Metodología para el Modelado y Generación de Código de Control de Sistemas Secuenciales mediante Redes de Petri Jerárquicas

Methodology for the Modeling and Generation of Control Code of Sequential Systems by means of Hierarchical Petri Nets

German Zapata M.MSc[1], John W. Branch B. PhD [2] y Luis F. Quintero H. Ing [1]
1. GAUNAL: Grupo de Automática de la Universidad Nacional de Colombia, Escuela de Ingeniería Eléctrica, Facultad de Minas, Universidad Nacional de Colombia sede Medellín
2. GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial, Escuela de Ingeniería de Sistemas, Facultad de Minas, Universidad Nacional de Colombia sede Medellín
gdzapata@unal.edu.co, jwbranch@unal.edu.co, lfquint0@unal.edu.co

Recibido para revisión 26 de Marzo de 2007, aceptado 22 de Junio de 2007, versión final 28 de junio de 2007

Resumen— En este trabajo se presenta una metodología para la generación automática de código para controladores lógicos programables (PLCs), según la norma IEC 61131-3. A partir del modelo del automatismo construido en redes de Petri jerárquicas, se presentan las reglas para generar el código en lenguaje de instrucciones, para garantizar con ello la portabilidad. Se retoman definiciones ya planteadas en la literatura referente al formalismo de la redes de Petri y se hace un aporte sobre la sintaxis y semántica de éste para que sea implementable en PLCs. Esta metodología permite aprovechar técnicas de ingeniería de software como la programación orientada a objetos y las capacidades de alto nivel embebidas en los controladores lógicos, para resolver problemas complejos de automatización industrial vía modularización y reusabilidad del código.

Palabras Clave— Redes de Petri Jerárquicas, Métodos Formales, Lenguajes de Programación, Modelos, Automatización, Diseño de Controladores, Ingeniería de Software.

Abstract— In this work a methodology is presented for the automatic generation of code for programmable logical controllers (PLCs), according to the norm IEC 61131-3. Starting from the model of the automatism built in hierarchical Petri nets nets, the rules are presented to generate the code in language of instructions, to guarantee with it the portability. Definitions are already recaptured outlined in the literature with respect to the formalism of the Petri nets and a contribution is made on the syntax and semantics of this so that it is appropriate for the implementation in PLCs. This methodology allows to take advantage technical of software engineering like the object-oriented programming and the capacities of high level embedded in the logical controllers, to solve complex problems of industrial automation via modularization and reusibility of the code.

Keywords— Hierarchical Petri Nets, Formal Methods, Programming Languages, Models, Automation, Controllers' Design, Software Engineering.

I. INTRODUCCIÓN

La tendencia tecnológica dentro de la industria en el control de sistemas discretos automatizados se centra en la implementación del controlador lógico programable (PLC). Sus aplicaciones presentes y futuras se caracterizan por la integración con otros dispositivos y sistemas [1][2], y por que ahora no sólo realizan operaciones secuenciales, sino también rutinas de autodiagnóstico, control supervisorio, diagnóstico de fallas de sensores, recuperación después de fallas, enlaces con niveles de automatización jerárquicos superiores, participación en redes de comunicaciones, manejo de recetas, e incluso, controlan procesos por lotes y procesos continuos. Además, requieren ser lo suficientemente flexibles para ser capaces de adaptar rápidamente las estrategias de control a

los cambios que requiera el proceso y la posibilidad de usar programas basados en el estándar de automatización ANSI-ISA S88.

Esta creciente complejidad en los procesos y en exigencias de desempeño ha generado también un aumento en la complejidad de los programas de control lógico, lo que lleva a que las técnicas convencionales de programación, basadas en lógica de relés, sean reemplazadas por técnicas que utilizan metodologías formales de diseño, que garanticen tiempos de desarrollo más cortos y soluciones más confiables, sistemáticas y seguras [2][3][4].

Se plantea además, que las técnicas de programación deben migrar hacia la programación en alto nivel [5] y aprovechar el potencial de la programación orientada a objetos [6]. En este sentido la norma IEC 61131-3 [7] define las especificaciones de la sintaxis y semántica de los lenguajes de programación de PLCs, incluyendo el modelo de software y la estructura del lenguaje. La programación puede ser evaluada mediante técnicas de ingeniería de software y soportada por plataformas formales robustas que permitan validar, también formalmente, el desempeño del sistema bajo control.

Además de la técnica de programación, se plantea la utilización de una metodología formal para modelar los algoritmos de control y que además permita el diseño orientado a objetos [8]. Las redes de Petri Jerárquicas (HPN) responden a estos requerimientos. Como una herramienta matemática y gráfica, proveen un ambiente uniforme para el modelamiento, el análisis formal y el diseño de algoritmos de control lógico. Una de las mayores ventajas del uso de modelos de Redes de Petri es que el mismo modelo es usado para el análisis de las propiedades de comportamiento y la evaluación de desempeño. Posterior a la validación y verificación del algoritmo de control [9][10], se exige que el método de diseño plantee unas directrices para la conversión del modelo al lenguaje de programación del PLC, según la norma IEC 61131-3 [1][3][11].

Este trabajo presenta una metodología para generar el código IEC a partir de modelos del automatismo en HPN. Para ello se requiere, además de la definición del formalismo, la presentación de las técnicas de jerarquización y modularización, la simbología empleada para representar módulos y subredes y las reglas para generar el programa en lenguaje de instrucciones, a partir de la dinámica de la red, la asignación de acciones y el llamado de módulos y subredes.

El artículo está organizado como sigue. En las secciones 2 y 3 se describe el formalismo de las HPN y las técnicas de jerarquización; en la sección siguiente se ilustra la metodología para la generación automática de código y finalmente, se presentan las conclusiones de la metodología.

II. REDES DE PETRI JERARQUICAS (HPN)

Los primeros modelos en redes de Petri para controladores lógicos se basaron en las redes interpretadas (IPN) y temporizadas [12][13][14], las cuales son la base para la definición de las PN temporizadas interpretadas por periferia (t–PIPN), las t–PIPN permiten modelar conexiones entre el algoritmo y su ambiente, asociando estados y eventos a los modelos y relacionándolos con la periferia del proceso (sensores y actuadores).

Para definir las HPN, se considera inicialmente la definición de las interpretadas [12][15], la cual es modificada para obtener la siguiente tupla (P, T, A, B, M0, I, Q, ω, E,φ, τ), donde P y T son los conjuntos de lugares y transiciones; A y B son las matrices de incidencia previa y posterior; M_0 es el marcaje inicial; I y Q son los vectores de las señales de entrada y salida respectivamente; ω la función de salida; E es una función que asigna a una transición un evento; φ es una función que asigna a una transición una función de conmutación; y τ la función de tiempo.

En un modelo t–PIPN, una transición ti sin retardo dispara si y solo los lugares de entrada de la transición ti tienen al menos el mismo número de marcas, que de arcos que van hacia la transición (sensibilidad) y que el producto de evento por la condición booleana sea diferente de cero (receptividad). Una transición ti retardada dispara después de transcurrido un tiempo τ_i si y solo si está sensibilizada y el producto del evento de fin de temporización y la condición booleana es distinto de cero.

La construcción de modelos t–PIPN complejos puede facilitarse mediante la construcción de redes más pequeñas y simples, implementando dos métodos de jerarquización: la fusión de lugares y la sustitución de transiciones. Estos métodos permiten descomponer un modelo complejo en subredes y módulos como se ilustra en la Figura 1.

Una HPN [15][16][17] se define como la tupla (t–PIPN, S, D, F, G, η, θ, λ, δ) donde S es un conjunto finito de páginas, D y F representan un conjunto con todas las transiciones de sustitución y conjuntos de fusión, G un conjunto de lugares puerto, η la función que asigna una subpágina a una transición de sustitución $\eta(t_i)$, θ la función que asigna a los elementos de G un tipo de puerto in, out o in_out; λ es una función que asigna un lugar puerto a un lugar en la superpágina y δ la función que asigna un tipo conjunto de fusión a un conjunto de lugares.

La Fusión de lugares es un conjunto de lugares considerados para ser idénticos, es decir, ellos representan un solo lugar conceptual y se define de manera formal como sigue:

$\forall p_i \in F: \exists\ p_j \in F / {}^\bullet p_i \cap {}^\bullet p_j = \phi \wedge \forall\ p_i^\bullet \cap p_j^\bullet = \phi \wedge\ \omega(p_i) = \omega(p_j) \wedge\ M(p_i) = M(p_j)$

donde $\bullet p$ y $p\bullet$ representan los conjuntos de transiciones de entrada y salida a un lugar respectivamente.

Figura 1. Red de Petri jerárquica.

Gráficamente, los conjuntos de fusión de lugares se etiquetan con "FG" y "MF" para módulos instanciados por FG. La Figura 2 muestra un ejemplo de un conjunto de fusión de lugares "FG".

Sustitución de transiciones. Una subred se define como una red asociada con una transición ti denotada por HS y asignada por la función η(ti). La subred es en sí misma una t–PIPN con las siguientes consideraciones y restricciones:

- Existe exactamente un lugar de entrada $\mathbf{P_{in}}$ y exactamente un lugar de salida $\mathbf{P_{out}}$.

Figura 2. Fusión de lugares.

- La subred es pasiva, es decir, mientras la subpágina asociada con la sustitución de transiciones no esté habilitada, las transiciones dentro de la subred no están habilitadas y la subred no influencia ninguna señal de salida.
- Los conjuntos de señales de entrada y salida de la subred son subconjuntos de los respectivos conjuntos en la HPN.
- Una subred presenta las siguientes propiedades:

$\exists\, p_{in} \in P_s$ con $\bullet p_{in} = \phi$, $\exists\, p_{out} \in P_s$ con $p^{\bullet}_{out} = \phi$,
$\forall p \in \bullet t_i : M(p) = 0 \Leftrightarrow M(p_{in}) = 0 \wedge M(p) = 1 \Rightarrow M(p_{in}) = 1$,
$\forall p \in t_i^{\bullet} : M(p) = 0 \Leftrightarrow M(p_{out}) = 0 \wedge M(p_{out}) = 1 \Rightarrow M(p) = 1$, $I_s \subseteq I$, $Q_s \subseteq Q$, $P_s \cap P = \phi$, $T_s \cap T = \phi$.

Las transiciones que contienen las subredes se representan gráficamente mediante un rectángulo ampliado y con una etiqueta (HS) junto con el nombre de la subred y la relación de los lugares que habilitan y deshabilitan la HS con pin y pout respectivamente, para distinguirlas de las transiciones de la t–PIPN. La sustitución de transiciones no puede tener asociado una función φ. Para una subred, el lugar de entrada se hace visible por una etiqueta "Pin". De igual forma el lugar de salida se etiqueta con "Pout". El prelugar y el postlugar de una HS serán llamados socket de entrada y de salida respectivamente [17]. La Figura 3 muestra un ejemplo de la representación gráfica.

Figura 3. HPN con sustitución de transiciones y fusión de lugares

III. MODULARIZACION Y REUSABILIDAD

Un modelo de un automatismo secuencial se puede descomponer de manera jerárquica, mediante una red principal, módulos y subredes.

La red principal es una t–PIPN que administra los llamados a las otras componentes y constituye el nivel de jerarquía más alto. Puede contener módulos o subredes de menor jerarquía.

Los módulos son redes que realizan tareas específicas y pueden ser de tres tipos: módulos, módulos reutilizables y módulos reutilizables instanciados por FG. Las subredes son redes instanciadas por HS y pueden ser reutilizables o no reutilizables.

En la Tabla 1 se indica la simbología para los diferentes niveles de jerarquía. Este símbolo se ubica en la esquina superior izquierda de la página que representa el modelo. En la página de la red principal se coloca el símbolo de la red principal y debajo de éste, los nombres de las subredes y módulos de menor jerarquía, como puede verse en la Figura 4.

Tabla 1. Símbolos de los diferentes niveles de jerarquía de una HPN

Modelo	Símbolo
Red principal	P
Subred	S
Subred reutilizable	SR
Modulo	M
Modulo instanciado por	MF
Modulo reutilizable	MR

La red principal tiene asociada una tabla que contiene las funciones de entrada y de salida correspondientes a las transiciones y lugares, tal como se muestra en la

Tabla 2. Para las transiciones interpretadas, la función contiene la receptividad y para las transiciones jerárquicas, la función contiene el símbolo HS. Para los lugares, la función contiene el set (S) y el reset (R) de la salida correspondiente.

Figura 4. Representación gráfica de la red principal.

Las subredes tienen asociada una tabla de funciones de entrada y de salida correspondientes a las transiciones y lugares, conservando para ello la estructura de la Tabla 2.

Tabla 2. Tabla de funciones para la red principal de la Figura 4.

Tran.	Función		Lugar	Función
	$\mathcal{E}(t_i)\bullet\varphi(t_i)$	$\tau(t_i)$		$\omega(p_i)$
t_1	i_0	--	p_1	Rq_0
t_2	--	--	p_2	Sq_1
t_3	$i_1 \bullet i_2$	--	p_3	Sq_2
t_4	i_3	--	p_4	Rq_2
t_5	i_0+i_4	--	p_5	Rq_3
t_6	--	k_τ	p_6	Rq_1

Las subredes reutilizables son instanciadas por varias HS y tienen asociada una tabla que contiene las funciones de entrada y de salida en función de variables genéricas de tipo IN, OUT, IN_OUT definidas por la norma IEC. En la Tabla 3 se ilustran las funciones para la red de la Figura 5.

Figura 5. Subred reutilizable.

Tabla 3. Funciones para la subred reutilizable SR1

Tran.	Función		Lugar	Función
	$\mathcal{E}(t_i)\bullet\varphi(t$	$\tau(t_i)$		$\omega(p_i)$
t_{16}	X_1	--	p_{in}	SY_1
t_{17}	X_2	--	p_{out}	RY_2
t_{18}	X_3	--	p_{16}	SY_3
t_{19}	--	X_4	p_{17}	RY_4
t_{20}	X_5	--	p_{18}	SY_5

La Tabla 4 ilustra la funciones de jerarquía pertenecientes a la página principal ilustrada en la Figura 4.

Tabla 4. Funciones de jerarquía

Lugar	Función		Tran.	Función		
	$\theta(t_i)$	$\delta(p_i)$		$\eta(t_i)$	$\lambda(t_i)$	
					Socket	Puert
p_2	IN	--	t_2	S1	p_2	p_{20}
p_{20}	IN	--			p_3	p_{21}
p_3	OUT	--				
p_{21}	OUT	--				
p_4	--	M@1				

Además, la subred reutilizable tiene una tabla que se denomina "tabla de direccionamiento indirecto" en la cual se encuentran los parámetros correspondientes a las variables (IN, OUT, IN_OUT) para cada instancia como se muestra en la Tabla 5.

Tabla 5. Direccionamiento indirecto para la SR1

Variables	$SR1_1$	$SR1_2$	Variables	$SR1_1$	$SR1_2$
IN			**OUT**		
X_1	i_0	i_5	Y_1	q_0	q_5
X_2	i_1	i_6	Y_2	q_1	q_6
X_3	i_2	i_7	Y_3	q_2	q_7
X_4	i_3	i_8	Y_4	q_3	q_8
X_5	i_4	i_9	Y_5	q_4	q_9
IN_OUT					
P_{IN}	p_2	p_{20}			
P_{OUT}	p_3	p_{21}			

Cuando una HS este relacionada con una SR, el nombre de instancia de la SR deberá llevar un subíndice para reconocer la parametrización que le corresponde. Para dos llamados de SR1 se tiene la Tabla 5 de direccionamiento indirecto. Las funciones asociadas a los módulos conservan la estructura de la Tabla 2.

La tabla de direccionamiento indirecto de los módulos reutilizables instanciados por FG es similar a la de direccionamiento indirecto de los módulos reusables con la diferencia de que su lugar de inicio (MF_i) deberá estar en la tabla de direccionamiento como variable tipo IN_OUT que lo relacionará con el MF que lo invoca. Para salir de un módulo de este tipo, debe hacerse por medio de un FG.

IV. GENERACIÓN DE CÓDIGO IEC.

A. *Unidades de organización de programa (POUs).*

El estándar IEC 61131-3 describe los programas, funciones y bloques funcionales (Function Blocks, FBs) como diferentes tipos de POUs (Program Organization Units), y pueden ser utilizados varias veces a lo largo de las distintas partes que componen una aplicación. Además, la IEC 61131-3 provee lenguajes estandarizados y métodos de ejecución de programas para que un amplio rango de problemas tecnológicos pueda ser programado como elementos de software independientes del fabricante.

Para la generación automática de código, se debe realizar la interpretación de cada elemento que compone la HPN: red principal, subredes y módulos y de los lugares y transiciones, como se muestra en la Figura 6.

Figura 6. Interpretación de una HPN a POUs basado en IEC 61131-3.

Un POU tipo programa es la equivalencia de red principal y un POU tipo bloque de función es la equivalencia de subredes y módulos.

Para la generación de código texto se extrae de la norma IEC 61131-3 un formato de programa como se ilustra en la Figura 7.

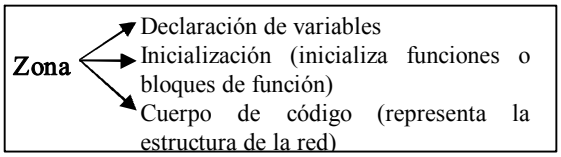

Figura 7. Formato de programa.

La Figura 8 ilustra el formato para la generación de código conforme a la norma 61131-3.

Figura 8. Formato para la generación de código IEC 61131-3.

B. *Codificación de la dinámica de la red.*

La dinámica de la red de una HPN se garantiza al realizar la transformación de transiciones. Este es el primer paso que se debe llevar a cabo en la generación de código. Este paso consiste en la correspondencia uno a uno de los elementos que componen la red con el segmento de código desarrollado, tal como se muestra en la Figura 9.

LD	Lugar_1	Pre_lugar
AND	i_1	Condición de disparo
S	Lugar_2	Post_lugar
R	Lugar_1	Pre_lugar

Figura 9. Codificación de la dinámica de la red.

Asignación de acciones. Después de tener toda la estructura dinámica de la red traducida a lenguaje texto IEC, se procede a realizar la asignación de acciones asociadas a las variables correspondientes a los lugares de la red como ilustra en la Figura 10.

LD	Lugar_1	Si el lugar_1 esta marcado
S	Y_1	se setea la salida 1 y
R	Y_2	se resetea la salida 2

Figura 10. Asignación de acciones

C. *Llamado de módulos y subredes.*

Subredes. Debido al comportamiento dinámico de las subredes asociadas a HS y que tanto el socket de entrada a la HS y el Pin como el socket de salida y el Pout se comportan como FGs, el Pin y el Pout se deben declarar como variables IN-OUT dentro del correspondiente bloque de función que contenga la subred. Dada la definición de pasividad de las subredes, el código debe garantizar su existencia. Esto se logra declarando una variable que condicionará el bloque de función asociado a la subred, esta variable se activara cuando se active el socket de entrada y se desactivara cuando se active el socket de salida. La Figura 11 ilustra el código en lógica de contactos para el llamado de subredes.

Módulos. Los módulos (M), los módulos reutilizables (MR) y los módulos reutilizables instanciados por FG (MF), no estarán condicionados, simplemente los bloques de función asociados a estos serán llamados con CALL. Para los módulos reutilizables asociados a FG (MF) se debe declarar dentro del bloque de función asociado al módulo una variable de entrada FG_i, (tipo IN_OUT), la cual tendrá asociado el lugar FG que invoca el módulo, igualmente se debe declarar dentro del bloque de función asociado al módulo una variable de salida FGo, (tipo OUTPUT), la cual tendrá asociado el lugar FG que retorna la marca a otra POU. La Figura 12 ilustra el llamado de los módulos MF.

Figura 11. Llamado de subredes.

Figura 12. Llamado de módulos MF.

CONCLUSIONES Y FUTUROS DESARROLLOS

Se ha presentado una metodología para generar de manera automática el código para PLC, de conformidad con la norma IEC 61131-3. A partir de un modelo del algoritmo de control en redes de Petri jerárquicas, se presentan las reglas para la codificación del programa en lenguaje de instrucciones, para garantizar así la portabilidad.

La correspondencia uno a uno de los lugares, los módulos, subredes y las transiciones en la red jerárquica, con variables, los POUs y los segmentos en el código respectivamente, con el modelo formal del programa de PLC, permiten la fácil reinterpretación y modificación del código, demostrando así que las HPN son un método formal robusto para el diseño de automatismos industriales.

La metodología contiene además métricas de transparencia y criterios de exactitud que no fueron presentados en este trabajo.

Se presentaron también las tablas asociadas a los componentes jerárquicos (módulos y subredes) y se aplicó el concepto de red reusable, las cuales representan un menor tiempo en el diseño y en la verificación y validación del programa generado.

Esta metodología ha sido probada exitosamente en aplicaciones industriales, tales como la automatización de subestaciones de energía eléctrica y la automatización de procesos de recubrimiento metálico.

Futuros trabajos están enfocados hacia la automatización de la metodología y a la explotación de las capacidades de programación de los PLCs en alto nivel, usando como plataforma formal para el modelamiento, las redes de Petri coloreadas jerárquicas.

AGRADECIMIENTOS

Este trabajo se pudo llevar a cabo gracias a la financiación de la dirección de investigación de Universidad Nacional de Colombia – Sede Medellín (DIME).

REFERENCIAS

[1] Norma IEC 61131-3: Programmable Controllers – Programming Languages. Ed.: International Electrotecnical Comission (IEC). http://plcopen.org/pc2/Intro_IEC_61131-3_Spanish.doc

[2] Castillo, J. (1998). Tendencias en arquitecturas de control. 3as Jornadas de Instrumentación y Control de Procesos. Madrid, España.

[3] Jaafar, H. (2004) Advances in logic controllers design. Control, Communications and Signal Processing. 2004. PP.:21 – 24.

[4] Frey, G. y Litz, L. (2000). Formal methods in PLC programming. Proceedings of the IEEE conference on System Man and Cybernetics. pp 2431-2436.

[5] Frey. G. (2002). Formal methods in PLC control demonstrated at a flexible manufacturing line. Proceedings of the 5th IFIP International Conference on Information Technology. pp. 501-508, Mexico.

[6] Uzam, M. y Jones, A.H. (1996) Towards a Unified Methodology for Converting Coloured Petri Net Controllers into Ladder Logic Using TPLL: Part I. Proceedings of International Workshop on Discrete Event System - WODES'96, Edinburgh, UK, pp. 178 – 183.

[7] Fogel, J., Kocian, M. y S.J. (1994) An approach to object_oriented modelling and control of a DEDS. 1st IFAC Workshop on new trends in design of control systems. Smolenice, Slovakia.

[8] Galán, F.J. y Cañete J.M. (2002). Métodos formales orientados a objetos. Departamento lenguajes y sistemas informáticos. Sevilla, España.

[9] Kearney, D. A. y Mlilo N. (1996). Compilation scheme for structured Petri net based software development of PLC controlled discrete event systems. ICARCV '96. Asian Technology Information Program (ATIP). Japan.

[10] Brinksma, E. y Mader, A. (2000). Verification and Optimization of a PLC Control Schedule. 7th SPIN Workshop. pp. 73-92.

[11] Feldmann, K., Colombo, A., Schnur, C., y Stöckel, T. (1999). Feldmann, Klaus. (1999). Specification, design and implementation of logic controllers based on CPN models. IEEE Transactions on control systems technology, Vol. 7, No. 6, pp. 657-665.

[12] Klein, S., Frey, G. y Minas, M. (2003). PLC Programming with Signal Interpreted Petri Nets. Proceedings of the ICATPN 2003, Eindhoven. Springer LNCS 2679, pp. 440-449.

[13] Silva, M. (1985). Las Redes de Petri en la Automática y la Informática. Editorial AC. Madrid, España.

[14] Zhou, M. (1998) Design of industrial automated systems via relay ladder logic programming and Petri nets. IEEE transactions on systems, man and cybernetics-Part C. Vol. 28, No. 1, pp. 137-150.

[15] Frey, G. (2003). Hierarchical design of logic controllers using signal interpreted petri nets. Proceedings of the IFAC AHDS 2003, Francia, pp. 401-406.

[16] Beard R.W. y Saridis, G.N. (1993). A Cost Measure for Efficient Scheduling in Intelligent Machines. Department of Electrical Computer and Systems Engineering Rensselaer Polytechnic Institute.

[17] Jensen, K. (1997). Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volumen 1, Springer-Verlag.

Germán Zapata M. Profesor Asistente de la Universidad Nacional de Colombia – Sede Medellín. Director de la Escuela de Ingeniería Eléctrica y Mecánica. Integrante del GAUNAL: Grupo de Automática de la Universidad Nacional de Colombia, Categoría A de Colciencias. Ingeniero Electricista de la Universidad Nacional de Colombia (1991). Especialista en Gerencia de la Calidad de la Universidad de Antioquia (1997). Magíster en Automática de la Universidad del Valle (2004).

Jhon W Branch B. Profesor Asistente, Universidad Nacional de Colombia sede Medellín. Director de Área Curricular Sistemas y Administración de la Universidad Nacional de Colombia – Sede Medellín. Integrante del GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial, Categoría A de Colciencias. Doctor en Ingeniería: Sistemas e Informática, Universidad Nacional de Colombia-Sede Medellín. (2007). El área de énfasis de su investigación es Inteligencia Artificial, Procesamiento Digital de Imágenes, Reconocimiento de Patrones, Computación Gráfica, Visión 3D, Realidad Virtual y Inteligencia Computacional.

Luis F. Quintero H. Integrante del grupo de Automática de la Universidad Nacional de Colombia-GAUNAL. Ingeniero Industrial de la Universidad Nacional de Colombia (2003). Estudiante de Maestría en Ingeniería de Sistemas, Universidad Nacional de Colombia, Medellín. Participación en proyectos de investigación y desarrollo en la línea de automatización industrial, financiados por Colciencias, Dirección de investigación Medellín-DIME y empresas del sector eléctrico y comunicaciones.

UNIVERSIDAD
NACIONAL
DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS

120 años
TRABAJO Y RECTITUD