Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data

Pekka Sillberg, Mika Saari, Jere Grönman, Petri Rantanen, and Markku Kuusisto

Tampere University, Faculty of Information Technology and Communication Sciences, Pori, Finland

Abstract Nowadays almost everyone has a mobile phone and even the most basic smartphones often come embedded with a variety of sensors. These sensors, in combination with a large user base, offer huge potential in the realization of crowdsourcing applications. The crowdsourcing aspect is of interest especially in situations where users' everyday actions can generate data usable in more complex scenarios. The research goal in this paper is to introduce a combination of models for data gathering and analysis of the gathered data, enabling effective data processing of large data sets. Both models are applied and tested in the developed prototype system. In addition, the paper presents the test setup and results of the study, including a description of the web user interface used to illustrate road condition data. The data were collected by a group of users driving on roads in western Finland. Finally, it provides a discussion on the challenges faced in the implementation of the prototype system and a look at the problems related to the analysis of the collected data. In general, the collected data were discovered to be more useful in the assessment of the overall condition of roads, and less useful for finding specific problematic spots on roads, such as potholes.

1 Introduction

It is important to keep road networks in good condition. These days, technology and mobile devices in particular enable the automation of environmental observation [1, 2]. Mobile phones can be deployed for a particular purpose for which they were not originally designed. In addition, applications that combine road maintenance and mobile devices have already been developed [3]. In Finland, there has been a similar study on how to utilize mobile phones for collecting road condition information [4]. In the study, bus companies tested mobile phone soft-ware that sends real-time weather condition data to road maintainers in winter time. Nevertheless, traditional road condition monitoring requires manual effort – driving on the roads and checking their condition, observing traffic cameras, and investigating reports and complaints received from road users. Automation of the monitoring process, for example by utilizing crowdsourcing, could provide a more cost-efficient solution.

Data gathering is an important part of research related to the Internet of Things (IoT) [5]. In this research, the focus of data gathering has been redirected toward a Wireless Sensor Network (WSN) [6] type of solution. Previously, we have studied technologies related to applications that automate environmental observations utilizing mobile devices. In a recent research study [7], we introduced two cases: the tracking and photographing of bus stops, and the tracking and photo-graphing of recycling areas. The first case used mobile phones and the second used a Raspberry Pi embedded system. Our other study [8] facilitated the utilization of information gathered from road users. As part of the research work, a mobile application was developed for gathering crowdsourced data.

The gathered data per se are not very usable and therefore some kind of processing is necessary. Ma et al. discussed IoT data management in their paper [9] and focused on handling data in different layers of WSN. Also, they discussed data handling challenges, approaches, and opportunities. In this study we use our previously introduced Faucet-Sink-Drain model [10]. In this model the data processing and data sources are combined in a controlled and systematic way.

This paper is an extension of Sillberg et al. [11], where the focus was on introducing the prototype system. In this extension paper, more emphasis is placed on the models behind the prototype system. We have developed a mobile application for sensing road surface anomalies (called ShockApplication). The purpose of this application is to sense the vibration of a mobile phone installed in a car. The application was tested by gathering data on real-life scenarios. The data were stored in a cloud service. In addition, we present methods that utilize the free map services available on the Internet for visualization of the data.

The research goal in this paper is to combine models of 1) data gathering and 2) analysis of the gathered data that enables effective data processing of large data sets. Both models were applied and tested in the developed prototype system. Our previous studies related to the models are presented in Section 3, where the data gathering model and the modifications made for this study are introduced in subsection 3.1. Data processing produces useful information for the user. Subsection 3.2 describes the processing model used in the prototype system. This model is designed as a general-purpose tool for systematic control and analysis of big data. With the use of these fundamentally simple models it is possible to create practical and interoperable applications.

The rest of this paper is structured as follows. In Section 2, we introduce the related research on crowdsourcing efforts in the collection of road condition data. Section 4 integrates the models presented in Section 3. In Section 5, we present the test setup and results. Section 6 includes a discussion and suggestions for future research on the topic and finally, the study is summarized in Section 7.

2 Background

Nowadays almost everyone has a mobile phone and even the most basic smartphones often come embedded with a variety of sensors [2]. This opens up the possibility of crowdsourcing through the use of mobile phones. The term crowdsourcing was defined by [12] in 2006. When several users use their devices for gathering data for a specific purpose, it can be considered a crowdsourcing activity. The idea of utilizing crowdsourcing as a model for problem solving was introduced in [13]. Furthermore, crowdsourcing can be used to support software engineering activities (e.g., software development). This matter has been widely dealt with in survey [14].

There have been several studies on using a mobile phone to detect road sur-face anomalies. One piece of research [15] presented an extensive collection of related studies. Further, the research introduced an algorithm for detecting road anomalies by using an accelerometer and a Global Positioning System (GPS) integrated into a mobile phone. The application was described as easy-to-use and developed for crowdsourcing, but the crowdsourcing aspects were not elaborated. The tests were performed with six different cars at slow speeds (20 km/h and 40 km/h). The route used in the test was set up within a campus area. The research paper did not discuss the visualization aspect nor the application itself and focused primarily on the algorithm that was presented.

The research presented in [16] and [17] was aimed at finding particular holes in a certain road. [16] used a gyroscope instead of an accelerometer and looked for spikes in the data. The other information logged was sampling time, speed, and GPS locations. The test was conducted on a route that was about four kilometers long and the test was repeated five times to ensure consistency and repeatability. The crowdsourcing aspect was not mentioned and, according to the paper, the data were collected "through a common repository." The research [17] presented an Android application for detecting potholes, but did not provide much detail on the technical implementation.

There are several studies where the research was performed in a real-life scenario using taxis [18, 19] or buses [20]. In study [18], the data were gathered by seven taxis in the Boston area. The data collection devices were embedded computers running on a Linux-based operating system. In study [19], the data were gathered by 100 taxis in the Shenzhen urban region. The devices consisted of a microcontroller (MCU), a GPS module, a three-axis accelerometer, and a GSM module. The devices were mounted inside the cars and sent the data to servers over a wireless connection. The main idea of the research [18] was to collect data and then train a detector based on the peak X and Z accelerations and instantaneous velocity of the vehicle. The result reported in the paper was that over 90% of the potholes reported by the system were real potholes or other road anomalies. The crowdsourcing aspect was not mentioned, and the visualization was limited to showing a set of detections on a map. In study [20], the data were gathered by phones installed in buses. The data were projected on a map, but the amount of data collected (100 MB/week) and how this would affect a larger crowd were not discussed.

3 Two-phased Model of Data Processing

The research goal in this paper is a combination of models for 1) data gathering and 2) analysis of the gathered data which enables effective data processing of large data sets. Both models were applied and tested in the developed prototype system. With the use of these fundamentally simple models, it is possible to create highly practical and interoperable applications that can improve the overall quality of software.

The data gathering model and the modifications made for this study are introduced in subsection 3.1. The model is one type of Wireless Sensor Network (WSN) solution. In addition, the usage of the model in our previous research is introduced.

Subsection 3.2 describes the processing model used in the prototype system. The processing model is designed as a general-purpose tool for systematic control and analysis of big data. However, the model is very flexible and should fit a wide range of applications.

3.1 Data Gathering

Data gathering is an important part of research on the Internet of Things (IoT). In this research, the focus of data gathering has been redirected toward the WSN type of solution. Because we use mobile phones as sensor nodes, it could be categorized as a mobile sensor network. The advantages of a mobile sensor network have been discussed by Dyo [21]. In addition, Leppänen et al. [22] discuss using mobile phones as sensor nodes in data collection and data processing. A survey conducted in 2002 compiled the basic features of sensor networks [23].

In this study, we used the previously presented data gathering model. This model was introduced by Saari et al. [24] and it has three main parts: sensor node, master node, and cloud. The sensor node sends data to the master node. The master node collects and saves data, but does not process the data significantly. The master node sends data to the cloud service which stores the data. The data gathering model includes the following WSN features presented in [23]:

- Sensor nodes can be used for continuous sensing - When using a mobile phone as a sensor node, this is enabled by dedicated software.
- The mobile phone includes the basic components of a sensor node: sensing unit, processing unit, transceiver unit, and power unit.

- A sensor network is composed of a large number of sensor nodes - The prototype design presented in this study does not limit the number of mobile phones used.
- The network - Mobile phones have the communication network provided by telecommunications companies.

The model has been tested with an off-the-shelf credit card sized computer and other instruments [24-26]. The data collector service [25] used a BeagleBone Black computer and sensors. The embedded Linux controlled sensor net-work [24] used Arduino boards and sensors for the sensor nodes and an Intel Galileo Computer for the master node. Communication between sensor nodes and master nodes was handled with ZigBee expansion boards. The third study [26] used the model to test a low-energy algorithm for sensor data transmission from sensor nodes to master node.

Fig. 1 shows the modified data gathering model. The present study differs from previous research in that we used mobile phones for data gathering, which caused changes to the data gathering model. Another difference from the previous model [24] is that the sensor nodes and master nodes are combined into one entity. This was due to the use of mobile phones as sensor devices. The mobile phone includes the necessary sensors, data storage, and communication channels for this prototype system. In addition, the mobile phones use the Android operating system (OS), which has enough capabilities to gather and store data. Also, the communication protocols are supported by OS. We developed the testing software during this research. This software, called the ShockApplication, and its properties are described later in Section 5.1.

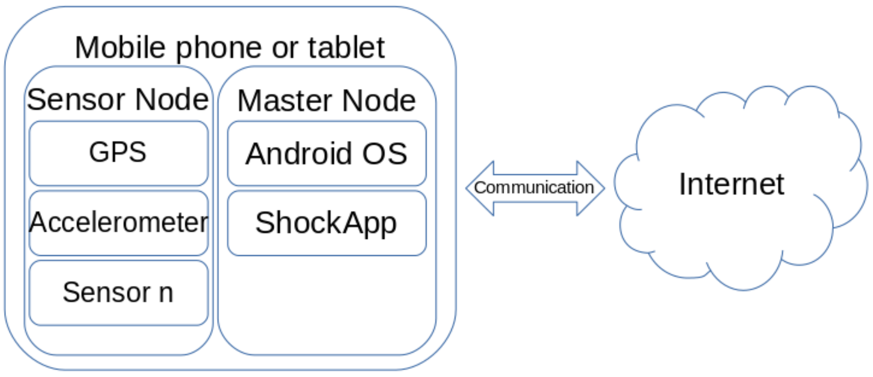


Fig. 1. The modified data gathering model.

The usage of mobile phones enabled the crowdsourcing idea. The developed ShockApplication can be installed on all modern Android phones. The user has an identification mark which helps to order the data points in the cloud. The data are stored in a cloud service.

3.2 Data Processing: Manageable Data Sources

For the data processing part, the Faucet-Sink-Drain model introduced in [10] is applied to the system architecture. The ultimate goal of the model is to enable realization of a framework that is able to manage data and data sources in a controlled and systematic way [10]. In this study, the model was applied to the prototype system, but the implementation of the framework was not carried out. This prototype is the first instance of the model in a real-world use case and will help in the further evaluation and development of the model.
The model considers that data processing can be modeled with a water piping apparatus consisting of five components: faucets, streams, sink, sieves, and drains [10]. The data flow through the model as many times as is deemed necessary to achieve the desired information. At each new cycle, a new set of faucets, sieves, and drains are created, which generate new streams to be stored in the sink. [10]

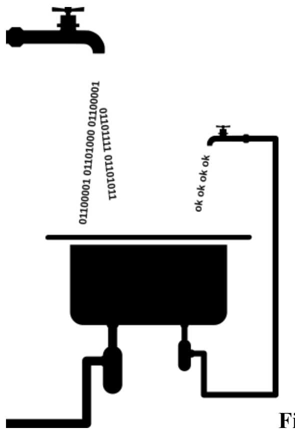


Fig. 2. Abstract data processing model. [10]

The components of the Faucet-Sink-Drain model are shown in Fig. 2. The faucet is the source of the data (e.g., original source or processed source). The running water (i.e., strings of numbers and characters) are instances of data streams, and the sink is used for storing of the data. The sieve is a filter component with the capability of selecting and processing any chunk of any given data stream. The drain is a piping system to transfer data to other locations. The drain may also be utilized for removal of excess data. [10]

The Faucet-Sink-Drain model, by design, does not specify how the data are gathered into it. As shown in Fig. 2, the initial data simply appear in the model by means of the attached faucet (or faucets). The gap can be filled by utilizing models that are stronger in this respect, such as the data collection model described in subsection 3.1.

4 Integration of the Models in the Prototype System

The models used lay out the basis for measurement and data analysis. By following them, it is then possible to implement the artifacts of the prototype system. The implemented prototype system has five identifiable high level tasks:

1. Acquisition: The data are gathered by a mobile device, which acts as a combined sensor-master node as it is capable enough for both of those tasks.
2. Storage: The cloud service receives and parses the data (communicated by the master node). Parsing of the data is the first task to be done on the system before the received data can be fully utilized. After parsing is finished, the service can then proceed by storing and/or by further processing the data.
3. Identification and Filtering: The data will be identified and filtered when the service receives an HTTP GET query on its REST (Representational State Transfer) interface. The selection is based on the rules that are passed in the request as parameters.
4. Processing: The selected data are processed further by the rules given out by the program.
5. Visualization: The data provided by the service are finally visualized in a client's user interface, e.g., web browser.

The data gathering is performed by a mobile phone by utilizing several of its available sensors. Secondly, the collected data are communicated to the cloud service where storage, selection, and further processing of the data are implemented. Once the data have been processed the last time, they are ready to be presented to the user, for example, to be visualized in a web browser or provided to another service through a machine-to-machine (M2M) interface.

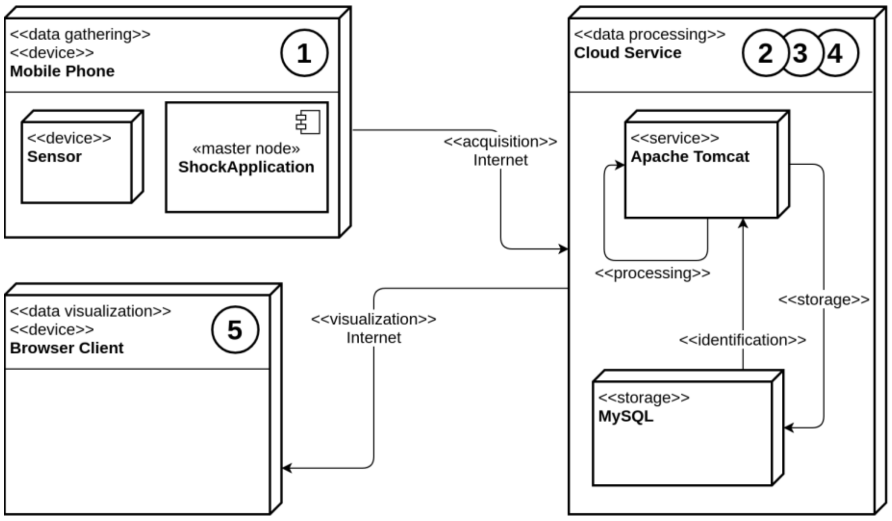


Fig. 3. System deployment diagram.

Fig. 3 shows the deployment diagram of the implemented system. It also depicts where the aforementioned tasks are carried out. These tasks can also be identified from the incorporated models, the Data Gathering model and the Faucet-Sink-Drain model. The first task, data acquisition, corresponds to the whole data gathering model and also to the combination of the (leftmost) faucet and stream icons in Fig. 2. The storage task matches the sink icon in Fig. 2. The (right-most) sieve in Fig. 2 represents the third task, identification and filtering whereas the combination of (rightmost) drain and faucet represent the processing task. The final step, visualization, is said to be handled by the sink as it is "used to store and display data" [10]. However, the visualization step could begin as early as when a data stream has emerged from a faucet and could last until the moment the data have finally been drained out from the sink.

5 Testing

The high-level description of our testing setup is illustrated in Fig. 4. The purpose was to gather data from mobile devices – primarily smartphones – that could be used to detect the surface condition of the road being driven on. These data could be further refined into more specific data, such as reports of bumps on the road, uneven road surfaces, roadworks, and so on. The traffic signs visualize the possible roadside conditions that users might be interested in. The data are sent to a central service and can be later browsed using a user interface running in a web browser.

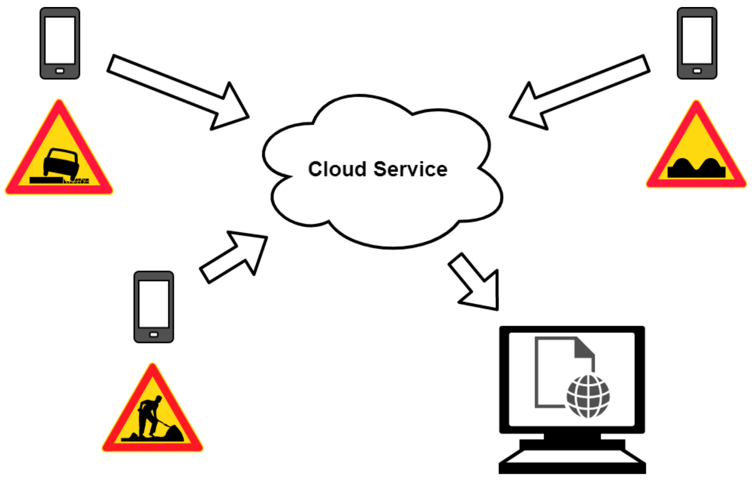


Fig. 4. High-level diagram of the test setup.

In our case, the users travelled by car. In principle, other road users such as cyclists or motorcyclists could be included, but in the scope of this study, only passenger car users were considered.

5.1 Setup

Existing studies often assume that the device is firmly attached in a specific place inside the vehicle, and in a specific way, but for crowdsourcing purposes this is not a feasible scenario. It should be possible to attach the device in a way that is the most convenient for the user, and in an optimal scenario the device could also be kept, for example, inside the pockets of the user. In our benchmarks, the device holder was not limited although we presumed that the devices were placed in a fairly stable location, and did not move about the vehicle in an unpredictable fashion (e.g., sliding along the dashboard).

In addition to the attachment of the device, several other factors (e.g., suspension, tires, vehicle load, and weight) may affect the sensor reading. It can be challenging to implement measurement of these factors in crowdsourcing scenarios. Due to these limitations, we decided to focus on sensors available in commonly used mobile devices.

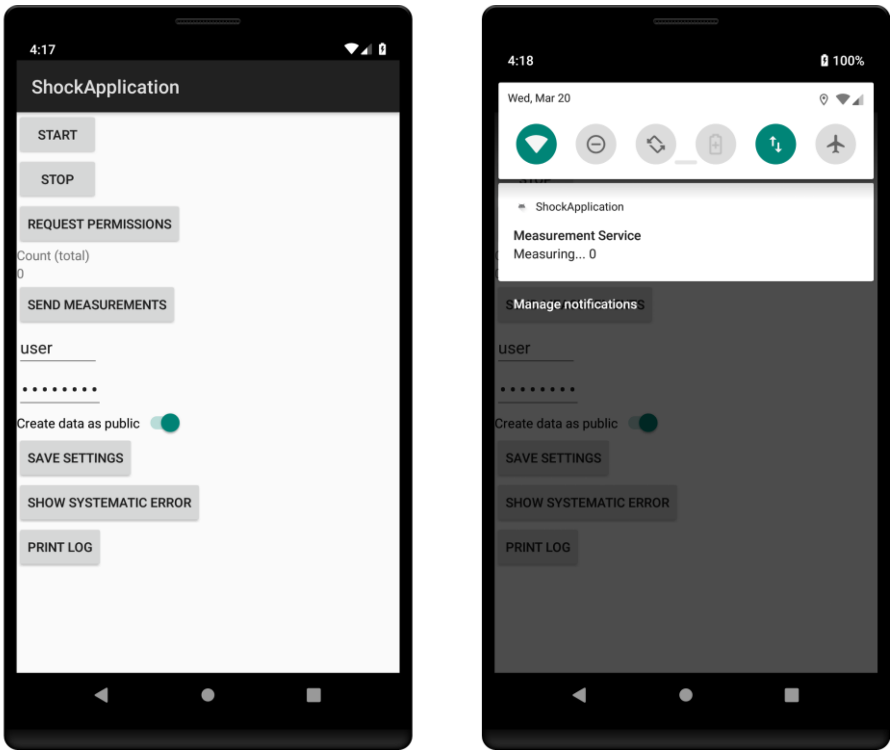


Fig. 5. The Android test client.

The testing software itself was a simple Android application, usable on any reasonably recent Android phone. Most of the newer smartphones generally contain all the necessary sensors required in our use case. The application consists of a single main view, shown on the left side of Fig. 5. In our case, the user only needs to input his/her credentials (in the example, “user”) and use the start and stop buttons to control when the sensors are active. The user interface also contains a few convenience functions: the possibility to attempt manual transmission of all collected data; a count, which shows the total number of measurements (a single measurement contains all sensor data collected at a particular point in time, in the example pictures taken from an Android emulator the value is shown simply as “0”); the option to create all measurements as “public”, which means that any logged-in user can see the travelled route and the collected measurements; the option to save the updated settings, mainly authentication details; and two debug options that the users do not generally need to use. The software will automatically select between the linear accelerometer (which is used, if available) and the basic accelerometer. If the device is set on a stable surface the linear accelerometer should show zero for all axes and the accelerometer should show gravity, but in practice the devices showed slight variances from the expected values. The “show systematic error” option can be used to show the currently measured values and to select whether the systematic error should be removed from the values before sending the results to the service. The “print log” can be used to show a debug log of the events (such as errors) detected since application startup. It would have also been a minor matter to simply hide the debug options from the user interface, but as the primary purpose of the application was to collect data and this version of the application would not be made available for public down-load and installation (e.g., in an application store), there was no specific need to polish the user interface. Thus, the users were simply instructed to input their credentials and use the start and stop buttons, and to ignore the other options.

The sensor measurements are collected by an Android foreground service, which runs as a background process. After the service has been started, the main application can be freely closed and the statistics of the collected data (number of measurements) can be seen in the Android’s pull-down menu, which is visible on the right side of Fig. 5. In the trial, the users kept the sensors on while driving (i.e., when “participating” in the trial) and off at other times. In addition to changing the user credentials, no further configuration was required by the users.

The application was used to measure accelerometer data (X, Y, and Z acceleration), direction, speed, location (GPS coordinates), and timestamps. The collected information was automatically sent to the service at pre-defined intervals (every 30 minutes) by the background process. In addition, gyroscope and rotation data were stored on-device in an SQLite database for possible future debugging or testing purposes (e.g., for detecting braking or acceleration events, or the orientation of the device in general), but these data were not synchronized with the service.

For practical reasons (e.g., limitations in the available server capacity), the user trial was not open to an unlimited number of users. A total of ten users participated in the trial, of which half were university personnel and the other half volunteers from the staff of the City of Pori and from a company participating in our research project. The users either used their own smartphones or borrowed one from the university. The user’s choice of car was not limited, but as the users generally drove their own cars, the selection of cars driven turned out to consist of smaller personal cars. A couple of users reported driving two different cars, so the number of cars was slightly higher than the number of users. The routes driven were a mixed set of commuting, work-related trips, and leisure. The majority of the driving involved consisted of driving from home to work, as reported by the users. This can also be seen in the collected data, as the same (identical) routes were driven on a daily basis.

Most of the driving was concentrated around the cities of Pori and Rauma, located on the west coast of Finland. Additional driving was done around the city of Tampere, which is located further inland, including the highway connecting Pori to Tampere. The distances were approximately 110 kilometers between Pori and Tampere and 50 kilometers between Pori and Rauma. Pori and Rauma are slightly smaller cities (with populations of about 85 000 and 40 000, respectively) whereas Tampere is the third largest city in Finland (with a population of about 232 000), although in the case of Tampere the routes driven were located mostly outside the city center. The routes are also illustrated in Fig. 6 (Section 5.3). The total duration of the testing period was about three months (from March 2018 to June 2018).

5.2 Results

The number of data points can be seen in Table 1, where the count and percentage figures of the data are grouped by different Shock Levels. The shock levels are arbitrary levels used for breaking down the data from the accelerometer readings. The first row ($L_{N/A}$) indicates the data points where the test device did not calculate the shock level. The highest level (L_4) represents the most intense values reported by the accelerometer. The levels can be recalculated afterwards for each device if needed. The shock levels are further discussed in Section 5.3.

Table 1. Breakdown of shock data points.

	$v \geq 0$ m/s		$v \geq 1$ m/s	
Shock Level	n	%	n	%
$L_{N/A}$	334730	69.3	312334	68.3
L_0	98367	20.4	98320	21.5
L_1	45083	9.34	42101	9.20
L_2	3419	0.71	3413	0.75
L_3	904	0.19	904	0.20
L_4	368	0.08	368	0.08
Total Count	482871	100	457440	100
Total Count with Level	148141	30.7	145106	31.7

The data point count on the left side of Table 1 includes all data regardless of the speed, and the right side omits speeds below 1 m/s. We have arbitrarily chosen 1 m/s to be the lowest speed recorded and taken into account in our test. This prevents the device from collecting data when the vehicle ought to be stationary, and helps to reduce the amount of unnecessary data.

In the further analysis of the data, only the pre-calculated shock level data where the speed is at least 1 meter per second are included ($n_{LEVEL} = 145106$). This represents approximately 30 percent of the total data collected. No further data have been eliminated from this data set. The relative percentage figures for each level in n_{LEVEL} are $L_0 = 67.7$, $L_1 = 29.0$, $L_2 = 2.35$, $L_3 = 0.62$, and $L_4 = 0.25$.

Tables 2 and 3 illustrate how the speed affects the measured shock intensity in the collected data. Rows 1 to 5 display the data of each individual level, while the last row ($L_{0—4}$) indicates the summarized information including each level. Table 2 indicates the average speed (v_{AVG}) and the standard deviation (v_{STD}) in each group. The average speed is quite similar on each level, while the standard deviation is only slightly lower on levels L_0 and L_1 than on the others. Additionally, the average speed and standard deviation of all data points (i.e., data with and without shock levels) was 68.0 km/h and 23.4 km/h. The respective values for data points without a shock level were 69.2 km/h and 21.6 km/h. The average speed and standard deviation information alone seem to support the fact that the reported shock levels occur around a speed of 65 km/h. However, when the data are further divided into speed-based intervals, the average speeds can be seen to be slightly higher, and about two-fifths of the data points are located above the 80 km/h limit.

Based on the data, it can be observed that algorithms used for detecting vibrations and road condition anomalies should cover at least the common urban area speed limits (from 40 km/h to 60 km/h) and preferably up to highway speeds (from 80 km/h to 100 km/h). In the area around the city of Pori, lower speeds were less represented than higher speeds. Thus, algorithms developed only for slower speeds would not be feasible for practical implementations.

Table 2. Average speed per shock level.

Shock Level	Speed (km/h)	
	v_{AVG}	v_{STD}
L_0	63.7	27.2
L_1	70.5	24.4
L_2	64.3	30.2
L_3	59.6	32.4
L_4	55.0	32.3
$L_{0—4}$	65.6	26.8

Table 3 displays the distribution of data points belonging to a given speed interval. There are six right-open intervals starting from 3.6 km/h (i.e., 1 m/s), and ending at 120 km/h. The last row ($L_{0—4}$) indicates the percentage share of data in each speed interval of all data points. The bulk of the data belongs to the lowest level. The lowest level (L_0) appears to be over-represented in the lowest three speed intervals (3.6—60 km/h) whereas a small amount of the percentage share seems to have shifted from the lowest level (L_0) to the next level (L_1) in the last two speed intervals (80—120 km/h).

It seems logical that higher speeds (i.e., greater energy) create more variance in the vibration detected by the sensor, but on the other hand, levels L_2, L_3, and L_4 appear slightly less often at higher speeds. It can only be speculated whether the reason is – for example – due to the better overall condition of roads with higher speed limits, or the fact that the phone/sensor is simply not able to record everything because it is not necessarily mounted in the car securely.

Table 3. Distribution of data points per shock level.

Shock Level	Data Point Distribution Based on Speed (%) Right-Open Intervals; km/h					
	[3.6, 20[	[20, 40[	[40, 60[	[60, 80[	[80, 100[	[100, 120[
L_0	76.9	70.8	78.2	68.2	60.8	63.0
L_1	17.8	25.5	19.1	29.5	35.9	32.9
L_2	3.50	2.51	1.90	1.74	2.53	2.87
L_3	1.28	0.76	0.53	0.43	0.54	0.91
L_4	0.56	0.40	0.25	0.14	0.20	0.29
$L_{0—4}$	7.83	13.6	14.8	22.1	36.7	4.99

Speeds above 120 km/h account for a negligible amount of data points (totaling 38 data points), thus the information is not shown in Table 3. Almost three-fifths (58.8 percent) of the data points are distributed between 60 and 100 kilometers per hour. The phenomena can be explained by two facts. First, the data collection was conducted mostly on longer distance journeys on the highways between major cities, corresponding to higher speed limits and a longer time spent on the road. Second, heavy traffic in the tested area is not commonly observed. More detailed information may be retrievable if the data are observed on the user/device level rather than on the global level. In future, it might also be worthwhile recalculating the data in four levels instead of five to obtain a clearer distinction between "good road condition" data and "bad road condition" data. Currently, levels L_0 and L_1 seem to overlap, and contain both data types.

5.3 Visualization

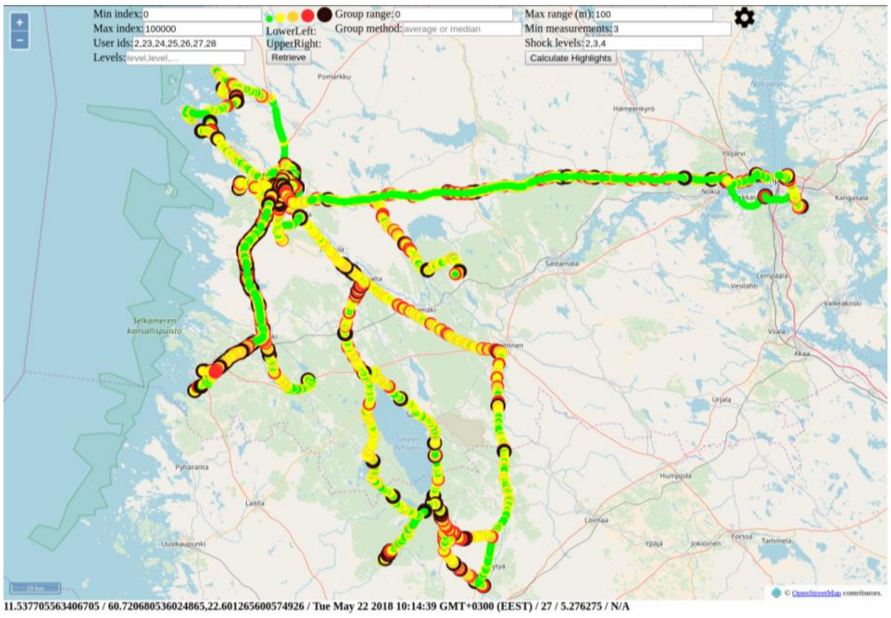
Five levels (0-4) were used for describing the detected condition of the road. The number of levels has no specific meaning, and another amount of levels could be chosen for more coarse or fine-tuned results. The levels are dynamically calculated per device, with level L0 being the "normal" of the device and L4 being the most extreme. In the current version of our application, the calculations do not take speed into consideration, even though speed does have an effect on the intensity of the measured values (e.g., variance). An exception to this is the exclusion of very low speed values (e.g., < 1 m/s), which could be caused by the user temporarily leaving the vehicle to walk about or be erroneous values caused by GPS inaccuracies when the vehicle is not in fact moving. In any case, even with-out utilizing the velocity data, the measured levels seem to correspond fairly accurately to the overall road conditions. Still, improved analysis of speed data could perhaps be used to further increase the accuracy of the level calculations.

In our case, the levels can be calculated either from the long-term data collected on the device (or from the data stored for testing purposes on the server), or by using a smaller data set, such as the data collected within the last 30 minutes. Ultimately, we decided to use smaller data sets when calculating the levels and showing the visualization on the map. The primary purpose of this was to minimize the effects caused by the user's change of vehicle as well as the cases where the user kept his/her device in a different holder or location on different trips. The test users also reported a few times when they had accidentally dropped the device, or the device had come loose from its holder. The former cases were fairly easy to recognize based on the reported, much higher than normal, acceleration values, but the latter cases tend to be erroneously detected as road condition problems.

In any case, the calculated levels should be fairly comparable regardless of the devices used, even when the individual values reported by the accelerometers are not. Unfortunately, rare cases where a user often changes vehicles remain a problem for detection. This problem would also be present if data were to be collected from, for example, public transportation utilizing the user's mobile devices.

The level markers and their use are illustrated in Fig. 6, Fig. 7, and Fig. 8. Fig. 6 shows a map using OpenStreetMaps, whereas Fig. 7 and Fig. 8 use Google Maps. The OpenStreetMaps implementation is slightly newer, but the features of both implementations are basically the same. One exception is the Street View functionality shown in Fig. 8, which is available only when using Google Maps. Both implementations also utilize the same underlying Representational State Transfer (REST) Application Programming Interfaces (API) provided by the cloud service.

The routes driven by the users are visualized in Fig. 6. The shock levels are illustrated by five colors (green, yellow, orange, red, and black – green being the best road condition, black the worst). The areas on the map are: the cities of Pori (top left), Rauma (bottom left), and Tampere (right). The various markers are also of slightly different sizes with the green "good condition" markers being the smallest and the black "bad condition" markers being the largest. This is in order to make the "bad condition" markers easier to spot among the data, which largely consist of green markers.



Min index: 0
Max index: 100000
User ids: 2,23,24,25,26,27,28
Levels: level,level,...
LowerLeft:
UpperRight:
Group range: 0
Group method: average or median
Max range (m): 100
Min measurements: 3
Shock levels: 2,3,4

```
11.537705563406705 / 60.720680536024865,22.601265600574926 / Tue May 22 2018 10:14:39 GMT+0300 (EEST) / 27 / 5.276275 / N/A
```

Fig. 6. Visualization of routes driven.

The user interface contains basic features for filtering data: viewing data from only a single user; excluding undesired shock levels, calculating highlights; selecting a specific date or time to observe; selecting the area to view; and the possibility to limit the number of level markers by only returning an average or median of the reported values within a certain area.

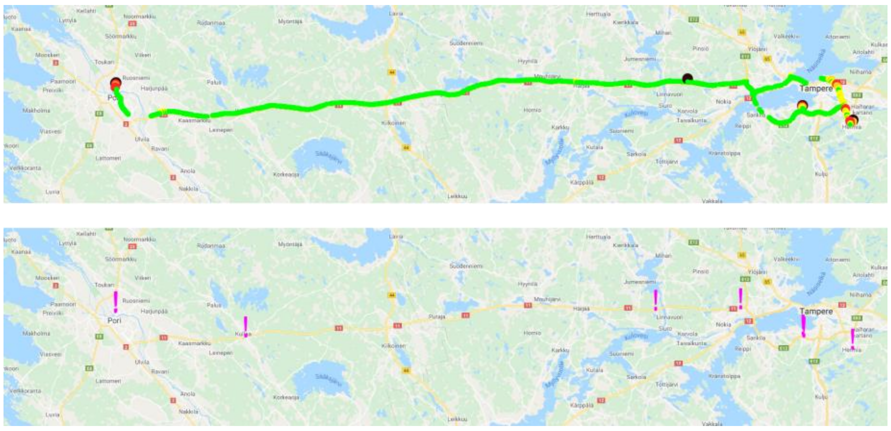


Fig. 7. Visualization of the route between the cities of Pori and Tampere.

The exclusion of undesired shock levels and highlights are illustrated in Fig. 7. The upper part of the figure shows basically the “raw data” selected from an area, in this case from a route between the cities of Pori and Tampere. In the lower part, the individual markers are removed and only the calculated highlights (exclamation marks) can be seen. The highlights represent an area where the measurements contain a large number of certain types of shock levels. The highlights can be calculated for any level, but naturally, are more useful for spotting places where there is a high concentration of “bad condition” markers. It would also be possible to show any combination of level markers with the highlights, e.g., red or black markers without green, yellow, and orange markers.

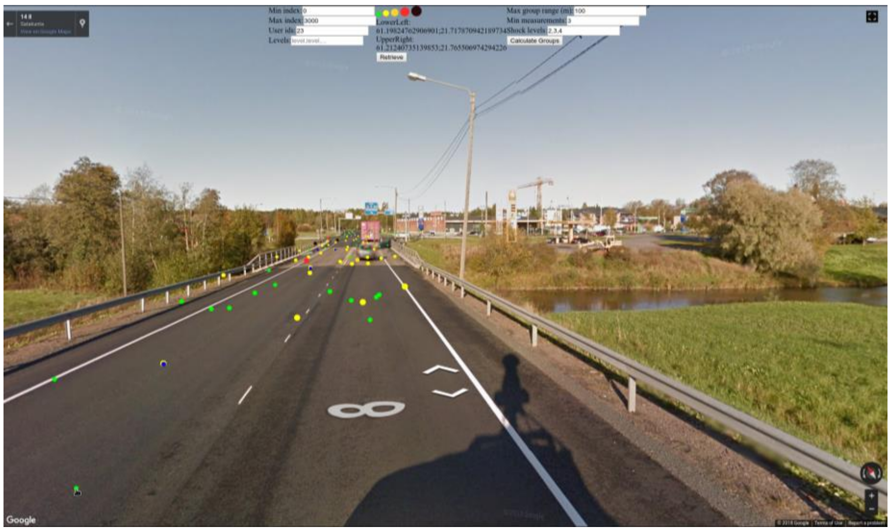

Min index: 0
Max index: 3000
User ids: 23
Levels: level,level...
LowerLeft: 61.19824762906901;21.717870942189734
UpperRight: 61.21240735139853;21.765500974294226
Max group range (m): 100
Min measurements: 3
Shock levels: 2,3,4

Fig. 8. Visualization in Google Maps Street View.

Finally, Fig. 8 shows the shock level markers in the Street View application. The Street View photos are not always up-to-date so the feature cannot be used as such to validate the results, but it can be used to give a quick look at an area. In this case, the cause of several orange, red, and black – “bad condition” – markers can be seen to be the bumps located on the entrance and exit sections of a bridge located on the highway.

6 Discussion

The basic programming task of creating a simple application for tracking the user’s location and gathering data from the basic sensors embedded in a mobile device is, in general, a straightforward process. Nevertheless, a practical implementation can pose both expected and unexpected challenges.

6.1 Technical Difficulties

We chose to use the Android platform because the authors had previous experience in Android programming. Unfortunately, the Android devices have hardware differences, which can affect the functionality of the application. In our case, there were two major issues. First, one of the older devices we used in our benchmarks lacked the support of a linear acceleration sensor, despite including a basic accelerometer. In practice, this means that all measured acceleration values included a gravity component without an easy or automated means of filtering the output. Filtering can be especially difficult on older models that do not contain proper rotation sensors that could be used to detect the orientation of the device.

Second, as it turned out, devices from different manufacturers and even different device models from the same manufacturer had variations in the reported accelerometer values, making direct comparison of values between devices challenging at best. Larger bumps are visible from the results regardless of the device, but smaller road surface features can become lost due to the device inaccuracies.

In practice, differences in the devices required the calculation of a "normal" for each device, against which variations in the data would be compared. Calculating a universal normal usable for all devices and users would probably be very difficult, if not entirely impossible. In any case, in laboratory conditions or in a controlled environment finding this normal is not a huge problem, but where a large crowdsourcing user and device base is concerned, finding the normal for each device can be a challenge. Additionally, the vehicle the user is driving can have a major impact on the detected values; after all, car manufacturers generally prefer to provide a smooth ride for the driver, and on the other hand, a car with poor suspension or tires can cause data variations that can be difficult to filter out. This also means that, if the user drives multiple vehicles, there should be a way for the application to either detect the vehicle used or adapt to the altered conditions.

In principle, the collected data could be analyzed to determine the device's normal, for example, if known "good condition" roads have been driven on. In practice, the data amounts (and the required server and network capacity) can be too extreme for this approach to be feasible. A better option would be to analyze the data on-device and the devices should only send the variances that exceed the calculated threshold values (i.e., detected potholes, roads of poor quality).

6.2 Interpretation of the Data

When examining the collected data set, the known places of data variance are visible, and in expected places. These include, among others, known roadworks, speed bumps, and bridge ramps, i.e., spots that the drivers cannot avoid can be easily seen in the collected data. Unfortunately, the same cannot be said about potholes or other larger, but in general, more infrequent road condition issues which are not always detected. We did not perform extensive studies to discover the driving habits of the users participating in our trial, although a quick interview revealed (perhaps unsurprisingly) that the drivers had tried to avoid driving into potholes.

In the initial phase of data analysis, validating the findings proved troublesome. As the drivers could drive along any road they wished, we did not have a clear idea of which of the roads driven were in bad shape or where on the road the bumps were located, nor was there available any conclusive database of speed bumps or other purpose-built road features that could be accidentally identified as road surface problems. Driving to the location of each detected bump for validation purposes in the case of a larger data set would be quite impractical. To get a basic idea of where the “bumpy” roads were located, the preliminary results were shared with the department of the City of Pori responsible for road maintenance and compared with their data. The data collected by the city are based on complaints received from road users or reported by the city maintenance personnel driving on the city roads. Thus, maintaining the data requires a lot of manual labor and the data are not always up-to-date. Nevertheless, this did give us some insight into the known conditions of the roads around the city. Furthermore, the discussion with the maintenance department gave a clear indication that an automated method for the collection of road condition data around the city would be a great help for the people responsible for road maintenance.

Moreover, collecting a sufficiently large data set with a very large user base could ultimately help in finding individual road problems as drivers would, for example, accidentally drive into potholes, but in our trials identifying specific road problems turned out to be quite challenging. On the other hand, the results showed, in a more general fashion, which of the driven roads were in the worst condition, and furthermore, which parts of a single road were in worse condition than the road on average. Both findings can be used for assessing road conditions, and with a much larger data set, even individual bumps could perhaps be more reliably detected.

A larger database is also advantageous in the elimination of unwanted data caused by individual random events – such as the user moving or tapping the phone during driving, sudden braking events or accidents – which could be erroneously detected as road condition problems. On the other hand, larger sets increase computing resource requirements and challenges in managing the data. In fact, even the amount of data collected in our user trials can be problematic. One of the main challenges is the visualization of large data sets.

For testing and validation purposes, all data generated by the mobile devices were stored on our server. Storing the “good condition” data can also help to map the roads the users have driven on as opposed to only reporting detected variations from the normal. Unfortunately, serializing the data – using JavaScript Object Notation (JSON) or Extensible Markup Language (XML) – and showing the measurements on a map in a web browser may be quite resource-intensive. Even when measurements are combined and indexed on the server to reduce the amount of transferred data, there can still be thousands of markers to be drawn on the map, especially if "good condition" data are included. Showing multiple roads in a large area simultaneously on a map can be a good method from a visualization point of view, but it can also make the web user interface sluggish or slow to load. For reference, loading and showing the map visible in Fig. 6 consisting of 100 000 measurement markers takes approximately 3—4 minutes, which is not an entirely impractical length of time for constructing the visualization, but can be an annoying delay when performing repeated work on the data set. Con-structing visualizations with smaller data sets (e.g., less than 10 000 data points), depending on the chosen filter settings, takes anything from a couple of seconds to almost half a minute.

6.3 Future Studies

One possible future action could be to open up the collected data for further analysis by other researchers. In general, the data are relatively easy to anonymize and do not contain any hard-coded user details. A method of generating anonymous data is also an advantage if a larger, more public user trial is to be performed in the future. Running the trials with a larger userbase would be one possible course of future action, although acquiring sufficient server resources for a wide-scale user trial could pose a challenge.

A less resource-intensive option could be to collect data for a longer period on a specific set of roads with the goal of discovering whether a gradual worsening of road conditions can be detected or how the results differ between winter and summer. Our current trials were run in spring and summer, and it is unknown how winter conditions would affect the results. Furthermore, the roads driven on were primarily paved and gravel roads were not included in the analysis of the data.

In addition, the increase in the number of dashboard cameras installed in vehicles, and the decrease in the prices of 360-degree cameras could provide an interesting aspect for data collection. The utilization of cameras could also make data validation easier during the trial phase, as there would be no need to go and check the detected road condition problems locally, or to use Google Street View or similar applications that may contain outdated images.

The Faucet-Sink-Drain model was used for the first time in an actual use case, and it could prove useful in other applications as well. However, the model requires more research and development to fully unlock its potential. Also, the framework [10] that is based on the model would require an actual implementation before more conclusions can be drawn of the model's usefulness.

Data security is an important factor that has not been addressed in this study. The prototype has basic user identification with username and password, but this was not used for filtering input data. Issues of data security, privacy, and anonymization of data need to be solved before commercialization.

7 Summary

This paper introduced a study that utilized data collected by sensors – primarily from an accelerometer and GPS – embedded in smartphones for detecting the condition of road surfaces. The data were obtained from a group of users driving on paved roads in western Finland. Furthermore, the test setup was described including a discussion on the challenges faced.

This paper showed how to combine a data gathering model and a data analysis model. Both of the models were applied and tested in the developed prototype system.

The results achieved from the trial period showed that even though the chosen methods could, in principle, find individual road surface problems (such as potholes), the results were more useful in the assessment of the overall condition of the road. In addition, the paper presented methods for visualizing road condition data collected from test users.

References

1. Krommyda, M., Sdongos, E., Tamascelli, S., Tsertou, A., Latsa, G., Amditis, A.: Towards Citizen-Powered Cyberworlds for Environmental Monitoring. In: 2018 International Conference on Cyberworlds (CW), pp. 454–457 (2018)
2. Satoto, K.I., Widianto, E.D., Sumardi., S.: Environmental Health Monitoring with Smartphone Application. In: 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 281–286 (2018)
3. Pyykonen, P., Laitinen, J., Viitanen, J., Eloranta, P., Korhonen, T.: IoT for Intelligent Traffic System. In: 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 175–179 (2013)
4. Yle Uutiset: Lapin Ely lupaa vähemmän lunta ja polanteita – Bussinkuljettajat keräävät tietoa Lapin teiden kunnosta. https://yle.fi/uutiset/3-9277596 (2016) Retrieved 27th. June 2018
5. Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., Jubert, I., Mazura, M., Harrison, M., Eisenhauer, M., Doody, P.: Internet of Things Strategic Research Roadmap. http://www.internet-of-things.no/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf (2009) Retrieved 23rd. March 2019
6. Hać, A.: Wireless Sensor Network Designs. Chichester, UK: John Wiley & Sons, Ltd. (2003)
7. Grönman, J., Rantanen, P., Saari, M., Sillberg, P., Jaakkola, H.: Lessons Learned from Developing Prototypes for Customer Complaint Validation. Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Serbia (August 2018)
8. Rantanen, P., Sillberg, P., Soini, J.: Towards the utilization of crowdsourcing in traffic condition reporting. 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Croatia, pp. 985–990 (May 2017)
9. Ma, M., Wang, P., Chu, C.-H.: Data Management for Internet of Things: Challenges, Approaches and Opportunities. In: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 1144–1151 (2013)

10. Sillberg, P.: Toward Manageable Data Sources. Information Modelling and Knowledge Bases XXX, Frontiers in Artificial Intelligence and Applications, vol. 312, IOS Press, pp. 101–111 (2019)
11. Sillberg, P., Grönman, J., Rantanen, P., Saari, M., Kuusisto, M.: Challenges in the Interpretation of Crowdsourced Road Condition Data. In: International Conference on Intelligent Systems (IS) (2018)
12. Howe, J.: The Rise of Crowdsourcing. https://www.wired.com/2006/06/crowds (2006) Retrieved 27th. June 2018.
13. Brabham, D.C.: Crowdsourcing as a Model for Problem Solving. Convergence: The International Journal of Research into New Media Technologies, vol. 14, no. 1, pp. 75–90 (February 2008)
14. Mao, K., Capra, L., Harman, M., Jia, Y.: A Survey of the Use of Crowdsourcing in Software Engineering. Technical Report RN/15/01, Department of Computer Science, University College London (2015)
15. Yi, C.-W., Chuang, Y.-T., Nian, C.-S.: Toward Crowdsourcing-Based Road Pavement Monitoring by Mobile Sensing Technologies. IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 4, pp. 1905–1917 (August 2015)
16. Y. A. Alqudah and B. H. Sababha, "On the analysis of road surface conditions using embedded smartphone sensors," in 2017 8th International Conference on Information and Communication Systems (ICICS), pp. 177–181, Jordan, April 2017.
17. Carrera, F., Guerin, S., Thorp, J.B.: By the People, for the People: The Crowdsourcing of "STREETBUMP": An Automatic Pothole Mapping App. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS), vol. XL-4/W1, no. 4W1, pp. 19–23 (May 2013)
18. Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The Pothole Patrol. In: Proceedings of the 6th International Conference on Mobile systems, applications, and services - MobiSys '08, Colorado, USA, p. 29 (June 2008)
19. Chen, K., Lu, M., Tan, G., Wu, J.: CRSM: Crowdsourcing Based Road Surface Monitoring. In: 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, China, pp. 2151–2158 (November 2013)
20. Alessandroni, G., Klopfenstein, L., Delpriori, S., Dromedari, M., Luchetti, G., Paolini, B., Seraghiti, A., Lattanzi, E., Freschi, V., Carini, A., Bogliolo, A.: SmartRoadSense: Collaborative Road Surface Condition Monitoring. The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), Italy (August 2014)
21. Dyo, V.: Middleware design for integration of sensor network and mobile devices. In: Proceedings of the 2nd International Doctoral Symposium on Middleware - DSM '05, New York, New York, USA: ACM Press, pp. 1–5 (2005)
22. Leppanen, T., Perttunen, M., Riekki, J., Kaipio, P: Sensor Network Architecture for Cooperative Traffic Applications. In: 2010 6th International Conference on Wireless and Mobile Communications, pp. 400–403 (2010)
23. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Networks: a Survey. Computer Networks, vol. 38, no. 4, pp. 393–422 (March 2002)
24. Saari, M., Baharudin, A.M., Sillberg, P., Rantanen, P., Soini, J.: Embedded Linux Controlled Sensor Network. In: 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1185–1189 (2016)
25. Saari, M., Sillberg, P., Rantanen, P., Soini, J., Fukai, H.: Data Collector Service - Practical Approach with Embedded Linux. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1037–1041 (2015)

26. Baharudin, A.M., Saari, M., Sillberg, P., Rantanen, P., Soini, J., Kuroda, T.: Low-Energy Algorithm for Self-controlled Wireless Sensor Nodes. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 42–46 (2016)