Smart Device Sensing Architectures and Applications

Koustabh Dolui
Electronics and Communications,
St. Thomas' College of
Engineering and Technology
Kolkata, India
doluikoustabh @gmail.com

Srijani Mukherjee
Electronics and Communications,
St. Thomas' College of
Engineering and Technology
Kolkata, India
mukherjeesrijani@gmail.com

Soumya Kanti Datta
Mobile Communication
Department
EURECOM
Biot, France
dattas@eurecom.fr

Abstract— **This paper illustrates the use of smart device sensors in various real time applications. Two types of sensor data processing architectures have been discussed. The on-device data processing architecture allows processing of the sensor data locally on the smart device itself and taking a desirable action. The on-server processing architecture requires the device to send the sensor data to a remote server for further computation and action. The pros and cons of these two types of architectures are pointed out. We have designed and implemented realistic Android applications using the smart device sensors and have categorized them broadly into these two architectures. The algorithms of the applications are described in detail along with the results. The paper also demonstrates an application of the atmospheric sensor, currently in research stages, in a virtually simulated device. Android platform is chosen to develop these applications since it is widely used and open source operating system.**

Keywords- Smart device; Sensors; On-device processing; On-server processing; Android.

I. INTRODUCTION

Sensors and various sensing methods have been at the center of research and many literatures [1] illustrate multiple aspects of sensing. With the advent of smartphones and tablets in recent past, embedded sensors have found their way into these devices. Today smart devices boast an array of sensors which include accelerometer, light sensor, proximity sensor and more. The mobile operating system and the applications take advantage of the sensor data to initiate several actions. Due to the growing popularity of Android devices, huge numbers of sensors are present in the today's society. The advantage of using the smart device sensors is that the device itself can collect the sensor output, store and process them locally or communicate them to a remote server. Based on the sensor outcomes, different actions can be triggered in the devices. This leads to research in smart device centric sensing and development of smart applications. Commonly available sensors in smart devices are accelerometer and gyroscope sensor, GPS receiver, battery level and temperature sensor, proximity sensor, ambient light sensor etc. Existing literature mentions various applications of the smart phone sensors including that of automatic battery charging and earthquake detection. The latest smart devices have inbuilt capability to internally disconnect the charger from the device once the charging is complete. However, our module is aimed at protecting the battery life of the older generation of devices by adding an extension to its charging mechanism at a nominal cost as well. This paper also discusses the novel idea of combining the functionality of the automatic charging mechanism and that of the earthquake detector into a unified model. The authors of [2] mention the use of smart devices in monitoring road traffic whereas [3] mentions road status monitoring. Our module devises both these mechanisms to intelligently determine the status of traffic taking into consideration the state of roads as well. Even though many sensory applications of smart devices have been discussed in previous literature [4], little has been mentioned about the architecture of each application. This paper broadly classifies the applications into two categories on board processing architecture and the on server processing architecture based on the nature of the application.

In this paper, two architectures have been put forward for collecting and processing sensor data e.g. on-device processing and on-server processing. We have developed several applications to demonstrate the architectures. The rest of the paper is organized in five sections. Section II illustrates the two types of sensor data processing architectures along with their merits and demerits. Section III discusses the applications that belong to on-device data processing while Section IV presents the applications using on-server data processing. Section V presents the future prospects and the paper concludes in section VI.

II. SENSOR DATA PROCESSING ARCHITECTURE

The output data from all the sensors are acquired and accessed through the Sensor Manager on the Android devices. The data is used by the Android operating system and applications for auto-rotation of screen, brightness control during calls etc. The outcome of sensors can be purposefully processed in other applications on the Android devices. These applications have been designed on Android SDK software using Eclipse. These applications were installed and tested in three android devices. These are Samsung Galaxy Y (Android 2.3), Samsung Galaxy I5510 (Android 2.2), and Samsung Galaxy S2 (Android 2.3.4) respectively.

In this section, these applications and the processing of data obtained from the sensors are discussed in detail. We define two kinds of architectures for these applications viz. on-device data processing and on-server data processing based on where the data is processed.

A. *On Device Data Processing Architecture*

This architecture is designed to acquire the data from the in-built sensors and process the data locally. This architecture should be used when the application is entirely local to the device and the computation is of a lower order. This is due to the fact that the computational capabilities of many smart devices are limited by various constraints like (i) battery capacity, (ii) limited processor speed, (iii) memory for higher order computations and (iv) limited availability of standard library of functions. The advantage of the approach is that it does not depend on network operations.

The applications based on the proximity sensor, accelerometer and gyroscope are primarily limited to the position or tilt of the device itself and are hence local to the device. Similarly the data acquired from the battery level sensor and ambient light sensor will be seldom used for an application remote to the device or to an application with complex computations. Hence the data from these sensors are at the center of the on device data processing architecture. Fig. 1 illustrates the block diagram of the on device data processing architecture.

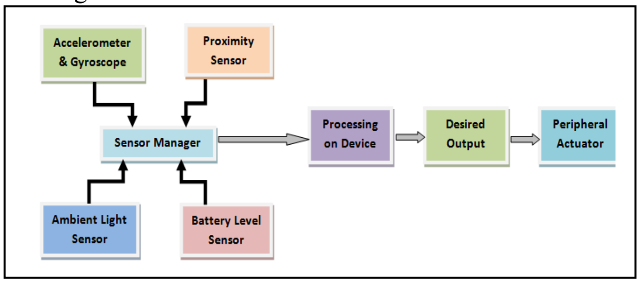


Figure 1. On device data processing architecture

The output from the device can be given through a Dual Tone Multi Frequency (DTMF) signal which can then be processed by a microcontroller to drive an actuator (e.g. motor or switch). If the peripheral actuator is remotely located from the device it can be controlled by sending the output from the device over Bluetooth connectivity or to another device connected physically to the actuator over the internet.

B. *On Server Data Processing Architecture*

In this architecture the data acquired from the sensor is sent to a remote server for further computation. The primary advantage of this kind of architecture is that it allows computation of a large amount of data as well as computations of complex nature. This architecture is used when remote monitoring of a device is necessary, for e.g. monitoring location of a device through data it sends to the server. This form of architecture is also implemented when accumulating data from multiple devices and deducing a pattern or common trend in them.

The GPS sensor is most widely used in this form of architecture. The location data can be used to monitor speed of vehicles in a traffic monitoring system. The GPS sensor can be coupled with the data acquired from the accelerometer of a device to correlate any sudden retardation to a pothole or speed breaker in the road. In another case, we have used the atmospheric sensor to simulate temperature logistics for a region by collection of temperature data from multiple Android devices. In this case, on-server processing is preferred. Fig. 2 depicts the components of the mentioned architecture.

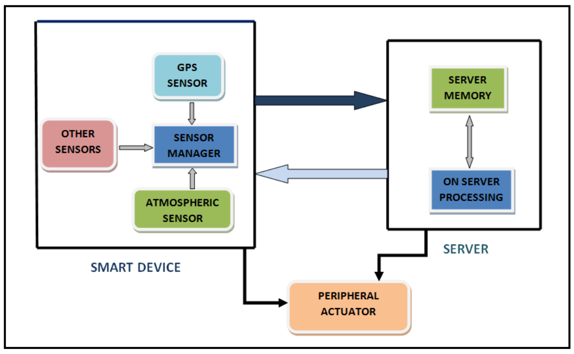


Figure 2. On server data processing architecture

From Fig. 2 it is understood that the smart devices act as clients to the server. The client application collects the sensor data and communicates them to the server. Once the computations are done, the server may send back the result to the client which acts accordingly. The peripheral actuator can be controlled from either the smart device itself or from the server.

III. ON DEVICE PROCESSING APPLICATIONS

In this section the various applications have been discussed based on the on device processing architecture using the sensors embedded in the smart device.

A. *Earthquake and Vibration Detector Prototype*

Earthquakes are detected primarily by seismometers which provide readings proportional to the vibratory motions on the ground. However, it is a costly instrument and is not accessible to the common people. Hence we have used the accelerometer sensor embedded in smart devices to simulate the working principle of a seismograph. A similar simulation has been conducted to detect earthquakes mentioned in the existing literature [5].

The accelerometer sensor measures the acceleration force on the device in the x, y and z axes on a scale of 0-10. When the position of the phone is disturbed by even a small variance the readings of the accelerometer changes. This principle is used to our advantage in this prototype. The values from the accelerometer sensor are accessed using the sensor manager and the following algorithm is applied to detect anomalous vibration. The entire computation of occurrence of an earthquake is done on the smart device itself. The algorithm is discussed below.

- Step 1: The phone is initially placed in a stable position
- Step 2: The constant values of the 3 axes are recorded in three constants a, b and c.
- Step 3: The threshold is set for each axis (T_x, T_y, T_z), depending on the region in which the phone is placed.
- Step 4: If A_x, A_y, A_z are the real times values of the accelerometer, report an earthquake if the following criterion is satisfied;

$$(|A_x-a|>T_x)|(|A_y-b|>T_y)|(|A_z-c|>T_z)$$

Where, |X-Y| denotes the modulus value of the difference. Each of the three criteria is connected using an OR function, i.e. if any of the three criteria along the three axes is satisfied the smart device reports an earthquake. The threshold value may vary from region to region and can be once calibrated for a region using a seismograph. The threshold value must be decided considering whether the region is earthquake prone, or is vulnerable to frequent vibrations, e.g. a region experiencing frequent landslides or a region based near a railway track. Fig. 3 depicts the building blocks of the prototype while screenshots of the developed Android application are portrayed in Fig. 4.

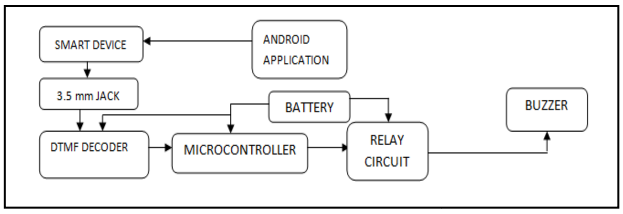
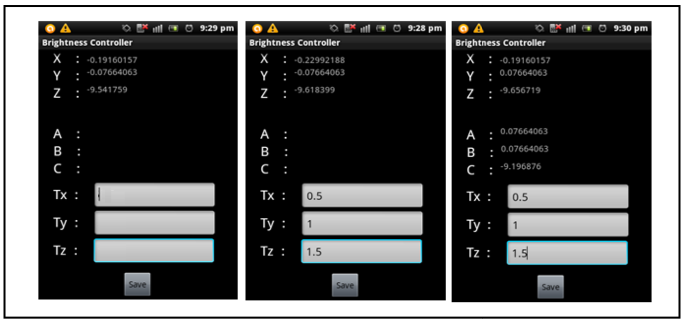


Figure 3. Building blocks of the prototype

This prototype is designed to alert the user of any anomalous vibrations during sleep. Hence a peripheral controller has been used to sound a buzzer if this anomalous vibration is detected. The user can be alerted by using the device tones as well. The smart device alerts the controller of the anomalous vibration using a set of DTMF tones. The controller in turn switches the buzzer on and off using a solid state relay.


9:29 pm
Brightness Controller
X : -0.19160157
Y : -0.07664063
Z : -9.541759
A :
B :
C :
Tx :
Ty :
Tz :
Save

9:28 pm
Brightness Controller
X : -0.22992188
Y : -0.07664063
Z : -9.618399
A :
B :
C :
Tx : 0.5
Ty : 1
Tz : 1.5
Save

9:30 pm
Brightness Controller
X : -0.19160157
Y : 0.07664063
Z : -9.656719
A : 0.07664063
B : 0.07664063
C : -9.196876
Tx : 0.5
Ty : 1
Tz : 1.5
Save

Figure 4. Screenshots of Earthquake Detector prototype application

B. Automatic Battery Charger Prototype

In this section the design and implementation of a prototype charger have been shed light on, which prevents the battery from getting overcharged during the charging process. This prototype is designed to prevent power losses and in turn increase battery life by disconnecting the charger from the device once the charging process is complete, i.e. the battery is 100% charged. The following algorithm has been developed to prevent overcharging of the device.

- Step 1: Initialize charging of the smart device.
- Step 2: Monitor the percentage of charge.
- Step 3: If charge of the device is not equal to 100% continue charging, else go to step 4.
- Step 4: Send DTMF signal of type 1, to disconnect the charger from device using relay.
- Step 5: If charge<15%, send DTMF signal of type 2, to restart charging device; repeat from step 2.

To start charging the device or disconnecting the device from the charger two types of DTMF tones are used. The DTMF tones are chosen as a combination of A, B, C and D, since 0-9 along with * and # are used for dialing purposes. The DTMF tones are sent to a peripheral controller which in turn controls the relay circuit for charging and discharging the battery. Fig. 5 illustrates the block diagram for the automatic battery charger prototype. Most high end smart devices are equipped with automated disconnection of charger after charging completion by default. However, we have designed this prototype as an extension to lower end devices and the older generation of smart devices to extend their battery life and reduce power loss while charging. With this prototype a user can connect the device to a charger overnight while the charger gets disconnected automatically once the charging is complete. Fig. 6 illustrates the screenshots of the application for automatic charging prototype.

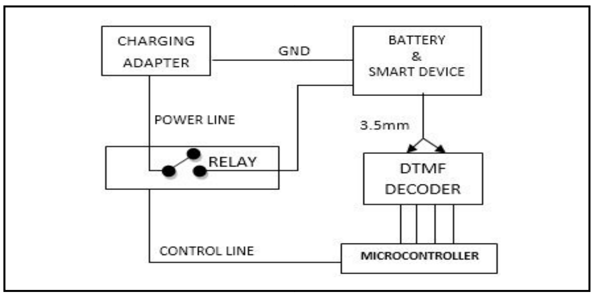
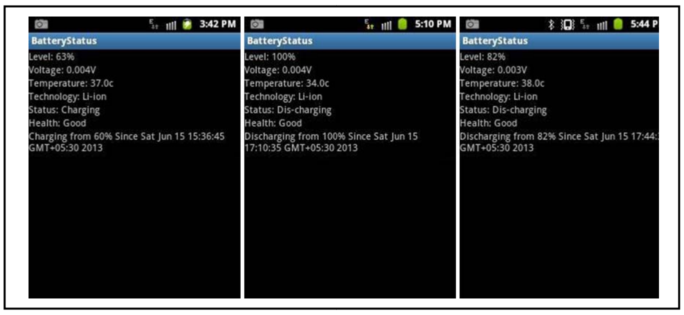


Figure 5. Building blocks for Automatic Charger prototype


3:42 PM
BatteryStatus
Level: 63%
Voltage: 0.004V
Temperature: 37.0c
Technology: Li-ion
Status: Charging
Health: Good
Charging from 60% Since Sat Jun 15 15:36:45 GMT+05:30 2013

5:10 PM
BatteryStatus
Level: 100%
Voltage: 0.004V
Temperature: 34.0c
Technology: Li-ion
Status: Dis-charging
Health: Good
Discharging from 100% Since Sat Jun 15 17:10:35 GMT+05:30 2013

5:44 P
BatteryStatus
Level: 82%
Voltage: 0.003V
Temperature: 38.0c
Technology: Li-ion
Status: Dis-charging
Health: Good
Discharging from 82% Since Sat Jun 15 17:44:
GMT+05:30 2013

Figure 6. Screenshots of Automatic Charger prototype application

If the set of DTMF tones used for the earthquake detector prototype can be defined uniquely with respect to the set of DTMF tones defined for the automatic charger, both the processes can be run simultaneously on the same device. The charger can be connected to the device and the breaking circuit along with the buzzer circuit for the alarm can be connected to the microcontroller. Thus the prototypes for smart battery charging and earthquake detection can be combined together to work simultaneously.

C. Noise and Location Sensitive Volume Control:

More often than not, there are reports of accidents caused by negligence from users when on the streets or crossing a railway track while listening to music. This module is aimed at reducing the possibility of such mishaps by reducing the music level being played on the device depending on the location and surrounding noise levels. The algorithm is as follows.

- Step 1: Check if the user is listening to music on headphones
- Step 2: Acquire surrounding noise levels (N_{sur}) from the mic and compare peak value to amplitude of music (V_{mus})
- Step 3: If N_{sur}>V_{mus}, reduce V_{mus} to less than N_{sur}
- Step 4: Check location of the device and compare with map, whether location is adjacent to railway tracks or accident prone region. If yes reduce the volume below a specified threshold (V_{th}).

This application uses the data acquired from the microphone and the GPS sensor to compute the need to reduce volume locally on the device itself. This prototype is still in the research stages and has further scope for improvement. The value of V_{th} can be dynamically defined depending on $N_{sur.}$ The detection of noise levels can be made more accurate if the collected data is converted to frequency spectrum.

D. Orientation based Brightness Control

In high end smart devices, high resolution displays consume a major percentage of battery. This prototype has been designed with an aim to turn off the display whenever necessary by assessing the position or orientation of the device. We consider the following three cases for three different orientations based on which the device may be turned off.

- Case A: If a device is placed on the desk, the display might still be on idly with the timeout counter set to a high value. If this orientation is recognized by the device as an idle situation, the device can then turn off the display. For this prototype calibration of the device is first necessary to execute the process efficiently. The calibration has been done by placing the device over multiple surfaces and a range of values have been selected for the accelerometer along each of the three axes.
- Case B: A user might place the device in a customized holder which may have its own orientation. Thus if the device is calibrated to this orientation and a range of values are selected, the device display can be switched off once the user places the device in the holder.
- Case C: It is seen most often that people place their phones in their pockets without locking or switching off the display. The phone is usually in a vertical position when placed in the pocket, or in a horizontal position while being sedentary. The orientations observed may vary from user to user. If the device is calibrated to get a set of values for the orientation of the device it can be combined with values from the proximity sensor. If the proximity sensor is detecting an obstacle/object close to it, then only turning off the device makes the process effective. This step is taken since a similar orientation may be achieved while using the device.

The Algorithm is given below.

- Step 1: Store calibrated values in variables for each of the three axes, for each of the three cases. Set a threshold value of time Th_{time}, for each of the three cases for which an orientation must hold to consider it a stable position.
- Step 2: Check if the values of each of the three axes lie in the range specified for a particular case for a time greater than Th_{time}.
- Step 3: Check if the device is running any front end process or application. If yes, do not turn display off, else jump to step 5
- Step 4: If the values are in range for case C, check if the proximity sensor detects an obstacle. If no, do not turn display off, else jump to step 5.
- Step 5: Turn off the display of the device.

This prototype is in the development stages and has further scope for improvement. Extensive calibration and a desirable selection of value for Th_{time} can further facilitate the process.

IV. ON SERVER SENSOR DATA PROCESSING:

The applications mentioned in the previous section require simple computations and hence are preferable for on device data processing. However, for complex computations dealing with large number of values acquired from multiple devices, on server data processing is more efficient and responsive. In this section three useful applications are described that use such type of architecture. Fig. 7 depicts the block diagram for traffic status computation and dissemination.

A. Traffic Status Monitoring Using Smart Devices

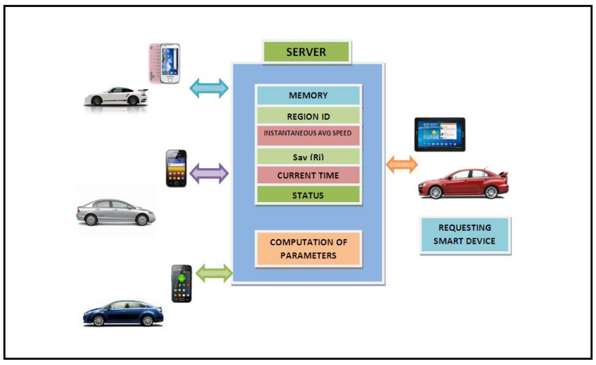


Figure 7. Server architecture for traffic status computation and dissemination

We have developed a prototype to determine traffic status of regions by monitoring the speeds of vehicles with the help of tracking the smart devices in the vehicles [7]. The GPS sensor generates the co-ordinates of the device along with a timestamp at which the data is recorded. These co-ordinates along with the timestamp are used to compute the speed of the vehicle.

The users in vehicles using smart devices are required to register to a server designated to monitor the traffic status by acquiring data from these smart devices. Once registered the server monitors the location of the vehicles and computes the speed from the data it received from the vehicles. Let us assume that the position of a car at time t_1 is (x_1, y_1) and at time t_2 is (x_2, y_2); where $t_1>t_2$. The speed of the vehicle (S) is measured by using the following equation (2).

$$S=\frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}}{(t_1-t_2)} \quad (2)$$

The server acquires the co-ordinates for multiple vehicles. Each city or town is divided into sub regions (Ri) with specific boundary co-ordinates. The server checks for the location of the vehicle with each region and assigns each vehicle to a particular region. In this way the server collects data for multiple vehicles in a particular region and computes the instantaneous average speed (Sins) of the vehicles moving in a region Ri. Using the data acquired from the speeds of vehicles over a long period of time, the server computes an average value of speed SAV (Ri) for a particular region which is cumulatively updated daily. The following algorithm is used to determine the status of the road from the above parameters.

- Step 1: Partition the city/town into multiple regions Ri with specific boundary co-ordinates.
- Step 2: Acquire location for registered smart device. Check co-ordinates of the device and assign to respective region Ri.
- Step 3: Repeat step 2 for all vehicles in each region for all regions.
- Step 4: Compute the speeds of each vehicle using equation (3) in a region and compute the value of Sins.
- Step 5: Set the road status based on the following criteria;
  - ❖ If $S_{ins} \geq S_{AV}$ (R_i); Set status to “Road traffic is normal”
  - ❖ If ½[S_{AV} (R_i)] < S_{ins} < S_{AV} (R_i); Set Status to “Road is congested”
  - ❖ If S_{ins}<½[S_{AV} (R_i)]; Set Status to “Road is severely congested”

The server stores the data with a region ID, which specifies the data for the particular region. A vehicle travelling into a particular region requests for an update of the status of that region from the server using this unique ID. The server acknowledges the request by sending the corresponding values of Sins and S_{AV} (R_i) along with the current system time and the status of the region.

The computation of the speed of vehicles may be done on the device itself and the resultant speed may be sent to the server. This process may allow some relief to the load of computation on the server. The device readings may display some anomalies due to sudden speed drops due to potholes and speed breakers. These anomalies can be removed by combining the GPS data with the accelerometer data of the device as mentioned in literature [8], provided the device is placed in a stable position inside the vehicle. Thus if a drop in speed is characterized by a sudden change in accelerometer readings, it can be considered as an anomalous reading and can hence be discarded. If implemented on a large scale, different servers should be used for different regions or a hierarchical architecture should be followed to prevent congestion of data computations on the server.

The Fig. 8 depicts the application on the Android device taken through a screenshot.

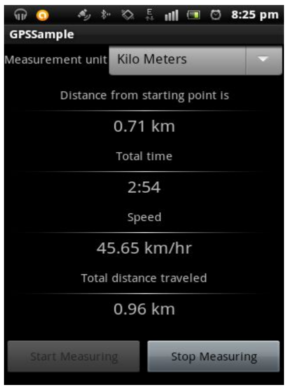

GPSSample
Measurement unit: Kilo Meters
Distance from starting point is
0.71 km
Total time
2:54
Speed
45.65 km/hr
Total distance traveled
0.96 km

Figure 8. Screenshot of the GPS sample application

Thus this application is used to display the status of the vehicle to the monitoring organization remotely. Fig. 9 illustrates the application running on the monitoring end on a PC.

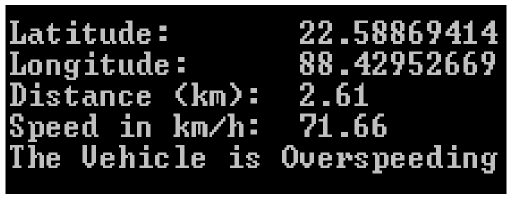

```
Latitude:          22.58869414
Longitude:         88.42952669
Distance (km):     2.61
Speed in km/h:     71.66
The Vehicle is Overspeeding
```

Figure 9. Screenshot of the PC monitoring the vehicles

B. *Remote Vehicle Monitoring*

Vehicles sent out on rent or driven by a chauffeur often require real time monitoring. This is done by equipping vehicles with dedicated GPS devices which cost quite on the higher side. Smart devices can be used to remotely monitor a vehicle for its location and speed. We have designed an application using which a device tags itself to a particular vehicle using a user ID. Once registered the smart device starts sending the GPS co-ordinates acquired from the GPS sensor back to the server. The server thus monitors the location of the vehicle and computes its speed from the data it receives. The server data can then be reported to another client smart device, preferably belonging to the owner of the vehicle. Using this prototype one can also penalize the chauffeur for over speeding. The server receiving the vehicular co-ordinates can map the co-ordinates to the speed limit of the particular location. This can result in a warning to the driver and can also be used to curb accidents due to rash driving on rented vehicles.

C. *Atmospheric Sensor Data Logistics:*

Atmospheric Sensor is a proposed sensor module designed to measure the atmospheric pressure in the region environing the device, the ambient temperature and the relative humidity of a region. This sensor thus if implemented will provide a personal measuring device for each user to keep track of the above mentioned atmospheric parameters. We have designed an application based on this principle to keep track of the parameters in local regions in a city or a town by acquiring and processing data from the sensors in the same region.

1) Algorithm for Processing Atmospheric Sensor Data:

- Step 1: Partition the city/town into regions Ri, with each region having respective boundary co-ordinates.
- Step 2: Acquire the location of the device from the GPS sensor and assign the device to corresponding region.
- Step 3: Send the atmospheric sensor data from the device to the server.
- Step 4: Repeat steps 2 and 3 for all devices in a specific region for all regions.
- Step 5: Compute the average values for each parameter on the server and store the values for the particular region Ri.
- Step 6: Send the values to any remotely located client device requesting data for any region Ri.

The atmospheric parameters are usually measured at a particular location where the weather department is located. It provides updates only during specific times of the day pertaining to only that region. This module if implemented can provide updates at all times of the day to users, remote to an area or visiting an area. This can be particularly useful for regions where changes in temperature and weather are abrupt over small distance.

V. Application and Future Prospects

With further development and refinement of the sensors, the above applications can be extensively implemented across the society. This section describes two future research directions.

A. Cross Platform Compatibility

The operating systems primarily used in the smart devices are Android, iOS, Blackberry and Windows Mobile. Most of these smart devices are equipped with the aforementioned sensors. The client applications for the on-server processing must be developed for all the major mobile platforms so as to reach out to the maximum users. The applications are proposed to be developed using cross platform tools like PhoneGap, Sencha Touch [6]. They allow developing the application using web technologies (HTML, CSS, and JavaScript) and same application can be built for multiple mobile platforms generating the executable. This process minimizes the application development cost.

B. Smart Charging Using Renewable Energy Sources

Researchers have developed a photovoltaic cell that could be mounted on the screen of smart devices. The cell is capable of charging the device with suitable the intensity of ambient light. The ambient light sensor can measure the intensity of the light incident on the sensor and that output could decide if the battery can be charged. This provides an eco friendly mechanism of charging the device.

VI. Discussion

This paper proposes the use of the smart device sensors to simulate simple yet effective applications which come of use in our daily lives. Applications and prototypes based on the sensors, which are proposed here, used singularly or in a combination of one or more sensors. A significant contribution has been made through this paper that is defining the two architecture models for data accumulation and processing. The proposed prototypes based on the on server sensor data processing architecture will show higher accuracy and efficiency with higher number of participants. Hence rapid growth of smart phones will contribute to the efficiency of the process. The availability of high speed internet access like UMTS and 4G at affordable rates will allow a smoother communication between server and the smart devices. With improvement of embedded microphones in the devices, the noise sensitive volume control prototype can be vastly improved. Our current research is focused on improving the position based brightness control and noise sensitive volume control prototypes and conduct extensive calibrations to improve their performance.

References

[1] C.Buratti, A.Conti, D.Dardari and R.Verdone, "An Overview onWireless Sensor Networks Technology and Evolution", Sensors 2009, 9(9), pp. 6869-6896

[2] L. Sahabandu, L.Samarkoon, D.Fernando, P.Chanthirasegaran, S.Udana, D.Asanga, "A sustainable mechanism for gathering road traffic data using smart-phones", 2012 International Conference on Advances in ICT for Emerging Regions (ICTer), 12-15 Dec. 2012, p. 224

[3] G. Strazdins, A. Mednis, G. Kanonirs, R. Zviedris, and L. Selavo, “Towards Vehicular Sensor Networks with Android Smartphones for Road Surface Monitoring,” 2nd International Workshop on Networks of Cooperating Objects (CONET’11), Electronic Proceedings of CPS Week'11, 2011.

[4] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, Andrew T. Campbel,"A Survey of Mobile Phone Sensing",IEEE Communications Magazine,September 2010

[5] T. Uga, T.Nagaosa, D.Kawashima, "An emergency earthquake warning system using mobile terminals with a built-in accelerometer", 2012 12th International Conference on ITS Telecommunications (ITST), 5-8 Nov. 2012, pp. 837 – 842

[6] I. Dalmasso, S.K. Datta, C. Bonnet, N. Nikaein, " Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tools," 9th International Wireless Communications & Mobile Computing Conference (IWCMC 2013), Italy, 1-5 July, 2013, pp. 323-328.

[7] K. Dolui, S. Mukherjee, S.K. Datta, “Traffic Status Monitoring Using Smart Devices”, unpublished

[8] G. Strazdins, A. Mednis, G. Kanonirs, R. Zviedris, and L. Selavo, “Towards Vehicular Sensor Networks with Android Smartphones for Road Surface Monitoring,” 2nd International Workshop on Networks of Cooperating Objects (CONET’11), Electronic Proceedings of CPS Week'11, 2011.