PRÁCTICA EMPRESARIAL EN LA UEE BIOINGENIERÍA DE LA FUNDACIÓN CARDIOVASCULAR DE COLOMBIA: DISEÑO E IMPLEMENTACIÓN DE DRIVERS DE COMUNICACIÓN Y GEOREFERENCIACIÓN PARA UN DISPOSITIVO MÉDICO DE MONITORIZACIÓN VITAL

FUNDACIÓN CARDIOVASCULAR DE COLOMBIA

MARTIN EDUARDO GUEVARA TARAZONA

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2013

PRÁCTICA EMPRESARIAL EN LA UEE BIOINGENIERIA DE LA FUNDACIÓN CARDIOVASCULAR DE COLOMBIA: DISEÑO E IMPLEMENTACIÓN DE DRIVERS DE COMUNICACIÓN Y GEOREFERENCIACIÓN PARA UN DISPOSITIVO MÉDICO DE MONITORIZACIÓN VITAL

MARTIN EDUARDO GUEVARA TARAZONA

Práctica empresarial para obtener el título de

INGENIERO ELECTRÓNICO

Supervisor académico:
MsC. ALEX ALBERTO MONCLOU

Supervisor empresarial:
Ing. LEONARDO RODRÍGUEZ SALAZAR

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2013

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

TITULO: **PRÁCTICA EMPRESARIAL EN LA UEE BIOINGENIERÍA DE LA FUNDACIÓN CARDIOVASCULAR DE COLOMBIA: DISEÑO E IMPLEMENTACIÓN DE DRIVERS DE COMUNICACIÓN INALÁMBRICA Y GEOREFERENCIACIÓN PARA UN DISPOSITIVO MÉDICO DE MONITORIZACIÓN VITAL.**

AUTOR: **MARTIN EDUARDO GUEVARA TARAZONA**

FACULTAD: **FACULTAD DE INGENIERÍA ELECTRÓNICA**

DIRECTOR: **MsC. ALEX ALBERTO MONCLOU**

La presente práctica se desarrolló en la Unidad Estratégica Empresarial Bioingeniería de la Fundación Cardiovascular de Colombia, la cual desarrolla productos de última tecnología para el bienestar de la comunidad enfocándose en productos del sector salud específicamente en los campos de instrumentación biomédica, electrofisiología, telediagnostico, automatización, procesamiento de señales y todas aquellas áreas orientadas a la mejora de la salud humana. El objetivo de la práctica empresarial consistió en diseñar e implementar drivers para dispositivos de comunicación inalámbrica enfocados a transmitir datos de monitorización vital y georeferenciación en un softcore embebido en una FPGA e integrarlos en un dispositivo médico el cual aun se encuentra en desarrollo.

Teniendo en cuenta un estudio previo de las capas de comunicación de los diferentes métodos de gestión de la información en la red, se dio inicio al diseño de los drivers para los módulos de comunicación basándose en la documentación obtenida de las hojas de datos y notas de aplicación de estos módulos. Inicialmente para la comunicación entre el procesador en la FPGA y los módulos inalámbricos se diseña un modulo UART desde cero en descripción de hardware (HDL). Posteriormente se inicia con la tarea de diseño del driver el lenguaje ANSI C para un módulo Wi-Fi el cual fue configurado a partir de comandos enviados mediante protocolo serial y se implementó de forma satisfactoria en software dentro la FPGA . De la misma forma se diseño y se implemento el driver que controla el módulo de transmisión de datos por Red Celular y Georeferenciación dada las

similitudes en sus modos de configuración mediante comandos y protocolo serial.

Finalmente estos drivers son implementados en el sistema embebido del dispositivo medico de monitorización vital en desarrollo, el cual transmite datos a un servidor Web mediante protocolo Websocket.

Todas las tareas fueron realizadas satisfactoriamente, bajo la supervisión del coordinador de la practica y el jefe de Diseño y Desarrollo, aportando un valioso recurso técnico en el desarrollo del sistema final del dispositivo medico de monitorización vital de desarrollo.

PALABRAS CLAVES:

DRIVERS, ANSI C, SOFTCORE, ETHERNET
USUARIO, UART, HDL, WEBSOCKET,
DIRECCIONES MAC, FPGA,

_____

VoBo DIRECTOR DE TRABAJO DE GRADO

ABSTRACT

TITLE: **BIOENGINEERING SBU PRACTICE IN CARDIOVASCULAR FOUNDATION OF COLOMBIA: DESIGN AND IMPLEMENTATION OF WIRELESS COMMUNICATION DRIVERS AND GEOREFERENCING FOR MEDICAL VITAL MONITORING DEVICE.**

AUTHOR: **MARTIN EDUARDO GUEVARA**

FACULTY: **ELECTRONIC ENGINEERING FACULTY**

DIRECTOR: **MsC. ALEX ALBERTO MONCLOU**

This practice was developed in the Bioengineering Strategic Business Unit in Cardiovascular Foundation of Colombia, which develops products of latest technology for the community welfare, focusing on healthcare products specifically in the fields of biomedical instrumentation, electrophysiology, remote diagnostics, automation, signal processing and all those areas designed to improve human health. The purpose of practice consisted of designing and implementing drivers for wireless communication devices to transmit data for vital monitoring and georeferencing embedded in an FPGA softcore and integrated into a medical device which is still in development.

Considering a previous study of communication layers of different methods of information management in the network, began the driver design for communication modules based in the documentation obtained from the data sheets and implementation notes of these modules. Initially for communication between the processor in the FPGA and wireless modules, a UART module is designed in hardware description (HDL). Later starts with the design task for a driver in ANSI C language for a Wi-Fi module, which was configured from commands sent via serial protocol and successfully implemented in software within the FPGA. In the same way, a driver to control the module to sent data by Red Cell and Georeferencing transmission was designed and implemented, given the similarities in their modes of configuration using commands and serial protocol.

Finally these drivers are implemented on the medical monitoring device embedded system; still under develop, which transmits data to a Web server using WebSocket protocol.

All tasks were performed satisfactorily, under the supervision of the practice coordinator and the head of design and development, providing a valuable technical resource in the development of the final system monitoring medical device development life.

KEYWORDS:

DRIVERS, ANSI C, SOFTCORE, ETHERNET
USUARIO, UART, HDL, WEBSOCKET,
DIRECCIONES MAC, FPGA,

_____

VoBo THESIS DIRECTOR

INTRODUCCIÓN

La UEE (Unidad Estratégica Empresarial) Bioingeniería de la FCV, en pro del crecimiento tecnológico y económico de la región desarrolla productos de última tecnología para el bienestar de la comunidad enfocándose en productos del sector salud específicamente en el área de Bioingeniería.

Diseño y desarrollo es uno de los procesos fundamentales de la UEE Bioingeniería porque dentro de esta división se idean, se prototipan y se desarrollan productos que en años posteriores estarán en constante uso prestando un servicio a toda la región y el país. Este proceso se encarga de proponer, diseñar y ejecutar proyectos de innovación y desarrollo tecnológico que permitan solucionar problemas en el área de la salud, especialmente en los campos de la instrumentación biomédica, electrofisiología, telediagnóstico, automatización, procesamiento de señales y todas aquellas áreas orientadas a la mejora de la salud humana.

Actualmente con la masificación de dispositivos móviles, *apps*, el uso constante de TIC's y las telecomunicaciones como medio para estar permanentemente informados de lo que se sucede en entornos remotos, el proceso de diseño y desarrollo dota a sus equipos electrónicos próximos a salir al mercado de medios de conectividad a la red. Cada uno de estos medios de conectividad debe cumplir ciertos requerimientos dependiendo del tipo de servicio prestado bien sea chat, video, voz, monitorización remota de parámetros fisiológicos en unidades móviles, etc. Este último (por el momento) es el servicio que se implementará en el dispositivo médico en desarrollo el cual podrá transmitir los parámetros fisiológicos de forma remota por medios como Ethernet, Wi-Fi y Redes celulares (basadas en tecnología 3G). Estos parámetros se podrán visualizar de forma gráfica en navegadores web de dispositivos móviles ó computadores convencionales. El desarrollo realizado por el practicante consistió en diseñar e implementar drivers para los dispositivos Wi-Fi y 3G-GPS los cuales se controlan desde un softcore embebido en una FPGA e integrarlos con el sistema ya desarrollado. Se debe tener en cuenta que estos drivers debieron ser implementados en el dispositivo médico de una forma transparente, permitiendo enviar datos de la misma forma en la que ya se encontraba implementado con el modulo Ethernet.

Como tarea adicional se solicito una cotización a la IEEE acerca del costo de diferentes direcciones MAC para dispositivo Ethernet las cuales son requeridas para que el dispositivo cuente con una dirección física única y pueda ser comercializado. Estas direcciones garantizan un correcta ejecución del protocolo DHCP usado para adquirir una IP y así permitir que el modulo pueda conectarse a internet.

Inicialmente no se contaba con un módulo UART (en descripción de hardware) eficiente a altas velocidades por lo cual se diseño un nuevo módulo UART (acelerador de hardware) descrito en Verilog HDL (Hardware Description Language) para la comunicación entre el procesador (softcore) NIOSII embebido dentro de la FPGA, y cada uno de los módulos de comunicación inalámbrica. Este

módulo fue de vital importancia para iniciar con el diseño de los driver que finalmente lograrian comunicar el NIOSII con los módulos Wi-Fi y 3G.

En la fecha de inicio de la práctica se encontraba implementado en el dispositivo médico la transmisión remota de datos de monitorización vital por Ethernet y protocolo Websocket. Estos datos eran recibidos e interpretados por un servidor local implementado por ingenieros de sistemas del Departamento de Tecnologías de la Información y Sistemas (DTICS) de la FCV, el cual presta el servicio de monitorización a diferentes clientes mediante protocolo Websocket.

1. DESCRIPCIÓN DE LA EMPRESA

1.1 RESEÑA HISTÓRICA

En 1985 un grupo de especialistas y personalidades de Bucaramanga se propuso crear una entidad privada sin ánimo de lucro dedicada a tratar las enfermedades del corazón, logrando que un grupo de médicos iniciara las actividades de consulta y prueba de esfuerzo en la Fundación Tercera Edad de la Congregación Mariana, y las primeras cirugías cardiovasculares en la Clínica Bucaramanga.

En el año de 1992 esta entidad entró a formar parte de la Clínica Carlos Ardila Lulle, ubicándose en el cuarto piso de sus instalaciones, lo cual permitió ampliar todos los servicios diagnósticos e intervencionistas de cardiología y cirugía vascular periférica, utilizando salas de cirugía, unidad de cuidados intensivos y hospitalización.

Posteriormente, en octubre de 1997 se inauguró la nueva sede del Instituto del Corazón, un edificio de 14 pisos con capacidad para 123 camas de hospitalización distribuidas entre la unidad de Cuidados Intensivos Post-quirúrgica, unidad de Cuidados Intensivos Pediátrica, unidad de Cuidados Intermedios Adultos, tres pisos de hospitalización, 4 salas de cirugía, 2 salas de Hemodinámica y una del servicio de urgencias durante las 24 horas del día, cumpliendo así con todos los requisitos y normas exigidas por el Ministerio de salud, relacionadas con enfermedades cardiovasculares.

En el año 2000, se propone la diversificación concentrada en la satisfacción de las necesidades del sector Salud, creando 5 nuevas empresas (unidades estratégicas de emprendimiento) FCV Software (DTICs), FCV Comercializadora, FCV Administración Hospitalaria, FCV Productos Hospitalarios y FCV Instituto de Investigaciones.

En el año 2006 Se inaugura el Centro Tecnológico Empresarial, con más de 5500 metros cuadrados construidos, espacio otorgado a las unidades estratégicas de negocio productivas con el fin de desarrollar tecnología y conocimiento, entre ellas la UEE Bioingeniería. También se inicia el desarrollo de los productos base de esta unidad estratégica empresarial, orientados a suplir de una mejor forma las necesidades de las entidades de salud a nivel nacional.

En el 2007 la Fundación Cardiovascular de Colombia recibe la visita de Recertificación ISO 9001 por parte del ente certificador ICONTEC, abre su nueva sala de neonatos, la nueva unidad de cuidado crítico y se consolida la UEN FCV Telemedicina como la más grande institución en este campo a nivel nacional. Su estructura actual se muestra en la figura 1.

Misión empresarial:
La Fundación Cardiovascular de Colombia (FCV) es una organización empresarial sin ánimo de lucro que provee servicios y productos de salud de alta calidad para el desarrollo del sector buscando permanentemente el bienestar de la comunidad.

Visión empresarial:
En el año 2023 la Fundación Cardiovascular de Colombia será una institución de referencia nacional e internacional en la prestación de servicios para enfermedades de alta complejidad centrado en las enfermedades del corazón, neurovascular, trasplantes y manejo de UCI.

Datos de la empresa:
El instituto del corazón se ubica actualmente en la Calle 155A #23-58, Floridablanca, Santander. Su número telefónico es 6399292.

El Centro Tecnológico Empresarial (CTE) se encuentra ubicado en la Carrera 5ª No.6-33 Floridablanca, Santander.

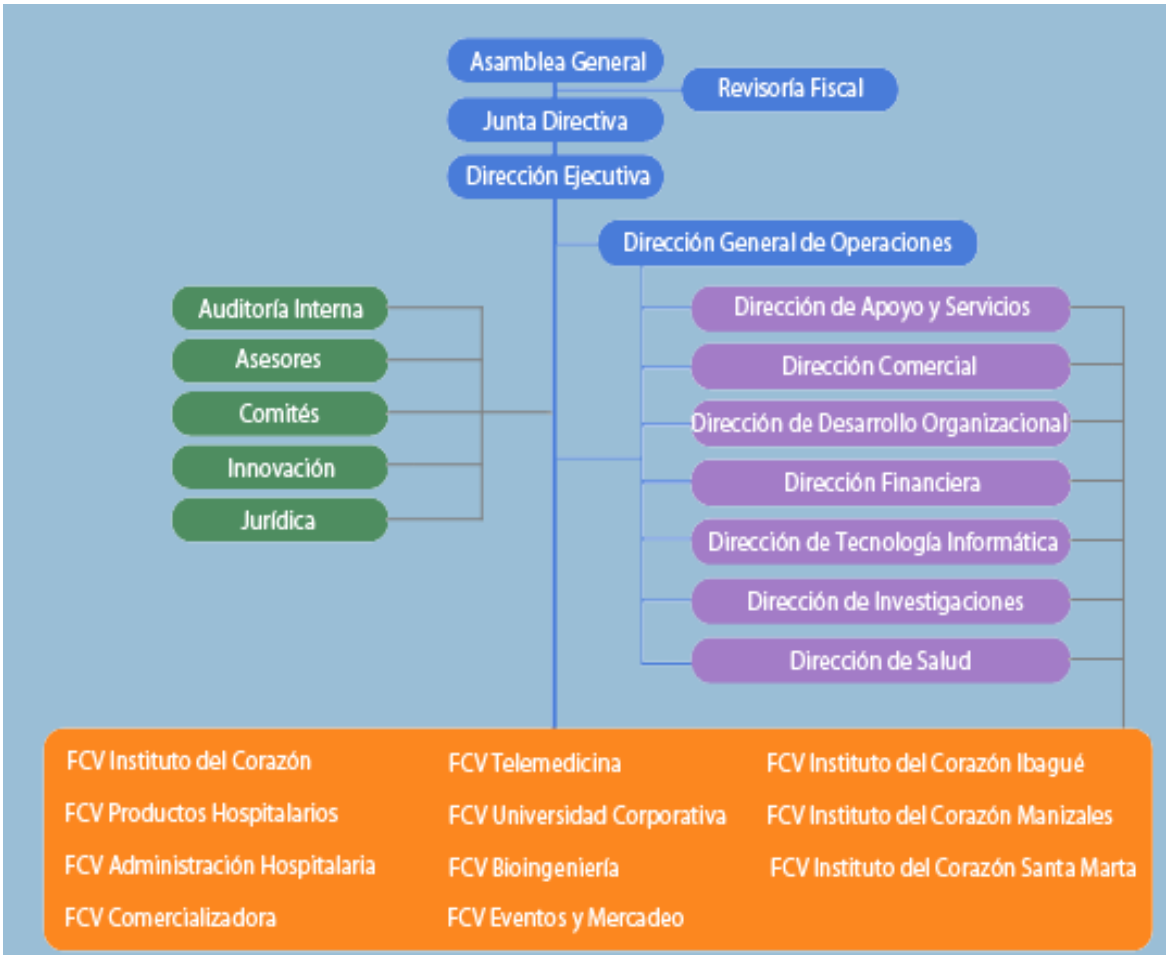


Figura 1. Organigrama de la Fundación Cardiovascular [1]

1.2. DESCRIPCIÓN DEL ÁREA DE PRÁCTICA

En Bioingeniería la innovación y el desarrollo tecnológico se traducen en la protección y preservación de la vida humana. El valor agregado de esta Unidad Estratégica de Negocios es la creación y ensamblaje de productos para solucionar las necesidades de equipamiento médico en los hospitales del país, contribuyendo al desarrollo científico y tecnológico en salud, mediante la producción de conocimiento, innovación, transferencia y apropiación de tecnologías, dirigidas al mejoramiento de las condiciones de vida de la población colombiana.

Dentro de sus áreas o procesos, se destaca Diseño y Desarrollo (D&D), área en la cual se da solución a las necesidades médicas por medio de un proceso controlado de diseño, el cual concluye en un prototipo del dispositivo diseñado. Es en este proceso en donde la investigación toma el papel más relevante, pues de manera creativa se debe dar respuesta a las necesidades medicas de forma novedosa, impactante y segura para el paciente. El área de Producción se encarga de fabricar en serie los dispositivos desarrollados por D&D, haciendo uso de proveedores nacionales e internacionales, contando con un proceso totalmente enmarcado en normas de salud ocupacional, buenas prácticas de manufactura y cuyo espacio físico fue rediseñado teniendo en cuenta las normas de habilitación como productor de dispositivos médicos. Soporte técnico es el proceso que presta servicios de instalación, capacitación, mantenimiento preventivo y/o correctivo de equipos biomédicos, y se destacan por el objetivo principal en sus servicios, la satisfacción del cliente. Por último está el laboratorio de Validación y Metrología, quienes prestan el servicio de control de calidad de producto interno terminado así como de la validación en Software, prototipos de equipos biomédicos, además de brindar calibración a los mismos. Todas las áreas de este gran proceso se lleva a cabo bajo los lineamientos de las áreas de administración y calidad.

Dentro de las actividades de Bioingeniería se encuentra diseñar, producir, proveer soporte técnico y brindar el servicio de metrología a los equipos biomédicos desarrollados o incluso a aquellos que no lo han sido. En estos procesos se incluyen actividades de control de calidad, estandarización, validación y acompañamiento permanente a sus clientes. Esta Unidad Estratégica Empresarial UEE, está orientada a la alta tecnología, y por ello tecnifica sus procesos, productos y servicios, buscando estandarizarlos internacionalmente. FCV Bioingeniería se destaca en el mercado, ya que se ha especializado en adaptar el producto a las necesidades del usuario, generando alternativas novedosas, usables y que resuelven las necesidades planteadas de forma sencilla, intuitiva y amigable con el usuario final.

Recientemente, Bioingeniería recibió la certificación ISO 13485, convirtiéndose en la primera institución Colombiana que realiza el proceso global de generación de dispositivos médicos, desde la generación de la idea hasta la línea de producción de biosistemas electrónicos de gran precisión y calidad, que suple a cabalidad los requerimientos del usuario. Este trabajo consistió en adoptar el sistema de calidad, la ISO 9001, lo que proporcionó la base para la norma internacional ISO 13485, así conllevó a implementar un mejoramiento continuo que logró centralizar y

unificar todos los elementos de la administración de calidad, en los estándares internacionales más altos. Adicionalmente el laboratorio de Validación y Metrología cuenta con la acreditación ISO/IEC 17025:2005, Norma internacional de requisitos generales para la competencia de laboratorios de ensayo y calibración, con lo cual los medios de comunicación publicaron a la institución como la primera en Colombia que se acredita para calibraciones en variables biomédicas.

Los resultados y competitividad del trabajo de Bioingeniería es notorio, ya que el gran impacto social, como reducción de muerte y morbilidad en pacientes de cuidado intensivo, surgió como consecuencia de implementar Telemedicina en Colombia, programa a través del cual, la FCV logró hacer presencia en diecinueve departamentos y treinta municipios de zonas rurales del país. La oferta de los equipos biomédicos producidos por Bioingeniería, ha permitido que los hospitales accedan a la más alta tecnología, desarrollada con los mejores estándares de calidad, a un bajo costo.

Actualmente se están desarrollando proyectos relativos a la monitorización de signos vitales a distancia o tele-monitorización, tecnologías ubicuas para la medición de constantes vitales de pacientes crónicos y se ha incursionado en el diseño y creación de dispositivos médicos terapéuticos basados en biomateriales usando nanotecnología lo que ha permitido utilizar las partículas más pequeñas de algunas sustancias para mejorar notablemente la vida humana.

Misión:

FCV Bioingeniería es una unidad empresarial de negocios de la Fundación Cardiovascular de Colombia que contribuye con el desarrollo científico y tecnológico en salud, mediante la producción de conocimiento, innovación, transferencia y apropiación de tecnologías dirigidas al mejoramiento de las condiciones de vida de la población colombiana con posicionamiento del desarrollo tecnológico y la producción nacional en el contexto internacional.

Visión:

En el 2020 la unidad empresarial FCV Bioingeniería será reconocida en el país como una entidad desarrolladora y productora de equipos médicos confiables y competitivos, con un alto componente de innovación tecnológica

1.3 DIAGNOSTICO DE LA EMPRESA

La FCV brinda productos y servicios de alta calidad enfocados en el sector salud, buscando permanentemente el bienestar de la comunidad, es por ello que su UEE Bioingeniería ha señalado la necesidad de implementar equipos médicos sobre sistemas embebidos usando dispositivos lógicos programables tales como FPGAs.

La FCV actualmente no cuenta con los drivers necesarios para controlar los módulos Wi-Fi y de transmisión de datos por Red Celular, los cuales ya se encuentran dispuestos físicamente en el dispositivo medico en desarrollo. Dada esta necesidad se busca diseñar e implementar los diferentes drivers que permitan la interacción del sistema entre el softcore “NIOSII” y los periféricos que dan conectividad a la red al sistema embebido.

Para dar solución a esta necesidad fueron adquiridos módulos de desarrollo (módulos de evaluación) Wi-Fi y de Red celular para el desarrollo de prototipos de los drivers previos a su implementación real en el dispositivo medico.

Adicionalmente cada uno de los módulos de comunicación debe dar soporte en la capa de aplicación al protocolo Websocket el cual fue utilizado para la transmisión de datos entre el monitor de signos vitales y el servidor.

La FCV en su UEE Bioingeniería, firma el convenio con el practicante en calidad de práctica empresarial permitiendo concluir la implementación de los drivers para los módulos de comunicaciones Wi-Fi y de Red Celular para transmisión de datos en un dispositivo médico permitiendo al practicante afianzar y aplicar sus conocimientos.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Diseñar y desarrollar drivers que permitan la conectividad del dispositivo médico desarrollado en la UEE Bioingeniería de la FCV para el envío de señales fisiológicas a través de protocolos Ethernet, Wi-Fi y Red celular para transmisión de datos.

2.2 OBJETIVOS ESPECÍFICOS

- Diseñar y desarrollar el *driver* para la transmisión de datos vía Wi-Fi del dispositivo médico en desarrollo.

- Diseñar y desarrollar el *driver* para la transmisión de datos mediante Red celular y Georeferenciación del dispositivo médico en desarrollo.

- Integrar los diferentes dispositivos de comunicación por medio del softcore NIOS II en el dispositivo médico de tal forma que permita gestionar de forma transparente y eficiente la transmisión de datos.

2.3 ACTIVIDADES A DESARROLLAR

Estudiar las capas de comunicación de los diferentes sistemas de gestión de la información

Es muy importante contextualizar el desarrollo a realizar con la teoría de las diferentes capas de comunicación y acceso a la Internet. Para esto es muy útil estudiar las diferentes funciones que contiene el *driver* Ethernet actualmente implementado debido a que en base a este se realizaran funciones equivalentes en los nuevos módulos de transmisión inalámbrica que se van a implementar.

Pruebas en el módulo Wi-Fi para analizar y verificar su funcionamiento e implementación en FPGA mediante modulo UART/Full Dúplex en Hardware

Es de vital importancia conocer todas las limitaciones de hardware y software que tiene el módulo Wi-Fi así como hacer pruebas reales de manera práctica tales como configuración de la dirección IP del servidor, adquirir dirección IP local mediante protocolo DHCP, envío de datos mediante protocolos UDP y TCP, configuración de la seguridad de la red Wi-Fi, SSID de la red, etc. Para realizar estas pruebas se aprovechara la característica del modulo Wi-Fly, que para su configuración cuenta con el envió y recepción de datos de forma serial mediante protocolo RS-232. Mediante este protocolo se puede configurar la red Wi-Fi y también enviar y recibir datos a través de los protocolos soportados por su stack TCP/IP. Se realizarán pruebas para verificar su configuración y funcionamiento en el envió y recepción de datos mediante el modulo UART/Full Dúplex implementado directamente en FPGA.

Pruebas en el módulo entrenador de Red celular para transmisión de datos y Georeferenciación y sus diferentes funcionalidades e implementación en FPGA mediante modulo UART/Full Dúplex en Hardware

Al igual que en el módulo Wi-Fi es de gran importancia conocer y probar los diferentes métodos de configuración del módulo entrenador de red celular. Se realizarán diferentes pruebas de configuración para envío y recepción de datos desde el monitor hacia la Red y viceversa. Seguido a las pruebas se implementa el driver en la FPGA mediante un módulo de comunicación en hardware UART/Full Dúplex que se encuentra directamente en la FPGA.

Implementación de los diferentes periféricos por medio del softcore NIOS II con sus respectivas funciones y verificación de compatibilidad con los otros módulos de comunicación

De manera general se debe diseñar un módulo en software que controle toda la información que requiera enviar ó recibir el dispositivo médico de forma independiente al tipo de hardware a utilizar. En el diseño de este módulo se deben tener en cuenta los diferentes tipos de funciones de cada driver para priorizar el envío y recepción de datos en caso de existir más de un medio de conexión a la red. Lo anterior teniendo en cuenta que el dispositivo médico dispone con tecnología Ethernet, Wi-Fi y Red Celular para transmisión de datos.

2.3.1. Cronograma de actividades

Tareas	Semana															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Estudiar la librería del controlador Ethernet implementado en el dispositivo médico																
Pruebas en el módulo Wi-Fly e implementación del driver directamente en el procesador NIOS II sobre FPGA DE0-Nano																
Pruebas en el módulo de red celular e implementación del driver directamente en el procesador NIOS II sobre la FPGA DE0-Nano.																
Diseño de los diferentes periféricos por medio del softcore NIOS II con sus respectivas funciones y verificación de compatibilidad con otros módulos de comunicación																

Tabla 1. Cronograma de actividades

3. MARCO TEÓRICO

3.1 NORMAS PARA EL DISEÑO DE DISPOSITIVOS MÉDICOS

La UEN Bioingeniería se encuentra certificada en las normas ISO 9001 y ISO 13485, sin embargo para posibilitar la comercialización de los equipos en otros mercados, estos deben basarse en la norma AAMI (Association for the Advancement of Medical Instrumentation), específicamente en AAMI 60601-1-2:2007(Compatibilidad electromagnética) y AAMI HE75:2009 (Ingeniería de factores humanos).

La asociación para el avance de la instrumentación médica (AAMI), tiene como objetivo el desarrollo seguro y efectivo de tecnología médica. Es la organización líder en su campo con más de 7000 individuos, hospitales y fabricantes de dispositivos médicos miembros. Se encarga de desarrollar, revisar y mejorar estándares a ser alcanzados por los dispositivos médicos [1].

3.2 INTERNET

Internet es un conjunto mundial de redes de computadoras que colaboran entre ellas para intercambiar información mediante estándares en común. A través de cables telefónicos, cables de fibra óptica, transmisiones inalámbricas y enlaces satelitales, los usuarios de Internet pueden intercambiar información de diversas formas. Permite tanto a personas como a empresas compartir información. Actualmente existen más de 1000 millones de usuarios de Internet[10].

3.3 ESTANDARES EN INTERNET

Un estándar es un conjunto de reglas que determina cómo se realiza algo. Los estándares de red y de Internet aseguran que todos los dispositivos conectados a la red utilicen el mismo conjunto de reglas. Al contar con estos estándares, pueden enviarse información entre sí diferentes tipos de dispositivos a través de Internet. Por ejemplo, el modo en que los dispositivos formatean, envían y reciben un correo electrónico se realiza de una manera estandarizada.

Un estándar de Internet es el resultado final de un ciclo completo de discusión, resolución de problemas y pruebas. Cuando se propone un nuevo estándar, cada etapa del desarrollo y del proceso de aprobación es registrada en un documento numerado de solicitud de comentarios (RFC, Request for Comments) para seguir la evolución del estándar. Gracias a estos estándares millones de personas pueden conectarse a Internet mediante distintos dispositivos, como las computadoras personales, los teléfonos celulares, los asistentes digitales personales (PDA) y portátiles, los reproductores de MP3 e incluso los televisores[11].

3.4 PROTOCOLOS DE COMUNICACIÓN

Todas las comunicaciones, tanto humanas como informáticas, están regidas por reglas preestablecidas o protocolos. Estos protocolos están determinados por las características del origen, el canal y el destino. En función del origen, el canal y el

destino, los protocolos definen los detalles relacionados con el formato del mensaje, el tamaño del mensaje, la sincronización, la encapsulación, la codificación y el patrón estándar del mensaje. Un protocolo debe tener una descripción formal en la cual se describan: identificadores, datos, autenticación y detección de errores (como se muestra en la Figura 2) [12].

El protocolo también define la sintaxis, sincronización de la comunicación y parámetros de la capa física a utilizarse (muchos de ellos tienen parámetros que pueden diferir de equipo a equipo, como la velocidad de transmisión o el número de bits). Un protocolo puede implementarse como hardware, como software ó como ambos, el objetivo es que se transmita y reciba siguiendo las mismas especificaciones.

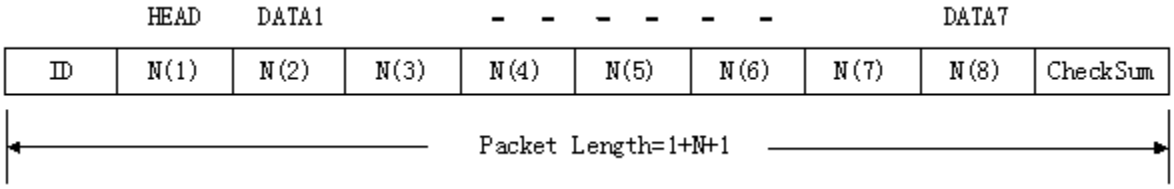


Figura 2. Ejemplo de un protocolo de comunicación serial [2]

3.5 ENVÍO DE INFORMACIÓN A TRAVES DE INTERNET

Para que los dos puntos remotos puedan comunicarse por Internet deben ejecutar software mediante el protocolo de Internet (IP). El protocolo IP es uno de los elementos del grupo de protocolos colectivamente denominados TCP/IP (protocolo de control de transmisiones/protocolo de Internet). El protocolo de Internet (IP) utiliza paquetes para transportar los datos. La información que envía o recibe cualquier medio físico de conexión a la red es transportada en forma de paquetes IP[11].

Cada paquete IP debe contener una dirección IP de origen y una de destino válidas. Si no hay información de dirección valida, los paquetes enviados no llegarán al host de destino. Los paquetes de respuesta tampoco llegarán a la fuente original. Un paquete IP posee un encabezado en el comienzo que contiene las direcciones IP de origen y de destino. También contiene información de control que describe el paquete a los dispositivos de red por los que pasa, como los routers, y ayuda a controlar su comportamiento en la red. El paquete IP también se denomina datagrama[11].

3.6 DIRECCIONES IP

Cada paquete enviado por Internet tendrá una dirección IP de origen y de destino. Los dispositivos de red requieren esta información para asegurarse de que la información llegue al destino y de que toda respuesta sea devuelta al origen.

Una dirección IP es simplemente una serie de 32 bits (unos y ceros). Para una persona seria muy difícil leer una dirección IP binaria. Por este motivo, los 32 bits están agrupados en cuatro bytes de 8 bits llamados octetos. Una dirección IP en

este formato no es fácil de leer, escribir o recordar. Para hacer que las direcciones IP sean más fáciles de entender, cada octeto se presenta como su valor decimal, separado por un punto decimal. Esto se conoce como notación decimal punteada[11].

Las direcciones IP pueden asignarse de manera estática o dinámica.

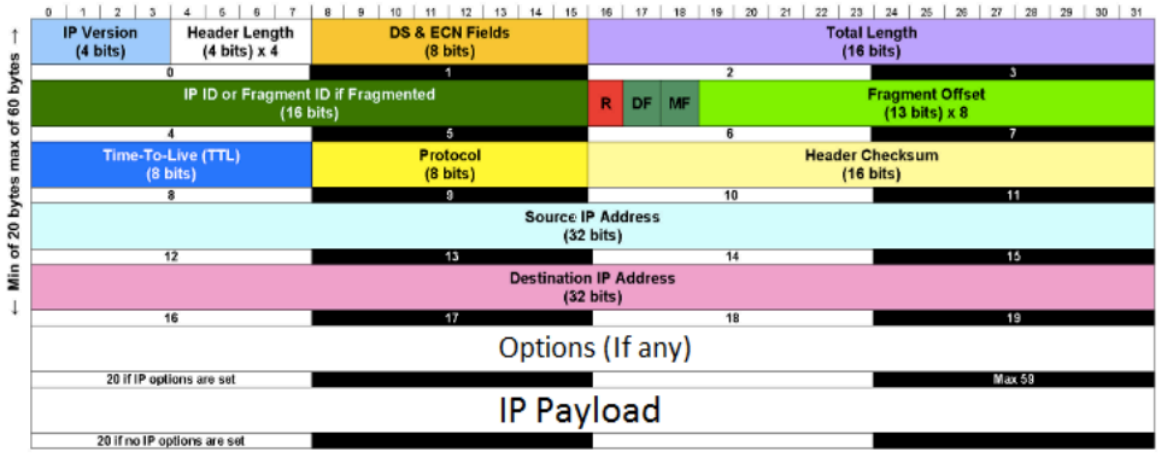


Figura3. Descripción de un paquete IP versión IPV4 [3]

3.6.1 ASIGNACIÓN IP ESTÁTICA

Con una asignación estática, el administrador de red debe configurar manualmente la información de red para un dispositivo. Estas direcciones son útiles para impresoras, servidores y otros dispositivos de red que deben estar accesibles para los clientes de forma fija. Cuando se utiliza el direccionamiento IP estático, es importante mantener una lista precisa de cuales direcciones IP se asignan a qué dispositivos para no crear conflicto en la asignación de direcciones. Además, estas direcciones son permanentes y generalmente no se reutilizan[11].

3.6.2 ASIGNACIÓN IP DINÁMICA

En las redes locales, es habitual que la población de usuarios cambie frecuentemente. Se agregan nuevos usuarios con computadoras portátiles, y esos usuarios requieren una conexión. En lugar de que el administrador de red asigne direcciones IP para cada estación de trabajo, es más simple que las direcciones IP se asignen automáticamente. Esto se logra a través de un protocolo denominado protocolo de configuración dinámica de host (DHCP) [11].

3.7 PROTOCOLO DHCP

El servicio del DHCP permite a los dispositivos de una red obtener direcciones IP y otra información de un servidor DHCP. Este servicio automatiza la asignación de direcciones IP, mascaras de subred, Gateway y otros parámetros de networking del IP[12].

Muchas redes domesticas y de empresas pequeñas utilizan un router integrado para conectarse al módem del ISP. En este caso, el router integrado funciona como cliente de DHCP y como servidor. El router integrado actúa como cliente para recibir su configuración IP del ISP y luego actúa como servidor de DHCP para los hosts internos en la red local[12].

3.8 MODELO CLIENTE SERVIDOR

En el modelo cliente-servidor, el dispositivo que solicita información se denomina cliente y el dispositivo que responde a la solicitud se denomina servidor. Los procesos de cliente y servidor se consideran una parte de la capa de aplicación. El cliente comienza el intercambio solicitando los datos al servidor, el cual responde enviando uno o más streams de datos al cliente. Los protocolos de la capa de aplicación describen el formato de las solicitudes y respuestas entre clientes y servidores. Además de la transferencia real de datos, este intercambio puede requerir información adicional, como la autenticación del usuario y la identificación de un archivo de datos a transferir[11].

La característica clave de los sistemas cliente-servidor es que el cliente envía una solicitud a un servidor, y este responde ejecutando una función, como enviar información al cliente. La combinación de un explorador Web y un servidor Web es quizás el ejemplo que más se utiliza en un sistema cliente-servidor[11].

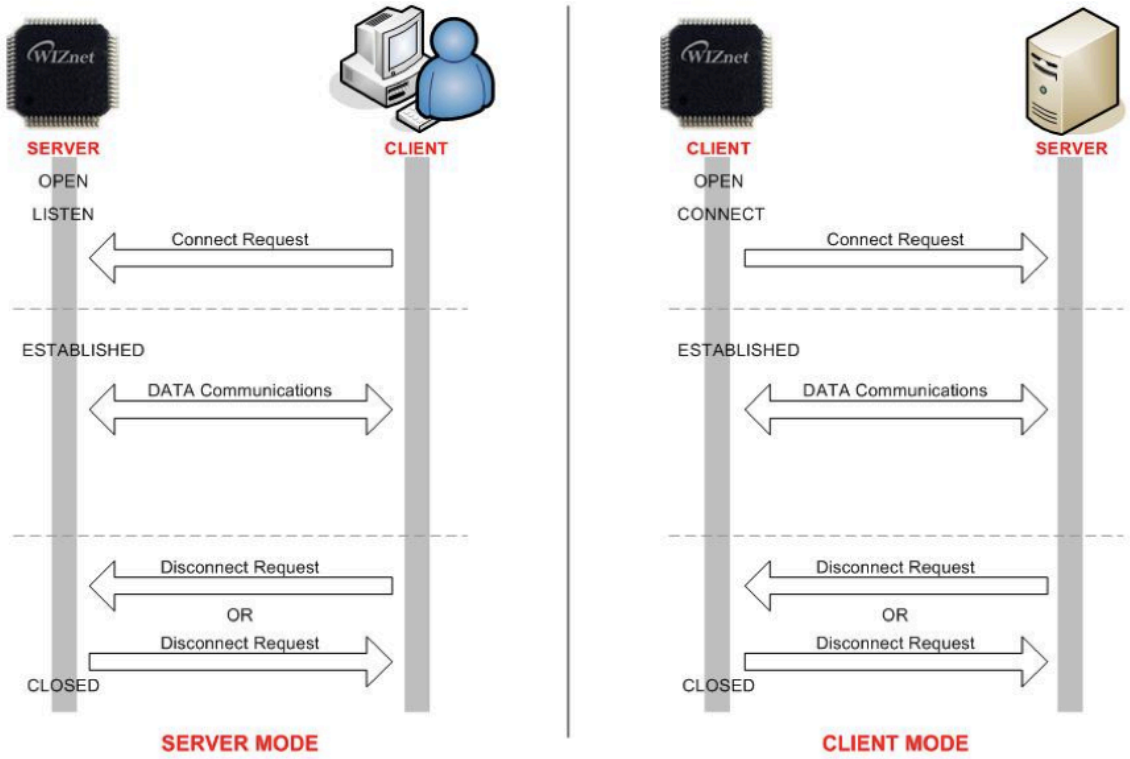


Figura 4. Diagrama de simplificado de conexión entre cliente y servidor[4]

3.9 CLIENTE DE RED

Nodo o programa de software que solicita servicios a un servidor[11].

3.10 MODELO OSI / TCP IP

Para visualizar la interacción entre varios protocolos, es común utilizar un modelo en capas. Este modelo describe el funcionamiento de los protocolos que se produce en cada capa y la interacción con las capas que se encuentran por encima y por debajo de ellas.

3.10.1 MODELO TCP IP

El primer modelo de protocolo en capas para comunicaciones de internet se creó a principios de la década de los setenta y se conoce con el nombre de modelo de Internet. Define cuatro categorías de funciones que deben existir para que las comunicaciones sean exitosas. La arquitectura de la suite de protocolos TCP/IP sigue la estructura de este modelo. Por esto es común que al modelo de Internet se le conozca como modelo TCP/IP[11].

La mayoría de los modelos de protocolos describen un stack de protocolos específicos del proveedor. Sin embargo, puesto que el modelo TCP/IP es un estándar abierto, una compañía no controla la definición del modelo. Las

definiciones del estándar y los protocolos TCP/IP se explican en un foro público y se definen en un conjunto de documentos disponibles al público.

3.10.2 MODELO OSI

Inicialmente, el modelo OSI fue diseñado por la Organización Internacional para la Estandarización (ISO, International Organization for Standardization) para proporcionar un esquema sobre el cual crear una suite de protocolos de sistemas abiertos. La idea era que este conjunto de protocolos se utilizara para desarrollar una red internacional que no dependiera de sistemas propietarios[11].

Lamentablemente, la velocidad a la que fue adoptada la Internet con base en TCP/IP y la velocidad a la que se expandió ocasionaron que el desarrollo y la aceptación de la suite de protocolos OSI quedaran atrás. Aunque pocos de los protocolos que se crearon mediante las especificaciones OSI se utilizan ampliamente en la actualidad, el modelo OSI de siete capas ha hecho más contribuciones al desarrollo de otros protocolos y productos para todo tipo de redes nuevas. [11]

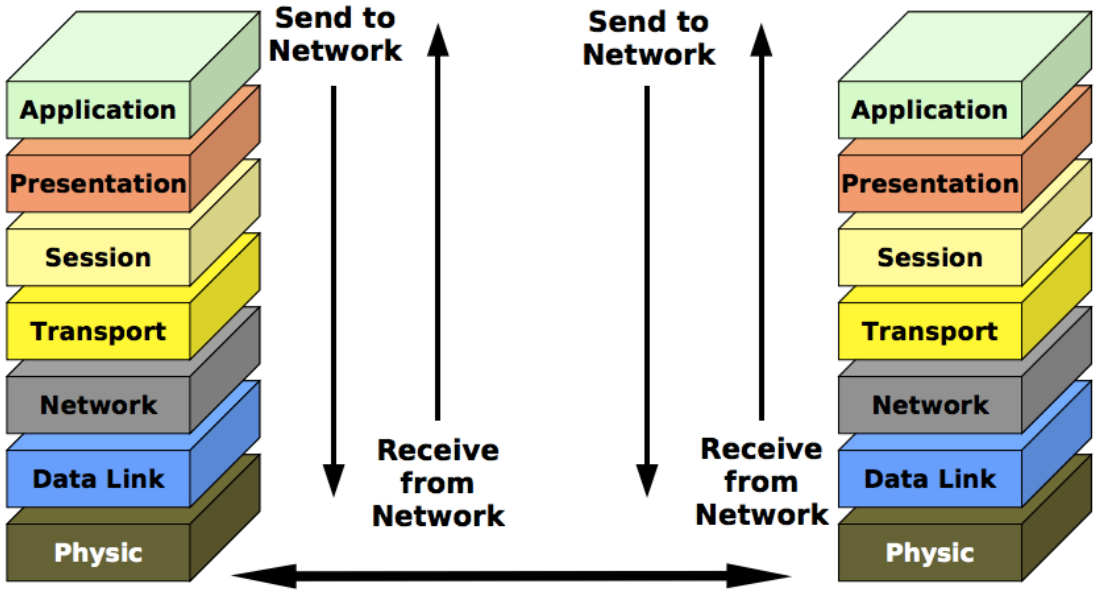


Figura5. Flujo de información en las diferentes capas del modelo OSI [5]

3.11 PROTOCOLOS Y SERVICIOS DNS

En las redes de datos, los dispositivos se etiquetan con una dirección IP numérica, de manera que pueden participar en el envió y la recepción de mensajes de la red. Sin embargo, la mayoría de las personas pasa mucho tiempo tratando de recordar estas direcciones numéricas. Por lo tanto, los nombres de dominio se crearon para convertir las direcciones numéricas en un nombre sencillo y reconocible.

En Internet estos nombres de dominio, tales como www.cisco.com, son mucho más fáciles de recordar para la gente que algo como 198.133.219.25, que es la dirección numérica actual para ese servidor. Además, si Cisco decide cambiar la dirección numérica, la acción será transparente para el usuario, ya que el nombre de dominio seguirá siendo www.cisco.com. La nueva dirección simplemente estará enlazada con el nombre de dominio existente, y la conectividad se mantendrá. DNS utiliza un conjunto distribuido de servidores para resolver los nombres asociados a estas direcciones numéricas[12].

3.12 PROTOCOLOS TCP Y UDP

Los dos protocolos TCP/IP más comunes de la capa de transporte son el Protocolo de control de transmisión (TCP) y el Protocolo de datagramas de usuario (UDP). Ambos protocolos gestionan la comunicación de múltiples aplicaciones.

3.12.1 Protocolo de datagramas de usuario (UDP)

Protocolo de la capa de transporte sin conexión en el stack de protocolos. UDP es un protocolo simple que intercambia datagramas sin acuses de recibo ni garantía de envió y que requiere que el procesamiento de errores y la retransmisión sean administrados por otros protocolos. El UDP se define en la RFC 768. Las aplicaciones que utilizan UDP incluyen aplicaciones donde la perdida de datos no sea de vital importancia como streams de video, voz sobre IP (VOIP), etc.[12].

3.12.2 Protocolo de control de transmisión (TCP)

TCP es un protocolo orientado a la conexión, descrito en la RFC 793. Es el protocolo utilizado como base para la mayoría de los servicios de Internet. Se utiliza en forma conjunta con el Protocolo de Internet (IP). Permite una comunicación confiable y garantiza que los paquetes lleguen a los destinos previstos. Cada segmento de TCP posee 20 bytes de carga en el encabezado, que encapsulan los datos de la capa de aplicación, mientras que cada segmento UDP solo posee 8 bytes de carga. Algunas de las aplicaciones que utilizan el protocolo TCP son exploradores Web, correo electrónico, transferencias de archivos, etc.[12].

3.13 ACK (Acuse de recibo)

Carácter de control de transmisión (o trama de transmisión) que confirma que un mensaje transmitido fue recibido sin daños ni errores ó que la estación receptora está lista para aceptar transmisiones[12].

3.14 ETHERNET

Es un estándar cableado de redes que emplea el método CSMA/CD (Acceso Múltiple por Detección de Portadora con Detector de Colisiones) que mejora notoriamente el rendimiento de dicha conectividad para acceso a la red.

Se trata de un estándar que define no solo las características de los cables que deben utilizarse para establecer una conexión de Red, sino también todo lo relativo a los niveles Físicos de dicha conectividad, además de brindar los formatos necesarios para las tramas de datos de cada nivel. Está regulado en la norma IEEE 802.3[12].

3.15 DIRECCIÓN MAC

Dirección de capa de enlace de datos estandarizada que se requiere para cada puerto o dispositivo que se conecta a una LAN. Otros dispositivos de la red usan estas direcciones para localizar puertos específicos en la red, y para crear y actualizar tablas de enrutamiento y estructuras de datos. Las direcciones MAC tienen seis bytes de largo y se controlan a través de la IEEE. Estas direcciones son únicas para cada dispositivo[12].

3.16 NÚMEROS DE PUERTO TCP/IP

Cuando se envía un mensaje utilizando TCP ó UDP, los protocolos y servicios solicitados se identifican con un número de puerto. Un puerto es un identificador numérico de cada segmento, que se utiliza para realizar un seguimiento de conversaciones especificas y de servicios de destino solicitados. Cada mensaje que envía un host contiene un puerto de origen y un puerto de destino[12].

3.17 SOCKET

Un socket puede considerarse como un punto final en una comunicación. Un socket cliente en una computadora utiliza una dirección para llamar a un socket servidor en otro computador. Una vez se han enlazado los sockets apropiados, los computadores pueden intercambiar datos.

3.18 MEDIOS DE TRANSMISIÓN INALÁMBRICOS

Tecnología que permite la comunicación sin necesidad de contar con conectividad física. Algunos ejemplos de tecnología inalámbrica son los teléfonos celulares, los asistentes digitales personales (PDA), los puntos de acceso inalámbrico y las NIC inalámbricas.

Además de la red conectada por cable, existen varias tecnologías que permiten la transmisión de información entre hosts sin cables. Esas tecnologías se conocen como tecnologías inalámbricas. Las tecnologías inalámbricas utilizan ondas electromagnéticas para transportar información entre dispositivos. Una onda electromagnética es el mismo medio que transporta señales de radio por aire[12].

3.19 WI-FI /WLAN

La WLAN se usa generalmente para ampliar los límites de la red de área local (LAN, local wired network). Las WLAN usan la tecnología RF y cumplen con los estándares IEEE 802.11. Permiten a muchos usuarios conectarse a una red

conectada por cable mediante un dispositivo conocido como punto de acceso (AP). El punto de acceso proporciona una conexión entre los hosts inalámbricos y los hosts en una red Ethernet conectada por cable.

El estándar IEEE 802.11 rige el entorno WLAN. Existen cuatro enmiendas al estándar IEEE 802.11 que describen diferentes características para las comunicaciones inalámbricas. Las enmiendas actualmente disponibles son 802.11a, 802.11b, 802.11g y 802.11n. Estas tecnologías se conocen grupalmente con el nombre Wi-Fi, amplia fidelidad. Todas operan en la banda de los 2.4GHz excepto la 802.

Otra organización, conocida como Wi-Fi Alliance, es responsable de probar los dispositivos LAN inalámbricos de distintos fabricantes. El logotipo Wi-Fi en un dispositivo significa que ese equipo cumple los estándares y debe interoperar con otros dispositivos del mismo estándar[11].

3.20 SSID

Cuando se genera una red inalámbrica es importante que los componentes inalámbricos se conecten a la WLAN apropiada. Esto se realiza mediante un identificador del servicio (SSID, Service Set Identifier). El SSID es una cadena alfanumérica que distingue entre mayúsculas y minúsculas y consta de hasta 32 caracteres. Se envía en el encabezado de todas las tramas transmitidas por la WLAN. El SSID se utiliza para comunicar a los dispositivos inalámbricos a qué WLAN pertenecen y con qué otros dispositivos pueden comunicarse. Normalmente el SSID es el nombre con el que se reconoce un AP por un cliente inalámbrico Wi-Fi que se quiere enlazar[11].

3.21 CLIENTE INALÁMBRICO

Cualquier dispositivo host que puede conectarse a una red inalámbrica[11].

3.22 ACCESS POINT (AP)

Cualquier dispositivo host que puede conectarse a una red inalámbrica[11].

3.23 ENCRIPTACIÓN DE UNA WLAN

La autenticación y el filtrado MAC pueden evitar que un atacante se conecte a una red inalámbrica, pero no evitaran que intercepte los datos transmitidos. Dado que no existen límites distintivos en una red inalámbrica y que el tráfico se transmite por aire, es fácil para un atacante interceptar o detectar tramas inalámbricas. La encriptación es el proceso de transformar datos de manera que, aunque sean interceptados, queden inutilizables[11].

3.23.1 WEP (Wired equivalency protocol)

Privacidad equivalente por cable (Wired Equivalent Privacy). Parte del estándar de red inalámbrica IEEE 802.11 que proporciona un bajo nivel de seguridad. El WEP utiliza claves preconfiguradas para encrestar y descifrar datos.

Una clave WEP se introduce como una cadena de números y letras, y generalmente consta de 64 o 128 bits. En algunos casos, el WEP admite también claves de 256 bits[12].

3.23.2 WPA (Acceso protegido a Wi-Fi)

Acceso protegido Wi-Fi (Wi-Fi Protected Access). Desarrollado para tratar problemas de seguridad en WEP. Proporciona un mayor nivel de seguridad en una red inalámbrica. El WPA también utiliza claves de encriptación de 64 a 256 bits. Sin embargo, el WPA, a diferencia del WEP, genera nuevas claves dinámicas cada vez que un cliente establece una conexión con el AP. Por esta razón el WPA se considera más seguro que el WEP, ya que es mucho más difícil de decodificar[12].

3.24 RED CELULAR 3G PARA TRANSMISIÓN DE DATOS

La tecnologías 3G (tercera generación) son la respuesta a la especificación IMT-2000 de la Unión Internacional de Telecomunicaciones. El estándar UMTS está basado en la tecnología W-CDMA. UMTS está gestionado por la organización 3GPP, también responsable de GSM, GPRS y EDGE. 3G (o 3-G) es una abreviatura para tercera generación de telefonía móvil. Los servicios asociados con la tercera generación proporcionan la posibilidad para transferir tanto voz (una llamada telefónica) y datos (como la descarga de programas, intercambio de correo-e, y mensajería instantánea) [13].

Gracias a este sistema mejorado de transmisión de datos utilizando la red celular, las empresas de telefonía móvil no solo ofrecen banda ancha en los teléfonos, sino que también comercializan módems 3G para conectar a cualquier computadora y tener acceso a internet en cualquier lugar[13].

3.25 APN (ACCESS POINT NAME)

El Nombre del Punto de Acceso (Access Point Name o APN) es la configuración que le suministra al teléfono para conocer a qué red de datos puede conectar. Cada operador tiene por lo menos un APN por el cual brinda acceso a la red a sus abonados. Es la puerta de entrada a la red celular de transmisión de datos y debe ser configurado en el dispositivo móvil[13].

3.26 GPS (GLOBAL POSITIONING SYSTEM)

El sistema de posicionamiento global (GPS: Global Positioning System) es un sistema global de navegación por satélite cuya tarea es proporcionar la infraestructura necesaria para permitir una localización precisa en el entorno planetario. Este sistema permite determinar en todo el mundo la posición de un objeto con una precisión en la escala de metros[14].

3.27 WEBSOCKET

Es un protocolo de comunicación que define un socket sencillo de conexión full dúplex sobre el cual se pueden enviar mensajes entre cliente y servidor. El estándar Websocket simplifica en gran medida la complejidad en la comunicación Web bidireccional y la administración de conexiones [15].

El modelo de seguridad usado para este protocolo está basado en el modelo de seguridad comúnmente usado por los navegadores Web.

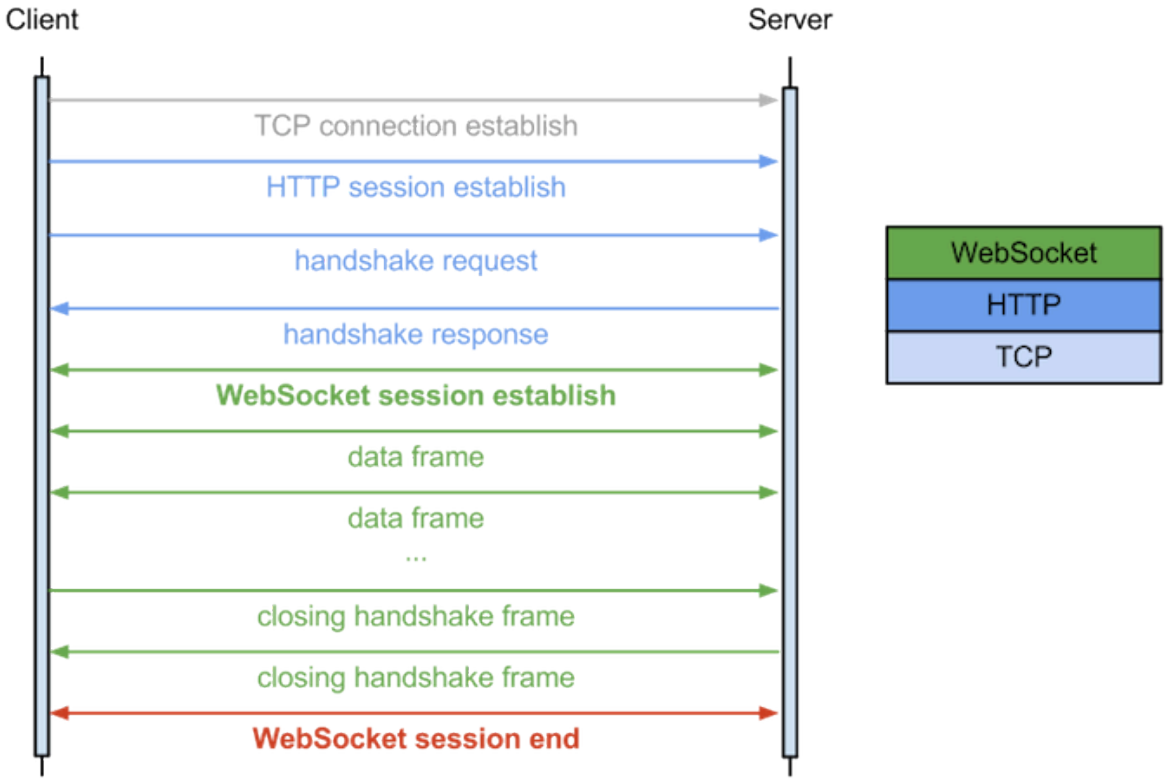


Figura 6. Ejemplo de comunicación mediante protocolo Websocket [6]

3.28 jWEBSOCKET

jWEBSOCKET es una solución de comunicación para la Web bidireccional y de alta velocidad desarrollada en Java/Javaescript. jWEBSOCKET es una implementación del protocolo Websocket HTML5 con un enorme conjunto de extensiones. Con jWEBSOCKET es posible implementar fácilmente para los programadores la comunicación entre cliente y servidor bajo el protocolo Websocket. jWEBSOCKET trae ventajas como bidireccionalidad para conexiones TCP de socket por un único socket para mas conexiones concurrentes entre cliente y servidor, mucho mas rápido comparado con otros protocolos de comunicación web, más sensible posibilitando baja latencia en aplicaciones web y además esta estandarizado mediante entidades como W3C e IETF [16].

3.29 HANDSHAKE

El protocolo tiene dos partes: Handshake e Intercambio de datos. El Handshake es un procedimiento que consiste en el intercambio de mensajes mediante el protocolo http entre cliente y servidor tal como se muestra en la Figura 7

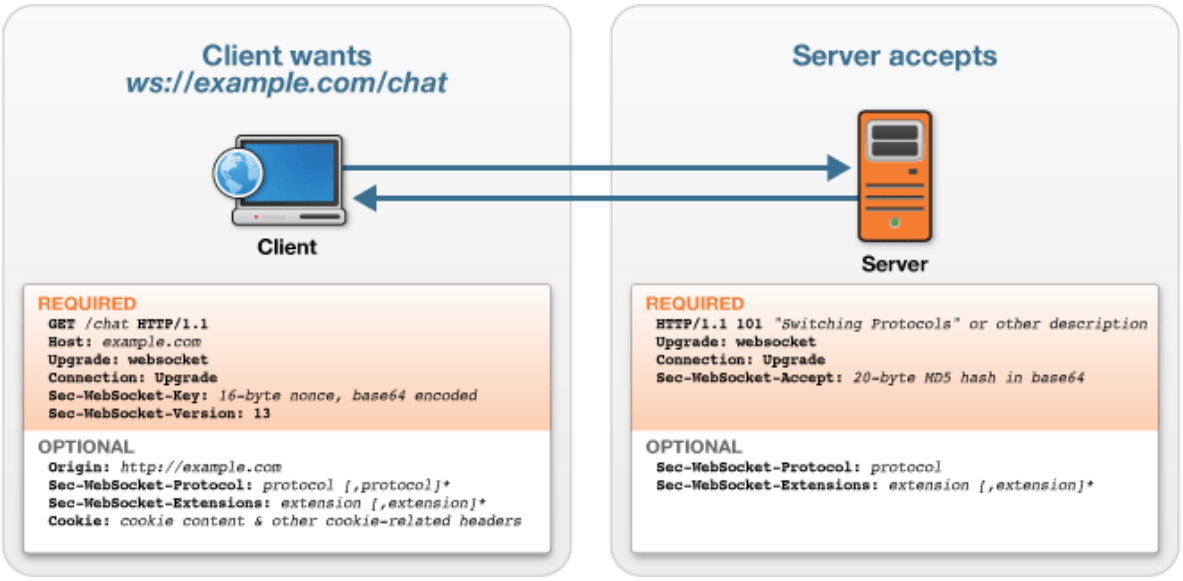


Figura 7. Intercambio básico de mensajes para Handshake [7]

El cliente envía un mensaje solicitando la conexión mediante protocolo Websocket con una clave única. El servidor responde a esta petición con otro mensaje en protocolo http enviando también una clave única. Una vez el cliente y el servidor intercambien satisfactoriamente sus respectivos mensajes de Handshake se puede iniciar con el intercambio de datos. Este intercambio de información es Full Dúplex y tanto cliente como servidor pueden enviar datos independientemente [17].

3.30 LOGIN

Es un término usado para referirse al ingreso a cuentas de usuario. Es el proceso de identificarse o registrarse para iniciar una sesión y obtener acceso a sistemas ó servicios en una red de datos.

3.31 AUTENTICACIÓN

Proceso implementado en una red de datos para verificar la identidad de un usuario.

3.32 JSON (Javascript Object Notation)

Es un formato ligero para intercambio de datos. En su notación se puede describir a sí mismo y se caracteriza por su fácil comprensión tanto para los humanos como para los procesadores. Json está basado en un subconjunto de Javascript (Standard ECMA-262 3rd Edition - Diciembre 1999.) Fue propuesto originalmente por Douglas Crockford y esta descrito en la norma RFC 4627[8].

Usualmente el formato JSON es usado para serializar y transmitir estructuras de datos a través de una conexión de red. Primariamente es usado para transmitir datos entre un servidor y una aplicación Web[8].

3.33 OBJETO JSON

Es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma) como se visualiza en la Figura 8 [8].

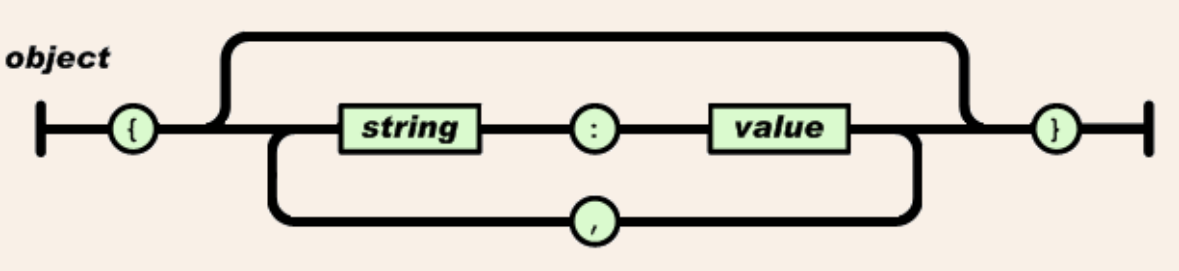


Figura 8. Objeto JSON [8]

3.34 FPGA (FIELD PROGRAMMABLE GATE ARRAY)

Circuito integrado programable, con capacidad de implementar cualquier función lógica que un ASIC pueda realizar. Son configurados en base a un lenguaje de descripción de hardware (HDL), de forma similar a los circuitos ASIC. Sus capacidades de actualizar su funcionalidad después de fabricada, su bajo coste en comparación a la fabricación en baja escala de un ASIC, y su alto paralelismo ofrece ventajas para varias aplicaciones. Las FPGA contienen elementos lógicos programables, y un conjunto de conexiones reconfigurables que permite a los bloques conectarse entre sí, permitiéndoles interactuar de múltiples formas.

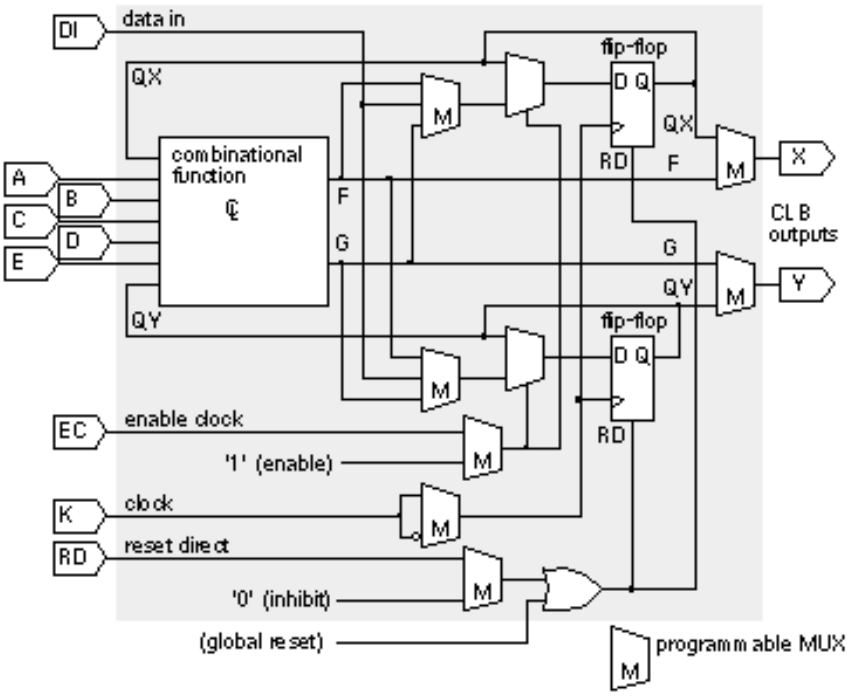


Figura 9. Estructura básica de un elemento lógico [9]

3.35 HDL (Hardware description language)

Es un lenguaje de computadoras avanzado usado para describir el diseño, la estructura y la operación de circuitos digitales lógicos. Esta descripción formal de un circuito lógico digital permite realizar análisis y simulación de las diferentes descripciones de circuitos lógicos a realizar. Entre los lenguajes más utilizados en el ámbito industrial y académico están VHDL y Verilog.

3.36 SOFTCORE

Un softcore processor es un modelo de descripción de hardware mediante HDL de un procesador especifico (CPU) que puede ser personalizado para una aplicación especifica y es implementado mediante un ASIC ó una FPGA. Los softcores tienen muchas ventajas sobre los procesadores convencionales porque a través de un diseño personalizado para cada aplicación permiten reducir costos, brinda flexibilidad en el desarrollo, inmunidad a la obsolescencia, etc. [18].

3.37 NIOS II

Arquitectura embebida de 32 bits diseñada para la familia de FPGAs de Altera. Su set de instrucciones es RISC y es personalizable usando el SOPC Builder de Quartus IDE. Es licenciable a terceros, lo cual permite llevar un desarrollo de etapa de prototipo (FPGA) a producción en masa (ASIC) [19].

3.38 uC/OSII (Micro-Controller Operating Systems Version 2)

Es un sistema operativo para microncontroladores, microprocesadores, DSP y actualmente se encuentra en su versión tres. Es un sistema operativo en tiempo real escrito en ANSI C, escalable, portable, predictivo y deterministico. Puede manejar más de 250 tareas ó hilos de ejecución, banderas de eventos, semáforos de exclusión mutua, manejo de memoria y gestión de recursos compartidos entre algunas de sus características. Es aprobado por la FDA para el desarrollo de dispositivos médicos [20].

3.39 *DRIVER* Ó CONTROLADOR

Un controlador (driver), es un programa ó librería que controla un tipo particular de dispositivo conectado a la unidad de procesamiento. Típicamente un controlador se comunica con el dispositivo a través del bus del sistema ó subsistema de comunicación a través del cual se conecta el dispositivo hardware. Cuando un programa invoca una rutina del controlador, este envía comandos al dispositivo, y cuando el dispositivo envía datos al host, el driver decodifica dicha información para su uso en la aplicación. Los controladores difieren según el hardware, y usualmente son específicos de un sistema operativo.

El objetivo de un controlador es simplificar el trabajo del programador actuando como traductor entre un dispositivo hardware y las aplicaciones que lo usan. Los programadores pueden entonces escribir código de alto nivel independientemente del hardware que posee el usuario final [12].

3.40 SPI (SERIAL PERIPHERAL INTERFACE)

Es un protocolo en comunicaciones digitales usado para transferir información entre dos dispositivos electrónicos. El bus SPI es un enlace serial de datos el cual puede trabajar transmitiendo datos en modo Full-Dúplex. Los dispositivos se comunican de modo maestro-esclavo en donde el maestro es el encargado de iniciar el envió de datos [21].

3.41 BAUDIOS

También conocida como velocidad de modulación. Es la velocidad a la que cambia el nivel de la señal de información. Esta velocidad es equivalente a un elemento de la señal por segundo es decir en el caso particular equivale a la tasa de bits por segundo [13].

3.42 FULL DÚPLEX

Transmisión de datos que puede ir en dos direcciones a la vez. Una conexión de Internet que utiliza el servicio DSL es un ejemplo de full dúplex. [11]

3.43 ASÍNCRONO

En términos de transmisión de datos, asíncrono significa que no se necesita un reloj ni otras fuentes de temporización para que el emisor y el receptor estén sincronizados.

3.44 PCB (Printed Circuit Board)

Es una tarjeta ó placa para realizar la conexión de los distintos elementos que conforman un circuito y las conexiones eléctricas entre ellos. Sirve para conectar mecánicamente y eléctricamente componentes electrónicos a través de rutas ó pistas de material conductor, grabados en hojas de cobre laminadas sobre un sustrato no conductor.

3.45 GPIO (GENERAL PURPOSE INPUT OUTPUT)

Es un pin genérico en un circuito integrado cuyo comportamiento se puede controlar por un usuario en tiempo real. Se pueden configurar como entrada ó salida y de igual forma reciben ó entregan valores digitales de voltaje en sus terminales[22].

3.46 STRING

Es una secuencia ordenada de bytes de longitud variable llamada conocida comúnmente como cadena de caracteres. Estos bytes son guardados como un vector de caracteres (en código ASCII) terminados con el carácter NULL ‘\0’ ó ACII ‘0’. El carácter NULL indica donde termina donde termina la secuencia de caracteres.

4. DESARROLLO DE LA PRÁCTICA

4.1. CUMPLIMIENTO DE OBJETIVOS

Antes de iniciar la práctica empresarial en la UEE Bioingeniería, se realizó de forma voluntaria una pasantía vacacional en el mes de diciembre en la cual se comprendieron conceptos básicos de redes de datos y transmisión de información mediante protocolo TCP/IP los cuales se debían tener muy claros para la comprensión del sistema de transmisión que se quiere implementar en el dispositivo medico. Fue de gran utilidad el afianzamiento de estos conceptos para la adecuada comprensión de las hojas de datos de los diferentes dispositivos de comunicación estudiados. Adicional al estudio de estos protocolos se analizó el driver ya implementado en el dispositivo Ethernet el cual ya se encontraba funcionando en el monitor. Se realizaron pruebas simples de transmisión de datos por protocolo TCP en modo cliente.

Se inicia la práctica estudiando conceptos básicos de redes de transmisión de datos. Posterior a la lectura de esta documentación de inicia con el estudio de la hoja de datos del modulo Wi-Fi. Este módulo tiene diferentes modos de configuración como SPI, UART, etc. pero como este modulo estaba ya implementado en una PCB ya diseñada y fabricada para el dispositivo médico, la decisión del equipo de diseño y desarrollo previa a la llegada del practicante fue configurarlo mediante UART debido a que solo se necesitan 2 señales digitales desde la FPGA para controlar el módulo (Tx y Rx). Para iniciar con el desarrollo del driver es facilitado al practicante un modulo UART en Verilog (HDL) para que sea implementado en la en la FPGA y se comunique con el procesador NIOSII y así realizar la configuración del modulo desde el procesador.

Después de realizar pruebas con el módulo a altas velocidades se obtuvieron muchos errores en la transmisión y recepción debidos al diseño en HDL del módulo. Por lo anterior se tomo la decisión de hacer un nuevo módulo en HDL para mejorar el error a altas velocidades en la transmisión de datos que se podrá evidenciar cuando el modulo transmita datos directamente por UART a la red.

Después del diseño y pruebas del nuevo módulo se procede con la realización de pruebas básicas de configuración mediante los comandos que se indican en el datasheet del módulo Wi-Fi. Se logra una transmisión simple de datos mediante un módulo UART-USB conectado a un computador convencional. Seguido a esto y ya con los pasos necesarios para la transmisión definidos, se comienza con la realización del driver en lenguaje C. De nuevo se hacen múltiples pruebas con el driver ya implementado en la FPGA y se corrigen errores es los tiempos de envió y conexión al AP (Access Point).

Utilizando el moóulo UART en HDL que se utilizó para hacer el driver del módulo Wi-Fi, se realizó la instanciación de otro módulo para la comunicación del módulo 3G-GPS. Se realizaron pruebas directamente en la PCB del dispositivo medico para comprobar además del funcionamiento del modulo 3G, los distintos traductores de voltaje y demás elementos de hardware que fueron implementados

en la PCB para el funcionamiento adecuado del módulo. Seguido a estas pruebas se procede con el estudio y familiarización de los comandos AT de configuración básica para transmisión de datos por protocolo TCP/IP, indicados en la hoja de datos del modulo.

Para el driver en código C se utilizaron la mayoría de los algoritmos diseñados previamente para transmisión y recepción de cadenas de caracteres en el módulo Wi-Fi pero esta vez con los comandos indicados en la hojas de datos del módulo de transmisión de datos vía celular dada la similitud en su forma de configuración para estos módulos. Estos comandos debieron ser estudiados a profundidad debido a la aparición de nuevos conceptos como configuración del APN según el operador, intensidad de la señal 3G, abrir un socket simple para la conexión con el servidor de red celular, etc. El diseño del *driver* culmino satisfactoriamente con el envío y comprobación de datos mediante protocolo TCP a un servidor instalado en un computador personal con una IP pública.

Finalmente para integrar estos módulos en el dispositivo médico se tuvieron que implementar los módulos UART diseñados y probarlos con todo el sistema y la descripción de hardware existente en la FPGA. Como la implementación final se hizo sobre Websocket.cpp, una librería hecha en C++ sobre la cual solo funcionaba el módulo Ethernet, se realizó un estudio de las diferentes capas de software del sistema y las tareas ejecutadas por el sistema operativo uC/OSII para así trabajar dentro del código del dispositivo médico implementando las nuevas funcionalidades 3G-GPS y Wi-Fi de los nuevos módulos sin afectar el sistema actual. Para esto se estudio el empaquetado y envió de datos mediante protocolo Websocket y se trabajo en conjunto con el ingeniero de sistemas encargado de crear y administrar el servidor Web con el cual se comunica el dispositivo médico. Así culmina la práctica dando un gran aporte al conocimiento de la FCV y haciendo posible el uso de tecnologías para transmisión de datos mediante dispositivos inalámbricos de última generación.

4.1.1 TAREAS DESARROLLADAS

4.1.1.1 Familiarización con el objetivo final

Como paso inicial se observa de forma general el esquema de conectividad que se quiere lograr a través de los diferentes medios de transmisión de datos con los que cuenta el dispositivo médico y sus posibles funcionalidades. Así la FCV logra estar a la vanguardia en este tipo de desarrollos para dispositivos médicos de monitorización vital con conectividad a la red para uso remoto.

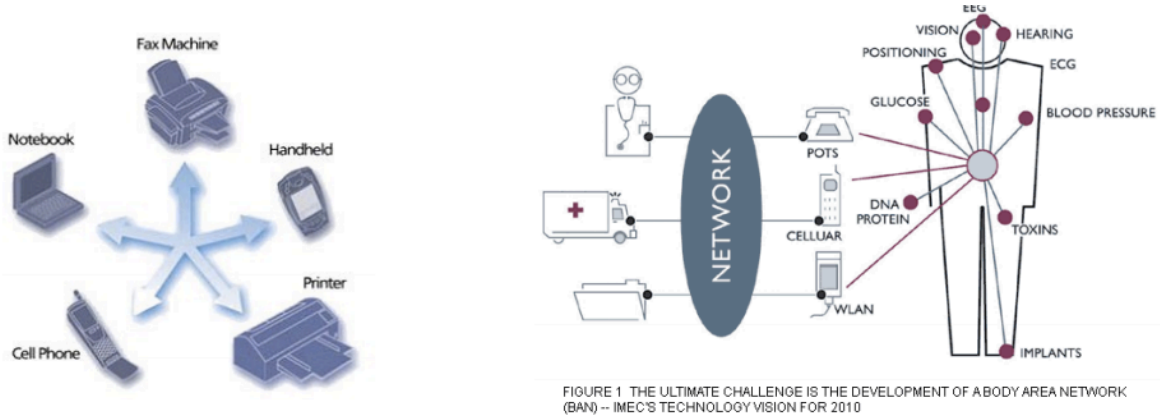


Figura10. Esquema de conectividad a lograr mediante diferentes dispositivos para monitorización remota[23]

4.1.1.2 Estudio de las capas de comunicación en Redes de Datos

Para el estudio de las capas de comunicación en Redes de Datos como paso inicial para comprender cómo funcionan los sistemas de comunicación actualmente, se adquirieron importantes conceptos a partir de material académico facilitado por el Ing. José Pablo Pinilla. Esta información fue tomada del curso de Fundamentos de Internet, de la Universidad de Western Ontario y se reforzaron estos conceptos con otras presentaciones sobre redes que el docente Jhon Jairo Padilla de la Universidad Pontificia Bolivariana Seccional Bucaramanga tiene en su página Web. Esto fue muy importante debido a que no se tenían los conocimientos mínimos necesarios acerca de cómo funcionan los sistemas de comunicación y gestión de la información en internet para comenzar con el estudio de las hojas de datos de los diferentes módulos de comunicación. Debido a lo anterior en esta etapa se estudió a profundidad las diferentes capas que gestionan la información en las redes de datos tanto en el modelo OSI como en el modelo TCP/IP que son utilizadas en internet para comunicar sus distintos puntos remotos.

Para este estudio se comenzó por comprender la utilidad de las distintas capas del modelo OSI y cómo actúa cada una de ellas en el proceso de comunicación entre dos puntos. Dependiendo del sentido en el que se envía la información estas capas funcionan de la forma en que observamos en la Figura 11 (marco teórico).

Es muy importante comprender este flujo de información porque dependiendo del dispositivo a usar se puede tener fácil acceso a cada una de las capas ó puede que el acceso sea realizado de forma autónoma por el módulo haciendo que el acceso sea transparente para los usuarios y el programador. Por ejemplo el integrado Wiznet5100 (Ethernet) permite el acceso y programación total de las capas superiores (red hasta aplicación) mientras que el modulo Wi-Fi permite su fácil programación mediante cadenas de caracteres, pero no cuenta con fácil acceso a todas las capas debido al stack TCP/IP que viene implementado de fabrica para hacer el proceso mas fácil para el programador.

Actualmente se toma el modelo simplificado TCP/IP como reemplazo equivalente del modelo OSI debido a que en muchos casos varias capas de integran en un mismo proceso. En la Figura 3 se puede observar en cuál sería el equivalente a estas capas entre los dos modelos. En la parte derecha se encuentra el modelo OSI y en la izquierda el modelo equivalente TCP/IP.

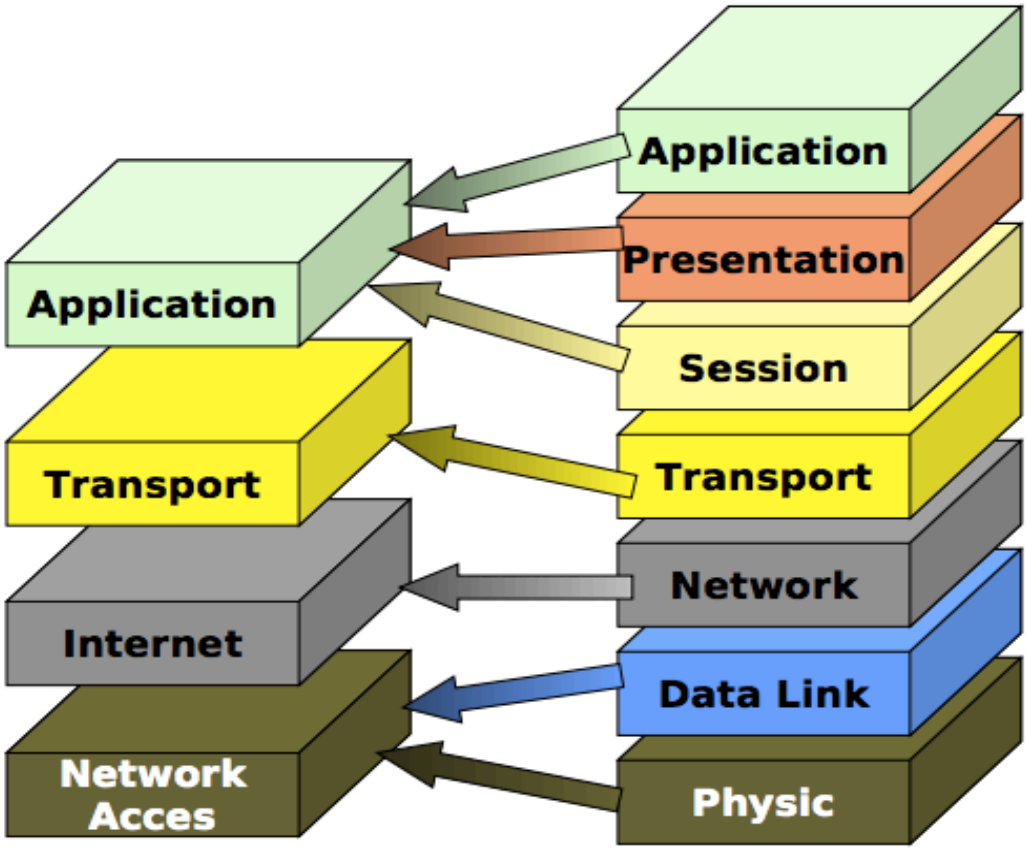


Figura11. Diagrama equivalente entre modelo OSI y TCP/IP [23]

En este breve estudio se identificaron las distintas capas de los modelos mencionados y la forma donde están ubicadas en los distintos drivers desarrollar. Por ejemplo para la capa física y de enlace de datos en el dispositivo Ethernet se evidencia en forma de cables con distintos tipos de modulaciones. Llega a un conector RJ-45 que tiene algunas bobinas que adaptan el voltaje de entrada para

que pueda funcionar de forma adecuada en el integrado. Por otro lado el dispositivo Wifly de la empresa Roving Networks (Wi-Fi) tiene involucrada esta capa pero de manera inalámbrica, transmitiendo los datos por el aire mediante RF (2.4GHz) utilizando un protocolo ya estandarizado por el IEEE llamado 802.11 el cual hace compatible esta tecnología con muchos otros dispositivos que también soporten Wi-Fi como Smartphones, laptops, etc. De forma similar el dispositivo 3G recibe y transmite a la red mediante su capa física inalámbricamente pero usando las redes celulares que funcionan a otras frecuencias y con distintos tipos de modulación.

Es importante concluir que la capa física sin importar el dispositivo es transparente para el usuario y para el programador porque los circuitos integrados que brindan la conectividad (Ethernet, Wi-fi y 3G) están listos para tomar la información de la red directamente desde sus registros, por lo cual el programador solo toma esta la información comunicándose con el módulo de comunicación y a su vez el módulo mediante su stack TCP/IP gestiona los distintos protocolos con la red. Un ejemplo de esto se observa en la facilidad para enviar datos que da el stack de protocolos TCP/IP con el que cuentan los módulos al transmitir información de forma natural sin necesidad de que el programador deba preocuparse por protocolos de diferentes capas como IP, DHCP, TCP, etc. Por esto es de vital importancia las entender los diferentes comandos de configuración de cada hoja de datos y poder lograr diseñar los 3 drivers que en su etapa final deben funcionar de forma transparente para el usuario.

Después de analizar la capa física de cada uno de los dispositivos con los que cuenta el dispositivo medico, pasamos al estudio de las capas superiores a las cuales si tenemos acceso. Estas capas se deben analizar a profundidad con el fin de diseñar los drivers de forma eficiente, autómata y libre de errores.

La capa de red ó capa IP es la capa que ayuda a entender cómo funciona internet y permite empezar a comprender como se enviarán los datos del dispositivo medico a través de la red. Todo funciona a partir de protocolos y sockets. Un socket es un enlace final de comunicación entre un punto remoto y otro. Actúan en el proceso un cliente y un servidor (punto remoto con IP fija). En la figura 12 se tiene la forma básica de una arquitectura cliente servidor.

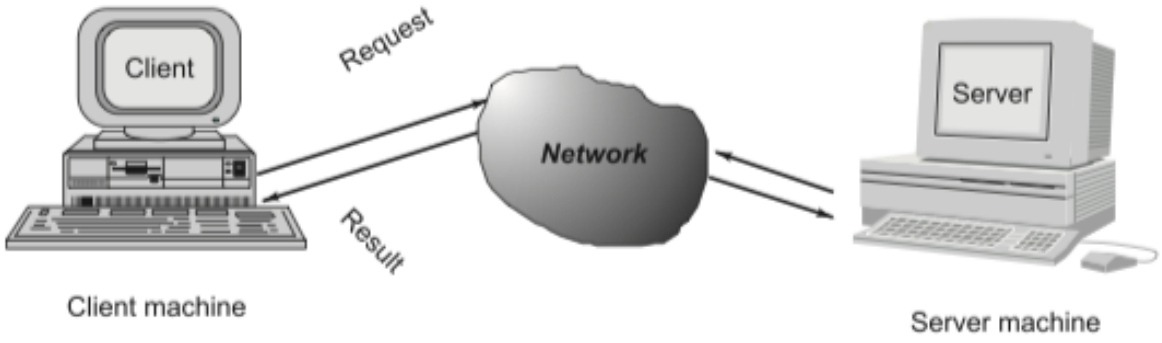


Figura12. Esquema de una topología típica cliente servidor[24].

La arquitectura cliente servidor se afianzo de forma práctica al observar cómo está funcionando el servidor Websocket que sirve como punto de llegada de los datos del dispositivo medico. A partir de estos datos un ingeniero de software realizó la programación necesaria mediante protocolo Websocket para que estas señales se puedan visualizar en un navegador web desde cualquier computador o dispositivo móvil.

Cada punto final de conexión tiene una IP y un puerto y esto es lo único que necesita cada uno para ubicarse en internet. Esta información se envía mediante paquetes de datos IP los cuales se comunican en la red mediante el número de la dirección IP (4 números de 8 bits) y su dirección de destino, todo lo anterior de forma binaria. En la Figura 13 se muestra la distribución de un paquetes IP en la versión IPV4. El paquete IP principalmente lleva la información de la IP de destino (dirección hacia dónde va dirigido) y la información útil que se quiere comunicar. Adicionalmente lleva otro tipo de información necesaria como tiempo de vida del paquete, check sum para asegurarse que el paquete llega satisfactoriamente al destino, etc.

Por ahora se concluye que en cada uno de los puntos de envío y recepción de datos (cada uno con dirección IP y un puerto), se cuenta con dos sockets de conexión los cuales se comunican mediante un mismo protocolo. Cada paquete de datos tiene un cliente y un puerto específico al cual apunta (dentro de un socket) como se observa en la figura 13. Los puertos pueden ser vistos como un canal por donde viaja determinado tipo de información, de hecho algunos puertos ya tienen asociados ciertos servicios como navegación web mediante protocolo http, chat, correo, etc. Los servicios que no tienen un protocolo asignado pueden utilizar cualquier puerto libre siempre y cuando la red local (router, proxy, ISP, etc.) tenga habilitado (abierto) el puerto.

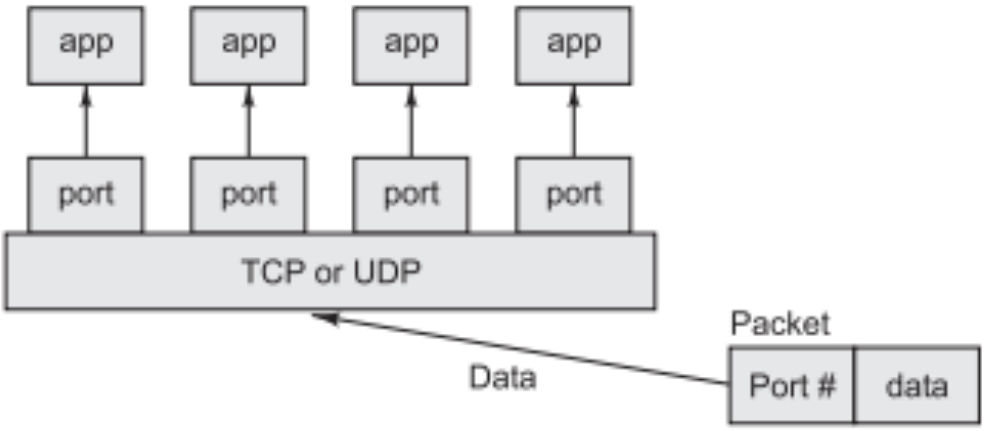


Figura13. Esquema del socket como punto de conexión remoto entre un cliente y un servidor[24]

Cuando el servidor establece la conexión con el cliente se escucha por un puerto único que cambia apenas se establece la conexión para así poder desocupar el puerto de escucha y recibir solicitudes de otros clientes como se muestra en la Figura 14.

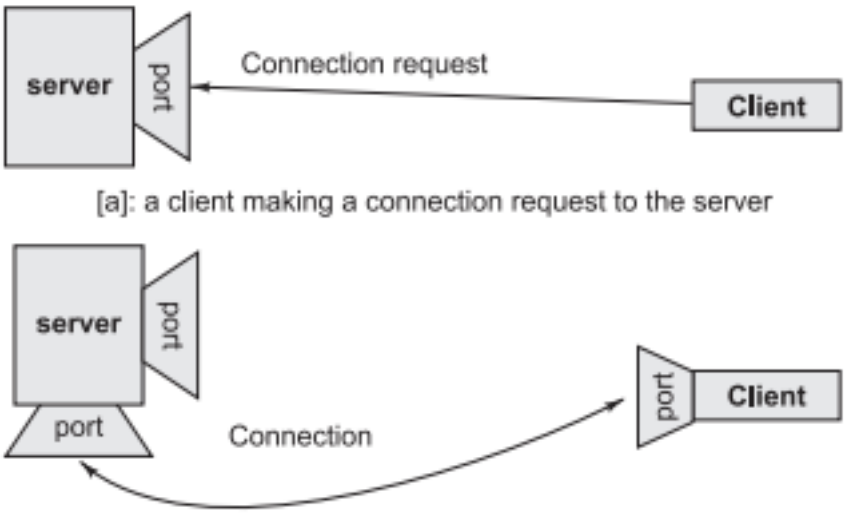


Figura14. Manejo de puertos de un servidor socket [24].

Para abrir los canales de comunicación de cada dispositivo de red se debe abrir un socket. Para esto se debe especificar si se quiere transmitir los datos por

protocolo UDP ó TCP y el numero del puerto. Seguido a esto se establece una comunicación típica cliente-servidor como la que se muestra en la figura 15.

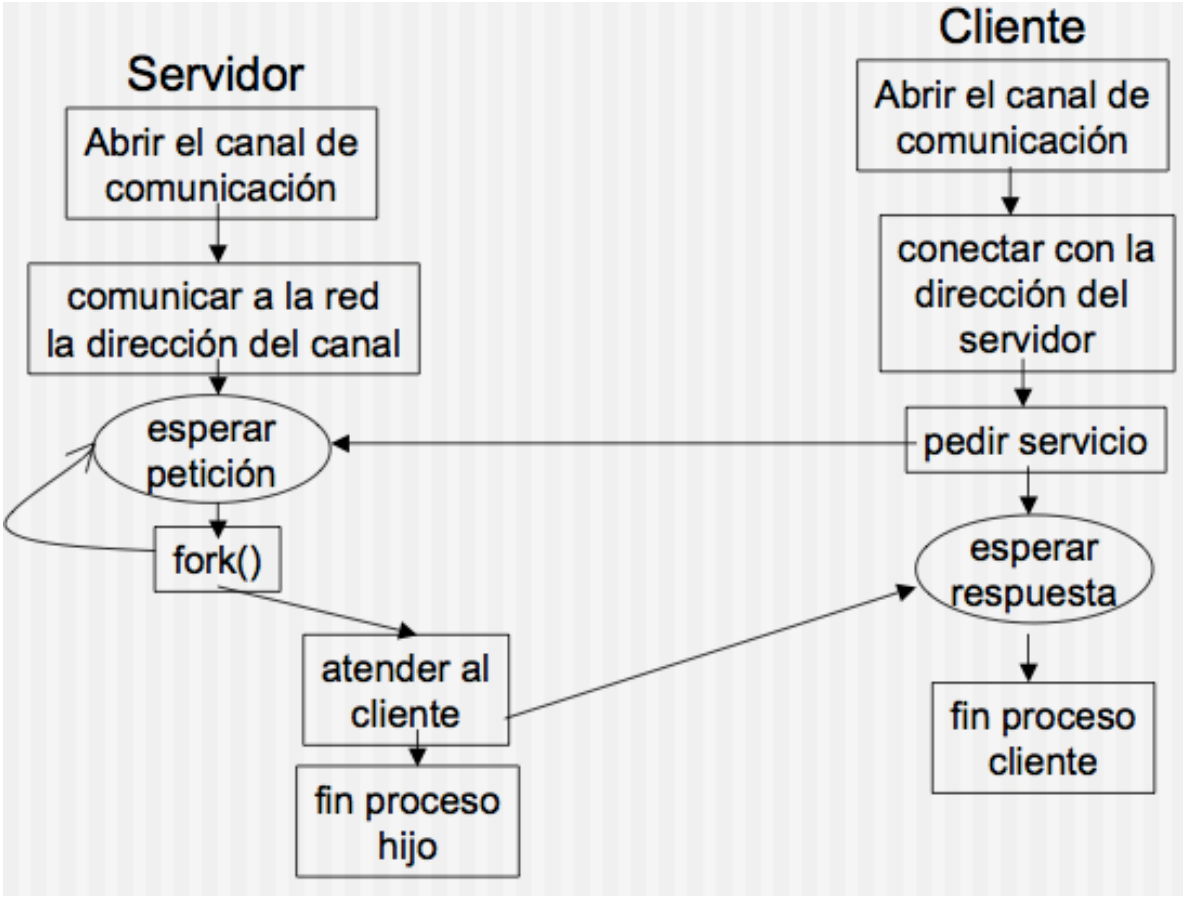


Figura15. Protocolo básico para una comunicación a través de sockets [25]

Cuando la conexión está establecida se tienen que tener en cuenta los tiempos denominados “timeout” porque después de que se envía un “frame” IP se debe habilitar un contador que nos indique el tiempo máximo que puede tardar el servidor en responder con un código “ACK” (código de confirmación) que generalmente es la suma binaria del frame como confirmación de que los datos llegaron de forma correcta (aplica para transmisiones por protocolo TCP como el implementado en el servidor Websocket de la FCV). Esto se puede observar en la figura 15. y como se explica más adelante se tuvo problemas con este acuse de recibo en el driver del módulo 3G.

De forma general se muestra el esquema de las capas afectadas en el desarrollo del protocolo de comunicación basado en sockets en la figura 16.

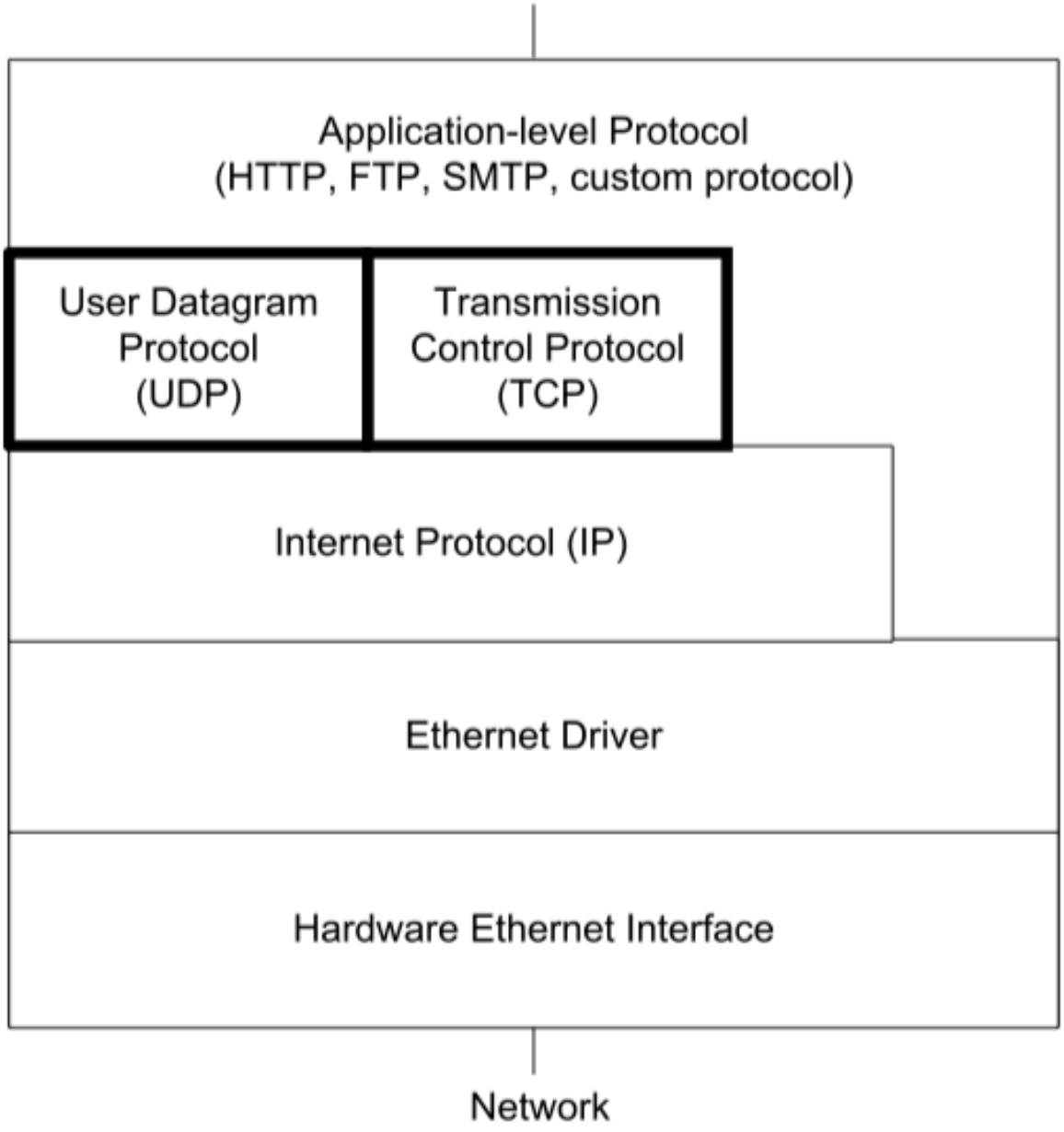


Figura16. Esquema del acceso a las capas desde el driver de cualquier dispositivo[26]

4.1.1.3 Estudio de sistemas multiprocesador para el dispositivo médico

Para lograr este objetivo fueron facilitados los informes hechos por el practicante Leyner Camargo quien fue el encargado de seleccionar cada uno de los dispositivos de comunicación, documentar el sistema multiprocesador del monitor de signos vitales y realizar el driver Ethernet que conecta el monitor con el servidor Websocket con sus respectivos informes.

En el informe del procesador multicore se evidencia y se comprende la necesidad de contar con un sistema multinucleo embebido en la FPGA junto con un sistema operativo (uC/OS2 que se analizará más adelante).

Actualmente los sistemas con múltiples procesadores poseen mayor rendimiento en la ejecución de tareas, debido a que cada procesador las hace de forma independiente aumentando la velocidad de ejecución de cada una de las tareas brindando ventajas en el procesamiento de datos.

Para el diseño de MSV signcare se hace necesaria la implementación de este sistema multinucleo debido a la gran cantidad de tareas que se tienen que ejecutar. El procesador #1 (CPU1) será el encargado de la adquisición de las señales fisiológicas, procesamiento de imágenes e interfaz gráfica, entre otras cosas estas tareas se incluyen en la visualización de señales, la configuración de las tareas de medición de parámetros fisiológicos, etc. La visualización de realizara por medio de una pantalla touchscreen resistiva. Al procesador #2 (CPU2) le corresponde el empaquetamiento y envió de datos por medio de los estándares Ethernet, Wifi, 3G y Bluetooth así como la georeferenciacion mediante el módulo GPS. Esta distribución de tareas se puede observar en la Figura 17.

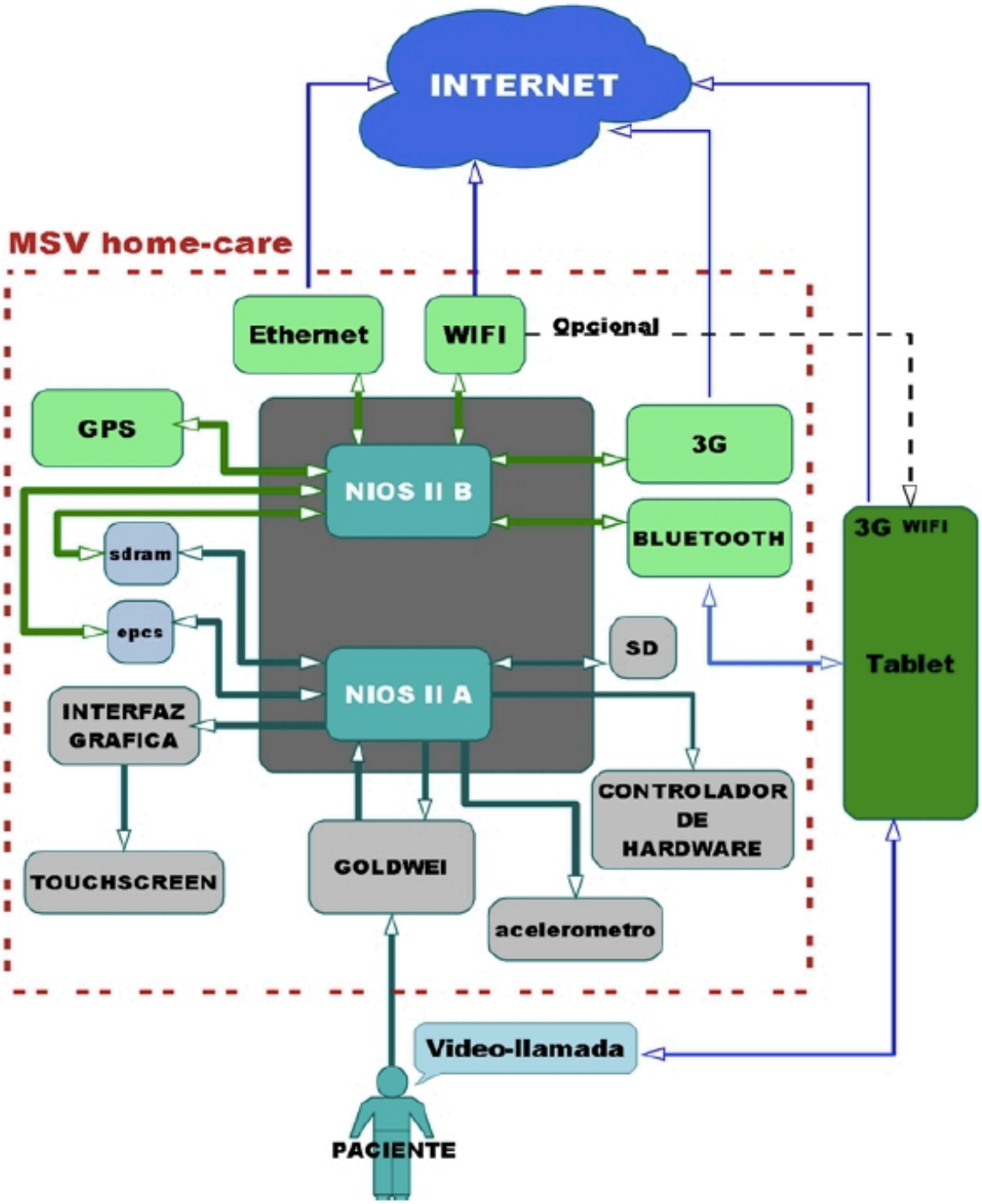


Figura17. Esquema general distribución de las tareas y el hardware dentro la FPGA [27]

En los sistemas multiprocesados normalmente existen recursos compartidos, uno de los más comunes es la memoria. Para hacer la gestión de está, se emplea usualmente un módulo en hardware llamado MUTEX (mutual exclusion) que se opera bajo el sistema operativo uC/OSII de la empresa Micrium. Este módulo permite la gestión de este recurso compartido permitiendo que solo un maestro acceda a él en un instante de tiempo determinado. Luego de utilizar el recurso el maestro libera el MUTEX para que otro pueda usarlo. De esta forma se evita interferencia en la ejecución de las tareas de cada procesador, y la corrupción de datos que es un común cuando se comparte memoria. Sin esta coordinación en el acceso a los recursos compartidos el sistema puede colapsar fácilmente debido a que este trabaja en forma desorganizada generando ineficiencia en el procesamiento de tareas.

El procesador #1 (CPU1) será el encargado de la adquisición de señales fisiológicas, procesamiento de imágenes e interfaz grafica. La visualización será realizada por medio de una pantalla touchscreen.

Al procesador #2 (CPU2) le corresponde el empaquetamiento y envió de datos por medio de los estándares Ethernet, Wifi, 3G y Bluetooth así como la georeferenciacion mediante un modulo GPS. Esta distribución de tareas se puede observar en la figura 12.

El Mutex permite que estos recursos se compartan eficientemente evitando problemas de corrupción de datos en la memoria. El mutex funciona como un semáforo que gestiona el uso de recursos compartidos. En el sistema del monitor de signos vitales se implementa Mutex para el SPI del módulo Ethernet, para la lactur y escritura de la tarjeta SD y del Mailbox (mensaje enviados entre las dos CPUs)

Un esquema que ilustra cómo están compartidos estos recursos dentro de la FPGA se muestra en la Figura 18.

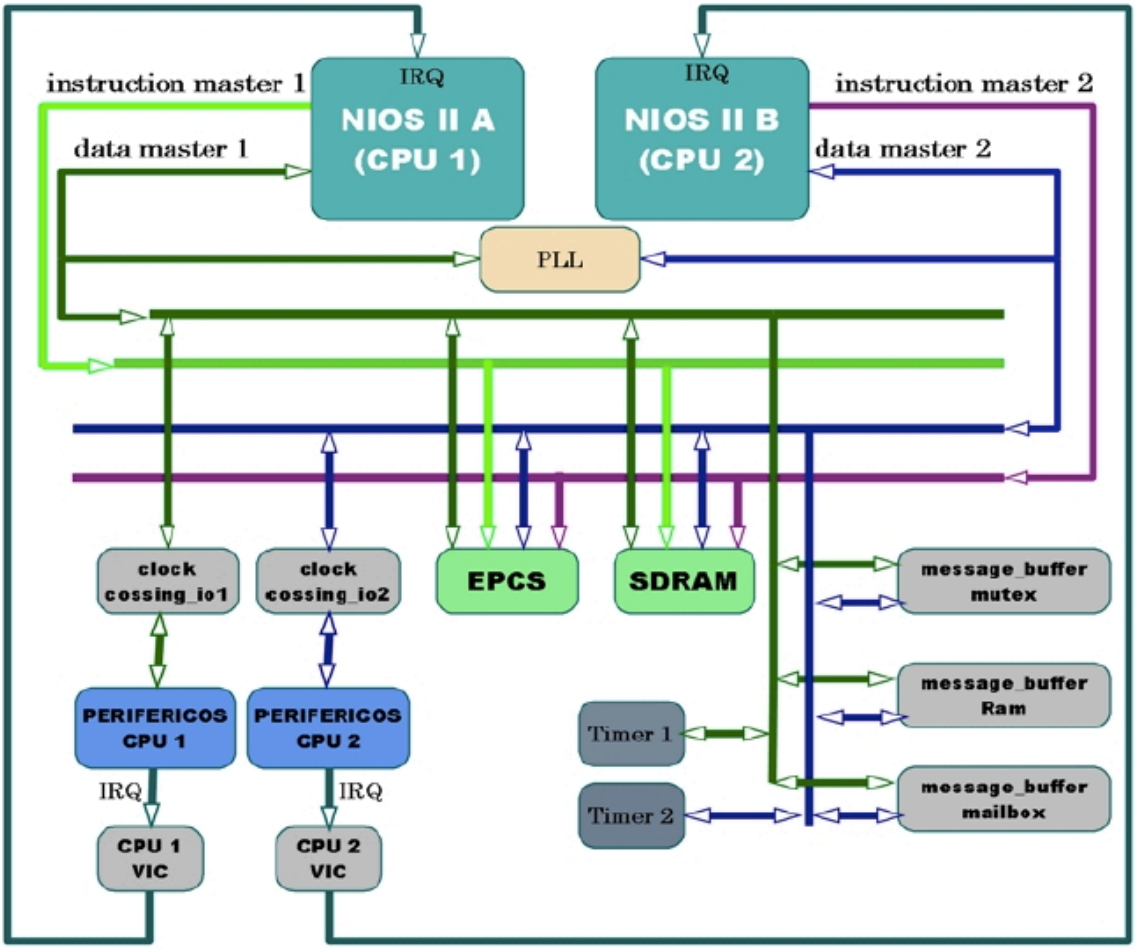


Figura18. Esquema general de los recursos de hardware dentro la FPGA [27]

Adicionalmente se observó el informe detallado que se realizó para la selección de los módulos para la comunicación y se identifico como criterio el bajo consumo en potencia dado que el monitor de signos vitales en muchas ocasiones se alimentará con una batería (interna) y se le quiere dar la mayor autonomía posible al dispositivo médico. El dispositivo Wi-Fly de la empresa Roving Network es uno de los de menor consumo energético en el mercado aprovechando por ejemplo la función “sleep” que mantiene el módulo en el modo ultra bajo consumo mientras no se esté utilizando. El dispositivo 3G se eligió aprovechando que cuenta con GPS integrado y esto servira para dar soporte para la georenferenciación en todo momento (en el mismo modulo). Por esto se tomo la decisión de utilizar 2 módulos UART independientes porque el modulo 3G-GPS debe estar activo en todo momento si el usuario lo requiere. Por ejemplo si el Wi-Fi esta encendido también se podría llegar a solicitar el servicio de Georeferenciación por lo cual se debe tener encendido el módulo 3G-GPS al mismo tiempo, por lo tanto se necesitan los 2 módulos UART independientes.

4.1.1.4 Estudio módulo Ethernet – Wiznet (W5100)

Ahora teniendo claro de qué forma está dispuesto el sistema multinucleo se puede empezar a analizar el sistema Ethernet. Tal como se dijo anteriormente el objetivo del practicante es hacer un sistema análogo al Ethernet pero funcionando en el dispositivo Wi-Fi y 3G. Para el adecuado estudio del funcionamiento de este módulo y dar conectividad Ethernet al dispositivo medico fue necesario retomar algunos conceptos estudiados de redes (los explicados anteriormente) tales como socket, DHCP (protocolo para la asignación de direcciones IP), envió de datos en la capa de transporte por UDP ó TCP, etc.

Fue de vital importancia analizar el funcionamiento del driver ya implementado en el monitor así como el análisis de la hoja de datos del Wiznet5100, debido a que el practicante tiene como tarea realizar un driver que funcione exactamente igual al ya implementado para Ethernet mediante el Wiznet5100 y que adicionalmente sea transparente para el usuario final.

El dispositivo Ethernet Wiznet W5100 tiene implementada una comunicación SPI junto a la CPU2 (procesador de comunicaciones). Esta comunicación SPI sirve para la configuración y el envió de datos por medio de protocolo TCP configurando su respectivo socket.

Para la comunicación SPI se utilizó directamente un módulo propietario de la compañía Altera que es licenciado mediante la compra de la licencia de utilización del procesador NIOS2, es decir si se compra la licencia del procesador NIOSII de Altera éste módulo se podrá usar sin ningún inconveniente ó limitación. Lo anterior quiere decir que no existe hardware de por medio entre el Wiznet y la CPU2 porque se utilizó este módulo que fue hecho en software por la compañía y va embebido en el procesador NIOS2. La figura 14 muestra la forma de conexión del W5100.

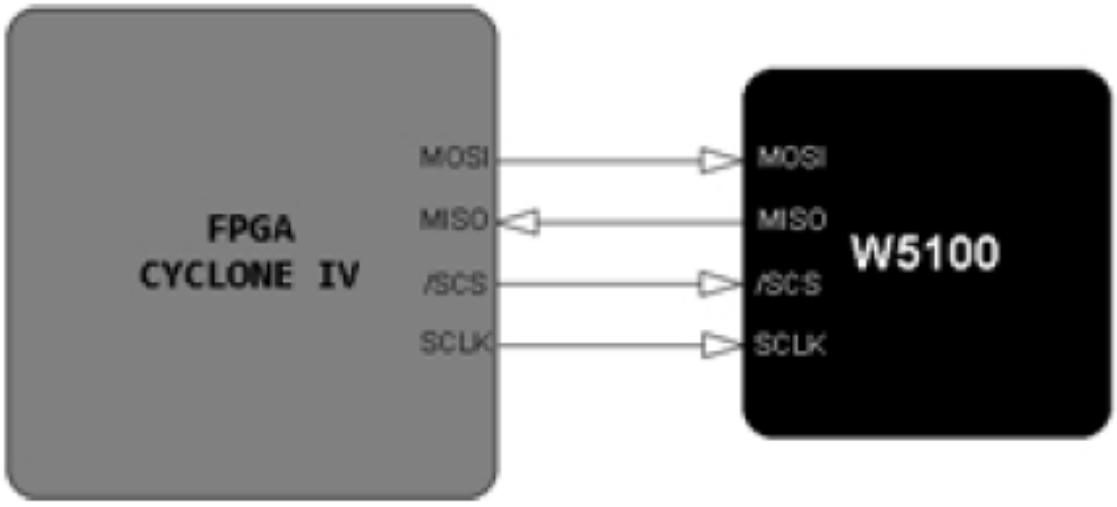


Figura19. Diagrama de bloques entre el dispositivo Ethernet y la FPGA [27]

El módulo Ethernet carece de funcionalidades de alto nivel. Por ejemplo el protocolo DHCP para adquirir una IP se hace “manualmente” es decir no existe ninguna función en su hoja de datos para hacerlo automáticamente, entonces el practicante que desarrolló este driver tuvo que ejecutarlo y adquirir la IP haciendo uso de este protocolo con este servidor. Los módulos Wi-Fi y 3G a utilizar tienen esta función de conexión a internet mediante DHCP directamente como una funcionalidad.

El dispositivo se configura escribiendo los datos adecuados en los registros de configuración que el datasheet indica. Se escribe la palabra digital deseada y se especifica el registro y así se empieza a configurar el “socket” mediante el protocolo SPI. Para abrir un socket son necesarios una serie de pasos y entender cómo funciona el abrir un socket por este medio para posteriormente hacer las funciones equivalentes (probablemente no existan) en el modulo Wi-Fi. El W5100 tiene la capacidad abrir 4 sockets (distintos protocolos con diferentes puertos). En la parte aplicativa del monitor de signos vitales únicamente se utilizara un socket al mismo tiempo por lo que esto no es de vital importancia. Para abrir un socket bien sea como cliente o servidor es de vital utilidad analizar el siguiente diagrama de la figura15.

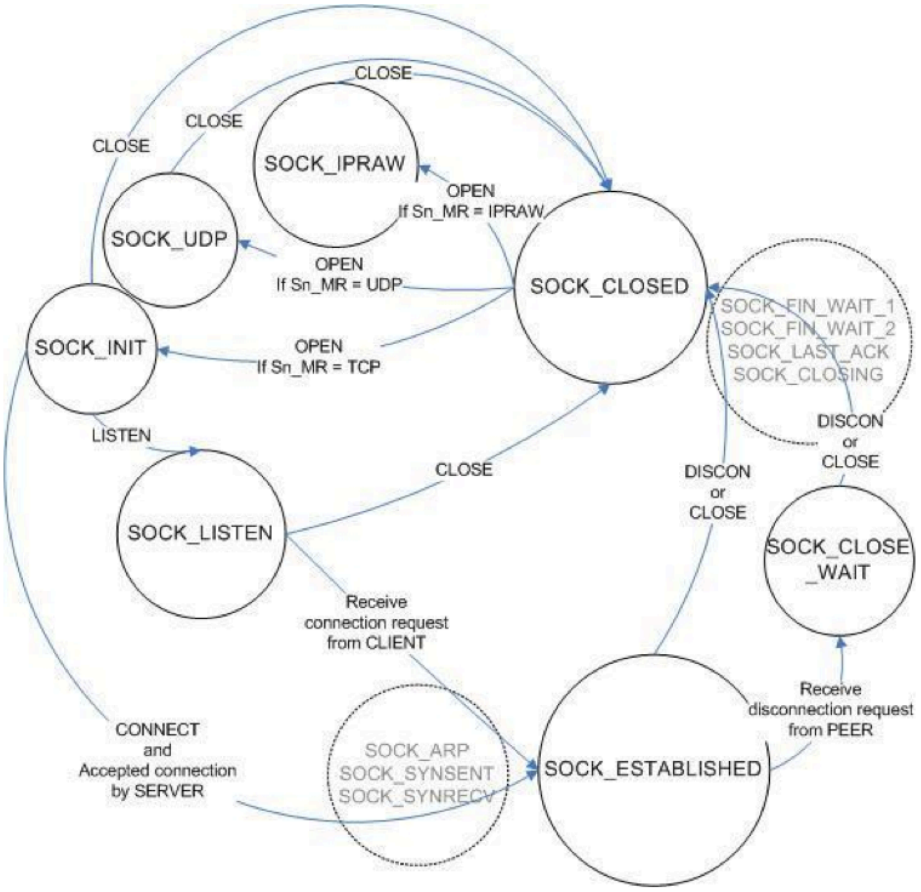


Figura 20. Diagrama de estados para establecer un socket en el W5100 [28]

Siguiendo el anterior diagrama de estados y revisando los registros respectivos de estado (solo lectura), se puede saber en qué estado se encuentra la conexión. Dependiendo si se quiere requiere ser cliente o servidor se puede establecer una correcta comunicación abriendo ó cerrando los sockets. Posteriormente se observara que el socket debe estar abierto para poder iniciar una conexión TCP y una posterior conexión al servidor Websocket. En la figura 16 se ve claramente donde se encuentran ubicadas las capas desde la física hasta la de aplicación y el driver en donde se encuentra ubicado. De forma clara se observa en donde estaría ubicada la app Websocket para comunicar la aplicación y el driver que traduce los datos del Wiznet.

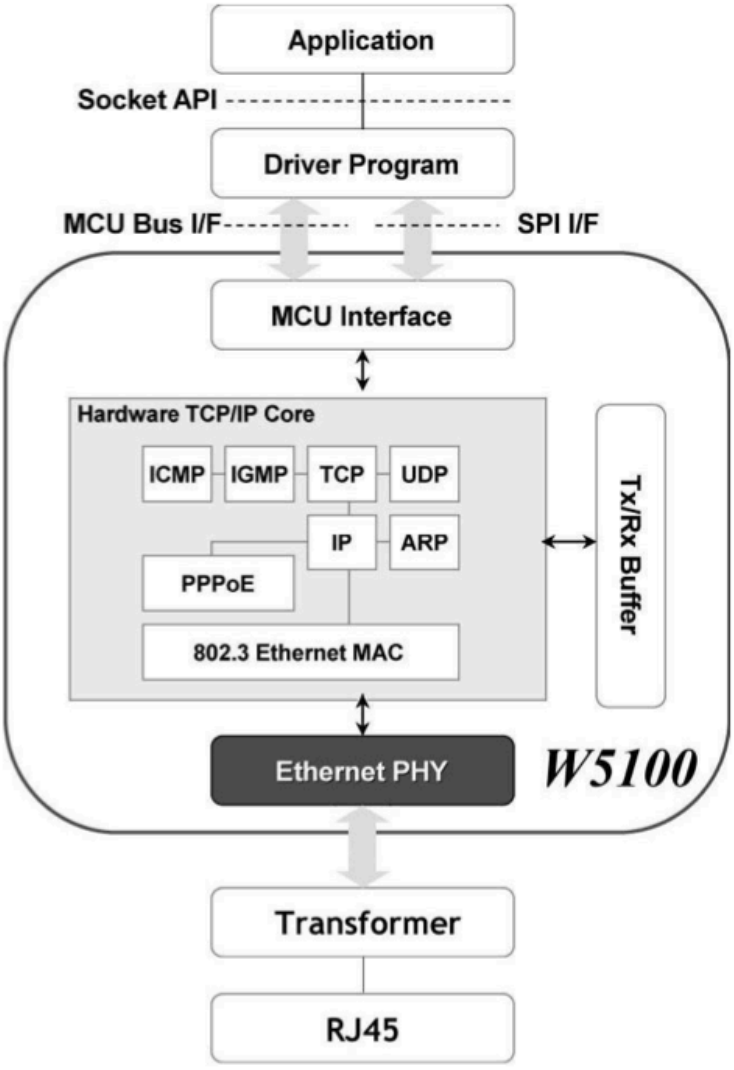


Figura 21. Diagrama de bloques entre el dispositivo Ethernet y las capas afectadas[28]

De forma general el módulo Wiznet W5100 cuenta con dos conjuntos de registros que describen y permiten configurar todo el módulo. El primer conjunto de ellos es el “common register” y permite inicializar cada socket con sus parámetros tales como dirección MAC, mascara de subred, dirección IP de destino, timers, modo de envió de datos (TCP ó UDP) , etc. El segundo conjunto es “Sockets Registers” los cuales me permiten acceder a la configuración de cada socket (de un total de 4) es decir, a conectar el socket a un punto remoto, desconectarlo, verificar el estado de la conexión, etc.

4.1.1.5 CONFIGURACIÓN INICIAL DEL MÓDULO WI-FI

El funcionamiento del módulo Wifly que brinda la conectividad Wi-Fi al monitor de signos vitales a desarrollar es comercializado por la empresa Roving Networks y en general su configuración es más simple en comparación a la configuración realizada en el driver para el modulo Ethernet Wiznet W5100.

Este módulo puede comunicarse por SPI ó por protocolo UART. Para el caso del dispositivo médico en desarrollo, la board del dispositivo médico ya fue desarrollada para que el Wifly RN131C funcione directamente mediante protocolo UART. Es decir solo se conectaron 2 GPIO de la FPGA al modulo Wifly es decir Tx y Rx Por este motivo se conoce que los mensajes y todo el driver lo diseñaremos basados en las instrucciones a enviar mediante UART.

Para el afianzamiento con el módulo Wifly se le fue dado al practicante el módulo referencia Wifly RN131C de la empresa Roving Networks que es exactamente el mismo que tienen implementado en la board del dispositivo médico. El módulo de entrenamiento el cual fue utilizado en esta etapa para hacer pruebas, tiene una board fabricada por la compañía Sparkfun y es útil porque sus señales de salida se pueden conectar fácilmente a la FPGA ó a cualquier micro controlador.

Se realizó un análisis rápido y una pequeña investigación acerca de la configuración de este módulo y se concluye que de forma análoga al módulo Ethernet el módulo Wifly puede configurarse en modo comando (análogo al “command mode” en el W5100) y en modo datos (análogo a los “Sockets Registers”). A diferencia del módulo Ethernet no es necesario que ingresar la dirección de memoria a escribir, simplemente se le envían comandos mediantes cadena de caracteres (Strings) por protocolo UART y el módulo responde (devuelve datos) de acuerdo al parámetro ingresado, esto para saber que el dato fue enviado correctamente y así asegurar su correcta configuración.

En un datasheet anexo en la página de Roving Networks se consultaron los distintos modos de alergización del módulo debido a que dentro de su hardware interno cuenta con una fuente boost para poder seguir en funcionamiento cuando se tienen niveles bajos de voltaje. Adicionalmente para hacer pruebas rápidas de configuración y afianzamiento con el modulo se utilizó un módulo serial RS-232 a USB que se comunica con el módulo Wi-fly por medio de un software de comunicación serial ejecutado desde un computador ordinario. En este caso se utilizo el software “commoperator” el cual se trabajo desde una versión gratis pero

perfectamente funcional (cualquier software para recepción UART funciona de la misma forma). Los datos se ingresan por el usuario en el software commoperator y salen por el puerto USB y son transformados por el módulo RS-232-USB a protocolo UART para que le lleguen de forma correcta al modulo Wi-Fi. Este esquema se puede observar en la figura 18.

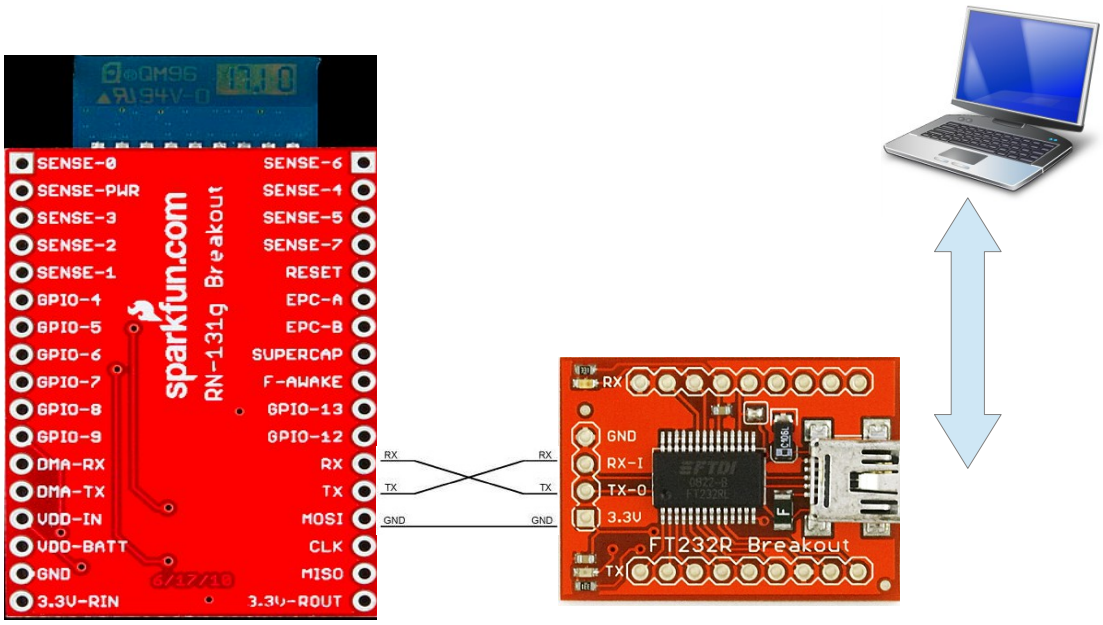


Figura 22. Diagrama de conexión del módulo Wifly para realizar pruebas con datos enviados desde un laptop.

Se debe configurar el software de comunicación UART a 9600 baudios que es la velocidad de transmisión y recepción configurada por defecto apenas se energiza el módulo de comunicación Wi-Fi (si aún conserva sus parámetros de fábrica).

El módulo se comunica mediante dos modos básicos: modo “datos” ó modo “comando”. Cuando el módulo se encuentra en modo datos configuramos todos los parámetros relacionados como nombre de la red Wifi (SSID), contraseña, tipo de seguridad de la red, etc. En el modo datos (si se ha configurado todo en el modo comando correctamente) se contaría con una conexión a internet esperando conexión ó intentando conectar con algún socket. Un ejemplo de los modos de configuración y el flujo de datos se puede observar en la figura 19.

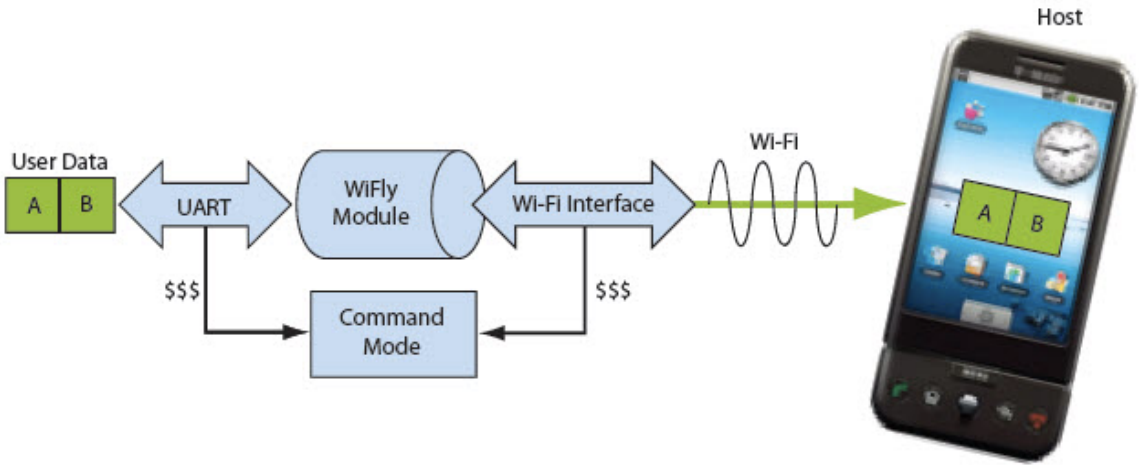


Figura 23. Diagrama de los dos modos de configuración del módulo Wi-Fly[29].

"User Data" en la figura anterior representa los datos de configuración que se le ingresan para configuración ó envió de datos (en este caso mediante un computador mediante el software commoperator y el módulo conversor USB-UART) dados por el usuario pero que en un futuro serán dados desde la FPGA de forma automatizada. El módulo por defecto entra en modo datos intentando conectar a una red inexistente llamada "roving". Para configurar los parámetros iniciales se debe ingresar al modo "comando" (recordemos que por defecto al energizar entra en modo "datos").

Para ingresar al modo "comando" se debe enviar (siempre por protocolo UART a una velocidad común entre ambos dispositivos) la siguiente cadena de caracteres "$$$". Después de que se envía, el módulo Wifly debe responder (se recibe) la secuencia "CMD" para confirmar que entramos en el modo comando. A partir de este momento se ingresa en el modo comando hasta que se le envié el comando "exit". Con el anterior comando volverá al modo datos.

Observando el diagrama se observa tener una comunicación bidireccional siguiendo la ruta "User Data – UART – WiflyModule – WIFI_Interface- Wi-Fi" ó en el orden inverso sin necesidad de pasar por el modo "comando". El modo comando solo es necesario para configurar los parámetros iniciales como SSID, tipo de seguridad de la red, dirección IP de destino, etc. Pero una vez conectado el modulo no se tendrá que volver a este modo porque ya en el modo datos de podrán enviar y recibir datos útiles por la red sin ningún inconveniente.

Analizando más a fondo el datasheet se encuentran de manera análoga las funciones que pueden ser ingresadas en modo comando (es el modo importante para la configuración). El conjunto de instrucciones GET (solo lectura) se utiliza para verificar el estado de parámetros configurados como modos de conexión, SSID seleccionada, contraseña, IP, etc. El conjunto de instrucciones SET

(escritura) por el contrario sirven para configurar ó cambiar los parámetros por los que el usuario desee.

Existen otro tipo de comandos muy importantes llamados “comandos de acción” los cuales prevalecen sobre los demás y se pueden usar en modo comando ó en modo datos dependiendo de la acción a ejecutar, entre estos comandos se encuentran “$$$” (sale del modo data y entra a modo comando”), “exit” (sale de modo comando y entra en modo data), “reboot” (resetea todo el wifly y deja los parámetros guardados), “reset_FACTORY” (coloca los valores de fabrica por default), “OPEN” y “CLOSE” (abre o cierran una conexión TCP respectivamente), etc.

Mediante tutoriales encontrados en la página Web del fabricante y el previo estudio de las funciones del modulo Wi-Fly RN131C con su hoja de datos, se procede a hacer una prueba de configuración y envió de datos mediante protocolo TCP. El modulo se configura con los parámetros básicos mínimos para obtener una conexión fiable a internet. Estos parámetros son:

- Tipo de antena (interna ó externa) en el caso particular del modulo utilizado se le adaptó una antena externa por lo tanto se configuro externa.

- Nombre de la red Wi-Fi

- Tipo de seguridad que tiene el router (WEP, WPA, WPA2, etc.)

- Contraseña (del router Wifi)

- Modo de conexión (conexión automática si la red se encuentra disponible)

- Numero de puerto a usar en la transmisión

- Dirección IP de destino

- Protocolo que se utiliza (TCP ó UDP)

Después de configurar estos parámetros básicos se ingresa el comando “save” para que guarde los anteriores datos en los registros de forma permanente. Luego se ingresa el comando “reboot” para resetear todo el sistema pero sin borrar los anteriores parámetros ya guardados. Después de el reboot el módulo queda inmediatamente en modo datos listo para comunicarse con un servidor TCP. Los datos enviados se muestran a continuación en color rojo. Lo que responde (se recibe) el módulo wifly se visualizá en color azul.

```
Send:(05:35:45) set wlan ext_antenna 1
Rec:(05:35:45)set wlan ext_antenna 1
AOK
<2.31>
Send:(05:49:38) scan 30
Rec:(05:49:38)scan 30
<2.31>
SCAN:Found 4
Num             SSID   Ch  RSSI   Sec    MAC AddressSuites
 1        FAMILIA ARIZA 01 -80 WPA2PSK 1c:7e:e5:a9:bc:ae AES/TKIPM-
TKIP   1104
 2              marktin 02 -26 WPA2PSK e2:9f:42:79:16:92 AESM-AES   1104
 3            ADRIANITA 11 -79   WPAv1 00:1a:2b:2b:6f:81 TKIPM-
TKIP   1104
 4      PRIVATE_NETWORK 11 -62 WPA2PSK 1c:7e:e5:a9:9b:6c AES/TKIPM-
TKIP   1104
Send:(06:04:33) set wlan ssid PRIVATE_NETWORK
Rec:(06:04:33)set wlan ssid PRIVATE_NETWORK
AOK
<2.31>
Send:(06:08:40) set wlan auth 4
Rec:(06:08:40)set wlan auth 4
AOK
<2.31>
Send:(06:09:45) set wlan phrase marktin8826
Rec:(06:09:45)set wlan phrase marktin8826
AOK
<2.31>
Send:(06:10:13) set wlan join 1
Rec:(06:10:13)set wlan join 1
AOK
<2.31>
Send:(06:10:51) set ip local 2101
Rec:(06:10:51)set ip local 2101
AOK
<2.31>
```

```
Send:(06:11:22) set ip protocol 2
Rec:(06:11:22)set ip protocol 2
AOK
<2.31>
Send:(06:20:58) set ip a 10.211.55.3
Rec:(06:20:58)set ip a 10.211.55.3
AOK
<2.31>
Send:(06:21:22) get ip
Rec:(06:21:22)get ip
IF=DOWN
DHCP=ON
IP=10.211.55.3:2000
NM=255.255.255.0
GW=0.0.0.0
HOST=0.0.0.0:2000
PROTO=TCP,
MTU=1524
FLAGS=0x7
TCPMODE=0x0
BACKUP=0.0.0.0
<2.31>
Send:(06:23:06) save
Rec:(06:23:06)save
Storing in config
<2.31>
Send:(06:33:49) reboot
Rec:(06:33:49)reboot
*Reboot*
Status:(06:33:49)Break detected. BreakState = False
Status:(06:33:49)Break detected. BreakState = False
WiFly Ver 2.31, 01-01-2012 on 131C83
MAC Addr=00:06:66:13:d5:75
Auto-Assoc PRIVATE_NETWORK chan=11 mode=WPA2 SCAN OK
Joining PRIVATE_NETWORK now..
*READY*
Associated!
DHCP: Start
DHCP in 1977ms, lease=86400s
IF=UP
DHCP=ON
IP=192.168.1.10:2101
NM=255.255.255.0
GW=192.168.1.1
Listen on 2101
Rec:(06:45:17)*OPEN*
Send:(06:45:47) martin
Rec:(06:46:03)jose
Send:(06:46:22) prueba de comunicacion
Rec:(06:46:29)recibido
Send:(06:50:24) set ftp address 208.109.14.133
Send:(06:51:20) close
Rec:(06:51:24)*CLOS*
```

Figura 24. Comandos necesarios para comunicación TCP en el módulo Wi-Fi [29]

El servidor de destino se configuró en el mismo computador mediante la aplicación COMM-OPERATOR y se recibieron los datos mediante protocolo TCP a través de la red de forma satisfactoria.

4.1.1.6 DISEÑO DE METODOLOGÍA PARA ACTUALIZAR EL FIRMWARE

Una de las tareas adicionales encomendadas fue realizar una metodología para actualizar el firmware de forma rápida para tener este documento en el departamento de Soporte Técnico y utilizarlo para actualizar los equipos que salgan al mercado. La metodología consiste en estandarizar los pasos de actualización del firmware en el módulo y así agregar funciones nuevas y mejoras técnicas, e incluso problemas internos que se detecten en un futuro por el fabricante. De forma general se buscan implementar mejoras en el módulo Wi-Fi mediante la actualización de su firmware de forma inalámbrica.

Para esto se investigó cómo realizar la actualización del firmware del módulo. Posterior a la investigación se realizó el procedimiento de actualización de forma real mediante comandos ingresados al modulo Wifly mediante protocolo UART utilizando la herramienta de software “Commoperator” y un conversor UART-USB que permitió la comunicación entre el modulo y el PC.

De esta forma se logró actualizar el firmware del dispositivo. Al tener el firmware actualizado se determino la secuencia mínima de comandos para actualizar el módulo Wi-Fi. De forma general el proceso es bastante sencillo. Como requerimiento previo se debe conectar el modulo Wifly a una red que tenga acceso a Internet, se configura el servidor donde se almacena el firmware (dirección del servidor ftp) y se ingresa el comando “ftp update, se espera a que cargue y automáticamente queda actualizado el firmware.

- Metodología para la actualización del firmware en la FCV

1. Se analiza y se documenta la forma correcta de conectar los pines de la tarjeta de prueba del modulo Wi.Fly-RN131C al módulo UART-USB.

2. Se realiza la actualización del firmware investigando en su hoja de datos cómo hacerlo. Se determina de forma rápida los secuencia mínima necesaria para lograrlo.

3. Se realizan las pruebas posteriores de verificación para comprobar que en realidad sea la última versión.

4. Se documenta el procedimiento teniendo en cuenta el nuevo servidor ftp para realizar la actualización.

5. El procedimiento paso a paso se dejará documentado en el informe y en un video.

Según lo anterior lo primero que se debe realizar es seleccionar la forma correcta de energizar la tarjeta Wifly.

Como paso inicial se asume que el módulo wifly está conectado a una red inalámbrica con conexión a internet (como está documentado en el registro de datos y observaciones). Para realizar el proceso se siguen los siguientes pasos:

1. Se verificará la versión actual del firmware (esto es opcional solo para verificación).

2. Se ingresará la dirección (esto es obligatorio debido a que el fabricante la cambio recientemente y es distinta a la que aparece en su hoja de datos) del servidor ftp.

3. Se enviará el comando “ftp update” (antes ingresando el comando “save” para guardar los cambios del servidor ftp) y esperar a que se descargue y se instale automáticamente la última versión del firmware .

4. Para comprobar los resultados se verificara la versión del firmware instalada. En modo comando se le enviara el comando “ver”.

El anterior procedimiento se documenta en base al algoritmo utilizado que se muestra en la Figura 25

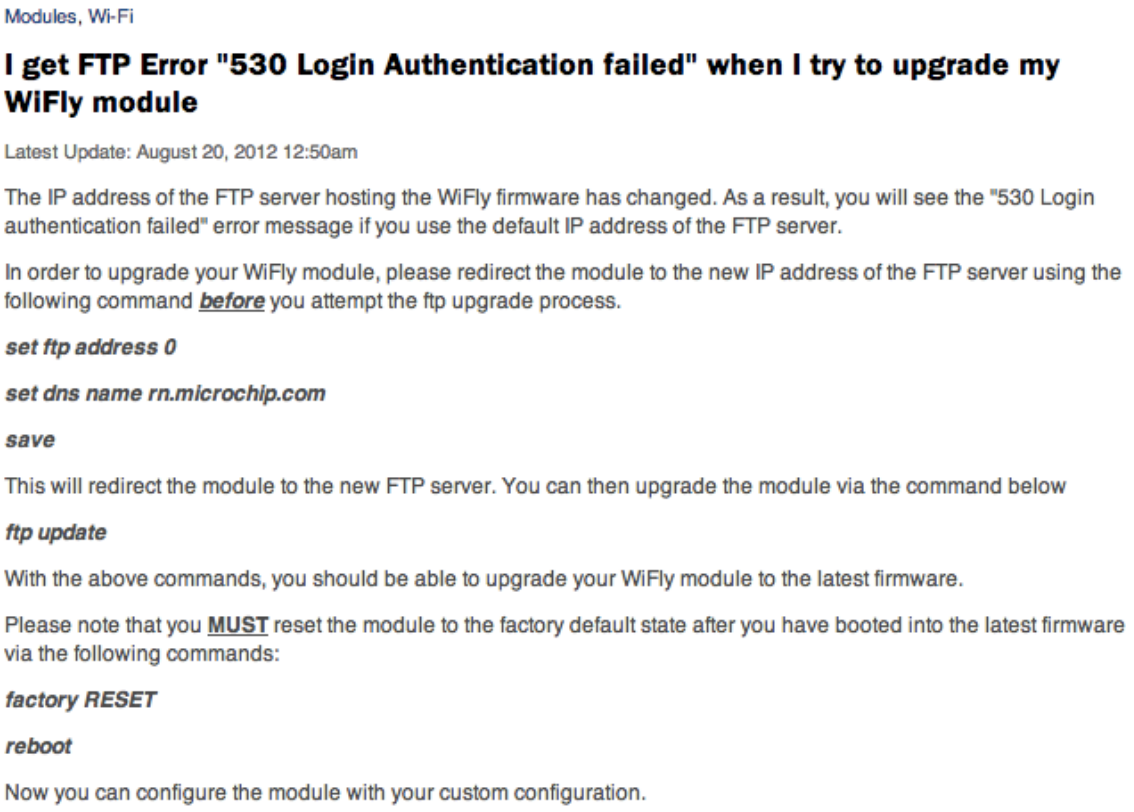
```
STEPS:

$$$                          (command mode)
set wlan ext_antenna 1 ()    (Determines which antenna is active, where <value> is 0 (use the chip antenna) or
                               1 (use the U.FL connector)).
scan 30                      (Performs an active probe scan of access points on all 13 channels. The default
                               is 200 ms/channel).
set wlan ssid PRIVATE_NETWORK (Sets the SSID with which the module associates).
set wlan auth 4              (Sets the FTP serverís IP address of the FTP server).
set wlan phrase marktin8826  (Sets the passphrase for WPA and WPA2 security modes).
set wlan join 1              (Sets the policy for automatically associating with network access points).
save                         (Saves the your configuration settings to a file).
reboot                       (Forces the module to reboot (similar to a power cycle)).


$$$                          (command mode)
ver                          (Displays the firmware version).
set ftp address 0            (Sets the FTP server ís IP address of the FTP server).
set dns name rn.microchip.com (Sets the name of the host for TCP/IP connections to <string>).
save                         (Saves the your configuration settings to a file).
ftp update                   (Deletes the backup image file, retrieves a new image file, and updates the boot
                              pointer to the new image).
factory RESET                (Loads the factory defaults into the moduleís RAM and writes the settings to the
                              standard configuration file. You must type the word RESET in capital letters).
reboot                       (Forces the module to reboot (similar to a power cycle)).


$$$                          (command mode)
ver                          (Displays the firmware version).
```

Figura 25. Secuencia de comandos mínima necesaria para la actualización del firmware.

En la hoja de datos se encuentra un error y en muchas páginas web debido a que no está actualizada la dirección del servidor FTP mediante el cual se hace la descarga para la actualización. Esta dirección es necesaria para la descarga del

firmware por lo cual sin esta dirección no es posible actualizar al último firmware. La dirección del servidor anterior viene precargada por lo cual si este servidor aun funcionara, después de conectarse a una red con internet el unico comando a enviar seria “ftp update”. Afortunadamente se encuentro en la pagina del fabricante una nota de aplicación donde menciona el cambio del servidor. En la figura 26 se explica claramente cómo hacer la actualización después de que el modulo esta correctamente conectado a alguna red con internet.

En un futuro se puede incluir este código se ejecución a manera de script en la driver creado para controlar el modulo Wifly con el procesador NIOSII y de esta forma actualizar el código de forma automática de forma inalámbrica desde el dispositivo médico en el momento que el usuario ó soporte técnico lo requiera.



Modules, Wi-Fi

I get FTP Error "530 Login Authentication failed" when I try to upgrade my WiFly module

Latest Update: August 20, 2012 12:50am

The IP address of the FTP server hosting the WiFly firmware has changed. As a result, you will see the "530 Login authentication failed" error message if you use the default IP address of the FTP server.

In order to upgrade your WiFly module, please redirect the module to the new IP address of the FTP server using the following command ***before*** you attempt the ftp upgrade process.

set ftp address 0

set dns name rn.microchip.com

save

This will redirect the module to the new FTP server. You can then upgrade the module via the command below

ftp update

With the above commands, you should be able to upgrade your WiFly module to the latest firmware.

Please note that you **MUST** reset the module to the factory default state after you have booted into the latest firmware via the following commands:

factory RESET

reboot

Now you can configure the module with your custom configuration.

Figura 26. Solución al error de conexión con servidor FTP dado en la hoja de datos. Se muestra la dirección del nuevo servidor[30].

Como tarea adicional (gracias a la investigación que se realizada) se hizo la retroalimentación con el Ing. José Pinilla acerca de cómo se debe energizar el módulo Wi-Fi dentro de la board principal del monitor del dispositivo médico la cual fue diseñada en la FCV. Debido a que el módulo Wi-Fi tiene un modo de bajo consumo se necesita observar los terminales que tiene conectados y así determinar la mejor forma de sacarlo de este modo cuando sea necesario. Cuando el módulo entra en modo sleep este no se vuelve a la normalidad mediante ningún comando por UART simplemente tiene otros terminales para tal fin. Investigando

detalladamente su hoja de datos se encontró que la única forma según las señales conectadas en la board para sacarlo del modo de *sleep* (bajo consumo) es mediante en pin Rx del UART el cual es configurado para salir del modo sleep. Se hizo la prueba de envió de comandos y recepción de datos mientras este pin estaba conectado al mismo tiempo y no se obtuvieron errores.

4.1.1.7 PRUEBA DEL MÓDULO UART FULL DUPLEX

Es de vital importancia el análisis, prueba e implementación del módulo UART para comunicar al NIOSII embebido en la FPGA con los módulos de comunicación. Este módulo diseñado en HDL se instancia dentro de la FPGA comunicando la CPU2 del sistema multicore de la FPGA con los pines físicos que se unirán con el modulo Wi-Fi y 3G dentro de la board del dispositivo medico. En la CPU2 tambien se encuentran capas de software ademas de los drivers conectados directamente al UART. Un esquema de las capas de software de todo el sistema integrado con los módulos UART para Wi-Fi y 3G/GPS se muestra en la siguiente figura.

Figura 27. Esquema general en capas de software de la CPU2 y sus módulos UART requeridos.

Como se puede observar es necesario un módulo de comunicación “UART WI-FI” y otro módulo “UART 3G/GPS” para comunicar el driver hecho en software (en el NIOS2) con los módulos reales Wi-Fi y 3G mediante protocolo UART en la board principal. Como los dos módulos UART mencionados tienen la misma función se puede hacer un solo diseño y simplemente instanciarlo 2 veces dentro de la FPGA. Los drivers para envió de comandos también deben tener un resultado final muy similar.

Para este diseño es necesario hacer la instanciación del módulo entregado en la FCV el cual es utilizado por el sistema actual. Se realizaron una serie de pruebas a velocidades de más de 9600 baudios, 14400 baudios, 19200 baudios, 28800

baudios y superiores. En estas pruebas se evidenciaron errores en la recepción de datos. Algunos datos no llegaban bien a velocidades superiores a 57600 baudios.

Debido a lo anterior se decidió emprender el diseño de un nuevo módulo en Hardware el cual sea más fiable en el envió y transmisión de datos a altas velocidades. Se tiene como objetivo transmitir a 460800 baudios que fue la velocidad máxima a la que se hicieron pruebas cuando se estuvo probando el dispositivo WiFly-RN131C con un PC convencional y un conversor UART-USB. Además se requiere que la transmisión de datos sea lo más rápida y eficiente posible al dibujar múltiples señales fisiológicas vía remota con la menor latencia posible.

4.1.1.8 DISEÑO DE UN NUEVO MÓDULO UART EN VERILOG (HDL)

Dado que se implementó un sistema completamente nuevo, es decir desde cero, para controlar y ejecutar las diversas funciones permitidas por el hardware “Wifly RN-131C”, en esta sección se explicara la creación del sistema en dos sub-descripciones básicas; la descripción hardware y la descripción software, en las que se explicará a su vez la finalidad y funcionalidad de cada módulo creado sea hardware o software.

DISEÑO DEL MÓDULO VERILOG PARA COMUNICACIÓN UART

Para el diseño de la descripción hardware se realizaron los siguientes pasos:

1. El primer paso fue crear el sistema hardware en el cual trabajará el software interactivo en cuestión, para ello se partió de generar una descripción de pines predeterminada en donde se tendrá como resultado un proyecto predefinido con la descripción completa de la tarjeta a utilizar (De0 Nano / Fpga), a través de la herramienta “System Builder” (Fig.1.1).

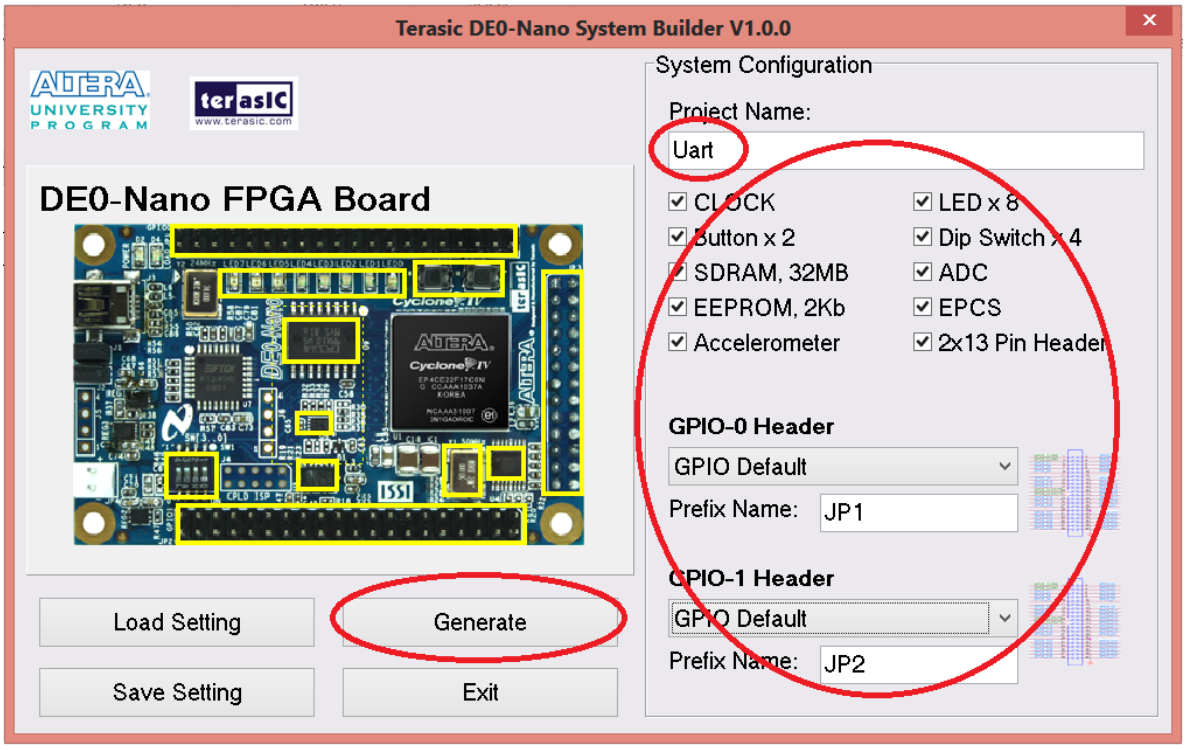


Figura 28. Generación de un proyecto prescrito con la ubicación de los pines usables del sistema hardware.

2. Se procedió a verificar que la descripción generada fuese correcta, de acuerdo a lo que se implementó al abrir el proyecto (uart.qpf) con la herramienta de altera “Quartus” y así para iniciar la creación del hardware a utilizar sobre esta plantilla.

3. Se modificaron aspectos básicos de funcionamiento para la respectiva tarjeta como la tensión de operación.

4. Se proceden a montar los distintos módulos hardware necesarios para programar un futuro software en el procesador de la tarjeta (Nios II) a través de la herramienta de altera “Qsys” (Fig.5); finalmente teniendo el hardware señalado y en posición, solo resta ir a la pestaña ‘Generation’ que se aprecia en esta misma figura, ejecutar la función de generación y esperar a que el programa cree los archivos necesarios que describan el hardware que se acaba de señalar en “Qsys” (nios_wifly.qsys).

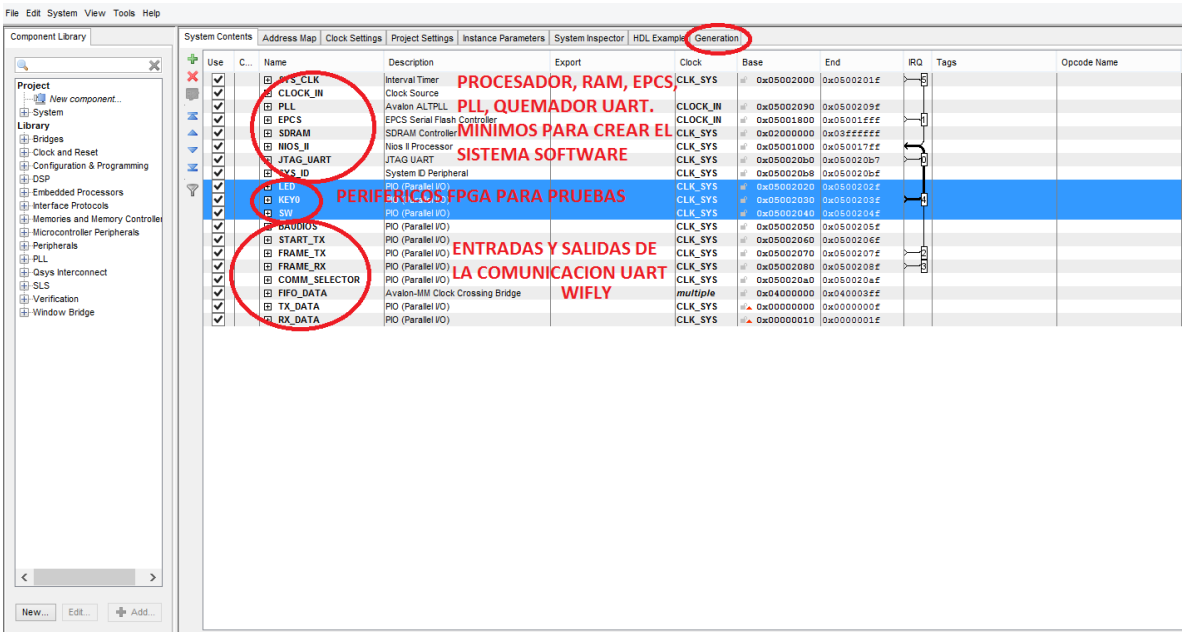


Figura 29. Generación del hardware básico en Qsys.

5. Hecho lo anterior se tendrán una serie de archivos en descripción de hardware de extensiones "*.qip", "*.v" entre otros generados en la carpeta de guardado del proyecto en cuestión. Este archivo principal (nios_wifly.qip) es el que se agregará al proyecto principal "uart", logrando así la vinculación de todas las descripciones señaladas en "Qsys", dentro de un solo módulo hardware (llamado "nios_wifly.v" para esta situación) que se instanciará en la descripción hardware "uart.v" del proyecto principal (uart.qpf), como un módulo hardware más que se adicionara al sistema hardware global.

6. Ya para esta instancia se tiene el enlace entre el hardware (Módulo de comunicación local tipo uart), y el software (Programación en código C descargada en el procesador "Nios II") del sistema; ahora solo resta agregar ese módulo hardware que funcionará como traductor hardware (Comunicación UART) de palabras en un nivel binario, que el software (Programación creada para el procesador) pueda entender en un nivel más alto de comandos (Librerías del sistema). Ya entendiendo globalmente como funciona un sistema (Hardware - Software), se sobreentiende que el siguiente paso es agregar ese hardware traductor como un chip más al hardware del sistema.

7. En esta etapa solo resta hacer las conexiones necesarias y pertinentes entre el módulos hardware "UART_FullDuplex.v" y "nios_wifly.v" a través de código Verilog como se muestra en la y así enlazar los módulos a través de cables que se conectan entre chips. En la figura 30 se muestra el circuito final obtenido.

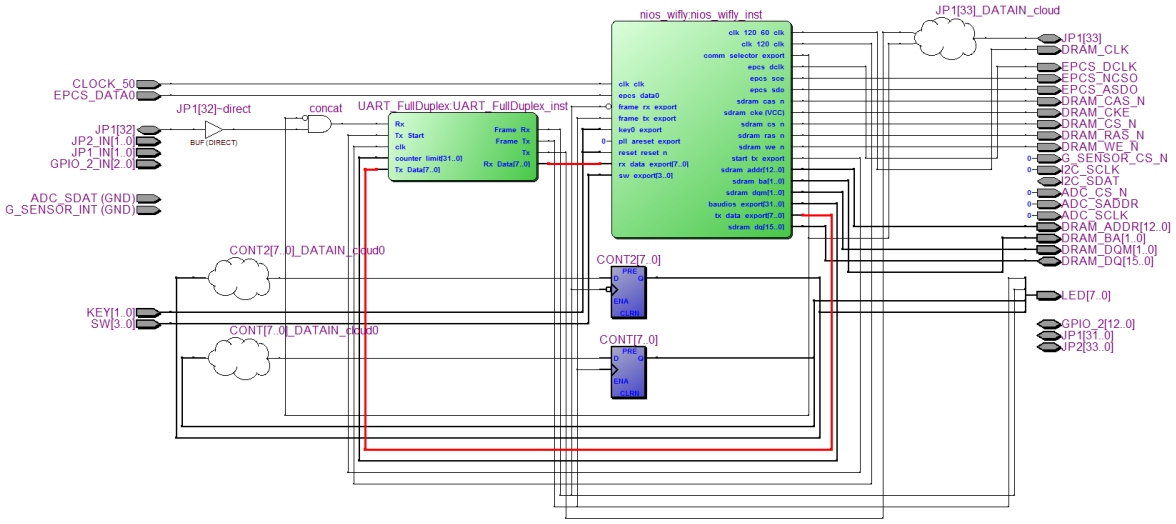


Figura 30. Circuito final.(Conexiones de envió y recepción de caracteres en rojo).

En este punto ya se tienen todas las conexiones hardware necesarias para que exista comunicación entre los módulos uart del dispositivo Wifly y el dispositivo para el cual se está diseñando este módulo el cual ya podrá reconocer el procesador embebido dentro del chip “nios_wifly.v”, así que solo resta generar (compilar y programar) el software necesario para empezar a enviar y recibir comandos completos locales desde y hacia el dispositivo de red “Wifly GSX” para finalmente establecer una conexión a internet utilizando las funciones de este último como sea posible y necesario.

4.1.1.9 INTEGRACIÓN DEL MÓDULO UART CON EL PROCESADOR NIOSII

1. Se utilizó la herramienta “Eclipse” de la compañía altera para desarrollar e integrar el módulo UART creado al procesador “Nios II”, en esta herramienta se crea la aplicación y además una carpeta “BSP” con todos los archivos necesarios que respaldan el manejo del hardware descrito previamente para el procesador en la herramienta “Qsys”. La herramienta “Eclipse” solicita un archivo que contenga la información del hardware previamente descrito. Esta información la contiene el archivo “nios_wifly.sopc” ubicado en la carpeta donde se encuentra toda la descripción de hardware.

2. Una vez creada la aplicación del proyecto software llamada para este caso “wifly”, se encontrará una carpeta con la extensión “BSP” en su nombre la cual contiene todos esos archivos que permitirán la interacción del hardware con el software, además de otra carpeta con el nombre “wifly” únicamente que tendrá

esos archivos que se agregarán al proyecto software y que finalmente serán los encargados de programar el procesador con una serie de tareas; nótese que existe un archivo principal “main.c” que contiene la función llamada “main” la cual el procesador ejecutará antes de cualquier otra función, ya que es un nombre reservado para programación en Ansi-C y solo imprimirá un mensaje con los caracteres “Hello from Nios II!” por el momento.

3. Así como en la plantilla hardware se trabajó con el módulo “UART_FullDuplex.v”, en la plantilla software también se trabajó la función “int main()” en conjunto con otras funciones desarrolladas para escribir y leer sobre el dispositivo “Wifly GSX” gracias a la labor del hardware “UART_FullDuplex.v”, todo con la finalidad de generar las librerías básicas de comunicación “uart.c” y “uart.h” además de una librería “includes.h” que contendrá en forma más ordenada y en un solo archivo, la vinculación de más librerías para que puedan ser reconocidas todas las funciones del sistema.

4. Finalmente las librerías se prueban y desarrollan al establecer una serie de envíos y respuestas al sistema hardware gracias a la función del procesador implementado; una vez estas pruebas están listas y se establece la correcta comunicación y control del dispositivo “Wifly”, ya solo es cuestión de proceder con las pruebas de comunicación por protocolo UART directamente desde la FPGA hasta el dispositivo Wifly.

La metodología a seguir para programar el procesador y controlar de manera más activa el hardware creado con anterioridad es siempre la misma. Escribir código en C, compilarlo, verificar errores y descargar el código correctamente compilado a la tarjeta (específicamente al procesador Nios II) a través de la herramienta “Comand Shell” de altera; lo anterior puede verificarse en la Fig.13, sin embargo a partir de ahora se utilizará el script llamado “j.sh”, que contiene la información mostrada en la Fig.13 para generar el proceso de compilación, descarga y visualización en el terminal, pero evitando la molesta tarea de escribirlos repetidamente, ya que al ejecutar el script “j.sh” a manera de línea de código “bash j.sh” se estarán ejecutando secuencialmente los comandos “make all”, “nios2-download” y “nios2-terminal”; igualmente esta ejecución debe ser realizada estando en la ruta de ubicación correcta para generar el archivo de extensión (*.elf) que finalmente se descargara como software en el procesador “Nios II” de la tarjeta de desarrollo.

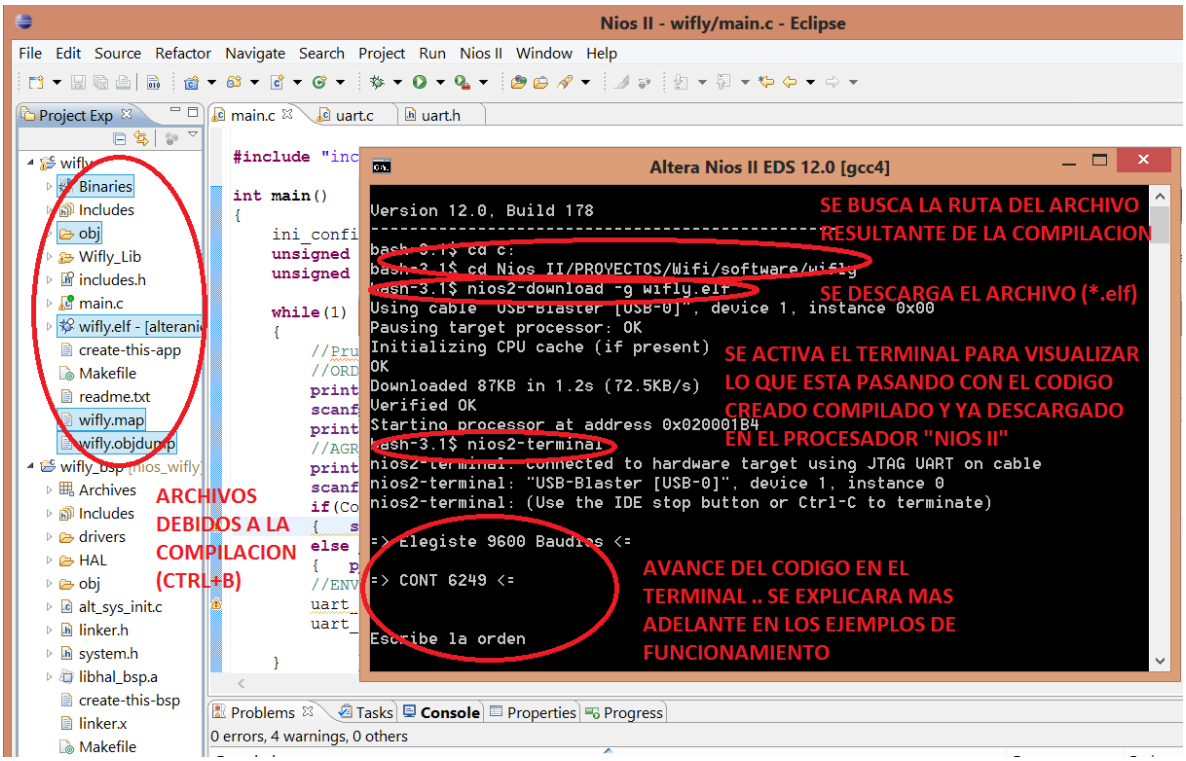


Figura 31. Secuencia de compilación y descarga del software creado al procesador “Nios II”

PRUEBAS DE ENVIÓ Y RECEPCIÓN DE DATOS

Para las pruebas de envío y recepción, se implementará un hardware externo funcional conocido como “Conversor USB-Serial” con el fin de depurar errores mínimos que se cometen en el proceso, al confiar plenamente en la funcionalidad de este hardware externo. Como se mostrará en los resultados, este hardware externo permitirá la traducción de todo lo que se reciba o transmita del mismo en forma serial (Uart) a USB, de manera que el computador a partir de la aplicación “Comm Operator” pueda mostrar lo que se trata de enviar o en el caso contrario, lo que se supone se deba recibir en la tarjeta de desarrollo DE0-Nano.

PRUEBA DE ENVÍO

Para esta prueba se implementará el código mostrado en la primera figura del registro de datos y observaciones en donde se aprecia el código que utilizan las librerías desarrolladas para este hardware y que finalmente envía el dato nombrado como “TEST”. Posterior a esto duerme cierto tiempo para no generar conflicto con la función de impresión “printf” y luego duerme nuevamente para alcanzar a frenar el programa en algún momento deseado y mostrar los resultados; en la gráfica también se muestran los datos que llegan por USB y se observaran en “Comm Operator – Trial Versión”.

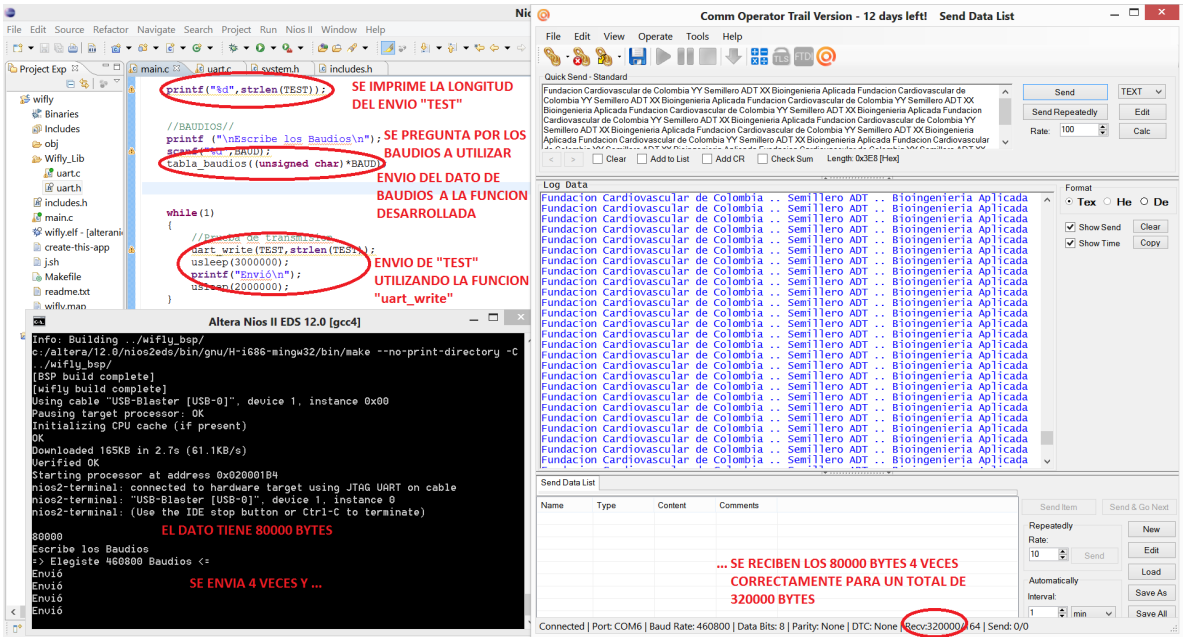


Figura 32. Envío de datos a utilizando el software “Comm Operator” en su versión Trial para visualización.

PRUEBA DE RECEPCIÓN

En esta prueba se enviaran los datos a través del puerto USB del computador y el “Conversor USB Serial” para ser recepcionados por el hardware y software desarrollados en la tarjeta “Fpga”; en esta prueba se implementara la función “uart_printf” de la librería “uart.c” para que cada cierto tiempo predefinido, revise e imprima lo que encuentra en el buffer de recepción del sistema como se puede ver en figuras de los resultados.

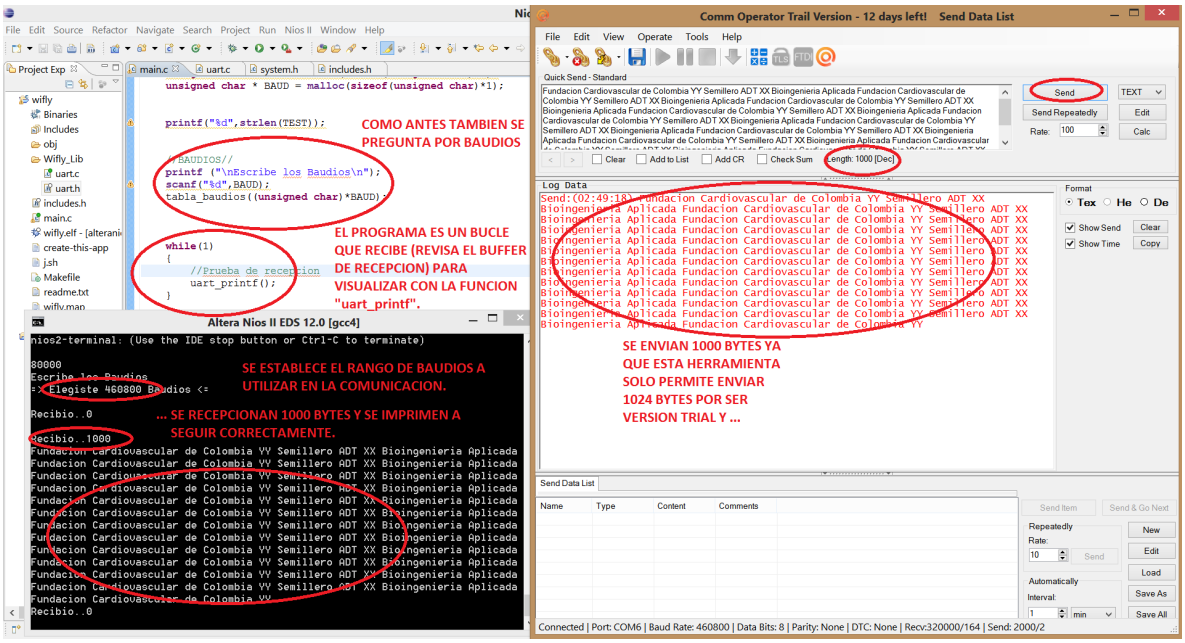


Figura 33. Recepción de datos. Envió del computador “Comm Operator”, hacia la tarjeta de desarrollo.

PRUEBA DE CORTO (TRANSMISIÓN Y RECEPCIÓN PARALELA)

En esta prueba no se utiliza ningún hardware externo, es solo el hecho de juntar los pines de transmisión y recepción “Uart” (Tx y Rx) que utiliza el modulo “UART_FullDuplex.v” desarrollado en la tarjeta “FPGA” físicamente con un cable, con el fin de lo que la misma tarjeta envié y pueda recibir inmediatamente, probando así el paralelismo que se tiene.

GENERACIÓN ARCHIVO LOG PARA ANÁLISIS DE DATOS

Se guardará la información del terminal del procesador NIOS en un archivo llamado “log.log”, este archivo tiene la tarea guardar para posterior verificación de los datos enviados y recibidos y así ser más precisos a la hora de recibir datos dado que las opciones de impresión del computador pueden fallar en cierta medida y no ser tan correctas dado la cantidad de datos que se están procesando.

El procedimiento en el terminal es el mismo, se ejecutara el script “j.sh” se eligirán los baudios y se dejará que el programa corra normalmente, cuando ya se haga la recepción de los datos, se detiene el programa gracias al tiempo de espera de tres segundos que se deja en el código “usleep(3000000)” (Fig.16); ahora solo resta revisar el archivo “log.log” para ser más precisos y saber qué es lo que realmente fue recepcionado de los ocho-mil datos que se auto-enviaron en esta prueba, este análisis se muestra en el registro de datos y observaciones en donde se cuentan las palabras “ADT” que indican una línea de 80 letras o bytes para este caso, para la cual se encontraron 1000 palabras (a través del software gratuito “Notepad ++”) que corresponderían a 1000 lineas de 80 palabras con un significado de 80000 datos auto-recepcionados correctamente.

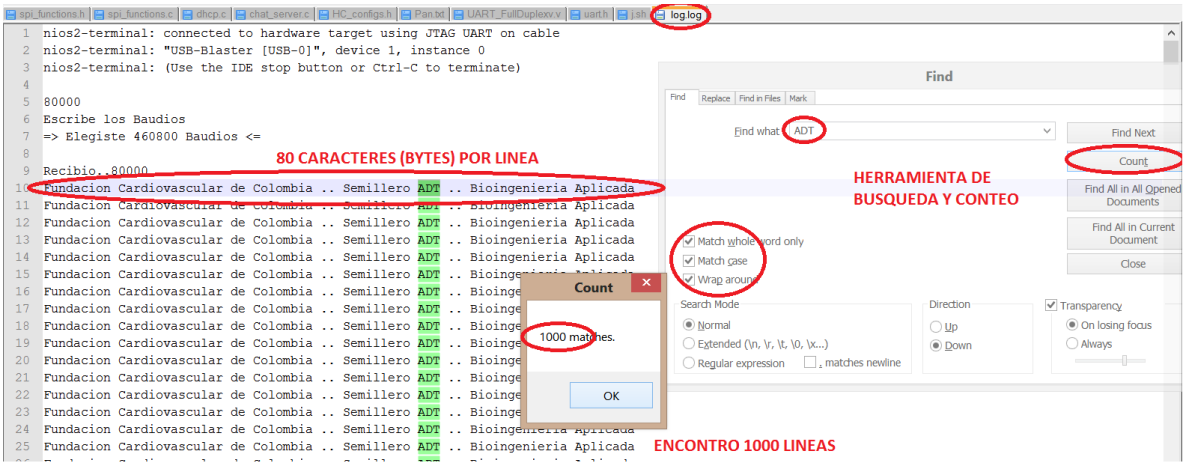


Figura 34. Confirmación más precisa de lo recepcionado a través de un archivo generado.

4.1.1.10 PRUEBA DE COMUNICACIÓN CON EL DISPOSITIVO WI-FI

En esta prueba se realizará la comunicación final ya con el dispositivo “Wifly GSX” una vez comprobadas las funciones de recepción, transmisión y paralelismo realizadas correctamente; para esta prueba ya no existe nada más que la tarjeta “FPGA De0-Nano” y el dispositivo “Wifly GSX” conectado a través de cables físicos (Tx,Rx,Gnd) en donde se envían una serie de comandos estudiados gracias al datasheet y se recibe una respuesta para lo enviado, esta prueba se observa claramente la imagen correspondiente en la descripción de los resultados en donde básicamente adquiere un dato del usuario para establecer el rango de baudios que utilizara la tarjeta “De0-Nano”, luego otro dato será la orden a enviar al dispositivo y una orden final que definirá si el dato que se va a enviar se enviará en modo comando o en modo dato (con un agregado o no de un par de bytes al envió total), una vez definido todo lo anterior se puede enviar normalmente el dato y esperar una respuesta del dispositivo que es finalmente impresa en el terminal de Nios (Ventana de color negro) como se aprecia igualmente en un par de ejemplos.

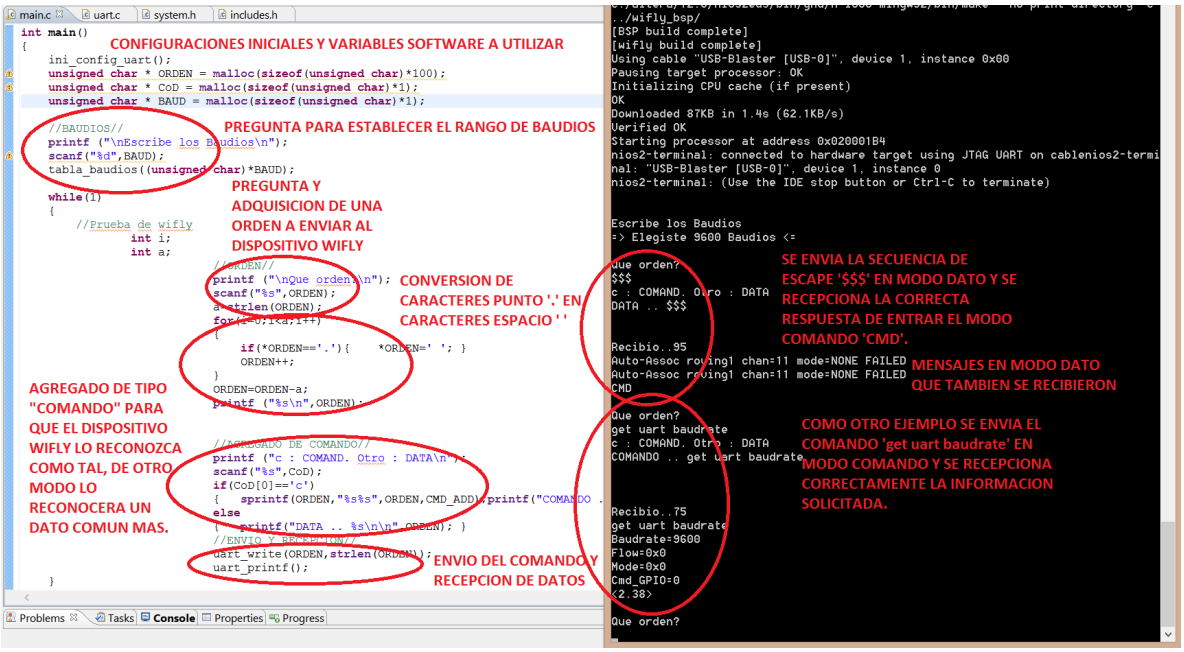


Figura 35. Ejemplo de comunicación entre el dispositivo y la FPGA (Envíos y Recepciones).

Esto fue lo que se desarrolló a través de esta metodología y resume la creación de un sistema hardware embebido para el control de la comunicación entre un usuario y el hardware externo “Wifly_GSX”. Lo anterior permitirá manejar dicho dispositivo a gusto en un nivel bajo de comandos simples a reconocer, esto gracias a una última función diseñada para reconocer comandos entrantes completos almacenados en el “FIFO” de recepción y retornar un valor entero que será la pista para reconocer más rápidamente las respuestas que el dispositivo “Wifly_GSX” otorgue y así tomar decisiones más rápidamente; esta función es un arreglo software sencillo que compara dos direcciones de memoria que contienen la constante que se espera y el dato recibido para así retornar una respuesta que se mencionó, esta función puede verse en la figura respectiva del registro de datos y no se mostró su correcto funcionamiento en este informe, ya que es un plus al sistema que no suma ni resta funcionamiento a la comunicación entre los dispositivos sino que únicamente permite un tratado más rápido de los datos que se van recibiendo en el “FIFO” de recepción indicado.

ANÁLISIS DE LAS PRUEBAS DEL MÓDULO UART

De acuerdo al procedimiento que se siguió en la descripción de cambios realizados, en donde se desarrollo la metodología para la creación del hardware y el software necesarios para la exitosa transmisión y recepción de datos mediante protocolo UART, se obtuvieron los resultados esperados y se comprobó la gran eficacia y adecuado funcionamiento del módulo diseñado al lograr transmisiones a altas velocidades.

Se cumplió satisfactoriamente el requerimiento con algunas ventajas adicionales. Por ejemplo se pudo transmitir a una velocidad máxima de 460800 baudios desde y hacia los dispositivos de comunicación de red externos.

4.1.1.11 DISEÑO DEL DRIVER PARA EL MÓDULO WI-FI EN LENGUAJE C

Para el diseño del driver, inicialmente deberán tenerse en cuenta los pasos mínimos necesarios para conectar el módulo a un router ó AP (acces point). Como ya se observó en la metodología para actualizar el firmware se deben configurar inicialmente el nombre de la SIID (nombre de la red inalámbrica creada por el AP) y su contraseña. Sin el SSID y la contraseña será imposible conectarse al AP por lo tanto estos son los parámetros mínimos de conexión. Adicionalmente se buscaron ejemplos de drivers para este módulo en internet para así tener una idea general del algoritmo que se debe desarrollar. Se encontraron librerías para módulos comerciales como *Arduino* y *mbed* y se tomaron ideas generales para implementarse en el driver a diseñar. Se tomó como guía inicial un documento a modo de nota de aplicación llamado “Training with Wifly-GSX” el cual sirvió de guía para la configuración básica de los diferentes modos de configuración y envió de datos.

Se realizó un estudio detallado a la hoja de datos específicamente al set de comandos de configuración del módulo Wi-Fi y según los requerimientos de conexión se decide hacer un resumen para su posterior implementación de los principales comandos a usar. Estos comandos se pueden agrupar según el datasheet como action commands (comandos de acción inmediata) y comandos de configuración de los cuales solamente se usarán los comandos de configuración para transmisión de datos por TCP, UDP, puertos e IP de destino así como los de configuración de los parámetros WLAN.

Como el driver apunta a un requerimiento general de alto nivel donde sus funciones serán usadas en el sistema final que incluye sistema operativo y diferentes llamados entre funciones de mas librerías, se decide trabajar con una estructura ya creada llamada “network”. Esta estructura contaba con variables creadas para el módulo Ethernet pero ahora se implementarán las que sean necesarias para los drivers de los módulos Wi-Fi y 3G. Estas variables dentro de la estructura network traen la ventaja de poder ser accedidas desde cualquier lugar dentro del sistema. Por ejemplo acá se puede guardar la IP asignada, el nombre de la SSID, contraseña ,etc. así como demás variables del módulo de transmisión de datos por red celular.

4.1.1.12 ENVÍO Y RECEPCIÓN DE MENSAJES DESDE EL NIOS II AL MÓDULO WI-FI

- ENVÍO

Para el envió de mensajes no hace falta ningún algoritmo distinto al ya usado en las pruebas del modulo UART. El algoritmo usado para el envió se muestra en la siguiente figura.

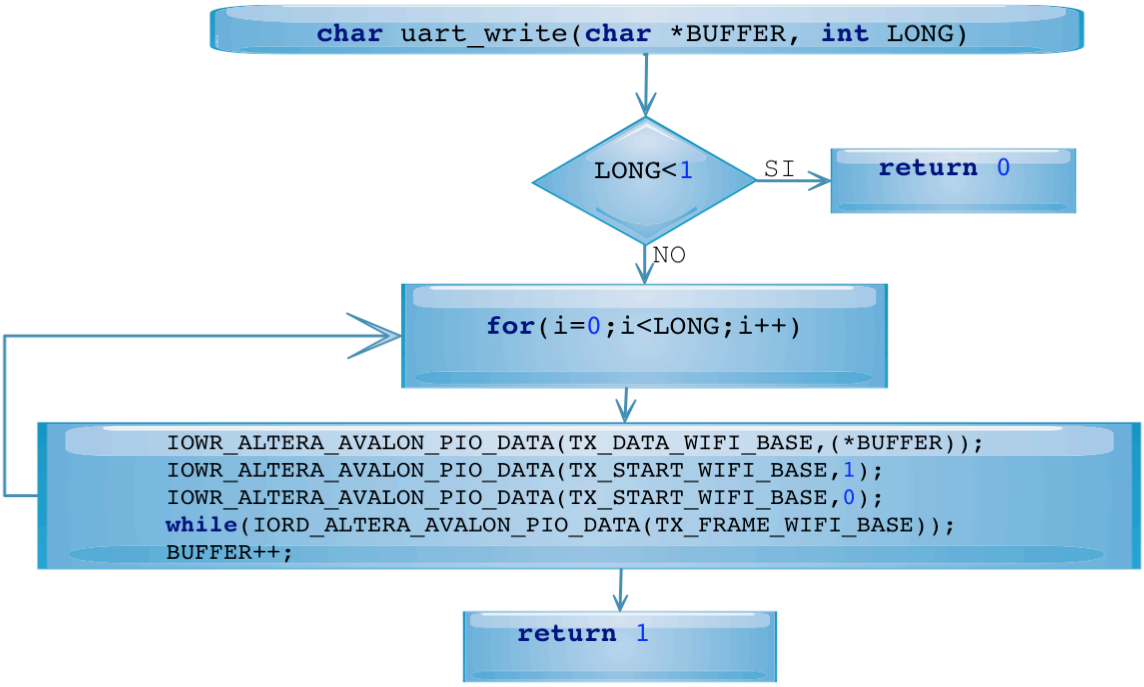


Figura 36. Diagrama de flujo para la función de envió de datos por UART

En el algoritmo se observa un ciclo iterativo *for* para el cual se tiene limitado su número de iteraciones con el número de bytes a enviar. La variable buffer aumenta en cada iteración para enviar el siguiente byte. En el while se observa la señal TX_WIFI_BASE del modulo UART que cambia su estado cuando se termina el envió de todos los bytes. De esta forma cuando el ciclo for llega a su número máximo de iteraciones la función devuelve 1 indicando que el string fue enviado satisfactoriamente.

- RECEPCIÓN

Antes de la versión final de la rutina de recepción de mensajes se tuvieron problemas porque a teniendo en cuenta que el driver funcionaba de forma correcta (recibia datos por UART, abría conexiones TCP y UDP y transmitía datos) se estaba probando el código sin el sistema completo, es decir, no había sistema operativo ni interfaz gráfica ni adquisición de datos de monitorización vital en el momento que se realizaron las pruebas. Por lo anterior y dada la experiencia que se tiene con el sistema junto al Ing. José Pinilla, se predice que la latencia aumentara cuando se ejecuten varios hilos de procesamiento al mismo tiempo por lo que un driver que en su recepción funcione a partir de tiempos (con la función *usleep* que “bloquea” el procesador por un tiempo dado en microsegundos) no es confiable y se debería replantear el código para la recepción de los mensajes por UART.

Para validar la hipótesis anterior se hizo una depuración del código de recepción probando cada una de las subrutinas del driver y se encontró un problema con la

rutina de “Reboot” debido a que cuando esta se ejecuta, el hardware Wifly por su pin de Tx envía ruido (mientras hace el reboot) el cual crea interrupciones en el UART de recepción de la FPGA y consecuentemente hace que lleguen bytes no deseados a la FPGA. Lo anterior en el software llena el FIFO de recepción haciendo que el ruido se sobrescriba en los datos útiles interrumpiendo el flujo del programa. La rutina “reboot” es necesaria para que el módulo guarde de forma permanente los comandos de configuración que se le han ingresado al módulo.

De forma detallada el problema se presenta en el momento en que se envía el comando “reboot” al módulo Wi-Fi, si este lo recibe de forma satisfactoria responde con el string “*Reboot*”, (el cual se debe capturar al inicio del FIFO de recepción y se utiliza en el driver para validar la recepción del comando y así dar continuidad al flujo del programa). Inmediatamente después de la anterior respuesta, comienza el reinicio general del modulo generando ruido por su terminales (en especial por el pin de Tx UART del modulo Wi-Fi) y hasta que no termina el “reboot” y el modulo no vuelve quedar en su estado funcional.

El anterior ruido genera interrupciones indeseadas en el módulo UART de recepción de la FPGA (realiza una interrupción cada que llega un byte de forma asíncrona mediante una interrupción por flanco de bajada), las cuales se activan debido al ruido que se presenta cuando se hace reboot, emulando la recepción de bytes que realmente no llegaron ni son información útil, esto hace que en el software se desborde el FIFO de recepción de 1Kb de tamaño y reescribe con ruido las primeras posiciones de memoria donde realmente se almacenó la información útil. En el caso particular estaba sobrescribiendo estructuras de datos en el sistema.

Para corregir este defecto de la función reboot se analizó y se decidió cambiar totalmente la función “rx2string” que adquiría los datos por tiempos como se muestra su código en la figura 37. En este código se realizaba un flush (reinicio de la cabeza y la cola en el buffer de recepción), al inicio y se le daba tiempo para que recibiera los datos mediante la variable de entrada “time2rec” (dada en microsegundos) dependiendo de la longitud de los datos a recibir y el tiempo de respuesta en el módulo wifly.

```
void rxfifo2str ( unsigned char* pbuffer, int time2rec)//(donde guardarlo
{

printf("\n Entra a ""Funcion rxfifo2str""..\n");
 rx_tail=rx_head;//like flush - Limpia el buffer de entrada para recibir
 usleep(time2rec);
//Tiempo para recibir los datos
    int buffsize=rx_tam();//Adquiere el tamaï¿½o de los datos que acaban
    printf(" \r\n Tamano a guardar en la recepcion: %d \r\n ",buffsize);
    memset(pbuffer, '\0', buffsize+1); //clarea todo el buffer de entrada
    memcpy(pbuffer,RX_FIFO+rx_tail,buffsize);//copie desde "RX_FIFO" en l
    printf("string guardado%s\n\n",pbuffer);
    rx_tail=rx_head;//like flush
}
```

Figura 37. Algoritmo función “rx2string” antigua

Debido a lo anterior se diseña un nuevo algoritmo que detecta la inactividad en la recepción después de recibir un paquete de bytes como se observa el código de la figura 2 y los respectivos diagramas de flujo de sus subrutinas en las figuras 2 y 3. Con este algoritmo basado en tiempos de latencia después de la recepción de un byte por UART se corrige la recepción de ruido en el módulo.

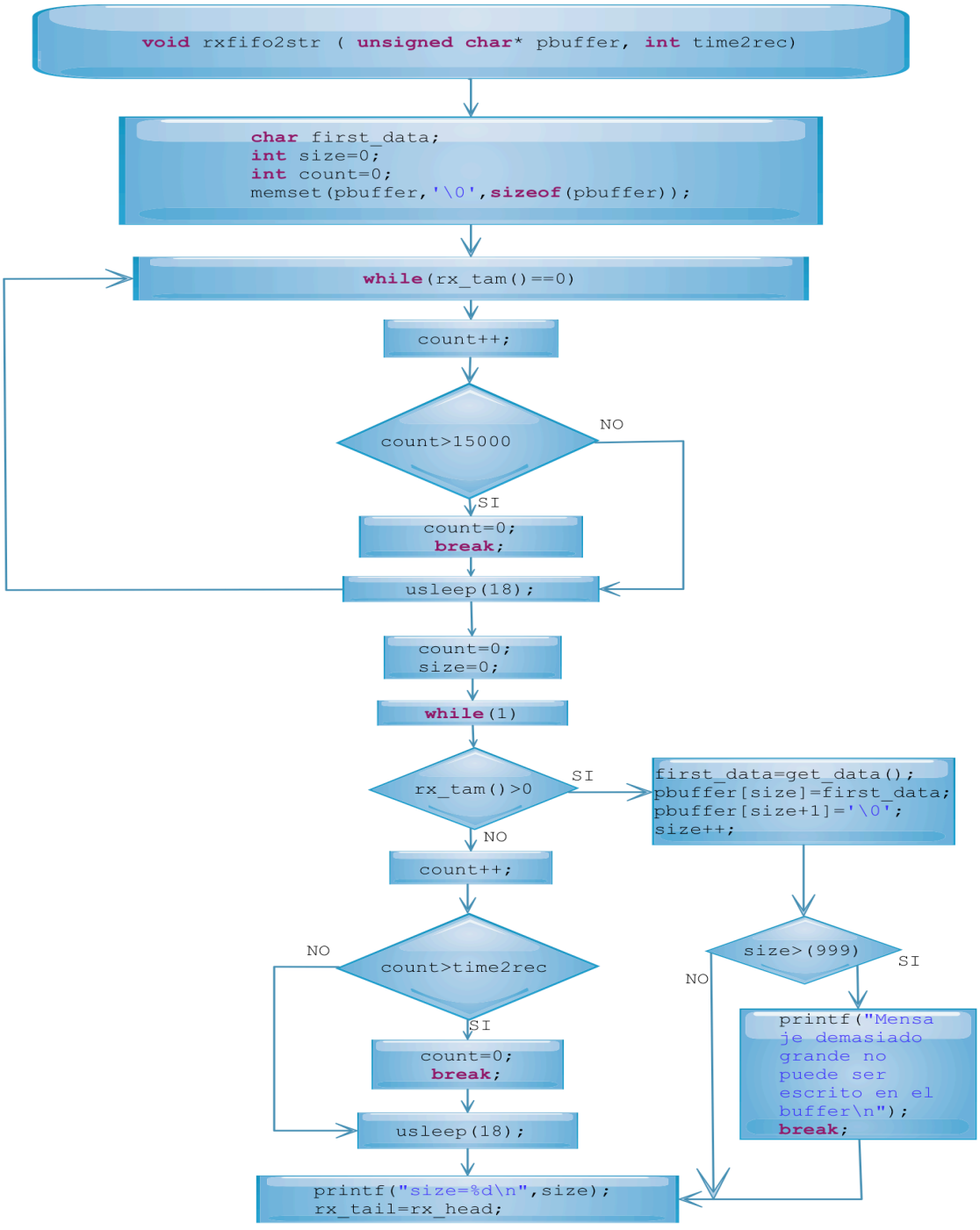


Figura 38. Diagrama de flujo Función “rxfifo2string”

La función inicia limpiando el buffer de recepción (ingresado como parámetro) con ceros para evitar errores posteriores en la adquisición del string final de recepción. Luego en el primer while se quedará esperando un tiempo prudencial a que llegue el primer byte en la recepción, aunque inmediatamente este byte sea detectado ó

este tiempo se supere saldrá del while directamente al while(1) donde se reciben los datos.

Dentro del while de recepción existe la función get_data() que es la encargada de ir creando el string de recepción consecutivamente cada que se reciba un byte. Este algoritmo se puede ver en la siguiente figura.

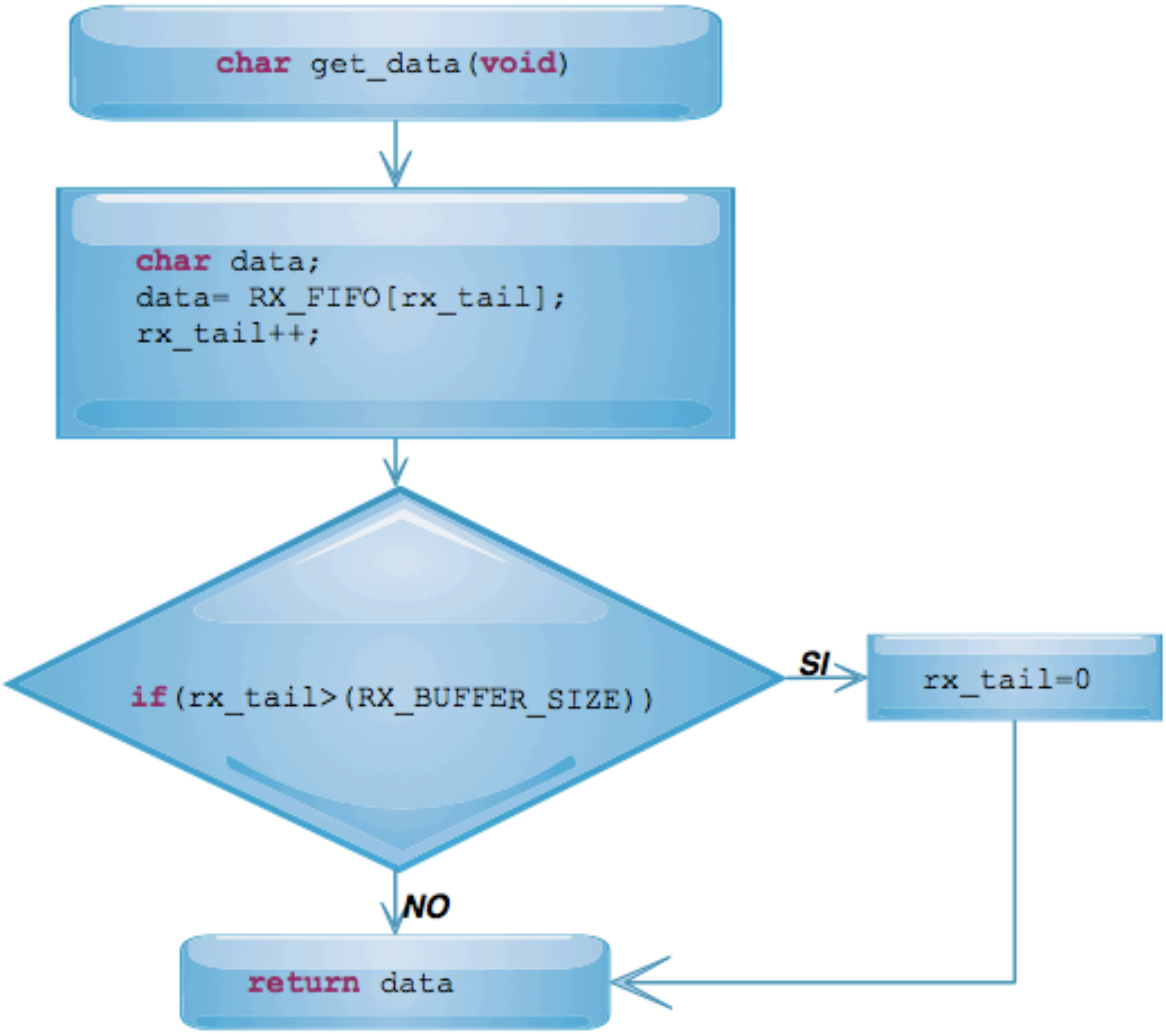


Figura39. Diagrama de flujo Función “Get Data”

La función *get_data()* toma el byte en el cual va la cola del FIFO de recepción y aumenta su índice para en que en el próximo llamado tome el siguiente byte. El índice de la cola va persiguiendo el de la cabeza hasta que se hagan iguales. La cabeza aumenta su índice mientras se generen interrupciones en el módulo UART de recepción es decir cuando llega un nuevo byte. Si el índice de la cola llega al tamaño máximo del buffer se reinicia en cero y se sobrescribirían los datos, situación que normalmente no debe suceder porque no se llegará a un tamaño mayor al string mas largo que el modulo pueda responder. Como este buffer es ingresado como parámetro se puede tener de un tamaño variable depende del

módulo a controlar (el módulo de transmisión de datos por red celular puede tener otro y funcionara bien dado que el buffer se ingresa como parámetro).

Siguiendo con el flujo del programa en el while(1) se quedará hasta que se reciban más de 1000 datos de entrada que coinciden con el tamaño máximo del buffer (en el caso particular del módulo Wi-Fi) o hasta que se cumpla un tiempo máximo dado por la variable “time2rec” multiplicado por 18uS (microsegundos). 18uS es el mínimo que se demora en recibir un byte a una velocidad de 460800 baudios que fue la utilizada para recibir datos como se muestra en la figura 40. Así la variable “time2rec” da tiempo a la función dependiendo de la longitud del mensaje a recibir según las pruebas preliminares.

$$18uS \approx \frac{1}{460800} \times 8$$

4.1.1.13 FUNCIÓN DE ENVIO Y CONFIRMACIÓN DE COMANDOS EN EL MÓDULO WI-FI

Como se observó en la prueba del módulo UART al módulo Wi-Fi se le envían comandos y el módulo responde con “AOK” si recibió el comando satisfactoriamente ó con “Error” si se recibió erróneamente ó con algún dato corrupto. Se creó una función que hiciera el proceso de envió, recepción y confirmación del mensaje después de enviar el comando. Esta función esta parametrizada por tiempo de respuesta (lo que tarda el módulo Wi-Fi en responder), mensaje enviado y mensaje esperado para confirmar que llegó adecuadamente. Esta función no solamente funciona con comandos SET, también funciona por ejemplo cuando se desea detectar la conexión al AP. Esto es posible porque todo comando cuenta con una respuesta conocida tanto si la acción es satisfactoria o fallida. Así se le puede dar control al flujo de configuración, conexión y envió de datos del modulo Wi-Fi en el driver. El diagrama de flujo se muestra en la figura 40.

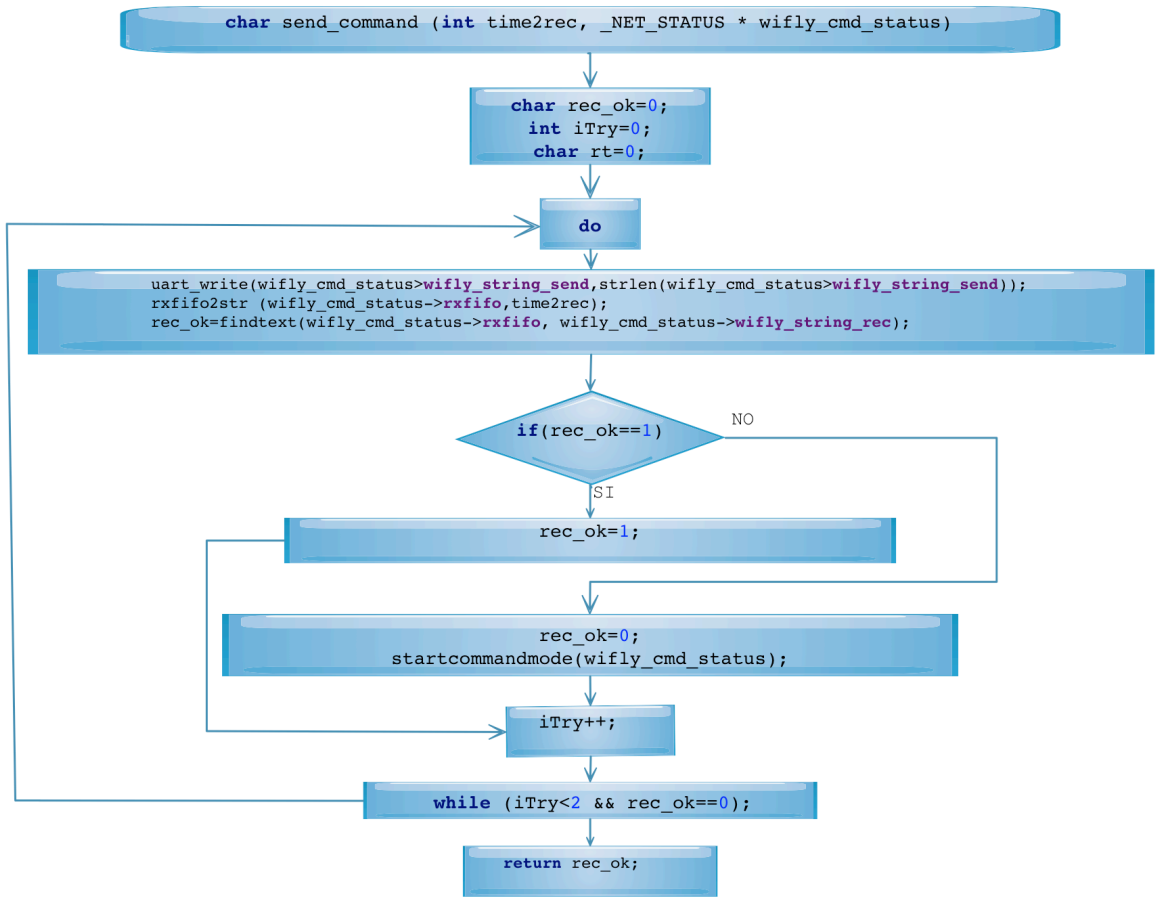


Figura40. Diagrama de flujo función para envío y recepción de comandos

Como se puede observar se utilizan las funciones ya explicadas de envío y recepción de mensajes por UART. El algoritmo comienza con la declaración e inicialización de sus variables globales, seguido a esto se ejecuta un ciclo do-while en el cual principalmente se ejecutan las 3 acciones ya mencionadas: enviar comando, guardar respuesta y comparar con la respuesta esperada. Si se detectó la respuesta al mensaje satisfactoriamente sale de la función retornando un uno. Si la respuesta al comando no fue la esperarada el ciclo do-while vuelve a ejecutarse nuevamente (en caso de que el comando enviado hubiera llegado con bytes corruptos al módulo Wi-Fi). El número de veces que se intenta enviar el comando se podría cambiar con la condición dada en el while por la variable *iTry*.

La función *findtext* fue la utilizada para encontrar la respuesta deseada después de enviar el comando dentro del string que se recibió por UART. Esta función fue una implementación mediante una librería de C++ externa, la cual simplemente se utilizó. El algoritmo de esta función se puede observar en el siguiente diagrama de flujo de la Figura 41.

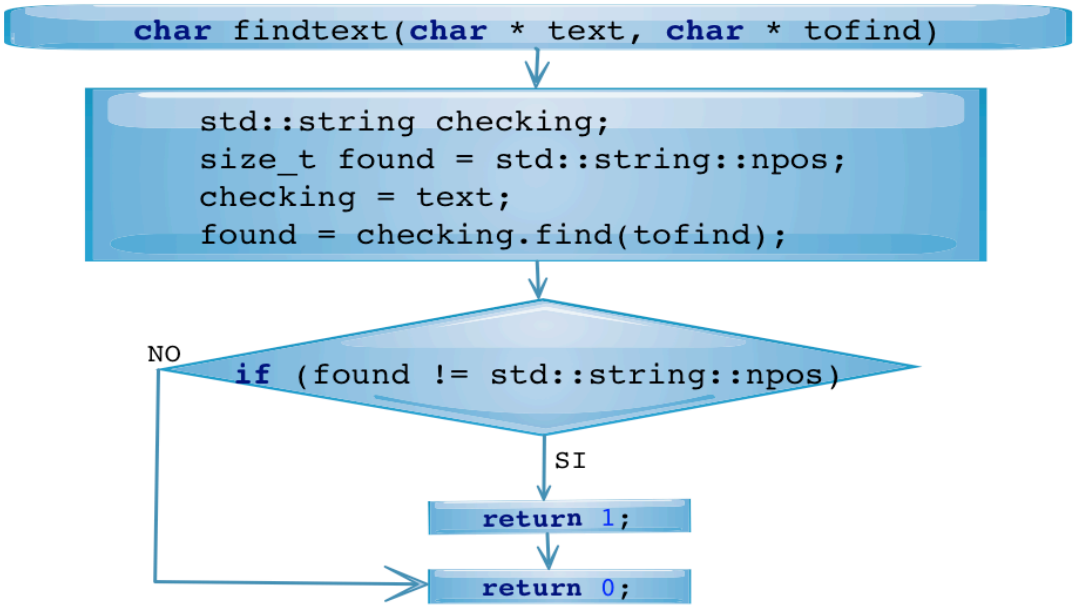


Figura 41. Diagrama de flujo del algoritmo en C++ para búsqueda del sub-string

En caso encontrar el string deseado la función retorna uno, en caso contrario retorna cero. La sintaxis y explicación de la función es propia de C++. Se utilizo el método *find* y se guarda en la variable *found*. Dependiendo de este valor se compara el valor de found dentro de un condicional if para así retorna uno ó cero si se encontró la respuesta deseada ó no se logro encontrar respectivamente.

Esta función es clave porque se utiliza en todas las funciones ya implementadas para crear esta nueva función que envía el comando y confirma su llegada con la función *findtext* en una sola función. De esta forma es posible enviar consecutivamente los comandos requeridos tanto para configuración como los comandos de acción utilizados para conectar y desconectar el modulo Wi-Fi del AP, abrir y cerrar conexiones TCP ó UDP, resetear el módulo, guardar cambios, etc.

- FUNCIÓN GENÉRICA PARA ENVÍO Y CONFIRMACIÓN DE COMANDOS

De forma general se debe crear una función genérica que envíe y devuelva como valor de retorno la confirmación de recibido de cualquier comando por UART utilizando la función *send_comand*. Esta función será replicada para poder enviar los distintos comandos de configuración de forma parametrizada. Por ejemplo el diagrama de flujo de la función para configurar la velocidad en baudios del módulo Wi-Fi se muestra en la Figura 42.

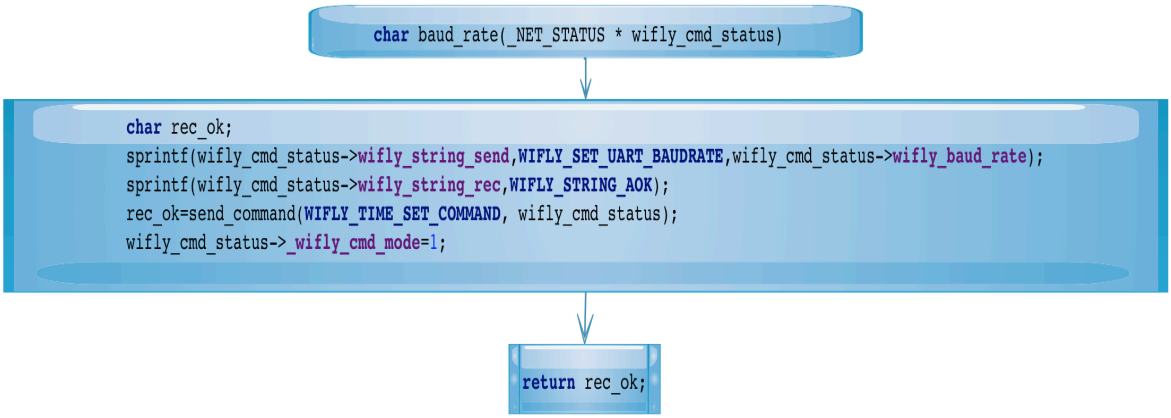


Figura 42. Diagrama de flujo de envió y confirmación del comando para configurar baudios en el módulo Wi-Fi

Como se puede observar en el diagrama de flujo la función es muy simple, inicialmente se cargan los datos a enviar en la variable de la estructura "wifly_string_send" mediante el macro WIFLY_SET_UART_BAUDRATE el cual contiene realmente el comando, en este ejemplo particular seria ""set u b %i\r\n"". Donde el valor entero que se ingresa como variable se toma de la estructura y es igual al valor en baudios deseado. De igual forma se carga la respuesta esperada en la variable "wifly_string_rec" mediante el macro WIFLY_STRING_AOK.

Con el comando a enviar y la respuesta esperada debidamente cargados en los vectores que guardan estos Strings en la estructura tipo NET_STATUS, se procede a ejecutar la función ya conocida *send_comand().* Esta función como ya se conoce, envía el comando, espera la respuesta y la compara con la esperada retornando uno si la respuesta fue la deseada ó un cero si no lo fue. Este mismo valor de retorno es guardado en nuestra función en la variable rec_ok y esta a su vez será el valor principal de retorno de nuestra función baud_rate(). De esta forma retornando esta valor se conoce si el modulo recibió satisfactoriamente el comando ó no.

La anterior metodología ejemplificada en el diagrama de flujo con la función baud_rate() se puede aplicar para cualquier comando que se requiera enviar al módulo Wi-Fi. Adicional a esto se actualiza la variable *wifly_cmd_mode* la cual se indica si el módulo queda en modo datos ó modo comando dependiendo de la función ejecutada en cual estado lo deja. Esta variable anidada de nuevo en la estructura tipo NET_STATUS es útil en el momento de entrar en la función que deja al módulo en modo comando y así saber en qué comando enviar para confirmar este modo.

4.1.1.14 FUNCIÓN PARA INICIALIZACIÓN DE PARÁMETROS DEL MÓDULO WI-FI

Antes de enlazar el módulo Wi-Fi con el AP además del SSID y la contraseña se debe configurar en sus registros (comandos de escritura SET en la hoja de datos) el tipo de protocolo por el cual se van a enviar los datos (TCP ó UDP), si se requiere modo cliente ó servidor, la velocidad de transmisión por UART, los mensajes que se envían por UART al abrir una conexión TCP ó al cerrarla, si se utilizara antena interna ó externa y el modo de conexión al router (manual ó automática). Estos son los parámetros mínimos que se necesitan para inicializar el módulo. Como paso final se le envía el comando "save" para que guarde los anteriores parámetros de forma permanente y adicionalmente se envía el comando "reboot" para que el módulo se reinicie y tome estos parámetros desde el inicio. Inmediatamente después del reboot el módulo conecta al AP el cual tiene la SSID y la contraseña configurada en la inicialización. Todos estos pasos son los que se deben ejecutar en la rutina de inicialización del módulo "init_wifly()".

Para que esta inicialización se realice de manera satisfactoria se requiere que todas las funciones que ejecutan los comandos de configuración sean confirmadas por el módulo Wi-Fi. Si por algún motivo uno de los comandos de configuración no es recibido por el módulo Wi-Fi la función *sendcomand()* retornara cero y el flujo de ejecución se debe interrumpir y se volverá al inicio de la rutina y se enviarán de nuevo todos los comandos hasta que todos sean enviados y confirmados de forma satisfactoria ó hasta que se cumpla un número máximo de intentos de envió.

Por lo anterior se crea la función *init_wifly* la cual asegura que todas las funciones que envían los comandos mínimos de configuración sean ejecutadas satisfactoriamente. En caso de que algún comando no logre ser confirmado, se ejecuta la función desde el inicio dos veces (numero dado en la condición del while) antes de salir y devolver un cero indicando que no se logro hacer la inicialización.

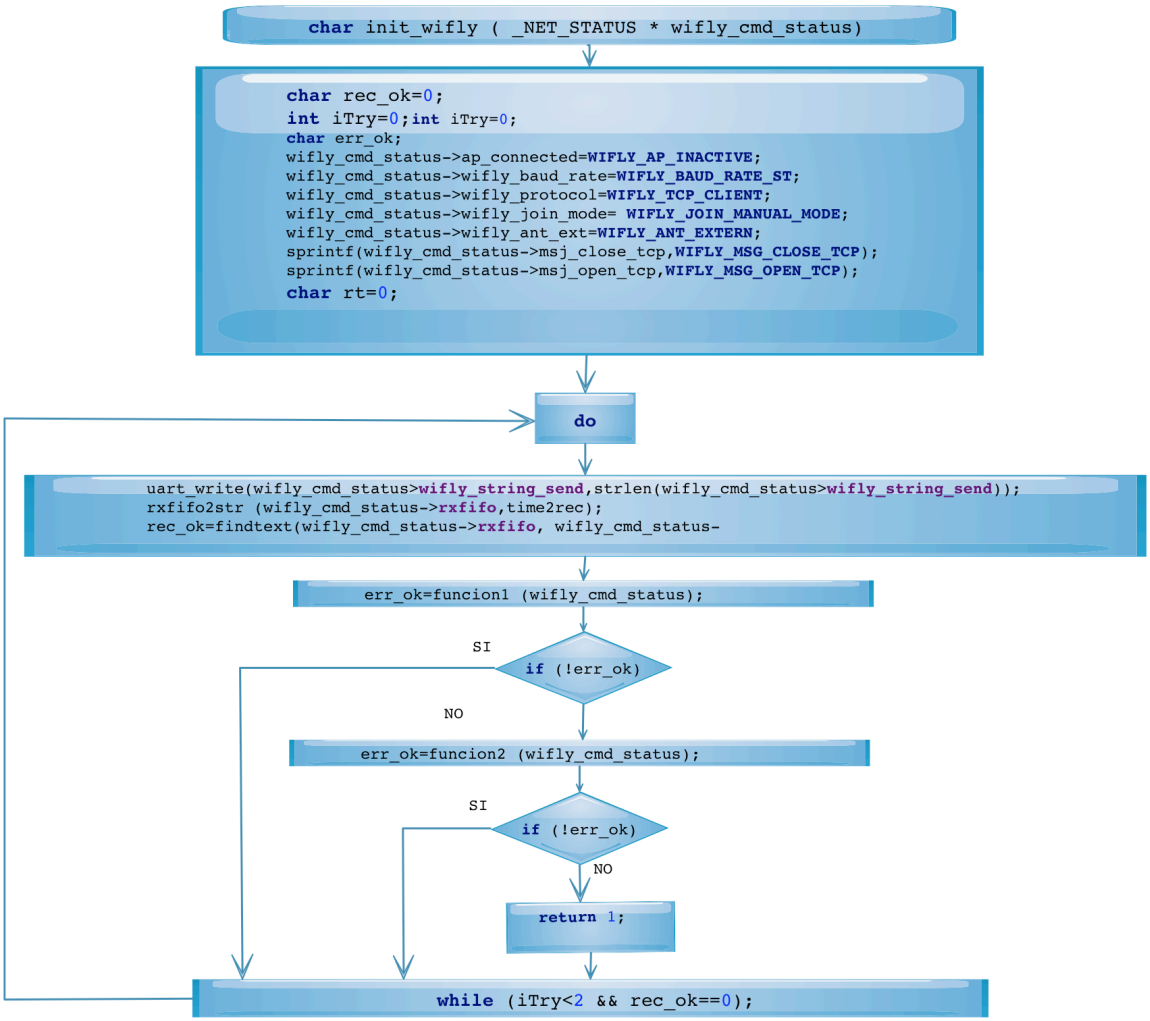


Figura 43. Diagrama de flujo del algoritmo que inicializa el módulo Wi-Fi

En la figura anterior se observa el diagrama de flujo de la función *init_wifly()*. El flujo principal de la función se ejecuta bajo un ciclo do-while (se ejecuta por lo menos una vez) el cual lega a su fin hasta que todas las funciones que envían comandos de configuración sean confirmadas por el modulo Wi-Fi (guardando ese estado en la variable err_ok) ó hasta que se cumplan 2 intentos de ejecución del ciclo para lograr ejecutar todas las funciones de forma satisfactoria.

Antes de iniciar la configuración se observa que se inicializan algunas variables de la estructura network del sistema (ingresada como parámetro con el nombre de wifly_cmd_status) como ap_connected (estado de conexión con el AP), wifly_baud_rate (velocidad en baudios deseada para la comunicación UART), wifly_protocol (variable que indica el protocolo elegido TCP ó UDP), etc. Estos valores con los cuales se inician las variables de la estructura son usados por cada una de las funciones de configuración al momento de enviar el comando con

la función *send_comand()* por esto es importante dejarlos en el valor correcto antes de ejecutar las funciones.

Por cuestiones de practicidad en el diagrama de flujo se dejaron indicadas de forma general las funciones de configuración e inicialización del módulo Wi-Fi como *funcion1()*, *funcion2()* y así sucesivamente con todas las funciones que son necesarias para inicializar el modulo. Es decir estas funciones genéricas son iguales al ejemplo dado en el numeral 3.1.1.13 en el cual se explica de forma general como se envía un comando de configuración al módulo Wi-Fi.

En el ciclo do-while principal se observamos la variable *err_ok* toma el valor de retorno de función1() (función 1, 2, 3 etc. hace referencia a las funciones necesarias para la inicialización del modulo) inmediatamente se hace la comparación de la variable en un if() para tomar la decisión de seguir con la siguiente función de inicialización ó para salir del flujo de la función y volver a ejecutar toda la inicialización de variables y llamado a las funciones necesarias para inicializar el modulo Wi-Fi.

4.1.1.15 CONEXIÓN DEL MÓDULO WI-FI CON UN AP

Si el módulo Wi-Fi logra conectarse con el AP, inmediatamente después se recibe un mensaje por UART indicando que se conectó a la Red, la IP asignada por el AP, el modo de configuración (TCP ó UDP), puerto de escucha, etc. Estos parámetros se pueden observar en la figura 44.

```
*Reboot*
Status:(08:44:55)Break detected. BreakState = False
Status:(08:44:55)Break detected. BreakState = False
WiFly Ver 2.38, 12-11-2012 on 131C83
MAC Addr=00:06:66:13:d5:75
Auto-Assoc marktin chan=1 mode=WPA2 SCAN OK
Joining marktin now..

*READY*
Associated!
DHCP: Start
DHCP in 1668ms, lease=85536s

IF=UP
DHCP=ON
IP=172.20.10.9:50000
NM=255.255.255.240
GW=172.20.10.1
Listen on 50000
```

Figura 44. Mensaje de confirmación de conexión al AP

En la figura 44 se puede observar el mensaje UART de confirmación de conexión al AP. Retomando los pasos para conexión, después de enviar el comando

“reboot” se recibe la confirmación de esté con la respuesta “*Reboot*” del módulo. En la figura 44 los mensajes visualizados en color negro se deben a ruido en los terminales UART generados por el “reboot”. Se observa en las lineas siguientes el nombre del dispositivo Wi-Fi, su modelo, su dirección MAC y la confirmación de conexión al AP cuya SSID en este caso es “marktin”. Posterior a esto, envía confirmación de que fue exitosa la solitud al servidor DHCP, muestra la IP asignada por el AP (como server DHCP), *la máscara de subred* y su *Gateway*.

Según se observa en el mismo paso de conexión al AP se ejecuta el protocolo DHCP y adquiere directamente la IP del AP. Ahora bien el protocolo DHCP en este caso se realiza entre el módulo Wi-Fi y el AP por lo tanto es una dirección IP local, esto quiere decir que no es una IP publica real. Un dispositivo con IP local solo puede garantizar la conexión a dispositivos conectados en el mismo AP, es decir en la misma red local. Para tener acceso a Internet el AP debe tener conexión a Internet mediante cable Ethernet el cual viene del ISP (Internet Service Provider). Con lo anterior concluimos que el hecho de que modulo Wi-Fi conecte con el AP y mediante protocolo DHCP entregue una IP, puede que no tenga acceso a internet debido a que las IP entregadas por el AP son locales. Para tener acceso a internet el AP a su vez deberá tener acceso internet. La IP dada por el AP únicamente puede garantizar conexión con equipos que estén conectados en su misma Red local.

4.1.1.16 CONEXIÓN DEL MÓDULO WI-FI CON UN SERVIDOR TCP

Según el estudio realizado previamente acerca del modo regular de funcionamiento del módulo de comunicación Wi-Fi se debe configurar en modo cliente mediante protocolo TCP. Modo cliente porque siempre se tendrá a un servidor escuchando las peticiones del dispositivo medico y protocolo TCP porque es el utilizado por Websocket (el protocolo usado por el servidor para recibir la información). Esta configuración se hace en la función init_wifly().

Después de ejecutar la función init_wifly la cual configura el módulo por el protocolo y modo deseado, y adicionalmente lograr la conexión con el router, se procede a abrir una conexión TCP en modo cliente. Para lograr esta conexión se ejecuta el comando *open* el cual tiene los siguientes parámetros:

open <address><value>

Donde address en la dirección IP del servidor y value es el número del puerto por el cual está recibiendo solitudes de conexión.

Para enviar el comando se utiliza la función habitual “Send_comand()” la cual retorna uno si la conexión se logró satisfactoriamente. Esta confirmación se logra gracias a que el módulo Wi-Fi devuelve por UART un mensaje predeterminado (aunque se pudo personalizar esto no afecta en nada el funcionamiento) que es detectado por la función *send_comand()*. Cuando se abre la conexión con el servidor TCP el modulo Wi-Fi el módulo queda listo para enviar y recibir mensajes por UART es decir, cualquier secuencia ó string enviado (datos) por UART serán a

su vez enviados por TCP hasta el servidor y lo que se recibe será las respuestas de este servidor.

Al igual que en las pruebas preliminares para el módulo UART se utilizó un software llamado Hércules el cual permite habilitar un servidor TCP en un computador convencional para hacer las pruebas de conexión y recepción de datos. Así pues se realiza una prueba final de transmisión y recepción de datos desde la FPGA mediante el driver diseñado hasta el servidor TCP ubicado en un computador mediante el software Hércules. En la Figura 45 se observa los mensajes recibidos por el servidor. Se utilizo el puerto 45000 al ser un puerto libre de la red privada de la FCV, lo cual lo hace ideal para hacer pruebas.

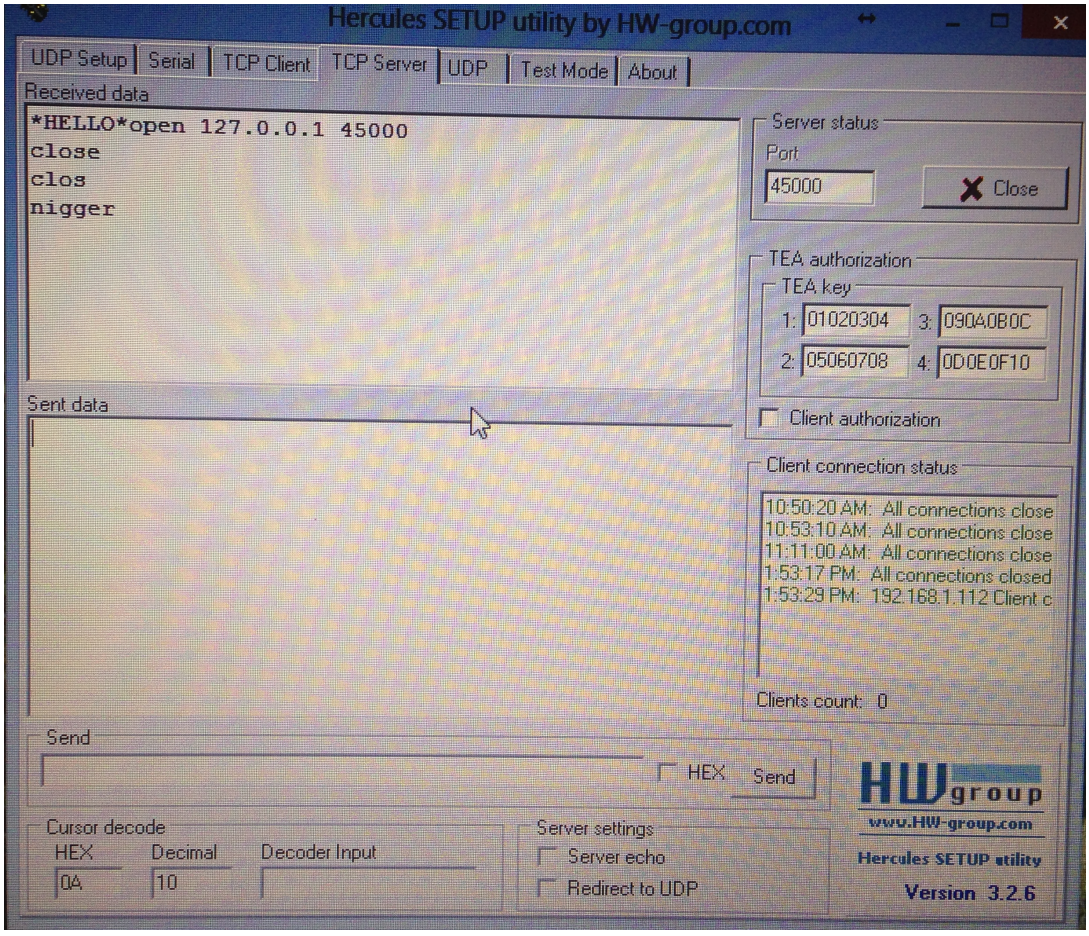


Figura 45. Recepción del mensaje TCP desde la FPGA

Adicional a esta prueba también se relazaron pruebas se transmisión y recepción de datos por protocolo UDP dado que el Ing. Holguer Becerra solicito incluir en la librería el soporte para este protocolo dado que en un futuro posiblemente se implementen servicios como streaming de video y voz en el dispositivo médico, por lo cual el protocolo ideal debido a su rapidez (comprometiendo errores en la transmisión) es UDP. De forma satisfactoria se hicieron y se concluyeron estas pruebas.

4.1.1.17 FUNCIÓN PARA EXTRAER STRING DEL UART DE RECEPCIÓN

La función para extraer Strings del UART de recepción se crea a partir de la necesidad de guardar de forma permanente la IP dada por el módulo Wi-Fi para que posteriormente en otro proceso sea mostrada por la interfaz gráfica del dispositivo médico.

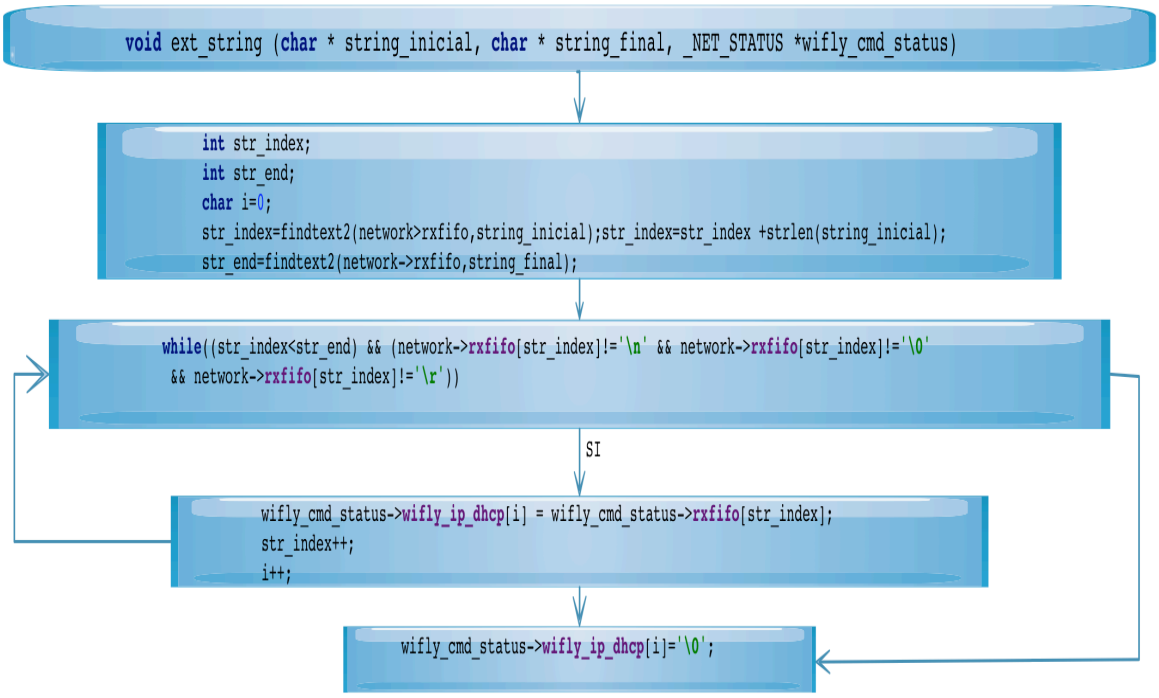


Figura 46. Diagrama de flujo del algoritmo para extraer String

A la función se le ingresa como parámetro dos cadenas de caracteres entre las cuales se encuentra el que queremos extraer llamadas *string_inical* y *string_final*. Se utilizo la función *findtext2()* implementada desde la misma función en C++ que se utiliza para detectar un string dentro de otro, pero esta nueva función devuelve un numero entero con la posición del primer carácter de la cadena de caracteres que se le ingresa como parámetro. Usando de esta forma la función para detectar la posición en el UART de recepción de las cadenas de caracteres ingresadas como parámetro es decir *string_inicial* y *string_final*.

Una vez detectada la posición entre la cual está el string que se quiere extraer, se procede a formar el nuevo string dentro del ciclo while en el cual se quedara mientras el índice de la posición del string final sea mayor al str_index que se va incrementando desde la posición inicial. Si el string a extraer está entre el *string_inicial* y un salto de línea ó el final del string del UART de recepción se tienen en cuenta estas situaciones en el while para truncar adecuadamente la formación del string a extraer.

4.1.1.18 CORRECIÓN DE ERRORES

PETICIÓN DE CONEXIÓN CONSECUTIVA BLOQUEABA EL MÓDULO WI-FI

La gran mayoría de pruebas realizadas con el módulo Wi-Fi fueron satisfactorias, sin embargo en algunas ocasiones el módulo se bloqueaba y dejaba de recibir comandos por UART. La solución para que el módulo volviera a la normalidad fue quitarle su alimentación y conectarlo de nuevo. Para descubrir la causa del problema se observó detenidamente el prompt del procesador NIOSII en el cual se puede observar cual comando se ejecuta un instante antes de bloquearse.

Después de muchas observaciones y con la lectura detallada de la hoja de datos se observó el tiempo que gasta en la inicialización el módulo (después del reboot) como se observa en la tabla 2. Se concluyó que el comando "reboot" que se le ingresaba para que conectara al AP (con su SSID asociada además de guardar los datos configurados de forma permanente al final de la función init_wifly) estaba bloqueando el módulo al enviar consecutivamente el mismo comando sin darle el tiempo necesario para su inicialización. Este extraño bloqueo sucedía porque se ejecutaba más de una vez el comando reboot después de ya estar enlazado con el AP() sin antes desconectarse del mismo. El comando reboot se ejecutaba equivocadamente varias veces debido a que el algoritmo iterativo para las pruebas (después se cambio) enviaba mensajes y si la conexión TCP se perdía el algoritmo volvía a inicializar el módulo por consiguiente la función reboot() volvía a ejecutarse, los datos iniciales volvían a guardarse y después del reboot se solicitaba una nueva conexión al router sin tener en cuenta que ya estaba enlazado.

Function	Description	Time (ms)
Power up	Power up time from reset high or power good to boot code loaded.	70
Initialization	Initialize ECOS.	50
Ready	Load configuration and Initialize application	30
Join	Associate using channel = 0 (full channel scan, mask = 0x1FFF).	80
	Associate using channel = 0 (primary channel scan, mask = 0x421).	15
	Associate using channel = X (fixed channel).	5 - 20
Authentication	Authenticate using WPA1 or WPA2 (highly dependent on access point response).	50 - 250
Aquire IP	DHCP obtain IP address (highly dependent on DHCP server response time).	AP dependent

Tabla2. Valores de tiempo de inicialización [31]

Para solucionar este problema se implementó una *bandera* de conexión al AP mediante la variable ap_connected la cual es guardada en la estructura *network* para ser accesible en cualquier lugar. Esta bandera ejecuta el comando únicamente el comando reboot si su valor es cero es decir si el módulo Wi-Fi no está enlazado al AP. Con este algoritmo condicional se soluciona el problema del bloqueo del módulo.

DESBORDAMIENTO DEL FIFO DE RECEPCIÓN GENERANDO ESCRITURA ERRONEA EN LOCALIDADES DESCONOCIDAS DE MEMORIA

En el análisis del problema anterior debido al reboot, se descubrió un segundo problema ocasionado por el ruido generado por el módulo después de enviar el comando reboot (explicado en la sección 3.1.1.15 e ilustrado en la Figura 47). Para corregir esto se analizó detalladamente lo sucedido y para solucionarlo se optimizó el código modificando la función de recepción de datos y así limitar el tamaño máximo de recepción al tamaño del buffer. La función sin corregir limpiaba el buffer de recepción es decir llenaba con ceros el vector que guardaba el string dependiendo de la longitud recibida por el UART de recepción. Después del reboot se generaban interrupciones por el UART de recepción las cuales no eran bytes sino ruido que genera el módulo después del reboot como se puede ver en la Figura 47.

Para evidenciar el tamaño de las recepciones erróneas se modifico el tamaño del FIFO de recepción de 1K a 200K y se imprime el tamaño de la recepción específicamente cuando se ejecuta la rutina “reboot”. En la Figura 47 se observa que debido al ruido se detectaron 26273 bytes de entrada de los cuales solo serian los útiles los 10 primeros el resto serian ruido. Se puede observar el tamaño de los bytes recibidos gracias a que se cambio el FIFO a 200k pero de forma real este FIFO solo tiene 1K de tamaño real por lo que se reescriben los datos más de 26 veces con el ruido que recibe por el UART de recepción.

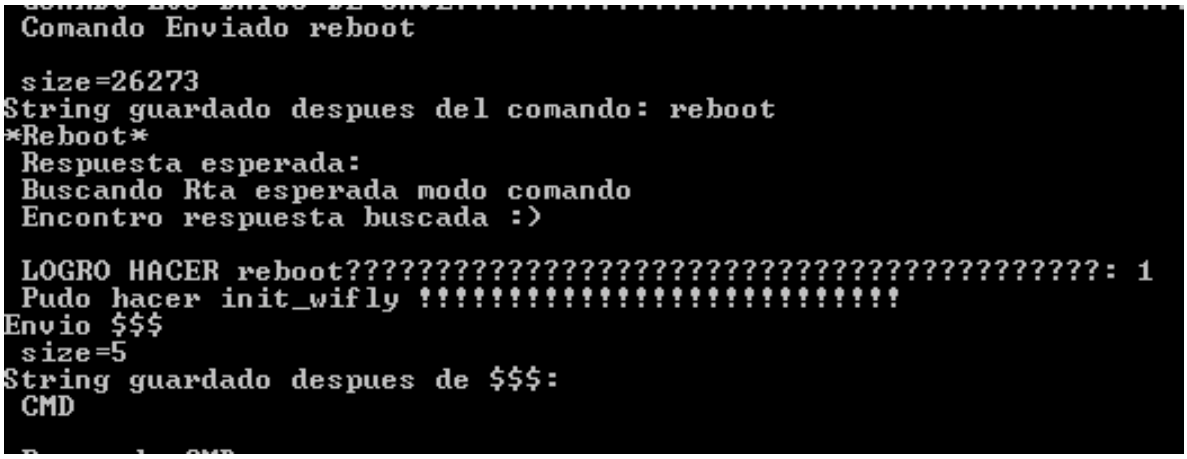


Figura 47. Evidencia del ruido generado al ejecutar el comando reboot

Posteriormente se realizó una prueba con la nueva subrutina y se verificó su funcionamiento de forma equivalente con la rutina “reboot” para comprobar la recepción de datos. Como se puede ver en la Figura 48 (el número se encuentra después de la palabra “size=#####” en la impresión de consola). Se muestra una impresión en la consola en donde observamos el mensaje “Mensaje demasiado grande no puede ser escrito en el buffer /r/n size=1000” Este mensaje confirma el correcto funcionamiento del código limitando el buffer de recepción a 1000 datos. Con esta solución se evita que se sobrescriba en el FIFO de recepción

garantizando que la respuesta inicial quede intacta y así poder dar continuidad en el flujo del programa en esta función.

Al modificar este tamaño máximo de escritura de ceros para limpiar el buffer al tamaño máximo del buffer se soluciono este problema y así la escritura de ceros no llenaría de ceros localidades de memoria que se desconocen que pueden generar un error fatal en el sistema.

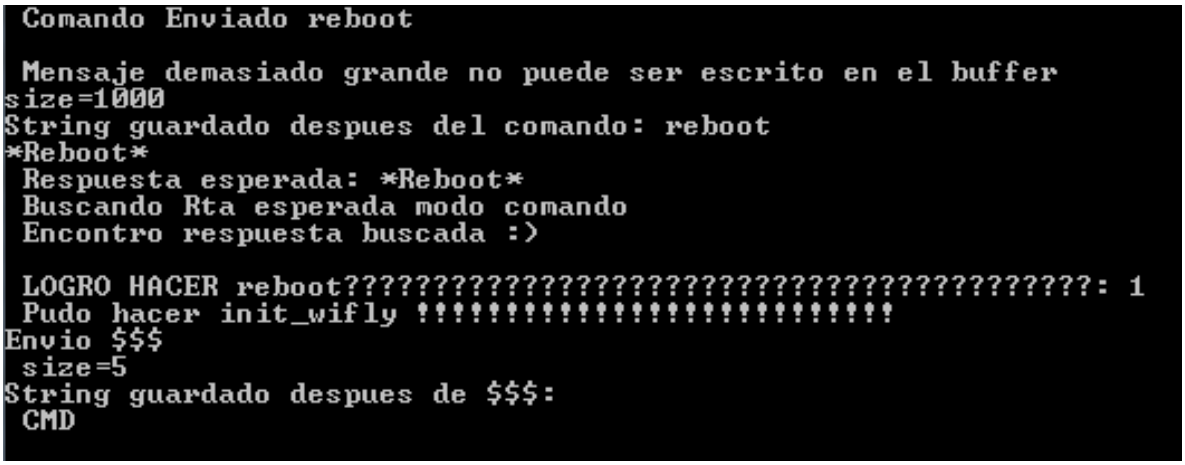


Figura48. Corrección en la recepción de ruido después del comando Reboot

CAMBIO EN LA POLÍTICA DE CONEXIÓN AL AP

Debido a los constantes problemas de ruido con la función reset se cambió el modo de conexión al AP. La conexión con el AP no se realizara inmediatamente después del reboot, solo cuando le sea dado el comando de acción JOIN. Este comando debe ir seguido del nombre de la red Wi-Fi (la SSID).

Value	Policy
0	Manual. Do not try to associate with a network automatically.
1	Try to associate with the access point that matches the stored SSID, passkey, and channel. If the channel is set to 0, the module will scan for the access point. (Default)
2	Associate with ANY access point that has security matching the stored authentication mode. The module ignores the stored SSID and searches for the access point with the strongest signal. You can limit the channels that are searched by setting the channel mask.
3	Reserved.
4	Create an ad hoc network using stored SSID, IP address, and netmask. You MUST set the channel. Unless another ad hoc device can act as DHCP server, set DHCP to 0 (static IP) or use automatic IP. You can use this policy instead of the hardware jumper to create a custom ad hoc network.
7	Create a soft AP network using stored the SSID, IP address, netmask, channel, etc. This mode applies only to firmware versions supporting soft AP mode, not ad hoc mode.

Tabla3. Política de conexión al AP [31]

Como se observa en la figura anterior se adopta el modo de conexión con valor 0 el cual permite el enlace con el AP de forma manual. Este valor de política de conexión ahora debe ser ejecutada en la rutina init_wifly la cual se encarga de configurar todos los parámetros iniciales de conexión.

AJUSTES FINALES DEL DRIVER

- Como paso final para dejar el driver funcional para que cualquier ingeniero de la FCV pueda utilizarlo y entenderlo fácilmente en un futuro se hacen retoques finales a la librería del modulo Wi-Fi. Por ejemplo dejan todos los macros creados en el driver con letra mayúscula como es estándar en los drivers de cualquier dispositivo.

- Se ajustó al valor real en bytes el tamaño de los buffers que tiene internamente el módulo Wi-Fi que se supusieron inicialmente y fueron creados al comenzar con el diseño del driver. En la hoja de datos del módulo Wi-Fi se describe el tamaño de cada uno como se muestra en la tabla 3.

Parameter	Value (Bytes)
FTP Parameters	
file	32
user	16
pass	16
dir	32
wlan Parameters	
ssid	32
phrase	64
DNS Parameters	
DNS host name	64
DNA backup host name	64
comm Parameters	
open	32
close	32
remote	64
deviceid	32

Tabla4. Tamaño de los buffers del modulo Wi-Fi [31]

- Por recomendación del ingeniero José Pinilla se cambian los tiempos que fueron generados con la función *usleep(time)* la cual bloqueaba el procesador por *time* microsegundos por tiempos de sistema operativo los cuales en el sistema final serán menos problemáticos debido a que el sistema funciona en base a tiempos y tareas ejecutadas por el sistema operativo uC/OSII. El formato de tiempo de sistema operativo es "OSTimeDlyHMSM(Horas, Minutos, Segundos, Milisegundos)". Como se tenían tiempos muy cortos (18uS por ejemplo) fue necesario cambiar las constantes de tiempo de todas la funciones porque el mínimo tiempo de sistema operativo es 1 milisegundo.

4.1.1.19 RECONOMIENTO DEL DRIVER PARA EL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

Finalizado el diseño e implementación del driver para el módulo Wi-Fi se emprende la tarea de desarrollar un driver para el módulo de transmisión celular. Por motivos de confidencialidad con la empresa no se da la referencia exacta del módulo ni su tecnología de transmisión pero se explicaran detalles generales de su funcionamiento y como se desarrollo el driver.

El dispositivo de transmisión de datos vía celular es un módulo de fácil configuración muy versátil y de implementación rápida. Su forma de configuración como ya se ha mencionado en el presente libro es muy similar al módulo Wi-Fi es decir se le envían comandos mediante protocolo UART y se configuran sus registros de operación para enviar datos mediante los protocolos como TCP, UDP, FTP, etc.

El módulo no solamente sirve para transmisión de datos, también puede hacer llamadas mediante la red celular, enviar mensajes de texto, hacer streaming de video por puertos especiales con los que cuenta y en general soporta cualquier acción que haga cualquier teléfono celular convencional. Para su funcionamiento se debe adquirir una tarjeta SIM con los servicios requeridos activados es decir si se requiere transmitir datos la SIM debe contar con un plan de datos activo ó si se desean hacer llamadas se debe adquirir un plan de voz. Un diagrama de bloques del dispositivo internamente se puede observar en la Figura 49

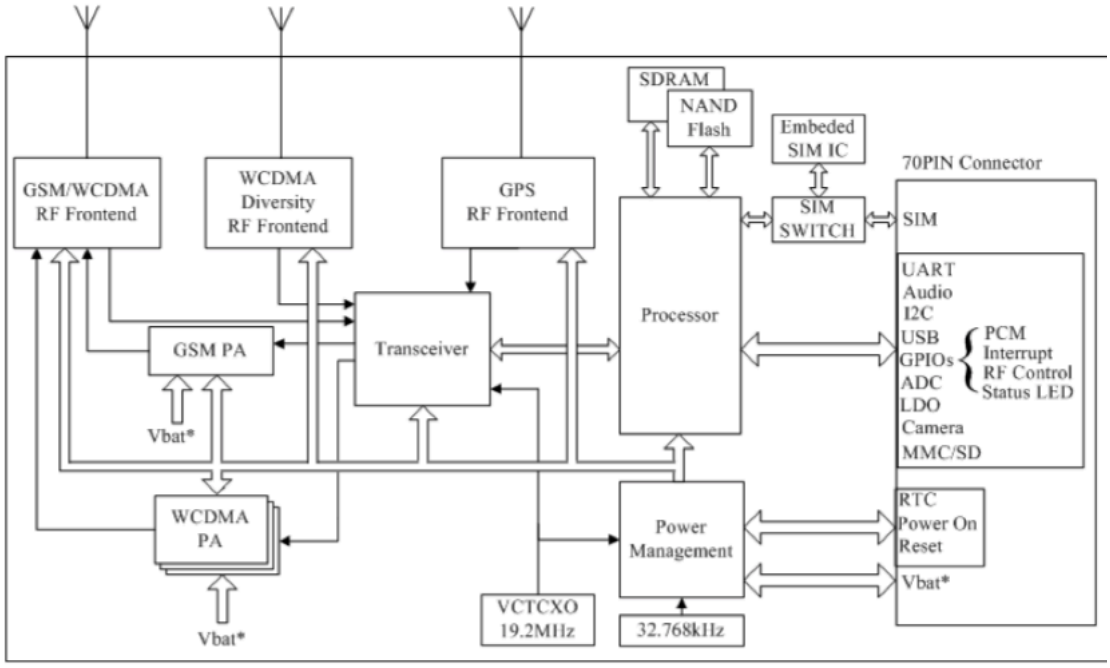


Figura 49. DIgrama en bloques del módulo de comunicacion celular[29]

Como se observa en la figura 49 el módulo cuenta internamente con módulos para soportar distintas tecnologías de red celular. También cuenta con memoria SDRAM, NAND, protocolos de comunicación UART, I2C, puertos de audio, para memoria SD, un procesador que interpreta los comandos y gestiona el manejo de todas las señales y sus módulos internos. Además cuenta con 3 conectores para antenas las cuales se explicaran más adelante.

Para hacer uso de los módulos extras de llamadas se deberán conectar en sus pines de propósito específico un manos libres ó a un parlante que permita escuchar el interlocutor y de igual forma un micrófono convencional. Estas funcionalidades no fueron incluidas en el dispositivo médico dado que el módulo solo se utilizara para transmisión de datos de monitorización vital por lo tanto en la board principal del dispositivo en desarrollo solo se tendrán las señales necesarias para enviar y recibir datos por UART, alergización del módulo con sus señales de control y las de la tarjeta SIM.

Como ventaja adicional este módulo tiene GPS integrado el cual soporta 2 modos para adquirir las coordenadas de Georeferenciación. La primera adquiere la señal directamente de los satélites y es bastante lento en su primera adquisición dado que tiene que ubicar los satélites disponibles (tiene que buscarlos dado que se podría encontrar en cualquier lugar del mundo). El segundo método es conocido como “GPS asistido” y se apoya en datos brindados por la red celular configurada mediante los datos de la SIM para así apoyarse en los datos que ella suministra y obtener las coordenadas de forma considerablemente mas rápida.

El módulo fue elegido dado su tamaño fácilmente integrable con la board del monitor de signos vitales, su bajo consumo teniendo en cuenta los parámetros que debe tener un dispositivo médico en la actualidad, tiene un módulo de Georeferenciación integrado y funciona con bajos niveles de voltaje.

4.1.1.20 DISPOSICIÓN DE LAS ANTENAS DEL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

El dispositivo cuenta con conectores para antenas tipo uFL las cuales son las encargadas de tomar la señal de red celular y del modulo GPS tal como se muestra en la Figura 50.

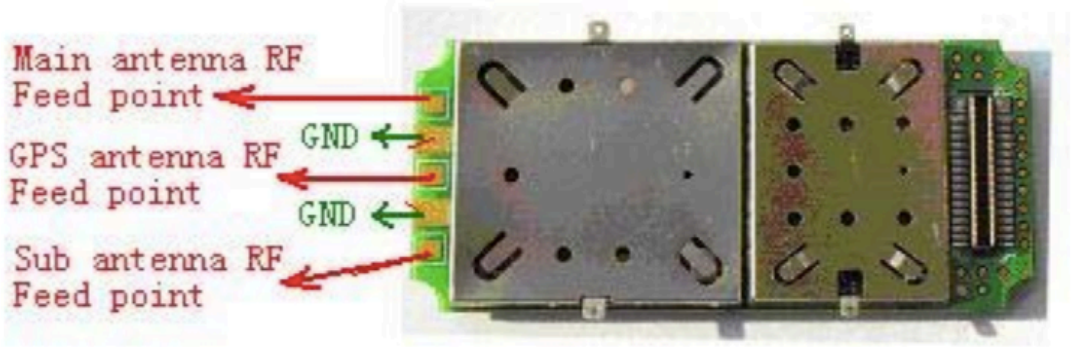


Figura 50. Base para conexion de las antenas del módulo de comunicacion celular y georeferenciación[29]

Cada conector de la antena está debidamente etiquetado para la adecuada conexión de sus antenas. “Main antenna” es la antena principal por la cual toma la señal de las redes celular para acceder a servicios como voz, sms ó transmisión de datos. El conector centra llamado “GPS antenna” como su nombre lo indica es el pad destinado a conectar la antena que tomara la señal GPS de los satélites para hacer la Georeferenciación del módulo.

Es de resaltar que la empresa que suministra estos módulos tiene 3 especificaciones distintas de este módulo dependiendo del lugar del mundo en el cual se vaya a usar. Esto es debido a que en diferentes países se manejan distintas frecuencias de portadora por lo cual se debió elegir el modulo adecuado para las redes celulares colombianas. Específicamente el que fue adquirido para el dispositivo médico es un módulo tribanda que soporta tecnologías UMTS/HSPA 850/900/2100 MHz Esto garantiza que funcionara bien con todos los operadores celulares colombianos dado que estos precisamente trabajan a estas frecuencias.

4.1.1.21 ENERGIZACIÓN Y SEÑAL POWER ON DEL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

Físicamente el módulo es bastante pequeño y plano por lo cual no incluye ningún periférico distinto a un conector con todas las señales para su comunicación y configuración disponibles. En este bus de datos se encuentran las señales necesarias para conectar manos libres, micrófono para llamadas, cámara, etc. En el caso particular en el que va a ser usado se debieron conectar las señales para SIM-CARD (tamaño estándar), Tx y Rx para su comunicación por UART, sus señales para energización y adicionalmente una señal llamada PWR_ON la cual permite encender ó apagar el modulo mediante un flanco de bajada tal como se muestra en la figura 51.

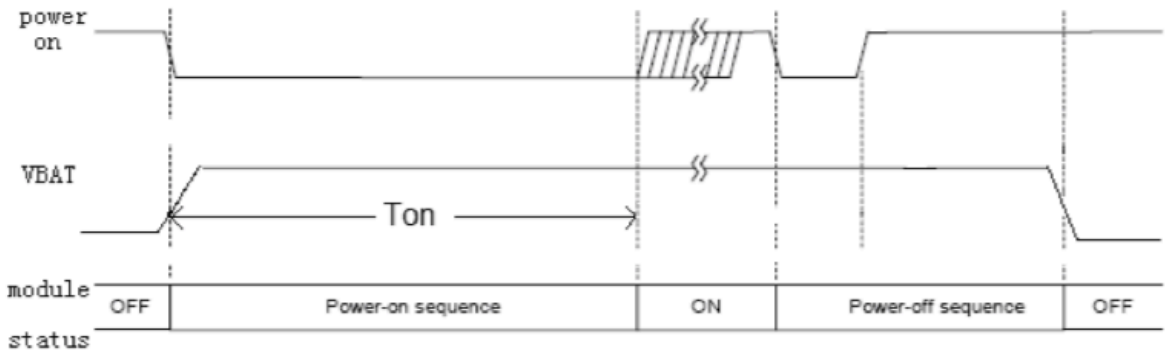


Figura 51. Diagrama de tiempos para encendido y apagado del módulo de comunicacion celular[29]

Como se puede observar en el diagrama de tiempo únicamente se debe dar un flanco de bajada es decir pasar de un estado lógico alto a bajo la señal power ON y esperar un tiempo Ton (64 milisegundos según la hoja de datos del modulo) para que este inicie y culmine su secuencia de encendido. Para el apagado del módulo se le da un flanco de bajada de igual forma que para encenderlo y se deja un tiempo para que ejecute su secuencia de apagado. El funcionamiento de esta señal power_on es similar a un *toogle*, por lo tanto si este pulso es descrito en una función en el driver cada vez que sea ejecutada se podrá encender ó apagar el modulo dependiendo de su estado inicial.

4.1.1.22 CONEXIÓN UART DEL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

Para lograr una adecuada conexión del módulo UART se consulto el datasheet del dispositivo y se observaron algunas particularidades en la salida y entrada de voltaje del modulo UART 3G.

El UART se puede configurar de dos formas: La primera utiliza las señales usuales TxD (transmisión) y RxD (recepción) y la señal de tierra como se ilustra en la Figura 52. El segundo modo utiliza adicionalmente las señales CTS y RTS. CTS indica si el módulo UART está preparado para recibir datos y RTS indica que el

módulo UART quiere enviar datos. Dado que en la board del equipo médico solo se tienen conectadas las señales TxD y RxD se elige el primer modo de configuración.

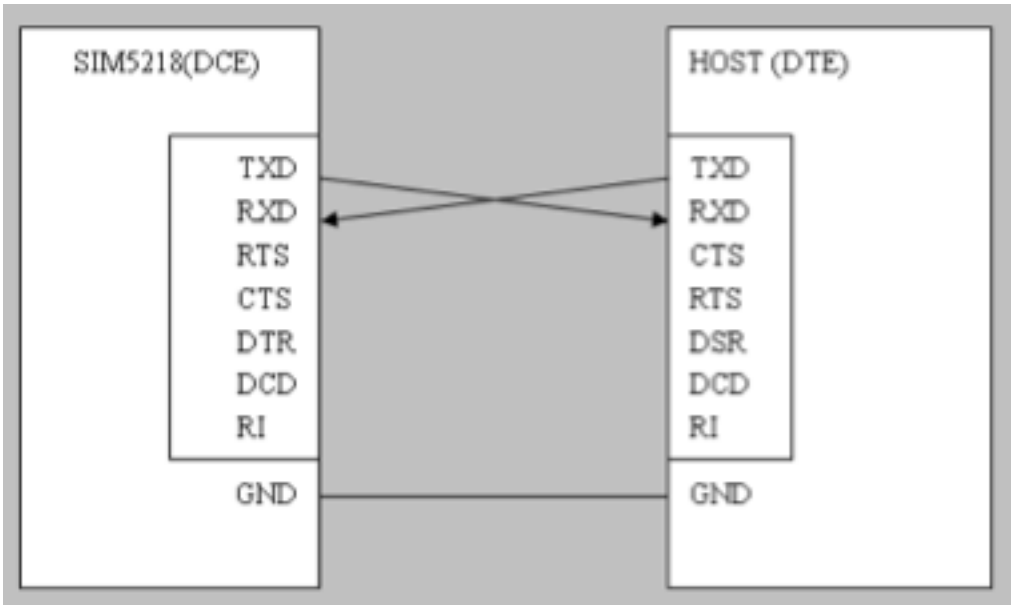


Figura 52. Diagrama de conexion del UART del modulo de comunicacion celular[29]

Adicionalmente se observa que los voltajes de salida no son a 3.3 Volts como en el módulo Wi-Fi, en este caso tiene su propio protocolo de voltaje (Tabla 5). Esto hace que sea incompatible para conectarse de forma directa con la FPGA que soporta solo voltajes de 3.3V.

Para poder usar el módulo se deben implementar traductores de voltaje los cuales adaptan el valor lógico de voltaje de la FPGA al modulo de transmisión celular y viceversa. En la tabla 5 se observan los valores lógicos de voltaje mínimos y máximos que recibe y entrega este módulo.

Parameter	Min	Max	Unit
Logic low input	0	0.3*VDD_EXT	V
Logic high input	0.7 *VDD_EXT	VDD_EXT +0.3	V
Logic low output	GND	0.2	V
Logic high output	VDD_EXT -0.2	VDD_EXT	V
Note: VDD_EXT=2.6V, is module internal IO reference voltage.			

Tabla 5. Tabla de valores de voltaje para estados lógicos [29]

En la board principal del dispositivo medico fueron implementados estos traductores de voltaje para que la FPGA pueda comunicarse con el módulo sin

problemas. En pruebas posteriores se probaran estos traductores después de las pruebas de configuración inicial del módulo.

4.1.1.23 PRUEBAS INICIALES CON EL KIT DE DESARROLLO DEL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

Como paso inicial para comenzar con el afianzamiento en el manejo del modulo celular se comenzó con la identificación del kit de desarrollo proporcionado en la empresa para hacer pruebas de configuración y así lograr ejecutar sus funciones básicas. Se buscó la hoja de especificaciones del kit de desarrollo fabricado por la propia empresa que fabrica el módulo celular debido a que en la FCV fue suministrada la hoja de datos un otro kit . En la figura 53 se puede ver físicamente el kit de desarrollo.

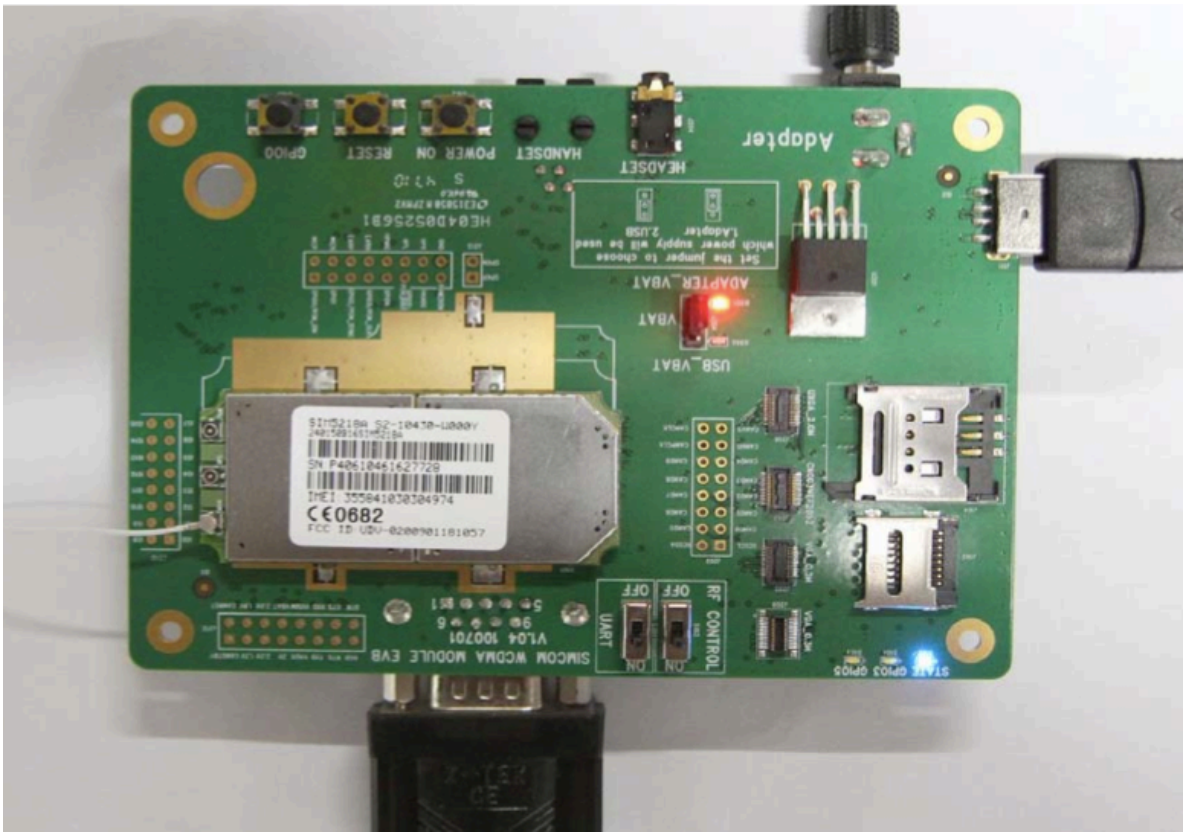


Figura 53. Kit de desarrollo del módulo de comunicacion celular

El kit consiste en una tarjeta en la cual de dispone de pulsadores, slot para tarjeta sim y micro sd, conectores para comunicación por UART, USB, puertos de audio para manos libres, parlantes y micrófono entre otras puertos y conexiones. De lo anterior lo único que se usó fue el slot para tarjeta SIM y el puerto USB el cual comunicaba directamente el módulo con un puerto COM habilitado para recibir los datos de forma serial en un computador convencional. Es decir internamente el kit

de desarrollo tiene un conversor UART a USB mediante el cual se pudieron hacer todas las pruebas iniciales.

Después se asegurarse del funcionamiento del kit de ealuación. Dependiendo de su forma de energización la cual podía ser USB ó externa de 5Volts se cambia de posición un jumper tal cual se indica en la hoja de datos del kit de desarrollo. Se trabajo la energía del puerto USB (distinto al de comunicación por UART-USB).

Además de energizar el kit de desarrollo e insertar la sim en el slot correspondiente se conectaron las antenas de red celular y GPS en los respectivos conectores del módulo. La conexión mecánica de las antenas y los conectores tiene un buen ajuste. En la figura 54 se observan las antenas conectadas.

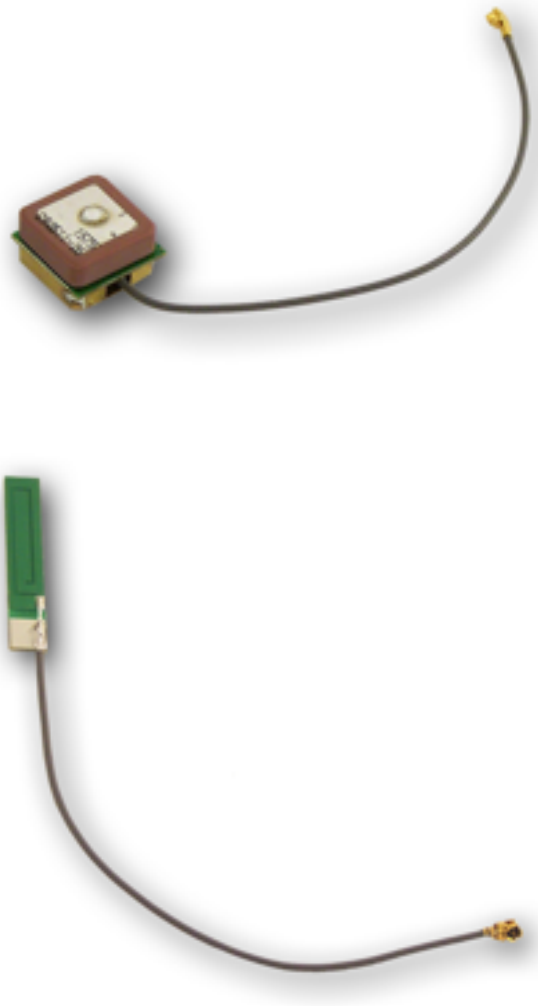


Figura 54. Antenas del módulo de comunicacion celular[30]

Se buscó el set de comandos AT del módulo celular abriendo una cuenta en la página del fabricante de forma obligatoria. Seguido a esto se investigó acerca del uso de estos comandos en módulos de comunicación y routers de distintas empresas. Los comandos AT son usados de forma estándar en módulos de comunicación celular, routers y diversos dispositivos de conectividad a internet los cuales permiten por ejemplo controlar ó un dispositivo de forma remota (Figura 55).

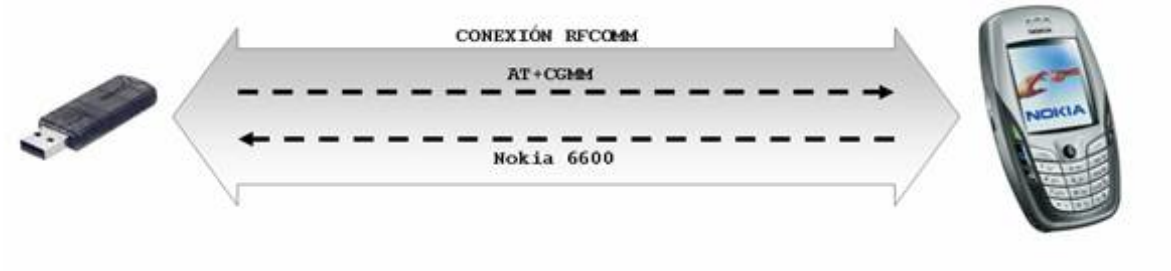


Figura 55. Ejemplo de configuracion de un dispositivo de comunicación mediante comandos AT[30]

A pesar del uso genérico de estos comandos como un estándar en muchos dispositivos de comunicación el fabricante recomiendo configurar el módulo solo con el Set de comandos suministrado en su página Web para el módulo especifico. Así se asegura de que los comandos AT ingresados serán validos y se garantizara su funcionamiento.

Se analizó de forma general el Set de comandos AT descargado de la página del fabricante y se comprendió de forma general como enviar comandos. A pesar de que el set de instrucciones es muy denso (mas de 500 hojas) los comandos a utilizar para transmisión por protocolos como TCP ó UDP solo son contenidos en un capítulo específico (Tabla 6).

De igual forma se investigó de forma general como hacer llamadas, enviar mensajes y demás funciones para aprender las diferentes funciones del módulo. El uso de estos comandos es muy simple. La configuración es similar al módulo Wi-Fi el cual se configura y con el envió de comandos por UART. Los comandos AT siempre empiezan con la palabra seguida del símbolo de suma “AT+” (esto denota que se está enviando un comando AT). Después de este encabezado se envía el comando especifico y el módulo responde con “OK” si el comando es válido y en algunas ocasiones con otros mensajes dependiendo del comando ejecutado tal como sucedía en el módulo Wi-Fi.

Se instalaron los drivers necesarios para comunicar el puerto COM en un computador convencional con el módulo celular ubicado dentro del kit de desarrollo. Por defecto el puerto UART del módulo celular viene configurado a una velocidad de 115200 baudios según dice en la guía del Kit de desarrollo.

Después de la identificación del kit de desarrollo, y energización del mismo se utiliza de nuevo el mismo software para transmisión y recepción de datos seriales

usado para el módulo Wi-Fi (Comm-Operator). Configurando previamente el software para recibir por el puerto COMM a 115200 baudios se empieza con el envió de comandos al modulo.

El primer comando que se envía para probar que se cuenta con una comunicación exitosa con el módulo es únicamente el comando “AT” a lo cual el módulo debe responder con “OK”. De forma practica en el kit de desarrollo se comunico exitosamente el módulo ejecutando distintos comandos y confirmando la respuesta deseada según indica el datasheet.

Adicionalmente de realizaron llamadas telefónicas satisfactoriamente por red celular con el modulo dentro del kit de desarrollo para probar su funcionalidad aunque esta función no se implementara de forma real en el dispositivo medico.

4.1.1.24 ENVÍO DE DATOS POR TCP EN EL KIT DE DESARROLLO

Después de lograr enviar comandos satisfactoriamente al módulo de red celular se procede con la prueba de envió de datos por TCP. En la guía que relaciona set de comandos se encontró un resumen de los comandos utilizados para el envío de datos en modo cliente TCP que así como en el módulo Wi-Fi será el modo normal de funcionamiento dentro del dispositivo médico (tabla 6).

8.2.2 TCP client

Commands and Responses	Comments
AT+NETOPEN=”TCP”,80 Network opened OK	Activate the specified socket’s PDP context and Create a socket.
AT+TCPCONNECT=”192.168.0.1”,80	Attempt to establish the TCP connection

OK	with the specified Tcp server.
AT+TCPWRITE=8 >ABCDEFGH +TCPWRITE: 8, 8 OK Send ok	Send data to server.
AT+NETCLOSE Network closed OK	Disconnect the connection with server and Deactivate the specified socket's PDP context.
+IPCLOSE: <client_index>, <close_reason>, <remote_IP>, <port>. +NETCLOSE: 1	Disconnected by the server **NOTE:** Client_index: This value identifies the client index; When the mode is single-client, this value is 255. Close_reason: This value is the reason for close. Remote_IP: This value identifies the IP address of sever. Port: This value identifies the port of sever, the range of permitted values is 0 to 65535

Tabla 6. Comandos AT para configuración del protocolo TCP/IP del módulo de comunicación celular [29]

Adicionalmente se consultaron drivers y diversa documentación en línea para encontrar y validar la secuencia mínima de comandos y configuración necesaria previa a la conexión por TCP.

Para conectar adecuadamente a la red celular de la empresa operadora que brinda el servicio de datos móviles, se debe configurar el APN el cual es el nombre del punto de acceso el cual es distinto para cada operador celular. Sin configurar el APN el módulo no podrá acceder al servicio de datos móviles. No sobra decir que se debe contar con un plan de datos previamente activado en un operador celular para poder hacer transmisión de datos.

El módulo tiene 15 registros para configurar diferentes APN en dado caso de contar con mas de un servicio de APN por operador celular. Un ejemplo de cómo utilizar el comando para configurar el APN es el siguiente:

"AT+CGSOCKCONT=1,\"ip\",\"internet.movistar.com.co\"\r\n"

La cadena de caracteres “internet.movistar.com.co” es APN que utiliza la red celular de Movistar en Colombia.

Con el anterior comando se configurar en el registro 1 (en el cual se guardan los diferentes APN) el APN de Movistar. Posterior a esto se envía el siguiente comando para que el modulo tome como APN el número 1 del registro de APNs.

“AT+CSOCKSETPN=1\r\n"

Con la adecuada configuración del APN se puede proceder a abrir el socket por protocolo TCP utilizando el puerto 80:

"AT+NETOPEN=\"TCP\",80\r\n"

Cuando se ejecuta el anterior comando se solicita al operador celular el servicio de internet móvil . Si la solicitud es satisfactoria el módulo responde el mensaje “Network opened” momento en el cual se tiene la certeza de tener una IP publica para acceder a los servicios Web. Si el modulo no logra abrir el socket posiblemente no se cuenta con plan de datos en la SIM utilizada.

Para comprobar que IP fue asignada se ejecuta el comando:

"AT+IPADDR\r\n"

Ingresado este comando el módulo responde con la IP asignada. Antes de abrir el socket este comando no mostrará ninguna IP.

Luego de tener una IP pública se procede a abrir una conexión TCP con un servidor remoto que también soporte una IP pública. En este punto del desarrollo se tuvo un problema debido a que en el sitio de trabajo en la FCV se contaba con acceso a internet mediante un router el cual siempre otorga direcciones IP’s locales las cuales no podrán ser accedidas desde un dispositivo con IP pública como el modulo celular. Por este motivo se discutió y se solucionó el problema conectando a internet el computador que actuara como servidor TCP a un Smartphone que permite compartir internet. Así es posible recibir datos porque las dos IP’s publicas se podrán transmitir datos sin problemas por internet (por primera vez).

Para realizar la transmisión se utilizó adicionalmente una app que funciona bajo el sistema operativo Android llamada “Port Forward” la cual sirve para redirigir puertos. Esto fue necesario debido a que el puerto por el cual entraba la solitud al servidor desde le módulo celular era el 80 y en el software “Hércules” el cual se uso para simular un servidor TCP se abría el puerto 45000 como siempre había sido usado. Entonces la aplicación recibe conexiones por el puerto 80 y las redirige al puerto 45000 en el computador al cual se le está compartiendo internet.

Después de solucionar este inconveniente de IP’s publicas se abre la conexión TCP con el comando:

"AT+TCPCONNECT=\"190.66.23.9\",45000\r\n"

en donde la IP del servidor en este caso es 190.66.23.9 el cual debería ser la IP del computador que actúa como servidor TCP y 45000 el puerto que tiene abierto para escucha de solicitudes TCP este mismo servidor.

Seguido al comando debe recibirse la respuesta de confirmación exitosa “Connect ok”. De lo contrario se recibirá “Connect faul”.

Para enviar comandos vía TCP se debe solicitar autorización al módulo para el envío del mensaje con el comando y luego de esto enviar el mensaje, es decir este comando tiene 2 pasos para su ejecución.

"AT+TCPWRITE=100\r\n"

En donde el número 100 es el número de bytes a transmitir (el buffer máximo de envió es de 1Kbyte). Si el módulo está disponible se autoriza a transmitir los datos con la respuesta “>”. De lo contrario el módulo responderá “TCP Busy”.

El segundo paso para concluir la transmisión (después de la autorización para el envío de datos) es simplemente enviar por UART los datos a enviar por TCP.

Si los datos se enviaron satisfactoriamente el módulo responderá con el mensaje “Send ok” es decir se recibió el acuse de recibo (ACK) por protocolo TCP.

Se enviaron datos satisfactoriamente y se hicieron pruebas mediante software wireshark comparando los datos enviados con los recibidos para revisar el error generado por la transmisión.

4.1.1.25 CONEXIÓN DEL MÓDULO DE RED CELULAR A LA BOARD PRINCIPAL DEL DISPOSITIVO MÉDICO

Después de las pruebas realizadas de forma satisfactoria en el módulo de red celular dentro del kit de desarrollo se procede a colocarlo en la board principal del dispositivo médico para probar los traductores de voltaje mencionados en el numeral 3.1.1.20. De esta forma se podrá probar el hardware aun no probado en la board principal del dispositivo medico y a su vez la comunicación directamente con la FPGA.

Para este módulo no se hizo conexión directa con la FPGA debido a los problemas de compatibilidad de voltaje mencionados en el numeral 3.1.1.20, inconvenientes que no se tuvieron en el módulo Wi-Fi. Por esto este paso se omitió.

Después de soldar el módulo celular a la board principal del dispositivo médico se ingresó la SIM y se probó con la instanciación de otro módulo UART en hardware idéntico al utilizado para hacer el driver Wi-Fi. Se enviaron comandos y se recibió satisfactoriamente su respuesta.

4.1.1.26 DISEÑO DEL DRIVER PARA EL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

Con los pasos de conexión mínimos definidos en las pruebas con el módulo de desarrollo se da inicio al diseño del driver.

Dadas las características estándar del protocolo de comunicación UART que tienen en común los dos módulos (Wi-Fi y celular) se utilizan exactamente las mismas funciones y librerías juntos con sus algoritmos de control para diseñar e implementar el driver en el dispositvo de transmisión celular. Es decir se sigue la misma lógica de envío de comandos y verificación de la confirmación de recibido para dar continuidad ó reintentar el envío de todos los comandos tantos en la rutinas de envío de datos simple como en las de inicialización del módulo.

De esta forma se usaron exactamente las funciones para el manejo de cadenas de caracteres diseñadas para el módulo Wi-Fi pero cambiando sus variables de nombre identificando claramente en su nombre que se trata del dispositivo de transmisión de datos via celular. Como función adicional se extrae la intensidad de la señal del modulo con la función ext_string usada de igual forma para el módulo Wi-Fi. Esta intensidad de señal sera usada para retroalimentar la GUI para que el usuario este informado de que tan fuerte es la señal para transmisión.

4.1.1.27 CORRECIÓN DE ERRORES Y OPTIMIZACIÓN DEL DRIVER PARA EL MÓDULO DE TRANSMISIÓN DE DATOS POR RED CELULAR

Se encontró que si no existe servidor el tiempo de espera en recibir el mensaje “Connect faul” es superior a un minuto, tiempo durante el cual el módulo deja de recibir comandos AT esperando conexión.

Para dar solución a este inconveniente se modificó los parámetros del socket en el cual se varia el tiempo de reintentos TCP que viene estándar en el módulo. De esta forma bajo la espera de un minuto a 10 segundos, tiempo suficiente para recibir confirmación de conexión o intento fallido de conexión (si el servidor está disponible la conexión es instantánea).

4.1.1.28 DISEÑO DEL DRIVER PARA EL MÓDULO DE GEOREFERENCIACIÓN GPS

Como el módulo GPS esta integrado en el mismo módulo de transmisión celular únicamente se tuvieron que ejecutar los comandos necesarios para adquirir la posición GPS y guardar estas coordenadas de latitud y longitud en la estructura “Network” para ser enviada posteriormente al servidor Websocket y graficar estas coordenadas mediante Google Maps.

Existen tres modos de operación del módulo GPS. El modo standalone y dos modos asistidos por la red celular. En el modo standalone el módulo GPS decodifica los datos directamente de los satélites que se encuentran disponibles. En el modo asistido el GPS se apoya de la red celular existente para obtener la información de los satelites presentes en determinada ubicación geografica y

posteriormente toma la señal de esos satelites para hacer el cálculo de la posición ó directamente la red celular calcula la posición y la envia al dispositivo (Tabla 7).

MS-assisted	**Server**	**Module**
Location server sends aiding data that is valid for the current fix	Send aiding data	
Module sends code phases		Code phases
Server calculates position	Calculate position	
MS-based	**Server**	**Module**
Location server sends aiding data that is valid for the current fix	Send aiding data	
Module calculates position		Calculate position
Standalone	**Server**	**Module**
Module demodulates data from GPS satellite		Demodulates GPS satellite data
Module calculates position		Calculate position

Tabla 7. Modos de operación del modulo GPS [31]

El modo standalone fue el modo de operación que se implementó en el driver GPS. Esta configuración a pesar de ser considerablemente mas lenta (alrededor de 90 segundos desde el momento en que se solicita la posición y el instante en que llegan las coordenadas) es el que puede dar la posición en todo momento sin importar si existe señal de red celular, si está la simcard ó si el APN fue configurado correctamente.

Para solicitar la información al módulo se tiene que encender el módulo GPS (internamente dentro del módulo de red celular) enviando el comando:

“AT+CGPS=1,1\r\n”

Después de encendido el módulo se le envia el siguiente comando para solicitar la posición:

“AT+CGPSINFO”

Si se requiere la posición GPS cada x segundos se le ingresan parámetros al comando anterior:

“AT+CGPSINFO=x,0”

En donde X es el valor en segundos en los que el módulo devuelve las coordenadas por UART.

Se recomienda a la empresa agregar al driver en un futuro el GPS asistido el cual mejoraría considerablemente la velocidad en la adquisición de la posición pero tendría que contar con una red celular activa y un plan de datos para tomar esta posición facilmente.

En la Figura 56 se muestran los comandos enviados y el tiempo que se tarda en entregar la posición (aproximadamente 90 segundos dado que se entrega la posición cada 30 segundos). En la misma figura se puede detallar el formato de las coordenadas. Únicamente los dos primeros valores son las coordenadas de latitud y longitud respectivamente es decir los valores que seran extraidos y ubicados en una estructura tal y como se ha hecho anteriormente con varios parámetros.

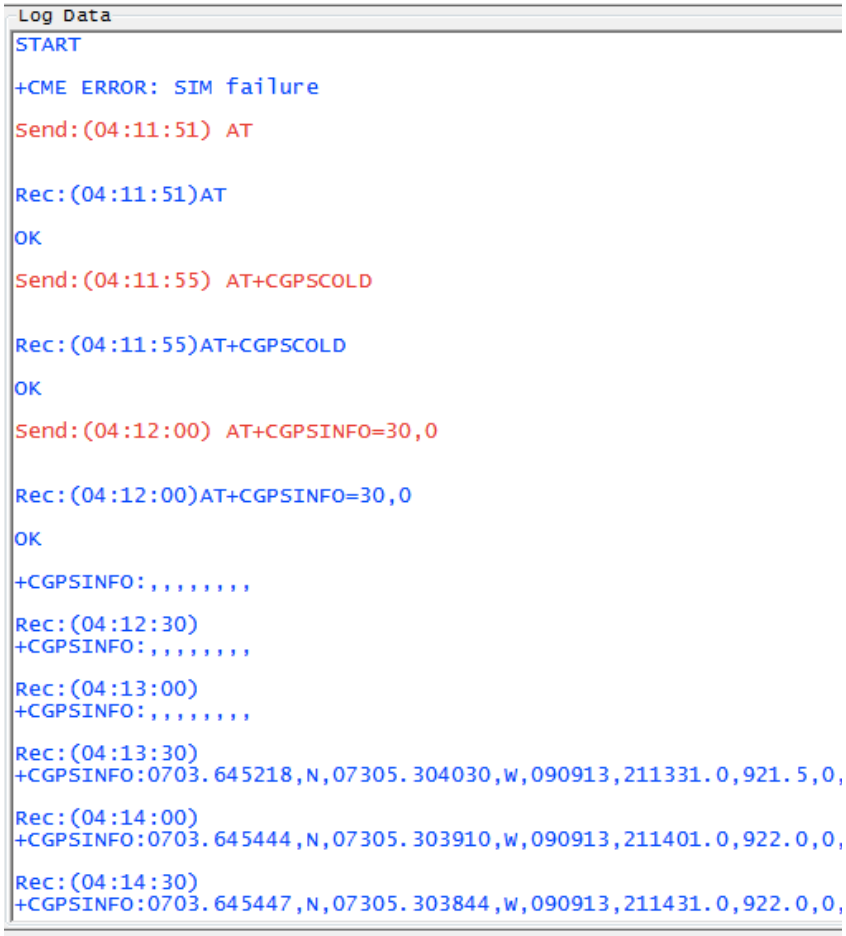


Figura 56. Adquisición de posición GPS mediante UART

4.1.1.29 INTEGRACIÓN FINAL DE LOS DRIVERS EN EL DISPOSITIVO MÉDICO

Finalmente se deben integrar los modulos UART en HDL en la descripción actual de hardware del dispositivo médico, y hacer la unión el software con sus respectivos drivers. Para esto se tuvieron que instanciar los módulos, compilar hardware, conectar las señales a la CPU2 que es la encarga de comunicaciones y otras tareas en el sistema multicore. Por lo tanto se estudio a fondo la transmisión de datos mediante protocolo websocket ademas de sus librerias descritas en lenguaje C++. El sistema cuenta con varias capas de software como se observó en la Figura 27. Los módulos de comunicaciones wi-fi, ethernet y de red celular se comunican únicamente con la capa en software de Websocket.

La capa de websocket como ya se mencionó tiene a su vez mas capas las cuales hacen uso de la librería de Ethernet para transmitir los datos de forma remota precisamente utilizando un protocolo homonimo a su nombre “Websocket”.

Para lograr incluir los drivers y que puedan transmitir datos de monitorización vital de forma transparente tal como lo hace el módulo Ethernet se estudio a profundidad cada una de las funciones de las capa baja de Websocket dentro del sistema asi como su máquina de estados.

La máquina de estados Websocket se encarga de comunicarse con el servidor mediante los diferentes drivers de comunicación (por el momento solo Ethernet) ejecutando una secuencia soportada por el servidor para abrir una sesión, autenticarse y transmitir los datos de monitorización vital. El esquema de la máquina de estados se puede observar en la Figura 57

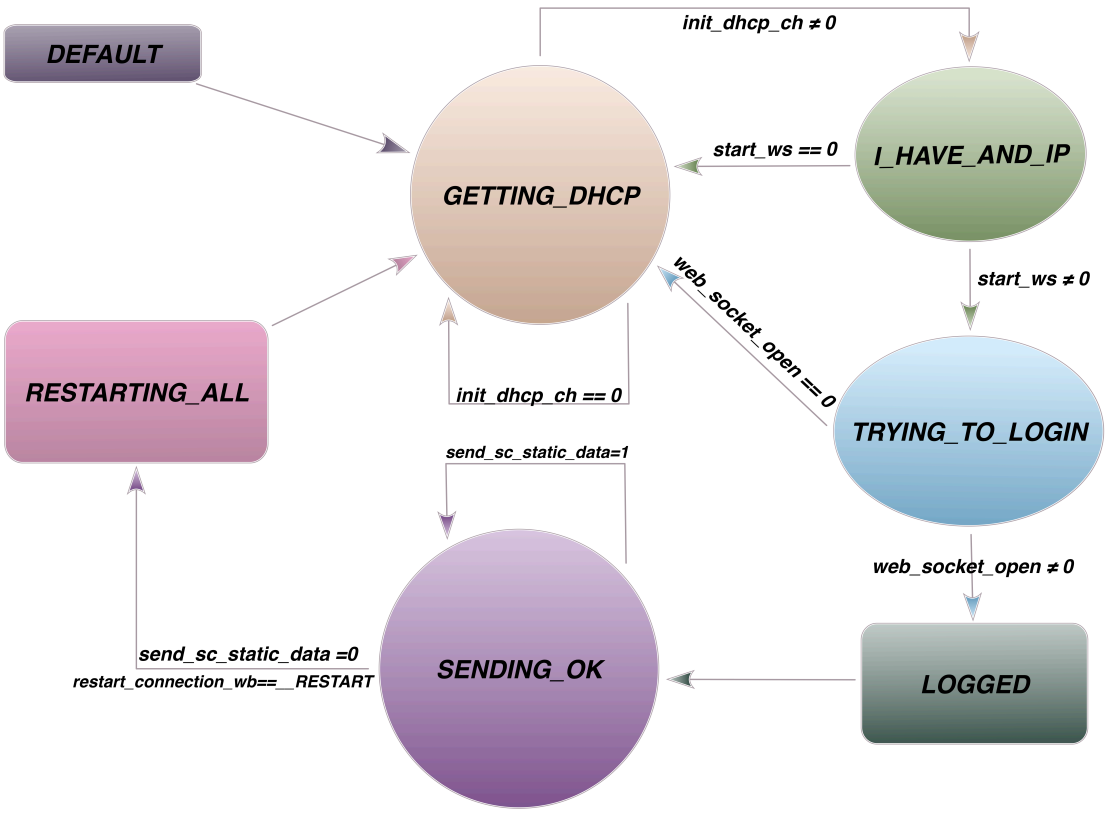


Figura 57. Diagrama de la máquina de estados Websocket

4.1.1.30 FUNCIONAMIENTO DEL PROTOCOLO WEBSOCKET

El protocolo websocket permite una conexión orientada a sockets TCP y es del tipo full-duplex en la comunicación entre el servidor y el cliente. Esta forma de comunicación añade la complejidad de transmisión bidireccional orientada a la comunicación web y la conexión arbitrada.

Los sistemas Websocket (Figura 6) son un paso en la evolución de la transmisión web para el uso en casi cualquier navegador y servidor web, este protocolo está siendo normalizado por el IETF (Interner Engineering Task Force).

A diferencia del http convencional la comunicación websocket establece una comunicación por medio de un hankshake o una petición al servidor para entablar comunicación una vez se ha conectado al socket por medio de cualquier puerto de comunicaciones. Este hankshake permite se encuentra ilustrado en la figura 7.

Una vez se realiza él enlace o se establece un canal de comunicaciones del tipo websocket cliente/servidor, se procede a hacer envió de datos para acceso a servicios o plugins que tenga disponible el servidor, estos datos deben hacerse utilizando Data Framing, este enmascaramiento de datos debe hacerse utilizando el estándar definido por el protocoló el cual se puede ver en la Tabla 8.

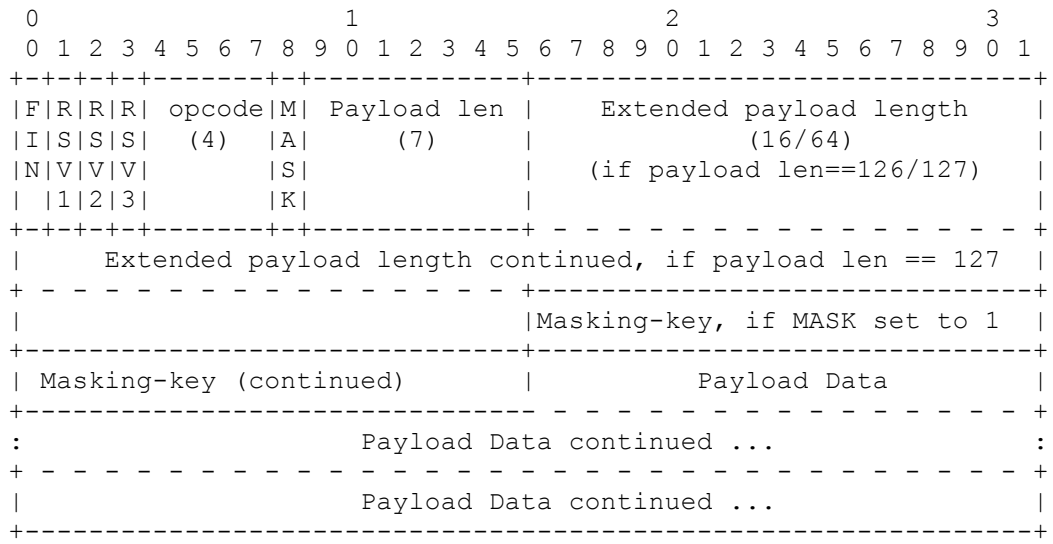


Tabla 8. Data framing Websocket [32]

La clase en C++ creada para soportar este protocolo tiene las librerias:

• Websocket.h

• Websocket.cpp

• Websocket_app.cpp

La máquina de estados utiliza funciones de “Websocket_app.cpp” y esta a su vez utiliza funciones de “Websocket.cpp”. Todas las capas pueden tener contacto con los drivers de los diferentes elementos de comunicación dependiendo del estado actual de la maquina de estados.

Para entende un poco mejor se ilustra en la Figura 58 las capas de abstracción del protocolo Websocket.

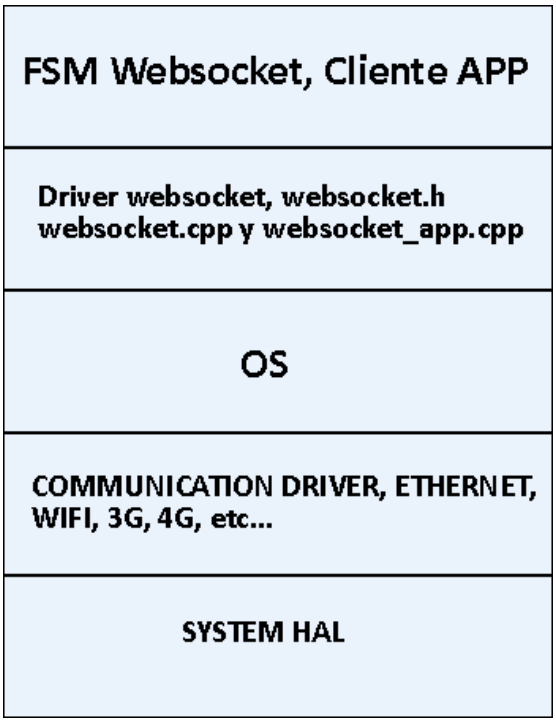


Figura 58. Capas de abstracción del sistema de comunicaciones basado en websocket.

Después del estudio de las funciones de la clase Websocket se procede con la abstracción de la máquina de estados para ejecutar de forma excluyente cada función de los módulos de comunicación teniendo en cuenta que solo se transmitirá por un solo módulo de comunicación al misma tiempo.

El estado GETTING_DHCP como su nombre lo indica (figura 57 diagrama estados) debe adquirir una IP para pasar al siguiente estado adicionalmente crea el objeto ws de la clase Websocket e inicializa sus valores con 2 contructores dependiendo de si el paciente tiene password ó no.

En el estado I_HAVE_AN_IP se efectuan dos procesos: el primero es abrir una conexión TCP con el servidor y el segundo hacer el Handshake para empezar a transmitir datos con empaquetado JSON y mediante protocolo Websocket el cual debe tener los encabezados propios del protocolo. Una vez se da la confirmación del handshake con la respuesta del server y un token único que se debe detectar se continúa con el siguiente estado.

TRYING_TO_LOGIN tambien tiene dos procesos: El primer proceso consiste en hacer Login (registro en la base de datos del servidor) el cual se ejecuta con el envío del mensaje Websocket solicitando el login, inmediatamente el servidor envía un mensaje de confirmación dependiendo de la solicitud. El segundo proceso solo se ejecuta si se recibe la confirmación de login es decir si se logró hacer login con el servidor. Este segundo proceso es llamado autenticación y de forma similar al Login se intercambian mensajes Websocket para determinar si la autenticación se pude realizar de forma satisfactoria. Se puede observar la

seguridad que se tiene en la conexión con el servidor para el envío de datos teniendo en cuenta que se debe hacer antes un Handshake para acceder a los servicios propios de la aplicación Web del servidor.

Finalmente el estado SENDING_OK envia los datos de monitorización vital para que sean visualizados en un navegador Web convencional mediante protocolo Websocket con el servidor.

Como se puede observar todos en todos los estados si no logran su propósito se llegará por defecto a GETTING_DHCP para volver a iniciar el proceso. Cada estado hace un número de reintentos pero si se cumple un tiempo máximo (timeout) se sale del estado devolviendo un valor que indica que no se logro hacer la rutina.

Es de resaltar que la máquina de estados Websocket es ejecutada como una tarea del sistema operativo mientras todo el sistema del dispositivo médico esta ejecutandos en otros hilos de procesamiento.

4.1.1.31 UNIÓN DE LOS DRIVERS DE COMUNICACIÓN CON LA CLASE WEBSOCKET

Ahora con los procesos que se ejecutan en la maquina de estados Websocket y su clase homonima en C++ que da soporte a los drivers de comunicación asi como el envío y recepción de datos, se procede a buscar una metodología para unir los drivers Wi-Fi y de transmisión de datos por Red celular con las diferentes capas de software Websocket.

Como paso final se creó una variable es la estructura Network llamada *phy_sel* la cual cambiará su valor dependiendo del módulo seleccionado de comunicación en la GUI del dispositivo medico. Es decir esta variable define que módulo de comunicación sera utilizado.

De esta forma se crean condicionales if y elseif los cuales permiten que dentro de cada función de toda la clase Websocketapp.cpp y websocket.cpp se puedan seleccionar las diferentes funciones de drivers de comunicación dependiendo del dispositivo seleccionado para la transmisión. Los otros módulos debéran permanecer apagados ó en modo de bajo consumo si no son activados por el usuario el cual solo tiene la posibilidad de activar un modulo de comunicación a la vez.

Entonces de forma general se crean los condicionales que eligen el driver a utilizar en cada uno de los métodos pertenecientes a la clase Websoscket en donde sean invocadas las funciones de comunicación. Por ejemplo si se encuentra la función abrir socket ó hacer DHCP en Ethernet se busca la función equivalente en los drivers de Wi-Fi y 3G para que sean ejecutadas en los condicionales excluyentes creados.

4.1.1.32 PRUEBA DE TRANSMISIÓN DE DATOS CON WEBSOCKET

Para probar el sistema se tuvo que hacer la simulación de los mensajes de la CPU1 la cual envia el nombre de la SSID, password, direción y puerto del servidor al cual apunta. Para el caso del módulo celular el nombre del APN y la dirección y el puerto del servidor. Esta simulación se realizo para demostrar el por que estas funciones aun no se encuentran integradas en la GUI y tampoco se contaba con un monitor para hacer las pruebas. Sin embargo el sistema dentro de la FPGA contaba de forma real con las dos CPUs corriendo bajo el sistema operativo y todo esto dentro de la board principal del dispositivo médico. Por esto se concluye que esta simulación de mensajes equivale a una prueba real con todo el hardware conectado.

Se obtuvieron buenos resultados al visualizar los datos enviados por Websocket en la interfaz grafica del servidor Web que permitia visualizar los datos enviados tanto del módulo Wi-Fi como el 3G.

Por cuestiones de confidencialidad no ser realizaron fotos del desarrollo hecho por Websocket.

5. APORTES AL CONOCIMIENTO

La UEE Bioingeniería ha señalado la necesidad de implementar equipos médicos sobre sistemas embebidos usando dispositivos lógicos programables tales como las FPGAs. Actualmente se encuentra en su fase final de desarrollo un dispositivo médico para monitorización de signos vitales portátil enfocado a entornos domiciliarios, salas de urgencia, entornos remotos, telemedicina, ambulancias, etc.

Este equipo en desarrollo también está provisto de tecnología para transmisión de datos de monitorización vital vía Ethernet, Wi-Fi y Red celular. Dado su tamaño se busca la optimización de su batería incrementando su autonomía usando hardware de bajo consumo acorde a los lineamientos de diseño de dispositivos médicos en la actualidad. Para esto fueron elegido módulos pequeños de última generación y de bajo consumo de corriente para dar cumplimiento a estos lineamientos técnicos de diseño.

Uno de los puntos fuertes de la FCV ha sido la implementación a nivel nacional de servicios de telemedicina los cuales están soportados bajo servidores que actualmente siguen brindando este servicio. Para implementar la monitorización remota de signos vitales para el dispositivo medico en desarrollo se decide crear un nuevo software en el servidor que recibe los datos del paciente que vaya acorde al uso actual de las tecnologías de la información (TIC's) y los nuevos estándares en la capa de aplicación para recepción de datos mediante protocolo IP. En respuesta a estas nuevas características se implementa este servicio mediante un protocolo llamado Websocket el cual esta soportado por los principales navegadores Web y es muy confiable debido a que funciona bajo conexiones cliente servidor por protocolo TCP. Este fue un valor agregado en la práctica dado que se optimizó el código existente corrigiendo errores y se adquirió el conocimiento de empaquetado de datos y transmisión mediante esta capa de aplicación.

Al momento de inicio de la práctica en la FCV el dispositivo médico no cuenta con los drivers necesarios para controlar los módulos Wi-Fi y de transmisión de datos por Red Celular, los cuales ya se encuentran dispuestos físicamente en el dispositivo médico en desarrollo. Dada esta necesidad se diseñaron e implementaron los diferentes drivers que permitieron el envío de datos de forma remota dentro del dispositivo médico hacia el servidor que brinda el servicio de recepción y visualización de estos datos.

El desarrollo en su primera fase fue investigativo dado que ya se contaba con el envió y recepción de datos mediante protocolo Ethernet. Durante el desarrollo se contó con el apoyo técnico de los ingenieros de la FCV en la resolución de algunas inquietudes en relación al sistema actual. Finalmente después de la prueba de los drivers se tuvo que integrar todo el desarrollo junto con los drivers en el dispositivo médico. Este último trabajo fue conjunto con los ingenieros de sistemas de la FCV debido a que las pruebas se realizaban apuntando al servidor ya desarrollado y en muchas ocasiones se realizaron ajustes a las librerías y

código ya hecho para mejorar errores en situaciones eventuales que no se tuvieron en cuenta y el practicante modificó para una mejor implementación final en todo el sistema.

En dado caso de necesitar documentación para mejorar los drivers existentes o agregar funcionalidades nuevas que en el futuro se requieran. Adionalmente queda la debida documentación e informes de los drivers realizados junto al modulo en HDL para transmisión de datos por UART que también se desarrolló contrastando el desempeño a altas velocidades de los módulos UART que se encontraban en el monitor. El módulo UART también podrá ser usado en desarrollos futuros ó como mejora al sistema actual para una mayor fiabilidad en la transmisión de datos por este protocolo.

6 RECOMENDACIONES A LA EMPRESA

- Se debe implementar mediante una tarea del sistema operativo la adquisición de las coordenadas de Georeferenciación del módulo GPS. Esta tarea puede hacer uso del driver para el módulo integrado 3G-GPS implementado por el practicante. De esta forma se podría brindar soporte a los requerimientos de Georeferenciación que solicita el servidor.

- Para mejorar el sistema de transmisión mediante Websocket sería ideal en un futuro cambiar el formato de transmisión de los datos numéricos entregados por la tarjeta de adquisición de parámetros vitales por un formato de compresión de datos que libere la carga de datos e incremente la velocidad y desempeño del sistema sobre todo en casos donde la velocidad de la red que brinda acceso a internet sea limitada.

- Dado que los módulos Wi-Fi y 3G tienen buffers de envió de 1.4Kbyes y 1Kbyte respectivamente, se debe rediseñar el paquete de envió de parámetros vitales debido a que el actual implementado con el módulo Ethernet es de 3Kbytes por lo cual estos módulos no podrían enviarlo en un solo paquete. Esto debe ser rediseñado en la librería Websocket.

- Para el envío de datos en el módulo 3G sería conveniente implementar el modo transparente de transmisión de datos por TCP el cual no necesita solicitud y autorización para el envió de un paquete de datos sino que se gestiona el envío y recepción en modo datos por UART de forma similar al módulo Wi-Fi.

- Así como se realizó el diseño de una metodología para actualizar el firmware del módulo Wi-Fi se debería realizar una metodología para actualizar el firmware del módulo de transmisión de datos por red celular. Esta actualización fue realizada por el practicante y requiere de un computador, un software, drivers y el módulo 3G dentro del módulo de entrenamiento.

- Se debe diseñar una metodología teniendo en cuenta la hoja de datos del módulo Wi-Fi y el driver realizado para detectar una posible desconexión con el AP para que se pueda dar una retroalimentación visual al usuario mediante la GUI.

- Para la FCV sería de gran utilidad adquirir los equipos necesarios para poder soldar circuitos integrados de empaquetado QFN los cuales hoy día son comunes y en el caso del módulo Wi-Fi debe ser soldado a la board del dispositivo médico por otras empresas que brindan este servicio. A futuro se debería poder ensamblar en su totalidad el dispositivo médico en las instalaciones de la FCV así como se producen todos los dispositivos que ya están en el mercado.

- Dado que no fue posible soldar el módulo Wi-Fi directamente en la PCB del dispositivo medico, hizo falta la prueba dentro de esta board para verificar su

funcionamiento en la transmisión de datos y también en su modo sleep (bajo consumo) el cual tiene un arreglo especial en la board para su funcionamiento.

- Las pruebas hechas tanto en la etapa de pruebas del driver como en la transmisión de datos por protocolo Websocket se hizo mediante una red local creada por un router lo cual no evidencia una transmisión de datos por internet hasta un servidor real desde un punto remoto. Se recomienda se hagan estas pruebas de transmisión desde una IP pública al servidor real para verificar el funcionamiento y la latencia generada en la transmisión. Esta prueba es muy importante dado que este escenario de transmisión remota es en el que se va usar el dispositivo normalmente en un futuro.

- Se debe incluir en el manual de usuario del dispositivo médico la advertencia de no enlazar el módulo Wi-Fi con Access points que tengan espacios en su SSID (por ejemplo “my Access point”) . Para que estas redes puedan ser accesibles y se puedan conectar con el modulo Wi-Fi según la hoja de datos del modulo Wi-Fi se debe reemplazar cada espacio con el carácter ASCII ‘$’. Se recomienda a futuro modificar el driver si se quiere brindar soporte a redes con ese tipo de SSID.

- Implementar una función en el driver Wi-Fi que haga uso de la función “Scan” (la cual muestra las redes inalámbricas disponibles) y pueda guardar en una estructura estas redes y puedan ser visualizadas en la interfaz gráfica del dispositivo médico por ejemplo mediante un combo-box.

- Se podría en un futuro implementar la actualización de firmware automática para el módulo Wi-Fi agregando una función al driver que ejecute todos los pasos documentados en el informe hecho por el practicante para lograr la actualización de su firmware fácilmente mediante comandos. Esta actualización sería útil para el departamento de soporte técnico facilitando esta tarea para ellos ó también podría el módulo actualizarse automáticamente cada cierto tiempo de funcionamiento.

- A pesar de que los módulos UART en HDL la gran mayoría de veces funcionan bien a 460800 baudios se recomienda siempre configurarlos en los drivers a 115200 baudios. En algunas ocasiones por razones de sincronización llegan bytes corruptos a altas velocidades. A 115200 baudios no se presentaron estos errores y la velocidad de transmisión es eficiente para el sistema.

7 CONCLUSIONES

La implementación del sistema operativo para microncontroladores y procesadores uC/OSII en el dispositivo médico hace que se puedan ejecutar de forma eficiente las diferentes y numerosas tareas presentes en el sistema.

La forma en que se multiplexaron los diferentes módulos de comunicación en la librería Websocket permite al sistema ahorrar energía al inhabilitar los módulos no usados y únicamente utilizar el hardware necesario para cada aplicación.

Los sistemas multicore en FPGAs gestionados bajo sistemas operativos de tiempo real permiten la optimización de procesos al trabajar varios hilos de procesamiento independientes al mismo tiempo.

El uso de estructuras para almacenar datos que serán accedidos por varios hilos de procesamiento del sistema es una buena práctica de programación dado que son accesibles desde cualquier capa de software.

El envío de datos mediante protocolo TCP soportado por el estándar Websocket garantiza una transmisión confiable y libre de errores condición requerida para la visualización adecuada de los parámetros vitales de forma remota.

El uso de la implementación jWEBSOCKET del protocolo Websocket es muy útil al permitir hacer plugins en software de forma fácil y transparente al programador además de permitir el envió y recepción de datos a gran velocidad en plataformas soportadas por la mayoría de navegadores Web. Esta última característica permite implementar la aplicación en cualquier navegador Web de un computador convencional ó en dispositivos móviles como tablets y Smartphones.

El envío de datos mediante el formato JSON permite de una forma fácil y segura de ingresar variables y parámetros de uso privativo en los mensajes que se transmiten por la red dada su simplicidad de uso y comprensión por cualquier persona y su amplio soporte para los lenguajes de programación mas utilizados.

La implementación de login y autenticación en el proceso de conexión con el servidor Websocket, además de la clave de seguridad implementada en el Handshake proporciona al sistema altos estándares de seguridad evitando que intrusos y personas ajenas a la FCV intenten acceder a sesiones privadas ó congestionen la red.

Los módulos de transmisión inalámbrica elegidos para el dispositivo médico de monitorización vital cumplen con los lineamientos de diseño que exigen los dispositivos médicos en la actualidad basados en sistemas embebidos tales como bajo consumo de potencia, altas velocidades para aproximar la transmisión de datos a “tiempo real”, protocolos de seguridad bajo estadandares internacionales de conectividad inalámbrica y de uso intuitivo para el usuario.

El uso de aceleradores de Hardware en sistemas multicore favorece la velocidad de ejecución de las tareas y disminuyen carga a los procesadores y el sistema operativo al ejecutar de forma paralela las diferentes tareas como por ejemplo el envió y recepción de datos por protocolo UART. De esta forma se optimizan recursos y se aporta velocidad en el sistema sin generar un mayor consumo en recursos de procesamiento. Mejorando tiempos en la ejecución de las tareas.

El uso de sistemas operativos en sistemas microprocesados es de vital importancia en sistemas complejos como monitores de signos vitales al asignarle una mejor distribución de los tiempos en las tareas a ejecutar en cada procesador.

La portabilidad y amplio uso del lenguaje C permiten interactuar de forma más eficiente y directa, con librerías y componentes del sistema operativo la ejecución de diferentes tareas como video, adquisición, alarmas, transmisión de datos, etc.

La práctica desarrollada en la UEN Bioingeniería de la FCV permitió participar de forma activa en la industria de desarrollo de equipos electrónicos, así como familiarizarse con los procesos y conductos regulares en una empresa, lo cual facilitara la integración en el mercado laboral.

BIBLIOGRAFIA

[1] Organigrama FCV. [En línea] <http://www.fcv.org/corp/index.php?option=com_content&view=article&id=47&Itemid=57>. Consultado el 20 de diciembre de 2012.

[2] OEM Developer's Manual [doc.]. Pag 32, Figura C. Consultado el 2 de Febrero de 2013.

[3]Class 05-Protocols.[En linea].http://www.csd.uwo.ca/courses/CS1011b/slides/c05Protocols_1spp.pdf. Consultado 15 de marzo de 2013

[4] Wiznet W5100 datasheet [pdf]. Pag 3, fig 1.Consultado el 16 de abril de 2013.

[5] Redes de Computadoras Introducción Arquitectura de Redes. [En linea]. <http://webdelprofesor.ula.ve/ingenieria/gilberto/redes/04_conceptosBasicos2.pdf> . Consultado el 24 de marzo de 2013.

[6] Communication topics. [En linea] <http://slid.es/mos/communication_topics> Consultado el 30 de abril de 2013.

[7] HTML5 programming. [en linea]. <http://apress.jensimmons.com/v5/pro-html5-programming/ch7.html>. Consultado el 4 de mayo de 2013.

[8] JavaScript JSON. [En linea]. < http://www.json.org/>. Consultado el 8 de mayo de 2013.

[9]Logic Element [en línea] <http://wwwdsa.uqac.ca/~daudet/Cours/Vlsi/DOCUMENTS/repertoire435/Livre-MJS-Smith/Book/CH05/CH05-6.gif>. Consultado el 13 de mayo de 2013.

[10] CCNA EXPLORATION COURSE: Aspectos basicos del Networking. Version 4.0. Atlacomulco: Pearson Educación de Mexico, 2011. ISBN 978-607-32-0380-7

[11] CCNA EXPLORATION COURSE: Networking para el hogar y pequeñas empresas. Version 4.0. Atlacomulco: Pearson Educación de Mexico, 2011. ISBN 978-607-32-0383-8

[12] STALLINGS, William. Comunicaciones y redes de computadoras. Septima Edición. Madrid: Pearson Education. 2004. ISBN 978-84-205-4110-5

[13] GPS SATALLITE SIGNALS. [En linea] <http://www.gmat.unsw.edu. au/snap/gps/gps_survey/chap3/312.htm.>. Consultado el 24 de mayo de 2013.

[14] WHAT IS WEBSOCKET?. [En linea].< http://www.websocket.org /index.html>. Consultado el 29 de mayo de 2013.

[15] jWebsocket - the open source solution for realtime web developers. [En linea].<https://code.google.com/p/jwebsocket/>. Consultado el 26 de mayo de 2013.

[16]The web socket protocol - RFC 6455. [En linea] http://tools.ietf.org/html/rfc6455# section-1.3. Consultado el 27 de mayo de 2013.

[17] Soft CPU Cores for FPGA. [En linea]. http://www.1-core.com/library/digital/soft-cpu-cores/. Consultado el 30 de mayo de 2013.

[18] NIOS II Processor: The world's most versatile Embedded Processor. [En linea]. <http://www.altera.com/devices/processor/nios2/ni2-index.html>. Consultado el 3 de junio de 2013.

[20]uC/OSII - The real time kernel. [En linea].<http://micrium.com/rtos/ucosii/overview />. Consultado el 10 de junio de 2013.

[21]SPI TUTORIAL. [En linea] .<http://www.corelis.com /education/SPI_Tutorial.htm>. Consultado el 13 de junio de 2013.

[22] GPIO PINS. [En linea]. <http://linuxtv.org/wiki/index.php/GPIO_pins>. Consultado el 15 de junio de 2013.

[23] Redes de computadoras. [En linea]. <http://webdelprofesor.ula.ve/ingenieria/ gilberto/redes/>. Consultado el 20 de junio de 2013.

[24] Socket programming.[En linea]. <http://www.buyya.com/java/Chapter13.pdf>. Consultado el 25 de junio de 2013.

[25] Fundacion cardiovascular de colombia. Documentacion: informe de resultados monitor Home care y gamma alta. Version 2. R-DDBIO-05 . Bucaramanga. 2011. 20p.

[26] Wiznet W5100 datasheet [pdf]. pag 7, fig 3. Consultado el 16 de abril de 2013.

[27] TRAINING WIGFLY [pdf]. Consultado el 25 de febrero de 2013.

[28] ROVING NETWORK. [En linea]. < http://www.rovingnetworks.com/FAQs/ I_get_ FTP _Error _530_Login_Authentication_failed_when_I_try_to_ upgrade _my_WiFly_module.>. Consultado el 15 de julio de 2013.

[29] SIM52xx_HW_V1.0 [pdf]. <http://wm.sim.com/upfile/2012125162033f.pdf> Consultado el 15 de Mayo de 2013.

[30] Cooking Hacks – New 3G [En linea]. <http://www.cooking-hacks.com/index.php/documentation/tutorials/arduino-3g-gprs-gsm-gps?utm_source=banner_3g_shield&utm_medium=banner> Consultado el 15 de Mayo de 2013.

[31] SIM52xx_GPS_Assist_Location_Application_Note_V1.0 [pdf]. <http://wm.sim.com/upfile/2012417141347f.pdf> Consultado el 15 de Mayo de 2013.

[32] The Websocket Protocol [En linea]. <http://tools.ietf.org/html/rfc6455 > Consultado 30 de Mayo de 2013.

ANEXOS

REALIZACIÓN DE LA COTIZACIÓN DE LAS DIRECCIONES MAC PARA EL DISPOSITIVO ETHERNET – WIZNET 5100

Se hizo la cotización de las direcciones MAC requeridas para la utilización del dispositivo Ethernet implementado en el monitor de signos vitales signcare como un dispositivo comercial. Esta tarea se realizó como parte de la depuración final del presupuesto requerido para la realización del monitor incluyendo licencias y demás gastos de última hora.

La IEEE es la única entidad autorizada para la distribución, comercialización de estas direcciones. Las direcciones MAC son las direcciones fijas que tiene cada dispositivo que se quiera conectar a la red alrededor del mundo como medio único de identificación. Se compone de 48 bits, normalmente dados en notación hexadecimal (8 dígitos en notación hexadecimal).

Por lo general los dispositivos ya vienen con una dirección MAC asignada por el fabricante del dispositivo. Por ejemplo los dispositivos Wi-Fi y 3G implementados en el dispositivo médico si tienen direcciones MAC asignadas de fabrica (están direcciones fueron compradas por el fabricante a la IEEE). Sin embargo los fabricantes de módulos Ethernet no la dejan fija de fabrica de modo que cada usuario puede colocar la dirección MAC que desee pero al momento de hacer un producto comercial tienen que comprarse.

La IEEE asigna determina un valor mínimo de 630USD por 4096 direcciones MAC (12 bits). Este valor fue dado como respuesta a una cotización para la cual se les hizo el requerimiento y se determino que esa cantidad es suficiente para los dispositivos médicos calculados para fabricar en un futuro. La figura 59 fue tomada de un *screenshot* en donde se muestra el valor de la dirección en la categoría que aplicaría la Fundación Cardiovascular de Colombia.

INDIVIDUAL ADDRESS BLOCK (IAB)

An IAB is for people who need less than 4097 unique 48-bit numbers (EUI-48) and thus find it hard to justify buying their own OUI. It is a particular OUI concatenated with 12 additional IEEE-provided bits, leaving only 12 bits for the owners to assign to their (up to 4096) individual devices.

Unlike an OUI, which allows the assignee to assign values in various different number spaces (for example, EUI-48, EUI-64, and the various CDI number spaces), the IAB can only be used to assign EUI-48 identifiers.

The Individual Address Block (IAB) can be used in conjunction with a number of standards. It does not limit your right to use your assignment for multiple purposes.

Registration Fees

Products	Fees	Total Due
Publicly Registered IAB (company name & address on the public listing)	US $645	US $645
Privately Registered IAB (company name and address NOT on the public listing)	US $645 + $1,130 (privacy fee addition)	US $1,775
Yearly Confidentiality Renewal Fee* (for privately registered assignments only)	US $1,130	US $1,130

Figura 59. Valor de 4096 direcciones MAC para distinto tipo de compañias.

Al ser compradores de direcciones cualquier empresa es añadida a una lista en donde se puede ver la lista de direcciones MAC asignadas para cada una. Si se requiere privacidad y que la empresa no sea incluida en esta lista se debe pagar un valor de 1300USD anuales. Se consulto acerca de este tipo de confidencialidad en la fcv pero esto no aplico, por lo tanto se dejo como valor final 630USD por las 4096 direcciones.

REVISIÓN DEL ARTÍCULO “DISPOSITIVOS MÉDICOS CON ARQUITECTURA OPTIMIZADA” por el Ing. José Pablo Pinilla.

Se hizo la petición a todo el equipo de diseño y desarrollo de revisar la primera versión del articulo “Dispositivos médicos con arquitectura optimizada” como forma de depurar este articulo destinado a publicarse en revistas de ámbito científico Biomédico a nivel nacional e internacional. El practicante al ser parte de este equipo reviso este escrito dándole sugerencias al ingeniero de cambiar algunas palabras y enfoque en ciertos apartados del articulo.

El artículo se fundamenta en los principales criterios que un dispositivo biomédico hoy día debería tener respondiendo a los actuales requerimientos en este ámbito tales como: respuesta en tiempo real, bajo consumo de potencia, conectividad, seguridad y una experiencia intuitiva para el usuario. Cada uno de estos parámetros se refleja en la experiencia del usuario final de una forma notoria.

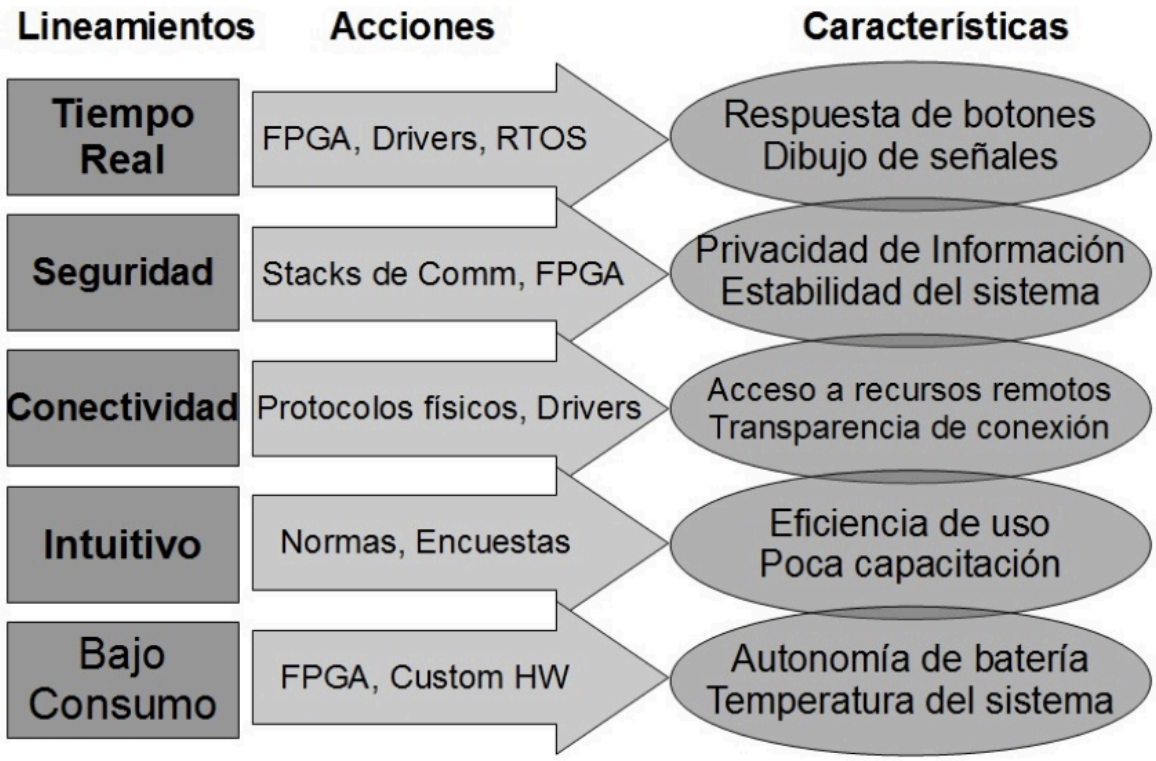


Figura 60. Lineamientos y acciones características de los dispositivos médicos desarrollados.

Gracias a este articulo el practicante logro comprender la importancia de implementar un sistema operativo que haga la gestión de recursos en un sistema embebido debido a la cantidad de tareas que se deben manejar al mismo tiempo. Adicionalmente se comprenden los criterios de selección que se tuvieron en cuenta en la elección de los dispositivos que dan conectividad a la red (bajo consumo de potencia y modo sleep en el modulo wifly y bluetooth).

Adicionalmente se evidencia porque es adecuado tener determinados núcleos encargados de tareas especificas por ejemplo un core puede ocuparse de la adquisición de señales y gestión de ventanas y otro puede ocuparse de las comunicaciones y demás hardware externo.