Internet of Things - Lab Notes#1

Lab 1: Basic connection of the device to cloud - Registration and Discovery

Software needed on your computer for this lab exercise

The mbed IDE is web based and requires no installation on your computer. There is some software that you will need for diagnostics and debugging:

1. Terminal Emulator program: e.g. putty for windows; mac users can use the “screen” program. For receiving diagnostic messages from the embedded software.
2. REST client software, e.g. Advanced REST Client for Chrome. To test REST API operation.
3. Windows users will need to install a serial link driver for diagnostics monitoring: http://developer.mbed.org/handbook/Windows-serial-configuration

Step 1. Get an mbed account

If you don’t already have an mbed account, go to
http://mbed.org/
and create a free account for yourself.

ARM mbed OS is a web based IDE and platform for creating embedded and connected software projects for ARM Cortex-M based microcontrollers (MCU). The IDE runs entirely on the web and downloads a binary file to your computer that you drag and drop to your mbed device, which appears as a storage drive when connected via USB. There is no software to install on your computer to use mbed. A serial terminal emulator is used to monitor diagnostic information from the mbed libraries and from your embedded application program.

Step 2. Connect the mbed board to your computer

1. Use a micro USB cable to connect the USB port labeled “OpenSDA” on the K64F to a USB port on your computer.
2. A new device called “mbed” should appear in the storage manager. You will drag and drop files here to program the mbed board.

Step 3. Import, Compile, and load the mDS demo application onto the mbed board

1. Go to the project page at:
   https://developer.mbed.org/teams/MBED_DEMOS/code/IoT_World_Hack_Demo/
2. Import the demo program using the “Import this program” button on the project page.
3. Edit main.cpp and change the endpoint name to something unique.

```
// Set our own unique endpoint name
```

```
#define MY_ENDPOINT_NAME                       "Changeme-LED-demo"
```

You also will eventually need to change the domain to your unique domain assigned for your project. For now, you can leave the domain set to the default “domain” for the purpose of running the example program, and change to your unique domain name after testing the example.

```
// My NSP Domain
#define MY_NSP_DOMAIN                          "domain"
```

4. Make sure the FRDM-K64F is selected in the upper right part of the screen. Compile the project. A few compiler warnings may be generated. If successful, a file named *IoT_World_Hack_Demo_K64F.bin* will be stored in your default location, e.g. Downloads folder.
5. Drag and drop the compiler output file to the “mbed” device in your storage manager which represents the K64F which is connected through USB.
6. When done copying (about 20 seconds), press the reset button on the K64F. The reset button is near the OpenSDA USB connector.
7. Status messages during startup and network connection can be monitored by connecting a terminal application to the USB serial port, baud rate = 9600. Pressing the reset button on the side of the mbed board at any time will cause a restart, enabling the diagnostic messages to be monitored from the beginning again.

Step 4. Connect your mbed board to the mbed Device Server instance.

After successfully installing the demo, the mbed board will automatically connect to the mbed Device Server instance at iot-hack-mds.cloudapp.net

Monitor the USB serial port during startup; you should see output similar to the following:

```
mbed mDS Sample Endpoint v1.0 (Ethernet)
plumbNetwork: pre-configure plumb of network...
plumbNetwork: configuring endpoint...
utils_configure_endpoint: setting defaults...
utils_configure_endpoint: gathering configuration overrides...
configure_endpoint: building endpoint configuration...
ThreadedResourceObserver being used for 3202/0/5600 (sleep_time=10000)
utils_configure_endpoint: finalizing configuration...
utils_configure_endpoint: endpoint configuration completed.
plumbNetwork: post-configure plumb of network...
net_stubs_post_plumb_network: Ethernet Address: 10.0.0.48
Start the endpoint to finish setup and enter the main loop...
Endpoint::start(): creating endpoint instance and registering with mDS...
net_stubs_post_plumb_network: initializing NSP..
.
NSP: libNsdl init successful.
NSP: libNsdl NSP_ADDRESS: 108.201.184.41 port: 5683
```

```
utils_init_and_register_endpoint: allocating endpoint instance...
utils_init_and_register_endpoint: registering endpoint and its resources...
Endpoint::initialize(): initializing NSP resource pointer...
Endpoint::initialize(): adding static resources...
Endpoint::initialize(): binding static resource: [3/0/0]...
StaticResource: [3/0/0] value: [Freescale] bound
Endpoint::initialize(): binding static resource: [3/0/1]...
StaticResource: [3/0/1] value: [K64F mbed Ethernet demo] bound
Endpoint::initialize(): adding dynamic resources...
Endpoint::initialize(): binding dynamic resource: [3311/0/5706]...
DynamicResource: [3311/0/5706] type: [Light] bound (observable: 0)
Endpoint::initialize(): binding dynamic resource: [3202/0/5600]...
DynamicResource: [3202/0/5600] type: [Slider] bound (observable: 1)
Endpoint::initialize(): binding dynamic resource: [3311/1/5706]...
DynamicResource: [3311/1/5706] type: [OnBoardLED] bound (observable: 0)
net_stubs_register_endpoint: calling NSP registration...
net_stubs_register_endpoint: NSP registration completed
Endpoint::start(): creating main loop plumbing...
Endpoint::start(): beginning main event loop...
```

The successful NSP registration indicates that your mbed board is now connected to the mbed Device Server and can be accessed through the RESTful web API. Your MAC and IP addresses may be different from this example.

5. Verify the registration with mbed Device Server

The mbed Device Server instance has 3 ports open for access:

http://iot-hack-mds.cloudapp.net:8080/domain/endpoints
REST API resources representing endpoint data, see the User Guide page for more detail and instructions.
User='app2'
Password='secret'

https://iot-hack-mds.cloudapp.net:8081
Administration interface to monitor endpoints, applications, and server status.
User='admin'
Password='admin'

coap://104.210.36.5:5683
CoAP endpoints connect to the ARM mbed Device Server platform and transfer data using the IETF CoAP protocol over UDP. The IP address is given here for use in the embedded software configuration.

Use your browser to navigate to https://iot-hack-mds.cloudapp.net:8081
The username is "admin" and the password is "admin". Select "Endpoints" and look in the list. You should find your unique endpoint name among the registered endpoints.

The resource schema for the demo follows the OMA LWM2M object model.

5. Use a REST Client to actuate the LED resource

The Base URL for all endpoints is:
http://iot-hack-mds.cloudapp.net:8080/domain/endpoints

Endpoint discovery: Doing a GET to this URL returns a JSON formatted list of registered endpoints.

Using a REST API client, such as the "Advanced Rest Client" Chrome plugin, perform a GET to the base URL and find your endpoint name in the returned JSON formatted list. A JSON containing entries similar to the following should be returned.

```
[1]
0:
{
name: "Changeme-LED-demo"
type: "mbed-eth-device"
status: "ACTIVE"
}
```

The endpoint name is added to this base path to identify an individual endpoint:
http://iot-hack-mds.cloudapp.net:8080/domain/endpoints/Changeme-LED-demo/

Resource discovery: Doing a GET to this URL returns a JSON formatted list of Objects and Resources for the specified endpoint.

Using a REST API client, such as the "Advanced Rest Client" Chrome plugin, perform a GET to the URL that represents your endpoint (substitute your endpoint name above). A JSON similar to the following should be returned:

```
0:
{
uri: "/3311/1/5706"
rt: "OnBoardLED"
obs: false
type: ""
}
-
1:
{
uri: "/3311/0/5706"
rt: "Light"
obs: false
type: ""
}
-
2:
```

```
{
uri: "/3303/0/5700"
rt: "Temp"
obs: true
type: ""
}
-
3:
{
uri: "/3/0/0"
obs: false
type: ""
}
-
4:
{
uri: "/3/0/1"
obs: false
type: ""
}
```

Finally, the resource is identified by adding the resource path uri as per the following:

`/3311/1/5706`

For example, the current LED setting for the above example board can be obtained by performing an http GET to the address:
http://iot-hack-mds.cloudapp.net:8080/domain/endpoints/Changeme-LED-demo/3311/1/5706?sync=true
Note ?sync=true to get a synchronous http response and see the value in your web browser or client. The server uses http basic authorization using the user name "demo" and password "secret".

The data type for controlling the LED is a hex color string RRGGBB where RR is a hex value for red, likewise GG for green and BB for blue. The onboard LED only responds to 0 and 1, so any hex color value with a "1" as the least significant bit will turn the corresponding color on. For example the string "00000000" turns all 3 LED colors off, and the string "01010100" turns all 3 colors on, producing white.

Using a REST API client, such as the "Advanced Rest Client" Chrome plugin, perform a GET to the URL that represents the LED resource (substitute your endpoint name above). A JSON STRING containing the current value should be returned:

```
000000
```

Using a REST API client, such as the “Advanced Rest Client” Chrome plugin, perform a PUT to the URL that represents the LED resource (substitute your endpoint name above). Supply a 8 character string of “1” and “0” to control the respective LED colors.

010101

Supplying the above string will illuminate the all 3 colors. Note that the string “FFFFFF” produces the same effect.

Your mbed board is now connected to mbed Device Server and you can control it using web applications through the mDS REST API.

The next lab note will explain and demonstrate the mbed library interfaces used to create new resources and embedded applications.

For more information:

ARM Nano Service Platform User Guide (mbed Device Server) - in the course materials

LWM2M Tutorial
http://community.arm.com/docs/DOC-8693

CoAP Tutorial
http://community.arm.com/docs/DOC-8633

General IoT Architecture and Standards
http://community.arm.com/docs/DOC-8649

White papers and free downloadable developer version of mbed Device Server platform
http://community.arm.com/groups/smart-and-connected/blog/2014/05/07/white-papers-on-arm-iot-software

A free developer version of mbed Device server may be downloaded and installed:
https://silver.arm.com/browse/SEN00