pebble

CIT 595 Final Project Report

04.22.2016

TianXiang Dong

LiJun Mao

Yi-ping Lin

Overview

This project is a pebble watch project implemented with C/C++ language / JavaScript / Arduino.

User Manual

Step 1. Connect the arduino with USB Port (**usbmodem1411**)

Step 2. On the Linux Terminal, after the standard compilation "**make**", you should direct run the obj file called "**server**" with specific port number: **./server 3001**.

Step 3. Start the Pebble Server/Handset. The default ip address is **158.130.111.87**. If you decide to change the ip address, please hard code line 15 (**var ipAddress = "158.130.111.87";**) in app.js

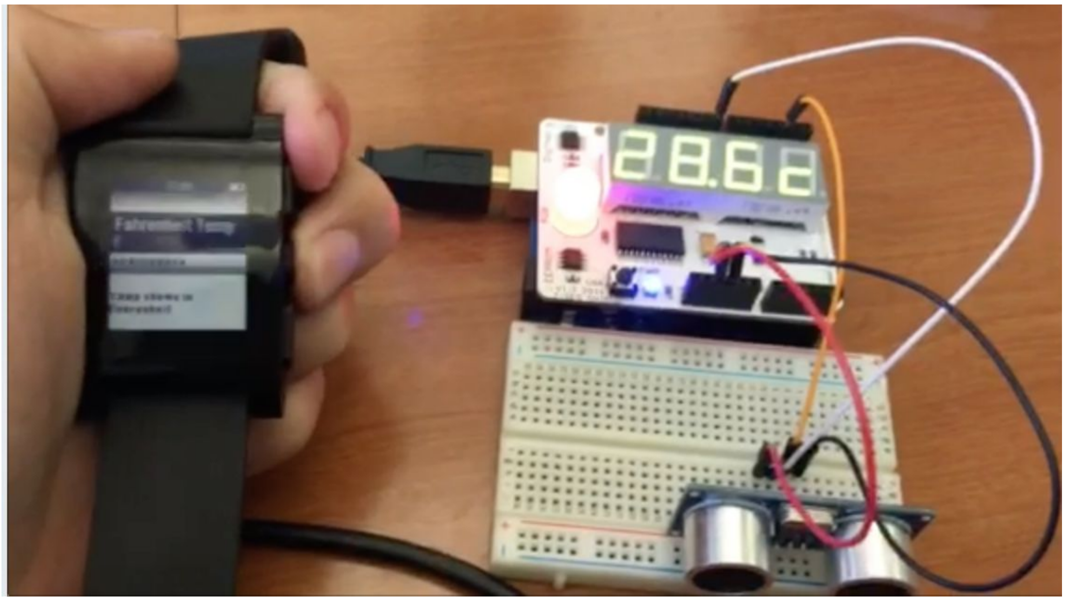
Step 4. You may now use the menu on Pebble Watch to select operations:

(1) get values: **Current Temp/ Avg Temp/ Highest Temp/ Lowest Temp**

(2) change mode: **Celsius Temp/ Fahrenheit Temp**

(3) start/pause the system: **Stand By/ Active**

(4) Extra features: **Phili Weather/ Show Outdoor/ Motion Sensor**

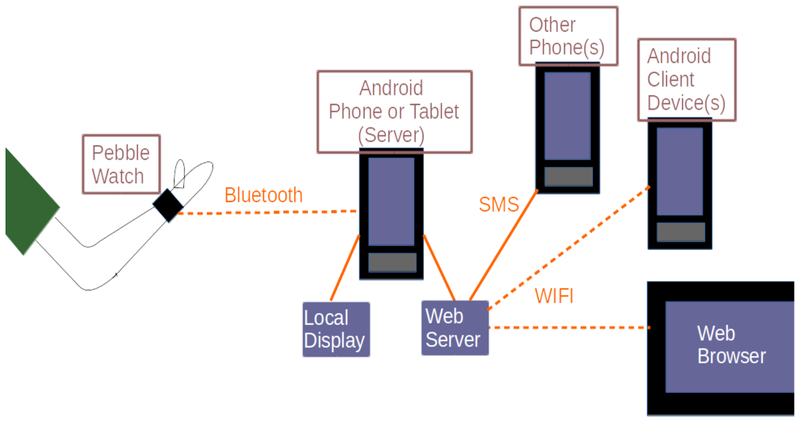


Architecture

The project is consisted of three layer: (1) Pebble front-end (2) Linux Terminal middleware (3) Arduino Sensor

Pebble front-end

It is the front-end device allowing users to give command with a menu UI. It also prints out the response message from the server on its screen.

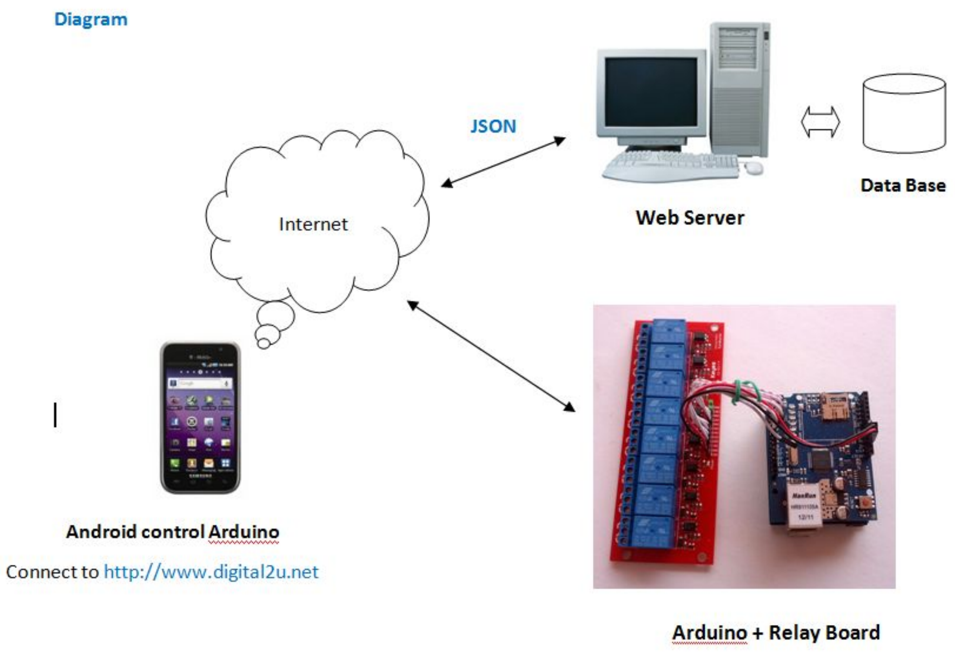


Linux Terminal middleware

It is a C language program running on Linux terminal. The main tasks are (1) receiving/sending messages from/to Arduino and Pebble (2) store and calculate the temperature values for temperature request from Pebble watch (3) integrate the API function

Arduino Sensor

It is an electronic board consisting of a temperature sensor and a motion sensor. We use it to detect the change of the environment and inform the middleware server.



Protocols

For each type of message, we use different protocol to communicate with other members.

Arduino-to-Server:

The protocol between Arduino board and server is based on USB port. We use the functions read(fd, buf, buf_length) & write(fd, request, request_length) to communicate.

The server sends Arduino the message string like:

(1) mode change: **“0”** for Celsius, **“1”** for Fahrenheit
(2) Philly Temperature(API): **” # \0 XX.X*2”**

Pebble-to-Server:

The protocol between Pebble & Server is based on TCP mechanism. We use the functions recv(fd, request, request_length, 0) & send(fd, reply, reply_length, 0) to communicate.

Pebble watch sends message string includes the action command ***(/CurTemp /AvgTemp /MaxTemp*** ...etc)

The server sends back the message string: “{\n\"name\": \"***Response Message***\"\n}\n“. Response Message could be:

(1) status update ***"Stand-by Mode"***,

(2) return request temperature value ***“XX F”***, ***“XX C”***,

(3) mode change ***"Now in Celsius Mode"***, ***"Now in Fahrenheit Mode"***,

(4) motion sensor ***"Now Active"***, ***"No motion detected."***, ***"Motion Detected!!"***, ***"Arduino Disconnected"***.

Explanation

Your group needs to submit a document that explains how your system works. Please include brief explanations of each of the following:

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

There are totally ten different requests which can be sorted into two types of messages sent from user interface to middleware. One type is getting temperature information message and the other is control message. The messages sent from user interface to cloud pebble is a descriptive keyword string. It first goes to the cloud pebble via our phone, and then send XMLHttpRequest to server with IP address and port number information appended with that descriptive word string. The request is sent to the server, waiting for information sending back. For example, “/curTemp” means current temperature, and asks server to send back current temperature from the sensor every three seconds which acts as updating itself. "/AvgTemp" represents the average temperature from setting up the Arduino board till now, and wait for the average value response from server. “/MaxTemp” and “ /MinTemp ” ask server to send back maximum and minimum temperature from the start. All of the above messages are get temperature information requests.

“/Celsius” and “/fahrenheit” are control command indicating ways of representing temperature. “/Celsius” means sent back all temperature information in Celsius degree while updating arduino display into Celsius. “/Fahrenheit” means sent back all temperature information in fahrenheit degree and updating arduino display into fahrenheit. “/Off” and ”/on” are also a pair of opponent command sent to server which means to turn on or off the arduino display. The “/off” here means turn off, and “/on” means turn on. The above request are control requests on Arduino or pebble watch display.

For additional feature part, "whether" means the outdoor temperature of philadelphia, we use the API address and secret key as URL and send XMLHttpRequest, and then wait for outdoor temperature information from API on http://openweathermap.org/. This message request is only sent to Open Weather API. Similarly, "outdoor" means the outdoor temperature of Philadelphia, and wait for response from Open Weather API. However, after it receives data, it also send an request to the server with messages "whether" appended with the data string. For example: "/whether:26.25" represents the weather request with outdoor information which is 26.25C. This request waits for no response. "/Motion" means detecting the motion sensor. This message request is sent to server per second, and wait for information on whether there is motion for the motion sensor. Therefore, "/motion" and "/whether" message requests get information from middleware. "/outdoor" is control request that controls the display of Arduino.

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

All messages sent from middleware to the user interface is JSON object, which is a key value pair. We use "name" as key and messages pebble watch waits for as the value. For example, if the user interface requests current temperature, middleware would sent a JSON object like "name: 26.765 C". "Name" represents key, and 26.765C is the value to be displayed on pebble watch. Similarly, if we request average temperature, lowest temperature or highest temperature, we would get JSON object with same format as "name: 27.250C", where 27.250C represents the kind of temperature we wait for in Celsius degree.

If user sends out control command requests as "celsius", "fahrenheit", "on" and "off". There will be no JSON object sent from middleware to user interface. However, if we requests "fahrenheit", we would get temperature data in fahrenheit degree such as "name: 87.150F". "87.150F" is the fahrenheit degree to be displayed on pebble watch.

For additional features, when we request Philadelphia whether, we would get a large JSON object from open weather API. We can get the temperature information with the keys main and temp. If we request for motion sensor detection, we would also get JSON object like "name:no motion detected" or "name: motion detected".

Therefore, all the messages sent from middleware to user interface are JSON object with "name" as key and temperature or motion sensor information as value. Except that the open weather API send a large JSON object containing the temperature.

What is the structure of the messages that are sent from the middleware to the sensor/display? give examples

There are three types of messages that are sent from middleware to sensor. The first is the message switching between Celsius and Fahrenheit. The second is the message turning on or off the stand-by mode. The last one is the message displaying outdoor temperature received through API on Arduino.

1. For the message of degree mode choice, all that sent by middleware to sensor is just a single flag, like 0 for Celsius mode, 1 for Fahrenheit mode.
2. For the message of standby mode, the flag is just 2. When the sensor receives the single sign of 2 from middleware, it simply displays four minus symbols and turns off the light, looking like it has been "turned off", during which it still sends temperature data to middleware. The display doesn't change until another message of sign 2 sent from middleware, making all the display return to normal.
3. For the message controlling the displaying of outdoor temperature, it's a little different. At first, the server receives the request from watch, as well as the outdoor temperature data gain through API on http://openweathermap.org/. While sending message to the sensor, the message like "#19.34*3" is sent. It's an easy encoding for message. The sensor receives the sign '#' first, and it knows the upcoming data is a double-type temperature data, then it initials related variables for receiving data. The sign '*' means the temperature data transferring ends. And the last sign 3 means the server requires the sensor to get into displaying outdoor temperature mode. This mode can also be affected by standby mode control and C/F mode switch. The sensor will get back to displaying indoor temperature data when another sign 3 comes in.

What is the structure of the messages that are sent from the sensor/display to the middleware? give examples

There are totally two different messages sent from sensor/display to the middleware. The message is sent every one second. It's something like "0 The temperature is 29.9375 degree C".

1. The first integer(0 or 1) in the returned message represents the value detected by the motion sensor. The integer becomes 1 when any motion is detected. The value gets back to 0 when no motion is detected in seconds since the last detection.
2. All the message after the first integer is the temperature message. What matters is the temperature data, which is always in Celsius degree.

3. Since the message is only sent every one second, there might be some latency between any motion is detected by the motion sensor and the motion detection message is displayed on the pebble watch.

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

For all the temperature data, we use an array, which has the size of 3600 to store the temperature data in the most recent hour. When the system runs more than one hour, the recent incoming data would replace the least recently used data in this array. There is also a variable specifically used to store the average temperature data.

In 20-25 lines, we define the array and related variables as global. From line 115-139, we get the temperature data from sensor and then put it into array, during which we complete all the computation, for instance the data of average, the highest and the lowest temperature.

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

First feature is the pebble watch can update the temperature automatically. We use a global AppTimer in main.c, then the update_temperature callback function will send request to the server whenever the current temperature is clicked. In the update_temperature function, we register the timer with the timer callback function. Whenever the updating flag is true, it will recursively register the timer and call itself. Therefore, it will repeatedly send request to server and update the temperature on pebble watch. The code lines implementing this feature is between 180-213 in main.c.

Second feature is the pebble watch can get access to the outdoor temperature in Philly through API on http://openweathermap.org/. We add weather API web address with our key number. Then parse the large JSON object and get the temperature information. After that, we calculate the celsius temperature with the given absolute temperature and display it on the pebble watch. The code lines executing the function is between 68-101 in app.js. Meanwhile, the pebble watch could send the outdoor temperature to sensor so that the display is always the outdoor temperature until the mode has been turned off. Of course, at this mode, the F/C switch still works, as well as the standby mode control. In the middleware, the code lines for receiving mode control request from pebble watch is from line 318-321. The code for sending control message from middleware to sensor is from line 205-218.

Third feature is the motion sensor set on the Arduino board. When the pebble watch get into the motion sensor mode, if there is any motion detected by the motion sensor, watch would display the warning. We set the pin of motion sensor as 2 in Arduino board, which is different from the Red(3), Green(5), Blue(6) light. The detection result would be sent to middleware using Serial.print() method, which is the first integer as mentioned in the last question. The middleware receives the data from sensor in line of 93-113. It sends the detection information while receiving the request from watch in line of 430-433 and 322-324, respectively.

Conclusion

Through this final project, we have implemented a system consisting of a front-end mobile device, a middleware linux program and an embedded program on Arduino devices. We fulfilled the goal of the project: to build a system which can monitor the change of the environment and display the information on the Pebble Watch.