Contiki NG con NB-IoT

Autores: Rosina D'Eboli, Rodrigo García, Josefina Schmitd

Emails: rosina.d@fing.edu.uy
rodrigo.garcia.ordeig@fing.edu.uy
maria.josefina.schmitd@fing.edu.uy

Tutor: Mariana Siniscalchi

Email: msiniscalchi@fing.edu.uy

Instituto de Ingeniería Eléctrica - Facultad de Ingeniería - UDELAR

Resumen

Se presenta el diseño, implementación y pruebas de un prototipo de nodo cliente UDP compuesto por el launchpad LAUNCHXL-CC2650, la placa LTE 2 IOT CLICK y una antena. Se basa en la obtención de información de un sensor de botón que luego será enviada a un servidor UDP. El trabajo se desarrolla en base una solución implementada para la comunicación MQTT con la utilización de protothreads, esta se adaptará a protocolo UDP incorporando también herramientas propias del sistema operativo Contiki-NG, en específico los procesos y los eventos. Para la conexión de red se hará uso de la tecnología NB-IoT.

Índice

1. Introducción **3**
1.1. Descripción del problema . . . 3

2. Marco Teórico y antecedentes **3**
2.1. Tecnología NB-IoT . . . 3
2.2. Protocolo UDP . . . 3
2.3. Biblioteca de ELIoT . . . 4
2.3.1. Identificación de respuestas a comandos AT . . . 4
2.3.2. Envío de comandos AT . . . 5

3. Objetivos **6**

4. Descripción del sistema **6**
4.1. Hardware . . . 6
4.2. Comandos AT . . . 6
4.3. Software . . . 9
4.3.1. Compilación . . . 9
4.3.2. Implementación . . . 9

5. Tests **10**
5.1. base-demo.c . . . 10
5.2. test-echo.c . . . 10
5.3. test-redirect.c . . . 10
5.4. test-boton.c . . . 11
5.5. main-quectel.c . . . 11

6. Conclusiones **12**

7. Anexo **13**
7.1. Planificación propuesta para Especificación Breve . . . 13

Bibliografía **14**

1. Introducción

1.1. Descripción del problema

El problema de este proyecto consiste en el manejo de un módulo de comunicación LTE mediante un microcontrolador. Este incluye un sensor del cual se quieren recibir datos para luego ser transmitidos como mensajes UDP, a través de una red con tecnología NB-IoT. El hardware a utilizar se compone del launchpad LAUNCHXL-CC2650 de Texas Instruments [1] y la placa LTE IOT 2 CLICK de MikroElectronica [2] que se compone del módulo de comunicación BG96 de Quectel. El módulo a utilizar es configurable mediante comandos AT. Además, se hará uso de las herramientas brindadas por el sistema operativo Contiki-NG [3] para llevar a cabo esta tarea. Contiki-NG es un sistema operativo de código abierto multiplataforma para dispositivos IoT (Internet of Things) Next-Generation. El mismo ofrece conectividad a internet para dispositivos de bajo costo y bajo consumo, requiriendo únicamente alrededor de 100 kbytes de código y bajo uso de memoria RAM.

2. Marco Teórico y antecedentes

2.1. Tecnología NB-IoT

NB-IoT (Narrow Band - IoT) [4] [5] es una tecnología que surge como solución a la necesidad de estandarizar las comunicaciones entre dispositivos IoT, que requieren de una latencia de transmisión baja y poco tráfico de datos. La misma es una red LPWAN (Low Power Wide Area Network) que evoluciona del LTE, siendo compatible con sus bandas y caracterizándose por: su bajo consumo de energía, bajo costo, ofrecimiento de conectividad fiable, disponibilidad de acceso para altas cantidades de dispositivos, cobertura de largo alcance, y utilización de anchos de banda limitados al orden de los 200 kHz.

2.2. Protocolo UDP

UDP (User Datagram Protocol) es un protocolo de capa de transporte basado en el protocolo de capa de red IP (Internet Protocol). Este es no orientado a conexión y no confiable, es decir, no ofrece ninguna garantía de entrega de datagrama o de protección de duplicación. Puesto que no es necesario establecer conexión con el receptor ni espera de respuesta, la información se envía rápidamente. La comunicación es del tipo cliente-servidor, en esta arquitectura varios clientes se conectan a un servidor y envían mensajes, este servidor responderá según las solicitudes de los clientes. Para establecer la conexión el cliente deberá especificar la dirección IP y puerto de destino del servidor. El proceso de cliente selecciona al azar un número de puerto del rango dinámico de números de puerto y lo utiliza como puerto de origen para la comunicación. El puerto de destino por lo general será el número de puerto bien conocido o registrado asignado al proceso del servidor.

Habitualmente UDP se utiliza en aplicaciones donde la confiabilidad no es crítica y en comunicaciones sujetas a limitación temporal. Se usa principalmente para consultas DNS, conexiones VPN y streamings de audio y video.

2.3. Biblioteca de ELIoT

[1] La implementación realizada para la configuración del módulo de comunicación es un problema ya abordado por el ayudante del IIE Andrés Seré (en adelante, Solución de Andrés). Su implementación se basó en el uso de un microcontrolador de la familia MSP432 y un módulo de comunicación BG96 de Quectel, utilizando algunas de las utilidades básicas de Contiki-NG.

2.3.1. Identificación de respuestas a comandos AT

Para poder recibir e interpretar las respuestas a comandos y las URCs (Unsolicited Result Code) del módulo de comunicación BG96 de Quectel se hizo uso de un intérprete de comandos basado en la estructura de datos tipo árbol y máquinas de estado. Esta solución fue implementada por el Ing. Leandro Francucci [6], a partir del cual se desarrollara el proyecto.

La implementación se basa en la clasificación de respuestas conocidas en nodos para formar una estructura de datos tipo árbol. El algoritmo de búsqueda de patrones es un mecanismo en el cual la misma se realiza a medida que se ingresan los caracteres desde el módulo de comunicaciones, consiste en comparar el caracter recibido con los esperados por el nodo actual, y determinar así el siguiente nodo. Mientras el caracter recibido pertenezca a los esperado por el nodo, la búsqueda continúa, caso contrario vuelve al nodo raíz. A continuación, se muestra un ejemplo de árbol con diferentes respuestas de un módulo GSM (Global System for Mobile).

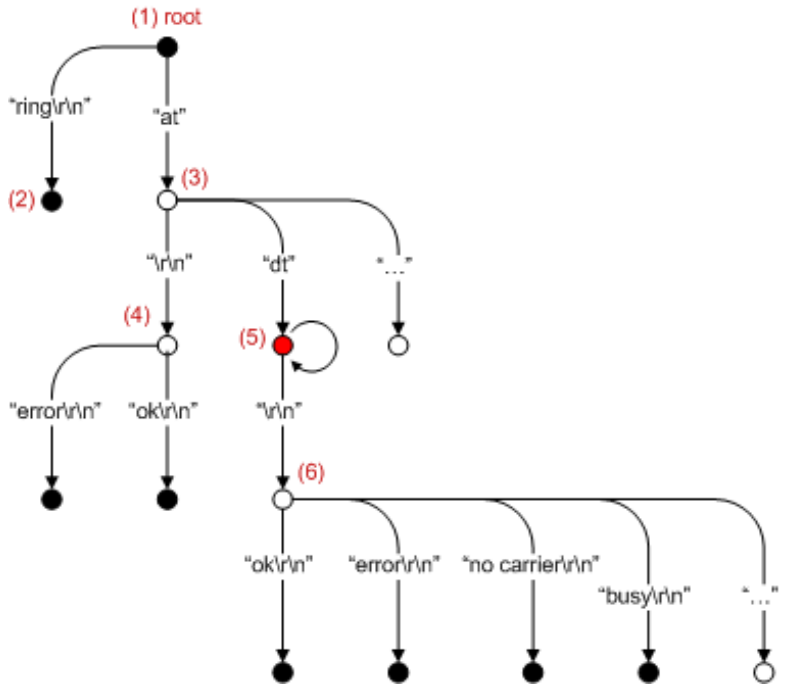

Figura 1: Ejemplo de árbol de recepción [6]

En cada rama se representa un patrón, que comienza desde un nodo raíz y se extiende hasta un nodo hoja, coloreados en negro. Además, se puede notar un nodo coloreado con rojo (también denominado nodo transparente), este cuenta con la característica de permanecer detenido hasta que arribe el caracter esperado. En la fig. 2 se modela el funcionamiento del algoritmo mediante una máquina de estados. Se observan dos estados posibles, 'Idle' e 'InSearch'.

[1]Biblioteca desarrollada por Andrés Seré en el contexto del convenio de la IM para la implementación de un controlador de luminaria inteligente

- En estado 'Idle' (en el cual se inicializa la máquina de estados), se recibe un primer caracter y se verifica si pertenece a alguno de los patrones asociados al nodo raíz con la función searchPatt(). Si c pertenece a uno de los patrones, se verifica si la búsqueda está completa, en ese caso vuelve a la raíz, de lo contrario continúa la búsqueda, pasando al estado InSearch. Si no encuentra coincidencia, la búsqueda se descarta, llamando a notMatch().
- En el estado 'InSearch' se procesa cada caracter recibido y se verifica si pertenece a algún patrón con la función isInPatt(), adicionalmente, se invoca una función callback del nodo actual y se le pasa como parámetro el caracter c mediante deliver(). Como en el estado antes explicado, las funciones isMatch() y notMatch() se utilizan para comprobar si la búsqueda está completa o descartarla si no hay coincidencia. En el caso de que la búsqueda no esté completa, se establece el próximo carácter a esperar con incPatt(). Si el caracter c es igual al anterior se mantiene la búsqueda llamando a isEqual().

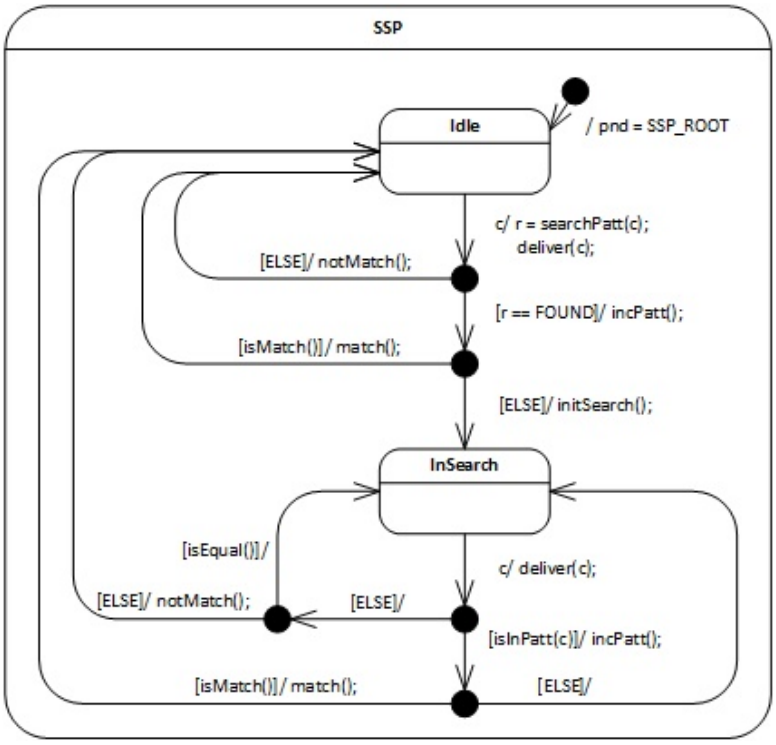

Figura 2: Máquina de estados de algoritmo de búsqueda [6].

2.3.2. Envío de comandos AT

La biblioteca ELIoT se basa únicamente en el uso de protothreads y timers del sistema operativo Contiki-NG. Para poder realizar el envío de comandos, se implementa un protothread principal en el que se invocan protothreads 'hijos' que realizan diferentes configuraciones según el estado en el que se encuentre el programa. Su código se basa en el uso de la tecnología NB-IoT para la conexión a la red celular y el uso del protocolo de comunicación MQTT para el envío de información de distintos sensores a diferentes tópicos. En este proyecto se desea abstraer la lógica de su solución y uso del árbol de nodos para la interpretación de comandos haciendo uso de procesos y posteo de eventos. Se utilizará la tecnología NB-IoT pero en este caso el envío será por UDP.

3. Objetivos

- Comprender el funcionamiento del sistema operativo Contiki-NG.
- Implementar un modulo de interprete de comandos utilizando Contiki-NG.
- Adquirir información del accionado del sensor de los botones del launchpad LAUNCHXL-CC2650.
- Implementar la transmisión de mensajes UDP utilizando el módulo de comunicación LTE.

4. Descripción del sistema

4.1. Hardware

El sistema completo se compone del launchpad LAUNCHXL-CC2650, un botón, la placa LTE IOT 2 CLICK y una antena. El launchpad cuenta con el system on a chip CC2650 y el botón que se utiliza en la aplicación, adicionalmente, se hará uso del periférico UART para la comunicación con la placa LTE IOT 2 CLICK. En la tabla 1 se muestran los pines utilizados de cada componente de hardware para la utilización del sistema. Los pines TXD(RX) y RXD(TX) son los destinados a la comunicación UART mientras que el pin DIO15(PWK) reinicia el módulo BG96 mediante un pulso a nivel alto.

Cuadro 1: Conexiones entre componentes.

Pines del LAUNCHXL-CC2650	**Pines del LTE IOT 2 CLICK**
DIO2(TXD)	RX
DIO3(RXD)	TX
DIO15	PWK
5V	5V
3.3V	3.3V
GND	GND

4.2. Comandos AT

El conjunto de "comandos AT" o "comandos Haye" [7] es un lenguaje creado por la compañía *Hayes Communications*. Estos fueron creados con la finalidad de estandarizar el envío de comandos para la configuración y parametrización de modems. En particular, la sigla "AT" refiere a "Attention". Hoy en día la lista de comandos mantiene el formato inicial ("AT+COMANDO") pero el contenido varía según lo fabricantes y los modelos.

Para el módulo BG96 se implementaron los siguientes comandos:

Para el registro en la red

- **AT+CPSMS=0:** Para deshabilitar el Power Save Mode.
- **ATE0:** Para deshabilitar la echo.
- **AT+QGMR:** Solicita versión del modem y del firmware.

- **AT+CIMI:** Para solicitar el IMSI (International Mobile Subscriber Identity) para identificar la SIM.
- **AT+CPIN?:** Para solicitar una cadena de caracteres alfanumérica que indica si se requiere de una contraseña o verificar si quedó correctamente configurado.
- **AT+CPIN:** Para ingresar el PIN.
- **AT+CFUN=4:** Para deshabilitar la transmisión y recepción de señales RF.
- **AT+QCFG="newscanmode",3,1:** Para habilitar que el modem solo busque conexiones LTE y que actúe de inmediato.
- **AT+QCFG=ïotopmode",1,1:** Para especificar conexión a redes LTE Cat NB1 y que actúe de inmediato.
- **AT+CGDCONT:** Para configuración del contexto PDP (Packet Data Protocol) e ingresar el APN de la red NB de ANTEL.
- **AT+QCFG="band":** Para especificar las bandas LTE permitidas.
- **AT+QCFG="servicedomain":** Para especificar el servicio de dominio registrado.
- **AT+CFUN=1:** Para reestablecer funcionalidades del modem.
- **AT+COPS:** Para seleccionar el operador. En este caso ANTEL.
- **AT+CSQ:** Para preguntar la intensidad de la señal recibida.

Para comenzar un servicio UDP:

En la fig. 3 se puede observar el flujo de acciones a seguir para establecer un servicio UDP luego de registrar el módulo en la red. El mismo se basa en el diagrama de flujos para el servicio TCP que se muestra en el manual de comandos del módulo BG96 [8] en la página 7.

En particular, se implementó el flujo de tal forma de comenzar un servicio de cliente UDP. A continuación se listan los comandos involucrados en cada una de las etapas:

Configurar contexto PDP[2]:

- **AT+QICSGP:** Utilizado para configurar los parámetros de la conexión como lo son el tipo de protocolo IP y el APN. También se configura el nombre de usuario, contraseña y el tipo de autenticación.

Activar/Desactivar contexto PDP:

- **AT+QIACT:** El módulo soporta hasta 3 contextos activados juntos. Con este comando se activa el contexto.
- **AT+QIACT?:** Este comando se utiliza para consultar el estado del contexto (activado o desactivado), el tipo de protocolo IP asociado y la IP local luego de la activación.
- **AT+QIDEACT:** Con este comando se desactiva el contexto.

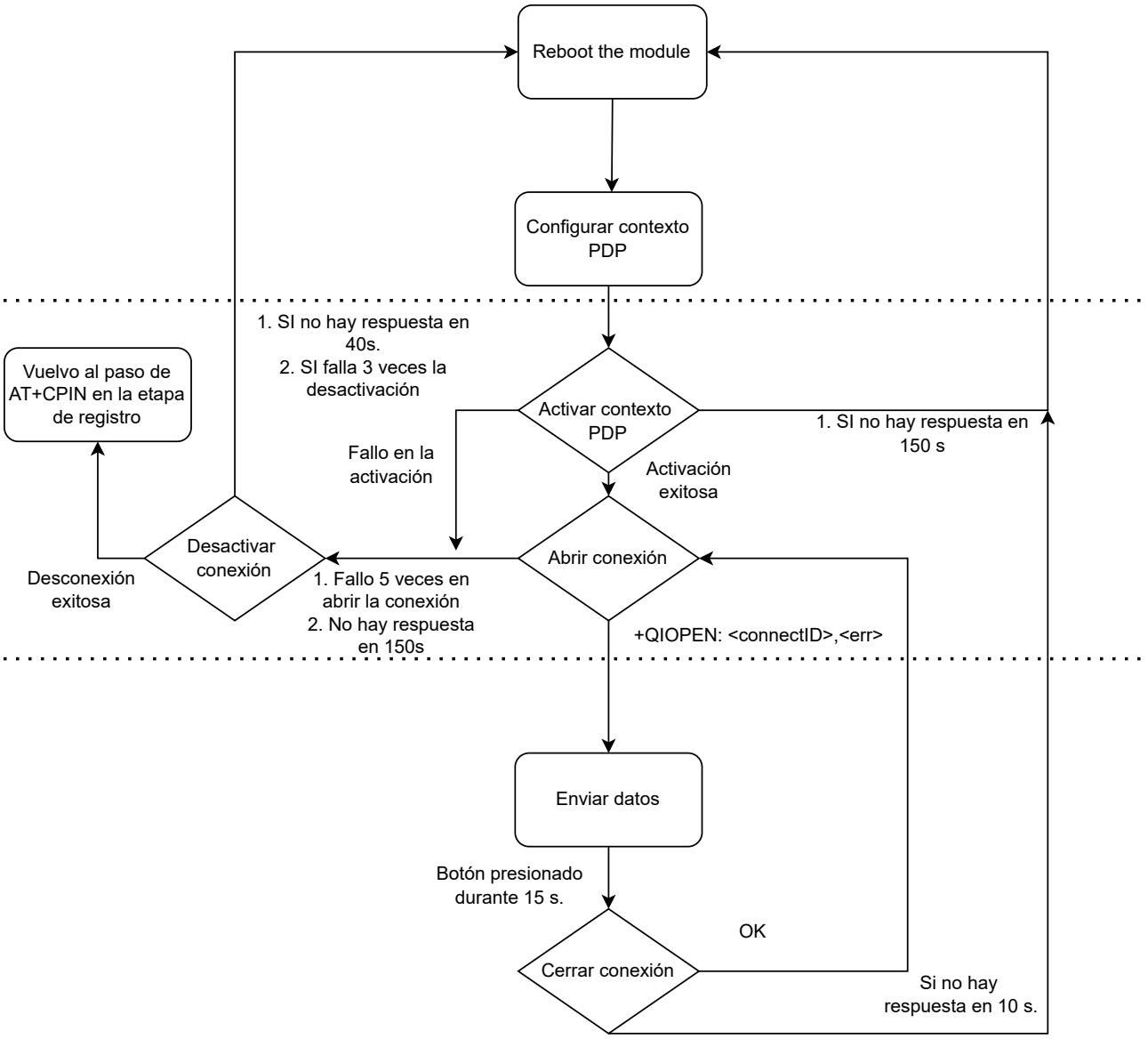

Figura 3: Diagrama de flujo para establecimiento de un servicio UDP.

Abrir conexión:

- **AT+QIOPEN:** Con este comando se abre un Socket Service, y con él se escoge el servicio de tipo "cliente" UDP, la dirección IP del servidor remoto, el puerto local, y el modo de acceso a los datos del Socket.
- **AT+QISTATE:** Devuelve el estado de las conexiones abiertas.

Enviar datos:

- **AT+QISEND:** Permite configurar parámetros del envío de datos como el largo en bytes a enviar, y con él se indica el connect ID, la IP remota y el puerto remoto.

Cerrar conexión:

- **AT+QICLOSE:** Permite cerrar la conexión.

4.3. Software

4.3.1. Compilación

Se utilizó una máquina virtual con el repositorio de Contiki-NG y la herramienta Uniflash para el cargado del programa en el Launchpad.

Es fundamental la incorporación de archivos del tipo "project_conf.h", donde se configuran banderas en tiempo de compilación; y "Makefile", donde se especifican los archivos a compilar. El proceso de ejecución se corresponde con la siguiente secuencia:

1. Dentro del directorio del programa a compilar se ejecuta el comando "make" desde la terminal [3].

2. Una vez compilado, el cargado del programa al Launchpad se realiza desde el programa Uniflash. Se selecciona el hardware con el que se está trabajando y se carga en archivo ".elf" generado en la compilación.

3. Ahora, la ejecución del programa se debe efectuar nuevamente empleando la terminal. En el mismo directorio se ejecuta "make login PORT='/dev/ttyACM0' " (este puedo puede variar según la computadora, pero siempre debe ser indicado) [4].

Para la compilación con el LAUNCHXL-CC2650 se utilizó TARGET = simplelink y BOARD = launchpad/cc2650.

4.3.2. Implementación

Todos los módulos creados fueron ubicados en la ruta "/contiki_ng/os/dev" para poder ser incluidos correctamente en los test realizados.

main-udp.c

Programa principal. En este se realiza la rutina descrita en el diagrama de flujo de la fig. 3. El proceso del botón, luego de soltar el botón, postea un evento para envíar un mensaje UDP a cierto socket.

serial-line.c

Librería de contiki donde se procesos y eventos asociados a la comunicación serial. Se usaron en particular la función serial_line_input_byte y el evento serial_line_event_message.

redirect.c

En este modulo se implementa la función de envió de strings vía uart, y el proceso que se llamara cuando llegue una respuesta del modulo, siendo esta analizada mediante el arbol de nodos.

comm_quectel.c

Por ultimo este modulo se encarga de todas los comandos necesarios de enviar al modulo para su correcta programación. Tiene definidos varios procesos para funcionalidades especificas que postean eventos a un proceso principal llamado UDP_main_process, este es el encargado de iniciar todos los procesos, una vez realizada la configuración, el proceso espera un input del

[3] El comando "make" funciona de esta manera si en el archivo Makefile se especifica el target y board para el cual se compila el programa. Si esto no se especifica, el comando a enviar tendrá la estructura "make TARGET=XXX BOARD=XXX"

[4] Este paso se realiza siempre y cuando los jumpers del Launchpad asociados a la UART se encuentren colocados, esto conecta las señales del microcontrolador al chip debugger

usuario ya sea para enviar datos o para cerrar la conexión. En este caso se implemento otro proceso asociado al botón que posteara el evento para el envió de la información de cuanto tiempo estuvo presionado.

5. Tests

5.1. base-demo.c

La primera prueba consistió en corroborar la correcta compilación y uso del launchpad. Se utilizó el test ejemplo "base-demo.c" propio de Contiki-NG asociado al LAUNCHXL-CC2650. Se puso a prueba el funcionamiento de los botones y leds que componen el hardware. En la fig. 4 se muestra los mensajes recibidos en terminal de varios sensores.

```
user@user-contikit-ng:~/contiki-ng/test-contiki-rsi/test-echo$ make login PORT='/dev/ttyACM0'
rlwrap ../../tools/serial-io/serialdump -b115200 /dev/ttyACM0
connecting to /dev/ttyACM0 [OK]
[INFO: Main      ] Starting Contiki-NG-release/v4.8-303-g29a2da51a-dirty
[INFO: Main      ] - Routing: RPL Lite
[INFO: Main      ] - Net: sicslowpan
[INFO: Main      ] - MAC: CSMA
[INFO: Main      ] - 802.15.4 PANID: 0xabcd
[INFO: Main      ] - 802.15.4 Default channel: 25
[INFO: Main      ] Node ID: 33927
[INFO: Main      ] Link-layer address: 0012.4b00.0f8e.8487
[INFO: Main      ] Tentative link-local IPv6 address: fe80::212:4b00:f8e:8487
[INFO: CC13xx/CC26xx] Operating frequency on 2.4 GHz
[INFO: CC13xx/CC26xx] RF: Channel 25, PANID 0xABCD
CC13xx/CC26xx base demo
----------------------------------------
Bat: Temp=26 C
Bat: Volt=3339 mV
Key Left press event
Key Left release event
Key Right press event
Key Right release event
```

Figura 4: Mensajes en terminal de test base_demo.c .

5.2. test-echo.c

Para el test de la comunicación UART se realizó una prueba tipo "echo", donde se verifica a través de una terminal la correcta recepción y transmisión de datos. En este método se ingresa una palabra en la terminal, esta es recibida por el microcontrolador y luego es transmitida y mostrada en la terminal. Se observa en la fig. 5 los mensajes desplegados en terminal.

```
user@user-contikit-ng:~/contiki-ng/test-contiki-rsi/test-echo$ make login PORT='/dev/ttyACM0'
rlwrap ../../tools/serial-io/serialdump -b115200 /dev/ttyACM0
connecting to /dev/ttyACM0 [OK]
string is being sent
HOLA
Data received over UART HOLA
string is being sent
COMO ESTAS
Data received over UART COMO ESTAS
```

Figura 5: Mensajes en terminal de test_echo.c .

5.3. test-redirect.c

En este test se realiza la implementación de los módulos asociados al algoritmo de búsqueda de comandos AT. En este se espera un evento que es posteado cuando se ingresan datos por

terminal, luego de suceder el evento, se invoca al proceso redirect en el cual se envían los caracteres recibidos por terminal al árbol de nodos ssp. Activando una flag de debug ubicada en el módulo "ssp_cfg.h" se pudo seguir la trayectoria de la cadena en el árbol de nodos notando si el mensaje fue descartado o coincidió con algún nodo hoja. En la fig. 6 se muestra los mensajes recibidos al realizar la búsqueda en el intérprete de comandos de la cadena de caracteres enviada por terminal.

```
user@user-contikit-ng:~/contiki-ng/test-contiki-rsi/test-redirect$ make login PORT='/dev/ttyACM0'
rlwrap ../../tools/serial-io/serialdump -b115200 /dev/ttyACM0
connecting to /dev/ttyACM0 [OK]
OK
In: 'O' (79)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  1)
In: 'K' (75)
        Not match. Reset to "root" node
POWERED
In: 'P' (80)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  0)
In: 'O' (79)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  1)
In: 'W' (87)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  2)
In: 'E' (69)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  3)
In: 'R' (82)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  4)
In: 'E' (69)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  5)
In: 'D' (68)
        Find in pattern "POWERED DOWN" from node "root" (match pos =  6)
```

Figura 6: Mensajes en terminal de test_redirect.c .

5.4. test-boton.c

Se estudia el posteo de un evento luego de que el botón sea presionado durante cierta cantidad de segundos y liberado. Se hizo énfasis en obtener la información correcta de cuántos segundos se mantiene presionado el botón. En este test se espera hasta que suceda el evento de presionado de botón y luego, se se espera hasta obtener el evento de soltado de botón para postear el evento y enviar los datos. En la fig. 7 se muestran la información obtenida del sensor botón.

```
user@user-contikit-ng:~/contiki-ng/test-contiki-rsi/test-boton$ make login PORT='/dev/ttyACM0'
rlwrap ../../tools/serial-io/serialdump -b115200 /dev/ttyACM0
connecting to /dev/ttyACM0 [OK]
El boton fue presionado 0 segundos
El boton fue presionado 3 segundos
El boton fue presionado 4 segundos
El boton fue presionado 7 segundos
```

Figura 7: Mensajes en terminal de test_boton.c .

5.5. main-quectel.c

Para este test se comenzó la implementación de algunos comandos AT básicos para su envío al módulo BG96. En específico se trabajó con el registro del módulo en la red para lo cual se implementaron los comandos de la sección 4.2. Para la corroboración del correcto funcionamiento se utilizó el osciloscopio y analizador lógico Analog Discovery 2 [9]. Mediante el osciloscopio se verificó la presencia del pulso necesario para prender el módulo BG96, y con el analizador lógico se siguieron los mensajes transmitidos desde el Launchpad hacia el módulo y

viceversa. En la fig. 8 se muestra una captura de pantalla del analizador lógico del AD2 (Analog Discovery 2) donde se muestran carácteres recibidos del módulo de comunicación BG96.

Figura 8: Mensajes en analizador logico de main_quectel.c .

Adicionalmente, se probó el funcionamiento completo del sistema utilizando un servidor UDP de Quectel. Se configuró el módulo de comunicación BG96 para envíar mensajes UDP a la IP y puerto que se muestran en la fig 9, en esta misma figura se pueden ver los mensajes recibidos.

Figura 9: Mensajes recibidos en servidor UDP provisto por Quectel.

6. Conclusiones

En cuanto al tiempo inicialmente calculado para la dedicación de este proyecto, el mismo sufrió modificaciones. La tarea de compilación ocupó casi la mitad de tiempo total invertido finalmente, puesto que se probaron muchos métodos de compilación antes de llegar a la secuencia correcta. Otra complicación vista al momento de testear el código se dio por el hecho de contar con un único módulo UART en el launchpad. Esto es debido a que la comunicación con el módulo de BG96 utiliza este protocolo para el envío de mensajes AT, lo cual imposibilitó el debuggeo del código por terminal (en un caso ideal se realizaría un redireccionamiento de los mensajes enviados y recibidos para ser desplegados en terminal utilizando otro módulo UART). Como alternativa, se hizo uso del Analog Discovery 2. Esto solucionó el problema de no poder ver mensajes enviados y las respuestas. De igual manera,esta herramienta no es recomendable para implementaciones de varios comandos, pues no se llegan a detectar todos los datos por el rango de lectura del programa.

Con respecto a la implementación del software, la principal dificultad se enfrentó al tratar con el Watchdog timer, pues en varias instancias se observó que el no ceder correctamente el procesador para que varios procesos se ejecuten, causaba el reinicio del programa. Un detalle

a tener en cuenta es que las variables que fueron utilizadas en los distintos procesos debieron ser estáticas, esto fue así para prevenir que las mismas sean modificadas por fuera del módulo. Por ultimo referido a los mensajes UDP podemos afirmar que se logró exitosamente debido a que logramoss recibir los mensajes utilizando el servidor externo de quectel. Confirmando tambien el correcto funcionamiento de la integracion de todos los modulos, ya que el mensaje se desplegaba con los datos correctos.

7. Anexo

7.1. Planificación propuesta para Especificación Breve

Se buscará terminar con el proyecto en tres semanas. En la primera semana nos centraremos al estudio del hardware y software dado además de los protocolos a utilizar. En la segunda semana comenzaremos con la adaptación del código a procesos y eventos de Contiki-NG. Por último, implementaremos la toma de medidas del sensor tag, el envío UDP y caracterización de consumo. A lo largo de las semanas se trabajará en paralelo con la documentación.

Bibliografía

[1] LAUNCHXL-CC2650 - SimpleLink™ CC2650 wireless MCU LaunchPad™ Development Kit. Texas Instrument. [Online]. Available: https://www.ti.com/tool/LAUNCHXL-CC2650

[2] LTE IOT 2 CLICK. MikroElectronika. [Online]. Available: https://www.mikroe.com/lte-iot-2-click

[3] Contiki-NG: The OS for Next Generation IoT Devices. Github. [Online]. Available: https://github.com/contiki-ng/contiki-ng

[4] (2019) ¿Qué es NBIOT? Luxnnia. [Online]. Available: https://luxnnia.com/portfolio-items/que-es-nbiot/

[5] (2018, Noviembre) ¿Qué es NBIOT? Universitat Oberta de Catalunya. [Online]. Available: https://luxnnia.com/portfolio-items/que-es-nbiot/

[6] L. Francucci. Identificación de respuestas a comandos AT en ISR. Intérprete de comandos. EMBEDDED EXPLOITED. [Online]. Available: http://embedded-exploited.blogspot.com/2012/12/identificacion-de-respuestas-comandos.html

[7] (2021, Diciembre) Conjunto de comandos de Hayes. Wikipedia. [Online]. Available: https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes

[8] *LTE Module Series BG96 TCP/IP - AT Commands Manual*, Quectel, 2019. [Online]. Available: https://sixfab.com/wp-content/uploads/2018/09/quectel_bg96_tcpip_at_commands_manual_v1-1.pdf

[9] Analog Discovery 2. Digilent. [Online]. Available: https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/