Pebble Temperature App

CIT 595

Group 13

Technical Document

• what is the structure of the messages that are sent from the user interface to the middleware? give examples

The user interface creates a XML HTTP request each time to send information to the middleware. The url is formatted as “http://” + ipAddress + “:” + port + “/” + index.
The ipAddress number is hard coded each time the middleware changes its IP address. By default, the port number we are using is 3001, which is the same port specified as argument to server. The index is corresponding to the information we want to get from the middleware based on the click command we received from the user interface.
e.g. index and its request
“temperature”: current temperature read by the sensor
“average”: average temperature read by the sensor in the last 1 hour (or average for all the records read by the sensor if the sensor has been running for less than an hour)
“low”: the lowest temperature read by the sensor in the last 1 hour (or average for all the records read by the sensor if the sensor has been running for less than an hour)
“high”: the highest temperature read by the sensor in the last 1 hour (or average for all the records read by the sensor if the sensor has been running for less than an hour)
“F/C”: change the temperature readings to the other unit (if the current temperature is read in Celsius, then the index is going to change it to Fahrenheit, and vice verse)
“standby”: put the sensor into a stand-by mode
“wakeup”: resume the sensor and user interface to report the readings after standby mode is running
“+”: calibrate the sensor by increasing 1 Celsius of temperature
“-”: calibrate the sensor by decreasing 1 Celsius of temperature
“tempHistory”: ask the middleware to provide the temperature during the last 1 minute so that the user interface can then generate a temperature curve on the watch

• what is the structure of the messages that are sent from the middleware to the user interface? give examples

The middleware are sending back the message using a string, which is structured as
“{\n\”name\”: \”” +outmsg +”\”\n}\n”
The key “name” does not change, and the outmsg is the information needed from the user interface. The outmsg will be retrived by the user interface after it receives the whole string from the middleware.
e.g. outmsg
“32.1250 C”: if current temperature is wanted

“98.1230 F”: if current temperature is wanted
“avg: 32.03 C”: if average temperature is wanted
“min: 32.03 C”: if the lowest temperature is wanted
“max: 32.03 C”: if the highest temperature is wanted
“Standing by!”: the user interface and the sensor enter into stand-by mode
“Waked up!”: the user interface and the sensor resumed from stand-by mode
“Calibrated!”: calibration has been performance (temperature increased or decreased)
max temperature + min temperature + 10* temperature in last 1 minute (6 seconds interval): the user interface want to show the temperature curve in the last 1 minute.
e.g. “23.25,22.56,22.56,22.62,22.69,22.81,22.88,23.00,23.06,23.06,23.19,23.25,” would be sent
“Sensor Disconnected!”: if the server and the sensor become disconnected

• what is the structure of the messages that are sent from the middleware to the sensor/display? give examples
To simplify the message sent from the middleware to the user interface, each message only contain 1 char variable. By checking the sigle character, the user interface could respond rapidly. There are 6 characters defined that would be sent out from the middleware.
‘s’: sleep-the sensor would go to standby mode (no temperature showing on display)
‘w’: wakeup-the sensor would start to show the temperature again on display
‘f’: change the temperature display in Fahrenheit
‘c’: change the temperature display in Celsius
‘-’: calibrate the temperature, reduce the current observed temperature in sensor of 1 degree Celsius
‘+’: calibrate the temperature, increase the current observed temperature in sensor of 1 degree Celsius

• what is the structure of the messages that are sent from the sensor/display to the middleware? give examples
There is only one type of message sent from the sensor to the middleware, which is a string. The string is structured as floating temperature+” “+(C)/(F).
The maximum decimal places are 4 for the temperature.
For example, a message like “32.6250 C”, “90.2340 F”, “32.00 C” will be sent out.
As the request from server changes (indicating the change between Celsuis and Fahrenheit), the sensor will send the temperature ending with “C” or “F” respectively.

• how did you keep track of the average temperature? describe your algorithm and indicate which part of your code (including line numbers) implements this feature
Since the sensor sends out the temperature to middleware each second, an array with size of 3600 is created to record the temperature in the last hour. All temperatures are recorded in Celsius. If the array is full, the new temperature will be stored looping back to the head of the array. Once the middleware receives a temperature request, max temperature, min temperature, and average temperature will be calculated. If the array is not full, the temperature from index 0 to current index will be looped to calculate the average temperature. If the array is already full, the temperature will be looped from next index to the end of the array and then from the head of the array to the current index to calculate the average temperature. Also, if the user request is in Fahrenheit mode, the average temperature will be converted into Fahrenheit.

Server:
project.c-line 115-167: calculating the highest, lowest, and average temperature
line 323-351: sending out the average temperature to user under request

Pebble:
app.js-line 15-17: implement the index for the average temperature (it is in the loop of showing temperature related number records)
main.c-line 210-224: click function added to show the average temperature

• what are the three additional features that you implemented? indicate which parts of your code (including line numbers) implement these features

Additional Features:
1. The user will be able to calibrate the temperature detected by the sensor. That is to say, if the user know the real temperature, the user can ask the sensor/arduino to adjust the output according to the real temperature.
2. The watch will alarm if the temperature is too high.
3. The watch should be able to show the history curve of temperature for the past 1 minute.

Arduino:
temperature.ino-line140-141: check if degree need to be incremented or decremented

Pebble:
app.js-line 31-36: implement the index for the three features
main.c-line 51-95: the function to write a temperature curve on a canvas layer
main.c-line 113-195: function in_received_handler modified in order to show the curve on the watch by adding/removing layers from the base window
main.c-line 272-284: click function added to show the temperature curve in last one minute
main.c-line145-170: check if the current temperature is “too high”, if it is, then vibrate to alarm
main.c-line 287-300: click function added to calibrate the temperature (increase)
main.c-line 302-315: click function added to calibrate the temperature (decrease)

Server:
project.c-line 233-261: change the flags and send responses to user after receive the calibration request;
line 474-488: send the request to sensor for calibrating temperature output
line 353-386: send the highest temperature, lowest temperature, and all temperature records in the last minute to user for ploting history curve