Laboratorio 4: NB-IoT

Este laboratorio se propone poner en práctica los principales conceptos de NB-IoT para realizar la conexión a la red celular (*attach*) y enviar datos via MQTT. Se interactuará directamente con el módulo de comunicación utilizando una terminal serie (programando *passthrough* en el microcontrolador).

Introducción

En este laboratorio del curso utilizaremos la placa de desarrollo Arduino Zero y el módulo BG96 de Quectel incorporado en la placa *LTE IoT 2 click* (Mikro-Electronica) conectados a través de la placa de adaptación. Opcionalmente se utilizará un Arduino MKR 1500 que incluye el mismo microcontrolador que el Arduino Zero y el módulo de comunicación SARA de ublox (específicamente SARA-R410M-02B).

Se utilizará Mosquitto que es un servidor/*broker* MQTT público basado en el software libre y de código abierto Eclipse Mosquitto™.

Objetivos

Objetivos generales

Realizar la conexión a la red celular (NB-IoT) para enviar y recibir datos a través de un *broker* MQTT.

Objetivos particulares

- Familiarizarse con los comandos AT en general y con los específicos del módulo BG96.
- Realizar el proceso de conexión (*attach*) a una red.
- Establecer una conexión a un *broker* MQTT.
- Enviar y recibir datos a través de MQTT.

Materiales

- Computadora con Arduino IDE instalado y configurado (ya utilizado en el laboratorio 1).
- Placa de desarrollo Arduino Zero.
- Placa de adaptación MikroBus para Arduino.
- Placa LTE IoT 2 Click (que incluye el módulo BG96).

Fundamentos

Comandos AT

Los comandos AT a utilizar en este laboratorio tienen distintas sintaxis, pueden clasificarse, a grandes rasgos, en cinco tipos distintos:

1. Comandos de sintaxis básica: `AT<x><n>` donde `<x>` es el comando y `<n>` es el valor a establecer.
2. Comandos de testeo: `AT+<x>=?` Retorna la lista de parámetros y rangos de valores posibles que se le pueden pasar al comando de escritura.
3. Comandos de lectura: `AT+<x>?` Retorna el valor de uno o más parámetros.
4. Comandos de escritura: `AT+<x>=<...>` Establece un parámetro con valor definido por el usuario.
5. Comandos de ejecución: `AT+<x>` Realiza una lectura de un parámetro no variable (no modificable por el usuario) afectado por el comportamiento interno del módulo.

Mosquitto

El servidor (*broker*) MQTT a utilizar en este laboratorio es Mosquitto; se trata de un sistema público basado en el software libre y de código abierto Eclipse Mosquitto™. Tiene diversas ventajas y funcionalidades que lo hacen una opción muy práctica para probar nuestros desarrollos:

- Es basado en software libre y de código abierto.
- Soporta tanto IPv4 como IPv6.
- Soporta opcionalmente autenticación de usuario/contraseña.
- Soporta opcionalmente encriptación SSL.
- Soporta WebSockets[1].

MQTTX

MQTTX es un cliente MQTT libre y de código abierto multiplataforma, disponible para Linux, Windows y macOS. El código fuente puede encontrarse aquí.

LTE IoT 2 Click (BG96)

El módulo LTE IoT 2 Click de MikroElectronica cuenta con un módulo de comunicación BG96 y una interfaz MikroBus. El módulo BG96 utiliza comandos AT para su configuración y operación. En la página del curso se encuentran publicados documentos que detallan los comandos para algunas aplicaciones.

Terminal serie

La terminal serie provista por el IDE de Arduino tiene la dificultad de que no facilita el envío del caracter `Ctrl-Z` (caracter 26 en decimal o 1A en hexadecimal) requerido por el comando `AT+QMTPUB`. Hay varias formas de solventar esto:

1. Utilizar el software Minicom. Se trata de un software muy potente pero que no es tan sencillo de utilizar y además no se encuentra disponible para Windows. En Windows y en Linux se puede utilizar PuTTY.

[1] Esto es muy útil cuando se quiere, por ejemplo, utilizar un cliente web como el que brinda HiveMQ aquí.

2. Buscar un editor de texto que soporte este caracter (ya sea presionando `Ctrl-Z` o `Alt-026`), luego copiarlo y pegarlo en la terminal.
3. Evitar el uso de este caracter, optando por una implementación alternativa del comando `AT+QMTPUB` (por ejemplo especificándole de antemano cuántos caracteres contendrá el mensaje).

En algunos terminales seriales, por ejemplo Minicom, puede resultar cómodo deshabilitar el *echo* del módulo, es decir, el hecho de que el módulo repita cada caracter que se le envía. Esto se realiza con el comando `ATE0`.

Pasos para envío y recepción de datos via MQTT

A continuación se exponen a modo de ejemplo los comandos AT para enviar y recibir mensajes desde un *borker* MQTT. Las líneas que comienzan con `#` corresponden a comentarios del comando, por más detalles se puede consultar la documentación publicada en la página del curso. Todos los comandos terminan con un caracter de retorno de carro (`<CR>` o `\r`) seguido de un caracter de retorno de línea (`<LF>` o `\n`).

```
# test de la comunicación (debe devolver OK):
AT

# Opcional para deshabilitar eco del módulo:
ATE0

# Consulta si la tarjeta SIM requiere PIN:
AT+CPIN?

# Proporcionar el PIN de la tarjeta SIM (en caso de ser necesario):
AT+CPIN=1323

AT+CPIN?

# Deshabilita la radio:
AT+CFUN=4

# Parámetros de configuración para la selección de redes y servicios:
AT+QCFG="nwscanmode",3,1

AT+QCFG="iotopmode",1,1

AT+QCFG="servicedomain",1,1

# Configuración de IP (IP -> IPv4; IPV6 -> IPv6), y APN (iie.iot):
AT+CGDCONT=1,"IPV6","iie.iot","",0,0

# Restablece la radio:
```

```
AT+CFUN=1

# Conecta con la red de Antel:
AT+COPS=1,2,"74801",9

# Consulta si se registró correctamente en la red
# (debe devolver 0,1, en caso contrario esperar unos segundos y volver a consultar:
AT+CEREG?

# Consulta la información de la red.
AT+QNWINFO

# Consulta el IP asignado por la operadora (notar la rareza en la sintaxis del comando):
AT+CGPADDR=1

# Abre la conexión con el broker:
AT+QMTOPEN=0,"test.mosquitto.org",1883

# Se identifica ante el broker:
AT+QMTCONN=0,"nodo_tiot","",""

# Publica un mensaje en el tópico TIoT/mensajes_de_prueba (el mensaje termina con Ctrl-Z).
AT+QMTPUB=0,0,0,0,"TIoT/mensajes_de_prueba"

# Se suscribe a un tópico o un grupo de tópicos:
AT+QMTSUB=0,1,"TIoT/#",2
```

Tareas y actividades a realizar

Intercambio de mensajes utilizando el *pasadizo* (passthrough)

1. Interactuar manualmente con la UART del módulo BG96 para enviar y recibir datos via MQTT, verificando el correcto funcionamiento. Esto es:
    1. Programar la Arduino con el programa *passthrough*, conectar el módulo *LTE IoT 2 Click* usando el adaptador y probar la comunicación con el comando de prueba `AT`.
    2. Abrir el cliente MQTTX, conectarse a test.mosquitto.org y suscribirse a tópico `TIoT/#`.
    3. Seguir la secuencia de comandos AT detallada más abajo para enviar un mensaje al *broker* en un sub-tópico `TIoT/#` y verlo en el cliente MQTTX. **Sugerencia:** Utilizar un tópico del tipo `TIoT/<nombre>` donde `<nombre>` es el nombre de pila del estudiante, por ejemplo `TIoT/Felisberto`.
    4. Continuar con la secuencia de comandos AT para suscribirse al tópico `TIoT/<nombre>/para_mi`.
    5. Enviar un mensaje desde el cliente MQTTX y verificar la recepción

por el módulo BG96.
6. Consultar la hora del dispositivo BG96.

2. Repetir los pasos anteriores pero realizando las modificaciones necesarias para conectarse en forma autenticada (con usuario/contraseña), tanto desde el módulo BG96 como desde el cliente MQTTX.

Envío de temperatura

Utilizando el ejemplo `esqueleto_AT` provisto por los docentes del curso realizar un programa que implemente la conexión con el *broker* `test.mosquitto.org`, se suscriba al tópico `TIoT/<nombre>/para_mi` y envíe secuencialmente la temperatura del módulo (cada 5 segundos aproximadamente) utilizando la sintaxis JSON:

```
{"msg_id": <msg_id>, "temperature": <temperature>}
```

donde `<msg_id>` es un contador del número de mensaje (comienza en 0 y se incrementa), y `<temperature>` es la temperatura interna del microcontrolador.

Las funciones utilizadas en `esqueleto_AT` tratan de mostrar en la consola información relevante para depurar errores:

- Los comandos que se envían al módulo BG96 aparecen entre corchetes: `{<comando>}`.
- La respuesta que NO es analizada por el analizador (desechada por la directiva `CLEAR_BUFFER`, aparece entre paréntesis rectos: `[<desecho>]`.
- Cuando el analizador se pierde al buscar la respuesta deseada, muestra el caracter numeral: `#`.
- Todo lo demás es analizado en busca de la respuesta que se espera recibir luego de cada comando.

No es necesario tener en cuenta los errores, desconexiones o códigos no solicitados (URC).

Verificar el correcto funcionamiento de la solución. Esto es, que los mensajes que se envían desde la Arduino se visualicen correctamente en el cliente MQTT, y, de forma análoga, que los mensajes enviados desde el cliente MQTT se vean en el terminal serie de la Arduino (aunque no sean procesados por el programa realizado).

Envío de datos usando FreeRTOS

Realizando un procedimiento análogo al del Laboratorio 2, escribir un programa que envíe datos de múltiples fuentes utilizando un tópico distinto para cada tipo de dato.

Verificar el correcto funcionamiento de la solución. Esto es, que los mensajes que se envían desde la Arduino se visualicen correctamente en el cliente MQTT, y, de forma análoga, que los mensajes enviados desde el cliente MQTT se vean

en el terminal serie de la Arduino (aunque no sean procesados por el programa realizado).

Entregables y Defensa

Cada estudiante debe asistir a la defensa con el laboratorio realizado en su completitud, y habiendo contestado de manera individual -y en tiempo y forma- la encuesta correspondiente publicada en la plataoforma EVA.

El día de la defensa deberán poder explicar cada una de las partes de este laboratorio.

Referencias

- Quectel_BG96_AT_Commands_Manual_V2.3.pdf
- Quectel_BG96_MQTT_Application_Note_V1.2.pdf
- https://docs.arduino.cc/hardware/zero
- https://www.mikroe.com/arduino-uno-click-shield
- https://www.mikroe.com/lte-iot-2-click