Communications Architecture in Vehicular Participatory Sensing

Addressing the challenges associated with big data management.

Fengjiao Lu, Stephen Brandon, Yunyi Zhu
Trinity College Dublin
Dublin, Ireland
lvfe@tcd.ie, brandons@tcd.ie, zhuyu@tcd.ie

***Abstract* — Smart cities create a lot of data! A significant challenge in any smart city is managing all the collected data and putting it to use without incurring massive cost. With communications technology improving areas such as vehicular networks and participatory sensing are becoming increasingly popular methods of sourcing data.**

This paper aims to evaluate vehicular networks and participatory sensing architectures as a method of collecting and managing big data in smart cities. Adopting a use case of detection of road deterioration we investigate how reliable data can be collected merged and processed in a smart city environment and what, if any impact, this would have on how this type of data is currently collected and used. Keywords— participatory sensing; potholes; accelerometers; algorithms; communications; smart cities;

I. INTRODUCTION

Today Smart Cities can take advantage of ubiquitous computing with a wide range of sensors and devices able to communicate and provide data that will benefit everyone. One of the challenges associated with using this data is the sheer size of the data set. With a growing number of sensors reporting data, some kind of architecture is required for processing and filtering data for usefulness.

In this paper we explore communication architectures with particular attention to vehicular communications and participatory sensing. While our research goal is to investigate how rich data sets, such as those available in smart city scenarios, can be best managed. To do this we select a use case where a communications architecture would be useful to city managers.

One challenge city authorities face is maintenance of the road network and monitoring road deterioration. Road deterioration is a serious problem because poor roads become more expensive to repair over time and can cause drivers to avoid bad patches of road increasing the risk of accidents occurring. A number of methods have been employed in the past, for example, using Ground Penetrating Radar [3] specially fitted to a car that can scan the road for signs of deterioration such as rough surfaces, potholes, and sunken manhole covers. Most of these methods can be expensive and time consuming.

Participatory sensing provides a viable alternative however it is important that the data is filtered correctly as false positives, i.e. recording road deterioration where it does not exist, could prove time consuming and expensive. Using participatory sensing in this context introduces complexity because sensor data needs to be collected, merged, and processed which can be expensive when servers are required to process the information.

In this paper we introduce a hierarchical participatory sensing network designed to handle the large amount of data without requiring expensive IT infrastructure. Filtering the collected data on road condition at each step of the hierarchy allows for more accurate outlier detection meaning more reliable and useful data is sent to the server at the top of the hierarchy.

Using Arduino based equipment and sensors as well as mobile phones we were able to create communications architecture with Bluetooth, WI-Fi, and 3G. We believe this communications architecture offers a viable alternative that will be cheaper to implement than existing road deterioration detection techniques.

The rest of the document is structured as follows. Section II introduces related work. In Section III the overview of the problem is presented. Section IV documents our implementation and Section V is an evaluation of our approach. The final section presents our conclusion and future work.

II. RELATED WORK

A. Exsiting Projects

Currently, several vehicle participatory sensing systems are investigated for pothole detection or air pollution detection.

BusNet [1] is a participatory sensing system developed by University of Colombo for monitoring air pollution caused public transport. Specifically, few sensors are mounted on public transportation such as buses and trains to collect air pollution data. Usually, such data will be stored in sensors temporarily until the bus or train arrives at a station which has

a hosting sensor. Afterwards, the hosting sensors will upload the air pollution data to a server of BusNet via a wireless network. BusNet provides an inexpensive solution to use few sensors to gather air pollution levels in a large geographical area in contrast to deploy a large number of sensors to cover entire area. Undoubtedly, BusNet provides an economical way to collect air pollution data via maximal utilization of public transport system. But the problem is that normally public transportations have fixed routes and specific interval time, which means that collected data are single-point especially in sparsely populated area.

Nericell [2] system developed at Microsoft Research India is monitoring traffic and road conditions by using an ensemble of mobile smartphones carried by members of the public. Typically, the person who is participator in Nericell system starts the software and then put his phone in the packet when he prepares to drive. The software will automatically record the data from embedded sensors such as accelerometer and GPS while the data is uploading to server via WIFI or Cellular Network. There are few limitations in Nericell system:

1. Regular battery is not able to maintain phone system running for a slightly long time in the absence of charge, because not only does the smartphones OS need energy, but the embedded sensors also required it.
2. Due to misbehaviours of the user, accuracy of data is hard to guarantee. For instance, accelerometer is much likely to measure fake data when driver shake his body.
3. Lack of pre-processing in stage of data collection is likely to lead to overload in system servers.

The paper [11] describes a reliable, measurable system that can monitor chronic patients constantly and makes notifications if serious situations are observed. This system leveraged GPS sensor and accelerometer sensors to generate data. Sensor data are gathered and stored in an SQLlite database on the phone and are sent from phone to remote monitoring web server through 3G or Wi-Fi to do further processing and data storage. The communication protocol used in the system is Bluetooth protocol, and Arduino board is used to connecting data between many sensors and software on a computer.

The main drawback of the project is that the android phone takes too much work load considering its limited storage and battery. Not only is the phone required to run complex algorithm which takes up much CPU but also phone needs to communicate with web server. The drawback can become worse with the increase of data processing complexity. So it is not ideal to let phone overload.

UbicKids [10] collects children health and activity information such as body movement, location and so on, and provides alert of abnormalities when potential emergency situation is detected. Besides the lightweight, portability and mobility feature of Respberry Pi and Intel Galileo, the two boards are used to host multiple sensors and implement intelligent reasoning for children safety. GPS, accelerometer and temperature sensors generate data and send to gateway boards via XBee radio. Raspberry Pi connects to local network via its Ethernet port and inserts an XBee radio shield to communicate with the sensors.

The main drawback of the project is that every vest needs to have gateway devices to host sensors which increases the whole cost. Currently, everyone has one or more smartphones. If adopting phones as hosts to gather sensor data and send it to a main gateway board. It can reduce the cost greatly.

Besides, phones can help to share the work of the Galileo board by optionally adding a phone into the system because smartphones have the ability to store and process temporary data.

B. Differences

Participatory sensing system undoubtedly is a realistic and effective way for acquisition of road quality information in a large geographical area. However, in terms of implementation, these existing projects both have some problems, so our approach for collection of data is distinct from the prior work in two different aspects:

1. Replace embedded sensors of smartphone with self-contained sensors suite to measure. This not only provides more accurate data but is easier on the users battery.
2. Implement Galileo Board to be hosting sensor to pre-process data before upload to server. The Board acts as intermediary device.

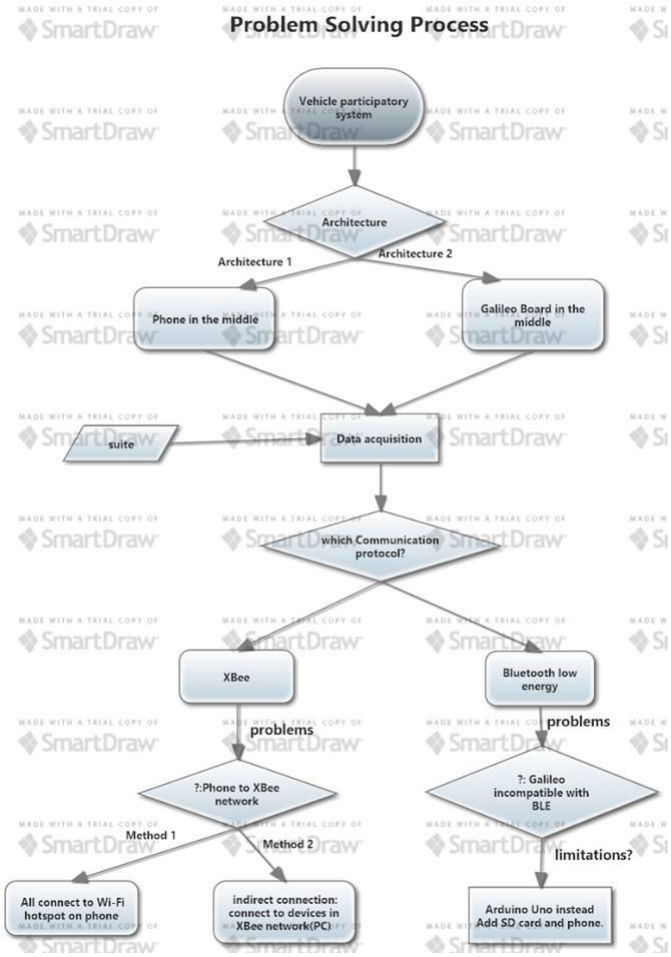
III. OVERVIEW

The roads deteriorate with age and maintenance works have to keep pace with other new construction works otherwise it can be a threat to citizen safety as well as other issues. In this paper, an improved way of detecting deteriorating stretches of road and communicating and processing this data by exploring the use of Intel Galileo boards for merging data gathered from participatory sensing and other devices, is implemented.

The design of the detecting system gives consideration to cost-effectiveness, performance and energy saving. It uses phones to act as intermediate agency to gather senor data from cars, and communicate with an Galileo board; in comparison, many existing project adopts one Galileo board per car. In addition, the processing of large volumes of data will be distributed among android phones, Galileo boards and a web server, which realises load balance and better response time.

A. Basic overview

In the paper, there are two main plans are designed to make comparison; they are: 1) a main phone elected in the middle to gather together and transfer data detected by cars to one Galileo board, 2) one Galileo board in the middle to gather data from cars.



B. Communication

In the two plans, the most important part is the communication protocol (Wi-Fi, Bluetooth or XBee) of any wirelessly connecting two devices, especially between phones and Galileo board; the system tried two ways: 1) XBee protocol connecting Galileo board with XBee shield, 2) Bluetooth communication connecting android devices with Arduino Uno board because the Intel Galileo proved difficult to set up with Bluetooth shields.

However in XBee communication, android phone is not compatible with XBee protocol, two ways are used to establish the communication; Method 1) is indirectly talking by letting phones talk with a constantly listening laptop and the laptop transfers data to XBee end devices (intel Galileo with XBee shield) and coordinator. Method 2) is enabling Wi-Fi hotspot on phone to build an accessing point. In this way, both phone and XBee end device likes Galileo board can connect to the access point but it has not put into implementation.

Participatory sensing plays a curial role in the system. It needs to guarantee constantly, real-time road map update. Given that phone can be out of internet and delaying the transfer time, some error handling algorithm will be listed in the following part.

IV. OUR APPROACH

In order to minimize power consumption, embedded sensors of smartphone is deprecated, instead of self-contained sensors suite. Each self-contained sensors suite is consisted of a Xadow Main Board, a Xadow IMU 6DOF (a 3-axis accelerometer) and a Xadow BLE Slave (a Bluetooth receiver/sender), which is capable of collecting acceleration of 3-axis and then sending information to smartphone via BLE Slave. Energy of such a self-contained sensors suite is supplied by an independent battery as Fig. 1 presented. In addition, specification of acceleration data is unified, and the format is (x-axis, y-axis, z-axis).

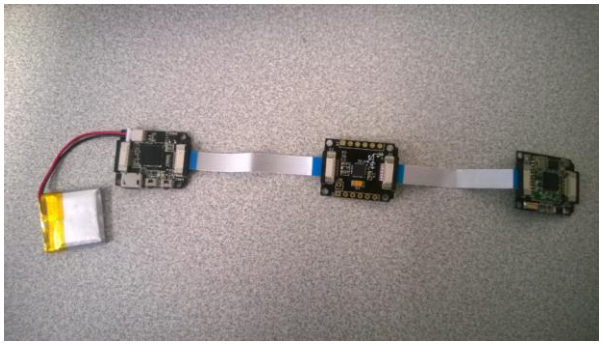


Fig.1. Sensors Suite

According to previous research, simultaneously measuring and mixing accelerations from numerous accelerometers installed in different places in the vehicle is capable of improving accuracy of data and mitigating the impact of misbehaviours of users such as stopping the car suddenly. Consequently, in the system, three sensors suites will be deployed in one car.

Typically, frequency of data acquisition on one accelerometer is frequent. As a consequence, one big participatory sensing system, which contains a huge number of accelerometers, intends to generate massive data that will require big computing power to process. In order to solve this problem, a pre-process module will be implemented, which consists of a Galileo board and wireless communication shield. The Galileo board is capable of gathering and analysing acceleration data deployed between sensors suites and servers, so that load of servers is able to be reduced. Specifically, procedure of pothole detection is completed by the Galileo board, and then the result of detection is sent to servers. Two different design architectures are provided in our project.

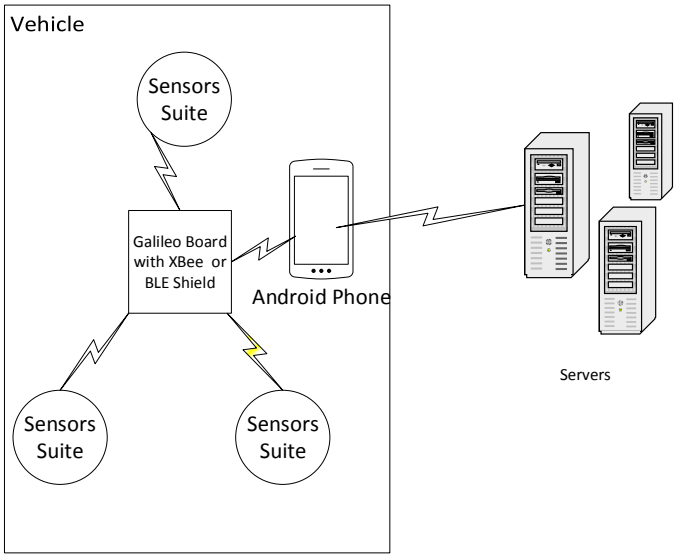


Fig.2. Architecture A

In Architecture A, Galileo board is located in the vehicle. In terms of working procedures, firstly, Galileo board keeps gathering acceleration data from 3 sensors suites via wireless connection, and use the machine learning algorithm presented in Pothole Patrol system [13] to detect the potential pothole. Next, the result of pothole detection will be sent to Android phone via XBee from Galileo board. When the phone received the results, it will transfer the results with GPS Location information to servers through WIFI or Cellular network. Finally, servers store these results and select places that pothole is frequently reported to label it as damaged road section.

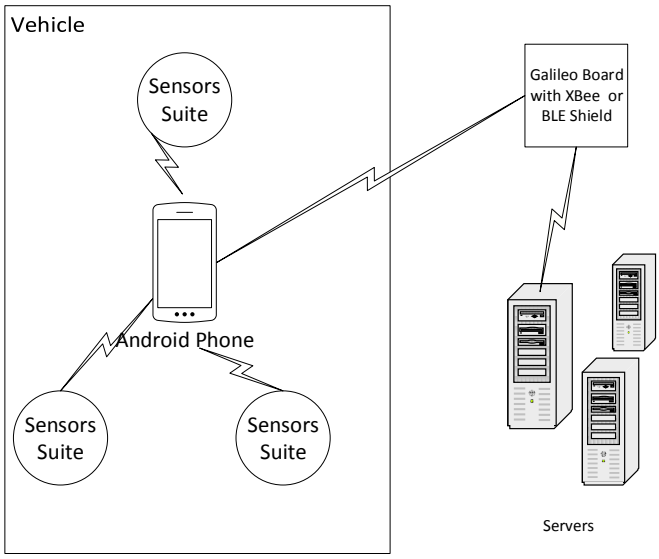


Fig.3. Architecture B

Unlike Architecture A, in the Architecture B, the Intel Galileo board will be outside the car. Its purpose is to connect to the mobile phones in the car, possibly at traffic lights or parking meters, and extract the detected pothole data. The Intel Galileo board will further run outlier detection to determine if other cars experienced the same problem in the same location. Once the Intel Galileo has determined that a pothole is present at a certain location it can send the data to a RESTful web service via HTTPS POST using JSON encoded data.

In comparison, the advantage of Architecture A is reflected on stability of wireless communication. As one can see, all devices which need to be communicated for data transmission are relatively static, so that dynamic connections of Bluetooth are avoided, which leads to a fact that likelihood of packet loss will be substantially reduced. However, in this architecture, one Galileo is merely responsible for pre-processing and filtering noisy data from three sensors suites in one vehicle. Consequently, servers still need to handle with massive data. That disobeys our pervious motivation.

Alternately, Architecture B intends to resolve this issue. Specifically, in our design, Galileo board is installed on traffic lights to monitor one specific area. All data come from this particular area, is supposed to be sent to the Galileo board via XBee. In this situation, the Galileo board takes responsibility to filter noisy data and process valuable data from all vehicles in this area. Hence, nothing but valuable result of road condition estimate will be uploaded to server, which is capable of improving effectiveness of system. Thus, Architecture B will be the final architecture.

In these two plans, communication channels between Boards and phones are needed to make choice. Some general design consideration and steps about two main channels (XBee and BLE) are described in part D. In addition, the differing channels influence some device choice and bring about other design problems, especially gateway boards.

A. Gateway

Both Arduino Uno and Intel Galileo is required to act as gateway to connect to a remote web server. The weakness of Arduino Uno, less storage and processing capability, is mitigated by adding a phone to Arduino Uno.

Galileo board is much capable than Arduino Uno to be the gateway. Galileo has bigger storage than Arduino, and it connects to web server by using the Ethernet port in a simple way. Owning to its processing ability of Galileo board, it is designed that part of data combining job like ruling out non-sensing data is done by Galileo board, which, as a result, reduces traffic with web server as well as response time. Meanwhile, Kalman filtering algorithm [12] will be used to calculate the mean of sensor data.

In order to overcome the weakness of Arduino Uno, a phone can share part of work of Arduino Uno as gateway. Arduino is mainly used for sending and receiving data from sensors by opening a listening port, then it sends data to a directly-connected phone. This phone with SD card will process filtering algorithm and communicate with web server, reliving the work load of Arduino Uno. However, the drawback of this method is increased cost.

In comparison, Galileo board can provide the ability to deal with big data individually, and send data to servers, which

can be ideal to use in the system. But the real problem is the limitation that Galileo board cannot support Bluetooth low energy protocol. Even though it supports another low energy protocol call XBee, this communication protocol has some problems like high error rate which will be described in the following part.

B. Communication

In the project per design, the communication is needed to be established between android devices and gateway boards. Two different communication protocols are implemented and compared: XBee with Galileo board, Bluetooth low energy with Arduino Uno.

Hence, the following part will illustrate in detail about needed devices, general steps of configuration, and android application. Importantly, in XBee channel, two ways are illustrated to overcome the problem that android phones do not support XBee protocol, even though there still exists some unsolved questions.

XBee

After creating XBee radio, it needs to create communications among Galileo board with XBee (as XBee end devices), phones, and XBee explorer with XBee (as XBee coordinator). The challenge of connecting phones to the XBee network is solved in part E. The following part of this section will describe the features, configuration, and architecture of XBee.

"*XBee is produced by Digi international, forming compatible radio modules in order to support point-to-point and star communications at over-the-air rate of 250 Kbit/s.*"[6]

XBee has two hardware footprints: Through-Hole (TH) which has 20-pin socket and Surface Mount which includes 37 pads. Both of them take up little space on PCB, and the benefits of footprints lie that 1) common design makes XBee interchangeable; 2) the design supports options of both low and high volume deployments.

XBee Network

Three kinds of devices in the XBee network:

Coordinator: Take responsible for composing the network, managing and assigning device address.

Router: Take responsible for agency that passing messages between devices. Many routers co-exist in the network. In addition, routers can relay messages in the network if the distances between devices are out of ZigBee range.

End device: Responsible for sending and receiving data but it requires a router or coordinator synced.

In every XBee network, there must have one XBee coordinator to broadcast XBee radio and many XBee end devices.

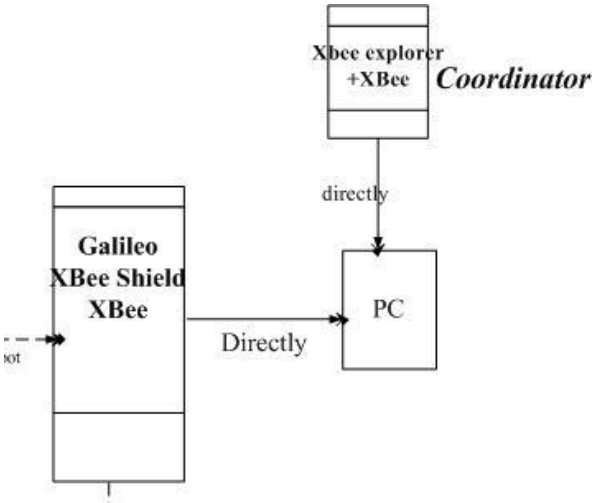


Fig.4. Architecture of XBee network

The figure shows a general XBee network. The explorer with XBee act as XBee coordinator to send broadcast radio while the Galileo board act as the XBee end device to send data to the coordinator wirelessly or wirily.

Configuration

To configure the XBee Modules, XCTU is needed. XCTU is software that used to manage and control XBee modules and communications, including configuring radio frequencies of each devices and management. The following conditions are required to fulfil the communication automatically.

- Each network configures one coordinator and at least one end devices.
- All XBee modules configure the agreed firmware and PAN-ID.[9]

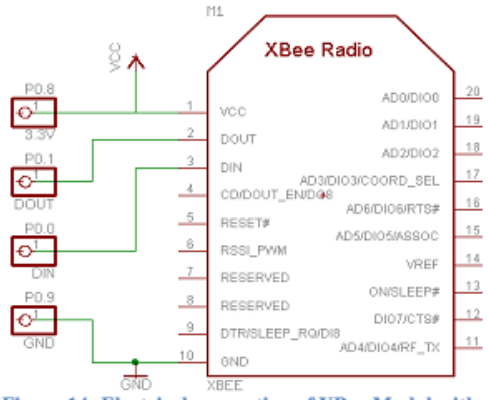


Fig.5. XBee module

The figure shows the pins of XBee radio, with the help of XBee shield, the connection between Galileo board and XBee can be easy establishing.

After everything is correctly configured, the coordinator is able to connect with many end devices. It is coordinator that broadcast commands and end devices communicate with coordinator.

As to participatory sensing in large network or some overlap of networks, the end-devices can act as a router to forward data to many networks.

Some issues in XBee protocol

1) The difficulty of communications with android phones

The problem of connecting android-based phone and XBee module is that phone and XBee are not in the same frequency band, and phone does not enable XBee protocol. However, Phone can be wired to XBee via the serial port of phone.

Besides, it can work in two ways to enable communication between XBee module and phone.

Method 1: Create a network on the phone by using android tethering, change the XBee's setting, and connect XBee to the phone SSID. This method is recommended online but we found it difficult to implement in practice.

Method 2: Use an access point to broadcast the SSID, connect both phone and XBee to the access point as infrastructure.

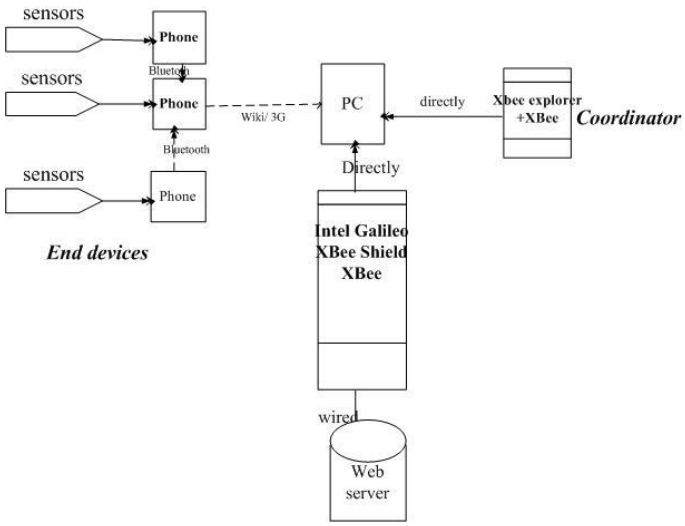


Fig.6. Design of method one

Figure 6 shows design of method one: phone interacts with Galileo board indirectly. The data flow will be from sensors, end devices to XBee coordinator. XBee explorer attached to PC via USB, and PC will import data from android device to XCTU. Hence, the data from android phone can reach to XBee coordinator.

The detailed android code used to retrieve sensor data and send to PC, and Java code in PC is implemented to receive data from phone and import it into XCTU.

The main drawback of this method is that we need put a PC at every detection point, so it can increase the cost and it is not portable. But it is implementable.

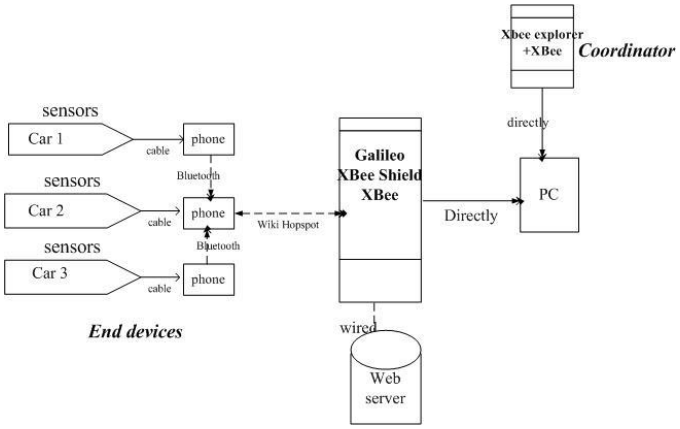


Fig.7. Design of method Two

Figure 7 shows turning on an access point to broadcast the SSID, connect both phone and XBee to the access point as infrastructure.

It seems implementable in theory. However, how XBee end devices can connect to the access point is a question. The benefits of this method are that it does not need add extra device which can save cost and take advantage of existing WI-FI network.

In this project, the method 1 is adopted and in experiments, it reveals some problems of XBee communication.

2) Loss of packet

The range of an XBee device is affected by several factors including the transmit power of this device and environments, the maximum distance from official website is 3km in outdoor surroundings and 30m in indoor environment, which reflects the range in an ideal conditions. However, in the project, many factors like walls, trees and other barriers do influence the broadcast distance.

The XBee radio does not support acknowledgement and is unreliable. In the experiment, one end device sent a 5 lines of data to XBee coordinator within 5 meters, however, one line of data met with loss of packet, so there is 20% loss of data in 5 meters.

BLE

This part will explain the Bluetooth low energy communication between Android mobile phone and Arduino board with BLE shield, as a comparison with XBee.

Galileo board is lack of supporting Bluetooth chipset, so it can be complex to use Galileo board to communicate with Bluetooth enabled device. So here, Arduino equipped with Bluetooth chipset is used. However, the limitation of Arduino board is its limited storage and processing capacity.

Because of the limitation of Arduino, the system is designed to mitigate the problem by adding a phone to share the work of Arduino board. The phone will take charge of filtering data while Arduino will gather sensor data and filter noisy data and send to server.

Architecture

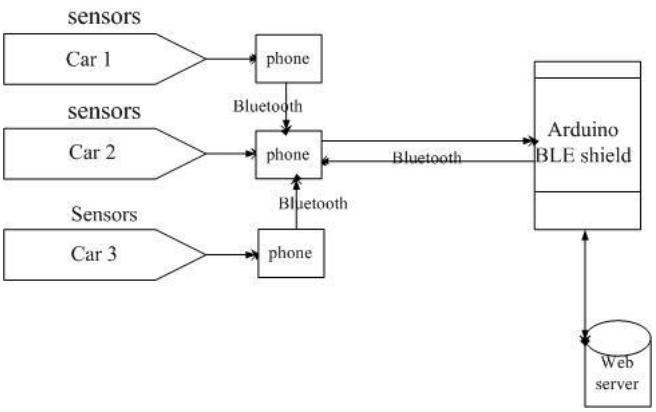


Fig.8. BLE Architecture

As figure shows, the architecture is a master/slave architecture, slave is able to talk to only one master while master can multicast many slaves. Here, Arduino UNOs are slaves. Android application is able to store temporary data in SQLlite, and take advantage of Java-based algorithm to filter data, Arduino board is able to include multiple sensors and embedded with a Bluetooth chipset. Hence, what needs here is an android application that can search and listen to Bluetooth supported device, and besides, an Arduino programmes to open the port to listen to the messages from android application.

Configuration

Step 1: Connect BLE shield to Arduino

BT Rx connects Arduino Tx; BT Tx connects Arduino Rx; VCC to 5v; GND to GND

Step 2: Upload example sketch to Arduino

Step 3: Setup the Android device

Step 4: data filtering in android phone

First we rule out noisy data at the suite. Then we use the Kalman Filter[12] to establish the mean,

Analysis

1) The issues of using Bluetooth low energy

Only parts of devices support Bluetooth Low energy, like iPhone 4s or higher, android 4.3 or above, windows 8.1.

In experiment, we find that the connection establish time is very long and transmission time is very long but it is better than classic Bluetooth.

"*BLE inherits the operating spectrum, implements lightweight link layer that provide ultralow power idle mode operation, fast device discovery and reliable and secure point to multipoint data transfers. So, BLE support maximum distance range of 50m, throughput is 0.27 Mbit/s, latency is 6 ms, power consumption is 0.01 to 0.5.*"[7]

2) The issues of Arduino Uno

The Arduino board has limited storage and process capacity. In order to avoid the problem, phones have to take work load which phone may become overload and require higher processor in case of big data.

3) Benefits of this design

Bluetooth low energy provide significantly low power consumption in contrast to classic Bluetooth.

The Arduino Uno equipped with Bluetooth chip is easy to manipulate, by inserting into USB of pc.

Besides the Arduino don't need extra power.

It is easy to configure by uploading Arduino sketch which opens ports to listen to the connection of matched Bluetooth-enabled device.

V. EVALUATION

A. Experiment of Project

In order to test availability of this project, a practical experiment has been implemented.

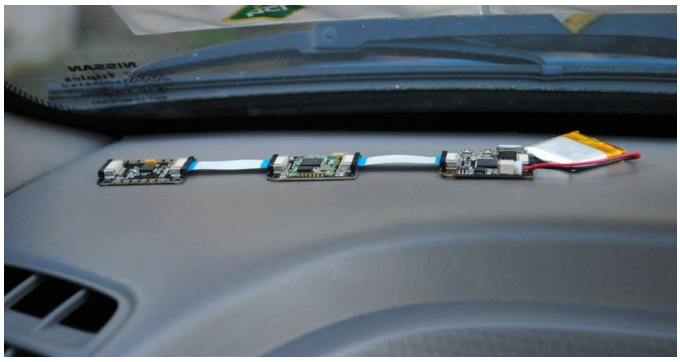


Fig.9. Sensors suite in the real car

1) The three sensors suites with one Samsung S4 are set on different places in the Nissan Micra (2001) K11. The car is driven on a road with potholes in Dublin.(We're not sure there are roads without potholes!)

2) The Galielo board and one laptop are placed on wayside of that road.

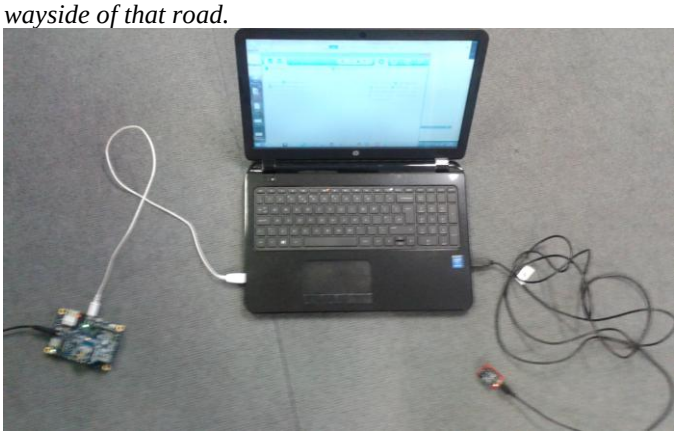


Fig.10. Galileo and XBee setup

3) The car is driven through the pothole. Acceleration data are recorded on the log of the App.

Fig.11. Log of Android App

4) The acceleration data with GPS coordinate will be sent to the Galileo board via the phone.

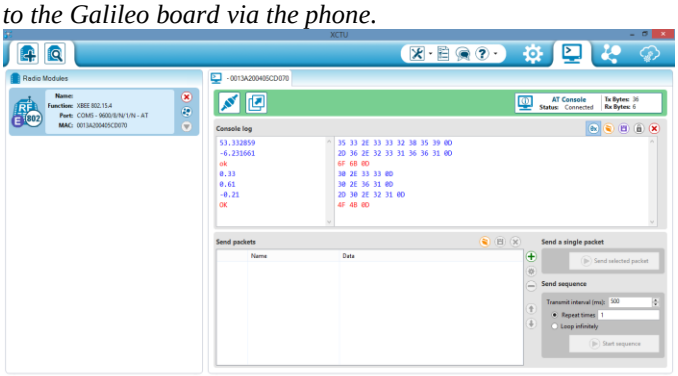


Fig.12. XCTU of XBee end device (Galileo+XBee)

.

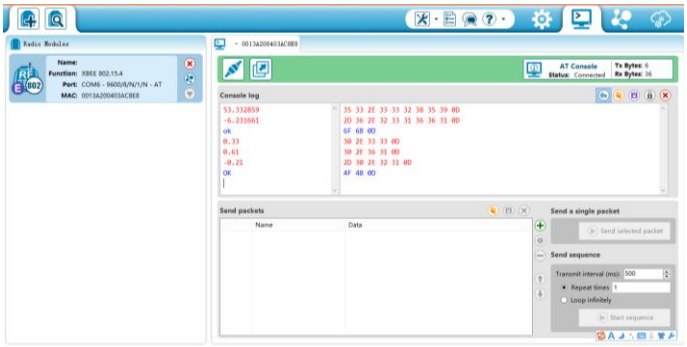


Fig.13. XCTU of XBee coordinator (XBee+ XBee explorer)

As is showed in Figure 7, the location data as well as accelerometer data are sent from phone to XBee network

5) Galileo board sends results of pothole detection to the laptop (The localhost server).

6) The app is able to present the potential pothole road section on Google Map.

Fig.14. User Interface

B. Evaluation of Intermediary (Galileo Board)

As discussed in Section II, related work, there are many ways that data can be collected and managed in participatory sensing networks. When these vehicular networks are scaled up to work in a smart city context the amount of data collected can be overwhelming for systems to deal with. With potentially hundreds of thousands or millions of cars communicating every day even a very small error rate is not acceptable.

Traditional methods of detecting errors in data sets involve outlier detection and creating rejection zones where incorrect or useless data can be discarded. The problem with this is the processing power involved to collect and process detection of errors.

In our implementation we create a communications hierarchy where an Intel Galileo board acts as an intermediary processing collected data before allowing it to proceed to the web server. In our experiment we think the Intel Galileo board should be mounted on a traffic light or parking meter for example. The board should have network connectivity via 3G and a Wi-Fi enabled shield to communicate with the vehicles. We think a traffic light or parking meter is a good place because it already has a power source and vehicles frequently stop there long enough to exchange data over Wi-Fi.

We believe that this intermediary system is important because the Intel Galileo is capable of comparing results to other collected results from that locality meaning it can detect false positives and discard them. This is important because sending a repair team to fix a road problem that doesn't exist would quickly become expensive. Another positive of using the board is the distributed nature of the system will allow for greater fault tolerance as well as separation of concerns.

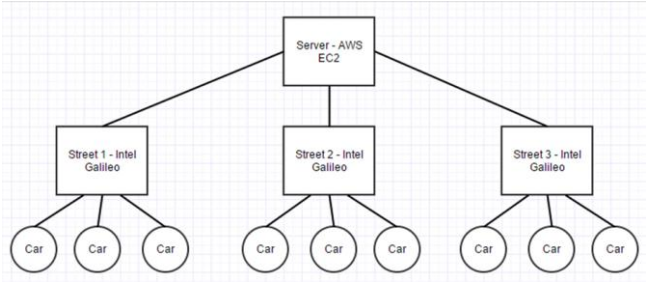


Fig.15. Hierarchy Communication Architecture

To evaluate this communications architecture and what benefits it has we must compare it to a vehicular participatory sensor network where there is no intermediary involved and vehicles simply report their data directly to a server via a 3/4G enabled mobile phone.

In this case the server has a lot more work to do because it must first establish confidence levels in road deterioration by area or street and then evaluate all data received from those streets and compare it to the confidence levels in order to detect outliers. Another problem with this centralized architecture is that it is vulnerable to errors or attacks that could stop the entire system working from one location. i.e. the web server. It's not as easy to manage users in the centralized system and all of the processing is done in one place creating additional expense with the cloud providers.

However the decentralized architecture proposed in this paper is not without its flaws. As well as security and privacy concerns it could be impractical to have a device such as an Intel Galileo attempt to collect data from cars for an entire street. Constrained with small memory and storage more expensive equipment may be required which could be just as expensive as purchasing the centralized architecture through the cloud provider. Another problem with our implementation is the cost of providing 3G connectivity to each board in a city scenario may be impractical.

While our design is the most manageable and scalable it could well be more expensive to implement and manage than existing techniques, however there are other areas in the smart city context that may benefit from this type of communications architecture.

C. Evaluation of XBee

In the project, two communication protocols are used to establish data exchange between android-based phone and gateway board; one is XBee protocol with Intel Galileo board, the other is Bluetooth Low energy along with Arduino Uno. XBee protocol is preferred than Bluetooth Low energy.
XBee radio is very easy to be affected; walls or other obstacles will noise the data.

1) Test1-loss rate of data

a) Tuning the parameters

The number of data based on set distance can get lost due to many factors. For example, the direction of data sent, the walls, the trees and the distance.

The measurement is directions of data transmission, and obstacles. The rate of data loss is calculated. In the experiment, the distance between XBee modules kept unchanged.

b) Experiment setup

The experiment is conducted in two steps. For the first, the two XBee models communicates from coordinator to end devices (explorer to Galileo) and then change the direction. In figure 8, the experiment result about data loss showed and table 1 showed the data loss rate.

c) Result

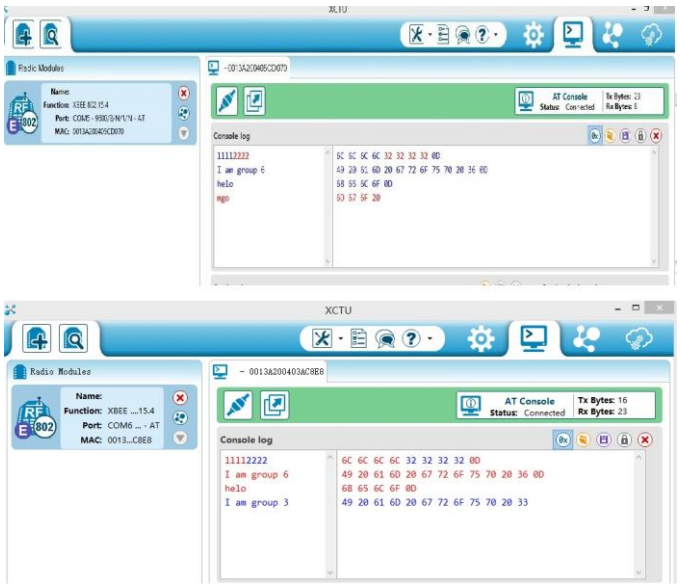


Fig.16. the experiment result

As we can see, the packet loss rate is better in the upper picture where he data is sent from coordinator.

	4 packets	10 packets	15 packets
From coordinator	0%	10%	13%
From end devices	20%	20%	33%

Table.1. Error rate (distance5 m)

As the table shows, the data sent from end device is easier to lose than data from coordinator.

However, the experiment is conducted with small amount of data and small distance range. The loss rate has been up to around 30%, which is high if put into road quality detection. Not even to mention that in real city street, many walls, trees are formed into obstacles to interfere the XBee communication.

It is not very ideal to use XBee in road quality measurement.

2) Test-response time of data communcations:

The experiment also conducts with BLE with Arduino Uno. We tested the response time of data exchange from android phone and Arduino board.

	Response time
XBee with Galileo board	<1 s
BLE with Arduino board	6 s

Table.2. Response Time

It is clear that Bluetooth low energy requires longer response time. But the reason could be the different processing time of Arduino board and Galileo board.

Refer to published report [7, 9]:

XBee is served in local area networks (LANs) while Bluetooth low energy (BLE) is served both in Local Area Networks (LANs) and Personal area networks (PAN).

The maximum distance of XBee is 50m in indoor environment and 3 km in outdoor environment. However, these data are determined in ideal environment.

XBee radio is very easy to be affected; walls or other obstacles will noise the data. Bluetooth is worse than XBee in anti-interference.

VI. CONCLUSION AND FUTURE WORK

A. *Conclusion*

In conclusion we think our chosen design provides a viable solution to the problem of collecting, merging, and processing data in vehicular participatory sensing networks which can otherwise be very difficult to manage. While the Intel Galileo board was an excellent research tool more work would need to be done to determine if it could handle as many connections as would be required in a real world scenario. Despite our concerns raised in the evaluation section we believe the introduction of intermediary devices into communications architectures will help collect better more accurate data, more securely, if managed properly.

Eliminating false positives by filtering data remains a high priority and we would like to see future work to determine if this architecture can successfully do that. We mentioned in our implementation about the use of Wi-Fi and 3G technology but unfortunately we did not have all the equipment we needed to test this, some future work to evaluate the most effective communication channels would be beneficial.

B. *Future Work*

First of all, in the section of evaluation, low energy consumption of XBee has been proved. However, its instability of data transmission has also been revealed. When the distance between Galileo board and Phone exceed 5 meters, the rate of packet loss will be increased, so that XBee is not capable of handling with slightly large range. In order to implement our project in real life, we want to replace XBee with WI-FI in the future.

Secondly, we intend to build an ad hoc network around the Galileo board that act as a coordinator. Such ad hoc network is able to achieve data exchange between vehicles. As a consequence, accuracy of collected data will be improved.

REFERENCES

[1] K. De Zoysa, C. Keppitiyagama, G. P. Seneviratne, and W. W. A. T. Shihan, "A public transport system based sensor network for road surface condition monitoring," in Proceedings of the 2007 workshop on Networked systems for developing regions, ser. NSDR '07. New York, NY, USA: ACM, 2007, pp. 9:1–9:6. [Online]. Available: http://doi.acm.org/10.1145/1326571.1326585

[2] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: using mobile smartphones for rich monitoring of road and traffic conditions," in Proceedings of the 6th ACM conference on Embedded network sensor systems, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 357–358. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460450

[3] Roadscanners.com, 'Roadscanners | Beyond The Surface', 2015. [Online]. Available: http://www.roadscanners.com/. [Accessed: 26- Apr- 2015].

[4] Playing with an Arduino and sensors. [online] Available at: http://www.den-uijl.nl/electronics/gyro.html [Accessed 29 Apr. 2015].

[5] X-CTU (XCTU) software. [online] Available at: http://knowledge.digi.com/articles/Knowledge_Base_Article/X-CTU-XCTU-software/?l=en_US&fs=Search&pn=1.

[6] Learn.sparkfun.com, (2015). Exploring XBees and XCTU - learn.sparkfun.com. [online] Available at: https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu?_ga=1.263212364.1303818453.1423872907.

[7] Mei, Z., Wang, D., Chen, J. and Wang, W. (2014). Investigation of Bicycle Travel Time Estimation Using Bluetooth Sensors for Low Sampling Rates. PROMET, 26(5).

[8] Project Gallery, (2015). Signal Fish Tutorial: Inside the Blue | Intel Communities. [online] Available at: https://communities.intel.com/message/240704.

[9] Sparkfun.com, (2015). XBee Buying Guide - SparkFun Electronics. [online] Available at: https://www.sparkfun.com/pages/xbee_guide.

[10] Jutila, M., Strömmer, E., Ervasti, M., Hillukkala, M., Karhula, P. and Laitakari, J. (2015). Safety services for children: a wearable sensor vest with wireless charging. *Pers Ubiquit Comput*.

[11] Pigadas, V. (2009). *Enabling Constant Monitoring of Chronic Patient Using Android Smart Phones*. 1st ed. Greece.

[12] Liam, P. (2010). Fast Kalman filtering on quasilinear dendritic trees. *Frontiers in Neuroscience*, 4.

[13] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in Proceeding of the 6th international conference on Mobile systems, applications, and services, ser. MobiSys '08. New York, NY, USA: ACM, 2008, pp. 29–39. [Online]. Available: http://doi.acm.org/10.1145/1378600.137860