**Technical Documentation**

**Pebble App is a Pebble smartwatch application that is able to get data from and send control information to a remote sensor and display driven by an Arduino microcontroller. This document explains the technical detail of the Pebble App system.**

**System Architecture**

**The system consists of three major components sensor/display, middleware and user interface.**

**Sensor and Display: A 4-digit 7-Segment Shield is used for display temperature and time. A temperature sensor is used as temperature sensing device. A RGB light is used for light switching. All the above components are attached to the Arduino microcontroller.**

**Middleware: The Arduino microcontroller with the sensor and display components attached is used as middleware. The microcontroller is connected to Mac machine via USB, to run a C++ program. The middleware handles all communication between the sensor and the user interface.**

**User interface: A Pebble smartwatch will be the user interface, which communicates with the middleware over the Internet via an iOS phone.**

**Inter-component communication**

**User interface to the Middleware:**

**Pebble watch, sends data to an iOS phone via Bluetooth, the iOS phone application then switch the data to middle via Internet by doing http POST request to the server on Mac (which is connected to the middleware via USB) and the server then switch the data to middleware via serial communication.**

**For example, when the user single click SELECT button of the Pebble watch to display the temperature statistic for the past one hour, the watch will send the message ‘b’ by the http POST request to the server via the iOS phone, who then switch the message to Arduino through the serial port /dev/cu.usbmodem1421.**

**Middleware to User Interface:**

**Arduino keeps sending message to the server on a regular basis via serial port. When the server receive a message from the user interface, a read thread reads desired reply message from Arduino, and will send the reply to the Pebble watch over the socket.**

**For example, when the server receive a message ‘b’ from the user interface to ask for temperature statistic for the past one hour, the server will read current sensor read from Arduino to update the record and sends the record back to the user interface over socket.**

**Middleware to Sensor/Display:**

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

**Sensor/Display to Middleware:**

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

**Arduino Algorithm**

* how did you keep track of the average temperature? describe your algorithm and indicate which part of your code implements this feature

**Additional Features**

* what are the three additional features that you implemented? indicate which parts of your code implement these features

**Besides the basic features, we also implemented three additional features:**

**1. Put the Arduino board into party mode in which the RGB light switch between seven colors.**

**2. Get stop watch counting in range 0 to 99 seconds and have the counted time displayed on Arduino.**

**3. Get the historical average, low, and high sensor reading for the period that the app has been working so far.**