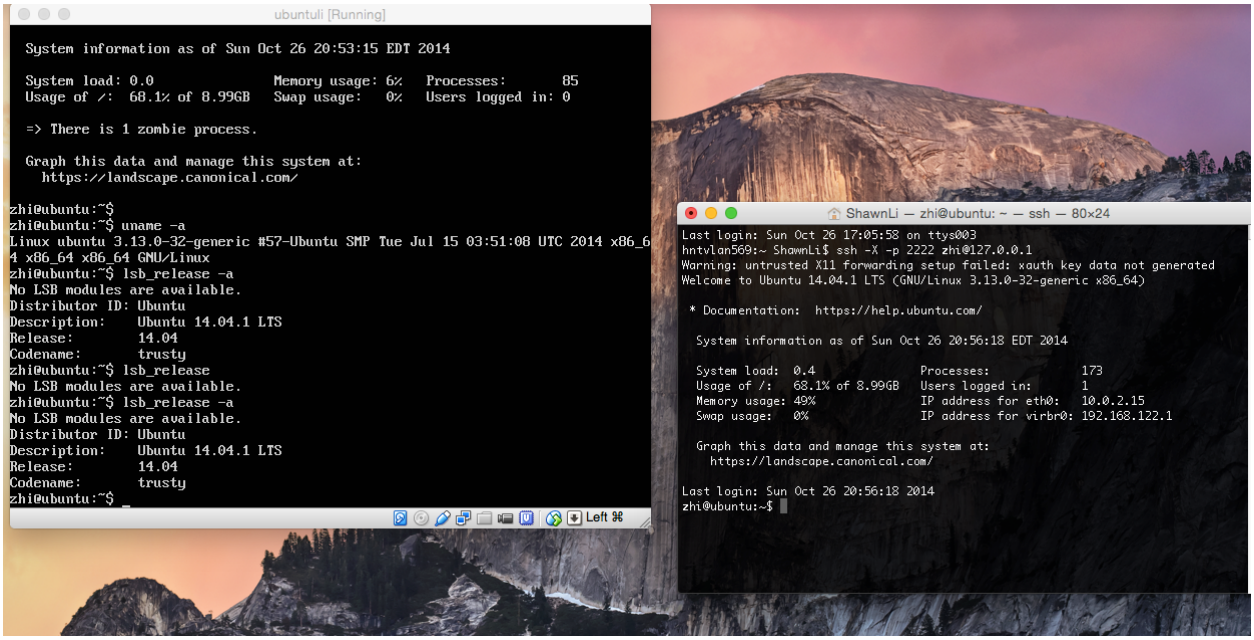
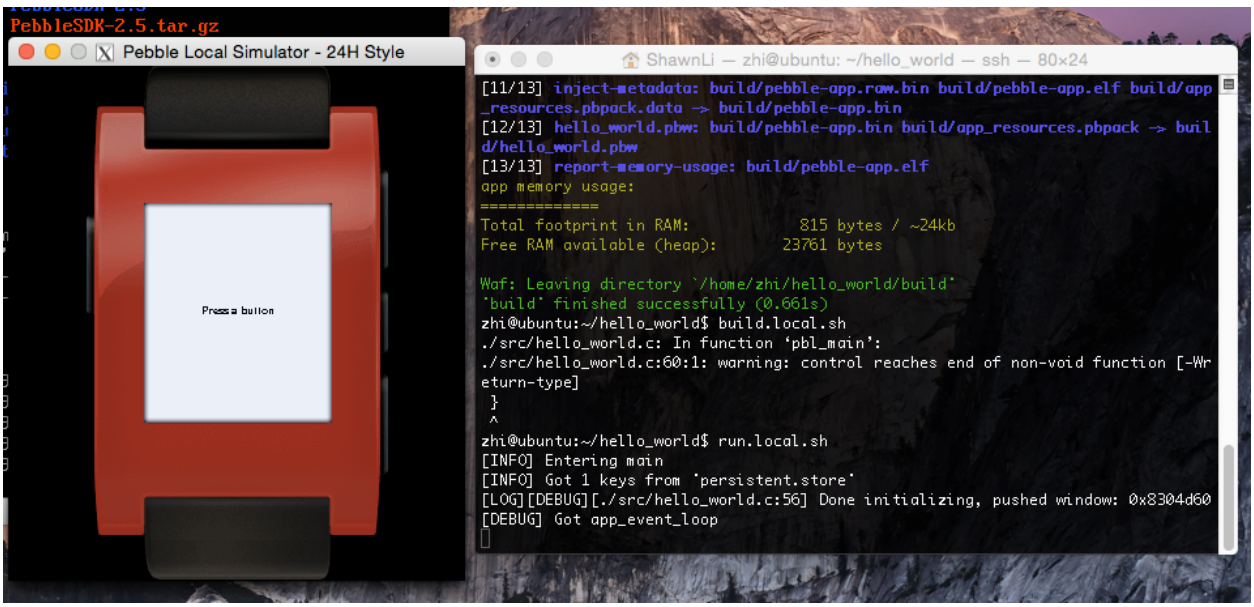
Section 1

requirement: write up and screen shots showing that setup is working

I use Ubuntu 14.04 server version, ssh -X over via Mac OS X 10.10 Yosemite, with X11 version XQuartz 2.7.7. A screen shot of ssh is given below.



The setup of pebble simulator is so much fun. Following the instructions provided by the instructor, I set up all the packages required, including pebble SDK, PebbleLocalSimulator from github, supportive python packages and environment configuration. I could use pebble build to build the hello_world program, but when I try to execute build.local.sh under the same directory of appinfo.json, I keep getting the error message of resource_ids.auto.h not defined. After attending one of the lab hours, I manage to solve the problem by switching to **PebbleSDK-2.5**.



And now the simulator could work just fine, you could find a screenshot of hello_world watchface above. Figure 2 contains the process of pebble build, build.local.sh, and run.local.sh. The watchface will pop out via X11 window.

Thus, now the simulator is properly installed, and my next move is to get the weather example working on the simulator, which indeed takes some time.

Section 2 (PebbleSDK 2.5)

requirement: the source code with comments included, screen shots.

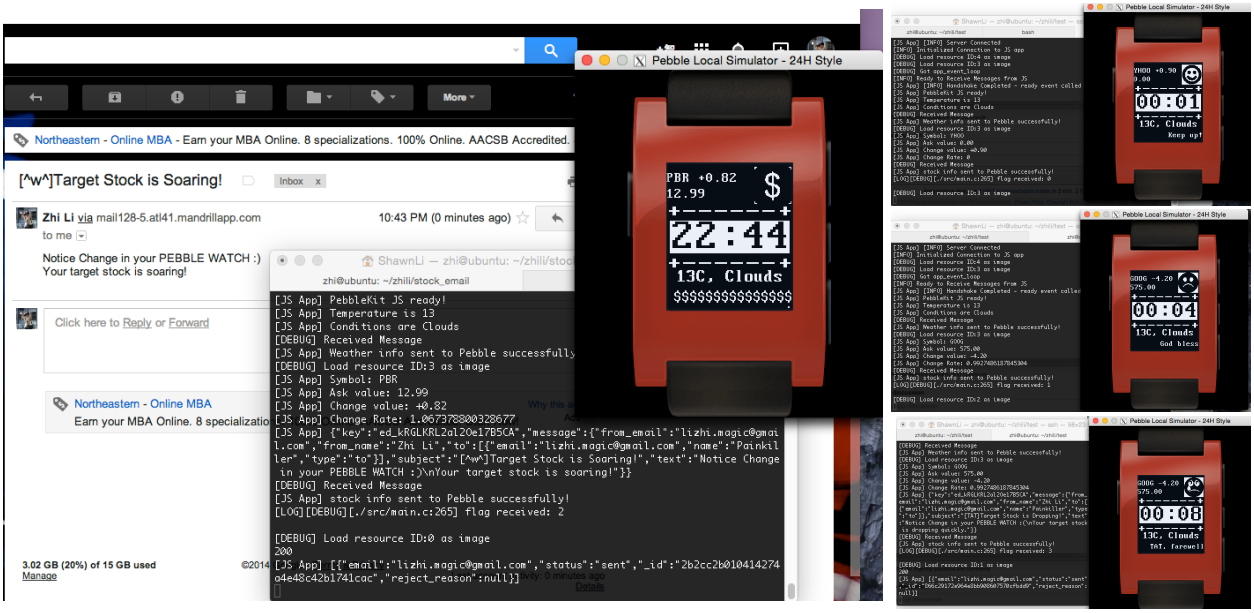
I grabbed the code from github, and went through every function call and commands to make sure everything in the weather example make sense, which was indeed time consuming. Essential comments are added in the js file as well as main.c. When trying to run the program via simulator, I met a general problem saying that node: command not found, failed to Initialize Connection to JS app. To solve the problem, I installed node using nvm, copied 'localStorage.js' from Simulator library to PebbleSDK-2.5/bin/localsim/simdata. Then I installed lodash through npm install lodash but still unable to get the program working until I add "window" before "navigator.geolocation.getCurrentPosition" in pebble-js-app.js. Since the simulator could not fetch location information via Cell phone, add window before it could get the job done. The following screenshot shows that the weather example can be run on the simulator properly, it could also work properly on the pebble watch, I forgot to take a picture because I then move on to testing the stock part, however, since the stock part is an updated version of the weather program, it still contains all former codes. Another thing I want to mention is that, the simulator is not always stable. I'm perfectly certain that there's nothing wrong with the code but sometimes the simulator refuse to work. Here's a solution to solve this. First reconnect to the virtual machine, run nvm install 0.11.13, then run node -v to check if program executable node can be called. If everything's fine then run.local.sh will do the job.

The source code is under directory **/weather**, with everything needed to build the watchface on either simulator or cloud pebble.

Section 3

requirement: the source code with comments included, screen shots.

The source code could be found under **/stock_email**, with everything needed to run the program on both simulator(server) and a real pebble watch. The following is a screenshot of program running on simulator, I set a trigger alarm that whenever the change of target stock value exceeds 5% (above/below) its original value, the display on the watchface will change "dramatically", the message box will be quite "noticeable", and an email will be sent to the user immediately. On the command line you could find data sent by HTTP POST request, and the info returned from the api. The background is my mailbox showing email sent in realtime. What I need to mention is that, in order to run on simulator, we still need to add "window." in front of "navigator.getLocation…", since no cell phone connection is enabled for now.



Here are more details about the software design.

<u>stock API usage</u>

Based on the weather example, I first try to find a financial API that could be used to fetch stock quotes info. Eventually I choose Yahoo's YQL via Yahoo finance. YQL enables developers to fetch a bunch of stock quotes information via specific url calls. For example, the url in the figure below is constructed to fetch "PBR", "YHOO", "NOK", "GOOG" and "MSFT". User could change the symbol (hardcoded in url) to fetch any feasible stock they want. Then xhrRequest function will send HTTP GET request to YQL and return a json object containing all stock quotes requested. The array json.query.results.quote[i] will contain i_th stock requested. Here one target of interest is chosen to display on watchface due to valuable and rather small space. Information of interest such as symbol, AskRealtime, ChangeRealtime will be further parsed

from the array and sent to pebble via keys (KEY_SYMBOL, KEY_CURRENT, KEY_CHANGE) in dictionary.

```
// construct query url
function stockSuccess() {
  // Construct URL, fetch data from "PBR","YHOO","NOK","GOOG","MSFT"
  var url = "https://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20yahoo.finance.quotes%20where
%20symbol%20in%20(%22PBR%22%2C%22YHOO%22%2C%22NOK%22%2C%22GOOG%22%2C%22MSFT%22)%0A
%09%09&format=json&diagnostics=true&env=http%3A%2F%2Fdatatables.org%2Falltables.env&callback=";

  // Send request to Yahoo Finance
  xhrRequest(url, 'GET',
    function(responseText) {
      // responseText contains a JSON object with stock quotes info
      var json = JSON.parse(responseText);
      // 0 for PBR, 1 for YHOO, 2 for AAPL, etc.
      var stock = json.query.results.quote[0];

      // record symbol information
      var symbol = stock.symbol;
      console.log("Symbol: " + symbol);
```

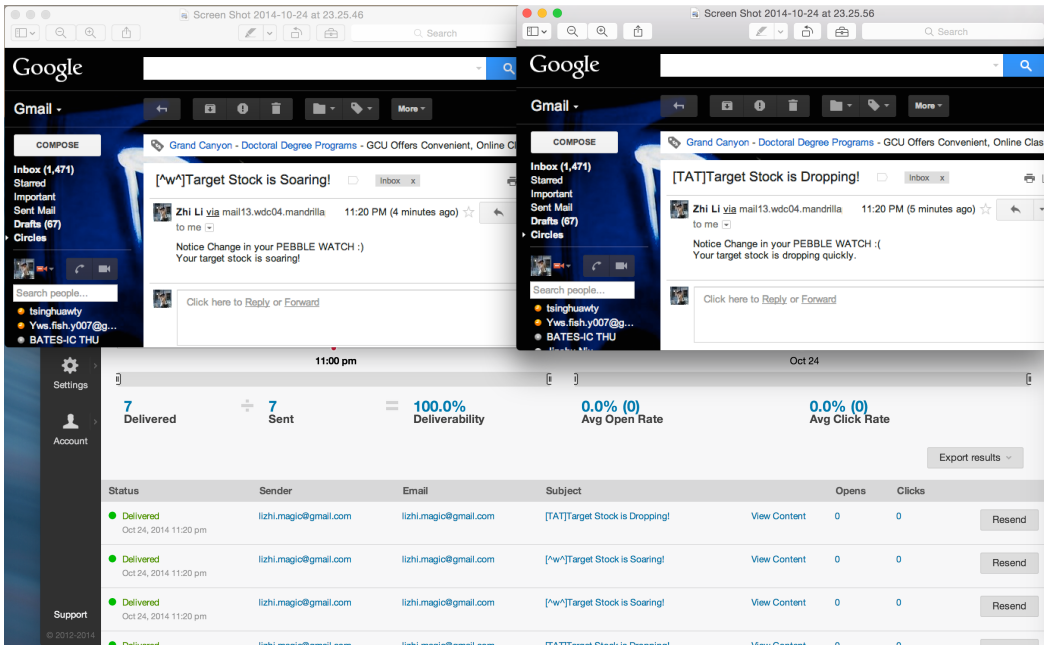
Face change (EC part1)

The software built contains four watchface layouts, including thrilled face, happy face, sad face, and frustrating face, referred to increase more than 5%, normal increase, normal decrease and decrease more than 5%. Noted that here if askValue equals zero, the display will be determined by whether change is positive/negative, and given happy/sad face.

I tried hard to calculate the change rate in main.c, but it seems that pebble C environment is not friendly with float number, making calculation and debugging rather difficult. The atof function(change string to float) will not work due to "_sbk not defined" related error. Thus, calculation is done under javascript environment. The option of face display is recorded in variable flag and sent to Pebble main.c via KEY_FACE, where it will be decoded and read by function face_change(s_main_window, flag), which will change the display of watchface and update message box to the following options according to change rate: "keep up!", "god bless", "TAT", "$$$$$".

Email Notification (EC part2)

In order to enable email notification on both simulator and pebble watch, I choose Mandrill mail server over nodemailer. Since although the latter could work under simulator environment, it won't work in real pebble watch. Thus, I figure out a way to use simple and clean HTTP post request to send email via Mandril API(no need Mandrill js library).



The mandrill main server will allow first 12,000 emails sent free. The screenshot above shows the notification emails sent in realtime and tracking information recorded by mandrill. In order to achieve the function above, mandrill requires user to register an API key with his/her email address. It also provides instructions on how to send email via mandrill api using mandrill library, but it seems that pebble doesn't support this kind of third-party library well. Also, on the internet an instruction on using ajax defined in JQuery is provided, but since normally JQuery is loaded in a corresponding html page, which I don't think is supported by pebble watch. Therefore, I need to come up my own way of doing this.

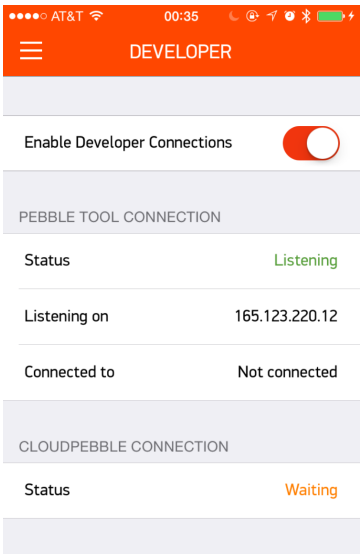
My solution is to use a simple and clean HTTP POST request to send email data to mandrill's API. A new js function emailNoticationUp/emailNotificationDown is added to pebble-js-app.js. The data array contains all essential data needed by mandrill API, including a user API key, message struct containing sender, receiver, subject and text. More features such as sent by predesigned template/html could work but we're not gonna talk about this now. Then JSON.stringify will format the data array to readable format required by HTTP POST Request (in a form of "key=abcd&message=…"). Next, just send the formatted data via HTTP POST request to target url (mandrill/…/send.json), the server will respond you and give proper feedback. You could find the successful communication between watch and weather, stock and mail servers on the next page.

```
}

// send email notifications when stock price increases more than 5%
function emailNotificationUp(){
    var data = {
    "key": "ed_kRGLKRL2al2Oe17B5CA",
    "message": {
      "from_email": "lizhi.magic@gmail.com",
      "from_name": "Zhi Li",
      "to": [{"email": "lizhi.magic@gmail.com", "name":"Painkiller","type":"to"}],
      "subject": "[^w^]Target Stock is Soaring!",
      "text": "Notice Change in your PEBBLE WATCH :)\nYour target stock is soaring!"
    }
  };

   var postdata = JSON.stringify(data);
   console.log(postdata);
   var x = new XMLHttpRequest();
   var url = "https://mandrillapp.com/api/1.0/messages/send.json";

   x.open('POST', url, true);
   x.onreadystatechange = function() {
        if (x.readyState == 4) {
        console.log(x.status);
        console.log(x.responseText);
        }
    };
  x.send(postdata);
}
```



APP LOG:

[PHONE] pebble-app.js:?: JS: starting app: 07E6AA29-566D-4592-99D2-9CAF8B7CEA33 stock&weather
[PHONE] pebble-app.js:?: app is ready: 1
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: PebbleKit JS ready!
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Temperature is 14
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Conditions are Clear
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Weather info sent to Pebble successfully!
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Symbol: PBR
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Ask value: 12.99
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Change value: +0.82
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: Change Rate: 1.067378800328677
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather:
{"key":"ed_kRGLKRL2al2Oe17B5CA","message":
{"from_email":"lizhi.magic@gmail.com","from_name":"Zhi Li","to":
[{"email":"lizhi.magic@gmail.com","name":"Painkiller","type":"to"}],"subject":"[^w^]Target Stock is Soaring!","text":"Notice Change in your PEBBLE WATCH :)\nYour target stock is soaring!"}}
[DEBUG] main.c:261: flag received: 2

[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: stock info sent to Pebble successfully!
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather: 200
[PHONE] pebble-app.js:?: JS: Pebble_stock&weather:
[{“email":"lizhi.magic@gmail.com","status":"sent","_id":"65d6d9dc597a4a7e885aa7fccd8b0655","reject_reason":null}]