EMBSYS 330 Assignment 6

IoT Level Meter

1 Goal

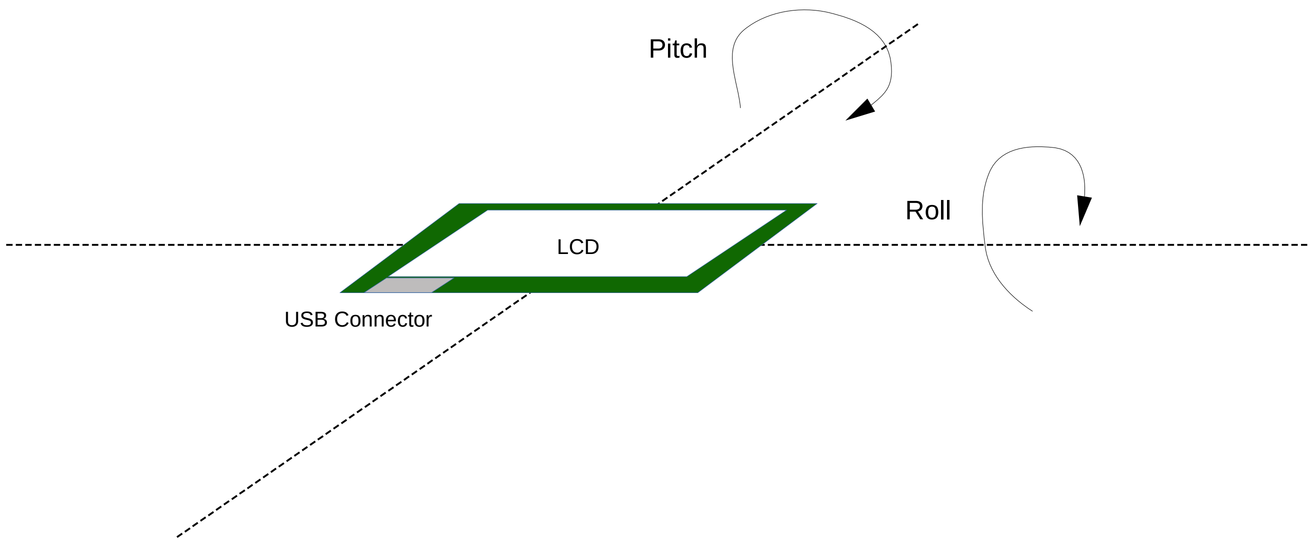
In previous assignments, students practiced state machine design and implementation through example projects such as LED patterns and traffic lights. In this assignment, students will apply similar techniques in building a more realistic application – an IoT level meter.

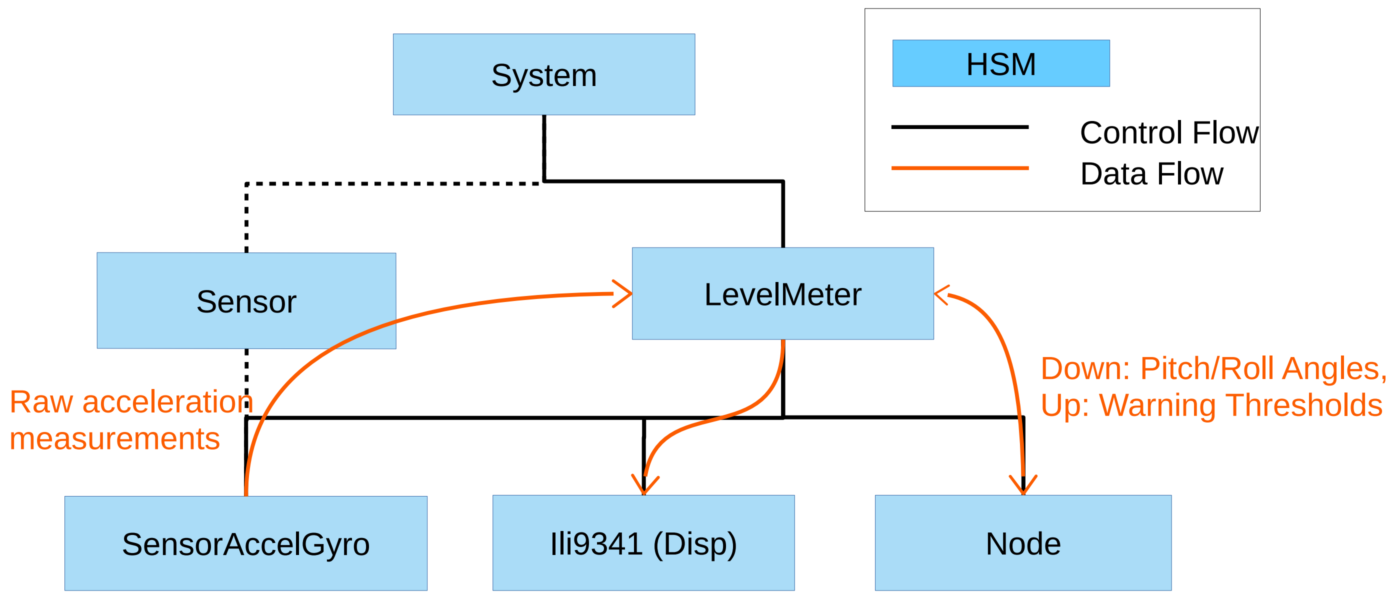
The **LevelMeter** active object acquires accelerometer data from the **SensorAccelGyro** HSM. It first averages measured data in each sample period and converts the raw x, y, z acceleration into pitch and roll angles.

The pitch and roll angles are updated periodically on the LCD via the **Ili9341** HSM. In the meantime, the same measurements are sent to the server (message router) via the **Node** active object. The server forwards the measurements to any connected web applications.

Conversely a web application can set the pitch and roll warning thresholds onto the **LevelMeter** active object (through the server and **Node** active object). The **LevelMeter** will then trigger an alarm to alert the end user. You may decide what kind of alarm to use, such as showing an LED pattern, displaying an LCD message, or changing text colors.

For the purpose of this application, the pitch and roll angles are ranged from -90 to +90 degree, with a precision of 2 decimal places (e.g. -45.12 degree, or 67.89 degree). You may use a full range of -180 to 180 degree but you would need to reduce the precision to 1 decimal place (due to LCD space limitation). You may also decide which rotational direction is positive.





2 Setup

2.1 Project Setup

1. Download the compressed project file (platform-stm32l475-disco-assignment6.tgz) from the Assignment 6 course site.

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

2. Backup your existing project folder, e.g.

   mv platform-stm32l475-disco platform-stm32l475-disco.bak

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

3. Decompress the tgz file with:

   tar xvzf platform-stm32l475-disco-assignment6.tgz

   The project folder will be expanded to **~/Projects/stm32/platform-stm32l475-disco**.

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

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

5. Clean and rebuild the project. Download it to the board and make sure it runs.

   On the LCD, you should see the following text:

   ```
   P= 000.00
   R= 000.00
   PT= 90.00
   RT= 90.00
   ```

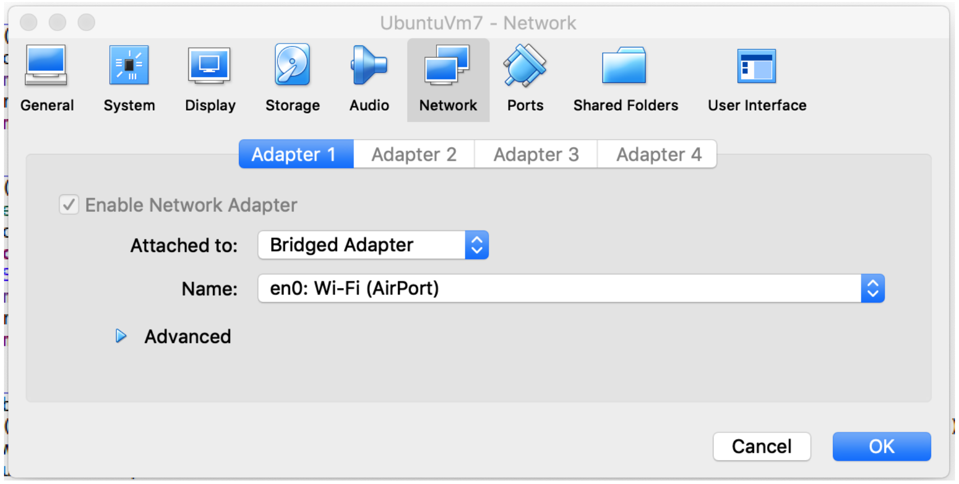
   The meaning of the above readings are:

   - P – Pitch in degree
   - R – Roll in degree
   - PT – Pitch Threshold beyond which an end user shall be alerted.
   - RT - Roll Threshold beyond which an end user shall be alerted.

2.2 Network Setup

To ensure data can be transferred between the STM32L475 discovery board and the VM, we need to check a few settings of our VM.

1. Ensure "**Bridged Adapter**" (rather than "NAT") is selected for the VM network adapter. It should be the *default setting*.



✓ Enable Network Adapter
Attached to: Bridged Adapter
Name: en0: Wi-Fi (AirPort)

2. Ensure firewall in your VM is disabled (default). You may check by typing:

   ```
   sudo ufw status
   ```

In case firewall is enabled, allow TCP ports 60000 and 60002 to go through by typing:

```
sudo ufw allow 60000
```

```
sudo ufw allow 60002
```

Port 60000 is used for the connection with a web application and port 60002 is used for the connection with an STM32L475 discovery board.

3. In a terminal window, note down the IP address of the network interface of your VM by typing:

   ```
   ifconfig
   ```

   The following is an example of the output. Note your interface name may be different, e.g. it may show "eth0" instead of "ens32". In my case, the IP address is **192.168.1.233**.

```
ens32     Link encap:Ethernet  HWaddr 00:0c:29:8d:18:6a
          inet addr:192.168.1.233  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::17c2:c186:53eb:5b4f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146082 errors:0 dropped:0 overruns:0 frame:0
          TX packets:155948 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:46790185 (46.7 MB)  TX bytes:112812196 (112.8 MB)
```

2.3 Server Setup

1. From the assignment site on Canvas, download the compressed file of the Node.js server application named **webserver-class.tgz**.

   Under **~/Projects**, create a subdirectory named **node** (if it doesn't exist already) and place the downloaded file under **~/Projects/node**:

   ```
   cd ~/Projects
   ```

   ```
   mkdir node
   ```

2. Decompress the tgz files under the **node** subdirectory:

   ```
   cd ~/Projects/node
   ```

   ```
   tar xvzf webserver-class.tgz
   ```

3. Launch the server by typing:

   ```
   cd ~/Projects/node/webserver-class/cmdapp/src
   ```

   ```
   node main.js
   ```

   Important – you must run the application under the src folder as shown above.

   The server prints out log messages in the terminal window. The last few lines looks like these:

   ```
         18780:86066250 <LOG> CmdSrv: Enter conn_root
   ```

```
18780:86066262 <LOG> CmdSrv: Enter auth_root
18780:86066262 <LOG> CmdSrv: Enter auth_idle
```

Keep the server running which is required for steps below. (To quit the server, hit Ctrl-C.)

2.4 Connection from STM32L475

1. In your STM32L475 Eclipse project, enter your WiFi credentials in **src/Wifi/NotCheckedIn/Credential.cpp**:

   ```
   char const *wifiSsid = "enter your ssid here";
   char const *wifiPwd = "enter your wifi password here";
   ```

2. Enter the IP address of your VM in **src/System/System.cpp**, e.g.

   ```
   #define SRV_DOMAIN      "192.168.1.233"  // Replace with your own VM's IP.
   #define SRV_PORT        60002
   ```

3. Rebuild and launch the application on your discovery board. It should join your WiFi access point and connect to the Node.js server started in Section 2.3 within a minute.

   Once connected, the minicom console will shows a lot of log messages such as:

   ```
   69839 NODE(15): Normal LEVEL_METER_DATA_IND from LEVEL_METER(37) seq=0
   69848 WIFI(14): Connected WIFI_WRITE_REQ from NODE(15) seq=208
   69848 WIFI(14): Calling ES_WIFI_SendData writeLen=84
   69858 WIFI(14): ES_WIFI_SendData return sentLen=84
   70172 NODE(15): Normal LEVEL_METER_DATA_IND from LEVEL_METER(37) seq=0
   70172 WIFI(14): Connected WIFI_WRITE_REQ from NODE(15) seq=209
   70172 WIFI(14): Calling ES_WIFI_SendData writeLen=84
   70186 WIFI(14): ES_WIFI_SendData return sentLen=84
   ...
   ```

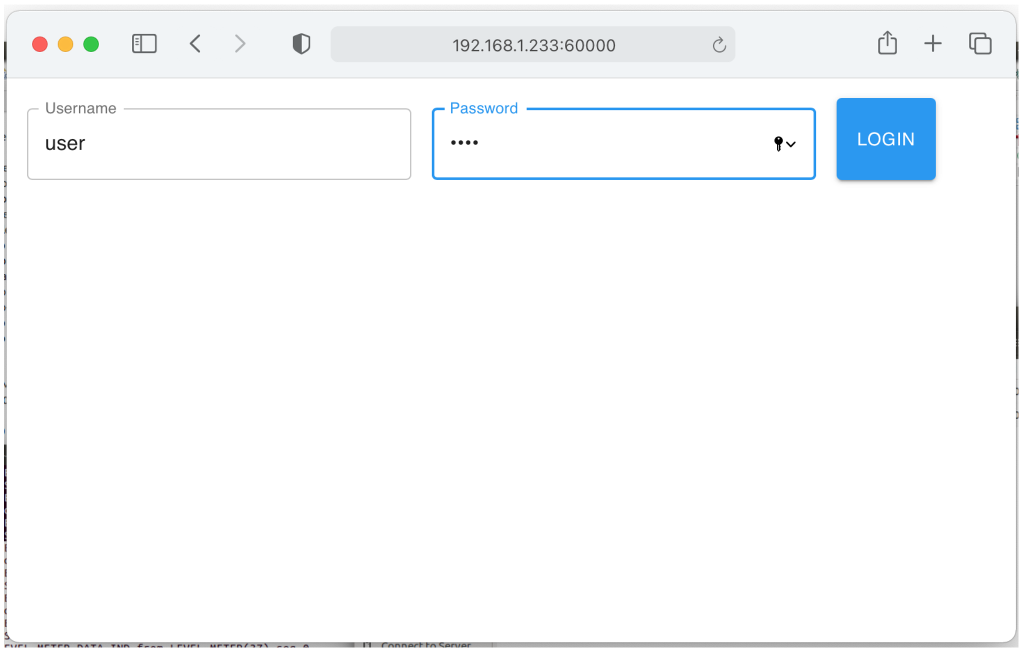
   You may disable these log messages by typing:

   ```
   log disable 14
   log disable 15
   ```

2.5 Connection from Web Application

1. In the address bar of your favorite web browser, enter the IP address of your VM server at port 60000, e.g. "**http://192.168.1.233:60000**". (Replace the IP address with the one you noted down in Section 2.2.)

2. You should see this following screen:

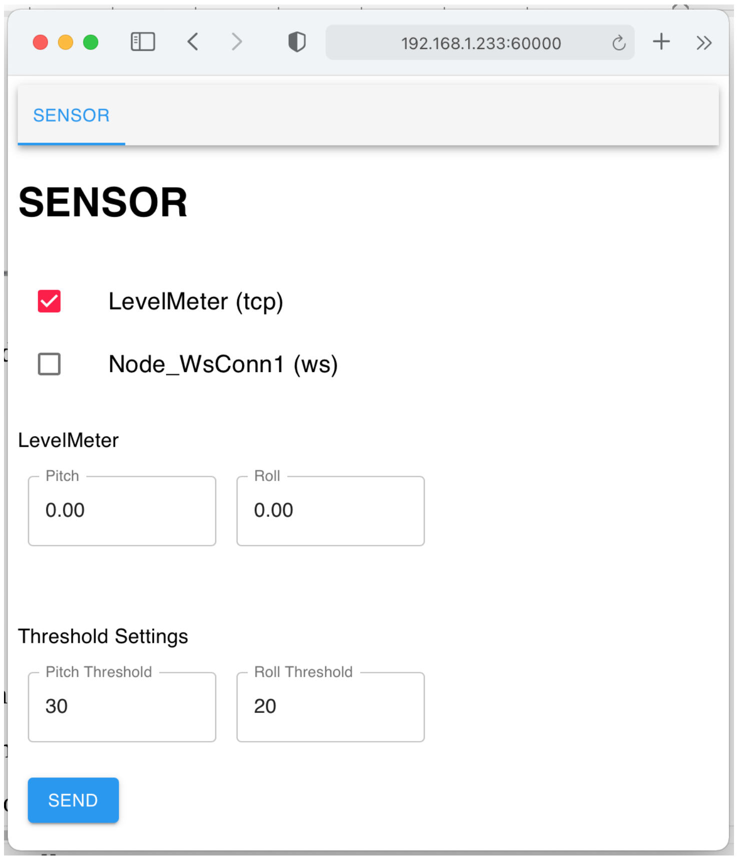

192.168.1.233:60000

Username: user

Password: ••••

LOGIN

3. Enter "**user**" as username and "**pwd**" as password. Click LOGIN. You should see the following page:


192.168.1.233:60000

SENSOR

SENSOR

- [✓] LevelMeter (tcp)
- [ ] Node_WsConn1 (ws)

LevelMeter

Pitch: 0.00

Roll: 0.00

Threshold Settings

Pitch Threshold: 30

Roll Threshold: 20

SEND

Note:

- The item "**LevelMeter (tcp)**" only appears if your discovery board has successfully connected to the server. If you don't see it, please check your connection setup. Try to place your discovery board closer to your access point for better signal conditions.
- Check the checkbox next to "**LevelMeter (tcp)**" to show the pitch and roll measurements sent by your discovery board. It currently shows 0.00 all the time.
- Enter the threshold values and click SEND. You should see the values appear on the LCD of your discovery board. (The LevelMeter checkbox must be checked for it to work.)
- Note – For some web browsers, putting the browser window in the background may cause connection timeout and bring it back to the login screen. Once logged back it, please make sure the LevelMeter checkbox is checked.

3 Tasks

After the above setup steps, we have verified the connection between the embedded device and a web application through a Node.js server. Now our tasks are:

1. Review the LevelMeter statechart and its implementation in **src/LevelMeter/LevelMeter.cpp**. Make sure you understand its behaviors and interaction with other state machines such as **SensorAccelGyro**, **Ili9341** and **Node**.
2. Calculate pitch and roll in **src/LevelMeter/LevelMeter.cpp**. It is marked with:

   ```
   // Assignment - Calculate pitch and roll from accelerometer X, Y and Z data.
   ```

   You will need to research for the formulae to convert raw x, y, z accelerometer data into pitch and roll angles.

   Note the readings from the accelerometer is in unit of 1/1000 G, i.e. a value of 1000 represents 1 G acceleration.
3. Design and implement pitch and roll threshold alarms. See the comment in code:

   ```
   // Assignment - Compares current measurement (m_pitch and m_roll) against the set thresholds.
   //              Displays an alarm (your own design choice) to alert the end-user, e.g.
   //              - Shows an LED pattern.
   //              - Shows a warning message or icon on the LCD.
   //              - Changes the text background/foreground colors on the LCD.
   //              The code may not be located right here.
   ```

4. Verify the expected pitch and roll readings are shown on the LCD display and web application. Verify alarm is triggered when a set threshold is exceeded.

5. Bonus (not required) – We don't have a calibration function yet. Can you design a simple calibration feature to set zero on a flat surface?

4 Submission

The due date is 6/14 Monday 11:59pm. Please submit:

1. A photo of the LCD showing the calculated pitch and roll readings.
2. A screenshot of the web application showing the calculated pitch and roll readings.
3. A photo or description of a threshold warning alarm.
4. A zip file containing the source code in **src/LevelMeter.**

Upload to the usual Canvas assignment upload location.