CIS 595 Project Group 15

Group Member: Rajveer Parikh, Jincheng Cao, Yuting Tan

Introduction

In this project, our group created a Pebble smartwatch applications that is able to get data from and send control information to a remote sensor and display driven by an Arduino microcontroller.

Technical documentation

1.Message structure from the user interface to the middleware

When user select press a button on Pebble watch, a single character message (e.g. “b”) is sent to the phone using key#0. In each page, upper, center and bottom buttons are bound to different characters. The phone connects to the server over the internet. The javascript code on CloudPebble will create an HTTP request with specific URL: http://" + ipAddress + ":" + port + "/" + message and send it to the server using "req.send". The ipAddress and port variable are modified to refer to the machine on which the server is running.

Page#	UpperButton	CenterButton	BottomButton
1	“a”-- Change degree	“b”-- Print temperature	“c”-- Display temperature page
2	“d”-- Not used	“e”-- Display PM 2.5 info	“f”--Display PM 2.5 info page
3	“g”-- Flash color to Green	“h”-- Flash color to Purple	“i”--Flash green or purple light page
4	“j”-- Threshold value up	“k”-- Threashold value down	“l”--Change threshold value page
5	“m”--Display avg temp	“n”--Display high/low temp	“o”--Display high/low and avg temperature page
6	“p”-- Enter standby mode	“q”--Quit standby mode	“r”--standby mode page

2.Message structure from the middleware to the user interface

In javascript code, "req.onload" defines a function that is called after the HTTP request is sent and then receives a response, which is a JSON object consists of key/value pairs. The function looks in the response for the key called "name" and send it to the Pebble usingPebble.sendAppMessage function if the key exists.

3.Message structure from the middleware to the sensor/display

The server(midddleware) sends control signal containing 1 character to the sensor. The Arduino then use Serial.read() to get the command. The following table represent the signal to the corresponding command.

Character	Command
0	change to Fahrenheit
1	change to Celsius
p	flash purple light
g	flash green light
u	threshold up
d	threshold down
s	enter standby mode
t	quit stanby mode

4.Message structure from the sensor/display to the middleware

The only data sent from sensor/display to server is the temperature in each second. Even if the temperature is displaying in Fahrenheit, it will sent the celsius value.

5.Average temperature inplementation

A global double array of size 3600 is used to store temperature values generated. If more than 1 hour passed, it will rewrite the previous values. When the server needs to return average temperature, it just call a function void calculateAverage(). This function iterates through array to calculate the temperature. If it has not get full before 1 hour, it will stop at some point according to a counter.

6.Additional features

The three additional features we implemented are:

1. Displaying PM 2.5 information: PM 2.5 means particulate matter of size less than 2.5 micrometers. Through our javascript code, we connect to an external server to display the amount of PM 2.5 particles in the air.
2. Flashing lights on the Arduino: Our watch allows the user to flash a green or purple light on the arduino. When the button for green or purple light is selected, our main server code writes the values ‘p’ for purple or ‘g’ for green to the file descriptor for the arduino. In our arduino code, we check for the particular value (which is the ascii equivalent of ‘p’ or ‘q’) and change the color of the light accordingly.
3. Allowing user to change the threashold value and change the color of the lights accordingly: We have set a range of 3 between the hot and cold temperatures and allow the users to increase or decrease the high and low values by pressing specific buttons on the watch. When the user presses the up button, we increase both high and low values by writing ‘u’ to the file descriptor for the arduino. On the arduino code, we change the value of the high and low accordinly. We similarly write ‘d’ to the file descriptor for

the arduino when the down button is pressed and decrease both high and low values accordingly. The range remains the same. Our ardunio code also simultaneously checks the temperature reading from the arduino. If the temperature read is lower than the low (‘cold) value of the threashold, the arduino will display a blue light, if it is higher than the high (‘hot’) value of the threashold, the arduino will display a red light and if it is within the range (between ‘hot’ and ‘cold’), the arduino will display a green light.

User documentation

Following is a description of how to use the our smartwatch application on the pebble smartwatch. All keys being used are the right hand side of the watch.

Once the smart watch is powered on, scroll all the way down using the key on the bottom right of the watch to get to our application called ‘Hello World’. Click the center key to enter the app to enter the first page

1. Display temperature: The first page of the app is to display the current temperature.
    a. Press the center key to display the temperature
    b. Press the top key change the degree of the temperature reading
        i. Press once to change from C to F
        ii. Press again to change from F back to C
    c. Press the down key to go to the next page
2. Display PM 2.5 information: The second page of the app is to display the amount of PM 2.5 particles in the air.
    a. Press the center key to display the PM 2.5 information
    b. Up button will not do anything
    c. Press the down key to go to the next page
3. Flash lights on the arduino: The third page of the app allows the user to flash either a green or purple light on the arduino.
    a. Press the center key for the arduino to flash a purple light
    b. Press the up key for the arduino to flash a green light
    c. Press the down key to go to the next page
4. Change threashold value: The fourth page of the app allows user to display and change the threashold values. The high (‘hot’) and low (‘cold’) can only be changed simultaneoulsy - the range cannot be changed.
    a. Press the center key to decrease both the hot and cold values of the threashold by one degree each
    b. Press the up key to increase both the hot and cold values of the threashold by one degree each
    c. Press the down key to go to the next page
5. Display high/low and average temperature: The fifth page of the app allows users to display the highest, lowest and average temperature readings within the last hour.
    a. Press the center key to display the high/low temperature readings, read by the arduino within the last hour

    b. Press the up key to display the average temperature readings, read by the arduino within the last hour
    c. Press the down key to go to the next page
6. Standby Mode: The sixth and final page of the app allows users to enter/exit standby mode. In the standby mode, the arduino will stop reading temperatures. The user will be allowed to move back and forth between the other pages of the app, but the watch will not perform any arduino related functionalities as long as it is in standby mode.
    a. Press the center key for the watch to enter standby mode
    b. Press the up key for the watch to exit standby mode
    c. Press the down button to get to the end of the app
7. End of App: This is the last page of the app. Displays messgae telling the user they are at the end of the app and to press down to get back to the first page of the app.
    a. Center button has no function
    b. Up button has no function
    c. Press the down button to get back to the first page of the app