buses y cada tipo de bus puede estar asociado a n puertos LPT.
TIPO ENTIDAD : TipoBuses
CARDINALIDAD : n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Sentidos
TIPO DE RELACIÓN : BusesTienenSentido
DESCRIPCIÓN : Un sentido de dirección (entrada, salida o bidireccional), puede estar asociado a n tipo de buses, y un tipo de bus solo posee un sentido de dirección.
TIPO ENTIDAD : TipoBuses
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : TipoBuses
TIPO DE RELACIÓN : DatosTipoBusLPT
DESCRIPCIÓN : Un tipo de bus puede enviar o recibir, de acuerdo a su tipo, n datos, y cada dato esta asociado a un solo tipo de bus.
TIPO ENTIDAD : DatosBusLPT
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : PuertosLPT
TIPO DE RELACIÓN : BusesERDatos
DESCRIPCIÓN : Un puerto LPT puede enviar n cantidad de datos por cada bus, y cada dato de cada bus, es enviado o recibido por solo un puerto LPT a la vez.
TIPO ENTIDAD : DatosBusLPT
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Sistema
TIPO DE RELACIÓN : SistemaTienePrtsLPT
DESCRIPCIÓN : Un sistema puede tener n puertos LPT, y cada puerto LPT solo puede estar en un sistema.
TIPO ENTIDAD : PuertosLPT
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : ClasesObj
TIPO DE RELACIÓN : ClasesTienenMetodos
DESCRIPCIÓN : Una clase expuesta por el sistema puede tener asociados n métodos, y cada método solo se puede asociar a una clase de objeto a la vez.
TIPO ENTIDAD : Metodos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Sistema
TIPO DE RELACIÓN : SistemaTieneClasesObj
DESCRIPCIÓN : N clases de objetos pueden estar definidas y expuestas por un sistema, y un sistema puede exponer muchas clases de objetos.
TIPO ENTIDAD : ClasesObj
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : TipoParametros
TIPO DE RELACIÓN : MetodosTienenParametros
DESCRIPCIÓN : Un tipo de parámetro puede ser utilizado por n métodos, y cada método solo utiliza un tipo de parámetro.
TIPO ENTIDAD : metodos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Dispositivos
TIPO DE RELACIÓN : DispositivosTienenAcciones
DESCRIPCIÓN : Un dispositivo puede tener asociadas n acciones, y cada acción está asociada a solo un dispositivo a la vez.
TIPO ENTIDAD : Acciones
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Acciones
TIPO DE RELACIÓN : AccionesEjecutanFunciones
DESCRIPCIÓN : Una acción puede ejecutar distintas funciones, y cada función es ejecutada por una acción.
TIPO ENTIDAD : Ejecutar
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Sistema
TIPO DE RELACIÓN : SistemaTieneDisp
DESCRIPCIÓN : Un sistema puede tener n dispositivos definidos por el usuario, y cada dispositivo pertenece a un sistema en particular.
TIPO ENTIDAD : Dispositivos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Usuarios
TIPO DE RELACIÓN : UsuariosTienenDisp
DESCRIPCIÓN : Un usuario puede definir n dipositivos en un sistema, y un dispositivo en particular le pertenece a un solo usuario.
TIPO ENTIDAD : Dispositivos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Habitaciones
TIPO DE RELACIÓN : HabitacionesTienenDispositivos
DESCRIPCIÓN : En una habitación pueden haber n dispositivos definidos por el usuario, y un dispositivo solo puede estar en una habitación a la vez.
TIPO ENTIDAD : Dispositivos
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : O: M

TIPO DE ENTIDAD : Metodos
TIPO DE RELACIÓN : EjecutarMetodo
DESCRIPCIÓN : Un método puede ejecutar varias funciones, y cada función es ejecutada por un método.
TIPO ENTIDAD : Ejecutar
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Parametros
TIPO DE RELACIÓN : EjecutarParametros
DESCRIPCIÓN : Un parámetro puede ser utilizado en n ejecuciones de funciones, y a cada ejecución le corresponde una sola combinación de parámetros.
TIPO ENTIDAD : Ejecutar
CARDINALIDAD : 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Acciones
TIPO DE RELACIÓN : Ejecutar
DESCRIPCIÓN : Una acción definida por el usuario para un dispositivo, puede ejecutar n combinaciones de metodos de objetos con sus respectivas combinaciones de parámetros, a su vez, una ejecución pertenece a solo una asociación de accion con metodos y sus respectivos parametros.
TIPO ENTIDAD : - Metodos
- Parametros
CARDINALIDAD : n, n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M: M

TIPO DE ENTIDAD : Sistema
TIPO DE RELACIÓN : SistemaTieneUsuario
DESCRIPCIÓN : Un sistema puede tener uno o más usuarios asignados, asi como un usuario puede estar asociado a uno o más sistemas.
TIPO ENTIDAD : Usuarios
CARDINALIDAD : n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Usuarios
TIPO DE RELACIÓN : CalibrarGuanteRV
DESCRIPCIÓN : Un usuario puede calibrar uno o mas guantes de realidad virtual, de igual forma, un GuanteRV puede ser calibrado por uno o más usuarios para su utilización.
TIPO ENTIDAD : GuanteRV
CARDINALIDAD : n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M

TIPO DE ENTIDAD : Teclado
TIPO DE RELACIÓN : TeclasAmbienteRV
DESCRIPCIÓN : Una tecla puede estar asociada a uno o más tipos de movimientos del ambiente virtual para una o más configuraciones del AmbienteRV en particular.
TIPO ENTIDAD : - TipoMovimientos
- AmbienteRV
CARDINALIDAD : n, n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M: M

TIPO DE ENTIDAD : Mouse
TIPO DE RELACIÓN : MouseAmbienteRV
DESCRIPCIÓN : Una acción del mouse puede estar asociada a uno o más tipos de movimientos del ambiente virtual para una o más configuraciones del AmbienteRV en particular.
TIPO ENTIDAD : - TipoMovimientos
- AmbienteRV
CARDINALIDAD : n, n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M: M

TIPO DE ENTIDAD : TipoPatronesGuanteRV
TIPO DE RELACIÓN : PatronesGuanteRV
DESCRIPCIÓN : Un tipo de patron de movimiento puede estar asociado a una o más calibraciones de patrones para uno o más usuario y uno o mas guantes de realidad virtual. A su vez, cada patron de movimiento calibrado, puede estar asociado a uno o más tipo de movimientos para una configuración del ambiente virtual.
TIPO ENTIDAD : - TipoMovimientos
- AmbienteRV
- Usuarios
- GuanteRV
CARDINALIDAD : n, n, n, n, n
EXISTENCIA (PARTICIPACIÓN): (O:M) : M: M: M: M: M

TIPO DE ENTIDAD	: Acciones
TIPO DE RELACIÓN	: AccionesAgrupanAcciones
DESCRIPCIÓN	: Una acción definida por el usuario, puede asociarse con una o más acciones, y cada acción agrupada esta asociada a solo una acción definida.
TIPO ENTIDAD	: Acciones
CARDINALIDAD	: 1, n
EXISTENCIA (PARTICIPACIÓN): (O:M)	: M: M

9.1.3 Identificar y asociar atributos con una entidad o relación

Del análisis de las entidades y relaciones antes planteadas, se presentaron los siguientes atributos para cada una de ellas.

Entidad Acciones

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdAccion	Carácter (30) Ej: “Encender”, “Cerrar”	PK	NO	Identificador de la acción definido por el usuario.
Imagen	Carácter (100)		SI	Almacena la ruta relativa al ejecutable donde se encuentra la imagen asociada a la acción.
Descripcion	Carácter (30)		NO	Descripción y / o información adicional.
GatillaEstado	Carácter (30)		SI	Estado del dispositivo que gatilla la acción.
ReqConfir	Boleano		NO	Establece si la acción requiere confirmación o no.
Visible	Boleano		NO	Establece si la acción es visible en el ambiente virtual o no.

Entidad AmbienteRV

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
ImagenFondo	Carácter (20)		SI	Nombre de la imagen de fondo que presentará el ambiente virtual.
TpoMaxReco	Decimal (3, 2) Ej: 0,5 (1/2 Segundo)		NO	Tiempo máximo que asignara el ambiente virtual para el reconocimiento de patrones de movimiento provenientes del guante de realidad virtual.
SoniArr	Carácter (15) Ej: “sonido.wav”		SI	Nombre del archivo de sonido (*.wav) que reproducirá el ambiente virtual al ejecutar un movimiento hacia arriba en el tarjetero vertical.
SoniAba	Carácter (15)		SI	Nombre del archivo de sonido (*.wav) que reproducirá el ambiente virtual al ejecutar un movimiento hacia abajo en el tarjetero vertical.
SoniIzq	Carácter (15)		SI	Nombre del archivo de sonido (*.wav) que reproducirá el ambiente virtual al ejecutar un movimiento hacia la izquierda en el tarjetero horizontal.
SoniDer	Carácter (15)		SI	Nombre del archivo de sonido (*.wav) que reproducirá el ambiente virtual al ejecutar un movimiento hacia la derecha en el tarjetero horizontal.
SintComando	Boleano		NO	Establece si se debe reproducir o no, por voz sintetizada, la selección actual en los tarjeteros.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
ControlTec	Boleano		NO	Establece si el ambiente virtual es controlado por medio de un teclado o no.
ControlRat	Boleano		NO	Establece si el ambiente virtual es controlado por medio de un teclado o no.
ControlGRV	Boleano		NO	Establece si el ambiente virtual es controlado por medio de un guante de realidad virtual o no.

Entidad ClasesObj

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdClaseObj	Carácter (30) Ej: “IR”, “LPT”, “API”	PK	NO	Nombre único que identifica la “Clase” al interior del software.
Descripción	Carácter (100)		SI	Descripción y / o información adicional.

Entidad Comandos

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdComando	Carácter (30) Ej: “encender”, “canalmas”	PK	NO	Nombre único que identifica el “Comando” y la señal infrarroja asociada a este controlada por el servidor “irTrans”.

Entidad DatosBusLPT

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
Dato	Entero 0-255	PK	NO	Número entre 0 y 255 que representa el byte que puede ser enviado o recibido por los distintos buses del puerto LPT.

Entidad DispEMMotoresDC

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
MaxValRet	Entero 0-99		NO	Máximo valor de retorno devuelto para este dispositivo por el sistema de retroalimentación del circuito Electrónico / Mecánico.
ValorRetActual	Entero 0-99		NO	Almacena el ultimo valor de retorno informado, para este dispositivo, por el sistema de retroalimentación del circuito Electrónico / Mecánico.
MensRet	Carácter (30) Ej: “Sistema de retorno accionado”		NO	Mensaje que retornará el sistema de retroalimentación para este dispositivo.

Entidad DispEMMotoresPAP

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
MaxValRet	Entero 0-99		NO	Máximo valor de retorno devuelto para este dispositivo por el sistema de retroalimentación del circuito Electrónico / Mecánico.
ValorRetActual	Entero 0-99		NO	Almacena el ultimo valor de retorno informado, para este dispositivo, por el sistema de retroalimentación del circuito Electrónico / Mecánico.
MensRet	Carácter (30) Ej: "Sistema de retorno accionado"		NO	Mensaje que retornara el sistema de retroalimentación para este dispositivo.

Entidad DispEMSolenoides

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
ValRetEst1	Entero 0-99		NO	Valor de comparación utilizado por el sistema de retroalimentación para el estado 1 del solenoide.
MensRetEst1	Carácter (30) "Solenoide extendido"		NO	Mensaje que retornara el sistema de retroalimentación para el estado 1 del solenoide.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
ValRetEst2	Entero 0-99		NO	Valor de comparación utilizado por el sistema de retroalimentación para el estado 2 del solenoide.
MensRetEst2	Carácter (30) "Solenoide retraído"		NO	Mensaje que retornara el sistema de retroalimentación para el estado 2 del solenoide.

Entidad DispElecMeca

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
PosAbsoluta	Entero 0-99		NO	Posición absoluta del dispositivo "ElecMeca" en el bus del circuito Electrónico / Mecánico.

Entidad Dispositivos

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdDispositivo	Carácter (30)	PK	NO	Nombre único definido por el usuario para especificar un "Dispositivo".
Imagen	Carácter (100)		SI	Almacena la ruta relativa al ejecutable donde se encuentra la imagen asociada al dispositivo.
Descripción	Carácter (30)		SI	Descripción y / o información adicional.
EstadoActual	Carácter (30) Ej: "Encendido"		SI	Almacena el ultimo estado informado o establecido sobre el dispositivo.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
Visible	Boleano		NO	Establece si el dispositivo es visible en el ambiente virtual o no.

Entidad DispositivosIR

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdDispositivoIR	Entero 0-15	PK	NO	Número entero que indetifica el dispositivo IR conectado al bus “irTrans” (0-15).

Entidad Ejecutables

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdEjecutable	Caracter (30) Ej; “Winamp”, “Excel”.	PK	NO	Nombre único utilizado para identificar al archivo ejecutable.
Ruta	Caracter (100) Ej: “C:\winamp.exe”	PK	NO	Ruta en donde se encuentra el archivo ejecutable.
Descripción	Carácter (50)		SI	Descripción y / o información adicional.

Entidad GuanteRV

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
PuertoCOMM	Entero 0-99	PK	NO	Número que identifica el puerto serial al cual esta conectado el guante de realidad virtual “NewGlove”.
LargoCable	Entero 0-999		SI	Extensión en metros del cable del guante de realidad virtual “NewGlove”.

Entidad Habitaciones

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdHabitacion	Carácter (50) Ej: “Sala, Cocina”	PK	NO	Identificador único de habitación.
Descripción	Carácter (50)	AK	NO	Descripción y / o información adicional.

Entidad Metodos

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdMetodo	Caracter (30) Ej: “EnviarSeñal”, “EnviarByte”, “EjecutarPrograma”	PK	NO	Nombre único que identifica el “Método” asociado a la “Clase” al interior del software.
Descripción	Caracter (100)		NO	Descripción y / o información adicional.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
CantPara	Entero 0-99		NO	Cantidad total de parámetros que necesita el método para su ejecución.

Entidad Mouse

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdAccionMouse	Entero 1-99	PK	NO	Identificador numérico de la acción del mouse.
Descripción	Carácter (30) Ej: “MouseUp”, “MouseClick”	AK	NO	Descripción y / o información adicional.

Entidad Parametros

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdParametro	Entero 1-9999	PK	NO	Identificador del parámetro generalizado.

Entidad PuertosLPT

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
DirLPT	Entero 1-999 Ej: 888 (LPT1)	PK	NO	Dirección base del puerto LPT en formato de número entero.

Entidad Remotos

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdRemoto	Carácter (30) Ej: “tv”, “cable”	PK	NO	Nombre único que identifica el control remoto y el archivo que contiene las señales infrarrojas manejadas por el servidor “irTrans”.

Entidad Sentidos

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
Disentido	Entero 1-99 Ej: 0, 1, 2	PK	NO	Identificador del sentido de dirección.
Descripción	Carácter (30) Ej: “Datos, ””Estado”, “Control”	AK	NO	Descripción y / o información adicional.

Entidad ServDispositivosIR

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
PuertoSocket	Entero 1-99999 Ej: 21000	PK	NO	Número entero que identifica el puerto socket al cual se conecta el software cliente al servidor de dispositivos ir “irTrans”.
DispDefecto	Boleano		NO	Establece si es el dispositivo servidor por defecto a utilizar por el sistema o no.

Entidad SintRecoVoz

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
VozSexo	Boleano Ej: 0, 1 (masculino, femenino)		NO	Tipo de voz: 1 – Masculina 0 – Femenina.
VozVel	Entero 1-99		NO	Velocidad o “Pitch” de la voz.
RepPalabra	Boleano		NO	Establece si se deben repetir por sintetizador de voz las palabras reconocidas.
RepConfir	Boleano		NO	Establece si se debe repetir una confirmación por sintetizador de voz cuando se reconoce una palabra.
PalConfir	Carácter (30) Ej: “ok”, “entendido” , “escuché”		SI	La palabra que se debe repetir para confirmar el reconocimiento de una palabra (depende del atributo anterior).

Entidad Sistema

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdSistema	Carácter (20) Ej: “DomoPC”, “HAL”	PK	NO	Identificador del sistema (equipo o computadora).
IPSistema	Carácter (50) Ej: “192.168.0.20”		SI	Dirección IP de la computadora.

Entidad Teclado

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdTecla	Entero 0-255 Ej: 65, 66	PK	NO	Identificador de tecla según el código en el sistema operativo.
Descripción	Carácter (20) Ej: “a”, “b”	AK	NO	Descripción y / o información adicional.

Entidad TipoBuses

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdTipoBus	Entero 0-2 Ej: 0, 1, 2	PK	NO	Identificador del tipo de bus (Datos, Estado, Control)
Descripción	Carácter (30) Ej: “Datos”, “Estado”, “Control”	AK	NO	Descripción y / o información adicional.

Entidad TipoControl

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdTipoControl	Entero 0-1 Ej: 0, 1	PK	NO	Identificador del tipo de control sobre el ambiente virtual (1 = manual o 0 = giro automático).
Descripción	Carácter (30) Ej: “Giro Manual”, “Giro Automatico”	AK	NO	Descripción y / o información adicional.

Entidad TipoMovimientos

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdTipoMovimiento	Entero 1-99	PK	NO	Identificador del tipo de movimiento en el ambiente de realidad virtual.
Descripción	Carácter (50) Ej: "Tarjetero Arriba", "Seleccionar"	AK	NO	Descripción y / o información adicional.

Entidad TipoParametros

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdTipoParametro	Entero 1-99	PK	NO	Identificador del "Tipo de Parámetro" (combinación, cantidad).
Descripción	Carácter (100) Ej: "Dirección, Bus, Dato"	AK	NO	Descripción y / o información adicional.

Entidad TipoPatronesGuanteRV

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdTipoPatron	Entero 0-11	PK	NO	Identificador del patrón de movimiento que puede realizar el guante de realidad virtual “NewGlove”.
SenMovPatron	Entero 0-3		NO	Sentido de movimiento del patrón de movimiento.
Descripción	Carácter (30) Ej: “Flectar Pulgar”, “Girar Derecha”	AK	NO	Descripción y / o información adicional.

Entidad Usuarios

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdUsuario	Carácter (20) Ej: “Usuario1”, “pgonzales”, “buho”	PK	NO	Identificador de usuario.
NomUsuario	Carácter (20) “Pedro R.”		NO	Nombre del usuario.
ApeUsuario	Carácter (20) “Gonzales S.”		NO	Apellido del usuario.

Relación AccionesAgrupanAcciones

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
NumEjecucion	Entero 1-999		NO	Número de ejecución de la acción agrupada

Relación CalibrarGuanteRV

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
SegmentosDibujo	Entero 1-99		NO	Resolución del dibujo que representa la mano virtual.
ValMaxPulgar	Entero 1-999		NO	Valor máximo que puede tomar el dedo pulgar en la representación virtual de la mano.
ValMinPulgar	Entero 1-999		NO	Valor mínimo que puede tomar el dedo pulgar en la representación virtual de la mano.
SenPulgar	Entero 1-999		NO	Valor que representa la sensibilidad con la que será interpretado el movimiento del dedo pulgar en el dibujo virtual, a mayor el número, menor sensibilidad.
ValMaxIndice	Entero 1-999		NO	Valor máximo que puede tomar el dedo índice en la representación virtual de la mano.
ValMinIndice	Entero 1-999		NO	Valor mínimo que puede tomar el dedo índice en la representación virtual de la mano.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
SenIndice	Entero 1-999		NO	Valor que representa la sensibilidad con la que será interpretado el movimiento del dedo pulgar en el dibujo virtual, a mayor el número, menor sensibilidad.
ValMaxMedio	Entero 1-999		NO	Valor máximo que puede tomar el dedo medio en la representación virtual de la mano.
ValMinMedio	Entero 1-999		NO	Valor mínimo que puede tomar el dedo medio en la representación virtual de la mano.
SenMedio	Entero 1-999		NO	Valor que representa la sensibilidad con la que será interpretado el movimiento del dedo medio en el dibujo virtual, a mayor el número, menor sensibilidad.
ValMaxAnular	Entero 1-999		NO	Valor máximo que puede tomar el dedo anular en la representación virtual de la mano.
ValMinAnular	Entero 1-999		NO	Valor mínimo que puede tomar el dedo anular en la representación virtual de la mano.
SenAnular	Entero 1-999		NO	Valor que representa la sensibilidad con la que será interpretado el movimiento del dedo anular en el dibujo virtual, a mayor el número, menor sensibilidad.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
ValMaxGiro	Entero 1-999		NO	Valor máximo que puede tomar el giro de la muñeca en la representación virtual de la mano.
ValMinGiro	Entero 1-999		NO	Valor mínimo que puede tomar el giro de la muñeca en la representación virtual de la mano.
SenGiro	Entero 1-999		NO	Valor que representa la sensibilidad con la que será interpretado el giro de la muñeca en el dibujo virtual, a mayor el número, menor sensibilidad.
ValMaxInclinacion	Entero 1-999		NO	Valor máximo que puede tomar la inclinación de la muñeca en la representación virtual de la mano.
ValMinInclinacion	Entero 1-999		NO	Valor mínimo que puede tomar la inclinación de la muñeca en la representación virtual de la mano.
SenInclinacion	Entero 1-999		NO	Valor que representa la sensibilidad con la que será interpretado el movimiento de inclinación de la muñeca en el dibujo virtual, a mayor el número, menor sensibilidad.
ActDibPulgar	Boleano		NO	Establece si se actualiza o no la posición del dedo pulgar en el dibujo virtual de la mano, según los datos provenientes del guante de realidad virtual.

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
ActDibIndice	Boleano		NO	Establece si se actualiza o no la posición del dedo índice en el dibujo virtual de la mano, según los datos provenientes del guante de realidad virtual.
ActDibMedio	Boleano		NO	Establece si se actualiza o no la posición del dedo medio en el dibujo virtual de la mano, según los datos provenientes del guante de realidad virtual.
ActDibAnular	Boleano		NO	Establece si se actualiza o no la posición del dedo anular en el dibujo virtual de la mano, según los datos provenientes del guante de realidad virtual.
ActDibGiro	Boleano		NO	Establece si se actualiza o no la posición de giro en el dibujo virtual de la mano, según los datos provenientes del guante de realidad virtual.
ActDibInclinacion	Boleano		NO	Establece si se actualiza o no la posición de inclinación en el dibujo virtual de la mano, según los datos provenientes del guante de realidad virtual.

Relación MouseAmbienteRV

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
NumVecesAccionMouse	Entero 1-5	PK	NO	Cantidad de veces que se debe repetir la acción en un tiempo determinado para que se realice el movimiento en el ambiente virtual.

Relación PatronesGuanteRV

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
RangoPatron	Entero 0-99		NO	Rango de movimiento del patrón que desea utilizar el usuario para que sea reconocido como un movimiento en particular.
PorcentajeError	Entero 0-100 Ej: 35 (35%)		NO	Porcentaje de error aceptable en el reconocimiento del patrón de movimiento.
NumVecesPatron	Entero 1-5	PK	NO	Cantidad de veces que se debe repetir el patrón de movimiento de la mano en un tiempo determinado para que se realice el movimiento en el ambiente virtual.

Relación SistemaTieneUsuarios

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
UsuDefe	Boleano		NO	Define si es el usuario por defecto del sistema o no.
ApliMini	Boleano		NO	Establece si la aplicación debe iniciarse minimizada.
RecoActi	Boleano		NO	Establece si la aplicación debe iniciar automáticamente el reconocimiento de voz o no.
AmbRVAct	Boleano		NO	Establece si la aplicación debe iniciar automáticamente el ambiente virtual.

Relación TeclasAmbienteRV

Nombre	Carac. del Dato	Restricción (PK, AK)	Nulo	Descripción
NumVecesTecla	Entero 1-5 Ej: 3, 4	PK	NO	Cantidad de veces que se debe repetir la presión de la tecla en un tiempo determinado para que se realice el movimiento en el ambiente virtual.

Relación Ejecutar

Nombre	Tipo de Dato	Restricción (PK, AK)	Nulo	Descripción
IdEjecucion	Entero 1-999	PK	NO	Identificador de la ejecución a realizar por la acción asociada la dispositivo.
TpoEspera	Decimal (10, 3) Ej: 0, 005 (segundos)		NO	Tiempo que esperará la ejecución actual antes de realizarse.

NOTAS:

- Los “Alias” para cada atributo, sin excepción, son exactamente el mismo nombre escrito completamente en mayúsculas.

- Para esta instancia del modelo no se consideraron valores por defecto.

- La información respecto a los atributos con características de “Multivalóricos”, “Derivados” y “Compuestos”, se encuentra incorporada solo en las tablas en donde se presentan este tipo de atributos.

- La notación para el tipo de dato “Decimal” es: (Entero, Decimal).

- En la columna “Tipo de Dato”, se incluye la información respecto al: “tipo de dato”, “tamaño”, “rango” y “ejemplos”, según corresponda.

9.1.4 Determinar dominios de atributos

La información requerida para este punto, se incluye en el punto 9.1.3 en la columna “Tipo de Dato”.

9.1.5 Determinar claves candidatas y primarias de los atributos

A continuación se presenta una tabla resumen con las claves candidatas y primarias de los atributos de cada entidad.

TABLA Nº 4: Listado de Claves Primarias y Candidatas.

Nombre	Alias	Tipo	Pertenece
IdSistema	IDSISTEMA	PK	Entidad 'Sistema'
IdUsuario	IDUSUARIO	PK	Entidad 'Usuarios'
IdSistema	IDSISTEMA	PK	Entidad 'GuanteRV'
IdTipoPatron	IDTIPOPATRON	PK	Entidad 'TipoPatronesGuanteRV'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'TipoPatronesGuanteRV'
IdTipoMovimiento	IDTIPOMOVIMIENTO	PK	Entidad 'TipoMovimientos'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'TipoMovimientos'
IdTipoControl	IDTIPOCONTROL	PK	Entidad 'TipoControl'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'TipoControl'
IdAccionMouse	IDACCIONMOUSE	PK	Entidad 'Mouse'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'Mouse'
IdTecla	IDTECLA	PK	Entidad 'Teclado'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'Teclado'

Nombre	Alias	Tipo	Pertenece
PuertoSocket	PUERTOSOCKET	PK	Entidad 'ServDispositivosIR'
IdDispositivoIR	IDDISPOSITIVOIR	PK	Entidad 'DispositivosIR'
IdComando	IDCOMANDO	PK	Entidad 'Comandos'
Remoto	REMOTO	PK	Entidad 'Remotos'
IdHabitacion	IDHABITACION	PK	Entidad 'Habitaciones'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'Habitaciones'
IdEjecutable	IDEJECUTABLE	PK	Entidad 'Ejecutables'
IdParametro	IDPARAMETRO	PK	Entidad 'Parametros'
IdTipoParametro	IDTIPOPARAMETRO	PK	Entidad 'TipoParametros'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'TipoParametros'
DirLPT	DIRLPT	PK	Entidad 'PuertosLPT'
IdTipoBus	IDTIPOBUS	PK	Entidad 'TipoBuses'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'TipoBuses'
Disentido	IDSENTIDO	PK	Entidad 'Sentidos'
Key_Descripcion	KEY_DESCRIPCION		Entidad 'Sentidos'
Dato	DATO	PK	Entidad 'DatosBusLPT'
IdMetodo	IDMETODO	PK	Entidad 'Metodos'
IdClaseObj	IDCLASEOBJ	PK	Entidad 'ClasesObj'
IdAccion	IDACCION	PK	Entidad 'Acciones'
IdDispositivo	IDDISPOSITIVO	PK	Entidad 'Dispositivos'

9.1.6 Identificar tipo de entidades de superclase y subclase

Para el modelo conceptual aquí presentado, se realizó la siguiente especialización:

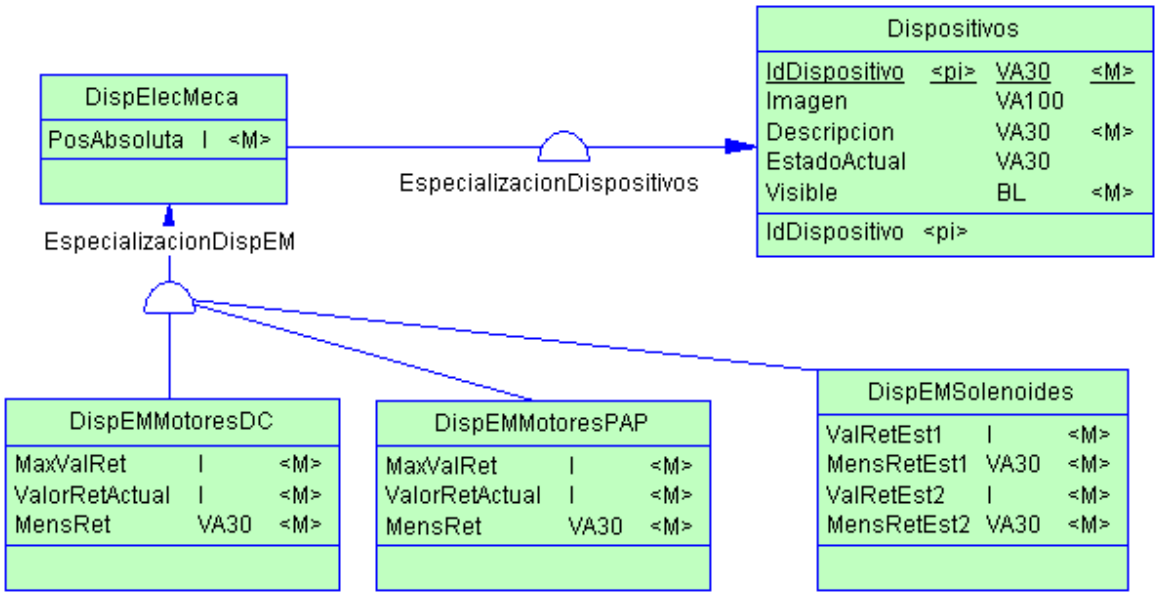

Diagrama Nº 23: Especialización Dispositivos.

Esta especialización doble se explica de la siguiente manera: Cada "Dispositivo" definido por el usuario, puede ser de algún tipo cualquiera, o bien, ser un dispositivo que haga referencia a algún aparato conectado al circuito de control "Electrónico / Mecánico". Si este es el caso, se necesitará información adicional para que el software, a través del sistema electrónico de

“Retroalimentación”, sepa en que posición (PosAbsoluta) dentro del circuito se encuentra conectado dicho aparato.

De la misma forma, si el dispositivo “Electrónico / Mecánico”, hace referencia a algún aparato como “cerradura eléctrica”, “aparato 220V” (lámpara, calentador, etc), no se necesitará más información adicional que la “Posición Absoluta” del aparato conectado al circuito, pero si el aparato al cual se hace referencia, es un “Motor DC” (motor de corriente continua), “Motor PAP” (motor paso a paso) o solenoide, se necesitará más información adicional tal como se muestra en el diagrama. Esto es para que el software en conjunto con el sistema electrónico de retroalimentación, pueda entregar correctamente la información de retorno. Ambas especializaciones son de tipo “disjuntas”.

La razón de mantener la primera especialización de “Dispositivo” a “DispElecMeca”, es que si bien, por ahora se podría haber hecho una sola especialización directa desde “Dispositivos” a “DispEMMotoresDC”, “DispEMMotoresPAP” y “DispEMSolenoides”, por motivos de escalabilidad, es muy probable que a futuro se necesiten especializaciones adicionales al nivel de “DispElecMeca”. Esto puede ser para dispositivos de tipo infrarrojos u otros, que aunque por ahora este tipo de dispositivos no lo necesiten, si sería necesario a la hora de implementar por ejemplo un sistema de retroalimentación para los dispositivos de este tipo.

En el caso de la sub-especialización “DispElecMeca” a “DispEMMotoresDC” y “DispEMMotoresPAP”, si bien por ahora mantienen los

mismos atributos, es muy probable que debido a las diferentes características entre estos tipos de motores, a futuro se necesiten otros valores diferentes para que el software pueda mantener correctamente actualizados los datos del sistema electrónico de retorno de información.

Además de la especialización presentada, se realizó la siguiente generalización:

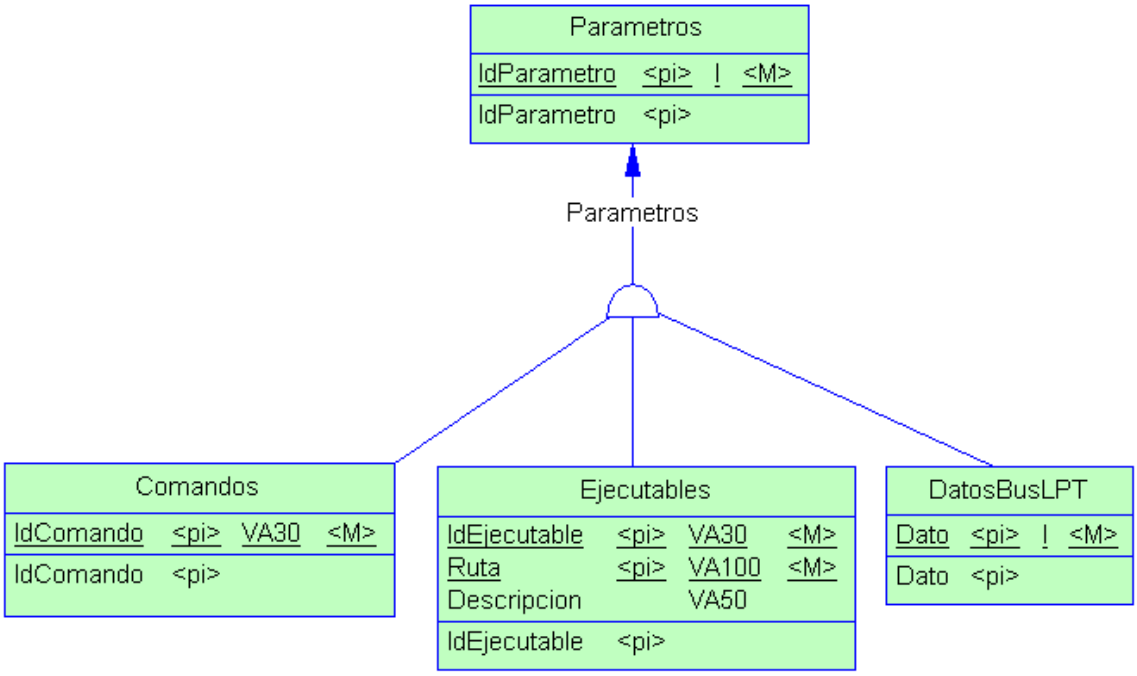

Diagrama Nº 24: Generalización Parámetros.

Esta generalización disjunta se explica de la siguiente forma: El sistema puede ofrecer, para su utilización, distintos tipos de datos o valores, estos pueden venir desde los dispositivos para manejo de señales infrarrojas, pasando por la entidad “Remotos” y llegando a la entidad “Comandos”, así como también datos respecto a los “Ejecutables” y a los “Datos” que pueden ser enviados o recibidos por los puertos LPT conectados al sistema (DatosBusLPT). Si bien todas estas entidades poseen claves primarias, estas difieren en forma y tipo unas de otras, y debido a que todos estos datos y valores pueden ser usados como parámetros por los “Métodos” que expone el sistema, es que se necesitó de un valor como clave primaria que generalice a todos los demás.

De esta forma se creó la entidad “Parametros” que genera un identificador único para cada instancia en las entidades de más abajo, esto también facilita la escalabilidad a futuro del modelo, puesto que si posteriormente se exponen nuevos métodos que necesiten nuevos tipos de valores como parámetros. Estas nuevas entidades pueden ser incorporadas a la generalización ya existente.

9.1.7 Diagrama Modelo Conceptual BDSAD

A continuación se presenta al diagrama conceptual de datos:

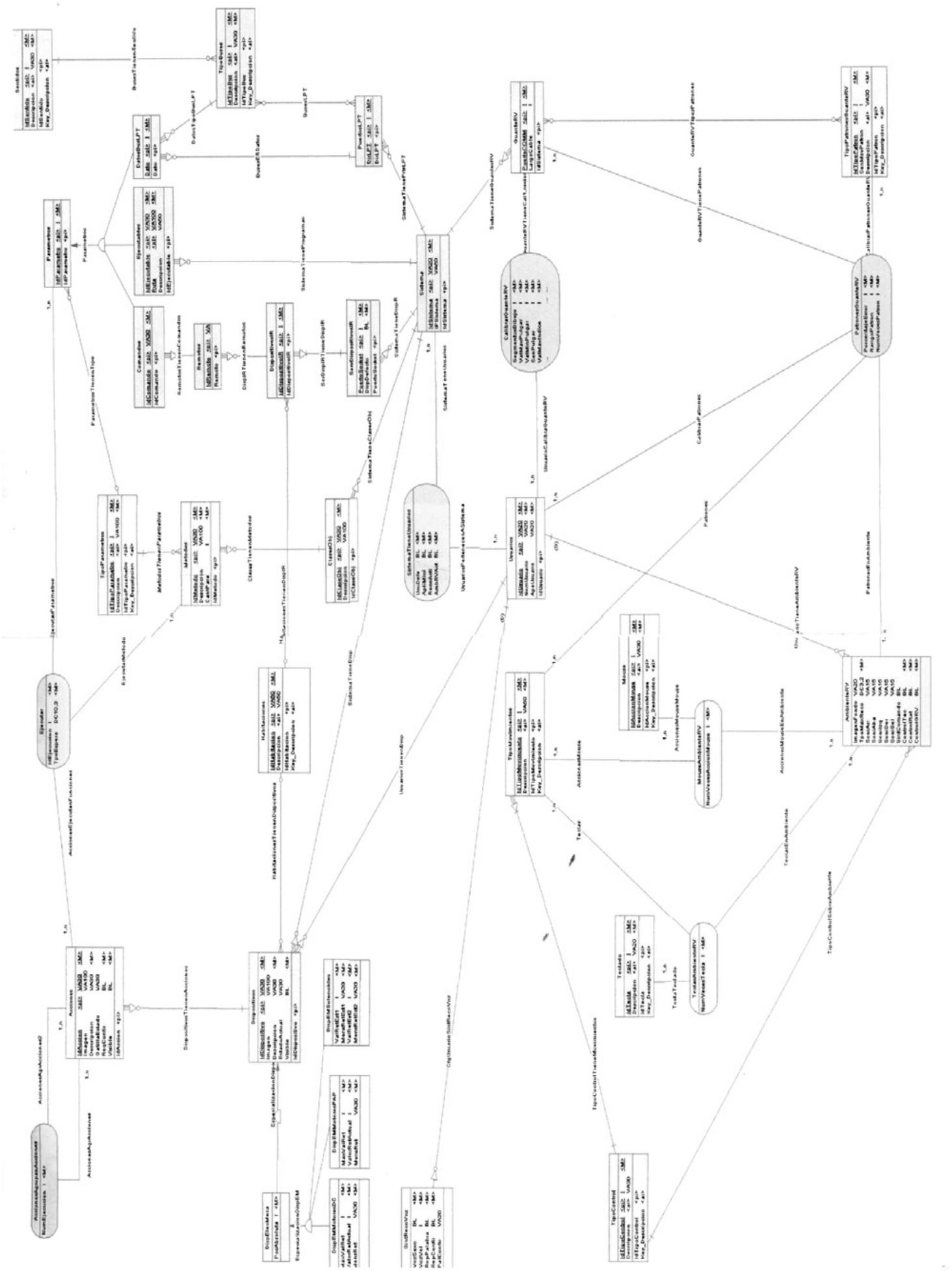

Diagrama Nº 25: Modelo Conceptual BDSAD.

9.2 Modelo Lógico BDSAD (“Base de Datos Sistema de Asistencia Domótica”)

A continuación, se presenta el desarrollo del modelo lógico para la base de datos “BDSAD”.

9.2.1 Mapeo del modelo conceptual a lógico

Para el presente modelo se realizó el siguiente mapeo de relaciones:

9.2.1.1 Relaciones “Muchos – A – Muchos”

La relación muchos-a-muchos formada por las entidades “GuanteRV” y “TipoPatronesGuanteRV”, se transformó tal como se muestra en el siguiente diagrama:

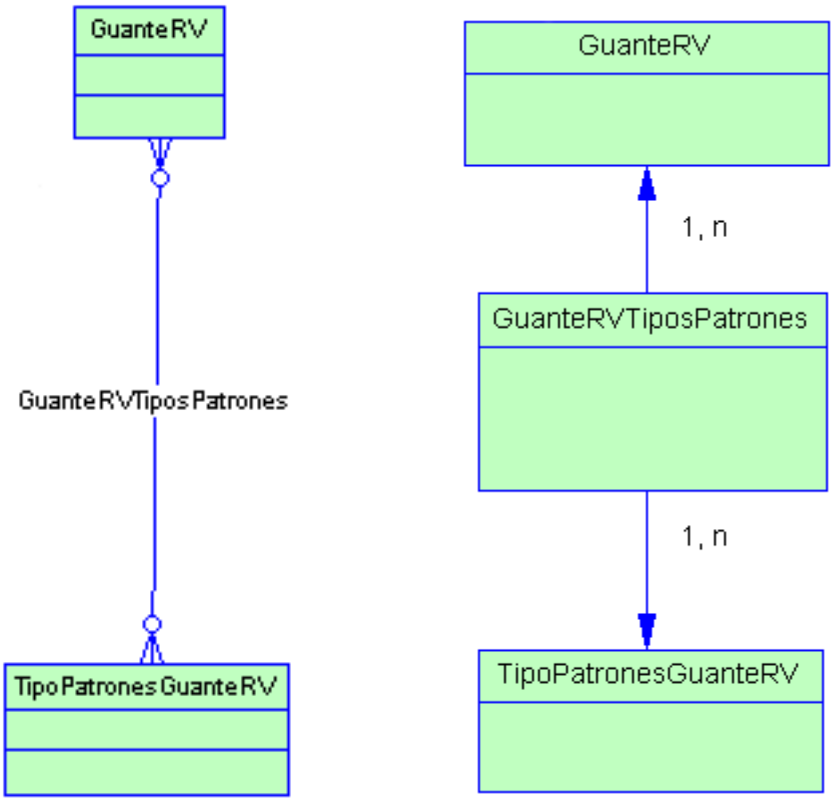

Diagrama Nº 26: Relación Muchos – A – Muchos "GuanteRV – A – TipoPatronesGuenteRV".

De la misma manera, la relación muchos-a-muchos formada por las entidades “TipoBuses” y “PuertosLPT”, se transformó tal como se muestra en el siguiente diagrama:

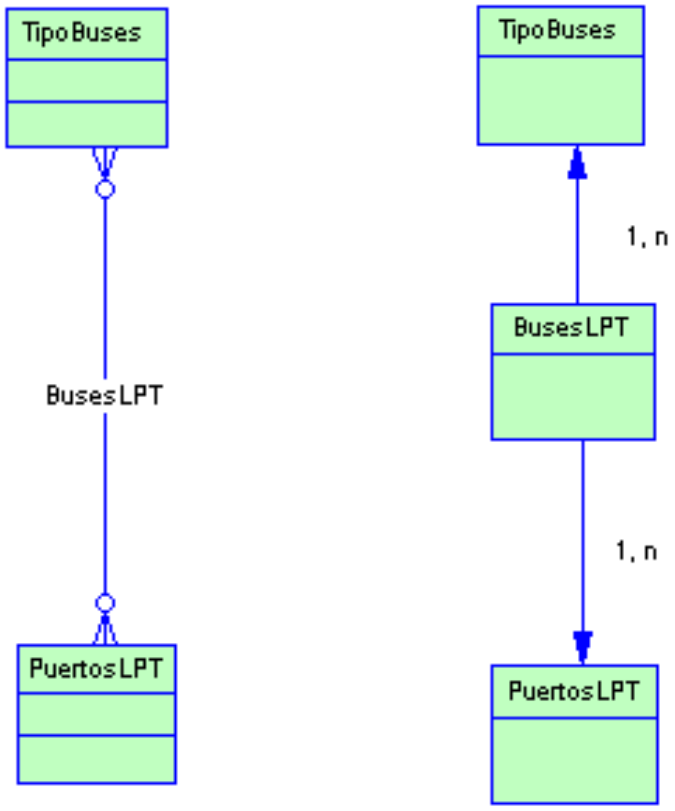

Diagrama Nº 27: Relación Muchos – A – Muchos “TipoBuses – A – PuertosLPT”.

9.2.1.2 Relaciones “Complejas”

La relación compleja “TeclasAmbienteRV” se transformó insertando una entidad intermedia tal como se muestra en el diagrama:

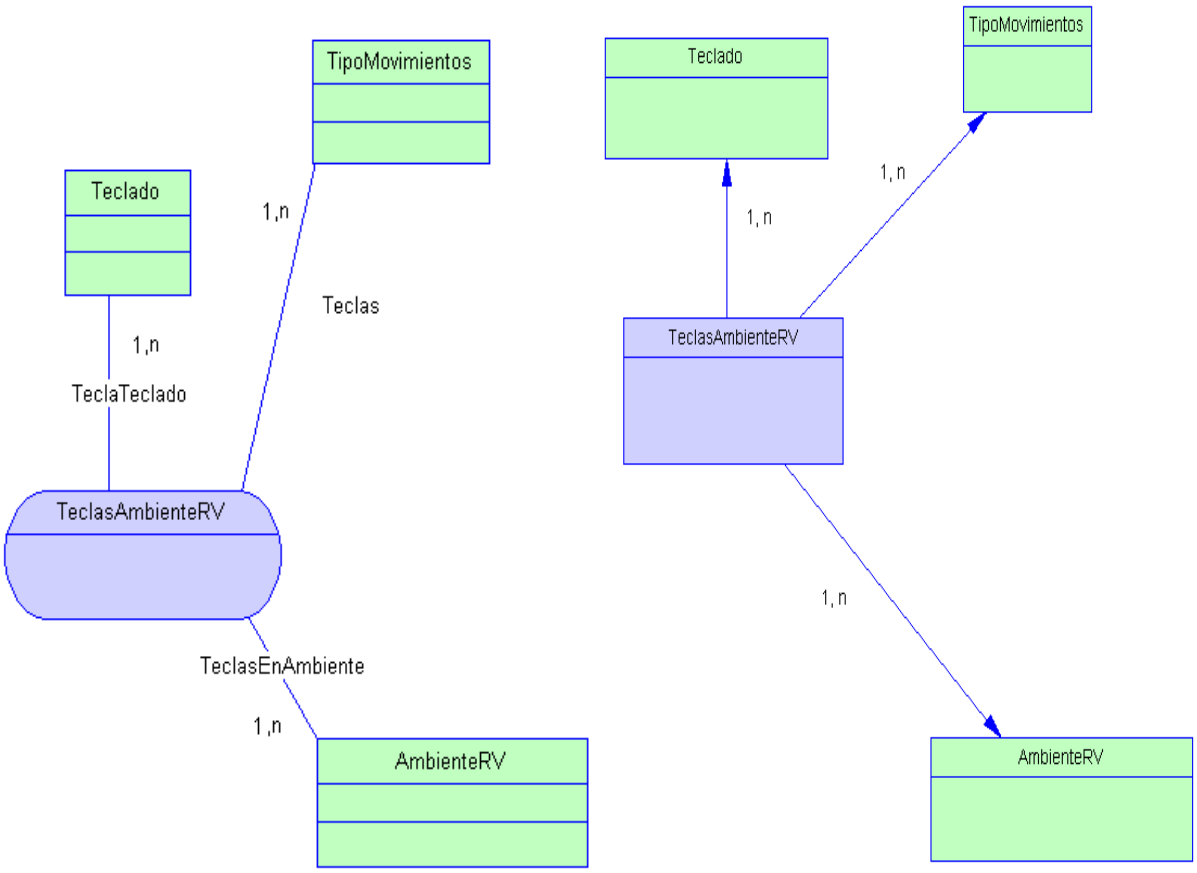

Diagrama Nº 28: Relación Compleja “TeclasAmbienteRV”.

De la misma manera, la relación compleja “MouseAmbienteRV” se transformo insertando una entidad intermedia tal como se muestra en el diagrama:

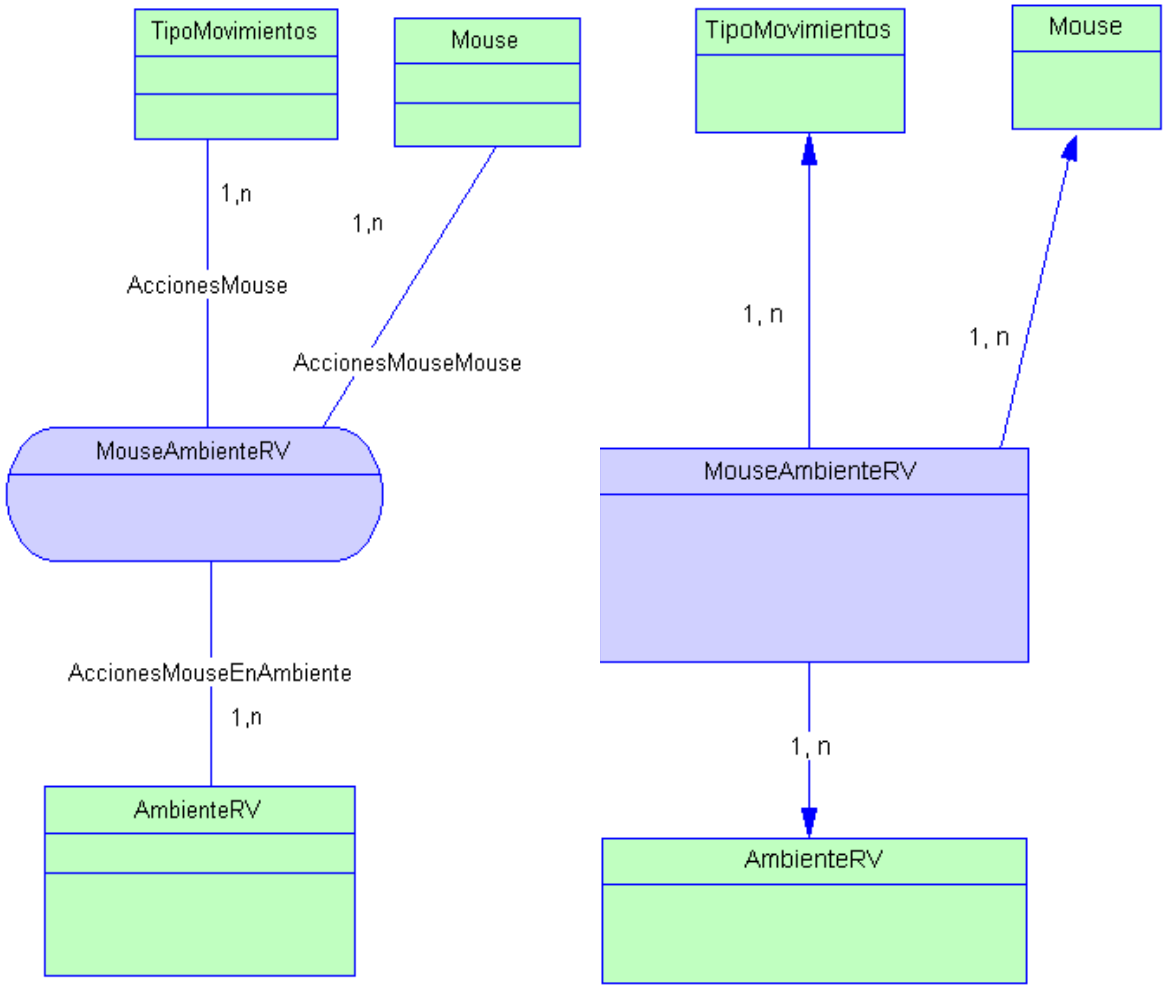

Diagrama Nº 29: Relación Compleja “MouseAmbienteRV”.

Para la relación “PatronesGuanteRV” se insertó una entidad intermedia del mismo nombre tal como se muestra a continuación:

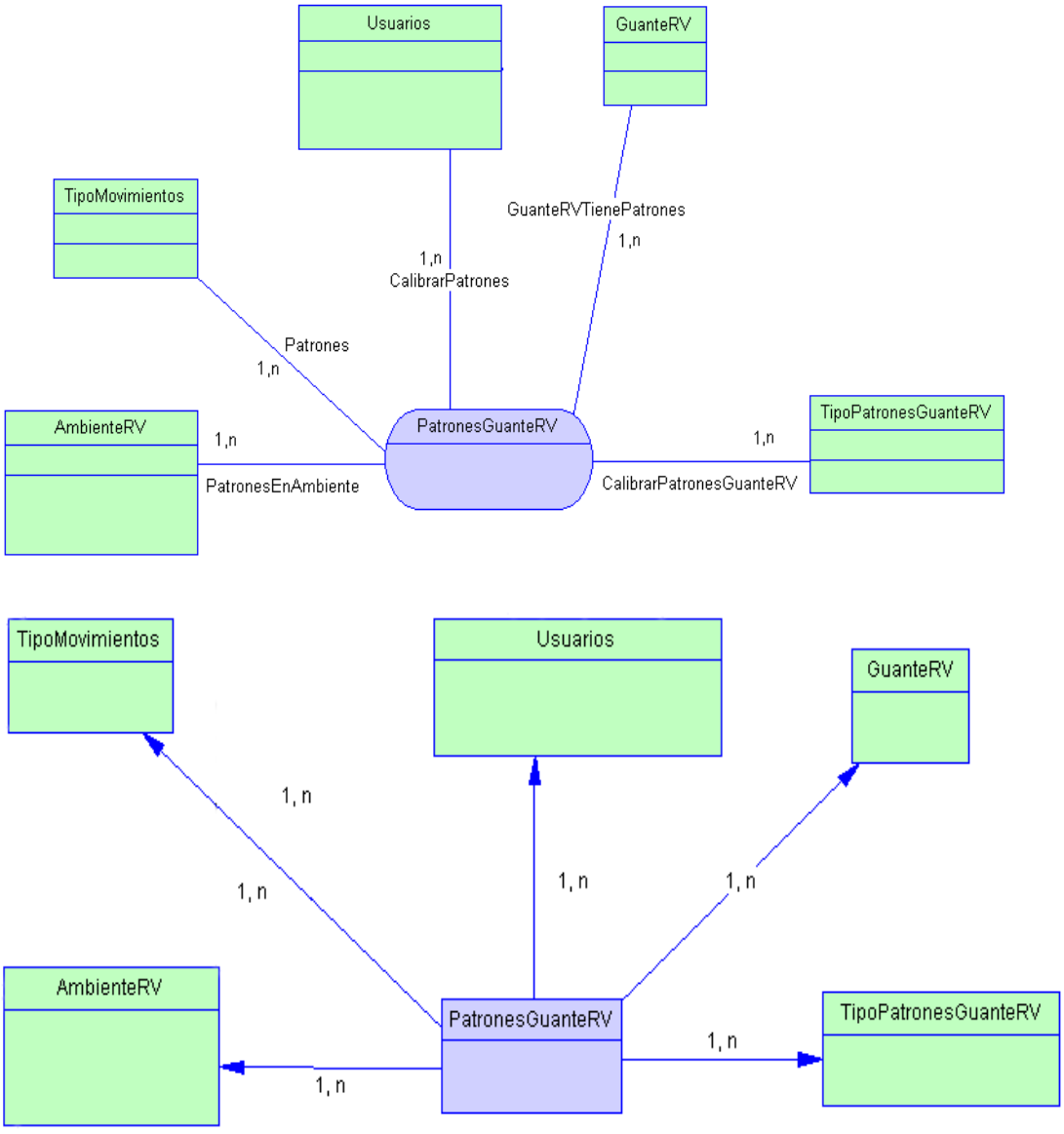

Diagrama Nº 30: Relación Compleja “PatronesGuanteRV”.

La relación “Ejecutar” se cambio por la tabla “Ejecutar”, tal como se muestra a continuación:

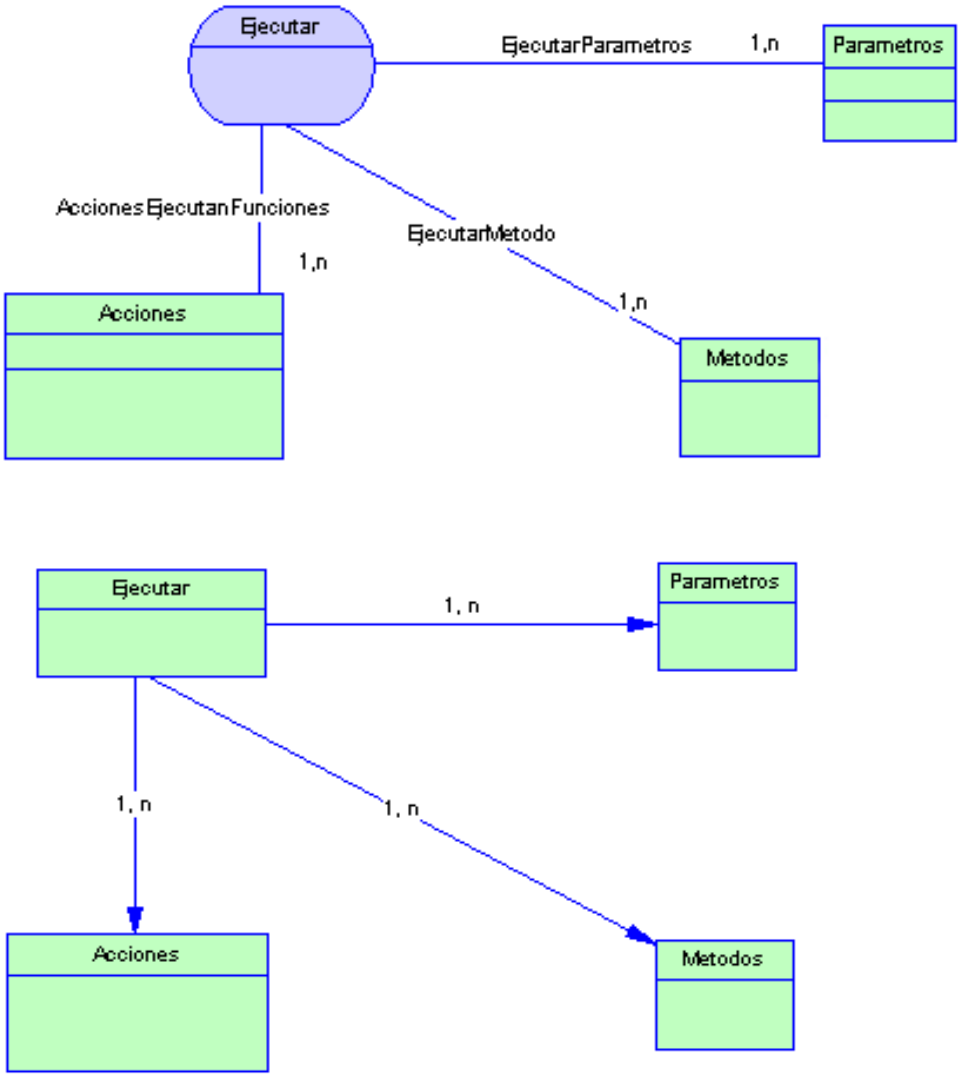

Diagrama Nº 31: Relación Compleja “Ejecutar”.

9.2.1.3 Relaciones “Recursivas”

La relación recursiva “AccionesAgrupanAcciones” se transformó de la siguiente manera:

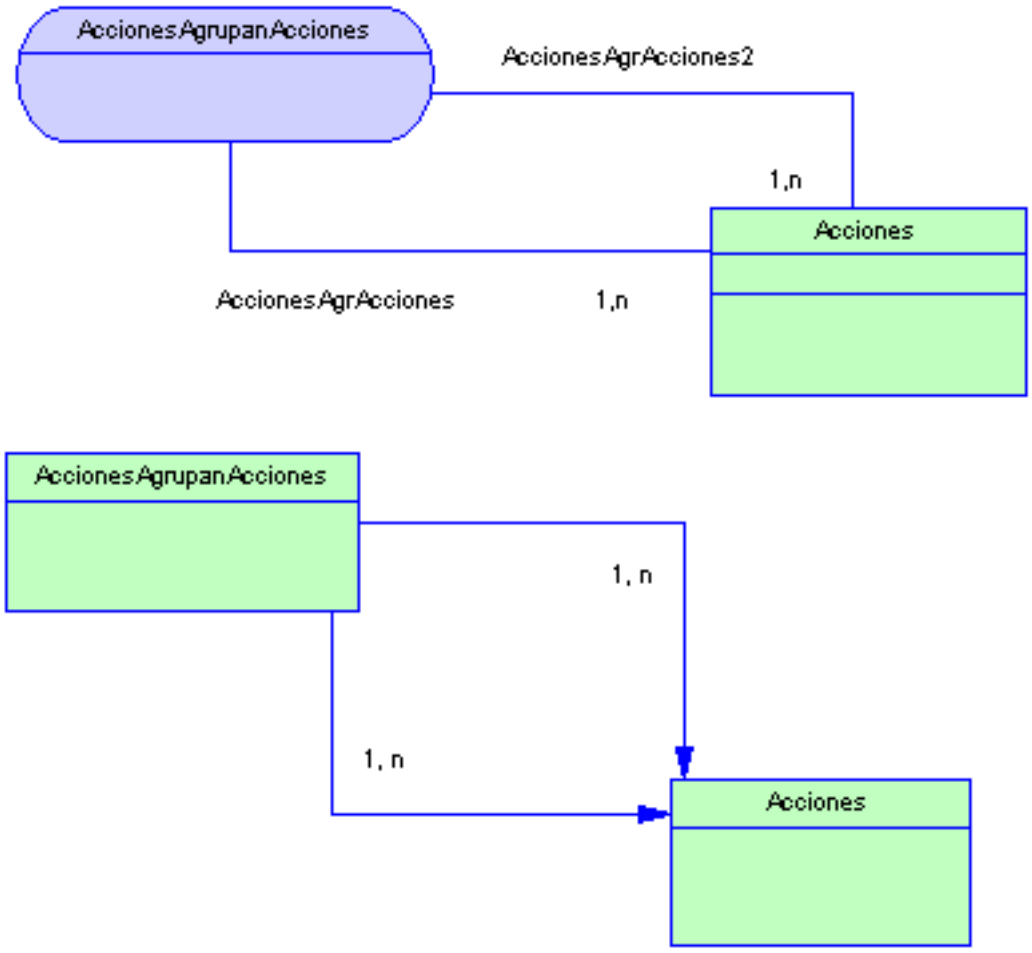

Diagrama Nº 32: Relación Recursiva “AccionesAgrupanAcciones”.

9.2.1.4 Relaciones con “Atributos”

La relación “CalibrarGaunteRV” se transformó cambiándola por la tabla “CalibrarGuanteRV, de la siguiente manera:

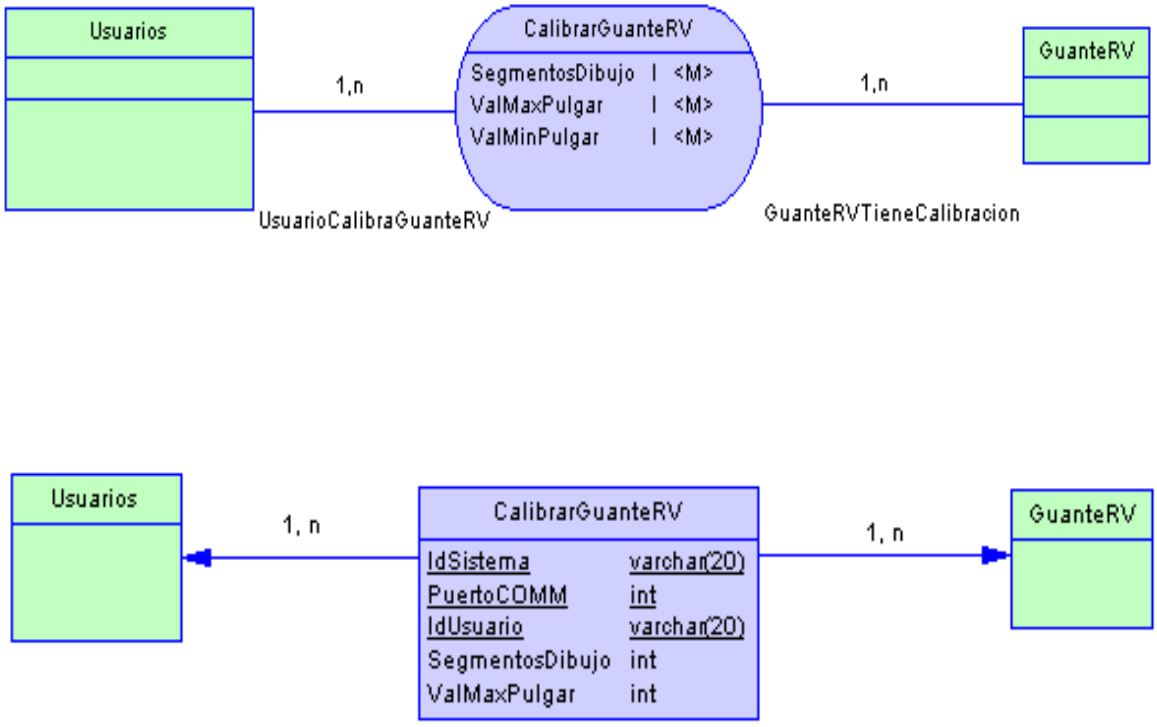

Diagrama Nº 33: Relación con Atributos “CalibrarGuanteRV”.

La relación “SistemaTieneUsuarios” se reemplazó por una tabla del mismo nombre, de la siguiente manera:

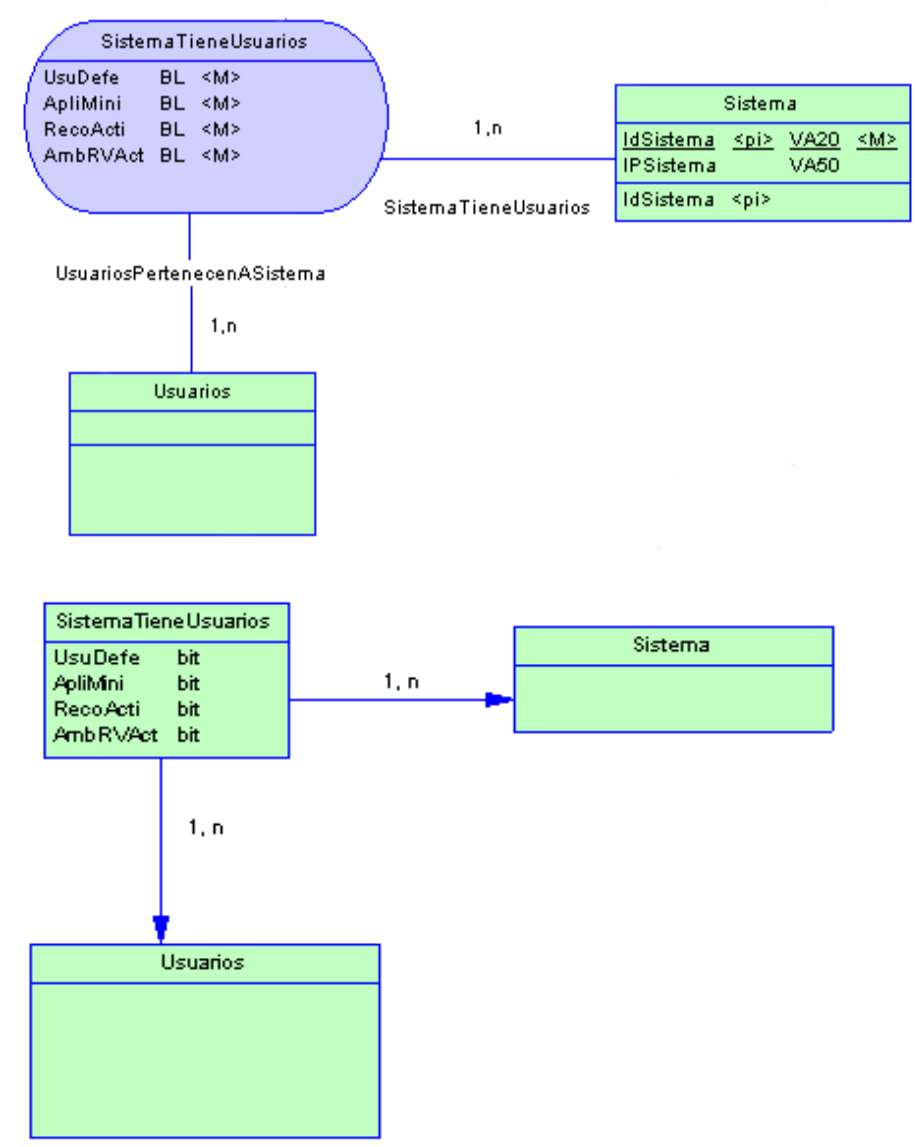

Diagrama Nº 34: Relación con Atributos “SistemaTieneUsuarios”.

9.2.1.5 Optimización de Relaciones

Para mejorar el desempeño de algunas consultas e incrementar la integridad referencial de datos del modelo, se realizaron las siguientes optimizaciones a las relaciones.

La entidad “CalibrarGuanteRV” anteriormente se relacionaba tanto con la tabla “Usuarios” como con “Sistema”, por lo que se reemplazó la relación por una del mismo tipo hacia “SistemaTieneUsuarios”. Con esto se mejora la integridad de datos al exigir que la calibración pertenezca a un usuario registrado en algún sistema, tal como se muestra a continuación.

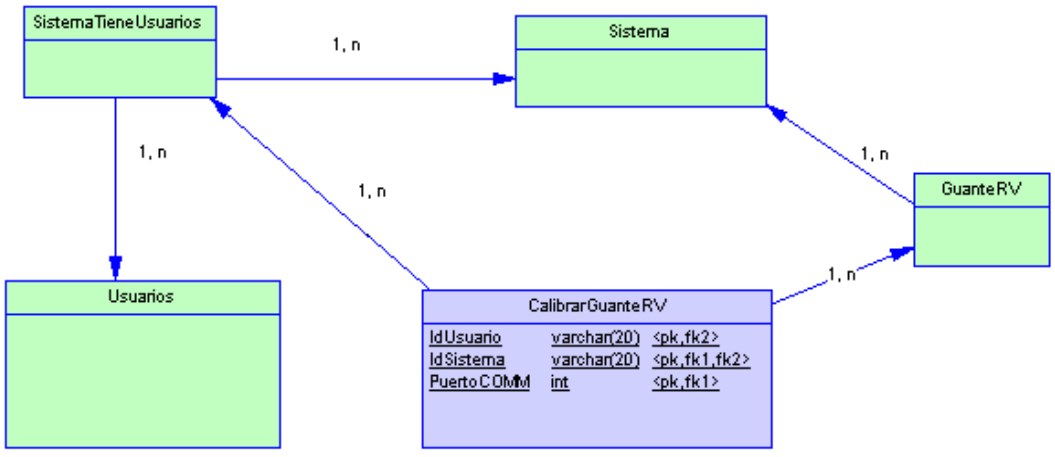

Diagrama Nº 35: Optimización relación “CalibrarGuanteRV”.

La entidad “PatronesGuanteRV” anteriormente se relacionaba con la tabla “Usuarios” pero para mejorar las consultas relacionadas con los datos de calibración de patrones de un usuario en particular, se cambió por una relación hacia la tabla “CalibrarGuante”, la cual ya incluye el identificador de usuario, tal como lo muestra el diagrama:

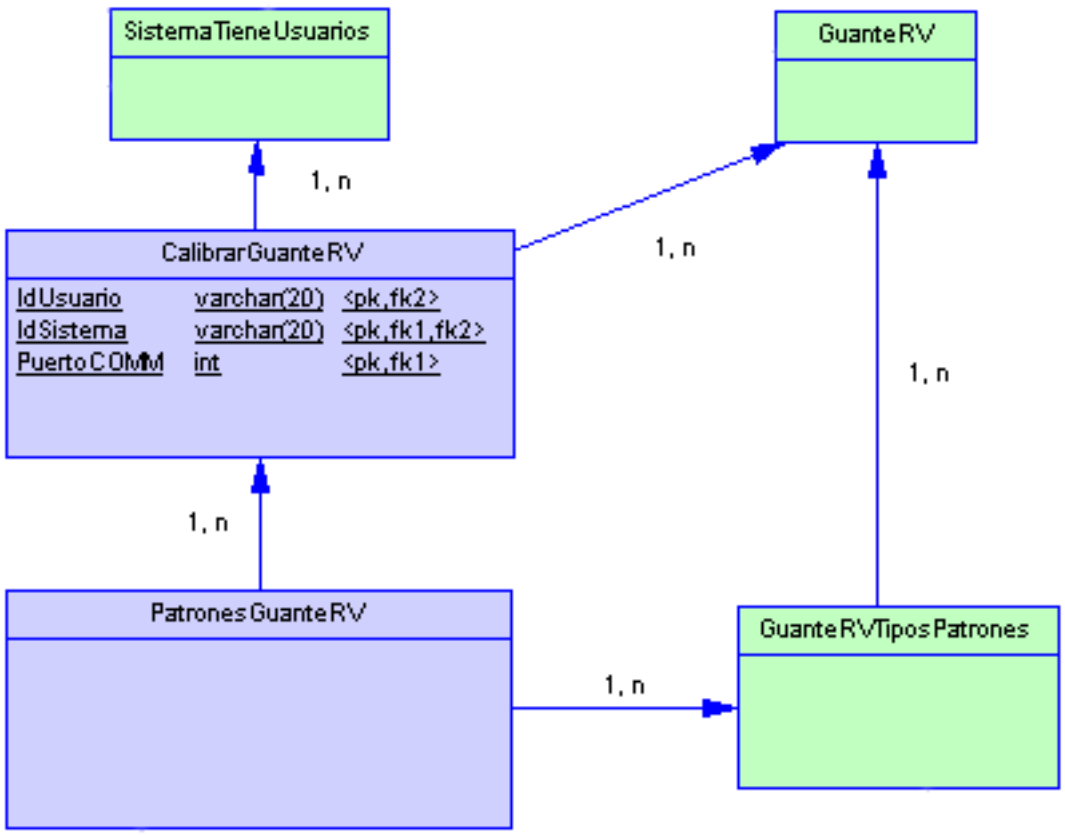

Diagrama Nº 36: Optimización “PatronesGuanteRV”.

Debido a que la entidad “DatosBusLPT” anteriormente se relacionaba tanto con “TipoBuses” como con “PuertosLPT”, cambiando dichas relaciones por una desde “BusesLPT”, se mejora el rendimiento de las consultas ya que “BusesLPT” ya posee tanto la clave del puerto LPT como el identificador del tipo de bus, tal como se muestra a continuación:

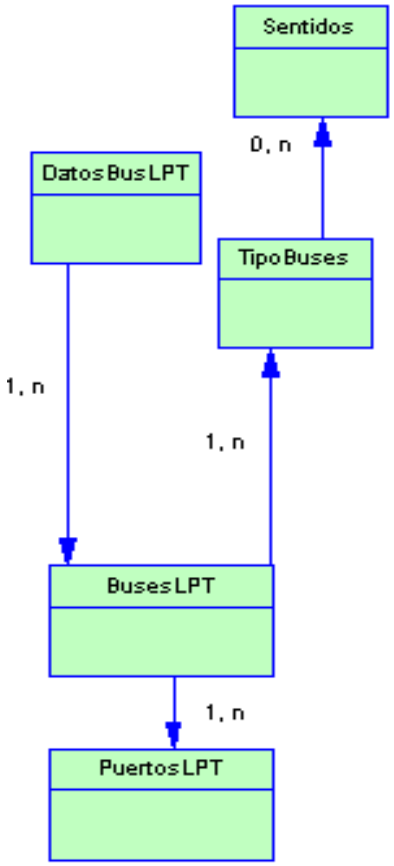

Diagrama Nº 37: Optimización “DatosBuseLPT”.

“Dispositivos” originalmente poseía relaciones tanto con “Sistema” como con “Usuarios”. Cambiando dichas relaciones por una desde “SistemaTieneUsuarios” se mejora el rendimiento de las consultas y se mejora la integridad de datos, al exigir que los dispositivos sean definidos por usuarios válidos en algún sistema, esto se muestra en el siguiente diagrama:

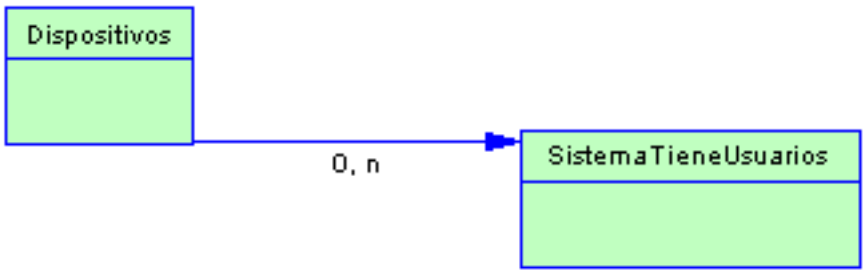

Diagrama Nº 38: Optimización “Dispositivos”.

NOTA:

En algunos de los diagramas de relaciones, se incluyeron los atributos, solo a manera de diferenciar aquellas modificaciones en relaciones normales de aquellas de tipo “Relación con Atributos”.

9.2.2 Derivar relaciones del modelo de datos lógico

A continuación se presenta la derivación de las relaciones del modelo en lenguaje DBDL (“Database Definition Language” o Lenguaje de definición de base de datos):

Acciones (IdSistema, IdUsuario, IdDispositivo, IdAccion, Imagen, Descripcion, GatillaEstado, ReqConfir, Visible)
Primary Key (IdSistema, IdUsuario, IdDispositivo, IdAccion)
Foreign Key (IdSistema, IdUsuario, IdDispositivo) **References** Dispositivos (IdSistema, IdUsuario, IdDispositivo)

AccionesAgrupanAcciones (IdSistema, IdUsuario, IdDispositivo, IdAccion, Acc_IdSistema, Acc_IdUsuario, Acc_IdDispositivo, Acc_IdAccion, NumEjecucion)
Primary Key (IdSistema, IdUsuario, IdDispositivo, IdAccion, Acc_IdSistema, Acc_IdUsuario, Acc_IdDispositivo, Acc_IdAccion)
Foreign Key (IdSistema, IdUsuario, IdDispositivo, IdAccion) **References** Acciones (IdSistema, IdUsuario, IdDispositivo, IdAccion)
Foreign Key (Acc_IdSistema, Acc_IdUsuario, Acc_IdDispositivo, Acc_IdAccion) **References** Acciones (IdSistema, IdUsuario, IdDispositivo, IdAccion)

AmbienteRV (IdUsuario, IdTipoControl, ImagenFondo, TpoMaxReco, SoniArr, SoniAba, SoniIzq, SoniDer, SoniSel, SintComando, ControlTec, ControlRat, ControlGRV)
Primary Key (IdUsuario)
Foreign Key (IdTipoControl) **References** TipoControl (IdTipoControl)
Foreign Key (IdUsuario) **References** Usuarios (IdUsuario)

BusesLPT (IdSistema, DirLPT, IdTipoBus)
Primary Key (IdSistema, DirLPT, IdTipoBus)
Foreign Key (DirLPT, IdSistema) **References** PuertosLPT (DirLPT, IdSistema)
Foreign Key (IdTipoBus) **References** TipoBuses (IdTipoBus)

CalibrarGuanteRV (IdUsuario, IdSistema, PuertoCOMM, SegmentosDibujo, ValMaxPulgar, ValMinPulgar, SenPulgar, ValMaxIndice, ValMinIndice, SenIndice, ValMaxMedio, ValMinMedio, SenMedio, ValMaxAnular, ValMinAnular, SenAnular, ValMaxGiro, ValMinGiro, SenGiro, ValMaxInclinacion, ValMinInclinacion, SenInclinacion, ActDibPulgar, ActDibIndice, ActDibMedio, ActDibAnular, ActDibGiro, ActDibInclinacion)
Primary Key (IdSistema, PuertoCOMM, IdUsuario)
Foreign Key (IdSistema, PuertoCOMM) **References** GuanteRV (IdSistema, PuertoCOMM)
Foreign Key (IdSistema, IdUsuario) **References** SistemaTieneUsuarios (IdSistema, IdUsuario)

ClasesObj (IdSistema, IdClaseObj, Descripción)
Primary Key (IdClaseObj, IdSistema)
Foreign Key (IdSistema) **References** Sistema (IdSistema)

Comandos (IdSistema, PuertoSocket, IdDispositivoIR, IdRemoto, IdComando, IdParametro)
Primary Key (IdComando, IdRemoto, PuertoSocket, IdSistema, IdDispositivoIR)
Foreign Key (IdRemoto, PuertoSocket, IdSistema, IdDispositivoIR)
References Remotos (IdRemoto, PuertoSocket, IdSistema, IdDispositivoIR)
Foreign Key (IdParametro) **References** Parametros (IdParametro)

DatosBusLPT (IdSistema, DirLPT, IdTipoBus, Dato, IdParametro)
Primary Key (IdSistema, DirLPT, IdTipoBus, Dato)
Foreign Key (IdSistema, DirLPT, IdTipoBus) **References** BusesLPT (IdSistema, DirLPT, IdTipoBus)
Foreign Key (IdParametro) **References** Parametros (IdParametro)

DispEMMotoresDC (IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValorRetActual, MensRet)
Primary Key (IdSistema, IdUsuario, IdDispositivo)
Foreign Key (IdSistema, IdUsuario, IdDispositivo) **References** DispElecMeca (IdSistema, IdUsuario, IdDispositivo)

DispEMMotoresPAP (IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValorRetActual, MensRet)
Primary Key (IdSistema, IdUsuario, IdDispositivo)
Foreign Key (IdSistema, IdUsuario, IdDispositivo) **References** DispElecMeca (IdSistema, IdUsuario, IdDispositivo)

DispEMSolenoides (IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, ValRetEst1, MensRetEst1, ValRetEst2, MensRetEst2)
Primary Key (IdSistema, IdUsuario, IdDispositivo)
Foreign Key (IdSistema, IdUsuario, IdDispositivo) **References** DispElecMeca (IdSistema, IdUsuario, IdDispositivo)

DispElecMeca (IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta)
Primary Key (IdSistema, IdUsuario, IdDispositivo)
Foreign Key (IdSistema, IdUsuario, IdDispositivo) **References** Dispositivos (IdSistema, IdUsuario, IdDispositivo)

Dispositivos (IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible)
Primary Key (IdSistema, IdUsuario, IdDispositivo)
Foreign Key (IdSistema, IdUsuario) **References** SistemaTieneUsuarios (IdSistema, IdUsuario)
Foreign Key (IdHabitacion) **References** Habitaciones (IdHabitacion)

DispositivosIR (IdSistema, PuertoSocket, IdDispositivoIR, IdHabitacion)
Primary Key (PuertoSocket, IdSistema, IdDispositivoIR)
Foreign Key (PuertoSocket, IdSistema) **References** ServDispositivosIR (PuertoSocket, IdSistema)
Foreign Key (IdHabitacion) **References** Habitaciones (IdHabitacion)

Ejecutables (IdSistema, IdEjecutable, Ruta, Descripcion, IdParametro)
Primary Key (IdEjecutable, IdSistema, Ruta)
Foreign Key (IdSistema) **References** Sistema (IdSistema)
Foreign Key (IdParametro) **References** Parametros (IdParametro)

Ejecutar (IdEjecucion, IdSistema, IdUsuario, IdDispositivo, IdAccion, IdClaseObj, IdMetodo, IdParametro, TpoEspera)
Primary Key (IdEjecucion)
Foreign Key (IdSistema, IdUsuario, IdDispositivo, IdAccion) **References** Acciones (IdSistema, IdUsuario, IdDispositivo, IdAccion)
Foreign Key (IdParametro) **References** Parametros (IdParametro)
Foreign Key (IdMetodo, IdClaseObj, IdSistema) **References** Metodos (IdMetodo, IdClaseObj, IdSistema)

GuanteRV (IdSistema, PuertoCOMM, LargoCable)
Primary Key (IdSistema, PuertoCOMM)
Foreign Key (IdSistema) **References** Sistema (IdSistema)

GuanteRVTiposPatrones (IdTipoPatron, IdSistema, PuertoCOMM)
Primary Key (IdTipoPatron, IdSistema, PuertoCOMM)
Foreign Key (IdSistema, PuertoCOMM) **References** GuanteRV (IdSistema, PuertoCOMM)
Foreign Key (IdTipoPatron) **References** TipoPatronesGuanteRV (IdTipoPatron)

Habitaciones (IdHabitacion, Descripción)
Primary Key (IdHabitacion)
Alternate Key (Descripcion)

Metodos (IdSistema, IdClaseObj, IdMetodo, IdTipoParametro, Descripcion, CantPara)
Primary Key (IdMetodo, IdClaseObj, IdSistema)
Foreign Key (IdClaseObj, IdSistema) **References** ClasesObj (IdClaseObj, IdSistema)
Foreign Key (IdTipoParametro) **References** TipoParametros (IdTipoParametro)

Mouse (IdAccionMouse, Descripcion)
Primary Key (IdAccionMouse)
Alternate Key (Descripcion)

MouseAmbienteRV (IdUsuario, IdAccionMouse, NumVecesAccionMouse, IdTipoMovimiento, IdTipoControl)
Primary Key (IdUsuario, IdAccionMouse, NumVecesAccionMouse, IdTipoMovimiento, IdTipoControl)
Foreign Key (IdTipoMovimiento, IdTipoControl) **References** TipoMovimientos (IdTipoMovimiento, IdTipoControl)
Foreign Key (IdUsuario) **References** AmbienteRV (IdUsuario)
Foreign Key (IdAccionMouse) **References** Mouse (IdAccionMouse)

Parametros (IdParametro, IdTipoParametro)
Primary Key (IdParametro)
Foreign Key (IdTipoParametro) **References** TipoParametros (IdTipoParametro)

PatronesAmbienteRV (IdUsuario, IdSistema, PuertoCOMM, IdTipoPatron, NumVecesPatron, IdTipoMovimiento, IdTipoControl)
Primary Key (IdUsuario, IdSistema, PuertoCOMM, IdTipoPatron, NumVecesPatron, IdTipoMovimiento, IdTipoControl)
Foreign Key (IdTipoMovimiento, IdTipoControl) **References** TipoMovimientos (IdTipoMovimiento, IdTipoControl)
Foreign Key (IdUsuario) **References** AmbienteRV (IdUsuario)
Foreign Key (IdUsuario, IdTipoPatron, IdSistema, PuertoCOMM) **References** PatronesGuanteRV (IdUsuario, IdTipoPatron, IdSistema, PuertoCOMM)

PatronesGuanteRV (IdUsuario, IdSistema, PuertoCOMM, IdTipoPatron, RangoPatron, PorcentajeError)
Primary Key (IdUsuario, IdTipoPatron, IdSistema, PuertoCOMM)
Foreign Key (IdTipoPatron, IdSistema, PuertoCOMM) **References** GuanteRVTiposPatrones (IdTipoPatron, IdSistema, PuertoCOMM)
Foreign Key (IdSistema, PuertoCOMM, IdUsuario) **References** CalibrarGuanteRV (IdSistema, PuertoCOMM, IdUsuario)

PuertosLPT (IdSistema, DirLPT)
Primary Key (DirLPT, IdSistema)
Foreign Key (IdSistema) **References** Sistema (IdSistema)

Remotos (IdSistema, PuertoSocket, IdDispositivoIR, IdRemoto)
Primary Key (IdRemoto, PuertoSocket, IdSistema, IdDispositivoIR)
Foreign Key (PuertoSocket, IdSistema, IdDispositivoIR) **References** DispositivosIR (PuertoSocket, IdSistema, IdDispositivoIR)

Sentidos (IdSentido, Descripción)
Primary Key (IdSentido)
Alternate Key (Descripcion)

ServDispositivosIR (IdSistema, PuertoSocket, DispDefecto)
Primary Key (PuertoSocket, IdSistema)
Foreign Key (IdSistema) **References** Sistema (IdSistema)

SintRecoVoz (IdUsuario, VozSexo, VozVel, RepPalabra, RepConfir, PalConfir)
Primary Key (IdUsuario)
Foreign Key (IdUsuario) **References** Usuarios (IdUsuario)

Sistema (IdSistema, IPSistema)
Primary Key (IdSistema)

SistemaTieneUsuarios (IdSistema, IdUsuario, UsuDefe, ApliMini, RecoActi, AmbRVAct)
Primary Key (IdSistema, IdUsuario)
Foreign Key (IdSistema) **References** Sistema (IdSistema)
Foreign Key (IdUsuario) **References** Usuarios (IdUsuario)

Teclado (IdTecla, Descripción)
Primary Key (IdTecla)
Alternate Key (Descripcion)

TeclasAmbienteRV (IdUsuario, IdTecla, NumVecesTecla, IdTipoMovimiento, IdTipoControl)
Primary Key (IdUsuario, IdTecla, NumVecesTecla, IdTipoMovimiento, IdTipoControl)
Foreign Key (IdTipoMovimiento, IdTipoControl) **References** TipoMovimientos (IdTipoMovimiento, IdTipoControl)
Foreign Key (IdTecla) **References** Teclado (IdTecla)
Foreign Key (IdUsuario) **References** AmbienteRV (IdUsuario)

TipoBuses (IdTipoBus, IdSentido, Descripción)
Primary Key (IdTipoBus)
Alternate Key (Descripcion)
Foreign Key (IdSentido) **References** Sentidos (IdSentido)

TipoControl (IdTipoControl, Descripción)
Primary Key (IdTipoControl)
Alternate Key (Descripcion)

TipoMovimientos (IdTipoMovimiento, IdTipoControl, Descripcion)
Primary Key (IdTipoMovimiento, IdTipoControl)
Alternate Key (Descripcion)
Foreign Key (IdTipoControl) **References** TipoControl (IdTipoControl)

TipoParametros (IdTipoParametro, Descripción)
Primary Key (IdTipoParametro)
Alternate Key (Descripcion)

TipoPatronesGuanteRV (IdTipoPatron, SenMovPatron, Descripción)
Primary Key (IdTipoPatron)
Alternate Key (Descripcion)

Usuarios (IdUsuario, NomUsuario, ApeUsuario)
Primary Key (IdUsuario)

9.2.3 Validar el modelo utilizando normalización

Para validar el presente modelo de datos, se realizaron la comprobación de las siguientes formas normales:

9.2.3.1 Primera Forma Normal (1FN)

Para que el modelo lógico de la base de datos se encuentre en primera forma normal, no deben existir grupos de datos repetitivos o multivalorados, y de haberlos se deben separar para formar sus propias relaciones.

Del análisis del modelo presentado se desprende que no posee este tipo de características, y que por lo tanto se encuentra en primera forma normal.

9.2.3.2 Segunda Forma Normal (2FN)

La condición para que una tabla se encuentre en segunda forma normal, es que primero cumpla con la primera forma normal y que posteriormente cada atributo que no forma parte de la clave primaria en una tabla, sea totalmente dependiente de la clave primaria de dicha tabla.

Para esto, en el presente modelo, se analizaron las tablas que poseen claves compuestas, ya que se asume que las tablas con claves primarias simples ya se encuentran en segunda forma normal.

TipoMovimientos(IdTipoMovimiento, IdTipoControlo, Descripcion)
Entonces: Descripcion. Depende completamente de la clave primaria (IdTipoMovimiento, IdTipoControl) y no solo de una parte de ella.

CalibrarGuanteRV(IdUsuario, IdSistema, PuertoCOMM, SegmentosDibujo, ValMaxPulgar, ValMinPulgar, SenPulgar, ValMaxIndice, ValMinIndice, SenIndice, ValMaxMedio, ValMinMedio, SenMedio, ValMaxAnular, ValMinAnular, SenAnular, ValMaxGiro, ValMinGiro, SenGiro, ValMaxInclinacion, ValMinInclinacion, SenInclinacion, ActDibPulgar, ActDibIndice, ActDibMedio, ActDibAnular, ActDibGiro, ActDibInclinacion)
Entonces: SegmentosDibujo, ValMaxPulgar, ValMinPulgar, SenPulgar, ValMaxIndice, ValMinIndice, SenIndice, ValMaxMedio, ValMinMedio, SenMedio, ValMaxAnular, ValMinAnular, SenAnular, ValMaxGiro, ValMinGiro, SenGiro, ValMaxInclinacion, ValMinInclinacion, SenInclinacion, ActDibPulgar, ActDibIndice, ActDibMedio, ActDibAnular, ActDibGiro, ActDibInclinacion. Cada uno de ellos depende completamente de la clave primaria (IdUsuario, IdSistema, PuertoCOMM) y no solo de una parte de ella.

GuanteRV(IdSistema, PuertoCOMM, LargoCable)
Entonces: LargoCable. Depende completamente de la clave primaria (IdSistema, PuertoCOMM) y no solo de una parte de ella.

PatronesGuanteRV(IdUsuario, IdSistema, PuertoCOMM, IdTipoPatron, RangoPatron, PorcentajeError)
Entonces: RangoPatron, PorcentajeError. Cada uno de ellos depende completamente de la clave primaria (IdUsuario, IdSistema, PuertoCOMM, IdTipoPatron) y no solo de una parte de ella.

ServDispositivosIR(IdSistema, PuertoSocket, DispDefecto)
Entonces: DispDefecto. Depende completamente de la clave primaria (IdSistema, PuertoSocket) y no solo de una parte de ella.

DispositivosIR(IdSistema, PuertoSocket, IdDispositivoIR, IdHabitacion)
Entonces: IdHabitacion. Depende completamente de la clave primaria (IdSistema, PuertoSocket, IdDispositivoIR) y no solo de una parte de ella.

Comandos(IdSistema, PuertoSocket, IdDispositivoIR, IdRemoto, IdComando, IdParametro)
Entonces: IdParametro. Depende completamente de la clave primaria (IdSistema, PuertoSocket, IdDispositivoIR, IdRemoto, IdComando) y no solo de una parte de ella.

Ejecutables(IdSistema, IdEjecutables, Ruta, Descripcion, IdParametro)
Entonces: Descripcion, IdParametro. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdEjecutables, Ruta) y no solo de una parte de ella.

DatosBusLPT(IdSistema, DirLPT, IdTipoBus, Dato, IdParametro)
Entonces: IdParametro. Depende completamente de la clave primaria (IdSistema, DirLPT, IdTipoBus, Dato) y no solo de una parte de ella.

SistemaTieneUsuarios(IdSistema, IdUsuario, UsuDefe, ApliMini, RecoActi, AmbRVAct)
Entonces: UsuDefe, ApliMini, RecoActi, AmbRVAct. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario) y no solo de una parte de ella.

ClasesObj(IdSistema, IdClaseObj, Descripcion)
Entonces: Descripcion. Depende completamente de la clave primaria (IdSistema, IdClaseObj) y no solo de una parte de ella.

Metodos(IdSistema, IdClaseObj, IdMetodo, IdTipoParametro, Descripción, CantPara)
Entonces: IdTipoParametro, Descripción, CantPara. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdClaseObj, IdMetodo) y no solo de una parte de ella.

DispEMMotoresDC(IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValorRetActual, MensRet)
Entonces: Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValorRetActual, MensRet. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo) y no solo de una parte de ella.

.
DispEMMotoresPAP(IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValorRetActual, MensRet)
Entonces: Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValorRetActual, MensRet. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo) y no solo de una parte de ella.

DispEMSolenoides(IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, ValRetEst1, MensRetEst1, ValRetEst2, MensRetEst2)
Entonces: Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, ValRetEst1, MensRetEst1, ValRetEst2, MensRetEst2. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo) y no solo de una parte de ella.

DispElecMeca(IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta)
Entonces: Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo) y no solo de una parte de ella.

Dispositivos(IdSistema, IdUsuario, IdDispositivo, Imagen, IdHabitacion, Descripcion, EstadoActual, Visible)
Entonces: Imagen, IdHabitacion, Descripcion, EstadoActual, Visible. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo) y no solo de una parte de ella.

AccionesAgrupanAcciones(IdSistema, IdUsuario, IdDispositivo, IdAccion, Acc_IdSistema, Acc_IdUsuario, Acc_IdDispositivo, Acc_IdAccion, NumEjecucion)
Entonces: NumEjecucion depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo, IdAccion, Acc_IdSistema, Acc_IdUsuario, Acc_IdDispositivo, Acc_IdAccion) y no solo de una parte de ella.

Acciones(IdSistema, IdUsuario, IdDispositivo, IdAccion, Imagen, Descripcion, GatillaEstado, ReqConfir, Visible)
Entonces: Imagen, Descripcion, GatillaEstado, ReqConfir, Visible. Cada uno de ellos depende completamente de la clave primaria (IdSistema, IdUsuario, IdDispositivo, IdAccion) y no solo de una parte de ella.

De acuerdo a la definición de 2FN y al análisis de las tablas que poseen claves primarias compuestas, podemos decir que el modelo se encuentra en segunda forma normal.

9.2.3.3 Tercera Forma Normal (3FN)

Según la definición de 3FN, las tablas de un modelo se encuentra en tercera forma normal si y solo si ya se encuentra en segunda forma normal y todos los atributos no pertenecientes a la clave primaria dependen de forma no transitiva de dicha clave. Ademas, ningun atributo no clave puede depender aun en forma indirecta de otro atributo no clave.

Dependencias funcionales:

AccionesAgrupanAcciones
IdSistema, IdUsuario, IdDispositivo, IdAccion, Acc_IdSistema, Acc_IdUsuario, Acc_IdDispositivo, Acc_IdAccion → NumEjecucion.

Acciones
IdSistema, IdUsuario, IdDispositivo, IdAccion → Imagen, Descripcion, GatillaEstado, RequConfir, Visible.

Dispositivos
IdSistema, IdUsuario, IdDispositivo → Imagen, IdHabitacion, Descripcion, EstadoActual, Visible.

DispElecMeca
IdSistema, IdUsuario, IdDispositivo → Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta.

DispEMMotoresDC
IdSistema, IdUsuario, IdDispositivo → Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValRetActual, MensRet.

DispEMMotoresPAP
IdSistema, IdUsuario, IdDispositivo → Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, MaxValRet, ValRetActual, MensRet.

DispEMSolenoides
IdSistema, IdUsuario, IdDispositivo → Imagen, IdHabitacion, Descripcion, EstadoActual, Visible, PosAbsoluta, ValRetEst1, MensRetEst1, ValRetEst2, MensRetEst2.

SintRecoVoz
IdUsuario → VozSexo, VozVel, RepPalabra, RepConfir, PalConfir.

Ejecutar
IdEjecucion → IdSistema, IdUsuario, IdDispositivo, IdAccion, IdClaseObj, IdMetodo, IdParametro, TpoEspera.

TipoParametros
IdTipoParametro → Descripcion.
Descripcion → IdTipoParametro.

Metodos
IdSistema, IdClaseObj, IdMetodo → IdTipoParametro, Descripcion, CantPara.

ClasesObj
IdSistema, IdClaseObj → Descripcion.

SistemaTieneUsuarios
IdSistema, IdUsuario → UsuDefe, ApliMini, RecoActi, AmbRVAct.

Habitaciones
IdHabitacion → Descripcion.
Descripcion → IdHabitacion.

Sentidos
IdSentido → Descripcion.
Descripción → IdSentido.

Parametros
IdParametro → IdTipoParametro.

Comandos
IdSistema, PuertoSocket, IdDispositivoIR, IdRemoto, IdComando → IdParametro.

Ejecutables
IdSistema, IdEjecutable, Ruta → Descripcion, IdParametro.

DatosBusLPT
IdSistema, DirLPT, IdTipoBus, Dato → IdParametro.

DispositivosIR
IdSistema, PuertoSocket, IdDispositivoIR → IdHabitacion.

TipoBuses
IdTipoBus → IdSentido, Descripcion.
Descripción → IdTipoBus.

ServDispositivosIR
IdSistema, PuertoSocket → DispDefecto.

Sistema
IdSistema → IPSistema.

CalibrarGuanteRV
IdUsuario, IdSistema, PuertoCOMM → SegmentosDibujo, ValMaxPulgar, ValMinPulgar, SenPulgar, ValMaxIndice, ValMinIndice, SenIndice, ValMaxMedio, ValMinMedio, SenMedio, ValMaxAnular, ValMinAnular, SenAnular, ValMaxGiro, ValMinGiro, SenGiro, ValMaxInclinacion, ValMinInclinacion, SenInclinacion, ActDibPulgar, ActDibIndice, ActDibMedio, ActDibAnular, ActDibGiro, ActDibInclinacion.

GuanteRV
IdSistema, PuertoCOMM → LargoCable.

PatronesGuanteRV
IdUsuario, IdSistema, PuertoCOMM, IdTipoPatron → RangoPatron, PorcentajePatron.

TipoPatronesGuanteRV
IdTipoPatron → SenMovPatron, Descripcion.
Descripción → IdTipoPatron.

AmbienteRV
IdUsuario → IdTipoControl, ImagenFondo, TpoMaxReco, SoniArr, SoniAba, SoniIzq, SoniDer, SoniSel, SintComando, ControlTec, ControlRat, ControlGRV.

Mouse
IdAccionMouse → Descripcion.
Descripción → IdAccionMouse.

TipoMovimientos
IdTipoMovimientos, IdTipoControl → Descripcion.
Descripción → IdTipoMovimientos, IdTipoControl.

Usuarios
IdUsuario → NomUsuario, ApeUsuario.

TipoControl
IdTipoControl → Descripcion.
Descripcion → IdTipoControl.

Teclado
IdTecla → Descripcion.
Descripcion → IdTecla

Conforme el análisis anterior y la definición de 3FN, las tablas del modelo presentado para la base de datos se encuentran en tercera forma normal.

9.2.3.4 Forma Normal Boyce-Codd (FNBC)

La definición de la forma normal de Boyce-Codd nos dice que, para que las tablas de un modelo se encuentren en FNBC, todas las determinantes del conjunto de dependencias funcionales de cada tabla, son claves alternas o es clave primaria o forman parte del conjunto de claves candidatas de la tabla.

De acuerdo a la definición anterior y comprobando el modelo presentado, se puede apreciar que el modelo cumple con la forma normal de Boyce-Codd.

9.2.3.5 Otras Formas Normales

Además de las normalizaciones aquí planteadas, existen otras 5 formas normales, estas son:

- Cuarta Forma Normal (4FN).
- Quinta Forma Normal (5FN) o Forma Normal de Proyección-Unión.
- Forma Normal de Proyección-Unión Fuerte.
- Forma Normal de Proyección-Unión Extra Fuerte.
- Forma Normal de Clave de Dominio.

Estos cinco niveles de normalización, no serán considerados, puesto a que son utilizados principalmente cuando se presentan problemas de dependencias múltiples y claves relacionales, los cuales no se exhiben en el presente modelo.

9.2.4 Transacciones de Usuario

Las transacciones definidas para los usuarios y como las resuelve el modelo son las siguientes:

- Configurar y calibrar Guante RV según las necesidades del usuario.

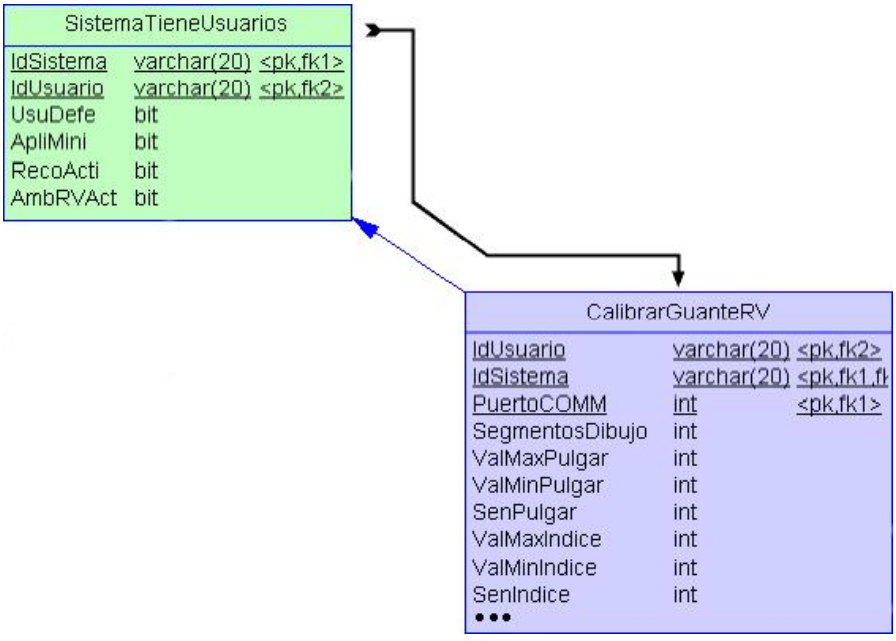

Diagrama Nº 39: Transacción configurar y calibrar GuanteRV.

- Configurar, según las necesidades de usuario, el Ambiente RV con el cual se interactuará con el sistema.

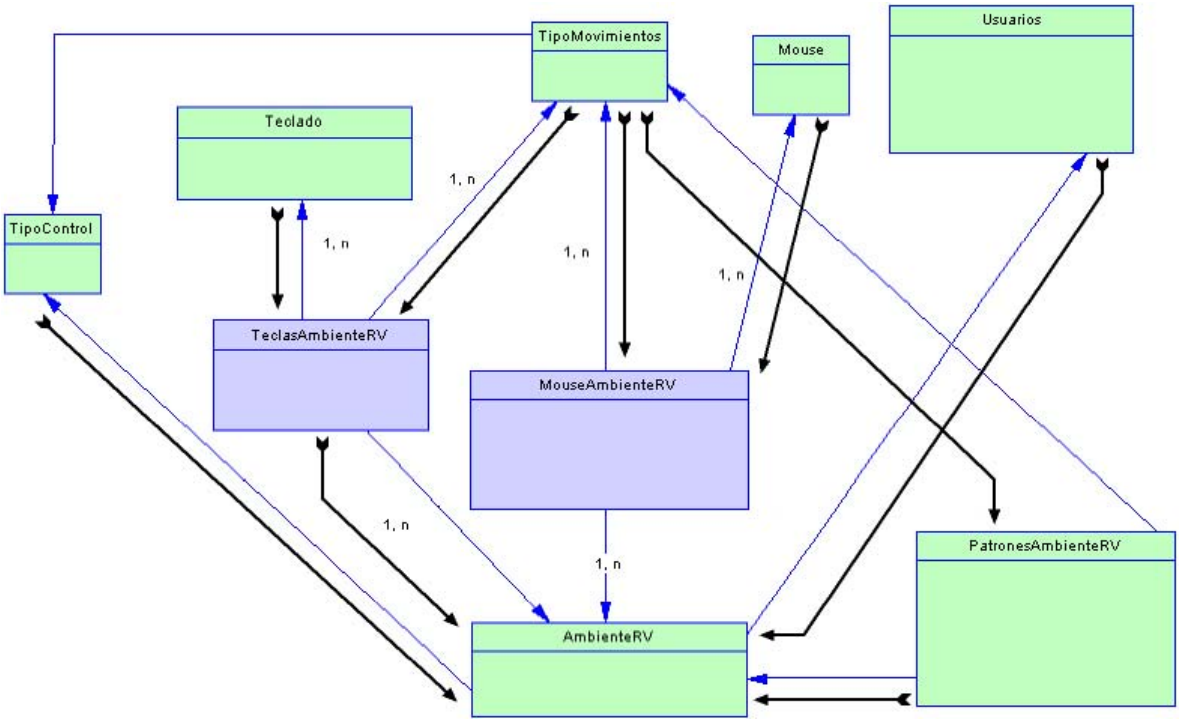

Diagrama Nº 40: Transacción configurar AmbienteRV.

- Configurar, según las necesidades de usuario, el sistema de reconocimiento y síntesis de voz que se utilizará con el sistema.

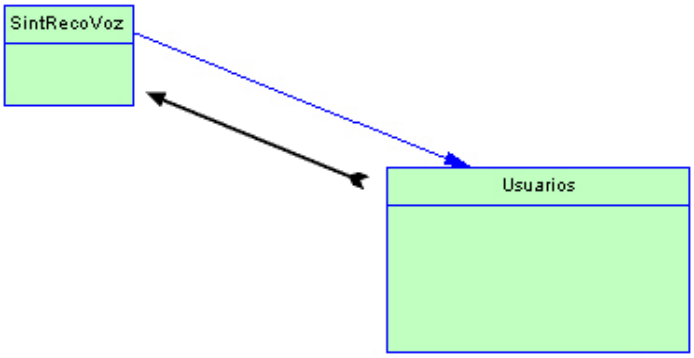

Diagrama Nº 41: Transacción configurar sintetizador y reconocedor de voz.

- Manipular señales infrarrojas previamente almacenadas (TV, VHS, DVD, etc.) para el control de dispositivos de este tipo.

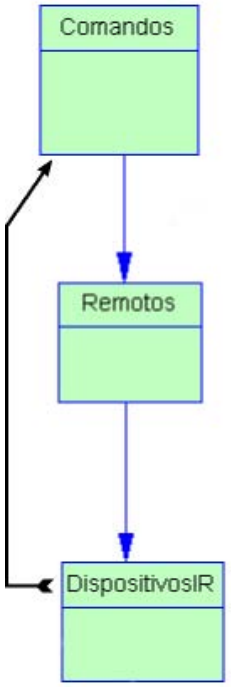

Diagrama Nº 42: Transacción manipular señales IR.

- Configurar el “servidor” y “puerto” en el cual se encuentra el dispositivo “irTrans-Servidor”.

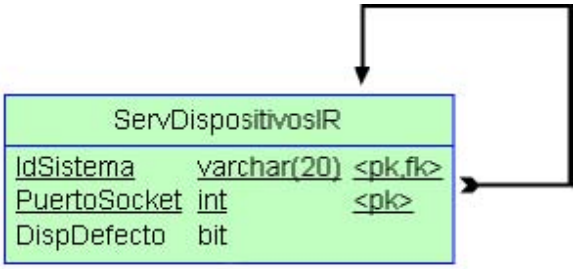

Diagrama Nº 43: Transacción configurar servidor irTrans.

- Monitoreo y control a bajo nivel del puerto paralelo LPT para control de dispositivos eléctrico/mecánico.

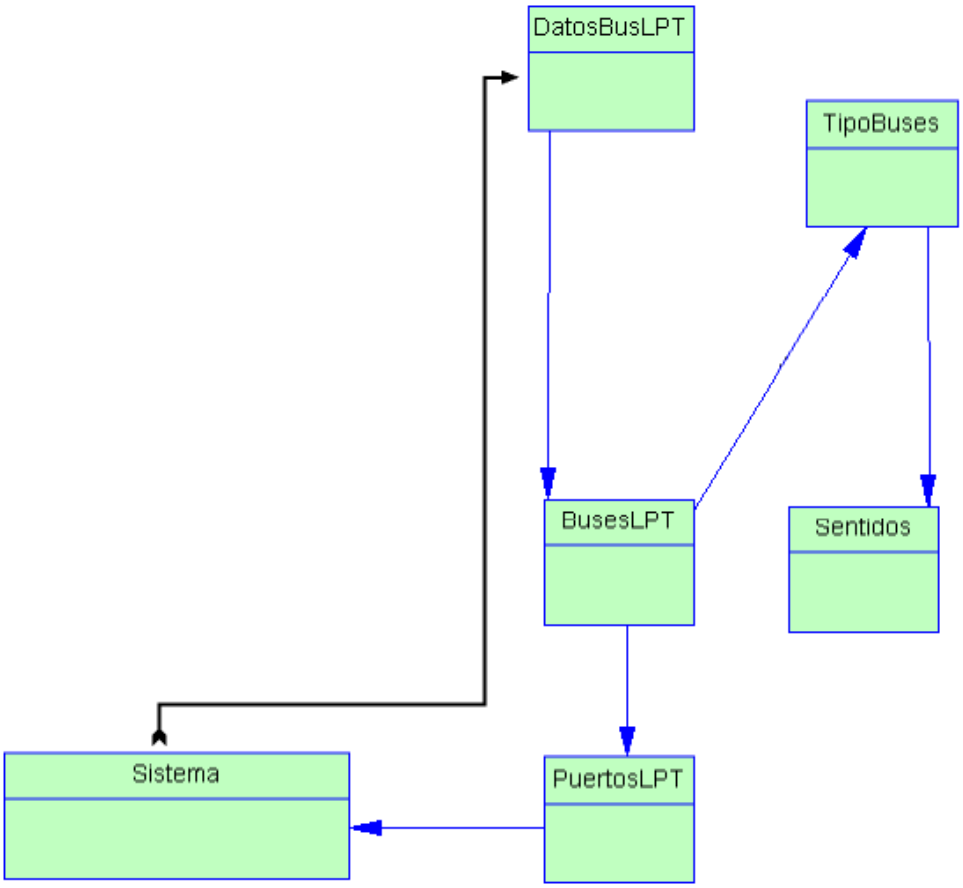

Diagrama Nº 44: Transacción manipulación bytes puerto LPT.

- Identificar dispositivos y sus acciones asociadas, así como su información y configuración respectiva.

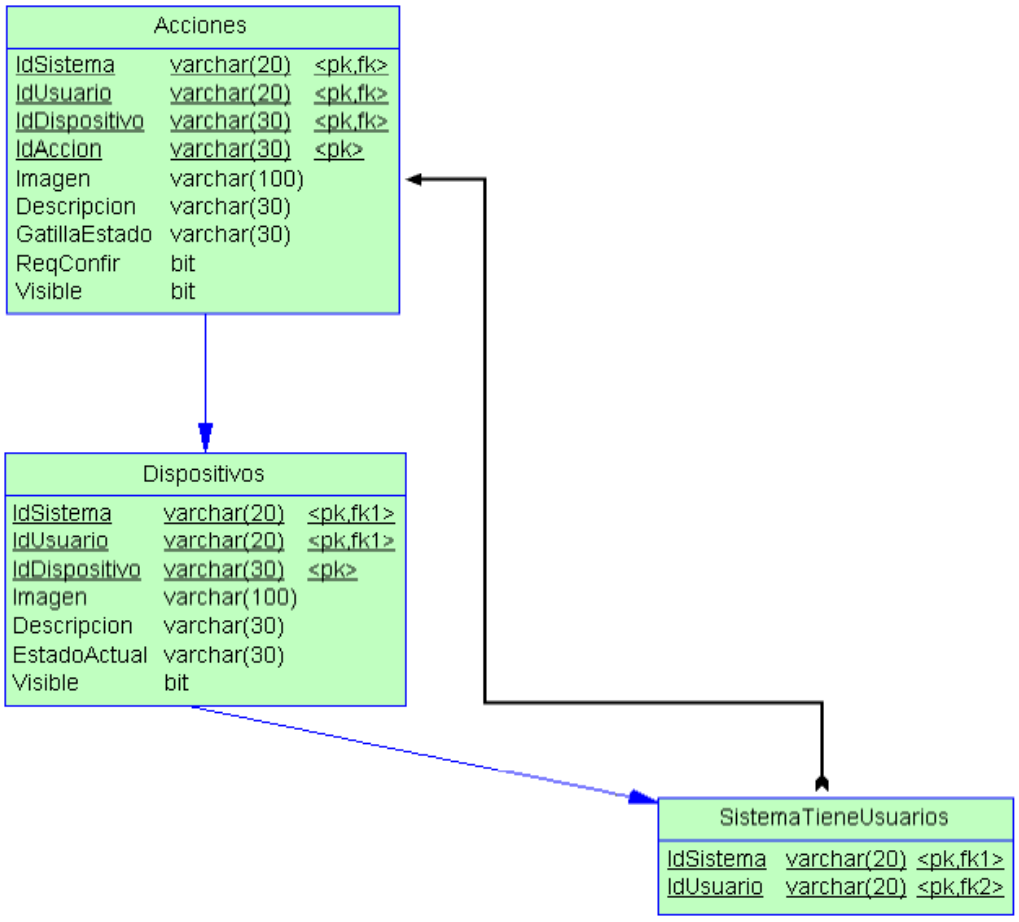

Diagrama Nº 45: Transacción dispositivos y acciones asociadas.

- Ejecutar las acciones según las órdenes dadas por el usuario a través de las interfaces propuestas.

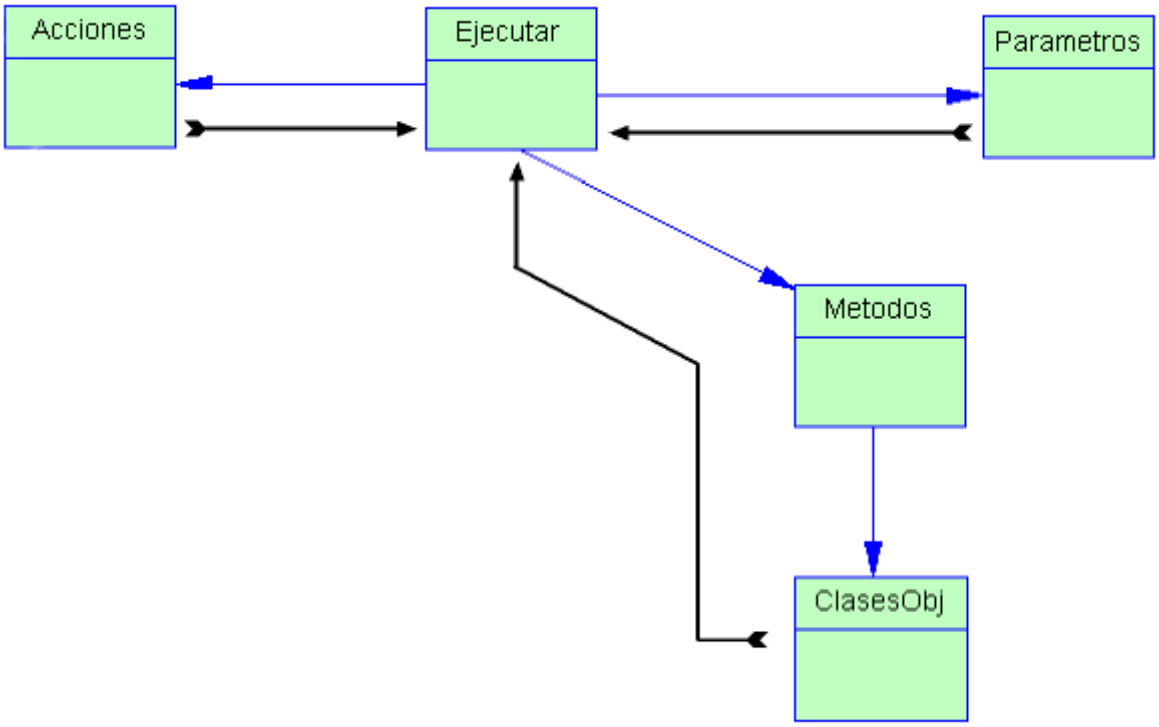

Diagrama Nº 46: Transacción ejecutar acciones.

9.2.5 Diagrama ER final

A continuación se presenta al diagrama Entidad-Relación final:

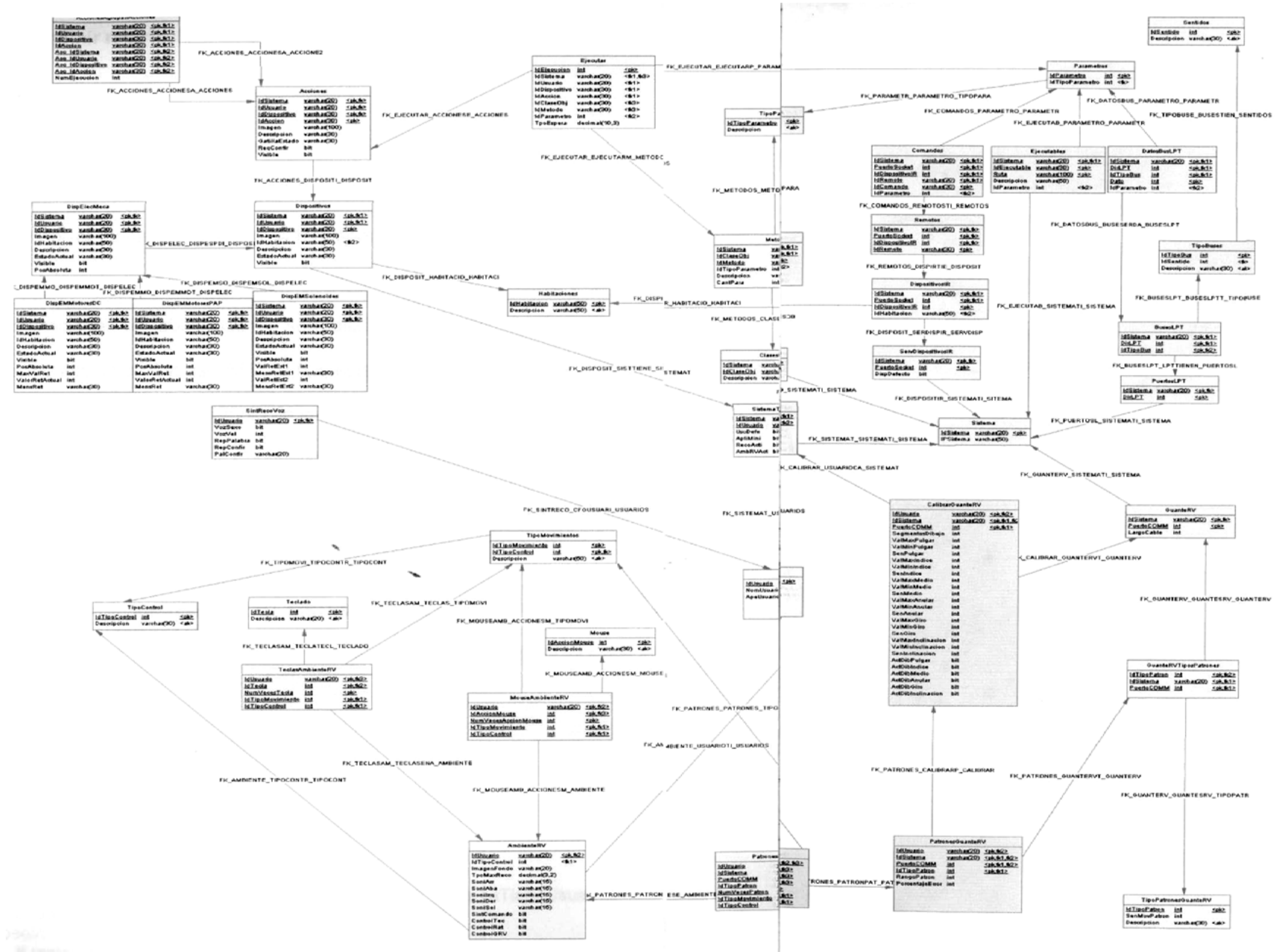

Diagrama Nº 47: Diagrama Entidad-Relación final BDSAD.

9.2.6 Definir restricciones de integridad

9.2.6.1 Integridad Referencial

Por el momento, y debido a requerimientos propios de la investigación, todo el manejo de la integridad referencial tanto para inserciones como para actualizaciones en tablas padres, se realizó en “cascada”, por lo tanto para la implementación en Sybase ASE 12.5.2, esta se realizó mediante “triggers” y no de forma declarativa, un ejemplo de implementación de los triggers de inserción y actualización, se presenta a continuación:

```
/* Trigger inserción "TI_SISTEMATIENEUSUARIOS" para tabla "SistemaTieneUsuarios" */
create trigger TI_SISTEMATIENEUSUARIOS on SistemaTieneUsuarios for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return

   /*  Padre "Sistema" debe existir al insertar un hijo en "SistemaTieneUsuarios"  */
   if update(IdSistema)
   begin
     if (select count(*)
```

```
        from   Sistema t1, inserted t2
        where  t1.IdSistema = t2.IdSistema) != @numrows
      begin
        select @errno  = 30002,
             @errmsg = 'Padre no existe en "Sistema". No se puede crear hijo en "SistemaTieneUsuarios".'
        goto error
      end
   end
   /*      Padre "Usuarios" debe existir cuando se inserta un hijo en "SistemaTieneUsuarios"  */
   if update(IdUsuario)
   begin
     if (select count(*)
        from   Usuarios t1, inserted t2
        where  t1.IdUsuario = t2.IdUsuario) != @numrows
      begin
        select @errno  = 30002,
             @errmsg = 'Padre no existe en "Usuarios". No se puede creaa hijo en "SistemaTieneUsuarios".'
        goto error
      end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

/* Trigger actualización "TU_SISTEMATIENEUSUARIOS" para tabla "SistemaTieneUsuarios" */

```
create trigger TU_SISTEMATIENEUSUARIOS on SistemaTieneUsuarios for update as
begin
  declare
    @numrows  int,
    @numnull  int,
    @errno    int,
    @errmsg   varchar(255)

    select  @numrows = @@rowcount
    if @numrows = 0
```

```
        return

  /*  Padre "Sistema" debe existir cuando se actualiza el hijo en "SistemaTieneUsuarios"
*/
      if update(IdSistema)
      begin
        if (select count(*)
           from   Sistema t1, inserted t2
           where  t1.IdSistema = t2.IdSistema) != @numrows
         begin
           select @errno  = 30003,
                 @errmsg  = 'Sistema" no existe. No se puede modificar el hijo en
"SistemaTieneUsuarios".'
           goto error
         end
      end
  /*    Padre   "Usuarios"   debe   existir   cuando   se   actualiza   el   hijo   en
"SistemaTieneUsuarios"  */
      if update(IdUsuario)
      begin
        if (select count(*)
           from   Usuarios t1, inserted t2
           where  t1.IdUsuario = t2.IdUsuario) != @numrows
         begin
           select @errno  = 30003,
                 @errmsg  = 'Usuarios" no existe. No se puede modificar el hijo en
"SistemaTieneUsuarios".'
           goto error
         end
      end
      /*    Modifica   código   padre   de   "SistemaTieneUsuarios"   para   todos   los   hijos   en
"Dispositivos"  */
      if update(IdSistema) or
        update(IdUsuario)
      begin
        if @@rowcount = 1
          update Dispositivos
            set   IdSistema = i1.IdSistema,
                 IdUsuario = i1.IdUsuario
          from   Dispositivos t2, inserted i1, deleted d1
            where  t2.IdSistema = d1.IdSistema
             and   t2.IdUsuario = d1.IdUsuario
             and  (i1.IdSistema != d1.IdSistema or
                  i1.IdUsuario != d1.IdUsuario)
        else
        begin
```

```
        declare cIns cursor for
        select
          IdSistema,
          IdUsuario
        from inserted
        declare cDel cursor for
        select
          IdSistema,
          IdUsuario
        from deleted
        declare @ins_IdSistema varchar(20)
        declare @ins_IdUsuario varchar(20)
        declare @del_IdSistema varchar(20)
        declare @del_IdUsuario varchar(20)

        open cIns
        open cDel
        fetch cIns into @ins_IdSistema,
                        @ins_IdUsuario
        fetch cDel into @del_IdSistema,
                        @del_IdUsuario

        while (@@sqlstatus = 0)
        begin
          update Dispositivos
            set   IdSistema = @ins_IdSistema,
                  IdUsuario = @ins_IdUsuario
          where  IdSistema = @del_IdSistema
           and   IdUsuario = @del_IdUsuario
          fetch cIns into @ins_IdSistema,
                          @ins_IdUsuario
          fetch cDel into @del_IdSistema,
                          @del_IdUsuario
        end
      end
    end

    /* Modifica código padre de "SistemaTieneUsuarios" para todos los hijos en
"CalibrarGuanteRV"  */
    if update(IdSistema) or
      update(IdUsuario)
    begin
      if @@rowcount = 1
        update CalibrarGuanteRV
          set   IdSistema = i1.IdSistema,
                IdUsuario = i1.IdUsuario
```

```
    from   CalibrarGuanteRV t2, inserted i1, deleted d1
      where  t2.IdSistema = d1.IdSistema
       and   t2.IdUsuario = d1.IdUsuario
       and  (i1.IdSistema != d1.IdSistema or
            i1.IdUsuario != d1.IdUsuario)
  else
  begin
    declare cIns2 cursor for
    select
      IdSistema,
      IdUsuario
    from inserted
    declare cDel2 cursor for
    select
      IdSistema,
      IdUsuario
    from deleted

    Set @ins_IdSistema = NULL
     Set @ins_IdUsuario = NULL
     Set @del_IdSistema = NULL
     Set @del_IdUsuario = NULL

    open cIns2
    open cDel2
    fetch cIns2 into @ins_IdSistema,
                @ins_IdUsuario
    fetch cDel2 into @del_IdSistema,
                @del_IdUsuario

    while (@@sqlstatus = 0)
    begin
      update CalibrarGuanteRV
        set   IdSistema = @ins_IdSistema,
              IdUsuario = @ins_IdUsuario
      where  IdSistema = @del_IdSistema
       and   IdUsuario = @del_IdUsuario
      fetch cIns2 into @ins_IdSistema,
                  @ins_IdUsuario
      fetch cDel2 into @del_IdSistema,
                  @del_IdUsuario
    end
  end
end

return
```

```
/*  Manejador de errores  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

9.2.6.2 Restricciones de la Empresa

Hasta el momento, y puesto que este modelo se enmarca dentro del contexto de una investigación, solo se conocen las siguientes restricciones adicionales:

En la tabla “TeclasAmbienteRV”, se almacena la información respecto a las teclas que ejecutan movimientos en el ambiente para algún usuario en particular, entonces las siguientes convinaciones no pueden presentarse:

TeclasAmbienteRV

IdUsuario	**IdTecla**	**NumVecesTecla**	**IdTipoMovimiento**	**IdTipoControl**
“Usuario”	65 (Ej: “a”)	1	1 (Ej: Arriba)	0
“Usuario”	65 (Ej: “a”)	1	2 (Ej: Abajo)	0

De ocurrir esto, el software podria interpretar la tecla con ID 65, tanto como una movimiento “Arriba”, como uno “Abajo” dentro del ambiente virtual. Esto no ocurre si se especifica una cantidad de veces distinta (“NumVecesTecla”), ya que si este valor es diferente, el software podrá

diferenciar cuantas veces se debe presionar la tecla para realizar un movimiento u el otro.

TeclasAmbienteRV

IdUsuario	IdTecla	NumVecesTecla	IdTipoMovimiento	IdTipoControl
“Usuario”	65 (Ej: “a”)	1	1 (Ej: Arriba)	0
“Usuario”	66 (Ej: “b”)	1	2 (Ej: Arriba)	0

Este tipo de combinación tampoco es válida, ya que dos teclas están haciendo referencia al mismo identificador de movimiento dentro del ambiente virtual, y por el momento, el software acepta que se configure solo una tecla para realizar uno de los movimientos.

Exactamente las mismas combinaciones inválidas las podemos encontrar en las tablas “MouseAmbienteRV” y “PatronesAmbienteRV”, que al igual que “TeclasAmbienteRV”, almacenan la información respecto a las acciones del mouse que gatillan movimientos dentro del ambiente virtual y los patrones de movimiento del guante virtual que también gatillan acciones al interior del ambiente.

Todas estas restricciones fueron resueltas agregando las validaciones necesarias dentro de los procedimientos almacenados de inserción y actualización tal como se muestra en el anexo digital (X:\Fuentes\BDSAD).

9.3 Construir y validar el modelo de datos lógico global

Puesto que el presente modelo no cuenta con vistas parciales y representa a la globalidad de los datos que intervienen en la problemática, es que este punto no es aplicable.

9.4 Traducir el modelo lógico global para el DBMS especificado

El DBMS utilizado para esta implementación fue “Sybase Adaptive Server Enterprise”, en su version Windows para desarrolladores 12.5.2.

9.4.1 Diseñar las relaciones bases para el DBMS especificado.

Las relaciones para Sybase ASE se adjuntan en el anexo digital “X:\Fuentes\BDSAD”. Un ejemplo de la implementación de estas relaciones, para el DBMS específico, se puede encontrar en el punto 9.2.6, en donde se muestra el manejo de la integridad referencial mediante triggers para inserción y actualización de la tabla “SistemaTieneUsuario”.

9.4.2 Diseñar las restricciones de la empresa para el DBMS especificado

La implementación de las restricciones de la empresa en Sybase ASE se adjuntan en el anexo digital “X:\Fuentes\BDSAD”.

A modo de ejemplo, y para cumplir con las restricciones planteadas en el punto 9.2.6.2, se insertó el siguiente código en el trigger de inserción y actualización de la tabla “TeclasAmbienteRV”:

```
/* Valida que el "Tipo de Movimiento" (IdTipoMovimiento) no este asignado en la tabla para un usuario y configuración en particular*/
	if (select count(*)
		from TeclasAmbienteRV t1, inserted t2
where  t1.IdUsuario = t2.IdUsuario and t1.IdTipoMovimiento = t2.IdTipoMovimiento) > 1
begin
		select @errno  = 100002,
	    @errmsg = 'Tipo de movimiento ya asignado.'
		goto error
end

/* Valida que la "Tecla" (IdTecla) no este asignada el mismo número de veces en la tabla para un usuario y configuración en particular*/
	if (select count(*)
		from TeclasAmbienteRV t1, inserted t2
		where  t1.IdUsuario = t2.IdUsuario and t1.IdTecla = t2.IdTecla
		and t1.IdTipoControl = t2.IdTipoControl
		and t1.NumVecesTecla = t2.NumVecesTecla) > 1
	begin
		select @errno  = 100003,
	    @errmsg = 'Repetición de tecla ya asignada.'
		goto error
end
```

Tal como se puede apreciar en el código del DBMS específico, de esta forma se evita que se cumplan las condiciones inválidas. De igual manera, esto se implementó para las tablas “MouseAmbienteRV” y “PatronesAmbienteRV” y sus triggers de inserción y actualización, ya que presentan exactamente las mismas restricciones.

9.4.3 Implementación

Debido a que el desarrollo del software, se realizó mediante una orientación a objetos, era imprescindible que la capa de datos se separara completamente de la capa de presentación, ya que esto es de vital importancia para realizar un mapeo lo más limpio posible del modelo de clases a las tablas de la base de datos. Además, lo anterior aumenta la independencia de las capas, con lo cual el software tiende a cumplir enteramente con el “Modelo de tres Capas”, el cual propone separar las capa de datos (DBMS Sybase ASE 12.5.2), lógica de negocios (procedimientos almacenados, triggers, etc.), y finalmente presentación (aplicación DomoSoft). Con lo cual el sistema aumenta su reusabilidad, flexibilidad, simplicidad de administración y escalabilidad.

De esta forma, se construyeron alrededor de 90 procedimientos almacenados, que solos o en conjunto, se encargan de establecer, obtener o actualizar los datos necesarios para el correcto funcionamiento de las distintas

transacciones requeridas por la aplicación. Si bien un número importante de estos procedimientos, solo presentan consultas e instrucciones tradicionales de SQL (insert, select anidados, inner join, left joinn, etc.), algunos otros presentaron casos especiales, en donde se tuvieron que emplear instrucciones o características especiales del DBMS utilizado y/o desarrollar algoritmos que cumpliesen con los requerimientos planteados.

Para este último tipo de procedimientos almacenados, se incluyen a continuación algunos ejemplos interesantes, los demás serán incluidos en su totalidad en el anexo digital (X:\Fuentes\BDSAD).

- **Procedimiento almacenado “pa_Datos_Ejecutar”:**

Luego de que el usuario selecciona un dispositivo definido y una de sus acciones asociadas para ser ejecutada, se requirió que un procedimiento retornara todos los datos de la tabla “Ejecutar”, en donde se asoció la acción definida por el usuario a una clase y método expuesto por la aplicación. Además, la aplicación también permite que una acción esté asociada a otras acciones del mismo u otro dispositivo (asociación recursiva entre tablas “Acciones” y “AccionesAgrupanAcciones”), para de esta forma lograr agrupar acciones en grupos de ejecuciones más complejos. Por lo tanto, además de obtener los datos de la tabla “Ejecutar”, también se requirió consultar la tabla “AccionesAgrupanAciones”, que alberga las asociaciones entre distintas acciones, y luego de obtener las acciones agrupadas, consultar nuevamente por los datos de la tabla “Ejecutar”, para cada una de estas acciones agrupadas. Puesto que de igual forma, cada acción agrupada, puede estar asociada a otras acciones, se requirió de un procedimiento que realizara llamadas recursivas para obtener las tuplas anidadas entre las tablas “Acciones” y “AccionesAgrupanAcciones”. Además, se debió implementar una manera en la cual el procedimiento almacenado no entrara en bucles infinitos, ya que si bien el procedimiento de inserción de datos de la tabla “AccionesAgrupanAcciones” previene este tipo de circunstancias, la incorrecta manipulación de los datos aún podría provocar que ocurriesen.

A continuación se presenta los procedimientos almacenados desarrollados para cumplir con lo anteriormente expuesto, al interior de los mismos se agregaron comentarios para las secciones importantes de código:

```
create procedure pa_Datos_Ejecutar @IdSistema varchar(20), @IdUsuario varchar(20), @IdDispositivo varchar(30), @IdAccion varchar(30)
as
begin
	declare @errno int, @errmsg varchar(255), @sqlString varchar(1024), @TblEje varchar(30)

	/* Nombre de la tabla única temporal, para los datos a ejecutar, el @@spid hace que sea única */

	set @TblEje = 'EjeDispAcci' + convert(varchar(4), @@spid)

	/* Se elimina por si existiera, producto de una terminación anormal en alguna sesion anterior */
	if exists (select 1 from sysobjects where id = object_id(@TblEje) and type = 'U')
	begin
		set @sqlString = 'drop table ' + @TblEje
		exec(@sqlString)
	end


	/* Variable SQL para ejecutar el contenido "string" de creación de la tabla temporal para albergar el resultado de lo que debe ejecutar una acción. */

	set @sqlString = 'create table ' + @TblEje
		+ ' (IdEje numeric identity, IdEjecucion int, IdSistema varchar(20), IdUsuario varchar(20),'
		+ ' IdDispositivo varchar(30), IdAccion varchar(30), IdClaseObj varchar(30), IdMetodo varchar(30),'
		+ ' IdParametro int, TpoEspera decimal(10,3))'

	exec(@sqlString)


	/* Creación de la tabla para albergar las raíces de los 'Dispositivos, Acciones' visitados y chequearlos durante la ejecución recursiva del procedimiento almacenado para evitar bucles infinitos. */
```

```
/* Si la tabla no se ha creado, entonces se crea, esta tabla no se eliminará para
usarla posteriormente */
if not exists (select 1 from sysobjects where id = object_id('RDisAcc' ) and type =
'U')
begin
        create table RDisAcc
         (SPID int, IdDispositivo varchar(30),
         IdAccion varchar(30), constraint PK_TBLRDISACC primary key (SPID,
        IdDispositivo, IdAccion))
end


/* Se limpia los valores asociados al @@spid de la tabla, por si existieran,
producto de una terminación anormal en alguna sesión anterior */
delete RDisAcc
        where SPID = @@spid


set @IdSistema = upper(@IdSistema)
set @IdUsuario = upper(@IdUsuario)
set @IdDispositivo = upper(@IdDispositivo)
set @IdAccion = upper(@IdAccion)


/* Se llama al procedimiento que realizará las llamadas recursivas */

exec pa_BuscaEjecutar @IdSistema, @IdUsuario, @IdDispositivo, @IdAccion,
@TblEje, @@spid

/* Se seleccionan y retornan todo los resultados de la busqueda */
set @sqlString = 'select * from ' + @TblEje
exec(@sqlString)


/* Se elimina la tabla utilizada */
if exists (select 1 from sysobjects where id = object_id(@TblEje) and type = 'U')
begin
        set @sqlString = 'drop table ' + @TblEje
        exec(@sqlString)
end


/* Se limpia los valores asociados al @@spid de la tabla */
delete RDisAcc
        where SPID = @@spid
```

```
        return

error:
        raiserror @errno @errmsg
        rollback  transaction*
end
```

Tal como se señaló anteriormente, el procedimiento debía realizar llamadas recursivas para obtener los datos de la tabla “Ejecutar”, asociados tanto a la acción seleccionada como de sus acciones agrupadas, por lo que se requería de una tabla temporal que albergara los datos que se fueran obteniendo. Esta tabla no podía ser temporal en términos de que solo existiese en memoria, ya que de esta forma solo el hilo de ejecución del procedimiento conocería y tendría acceso a dicha tabla, por lo tanto esta debería ser una tabla temporal física en la base de datos, pero además, no podría tener siempre el mismo nombre, ya que si distintas instancias de la aplicación necesitaran consultar por los datos a ejecutar para dos acciones distintas, esta tabla se poblaría con los datos de ambas, provocando ejecuciones inválidas. Es por esto que se utilizó la variable provista por el motor de Sybase “@@spid” (“Stored Procedure ID”, o “Identificador del Procedimiento Almacenado”), la cual retorna el ID único del hilo de ejecución del procedimiento almacenado, el cual se empleo para construir el nombre de la tabla tal como se muestra en la línea:

```
        set @TblEje = 'EjeDispAcci' + convert(varchar(4), @@spid)
```

Otro aspecto interesante de destacar, es que debido a que el nombre de la tabla se crea de forma dinámica dentro del procedimiento, también era necesario construir de la misma forma su estructura, esto se pudo realizar concatenando las instrucciones SQL necesarias para la estructura de la tabla dentro de una variable string, para luego ejecutar el contenido del mismo string como si fuese una consulta. Esto último fue posible de realizar gracias a la instrucción “exec”, la cual recibe como argumento la cadena que se debe ejecutar, tal como se aprecia en el siguiente extracto:

```
set @sqlString = 'create table ' + @TblEje
        + ' (IdEje numeric identity, IdEjecucion int, IdSistema varchar(20),
        IdUsuario varchar(20),'
        + ' IdDispositivo varchar(30), IdAccion varchar(30), IdClaseObj
        varchar(30), IdMetodo varchar(30),'
        + ' IdParametro int, TpoEspera decimal(10,3))'

exec(@sqlString)
```

Para ayudar a los procedimientos almacenados a resolver el problema de las búsquedas recursivas, se creó de forma estable dentro de la base de datos, una tabla de nombre “RDisAcc”, la cual alberga las “raíces” o pares “Dispositivo, Acción” en los cuales ya se ha buscado, y para diferenciar las raíces insertadas por una u otra ejecución de los procedimientos de búsqueda. Esta tabla posee un identificador único para las tuplas que es el “@@spid” del procedimiento que esta realizando las inserciones o consultas a esta tabla. Debido a esto último,

es que se encuentran dentro del procedimiento diversas validaciones relacionadas con esta tabla, para crearla de no existir o eliminar tuplas que hubiesen permanecido en la tabla con el valor de “@@spid” actual producto de alguna terminación anormal anterior.

En la parte central, el procedimiento almacenado realiza una llamada a “pa_BuscaEjecutar”, que será el encargo finalmente de realizar las llamadas recursivas sobre si mismo.

En la parte final el procedimiento realiza la limpieza correspondiente de las tablas utilizadas y algunas otras validaciones adicionales.

- **Procedimiento almacenado “pa_BuscaEjecutar”:**

El procedimiento almacenado “pa_BuscaEjecutar”, es llamado inicialmente por “pa_Datos_Ejecutar”, y recibe entonces como parámetros el identificador del dispositivo y la acción seleccionada, así como también el ID del procedimiento almacenado, contenido en “@SPID”.

Básicamente este procedimiento comienza obteniendo los datos de la tabla “Ejecutar” para el dispositivo y acción recibidos como parámetros, para luego consultar, con estos mismos identificadores, por las acciones asociadas a estos en la tabla “AccionesAgrupanAcciones”. El resultado de esto es pasado a un cursor el cual se irá recorriendo hacia delante con el fin de obtener de uno en uno los pares “Dispositivo, Acción” y enviarlos como parámetros de forma recursiva al procedimiento “pa_BuscaEjecutar”. De esta forma se repite el proceso anterior, hasta obtener todos los datos asociados directa o indirectamente de la tabla “Ejecutar”.

A continuación se muestra el código de este procedimiento, con comentarios en las secciones importantes de este:

```
create procedure pa_BuscaEjecutar @IdSistema varchar(20), @IdUsuario varchar(20),
@IdDispositivo varchar(30), @IdAccion varchar(30), @TblEje varchar(30), @SPID int

as
begin
        declare @errno int, @errmsg varchar(255), @sqlString varchar(1024),
        @BucleInf int

        /* Si la tabla no se ha creado, entonces se crea, esta tabla no se eliminará para
        usarla posteriormente */

        if not exists (select 1 from sysobjects where id = object_id('RDisAcc' ) and type =
        'U')
        begin
                create table RDisAcc
                 (SPID int, IdDispositivo varchar(30),
                 IdAccion varchar(30), constraint PK_TBLRDISACC primary key (SPID,
                IdDispositivo, IdAccion))
        end


        set @BucleInf = 0

        if (select count(*)
                from   RDisAcc
                where  RDisAcc.SPID = @SPID
                and   RDisAcc.IdDispositivo = upper(@IdDispositivo)
                and   RDisAcc.IdAccion = upper(@IdAccion)) != 0
        begin
                /* Si ya esta como raíz dentro de los "Dispositivos, Acción", quiere decir
                que de seguir se producirá un bucle infinito, por lo tanto se cambia el
                switch para evitar esto */

                set @BucleInf = 1
        end
      else
        begin
                /* Si no está, entonces se agrega como una raíz "Dispositivo, Acción"*/

                insert into RDisAcc values (@SPID, upper(@IdDispositivo),
                upper(@IdAccion))
        end
```

```
if @BucleInf = 0
begin

	/* Primero se insertan las acciones directas de la tabla Ejecutar */
	set @sqlString = 'insert into ' + @TblEje
		+ ' (IdEjecucion, IdSistema, IdUsuario, IdDispositivo, IdAccion,'
		+ ' IdClaseObj, IdMetodo, IdParametro, TpoEspera)'
		+ ' (select'
		+ ' Ejecutar.IdEjecucion, Ejecutar.IdSistema, Ejecutar.IdUsuario,'
		+ ' Ejecutar.IdDispositivo, Ejecutar.IdAccion,'
		+ ' Ejecutar.IdClaseObj, Ejecutar.IdMetodo,'
		+ ' Ejecutar.IdParametro, Ejecutar.TpoEspera'
		+ ' from Ejecutar'
		+ ' where Ejecutar.IdSistema = '''  + upper(@IdSistema)  + ''' and'
		+ ' Ejecutar.IdUsuario = ''' + upper(@IdUsuario) + ''' and'
		+ ' Ejecutar.IdDispositivo = ''' + upper(@IdDispositivo) + ''' and'
		+ ' Ejecutar.IdAccion = ''' + upper(@IdAccion) + ''')'

	exec(@sqlString)


	/* Se seleccionan las acciones agrupadas */
	declare cAccAgrAcc cursor for
	select
		AccionesAgrupanAcciones.IdSistema,
		AccionesAgrupanAcciones.IdUsuario,
		AccionesAgrupanAcciones.Acc_IdDispositivo,
		AccionesAgrupanAcciones.Acc_IdAccion

		from AccionesAgrupanAcciones	where
		AccionesAgrupanAcciones.IdSistema = upper(@IdSistema) and
		AccionesAgrupanAcciones.IdUsuario = upper(@IdUsuario) and
		AccionesAgrupanAcciones.IdDispositivo = upper(@IdDispositivo)
		and
		AccionesAgrupanAcciones.IdAccion = upper(@IdAccion)

	order by AccionesAgrupanAcciones.NumEjecucion

	declare @Acc_IdSistema varchar(20), @Acc_IdUsuario varchar(20)
	declare @Acc_IdDispositivo varchar(30), @Acc_IdAccion varchar(30)
```

```
        open cAccAgrAcc
                fetch cAccAgrAcc into @Acc_IdSistema, @Acc_IdUsuario,
                @Acc_IdDispositivo, @Acc_IdAccion

                while (@@sqlstatus = 0)
                begin
                        /* Llamada recursiva al procedimiento almacenado */
                        exec pa_BuscaEjecutar @Acc_IdSistema, @Acc_IdUsuario,
                        @Acc_IdDispositivo, @Acc_IdAccion, @TblEje, @SPID

                        fetch cAccAgrAcc into @Acc_IdSistema, @Acc_IdUsuario,
                        @Acc_IdDispositivo, @Acc_IdAccion
                end
        end

        return

error:
        raiserror @errno @errmsg
        rollback  transaction
end
```

Además de algunas validaciones iniciales, el procedimiento comienza consultando a la tabla “RdisAcc” si el par “Dispositivo, Acción” no ha sido recibido antes, lo cual indicaría que se ha entrado en un bucle infinito, tal como se muestra en las siguientes líneas:

```
        if (select count(*) from   RdisAcc where  RDisAcc.SPID = @SPID
                and   RDisAcc.IdDispositivo = upper(@IdDispositivo)
                and   RDisAcc.IdAccion = upper(@IdAccion)) != 0
        begin
                set @BucleInf = 1
                …
```

Más tarde el valor de la variable “@BucleInf” indicará si se ha producido o no en un bucle infinito. De no encontrarse el par “Dispositivo, Acción” en la tabla “RdisAcc”, se debe almacenar en esta para futuras búsquedas recursivas.

Si la variable “@BucleInf” no señala un bucle infinito (@BucleInf = 0), el procedimiento busca los datos en la tabla “Ejecutar”, para el ID de dispositivo y acción recibidos como parámetros, y los almacena en la tabla temporal de nombre “@TblEje”. Posteriormente, almacena en el cursor “cAccAgrAcc” el resultado de la consulta a la tabla “AccionesAgrupanAcciones” para el par “Dispositivo, Acción” actual.

Finalmente, el procedimiento consulta por el primer resultado almacenado en el cursor, y comienza un ciclo “while” recorriendo “cAccAgrAcc”, y realizando llamadas recursivas a si mismo por cada tupla almacenada en “cAccAgrAcc”, y repetir el procedimiento anterior hasta obtener todos los valores asociado en la tabla “Ejecutar”.

A continuación, se muestra el extracto en donde se realizan las llamadas recursivas:

```
while (@@sqlstatus = 0)
        /* “@@sqlstatus”Establece cuando se termino de recorrer el
cursor*/
begin
        exec pa_BuscaEjecutar @Acc_IdSistema, @Acc_IdUsuario,
        @Acc_IdDispositivo, @Acc_IdAccion, @TblEje, @SPID
                ...
```

- **Procedimiento almacenado “pa_InsertaDatos_AccAgrAcc”:**

Tal como se menciono anteriormente, la aplicación y el modelo de datos, permiten que una acción pueda asociarse con otras del mismo dispositivo u otro, para conformar acciones más complejas útiles al usuario. Debido a esto, este procedimiento debió cerciorarse de que las agrupaciones que se fueran insertando a la tabla “AccionesAgrupenAcciones”, no provocaran posteriores bucles infinitos a la hora de consultar por sus datos.

Básicamente, este procedimiento, además de realizar algunas validaciones iniciales, almacena el ID de dispositivo y acción recibidos como parámetros, en la tabla “RDisAcc”, para evitar bucles infinitos en sus llamadas recursivas.

Luego se realiza la llamada inicial al procedimiento “pa_BuscaBucleInfAgrDisAcc”, el cual determinará si la asociación que se intenta ingresar produce un bucle infinito o no, esto ultimo, lo devuelve en el valor de salida contenido en “@BucleInf”.

A continuación, se muestra la implementación de este procedimiento almacenado:

```
create procedure pa_InsertaDatos_AccAgrAcc @IdSistema varchar(20), @IdUsuario
varchar(20), @IdDispositivo varchar(30), @IdAccion varchar(30), @Acc_IdSistema
varchar(20), @Acc_IdUsuario varchar(20), @Acc_IdDispositivo varchar(30),
@Acc_IdAccion varchar(30)
as
begin
	declare @errno int, @errmsg varchar(255), @NumEjecucion int, @BucleInf int

	/* Se asume inicialmente la no existencia de bucle infinitos */
	set @BucleInf = 0

	/* Se valida la posibilidad de un ingreso que produzca un bucle infinito */
	/* Si la tabla no se a creado, entonces se crea, esta tabla no se eliminara para
	usarla posteriormente */

	if not exists (select 1 from sysobjects where id = object_id('RDisAcc' ) and type =
	'U')
	begin
		create table RDisAcc
		 (SPID int, IdDispositivo varchar(30),
		 IdAccion varchar(30), constraint PK_TBLRDISACC primary key (SPID,
		IdDispositivo, IdAccion))
	end

	/* Se limpia los valores asociados al @@spid de la tabla, por si existieran,
	producto de una terminación anormal en alguna sesion anterior */
	delete RDisAcc
		where SPID = @@spid

	set @IdSistema = upper(@IdSistema)
	set @IdUsuario = upper(@IdUsuario)
	set @IdDispositivo = upper(@IdDispositivo)
	set @IdAccion = upper(@IdAccion)
	set @Acc_IdSistema = upper(@Acc_IdSistema)
	set @Acc_IdUsuario = upper(@Acc_IdUsuario)
	set @Acc_IdDispositivo = upper(@Acc_IdDispositivo)
	set @Acc_IdAccion = upper(@Acc_IdAccion)

	if (select count(*) from   RDisAcc      where  RDisAcc.SPID = @@spid
		and   RDisAcc.IdDispositivo = upper(@IdDispositivo)
		and   RDisAcc.IdAccion = upper(@IdAccion)) = 0
	begin
	/* Si no esta, entonces se agrega como una raiz "Dispositivo, Accion" */
		insert into RDisAcc values (@@spid, upper(@IdDispositivo),
		upper(@IdAccion))
	end
```

```
        /* Se llama al procedimiento recursivo */
        exec pa_BuscaBucleInfAgrDisAcc @Acc_IdSistema, @Acc_IdUsuario,
        @Acc_IdDispositivo, @Acc_IdAccion, @@spid, @BucleInf output

        if @BucleInf = 0
        begin
                select @NumEjecucion =
                max(AccionesAgrupanAcciones.NumEjecucion) from
                AccionesAgrupanAcciones where AccionesAgrupanAcciones.IdSistema
                = upper(@IdSistema)
                and AccionesAgrupanAcciones.IdUsuario = upper(@IdUsuario)
                and AccionesAgrupanAcciones.IdDispositivo = upper(@IdDispositivo)
                and AccionesAgrupanAcciones.IdAccion = upper(@IdAccion)

                if @NumEjecucion = NULL
                begin
                        Set @NumEjecucion = 0
                end

                insert into AccionesAgrupanAcciones values (upper(@IdSistema),
                upper(@IdUsuario), upper(@IdDispositivo),
                upper(@IdAccion), upper(@Acc_IdSistema), upper(@Acc_IdUsuario),
                upper(@Acc_IdDispositivo), upper(@Acc_IdAccion), @NumEjecucion +
                1)
        end
      else
        begin
                select @errno  = 100005,
                @errmsg = 'Agrupación de acción produce bucle infinito.'
                goto error
        end

        /* Se limpia los valores asociados al @@spid de la tabla */
        delete RDisAcc
                where SPID = @@spid

        return

error:
        raiserror @errno @errmsg
        rollback  transaction
end
```

- **Procedimiento almacenado “pa_BuscaBucleInfAccDisAcc”:**

Básicamente este procedimiento, realiza la misma función que “pa_BuscaEjecutar” en el sentido de que ambos, validan las raíces existentes en “RDisAcc” para evitar entrar en bucles infinitos, y luego pueblan un cursor con los pares “Dispositivo, Acción” encontrados en la tabla “AccionesAgrupanAcciones” para los ID recibidos como parámetros, para luego recorrer el cursor en un ciclo “while” y realizar llamadas recursivas hasta completar el proceso.

La diferencia principal, es que “pa_BuscaEjecutar”, además, se encarga de obtener los datos en la tabla “Ejecutar” para los pares encontrados, y “pa_BuscaBucleInfAccDisAcc”, solo se encarga de validar la existencia de un posible bucle infinito.

Esto nos señala que posiblemente “pa_BuscaEjecutar”, podría ser una función dentro de la base de datos, que pudiese ser utilizada tanto por “pa_InsertaDatos_AccAgrAcc” y “pa_BuscaEjecutar”, pero por el momento y debido a que la aplicación se enmarca dentro de una investigación, y no pasará por la etapa de implementación final, se decidió mantener ambos procedimientos separados, para realizar mas fácilmente depuraciones en uno u otro, sin comprometer a los demás procedimientos que los utilizan, luego de que estén lo suficientemente depurados, se podrá diseñar una forma en la cual la reutilización del código sea mas eficiente.

A continuación, se presenta el código del procedimiento, con comentarios en las secciones importantes:

```
create procedure pa_BuscaBucleInfAgrDisAcc @IdSistema varchar(20), @IdUsuario varchar(20), @IdDispositivo varchar(30), @IdAccion varchar(30), @SPID int, @BucleInf int output
as
begin
	declare @errno int, @errmsg varchar(255), @sqlString varchar(1024)

	/* Si la tabla no se ha creado, entonces se crea, esta tabla no se eliminara para usarla posteriormente */
	if not exists (select 1 from sysobjects where id = object_id('RDisAcc' ) and type = 'U')
	begin
		create table RDisAcc
		 (SPID int, IdDispositivo varchar(30),
		 IdAccion varchar(30), constraint PK_TBLRDISACC primary key (SPID, IdDispositivo, IdAccion))
	end

	set @BucleInf = 0


	if (select count(*) from   RDisAcc      where  RDisAcc.SPID = @SPID
		and   RDisAcc.IdDispositivo = upper(@IdDispositivo)
		and   RDisAcc.IdAccion = upper(@IdAccion)) != 0
	begin
		/* Si ya esta como raíz dentro de los "Dispositivos, Acción", quiere decir que de seguir se producira un bucle infinito, por lo tanto se cambia el switch para evitar esto*/

		set @BucleInf = 1
	end
      else
	begin
		/* Si no está, entonces se agrega como una raíz "Dispositivo, Acción"*/

		insert into RDisAcc values (@SPID, upper(@IdDispositivo), upper(@IdAccion))
	end

	if @BucleInf = 0
```

```
	begin

		/* Se seleccionan las acciones agrupadas */
		declare cAccAgrAcc cursor for
		select
			AccionesAgrupanAcciones.IdSistema,
			AccionesAgrupanAcciones.IdUsuario,
			AccionesAgrupanAcciones.Acc_IdDispositivo,
			AccionesAgrupanAcciones.Acc_IdAccion

			from AccionesAgrupanAcciones where
			AccionesAgrupanAcciones.IdSistema = upper(@IdSistema) and
			AccionesAgrupanAcciones.IdUsuario = upper(@IdUsuario) and
			AccionesAgrupanAcciones.IdDispositivo = upper(@IdDispositivo)
			and AccionesAgrupanAcciones.IdAccion = upper(@IdAccion)

		order by AccionesAgrupanAcciones.NumEjecucion

		declare @Acc_IdSistema varchar(20), @Acc_IdUsuario varchar(20)
		declare @Acc_IdDispositivo varchar(30), @Acc_IdAccion varchar(30)

		open cAccAgrAcc
		fetch cAccAgrAcc into @Acc_IdSistema, @Acc_IdUsuario,
		@Acc_IdDispositivo, @Acc_IdAccion

		while (@@sqlstatus = 0)
		begin
			/* Llamada recursiva al procedimiento almacenado */

			exec pa_BuscaBucleInfAgrDisAcc @Acc_IdSistema,
			@Acc_IdUsuario, @Acc_IdDispositivo, @Acc_IdAccion, @SPID,
			@BucleInf output

			fetch cAccAgrAcc into @Acc_IdSistema, @Acc_IdUsuario,
			@Acc_IdDispositivo, @Acc_IdAccion
		end

	end

	return

error:
	raiserror @errno @errmsg
	rollback  transaction
end
```

Finalmente, respecto a los procedimientos almacenados recursivos, es importante señalar, que si bien han demostrado funcionar en los casos de pruebas empleados, la manipulación de datos en forma recursiva, plantea muchas posibilidades de combinatorias entre ellos, por lo cual todavía se encuentran en proceso de depuración, sin embargo, y aunque los procedimientos no detectaran bucles infinitos, la aplicación no entraría en uno de estos, ya que Sybase ASE 12.5.2 tiene un limite por defecto de 16 bucles anidados, lo cual evitaría este tipo de situaciones. La idea de implementarlos, es para lograr un mayor control sobre los datos y realizar consultas más eficientes a estos.

- **Procedimiento almacenado “pa_Datos_ParametrosPorTipo”:**

Cuando la aplicación le ofrece al usuario asociar una acción definida por el, con una clase y método expuesto por la aplicación, esta ultima debe listar todos los parámetros disponibles para ese método, y puesto a que los parámetros están generalizados, entonces, se debe consultar a la tabla “Parámetros”, pasando como parámetro el identificador del “Tipo de Parámetro”, (tanto el método, como los parámetros generalizados, tienen tipo), tal como lo muestra el siguiente código de implementación del procedimiento:

```
create procedure pa_Datos_ParametrosPorTipo @IdTipoParametro int
as
begin
	declare @errno int, @errmsg varchar(255)

	select
		Comandos.IdSistema,
		Comandos.PuertoSocket,
		Comandos.IdDispositivoIR,
		Comandos.IdRemoto,
		Comandos.IdComando,
		Comandos.IdParametro,
		Ejecutables.IdSistema,
		Ejecutables.IdEjecutable,
		Ejecutables.Ruta,
		Ejecutables.Descripcion,
		Ejecutables.IdParametro,
		DatosBusLPT.IdSistema,
		DatosBusLPT.DirLPT,
		DatosBusLPT.IdTipoBus,
		DatosBusLPT.Dato,
		DatosBusLPT.IdParametro

	from Parametros
		left join Comandos on
			Parametros.IdParametro = Comandos.IdParametro
		left join Ejecutables on
```

```
                Parametros.IdParametro = Ejecutables.IdParametro
        left join DatosBusLPT on
                Parametros.IdParametro = DatosBusLPT.IdParametro
    where
        Parametros.IdTipoParametro = @IdTipoParametro

    return

error:
    raiserror @errno @errmsg
    rollback  transaction
end
```

Dentro de este procedimiento, es importante notar que se deben consultar todas las tablas que se generalizan en “Parámetros”, ya que cualquiera de estas podría contener el “Tipo de Parámetro” que se está buscando, lo cual nos indica, que si se agregaran tablas a la generalización de parámetros, se deberán agregar estas a la consulta.

- **Procedimiento almacenado “pa_InsertaDatos_DatosBusLPT”:**

Finalmente, se incluye este procedimiento almacenado, para ejemplificar como se insertan los datos en una de las tablas perteneciente a la generalización “Parámetros”, en este caso la tabla “DatosBusLPT”, la cual contiene los valores que pueden ser enviados o recibidos, por los distintos buses del puerto LPT.

Tal como se mencionó anteriormente, los métodos expuestos por la aplicación utilizan valores como parámetros que pueden provenir de diferentes tabla en el sistema, y debido a que estas componen sus claves primarias de distinta manera y con distintos tipos de datos, es que se creo una generalización en la tabla “Parámetros”, la cual generará y les heredará a las tablas asociadas, un identificador único, para cada tupla en estas.

La implementación del procedimiento almacenado es la siguiente:

```
create procedure pa_InsertaDatos_DatosBusLPT @IdSistema varchar(20), @DirLPT int, @IdTipoBus int, @Dato int, @IdTipoParametro  int
as
begin
        declare @errno int, @errmsg varchar(255), @IdParametro int

        exec pa_InsertaDatos_Parametros @IdTipoParametro, @IdParametro output

        insert into DatosBusLPT  values (upper(@IdSistema), @DirLPT, @IdTipoBus, @Dato, @IdParametro)

        return
error:
        raiserror @errno @errmsg
        rollback  transaction
end
```

Como se puede apreciar, lo primero que hace el procedimiento, es ejecutar “pa_InsertaDatos_Parametros”, el cual generará e insertará en la tabla “Parametros”, un identificador que devolverá en la variable de salida “@IdParametro”, tal como se muestra a continuación:

```
exec pa_InsertaDatos_Parametros @IdTipoParametro, @IdParametro output
```

Luego el valor devuelto, es utilizado como valor de clave foránea en la inserción de datos en la tabla “DatosBusLPT”:

```
insert into DatosBusLPT  values (upper(@IdSistema), @DirLPT, @IdTipoBus, @Dato, @IdParametro)
```

Para terminar, cabe señalar que si bien la clave primaria de la tabla “Parámetros”, es un correlativo, no se utilizaron tipos de datos “Auto numéricos” generados por el mismo motor, y se prefirió utilizar el siguiente método para generarlos:

```
select @IdParametro = max(Parametros.IdParametro)+1 from Parametros
if @IdParametro = NULL
begin
        Set @IdParametro = 0
end
insert into Parametros  values (@IdParametro, @IdTipoParametro)
```

Si bien este método podría provocar problemas en ambientes en donde se estuviesen realizando múltiples inserciones en la tabla “Parametros”, este no es el caso para la presente aplicación, en la cual esta tabla no es utilizada para inserciones constantemente. Además, de esta forma se mantiene un control mas estrecho sobre la creación del código, y posibilita que se realicen mejoras sobre la generación de éste, así como futuras búsquedas por filtros basados en rangos de uno o mas códigos, etc.

Esto último se aplica a todas las tablas del modelo, que utilizan como claves, valores numéricos incrementales.

9.5 Diseñar representación física

9.5.1 Analizar transacciones

Las transacciones del presente modelo de datos se realizaron mediante procedimientos almacenados, y puesto que este modelo se enmarca dentro de una investigación, no es necesario por el momento un análisis más profundo respecto a la carga de transacciones.

9.5.2 Elegir organización de archivos

Para la presente implementación, se utilizo el sistema de archivos por defecto utilizados por Sybase ASE 12.5.2.

9.5.3 Elegir índices secundarios

En este punto solo cabe señalar, que se mantuvieron las claves alternas ya presentadas en el punto 9.1.5 como índices secundarios para las tablas.

9.5.4 Introducción de redundancia controlada

Tal como se puede apreciar en el diagrama final del modelo, se introdujo la siguiente redundancia controlada en la forma de una “Partición Vertical” para la tabla “PatronesGuanteRV”:

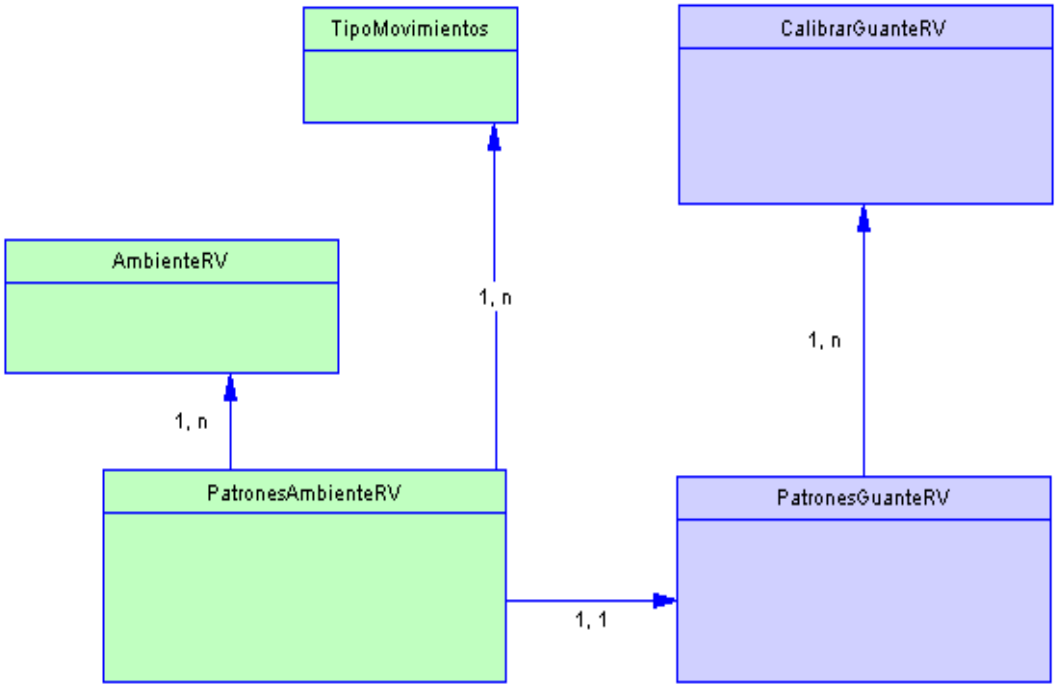

Diagrama Nº 48: Redundancia controlada tabla “PatronesGuanteRV”.

Esto se realizó para permitir que este modelo relacional se adapte de mejor forma al modelo de clases con el cual se construyó el software. De esta forma, los datos respecto a los patrones de movimiento para una calibración del guante en particular, quedan separados de los datos que definen que patrones gatillarán movimientos dentro del ambiente, mejorando el aislamiento de los datos entre las clases “AmbienteRV” y “GuanteRV”.

9.5.5 Estimar los requerimientos de espacio en disco

Puesto que el software, y por ende la base de datos, se desarrollaron dentro del marco de una investigación, por el momento no existe una implementación formal que ayude a realizar estimaciones respecto a los requerimientos de espacio en disco, sin embargo, actualmente la base de datos se encuentra poblada y no ha presentado problemas de espacio desde su asignación original de 250Mb (200Mb para el “Device” de la BD y 50Mb para “log” o “registro de transaciones” de la BD).

9.6 Diseñar mecanismos de seguridad

9.6.1 Diseñar vistas de usuarios

Debido a que por el momento la investigación no se centra en definir los permisos particulares para cada transacción, es que actualmente todas las transacciones tienen permisos totales de ejecución para los miembros del grupo “Administrador” de la base de datos.

9.6.2 Diseñar reglas de acceso

Aunque actualmente la implementación no requiere de reglas especiales de acceso, se consideró como regla general, que los usuarios que ingresen a la base de datos, deben ser usuarios válidos tanto del equipo Windows como un “login” válido en la BD, luego, de acuerdo al grupo al cual pertenezcan dentro de la base de datos, tendrán acceso a ejecutar o no una transacción en particular. Actualmente por motivos de la investigación, solo se requirió de un

solo grupo, en este caso “Administrador” en el cual se tiene permisos totales sobre todas las transacciones.

9.7 Monitoreo y refinamiento del sistema operacional

Debido a que este sistema se enmarca dentro del contexto del resultado de una investigación, es que este punto no se aplica, ya que el sistema no ha pasado por una etapa de implantación o “marcha blanca” formal.

10. Conclusiones y/o Recomendaciones

A pesar del relativamente corto periodo de desarrollo, ya se pueden apreciar los resultados prácticos de la investigación, y además, es posible demostrar que los conceptos empleados son válidos y que son factibles de implementar.

Si bien el sistema ha demostrado ser útil, práctico y fácil de usar, se debe considerar que un proyecto de este tipo no incluye, de manera formal, la fase de implementación, por lo cual, el sistema no tiene el grado de madurez suficiente para ser considerado en ambientes críticos, en donde la disponibilidad del sistema sea demasiado importante. Aunque sin duda alguna, con el tiempo y pruebas suficientes, es muy posible que pueda llegar a ser considerada su utilización en ambientes de este tipo ya sea de forma parcial o total.

Como resultado de esta investigación inicial, se ha logrado desarrollar un primer prototipo para un sistema de asistencia Domótica o “Primera aproximación a la solución”, que aún hoy no explota todas las capacidades que posee, y que sin duda alguna abre las puertas a un sin número de investigaciones adicionales que puedan aprovechar las innumerables ofertas que ofrece un proyecto multidisciplinario como este. Algunas de las áreas más interesantes de destacar para futuros avances son: Todo tipo de desarrollos

mecánicos y electrónicos; Portación del software a otras plataformas; Diferentes tipos de controles de acceso y autentificación en el sistema, así como también aumentar el control y administración de los permisos para los usuarios; Creación de interfaces Web y para dispositivos móviles (Palm, Handled PC, Teléfonos Celulares, etc.); Desarrollo de compiladores capaces de interpretar la sintaxis del lenguaje humano, permitiendo un control más natural sobre el sistema; etc.

También es importante mencionar que el software desarrollado para este proyecto, se ha registrado dentro “O.S.I” (“Open Source Initiative”, o “Iniciativa para Fuentes Abiertas” por sus siglas en Ingles), lo cual lo convierte en un software de libre distribución y de código abierto, bajo los términos de la licencia “G.P.L” (“General Public License”, o “Licencia Publica General”), lo cual, sumado a los esquemas electrónicos y mecánicos también liberados, hacen de éste, un sistema de “Arquitectura Abierta”, permitiendo a futuro integrar a profesionales de distintas áreas de la salud y del conocimiento en general, en pro de la integración e igualdad de las personas con discapacidades de todo tipo.

11. Bibliografía

[Booch, Jacobson, Rumbaugh] Grady Booch, Ivar Jacobson, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Addison Wesley. Primera Edición. 2000.

[DLSIIS2000] DLSII. Departamento de Lenguas y Sistemas Informáticos e Ingeniería del Software. Disponible en: http://lml.ls.fi.upm.es/mdp/si/, 20 de abril 2004.

[Fowler, Scott] Martin Fowler, Kendall Scott. UML Gota a Gota. Addison Wesley. Primera Edición. 1999.

[Stevens, Pooley] Perdita Stevens, Rob Pooley. Utilización de UML en Ingeniería del Software con Objetos y Componentes. Addison Wesley. Primera Edición. 2002.