Wake-Up Manager Application

1.Installation setup

1) Download the required android project in GitHub repository specified in the following links:
   a) Context Aware Application: https://github.ncsu.edu/tyeh3/csc450project2f1f
   b) Context Middleware: https://github.ncsu.edu/tyeh3/ContextMiddleware
2) Unzip the downloaded zipped file.
3) Open android studio and go to “File” -> “Open”. Select the android studio project for context aware application and context middleware.
4) Make sure to run the context middleware first and then context aware application. The context aware application should automatically connect to the middleware after it launches.

Note: When you FIRST run context aware application it will crash. Learn how to resolve this problem in the next section “Pre testing configuration”.

Note: I have verified the installation setup on my own Mac machine, let me know if you have any problem by sending email to tyeh3@ncsu.edu. I will be willing to schedule a time with you to meet and install the application for you.

Note: The following sections (2,3,4,5) are all from f1e but they are still useful and necessary to follow. However, to distinguish the difference between f1f and f1e I create new section 3.1 and section 4.1.

2.Pre-testing configuration (From f1e)

The application compiles fine but fails in running because the permission for the application to access several services need to be enable on the android device.

Go to **Setting -> Apps -> CSC450Project2 -> Permissions**, enable all the permissions for the application (Calendar, Location, Microphone and Storage).

Relaunch the application from android studio. The application should launch successfully and the user interface should be displayed without any problem.

The following configurations need to be performed before the verification process to avoid unexpected behavior:

1) Turn off device auto rotation: Go to **Setting->Device menu -> Display -> When device is rotated** -> Set to be stay in portrait view.
2) Make sure the android device is connected to the Internet (Wi-Fi or data plan).
3) Setting up the google account in the android device so that the application can have access to user's Google calendar. Go to **Setting->Accounts->Add account->Google**. Enter the following information as prompted. Email: sampleusercsc450@gmail.com Password: samplecsc450. Accept the term of service. Tap into the account you have just created then **make sure that the calendar option is enabled**. In order to add events to the calendar, go to **Calendar** on android device and you should be able to add the event.

3.Verification process (From f1e)

1) User Interface
   a) First line displays the ambient noise around the android device.
   b) Second line displays the light intensity captured by light sensor on the android device.
   c) Third line displays the user's current location.
   d) Fourth line displays count down for activity detection.
   e) Fifth line displays the result for the activity detection.
   f) Six line displays the count down for the alarm if created.
   g) Button "stop alarm" stops the ringtone played by android device when alarm happens.
   h) Button "test version" set the recommended sleep duration to be 15 seconds (default is 8 hours and you will have to wait for a long time for the alarm to happen).
   i) Button "restart detect" restarts the activity detection after the alarm happens.
2) Calendar events
   a) In order to add events to the calendar for the application to read, go to **Calendar** on the android device and edit events for sampleusercsc450@gmail.com.
   b) The application will retrieve and analyze the events from the calendar **ONLY** for the following day (If you test on Nov.13 2016 then the events need to be added to happen on Nov.14 2016 for the application to capture).
3) Test version
   a) Hit the "Test Version" button **BEFORE the alarm is set** will make the recommended sleep duration to be 15 seconds. If you want to restore the recommended sleep duration to 8 hours, you have to restart the application.
4) General Program flow
   a) As the program starts, there will be 5 seconds delay before the start of activity detection count down.
   b) After 30 seconds count down the user context (location, ambient noise and light intensity) is captured and the following decision is made:
      i) If user location does not change AND light intensity < 10lux AND ambient noise < 20000, then the alarm is created.
      ii) Otherwise the alarm will not be created and the reasoning is displayed.
   c) When the alarm is created, the countdown will be displayed on screen and when the alarm fires, one of the option is chosen according to user context:
      i) If user does not have any event tomorrow and ambient noise < 20000, then the ringer volume is set to be normal and vibrate for 10 seconds.
      ii) If user does not have any event tomorrow and ambient noise > 20000, then the ringer volume is set to be loud.
      iii) If the user has event tomorrow and ambient noise < 20000, then the ringer volume is set to be loud.
      iv) If the user has event tomorrow and ambient noise > 20000, then the ringer volume is set to be loud and vibrate for 10 seconds.
   d) When the android device ringing, you can hit "stop alarm" button to stop the ringtone.
   e) After you stop the ringtone, you can hit "restart detect" to restart the process discussed in b).

3.1 Verification process (f1f added)

1) To make context aware application receives callbacks from context middleware, you should press the button "CONNECT TO CONTEXT MIDDLEWARE." On the first line.
2) Instead of having the real time light intensity, it will only display whether the light intensity is less than 10lux. This information is provided by context middleware. You can try to cover the phone with your hand and then remove it.
3) After connect to the context middleware, the location in latitude and longitude will only be displayed when location changed is detected by context middleware. So ***please*** try to walk around with the phone in order to see the corresponding location change.
4) When the context middleware triggers callback for ***light intensity***, you can find the similar message provided below from android monitor log cat:

```
12-07 04:00:17.306 4198-4212/com.ncsu.tyeh3.csc450project2 D/Light < 10 lux? true

12-07 04:00:18.111 4198-4212/com.ncsu.tyeh3.csc450project2 D/Light < 10 lux? false
```

5) When the context middleware triggers callback for ***location***, you can find the similar message provided below from android monitor log cat:

```
12-07 04:01:49.443 4198-4289/com.ncsu.tyeh3.csc450project2 D/Location Update: 35.77196466, -
78.6742122

12-07 04:01:54.505 4198-4289/com.ncsu.tyeh3.csc450project2 D/Location Update: 35.77200704, -
78.67411335
```

4.Additional information (From f1e)

There is a chance that Google distance matrix API cannot resolve the address for the destination. The query to Google API server can be found in android studio **logcat** and looks the following:

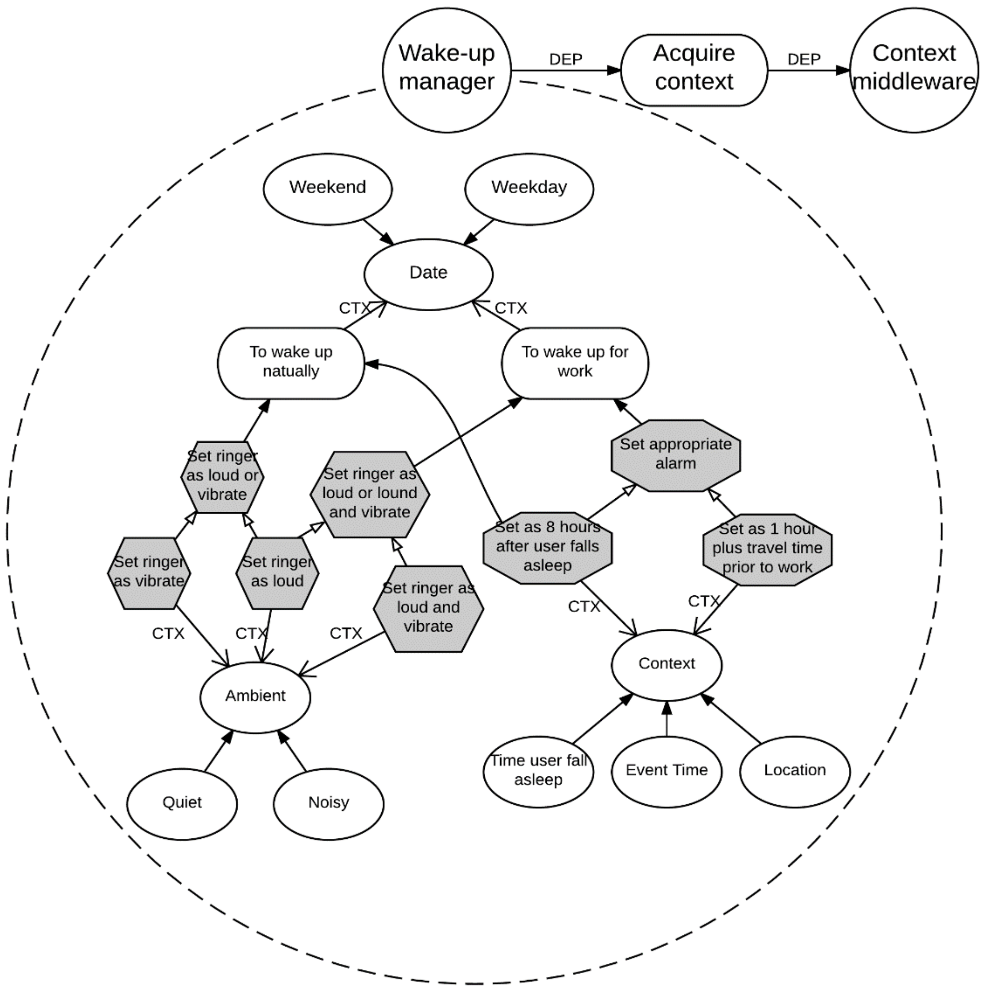
```
11-11 13:30:56.094 3495-3556/com.ncsu.tyeh3.csc450project2
I/System.out: https://maps.googleapis.com/maps/api/distancematrix/json?origins=35.7714492,-
78.6737046&destinations=Red+Hat+Amphitheater,+500+S+McDowell+St,+Raleigh,+NC+27601,+United+State
s&key=AIzaSyC_2YcyF4IHtPAIm9RYifs-krlEK_N9QU0
```

If you enter and go to the URL in the web browser, you can observe the problem I discussed. (The example I provide here is fine. I just want to notice if the application has unexpected behavior to set the alarm time, the problem can be observed by going to the URL)

4.1 Context middleware justification (f1f added)

The context middleware takes the responsibility to detect location change as well as measure the light intensity sensed by the device. As a result, two classes ***LocationManager*** and ***MySensorManager*** are removed from original context aware application and be a part of context middleware. The context middleware will only trigger callback when the location changes and when the light intensity go higher or below 10lux. The benefit of implementing context middleware is that the same information retrieved by context middleware can be reused by difference context aware application and trigger the callback according to their own reasoning as long as both application conform to the ***aidl*** interface. Last but not least, the complexity of the context aware application can be reduced by letting context middleware handle the device sensors.

5.Xipho system model graph (From f1e)


Wake-up manager
DEP
Acquire context
DEP
Context middleware
Weekend
Weekday
Date
CTX
CTX
To wake up natually
To wake up for work
Set ringer as loud or vibrate
Set ringer as loud or lound and vibrate
Set appropriate alarm
Set ringer as vibrate
Set ringer as loud
Set ringer as loud and vibrate
Set as 8 hours after user falls asleep
Set as 1 hour plus travel time prior to work
CTX
CTX
CTX
CTX
CTX
Ambient
Context
Quiet
Noisy
Time user fall asleep
Event Time
Location