EXS82 connectivity demonstration (MQTT)

MicroEJ

Mar 09, 2021

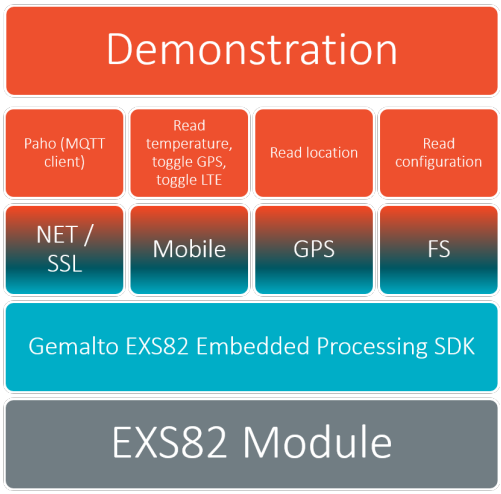
CONTENTS

CHAPTER

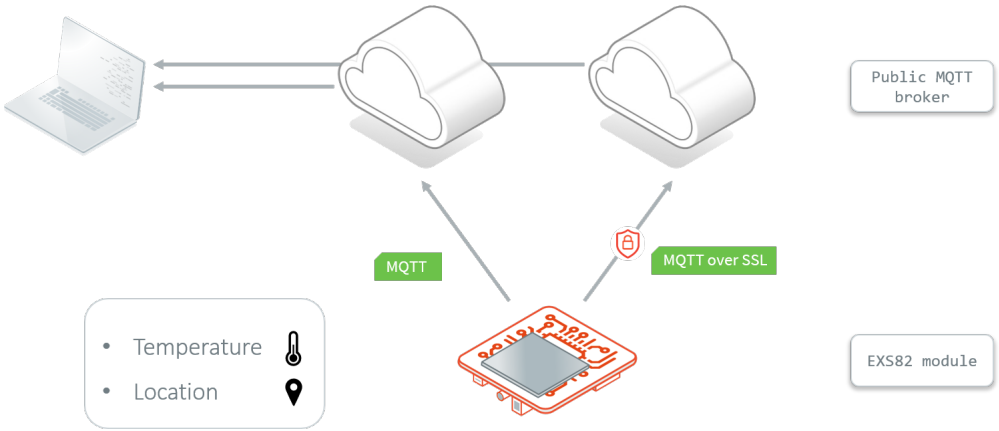
ONE

OVERVIEW

This package demonstrates the MicroEJ support of the Gemalto EXS82 Embedded Processing SDK.



It contains a MicroEJ (Java) application that is derived from the MQTT THALES showcase : https://developer.gemalto.com/showcase/concept-board-connection-azure-iot-cloud-mqtt-protocol.



CHAPTER

TWO

REQUIREMENTS

To follow the steps described in this documentation, you will need:

- A Gemalto Cinterion LGADevKit with:
  - a Gemalto EXS82 module
  - a wideband antenna
  - a SIM card
  - cables
- The Gemalto EXS82 Embedded Processing SDK Version 00.024 with:
  - the THALES firmware for EXS82 version 00.024
  - the tools to interact with the module
- A Python 2.7 installation with the serial module
- A web browser

CHAPTER

THREE

USAGE

This demo is divided in two parts:

- *MQTT demo on the device*: run the MicroEJ application on the device and get messages from the Internet.
- *Build the application binaries*: rebuild the MicroEJ application binaries from sources.

The first requires the hardware and the Gemalto SDK tools. The second requires the MICROEJ SDK and aims to rebuild the binaries used by the two previous steps from sources.

MQTT demo setup helps to setup a nice looking MQTT web client to see messages from the (virtual) device. *Configuration* explains how to configure the application.

3.1 MQTT demo setup

To see messages sent by the (virtual) device, one will need to setup an MQTT client.

The following is a nice looking MQTT web client:

1. Open your web browser and access http://www.hivemq.com/demos/websocket-client/
2. In the "Connection" section, set the configuration (See *Configuration*). The default configuration for this demo is:
   - host : `broker.hivemq.com`
   - port : `8000`
   - leave other fields unchanged
3. Click on "Connect" to open the connection.
4. Skip the "Publish" section.
5. In the "Subscriptions" section, subscribe to the MQTT topic used by the demo (`MQTTDemo` by default):
   - Click on "Add New Topic Subscription"
   - Set topic to `MQTTDemo`
   - Click on "Subscribe"
6. The messages sent by the demo will be shown in the messages box.

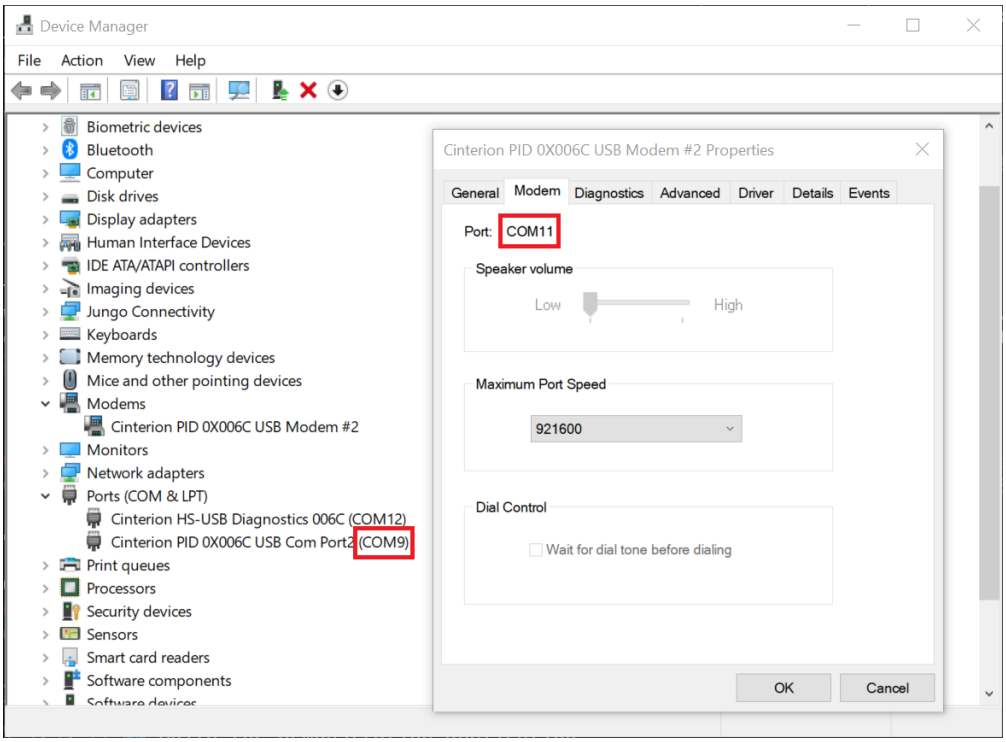
3.2 MQTT demo on the device

Requirements: the Gemalto EXS82 Embedded Processing SDK tooling (tested with version 00.024) and a Cinterion module & LGA DevKit, a SIM card, an LTE antenna and a GPS antenna (see the SDK documentation for setup).

The GPS capability may require to run the demo outdoor. However, it is possible to disable it to skip the localisation detection phase (see *Configuration*). Additionally, if you do not skip the location detection phase, make sure the assistance is disabled (`AT^SGPSC="Engine/StartMode",0` , it is the default configuration).

In following steps,

- `<COM port>` refers to the serial port to the modem (`Cinterion PID 0X006C USB Modem #2`).
- `<Log port>` refers to the secondary serial port (`Cinterion PID 0X006C USB Com Port2`).



1. Plug the DevKit into your computer.
2. Follow Getting Started from the Gemalto EXS82 Embedded Processing SDK to update the firmware to version 00.024.
3. Configure the SIM credentials : PIN code, APN, username & password (see *Configuration*).
4. Upload the configuration file `demo.properties.list` to the board:

```
<PYTHON_PATH>/python.exe <GEMALTO_SDK>/tools/fs.py download -p <COM port> demo.properties.list A:/
↪demo.properties.list
```

5. Optionally, listen on the log port:

```
<PYTHON_PATH>/python.exe <GEMALTO_SDK>/tools/log.py read -p <COM port> -d "<Log port>, 460800"
```

6. Upload the application `demo-mqtt.bin` to the board and start the application:

```
<PYTHON_PATH>/python.exe <GEMALTO_SDK>/tools/app.py download -p <COM port> demo-mqtt.bin
<PYTHON_PATH>/python.exe <GEMALTO_SDK>/tools/app.py start demo-mqtt
```

7. The demo will start pushing messages to the MQTT broker after the first GPS fix is established (or the timeout expires).

3.3 Configuration

It is possible to configure the demo application using the `demo.properties.list` at the root of the filesystem.

The following properties are configurable:

- `broker.url` : the URL of the MQTT broker (`[tcp/ssl]://<domain>:<port>`)
- `broker.certificate` : the path of the root CA certificate of the MQTT broker (`/path/to/root-ca.cer`), only required for secured MQTT broker connections.
- `broker.username` : username used to access the broker (optional).
- `broker.password` : password used to access the broker (optional).
- `sim.pin` : PIN code used to unlock the SIM.
- `sim.apn` : APN of the network used for Internet communications.
- `sim.username` : username used to access the network.
- `sim.password` : password used to access the network.
- `mqtt.topic` : topic on which messages are published (defaults to `MQTTDemo`).
- `skip.gnss` : enable/disable the location detection (defaults to `false`).

The application must be restarted for the change to take effect.

3.4 Build the application binaries

3.4.1 Installation

1. Install the MicroEJ SDK distribution from https://repository.microej.com/packages/SDK/20.12/.
2. Launch the IDE and create a workspace on your filesystem.
3. Select the default Microej repository.
4. Import `offline-repository.zip` with “File > Import… > General > Projects from Folder or Archive > Archive…”.
5. Configure the Microej Module Manager to use the repository with “Window > Preferences > MicroEJ > Module Manager > Workspace…” and select the `ivysettings.xml` at the root of the project imported.

3.4.2 Build the platform

1. Import all platform projects from `platform-sources.zip` with "File > Import… > General > Existing Projects into Workspace > Select archive file > Browse…".
2. Right click on the `exs82-threadx-configuration` project and click on `Build Module`.
3. Wait for the `exs82-threadx-platform` project to be populated.

3.4.3 Build the demonstration application

1. Import all platform projects from `demonstration-sources.zip` with "File > Import… > General > Existing Projects into Workspace > Select archive file > Browse…".

3.4.4 Build the demonstration application binary

2. Set the following environment variables: - `GEMALTO_SDK` the path to the Gemalto EXS82 Embedded Processing SDK root directory (typically extracted from `exs82_rev00.024_arn01.400.00_SDK.tgz`). - `PYTHON27_PATH` the path to the Python 2.7 executable (not required if it is `C:\Python27\python.exe`).
3. Restart the IDE to apply the changes.
4. Run the `[Emb] demo-mqtt` launcher in "Run > Run configurations… > MicroEJ Application".
5. The firmware is built at `exs82-threadx-bsp/projects/microej/bin/microej.bin`, refer to *MQTT demo on the device* to play with the demo.

CHAPTER

FOUR

KNOWN ISSUES/LIMITATIONS

- The application doesn't currently support concurrent GNSS and network operations. Therefore, the network interface is shut down before locating the device.
- In case of failure, one may have to reboot the device as some resources may not be properly cleaned when stopping the application.