Implementación de un Protocolo de Comunicación para el Control de los Movimientos de un Brazo Robot a través del Interfaz Bluetooth de un Teléfono Celular

Almeida P. Marco[1], Roldán M. Elsa[2], Sinche M. Soraya, Msc[3]

Escuela Politécnica Nacional
Quito-Ecuador

Resumen **- El presente trabajo trata del diseño e implementación de un protocolo de comunicación que estandariza el canal de comunicación de un sistema distribuido.**

El sistema distribuido está compuesto por un teléfono celular *Nokia* y el brazo robot *Lego Mindstorm*. El teléfono celular enviará comandos hacia el brazo robot a través de la interfaz inalámbrica *Bluetooth* para controlar su movimiento. El brazo robot deberá tomar una pelota roja o azul según las órdenes del usuario.

Para el diseño del protocolo se utilizó la herramienta *Spin* que permite la validación de modelos para sistemas distribuidos y para su implementación el lenguaje *JAVA*.

I. INTRODUCCIÓN

Un protocolo de comunicación es un conjunto de reglas, formatos y procedimientos que gobiernan la interacción de procesos concurrentes que se ejecutan en varios equipos de comunicación de un sistema distribuido.

Los equipos de comunicación que conforman un sistema distribuido pueden ser distintos, y también pueden estar conectados entre sí por diferentes tipos de redes, por tal razón los objetivos para el diseño de un protocolo de comunicación son:

- Establecer acuerdos para el uso de recursos compartidos entre los equipos de comunicación.
- Formalizar la interacción entre los equipos de comunicación estandarizando el uso del canal de comunicación.

Para cumplir con estos objetivos, el protocolo de comunicación debe ser especificado tomando en cuenta los siguientes aspectos:

- Los servicios que va a ofrecer el protocolo.
- La hipótesis acerca del medio donde se ejecutará el protocolo.
- Los tipos de mensajes usados para implementar el protocolo (semántica).
- El formato de cada mensaje (sintaxis).
- Las reglas y procedimientos para garantizar la consistencia en el intercambio de datos (gramática).

Debido a que un protocolo de comunicación considera aspectos como la semántica, sintaxis y gramática para el intercambio de información, se puede decir que su definición es similar a la de un lenguaje.

Dado el creciente desarrollo en las comunicaciones inalámbricas y las investigaciones que se han venido realizando para controlar dispositivos electrónicos mediante la tecnología inalámbrica *Bluetooth*, se presenta este proyecto como un aporte a estas investigaciones, implementando un protocolo de comunicación que permita el intercambio de información entre un brazo robot y un teléfono celular de una manera formal.

II. EL MODELO DE REFERENCIA *OSI* Y LA TECNOLOGÍA INALÁMBRICA *BLUETOOH*

A. El modelo de referencia OSI

Para diseñar, implementar y administrar protocolos de comunicación existen modelos que proporcionan un marco teórico y tecnológico. Típicamente se basan en una estructura por capas, lo que permite dividir las distintas tareas que realizará el protocolo de comunicación en módulos. Cada módulo tendrá la capacidad de realizar una sub tarea y de interactuar con otros módulos.

El modelo *OSI* (*Open Systems Interconnection*) fue diseñado por la *ISO* (*International Organization for Standarization*) en 1997 con el objetivo de proporcionar una directriz conceptual para que, equipos que tengan

diferentes características de *hardware*, *software*, sistemas operativos y protocolos puedan comunicarse.

Este modelo divide a todo el proceso de comunicación en varias funciones, las mismas que se encuentran distribuidas en siete capas como se observa en la Fig.1.

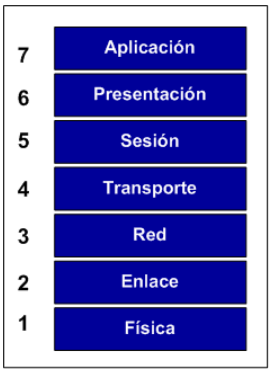

Fig. 1. El modelo de referencia OSI

En cada capa, un proceso que se encuentra en una máquina se comunica con su proceso par en otra máquina como se muestra en la Fig. 2. Según la terminología de la *OSI*, los procesos que se ejecutan en una capa n se les denominan entidades de capa n. El intercambio de información entre entidades se realiza utilizando *PDUs* (*Protocol Data Units*). Cada *PDU* está compuesta por una cabecera que contiene la información de control y por la información de usuario ó *SDU* (*Service Data Unit*). El comportamiento de las entidades de capa n es administrado por un protocolo de capa n.

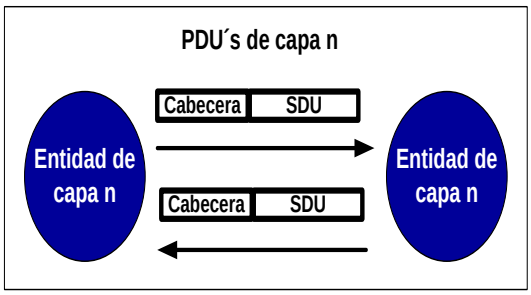

Fig. 2. Comunicación entre dos entidades pares de capa n

La comunicación entre procesos pares es virtual en el sentido de que no existe un enlace físico entre ellos. Para que la comunicación tenga lugar, la entidad de la capa n + 1 hace uso de los servicios ofrecidos por la capa n. La forma en que una capa solicita un servicio a otra capa es a través de primitivas.

Una primitiva especifica una operación o una acción que va a ocurrir. Puede ser una solicitud de un determinado servicio, o una indicación de que una determinada acción o evento, ha sucedido. Existen cuatro primitivas que son:

- ***Request*:** se utiliza cuando una entidad requiere un servicio.
- ***Indication*:** se utiliza para informar a una entidad que una acción o evento tuvo lugar.
- ***Response*:** respuesta de la entidad ante un evento o acción que ocurrió.
- ***Confirm*:** reconocimiento de que una solicitud anterior se ha concedido.

La Fig. 3 muestra como las entidades de capa n + 1 solicitan los servicios de las entidades de capa n por medio de las primitivas para comunicarse con sus entidades pares:

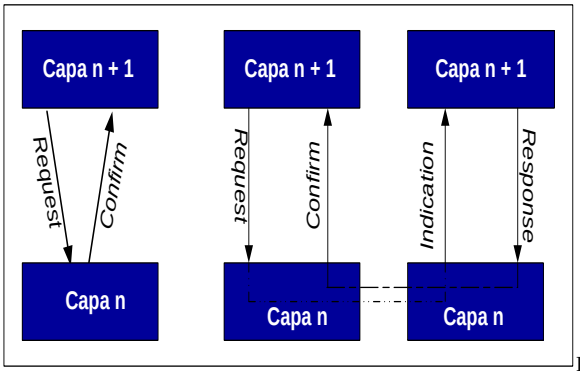

Fig. 3. Comunicación entre capas usando primitivas

La transmisión de la *PDU* de la capa n + 1 se realiza a través de un puerto *software* que pertenece a la capa n denominado *SAP* (*Service Access Point*) como se muestra en la Fig. 4.

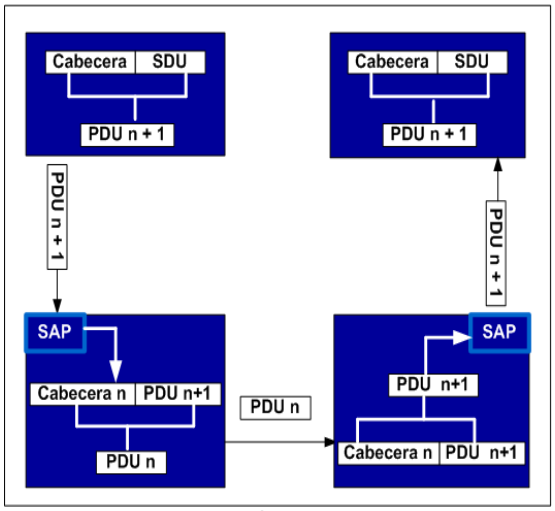

Fig. 4. Comunicación entre entidades pares

B. Tecnología Inalámbrica Bluetooth

La Tecnología Inalámbrica *Bluetooth* es un sistema de comunicación de corto alcance destinado a sustituir los cables de conexión entre dispositivos electrónicos portables o fijos.

Soporta voz y datos, permitiendo a los dispositivos trasmitir cualquier tipo de contenido.

Opera en la banda *ISM* (*Industrial, Scientific and Medical)* de 2.4 GHz, que no requiere de licencia para su uso y se encuentra disponible a nivel mundial.

1) Pila de protocolos de la Tecnología Bluetooth

A la pila de protocolos *Bluetooth* se la puede dividir en los siguientes grupos:

- **Protocolos del núcleo de *Bluetooth*** : Radio, Banda Base, Protocolo de Administración de Enlace (*LMP*), Protocolo de Control y Adaptación de Enlace Lógico (*L2CAP*), Protocolo de Descubrimiento de Servicio (*SDP*).
- **Protocolos sustitución de cable**: *RFCOMM* (Comunicación por radio frecuencia)
- **Protocolos adoptados:** *PPP*, *UDP*, *TCP*, *IP*, *OBEX*, *WAP*, *WAE*
- **Protocolos de control de telefonía**: *TCS*-Binario, Comandos *AT*.

En la Fig. 5 se presenta una comparación entre la pila de protocolos *Bluetooth* y el modelo de referencia OSI.

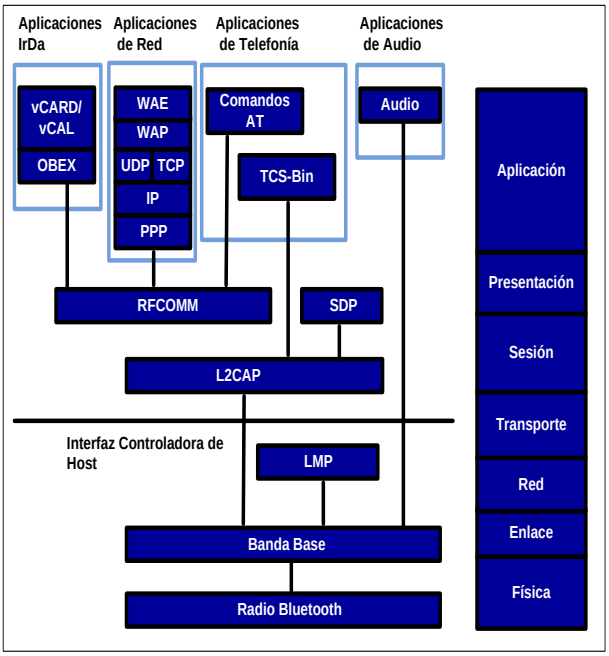

Fig. 5. Comparación entre *Bluetooth* y *OSI*

2) Clases de dispositivos Bluetooth

Bluetooth define tres clases de dispositivos de acuerdo a la potencia de transmisión como se muestra en la Tabla I

TABLA I
CLASES DE DISPOSITIVOS BLUETOOTH

Clase	Potencia de salida máxima	Potencia de salida nominal	Potencia de salida mínima	Alcance
1	100 mW (20 dBm)	--	1 mW (0dBm)	100 m
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (-6 dBm)	10 m
3	1 mW (0 dBm)	--	--	1 m

3) Perfiles Bluetooth

Para garantizar la interoperabilidad entre productos y aplicaciones *Bluetooth* de diferentes fabricantes la especificación *Bluetooth* define un conjunto de perfiles.

Cada perfil representa un posible escenario de uso en el que dos o más dispositivos con tecnología *Bluetooth* deben interactuar para proporcionar al usuario un determinado servicio. En cada perfil se definen los protocolos a utilizar y los procedimientos a seguir en distintos escenarios de aplicación.

Todos los dispositivos *Bluetooth* deben soportar el *GAP* (*Generic Access Profile*), a partir de éste se derivan los demás perfiles.

El perfil utilizado por los dispositivos del sistema es el perfil *SPP* (*Serail Port Profile*) que permite establecer una conexión serial emulada entre dispositivos *Bluetooth*.

III. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

A. Metodología para desarrollo de software en espiral

Este modelo permite realizar una representación real del sistema mediante una serie de prototipos estratégicos y análisis de riesgos de cada uno de ellos a lo largo del ciclo de vida del desarrollo de la aplicación *software*.

El modelo en espiral se muestra en la Fig. 6.

La función que cumple cada etapa se describe a continuación:

- **Requerimientos :** en esta etapa se requiere que el desarrollador plantee los requerimientos del sistema en detalle

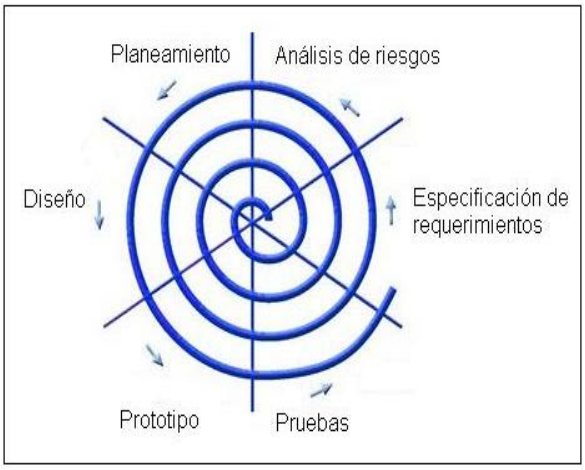

Fig. 6. Etapas que conforman el modelo en espiral

- **Análisis de riesgos:** se identifica todos los riesgos que deberá enfrentar el desarrollador para cumplir con todos los requerimientos que fueron planteados en la primera etapa
- **Planeamiento:** el desarrollador planteará una estrategia que permita la implementación de la aplicación *software* de manera breve y eficiente
- **Diseño:** se procede a la escritura del programa
- **Primer prototipo del sistema:** se construye el primer prototipo del sistema, el mismo que se debe aproximar a las características finales del sistema ya que es la base para los siguientes prototipos
- **Pruebas:** el desarrollador prueba el primer prototipo y si este no cumple con ciertos requerimientos, entonces este procederá al análisis del segundo prototipo siguiendo los mismos pasos anteriores

Este modelo se utiliza particularmente en sistemas embebidos por su fortaleza en el análisis de riesgos para cada prototipo que se implementa. Esta etapa permite que el resultado sea una aplicación *software* robusta.

1) Especificación de requerimientos

a) Especificación de la hipótesis del medio

Para cumplir con sus funciones, el protocolo de comunicación debe comunicarse con su ambiente, que está compuesto por los elementos que se muestran en la Fig. 7 y son:

- El usuario
- El brazo robot *NXT Lego Mindstorm*
- El teléfono celular (Nokia 5220)
- El canal de comunicación

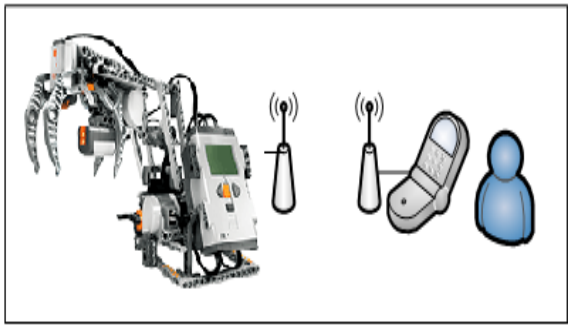

Fig. 7 Elementos del sistema distribuido

El usuario podrá manipular tanto al brazo robot como al teléfono celular para iniciar la fase de establecimiento de la conexión, transferencia de información, y finalización de la conexión.

Los mensajes del protocolo de comunicación serán intercambiados sobre un canal de comunicación inalámbrico *full duplex*, el mismo que será configurado utilizando el perfil *SPP* (*Serial Port Profile*) de la tecnología inalámbrica *Bluetooth* versión 2.0 con *EDR* (*Enhanced Data Rate*).

El enlace alcanza una velocidad efectiva asíncrona entre 108.8 Kbps y 2.1 Mbps, dependiendo del tamaño del mensaje.

El *BER* (*Bit Error Rate*) para la tecnología *Bluetooth* versión 2.0 con *EDR* es de 0.01% y en modo básico (sin *EDR*) 0.1%.

La Tabla II muestra las características, respecto a la tecnología *Bluetooth*, de los equipos de comunicación que se emplearán.

Se debe tomar en cuenta para el diseño del protocolo de comunicación que la tecnología *Bluetooth* utiliza las siguientes características:

- En la capa banda base se encuentra implementado el método de control de flujo *stop and wait* con *ARQ* y un temporizador denominado *flush time out*. Por defecto está configurado para realizar un número indefinido de retransmisiones hasta recibir un acuse de recibo positivo del paquete enviado. *JSR-82* no permite modificar ese parámetro.
- *Bluetooth* utiliza polinomios *CRC* de orden 16 para los distintos tamaños de mensajes. Este factor produce la posibilidad de que algunos mensajes errados no sean detectados por dicho polinomio debido a la distancia de *Hamming* y pasen a capas superiores.

TABLA II
CARACTERÍSTICAS DE LOS EQUIPOS DE COMUNICACIÓN

Característica	Brazo robot	Nokia 5220
Bluetooth v2,0 + EDR	**Si**	**Si**
Clase de dispositivo	**2**	**2**
Serial Port Profile	**Si**	**Si**
JSR 82	**Si**	**Si**

b) Especificación de los servicios del protocolo de comunicación

El protocolo de comunicación para la transmisión de la información será orientado a la conexión y ofrecerá los siguientes servicios:

- Preservación de secuencia
- Sincronización de unidades de datos orientado a bloques
- Detección y control de errores
- Control de flujo

El protocolo de comunicación deberá ser capaz de enviar un comando a la vez, controlar posibles errores que provengan de las capas inferiores debido al polinomio *CRC*-16 que utilizan, y de limitar el número de retransmisiones del temporizador *flush time out*. Por lo tanto se utilizará para el control de flujo, el método *stop and wait* con la posibilidad de configurar el número de retransmisiones según sea necesario, combinado con la técnica *ARQ* para el control de errores. La técnica *ARQ* implementará *CRC* para la detección de errores.

2) Análisis de riesgos del sistema

Los riesgos para implementar los requerimientos anteriores además de ciertas consideraciones son:

- Todo protocolo de comunicación se desempeña en un ambiente multiproceso, por tal razón es necesario que se estudie en profundidad toda la problemática referente a procesos
- El lenguaje de programación *JAVA* utiliza procesos ligeros para permitir la creación de sistemas multitarea, por tal razón, se debe tener cuidado ya que éstos comparten recursos entre sí de una manera no determinística
- Todos los hilos de ejecución que se produzcan debido al protocolo de comunicación deben ejecutarse una sola vez durante el ciclo de vida del protocolo de comunicación.
- La capacidad de procesamiento del *NXT brick* y su capacidad de memoria es baja. Estas características limitan el uso excesivo de hilos de ejecución dentro de este equipo
- El protocolo de comunicación será implementado en base al modelo de referencia *OSI*, en donde los módulos que se implementen en el teléfono celular podrían re utilizarse en el brazo robot bajo pequeñas diferencias. Para lograr este objetivo, se debe observar que ambos equipos trabajen con librerías de *JAVA* comunes.
- El tiempo de procesamiento de los mensajes debe ser mínimo para que los equipos consuman menos recursos y respondan de una manera rápida
- El *NXT brick* no dispone para los teléfonos celulares de las bases de datos de servicios que ofrece *J2ME*, por tal razón se debe proporcionar un método especial para el establecimiento de la conexión
- El acceso a los servicios a través del *SAP* debe ser inmediato e independiente de la pantalla en la que se encuentre el usuario. Esto facilitará al desarrollar la escritura de los programas que formen parte del sistema

3) Planeamiento y Diseño para la implementación del sistema

La implementación del sistema estará dividida en tres fases que corresponden al ambiente donde se desempeñará el protocolo de comunicación:

- El protocolo de comunicación
- La interfaz gráfica de usuario
- El programa para el *NXT brick*

a) Protocolo de comunicación

a.1) Diseño del modelo del protocolo de comunicación

Previo a la implementación del protocolo de comunicación, se sugiere la creación de un modelo del mismo debido a la problemática que envuelve a los sistemas distribuidos.

La creación de este modelo comprende las siguientes etapas:

- Especificación de los tipos de mensajes que se intercambiarán durante la simulación

- Especificación de las reglas y procedimientos para el intercambio de datos
- Escritura del programa en base a las condiciones anteriores en lenguaje *PROMELA* (*Process Meta Language*) para verificar su validez

a.2) La Herramienta SPIN para la validación de modelos de protocolos de comunicación escritos en lenguaje PROMELA

SPIN es una herramienta *software* que soporta el análisis y verificación de sistemas asíncronos, concurrentes y distribuidos. Para la creación del modelo del sistema se utiliza los fundamentos de la notación matemática *CSP* (*Communicating Sequential Processes*) de *Hoare* y para su descripción un lenguaje de alto nivel llamado *PROMELA*. La sintaxis de este lenguaje es derivada del lenguaje C.

Un sistema en *PROMELA* está compuesto por los siguientes componentes:

- Procesos
- Canales sincrónicos y asincrónicos
- Variables.

La concurrencia de los procesos es asíncrona y modelada mediante el intercalado de instrucciones.

Dado un modelo escrito en *PROMELA* como entrada, *SPIN* genera un programa en C que realiza la verificación del sistema mediante la enumeración de su espacio de estados, también conocido como estados de un programa. En caso de que el modelo presente errores, *SPIN* genera los denominados contra ejemplos, que permiten al programador saber cómo y dónde el modelo falla al momento de satisfacer una propiedad específica.

Al momento de realizar la simulación del protocolo de comunicación se comprobó que está libre de los siguientes problemas:

- **Exclusión mutua:** Las secuencias de instrucciones de las secciones críticas de dos o más procesos no se deben intercalar.
- **Ausencia de *deadlocks*:** Si algunos procesos están tratando de entrar en su sección crítica, entonces al menos uno de ellos debe tener éxito.
- **Libertad de iniciación:** Si cualquier proceso intenta entrar en su sección crítica, entonces ese proceso debe tener éxito.

a.3) Implementación del protocolo de comunicación

El protocolo de comunicación deberá ser diseñado tomando en cuenta principalmente la propiedad de modularidad. Cada servicio que éste ofrecerá deberá corresponder a una clase o a un método, dependiendo de la flexibilidad para futuros cambios.

La Fig. 8 muestra como se podría implementar el protocolo de comunicación utilizando la propiedad de modularidad. Debido a que ambos equipos son sistemas diseñados para un propósito específico, los servicios que ofrecerá el protocolo de comunicación variarán en función de las características de *hardware* ó *software* del equipo.

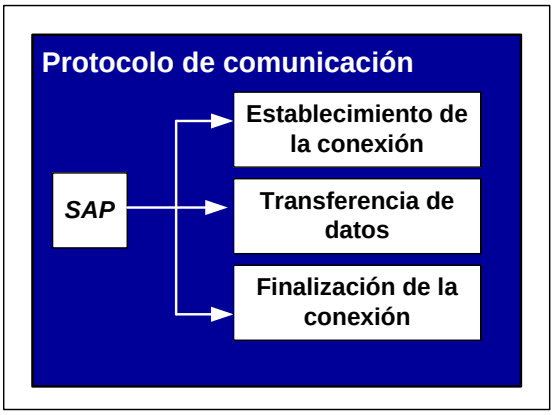

Fig. 8. Protocolo de comunicación utilizando la propiedad de la modularidad de *JAVA*

Cada servicio del protocolo de comunicación estará representado por una clase y agrupados en paquetes.

Debido a que algunos servicios pueden ser hilos de ejecución, la clase protocolo de comunicación será una clase *singleton* para evitar la clonación innecesaria de hilos de ejecución.

La clase PuntoAccesoServicio se encuentra encapsulada en la clase *singleton* ProtocoloComunicacion, la misma que está compuesta por todos los servicios que puede brindar el protocolo de comunicación.

Para solicitar un servicio a la clase PuntoAccesoServicio, se necesita del manejo de las primitivas. Para implementar las primitivas se utilizó el método *CubbyHole*. Este método implementa un *slot* que puede tener un solo objeto a la vez para la comunicación entre hilos. Un hilo coloca un objeto en el *slot* y otro lo toma a través de dos métodos. Si otro hilo trata

de poner un objeto en el *slot* cuando éste se encuentra ocupado, este hilo se bloquea hasta que el *slot* se libere. De esta manera, cualquier clase podrá solicitar un servicio al *SAP* de una manera segura.

Los mensajes se clasificarán en:

- **Mensajes de datos:** Pueden ser de ejecución de comando, color, y fin de conexión
- **Mensajes de control:** Pueden ser de *NACK*, *ACK* de ejecución de comando, *ACK* de color, y *ACK* de fin de conexión

La información que conformará la cabecera se presenta en la Fig. 9 y estará definida por los siguientes criterios:

- Tp (1 bit) para identificar el tipo de mensaje
- Sec (1 bit) para establecer el número de secuencia del mensaje para el control de flujo
- A/N (1 bit) para identificar si se trata de un *ACK* o un *NACK*
- Clase (2 bits) para identificar el tipo de mensaje de control
- Tx (1 bit) para identificar el origen del mensaje, debido a que un mensaje de datos podrá ser utilizado para distintos propósitos, dependiendo de quién envíe el mensaje
- F (1 bit) para indicar cuándo se debe finalizar la conexión

Los valores de los bits que conforman la cabecera del protocolo de comunicación y su descripción se presentan en la Tabla III.

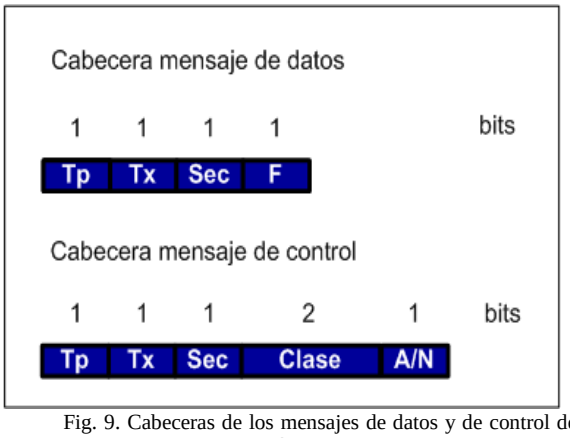

Fig. 9. Cabeceras de los mensajes de datos y de control del protocolo de comunicación

TABLA III
VALORES Y DESCRIPCIÓN DE LOS BITS QUE CONFORMAN LA CABECERA DEL PROTOCOLO DE COMUNICACIÓN

Bit	Valor	Descripción
Tp	0	Mensaje de datos
	1	Mensaje de control
Tx	0	Transmite el teléfono celular
	1	Transmite el brazo robot
Sec	0	Número de secuencia
	1	Número de secuencia
F	0	No finaliza la conexión
	1	Finaliza la conexión
Clase	00	*NACK*
	01	*ACK* de comando
	10	*ACK* de fin de conexión
	11	*ACK* de mensaje de color
A/N	0	*ACK*
	1	*NACK*

La carga útil estará definida por los siguientes criterios:

- Cinco bits que se utilizarán para indicar el comando para: ejecución del movimiento, color de la pelota leída por el brazo robot, o para configurar el color de la pelota por defecto para que el brazo robot realice la captura de la misma. Cuando los cinco bits de un mensaje de datos se encuentran en cero, significa que se trata de un mensaje de finalización de la conexión.

  La Fig. 10 muestra la carga útil de los mensajes de datos del protocolo de comunicación, y en la Tabla IV se muestran los valores correspondientes que puede tomar la carga útil.

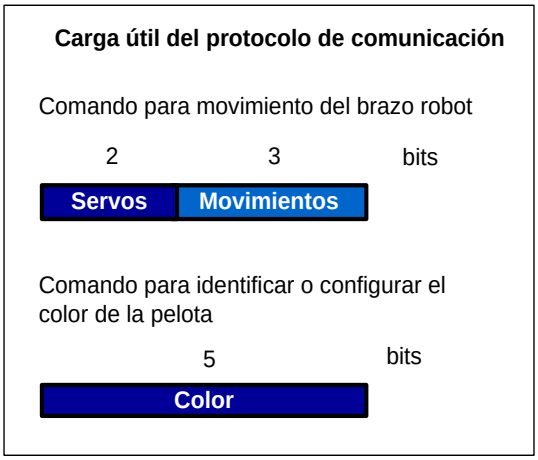

Fig. 10. Carga útil de los mensajes de datos del protocolo de comunicación

La cola estará compuesta por la *FCS* del polinomio *CRC* y estará definida por los siguientes criterios:

- La eficiencia relativa E del protocolo de comunicación, en presencia de errores, según la ecuación (1):

$$E = \frac{d}{R(d+t+a)} \quad (1)$$

Donde:
d = tamaño de la carga útil
t = tamaño de la cabecera y cola, y
a = tamaño del mensaje de control en bits
R = número de transmisiones por mensaje y viene dada por la ecuación (2).

$$R = \frac{1}{1-pr} \quad (2)$$

La variable *pr* de la ecuación (2) representa la probabilidad de que un mensaje sea retransmitido.

TABLA IV
VALORES Y DESCRIPCIÓN DE LOS BITS QUE CONFORMAN LA CARGA ÚTIL DEL PROTOCOLO DE COMUNICACIÓN

Bits	Valor	Descripción
Servos	0 1	Servo A
	1 0	Servo B
	1 1	Servo C
Mov	1 0 1	Mover el servo hacia la izquierda
	0 1 1	Parar es servo
	1 0 0	Mover el servo hacia la derecha
	0 0 1	Mover el servo hacia la arriba
	0 1 0	Mover el servo hacia abajo
	1 1 0	Abrir la mano del brazo robot
	1 1 1	Cerrar la mano del brazo robot
Color	01010	Rojo
	01111	Azul

- El tamaño de un bloque para el mensaje de datos será de un byte debido a la característica de los *streams* del lenguaje *JAVA*
- La distancia de *Hamming*

Reemplazando los criterios antes mencionados, para un *BER* de 0.01% en la ecuación (1), se tiene:

$$E \approx \frac{5}{(15+tc)} \quad (3)$$

tc representa el tamaño de *FCS* a implementar.

La Tabla IV contiene un listado de los polinomios *CRC* más comunes para un tamaño de mensaje recomendado.

El tamaño del mensaje recomendado para los polinomios de la Tabla V se deduce considerando la distancia de *Hamming* de cada polinomio *CRC*.

El polinomio *CCITT*-4 ofrece un buen rendimiento para tramas de diversos tamaños, y además una alta eficiencia debido a los cuatro bits de redundancia que ofrece.

TABLAV
POLINOMIOS *CRC* Y LA EFICIENCIA PARA EL PROTOCOLO DE COMUNICACIÓN UTILIZANDO LA ECUACIÓN (3)

Nombre del polinomio	Orden polinomio	Tamaño mensaje recomendado en bits	Efic %
CCITT-4	4	8 – 2048	26,32
CRC-5	5	8 – 10	25,00
DARC-6	6	12 – 25	23,81
DARC-8	8	8 – 9	21,74
CCITT-16	16	1015 - 2048	16,13

Los mensajes de datos estarán conformados por trece bits y los de control por seis bits, sin embargo, debido a que los *streams* del lenguaje *JAVA* están orientados a bloques de ocho bits, se deben añadir tres y dos bits de relleno a cada mensaje respectivamente.

La eficiencia relativa del protocolo de comunicación, debido a los bits de relleno podría variar entre el 20.8 % y el 33.3%.

b) Interfaz gráfica de usuario

La interfaz gráfica debe permitir el acceso a los servicios del protocolo de comunicación para el control del brazo robot, por tal razón las pantallas que permitirán cumplir con este objetivo son:

- **Presentación:** muestra información respecto al proyecto
- **Búsqueda:** permitirá el ingreso del nombre del brazo robot requerido por *Bluetooth* para iniciar la fase del establecimiento de la conexión
- **Espera servidor:** se muestra mientras se realiza el proceso de establecimiento de la conexión
- **Control robot:** contiene los controles que permiten el control del brazo robot y aparecerá una vez que se halla establecido la conexión

- **Configuración Color:** permite la selección del color de la pelota por parte del usuario para que el brazo robot la tome o no después de identificar el color
- **Espera:** se muestra cuando un acuse de recibo no llegue en un determinado tiempo
- **Log:** mantiene un historial de los mensajes enviados y recibidos en el teléfono celular en decimal y en binario

La implementación de las pantallas para cada fase se muestra a continuación:

Fig. 11. Pantallas para la fase del establecimiento de la conexión: Presentación, búsqueda y espera servidor

Fig. 12. Pantallas para el control y configuración del brazo robot: Control robot y menú Control robot

Fig. 13. Pantalla para la configuración del color de la pelota que el brazo robot tomará por defecto

Fig. 14. Pantallas para las fases de intercambio de datos y de finalización de la conexión en el teléfono celular: Log y la pantalla de cierre de aplicación por defecto del teléfono celular

4) Pruebas del sistema

Las pruebas consistieron en verificar el correcto funcionamiento de los principales servicios del protocolo de comunicación. Las pruebas se realizaron por cinco ocasiones en cada punto. Los resultados que se obtuvieron se encuentran tabulados en la Tabla VI.

TABLAVI
RESULTADO DE LAS PRUEBAS REALIZADAS A LAS FASES DEL PROTOCOLO DE COMUNICACIÓN

	Fase		
Distancia (metros)	**Fase 1[1] (% Éxito)**	**Fase 2[2] (% Éxito)**	**Fase 3[3] (% Éxito)**
9	**100**	**100**	**100**
10	**100**	**100**	**100**
11	**100**	**100**	**100**
12	**100**	**100**	**100**
13	**100**	**100**	**100**
14	**100**	**100**	**100**
15	**100**	**100**	**100**
16	**100**	**100**	**100**
17	**100**	**100**	**100**
18	**60**	**60**	**60**
19	**40**	**40**	**40**

En todas las pruebas el robot reaccionó con un valor promedio de 0.48 segundos de lo que se emitió el comando.

[1] Fase 1 : Establecimiento de la conexión
[2] Fase 2 : Transferencia de datos
[3] Fase 3 : Finalización de la conexión

IV. CONCLUSIONES

- Las herramientas para la validación de modelos minimizan el tiempo de verificación del comportamiento de los procesos concurrentes en un sistema, y también la localización fácil de errores en el mismo.
- Si el modelo no es diseñado de una manera adecuada ó presenta demasiadas características, SPIN podría generar un espacio de estados demasiado grande, con lo cual aumentaría el tiempo de procesamiento.
- El protocolo de comunicación permite manejar e informar de los errores que se pueden producir durante el intercambio de datos entre los equipos.
- El lenguaje J2ME funciona como *middleware* en los diferentes sistemas embebidos, facilitando la manera en que se comunican y el desarrollo de aplicaciones.
- La herramienta *SPIN* facilita el desarrollo de software donde actúen procesos concurrentes, sin embargo el gran limitante de esta herramienta es la capacidad de memoria de los computadores donde se encuentre instalado.
- *SPP* simula la transmisión serial de la interfaz RS232. En esta clase de transmisión los errores de ráfaga son comunes.
- Los servicios del protocolo de comunicación están en función de las características de los dispositivos.

V. BIBLIOGRAFÍA

[1] HOLZMANN, Gerard, "*Design and Validation of Computer Protocols*". Primera edición. Prentice-Hall. New Jersey. 1991.

[2] SHARP, Robin, "*Principles of Protocol Design*". Primera edición. Springer. Berlin. 2008.

[3] BEN ARI, Mordechai, "*Principles of the Spin Model Checker*". Primera edición. Springer. London. 2008.

[4] BEN ARI, Mordechai, "*Principles of Concurrent and Distributed Programming*". Segunda edición. Addison-Wesley. London. 2006.

[5]KNUDSEN, Jonathank, "*Beginning J2ME from novice to profesional*". Tercera edición. Apress. Estados Unidos 2005.

[6]GOYAL, Vikram, "*Pro Java ME MMAPI Mobile Media API for Java Micro Edition*". Primera edición. Apress. Estados Unidos. 2006.

[7]http://www.ece.cmu.edu/~koopman/crc/index.html

[8]SCHOLZ, Matthias, "*Advanced NXT The Da Vinci Inventions Book*". Primera edición.Apress. Estados Unidos. 2007.

[9]BAUM, Dave, "*Extreme Mindostorms*". Primera edición. Apress. Estados Unidos. 2000.

VI. BIOGRAFÍAS

Marco Almeida Pazmiño, nació en Quito el 12 de Abril de 1983, realizó sus estudios secundarios en el Colegio Técnico Aeronáutico de Aviación Civil y en la actualidad se encuentra terminando su proyecto de titulación en la Escuela Politécnica Nacional para obtener el título de Ingeniero en Electrónica y Redes de Información. Participó en los tutoriales *ANDESCON* 2006: "Diseño proyección y evaluación de proyectos de telecomunicaciones" y "Sincronización de redes de telecomunicaciones digitales" organizados por la *IEEE* el 11 de noviembre del 2006. Obtuvo la certificación otorgada por la Empresa S.C. '*PIC* Electrónica y computación por aprobar el curso orientado al manejo de microcontroladores *PIC* en septiembre de 2006. Asistió al seminario "Procesamiento digital de señales e imágenes usando *MATLAB*" dictado en la semana del 19 al 23 de noviembre de 2007 en la Escuela Politécnica Nacional. Cursó sus estudios para certificarse como *CCNA (CISCO Certified Network Associated)* en la Academia *ACIERTE* de la Escuela Politécnica Nacional.

e-mail: malmeidap@hotmail.com

Elsa Roldán Molina, nació en Quito-Ecuador el 24 de Marzo del 1985, Curso sus estudios primarios y secundarios en La Unidad Educativa San Francisco de Sales obteniendo el título de bachiller en Humanidades Modernas especialidad Físico Matemático. Actualmente se encuentra terminando su proyecto de titulación en la Escuela Politécnica Nacional para obtener el título de Ingeniero en Electrónica y Redes de Información. Participo en el curso organizado por el CIEEPI (Colegio de Ingenieros Eléctricos y Electrónicos de Pichincha) de Voz y Video sobre el protocolo INTERNET realizado en Quito, Ecuador en Septiembre del 2007. Cursó sus estudios para certificarse como *CCNA (CISCO Certified Network Associated)* en la Academia *ACIERTE* de la Escuela Politécnica Nacional.

e-mail: elsyjmr24@hotmail.com

Soraya Lucía Sinche Maita, Nace en Loja el 21 de junio de 1974, en Mayo de 1999 obtuvo el título de Ingeniera en Electrónica y Telecomunicaciones en la Escuela Politécnica Nacional. En el año 2004 obtiene el título de *Master of Science* en Sistemas Inalámbricos y Tecnologías Relacionadas en el Politécnico de Torino, Italia. Terminó los estudios en la Maestría de Conectividad y Redes de Telecomunicaciones de la Escuela Politécnica Nacional. En la actualidad se desempeña como Profesor Principal del Departamento de Telecomunicaciones y Redes de Información de la Escuela Politécnica Nacional.

e-mail: soraya.sinche@epn.edu.ec