EMBSYS 110 Assignment 6

IoT Traffic Light

1 Goal

In this assignment, students will integrate the WiFi module to the project such that:

1. The traffic light status is reported to a server application.
2. The traffic light can be controlled by a server application.

2 Reference

1. ST UM2183 X-NUCLEO-IDW04A1 User Manual.pdf
2. STSW-IDW004 WiFi Hands On Training.pdf
3. ST UM2114 SPWF04Sx AT Command Reference.pdf

3 Basic Setup

1. *Carefully* plug in the following extension modules in the exact order onto your Nucleo board:

   (a) X-NUCLEO-IKS01A1 (IMU sensor).

   (b) X-NUCLEO-IDW04A1 (WiFi).

   (c) LCD module.

   Please ensure you align the pins correctly and do not bend any pins.

   Note – In this assignment we are only using the LCD and WiFi modules.

2. Download the compressed project file (platform-stm32f401-nucleo_assignment6.tgz) from the Assignment 6 course site.

   Place the downloaded tgz file in your VM under **~/Projects/stm32**.

3. Move your existing project folder to a backup folder, e.g.

   ```
   mv platform-stm32f401-nucleo platform-stm32f401-nucleo.bak
   ```

   Note – You may want to pick another backup folder name if the one shown above already exists.

4. Decompress the tgz file with:

Copyright (C) 2020 Lawrence Lo. All rights reserved.

```
tar xvzf platform-stm32f401-nucleo_assignment6.tgz
```

The project folder will be expanded to **~/Projects/stm32/platform-stm32f401-nucleo**.

5. Launch Eclipse. Hit F5 to refresh the project content.

   Or you can right-click on the project in Project Explorer and then click "Refresh".

6. Clean and rebuild the project. Download it to the board and make sure the LCD shows the traffic light graphics.

7. If you have not installed UMLet, please refer to the Setup notes in the previous assignment.

4 Tasks

4.1 Install Node.js

Node.js is a popular software platform for developing server applications with Javascript. To complete this assignment, you need to first install Node.js on your VM by following the steps below:

1. Install nvm - Node Version Manager.

   Open a Terminal, go to your home directory ("cd ~/") and type:

   `sudo apt-get update`

   `sudo apt-get install curl`

   (Note – if you see an error about resource temporarily unavailable, type this to remove a lock file: `sudo rm /var/lib/dpkg/lock`)

   `curl -sL https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash`

   (Note – The above is one line.)

   `source ~/.profile`

2. Type:

   `nvm ls-remote`

   It should list out available versions.

3. Type:

   `nvm install 11.14.0`

   `nvm alias default 11.14.0`

   It installs the specified node version (must *not* be 12.x.x). Type "**`node --version`**" to check version. If you see "v11.14.0", congratulations for having successfully installed Node.js.

Copyright (C) 2020 Lawrence Lo. All rights reserved.

4.2 Install TCP Server/Client Apps

1. Create a subdirectory named "node" in ~/Project:

   `cd ~/Projects`

   `mkdir node`

2. Download the following compressed files from the Assignment folder on the course website and put them in the "node" subdirectory created in step (1):

   statenode-tcpclient.tgz

   statenode-tcpserver.tgz

3. Decompress the tgz files under the "node" subdirectory:

   `cd ~/Projects/node`

   `tar xvzf statenode-tcpclient.tgz`

   `tar xvzf statenode-tcpserver.tgz`

4. Setup the TCP server application:

   `cd ~/Projects/node/statenode-tcpserver`

   `npm install`

   It is okay to ignore any warnings (in yellow). You shouldn't see any errors (in red) though.

5. Setup the TCP client application *in another Terminal window*:

   `cd ~/Projects/node/statenode-tcpclient`

   `npm install`

   You are now all set with the Node apps.

4.3 Test TCP Server/Client Apps

1. In the server Terminal (assuming it is in *statenode-tcpserver* folder), type:

   `node src/main.js`

   You should see some log messages ending with:

   "<LOG> TcpSrv: Enter started"

2. In the client Terminal (assuming it is in *statenode-tcpclient* folder), type:

   `node src/main.js`

   You should see some log messages ending with:

Copyright (C) 2020 Lawrence Lo. All rights reserved.

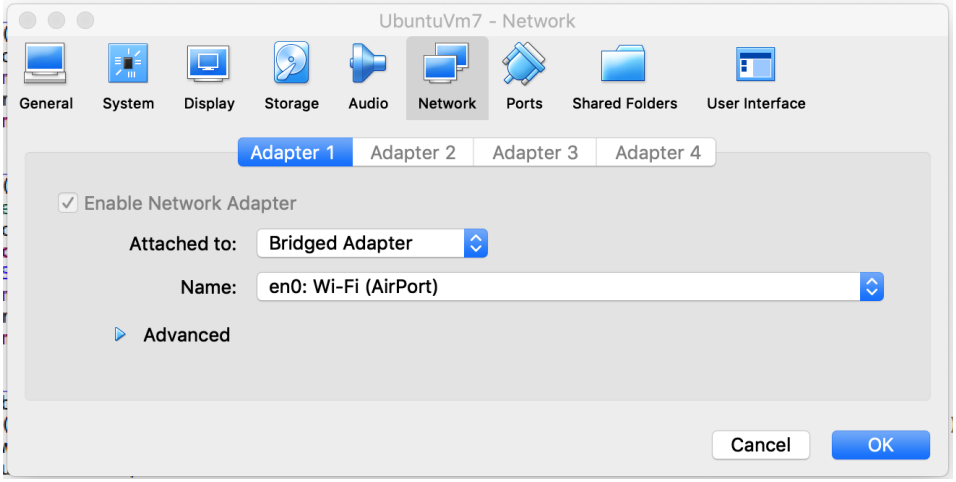
"<LOG> TcpClient: SockOnMessage: Received OK"

3. This shows that your TCP server is running correctly and is ready to get connected with your Nucleo board.

4.4 Network Settings of VM

To ensure data can be transferred between the VM and external device (i.e. the Nucleo board), we need to check a few settings of our VM

1. Ensure "Bridged Adapter" (rather than "NAT") is selected for the VM network adapter.



2. Ensure the port number 60002 is enabled in the firewall (in case it is enabled):

   ```
   sudo ufw allow 60002
   ```

4.5 WiFi Module Setup

The purpose of this section is to setup the WiFi parameters of your X-NUCLEO-IDW04A1 module so that it will connect to your home WiFi router (to which your VM is also connected).

1. First run the firmware from the Eclipse project `platform-stm32f401-nucleo`.
2. In minicom, type the command "`wifi interact`" to enter interactive mode.

   In interactive mode, the command console passes-through to the UART port of the WiFi module. That is, what you type on the console is sent to the WiFi module directly and its output is displayed on the console.
3. Follow the instructions on page 75, Section 4.2.1 of STSW-IDW004 WiFi Hands On Training.pdf to configure your WiFi module.

Copyright (C) 2020 Lawrence Lo. All rights reserved.

```
AT+S.WIFI=0
AT+S.SSIDTXT=<ssid>
AT+S.SCFG=wifi_wpa_psk_text,<password>
AT+S.SCFG=wifi_priv_mode,<mode>
```

where <mode> = 0 for none, 1 for WEP, **2 for WPA/WPA2**

```
AT+S.SCFG=wifi_mode,<mode>
```

where <mode> = **1 for STA**, 3 for MiniAP

```
AT+S.WIFI=1
AT+S.WCFG
```

You can check the connection status including the assigned IP address by typing:

```
AT+S.STS
```

You can check your WiFi settings by typing:

```
AT+S.GCFG
```

For example, if you want your WiFi module to connect to your AP with ssid = "My Router AP", type these:

```
AT+S.WIFI=0
AT+S.SSIDTXT=My Router AP
AT+S.SCFG=wifi_wpa_psk_text,my_password
AT+S.SCFG=wifi_priv_mode,2
AT+S.SCFG=wifi_mode,1
AT+S.WIFI=1
AT+S.WCFG
```

Alternatively, you may configure your WiFi module as an mini-AP (e.g. named "My Mini AP"), to which your PC/laptop connects directly (without both connected to a WiFi router):

```
AT+S.WIFI=0
AT+S.SSIDTXT=My Mini AP
AT+S.SCFG=wifi_priv_mode,0
AT+S.SCFG=wifi_mode,3
AT+S.WIFI=1
AT+S.WCFG
```

Copyright (C) 2020 Lawrence Lo. All rights reserved.

4. Once your WiFi module is properly configured you can exit the interactive mode by hitting CTRL-C, or simply resetting the Nucleo board. This will bring the firmware back to the command mode. Since the configuration has been saved to flash (of the WiFi module), you shouldn't need to configure it again unless you want to change the parameters.

4.6 Testing WifiSt

WifiSt (under src/Wifi/WifiSt) is the active object responsible for interfacing with the STMicro's WiFi module. It is partially implemented to demonstrate basic connectivity.

It handles the following events:

1. WIFI_CONNECT_REQ – Connects to the TCP server listening at the specified IP address and port number. The IP address should be that of your VM running the Node.js TCP server. You can check your VM's IP address by typing "**`sudo ifconfig`**". The port number used by the TCP server is fixed to be **60002**.
2. WIFI_DISCONNECT_REQ – Disconnects a previously established TCP connection.
3. WIFI_SEND_REQ – Sends the data carried by this event over a previously established TCP connection.
4. UART_IN_DATA_IND – Handles the UART data received from the WiFi module. These data are ASCII characters which can either be:

   a) AT response strings starting with "**`AT-S"`**

      See ST UM2114 SPWF04Sx AT Command Reference.pdf.

   b) Asynchronous messages starting with "**`+WIND:<WIND ID>"`**

      See page 57 of ST UM2114 SPWF04Sx AT Command Reference.pdf.

   Note that when the TCP server sends a message to the WiFi module, the module will first notify the Nucleo firmware via an asynchronous message "**`+WIND:55".`** Then the Nucleo firmware will issue an AT command "**`AT+S.SOCKR=0,\n\r`**" to request to read all the received data from the module. Then the module will reply with an AT response starting with "**`AT-S.Reading`**", followed by the message payload, and ending with "**`AT-S.OK`**".

   This is an example:

```
1122669 WIFI_ST(14): Received: +WIND:55:Pending Data::0:31:31

1122690 WIFI_ST(14): Received: at+s.sockr=0,
AT-S.Reading:31:31
this is my hello world message
AT-S.OK
```

Copyright (C) 2020 Lawrence Lo. All rights reserved.

Currently WifiSt simply prints out any received messages to the console, like the ones shown above. You will have to parse these received messages and appropriate actions (see below.)

You can test out the above functions with the following console commands. See the help text for details:

1. **`wifi connect`**
2. **`wifi send`**
3. **`wifi disconnect`**

To test data reception, just type a message in the Node.js TCP server terminal window (under **`~/Projects/node/statenode-tcpserver`**). You should see the same message printed on the command console of the Nucleo board.

4.7 Requirements

The current design of the **WifiSt** active object, together with the "wifi" console commands demonstrates basic functions using the WiFi module. Now you need to complete the design to meet the following requirements:

1. Send traffic light status to the Node.js TCP server which will print it out on the Terminal.
   a) Refer to the **`Conn()`** and **`Send()`** function in WifiStCmd.cpp (under src/Wifi/WifiSt) for an example of how to connect to the Node.js TCP server and send message to it. You can hardcode the IP address of your server and 60002 as the port number in your code (see Section 4.6 for details).
   b) Modify Traffic.cpp to connect to the TCP server upon start up (e.g. entry to the Started state). Note you will need to add an include file for "WifiInterface.h" at the top of the file.
   c) Modify Lamp.cpp (under src/Traffic/Lamp) to send a message (text string) to the TCP server whenever the traffic light is updated (similar to how you call the Draw() function in the previous assignment). Note you will need to add an include file for "WifiInterface.h" at the top of the file. You can design your own message format.
2. Handle commands received from the Node.js TCP server to control the traffic light direction (NS vs EW).
   a) Parse the data received from the WiFi module in the Normal state of WifiSt.cpp upon UART_IN_DATA_IND. See Section 4.6 item 4 for details of the data format. Your goal is to extract the message payload sent by the TCP server, i.e. to filter the AT command/response wrappers added by the WiFi module.
   b) For the purpose of this assignment your parser does not need to be perfect. You can assume the entire message payload always arrives together in a single AT response. You can design

Copyright (C) 2020 Lawrence Lo. All rights reserved.

your own message format to make your parsing easier. For example you can have a message "TRAFFIC NS" to represent a car request along the North South direction. You can use common C string library functions, such as strstr(), for parsing. Be careful not to read beyond the end of a received message.

c) Once a server message is identified in **WifiSt**, you can send the corresponding event (TRAFFIC_CAR_NS_REQ or TRAFFIC_CAR_EW_REQ) to **Traffic**, in a similar way as you send events from **System** to **Traffic** upon button press and hold in the previous assignment. The **Traffic** active object does not need to be modified.

5 Submission

The due date is 6/1 Monday 11:59pm. Please submit:

1. A description of your message format for reporting traffic light status and for sending requests.
2. A screenshot of the Terminal window running the Node.js TCP server displaying at least two different traffic light status messages.
3. A zip file containing the source code in **src/Wifi and src/Traffic.**

Upload to the usual Canvas assignment upload location.

Copyright (C) 2020 Lawrence Lo. All rights reserved.