A Low Cost Adaptive Wireless Sensor Network for Accelerometer Data Collection

Ying Li, Peter Gould
Multiple Access Communications Limited
Southampton, UK
Email: ying.li@macltd.com, peter.gould@macltd.com

***Abstract*—This paper presents an accelerometer data collection system implemented with low cost off-the-shelf wireless sensor nodes. The design is focused on addressing some practical issues including effective sensor data offloading schemes suitable for different usage scenarios. Making the system highly configurable and adaptive in terms of selecting appropriate triggers for starting/stopping data collection and appropriate data offloading schemes was also a key design focus. The system has been developed for carrying out field tests, which involved attaching some sensor nodes to railway sleepers and gathering raw accelerometer data. The initial field test on an operational rail track proved the configurability and adaptability of our wireless sensor network system.**

Keywords-wireless sensor; sensor network; accelerometer.

I. INTRODUCTION

In recent years, there has been increasing interest in studying and developing wireless sensor network (WSN) systems that use accelerometers for monitoring problems relating to railway tracks or trains. [1] proposed a method of using accelerometers to detect arriving trains in order to warn maintenance personnel working on tracks. The authors of [2] and [3] investigated the detection and classification of train events by analyzing acceleration sensor data. [4] reported that by processing accelerometer readings measured at different locations of a train it is possible to distinguish between the vibration due to the train itself and the vibration due to deformation of the track. A prototype design of a WSN for monitoring a railway bridge is presented in [5]. It is evident from these studies that accelerometer sensor data is likely to play an important part in the application of the WSN technology to the railway sector. However, studies and research reported in the literature so far are mostly proposals, models and prototypes. What is needed is a robust WSN that could be used to make in-situ measurements in a reliable and flexible way.

Innovate UK has co-funded a two-year project called Smart Green Railway Sleepers (SGRS), which started in January 2014. One key aspect of this new project is to design, develop and pilot a wireless sensor-enabled tag and track system for use within railway sleepers (or railroad ties). Embedding sensors into railway sleepers opens a host of potential ways to improve the railway maintenance and sleeper recycling approaches. However, there is a range of challenging issues that need to be addressed before designing and embedding an optimal WSN system into railway sleepers becomes feasible, as highlighted in [6]. This paper reports on our experience of designing such an adaptive WSN system and provides an insight into practical issues encountered in collecting raw accelerometer data, along with approaches and methods used to address these issues.

The rest of this paper is organized as follows. Section II presents the main system requirements and a design overview. Section III details the challenging issues encountered in our design. Section IV presents our approaches and solutions. Section V draws conclusions to this paper.

II. SYSTEM REQUIREMENTS AND DESIGN OVERVIEW

Our aim is to design a low cost WSN for collecting raw accelerometer readings from sensors attached to railway sleepers while a train passes. The WSN should include a data sink node, a set of sensor nodes and optionally one or more relay nodes. The data sink node is connected to a laptop where an application controls the operation of the WSN system and also handles the collected measurement data, e.g., data visualization. One or more relay nodes are required in case the data sink node is located outside the radio range of the sensor nodes attached to railway sleepers. These requirements can be met by existing low cost off-the-shelf development boards, such as the CC2530ZNP-Mini kit from Texas Instruments [7].

The CC2530ZNP-Mini node includes two processors: a CC2530 system-on-chip running ZigBee Network Processor (ZNP) firmware and a MSP430F2274 microcontroller running application software, which controls the operation of the ZNP. Each CC2530ZNP-Mini node includes a 3-axis accelerometer that can be configured to sense acceleration in the range of ±2g or ±8g at a sampling rate of 10Hz, 40Hz, 100Hz or 400Hz. A CC2530ZNP-Mini node can be configured as a ZigBee coordinator, a ZigBee router or a ZigBee end device. We have used a coordinator node as a data sink node and end devices as sensor nodes.

An end device is designed to support the following simple operational states.

- ***Network Discovery***: At power on or wake up from sleep, a node searches for its network. If successful, it configures the accelerometer and enters the Wait for Event state. Otherwise, it enters the Sleep state.
- ***Sleep***: The Sleep duration is user configurable. At the expiration of the sleep time, an end device enters the Network Discovery state again.
- ***Wait For Event***: In this state, the MSP430F2274 stays in a low power mode until one of the following

 ISBN: 978-1-61208-391-9

three events occurs: 1) a trigger event for starting sensor data measurement; 2) the reception of a control message from the coordinator; 3) a detection of the network being lost. At the reception of a control message, the end device processes the message, e.g., changing motion detection threshold level, and remains in the current state. At the detection of the network being lost, the node enters the Sleep state. At the detection of a trigger event, the node enters the Data Measurement state.

- ***Data Measurement***: In this state, an end device measures accelerometer values and stores the measurement data locally. After a user configurable number of data samples have been collected, the end device enters the Data Offload state.
- ***Data Offload***: In this state, the sensor data collected by an end device is offloaded to the data sink node. Once the data has been offloaded, the end device goes back to the Wait For Event state.

The transitions between the above states are illustrated in Figure 1. Note that, when no network is present, an end device will enter the Sleep state in order to limit the power consumption of the end node.

The number of data samples collected and stored locally in a sensor node before being offloaded takes into consideration the memory available on the hardware platform and the usefulness of the measurement data. Among the available 32 KB FLASH space in a sensor node, 12 KB is reserved for buffering measurement data. This would give a maximum of 10 seconds worth of accelerometer readings at a sample rate of 400 Hz, which is sufficient for a useful fast Fourier transform (FFT) analysis.

To initiate a measurement session, the coordinator is plugged into a laptop via a USB port and forms a network for other nodes in its radio range to join. It has two main functions: 1) taking control and configuration commands from the application running on the laptop and transmitting them to end devices; 2) receiving accelerometer measurement data from end devices and passing them to the application running on the laptop where measurement data can be displayed graphically.

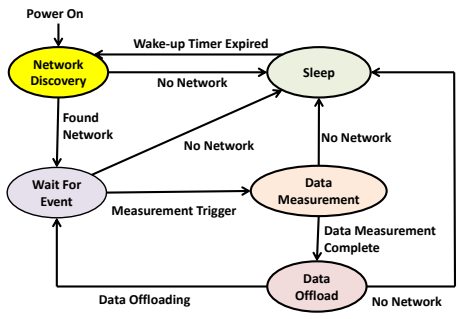

Figure 1. End device state transition diagram.

The data sink node is designed to pass each received packet to the laptop immediately for two reasons: firstly, there is not enough memory space for storing sensor data from more than one sensor node without adding external memory to the hardware platform; secondly, minimizing data processing at the data sink node reduces the time required for the measured sensor data to reach the laptop so that sensor data can be inspected in near real time.

III. CHALLENGING ISSUES

One of the challenging issues associated with this system is the design of data offloading schemes. Ideally, an end device starts to transmit collected measurement samples as soon as it enters the Data Offload state. The sooner a node offloads its collected data, the sooner it becomes ready for the next round of measurements. However, in an application where several nodes take sensor data measurements in a correlated manner, e.g., all sensor nodes start to take measurements as a train passes and want to offload their collected data as soon as possible, excessive packet collisions at the receiver of the data sink node would occur unless some sort of data offload scheduling scheme is deployed. Missing samples from a set of collected accelerometer readings could potentially lead to some distortions in data analysis results, rendering that whole data set useless. So, it is vitally important that a complete set of measurement samples reaches the data sink node reliably.

We could design a handshake process between the data sink node and each sensor node to schedule data offloading, e.g., messages to indicate which sensor node has data to offload and messages to dictate when and which sensor node should offload its data. This approach has at least two drawbacks. Firstly, this signalling would consume precious coding space in the MSP430F2274. Secondly, the signalling overhead increases when the number of sensor nodes increases and the channel condition gets poorer. For these reasons, we adopted an approach of making maximum use of what is available on the hardware platform.

The basic mechanism for reliable packet delivery in the presence of packet collisions and/or radio interference is packet acknowledgement and retransmissions. The ZNP offers two acknowledgement modes: medium access layer acknowledgement (MAC-ACK) mode and application support layer acknowledgement (APS-ACK) mode.

- ***In MAC-ACK mode***, the acknowledgement is from a neighbouring node. If there are multiple hops between a sensor node and the data sink node, receiving a positive MAC-ACK cannot be used as an indicator that a packet has reached the data sink node. The MAC-ACK mode is always on and cannot be disabled on this platform via the application programming interface (API). The two parameters governing the MAC-ACK mode operation are the maximum duration of waiting for an acknowledgement, T_{mac_wait}, and the maximum number of retries, N_{mac_retry}. These two parameters are also out of the control of the MSP430F2274 in this platform. However, the MSP430F2274 is informed of each packet transmission result: a positive result means that an acknowledgment to the packet has been received while a negative result

means that there is no acknowledgement for a packet after the packet has been transmitted N_{mac_retry} + 1 times and the T_{mac_wait} timer has expired after each transmission.

- ***In APS-ACK mode***, an acknowledgement is from the final destination node. Receiving an acknowledgement for a packet in APS-ACK mode is an indication that the packet has reached the ZNP of the data sink node. Unlike MAC-ACK mode, the APS-ACK mode can be enabled or disabled by the MSP430F2274 via the application programming interface (API). The maximum duration of waiting for an acknowledgement, T_{aps_wait}, and the maximum number of retries, N_{aps_retry}, for the APS-ACK mode operation are also under the control of the application processor.

One problem with MAC-ACK and APS-ACK is that an acknowledgement is sent by the ZNP. The ZNP and the MSP430F2274 processor in a node communicate via an internal serial peripheral interface (SPI). At the reception of a packet from a sensor node, the ZNP in the data sink node puts the packet into a buffer and notifies the MSP430F2274 that the packet is available via the SPI. ZigBee has an over-the-air data rate of 250 kb/s while the serial port data rate is configurable from 9.6kb/s to 115.2 kb/s. This means that while the MSP430F2274 processor is busy sending a received packet to the connected laptop, the next packet received and buffered by the ZNP may be over-written by subsequent packets, leading to the loss of one or more packets. To avoid losing packets in this way, one possible approach is for a sensor node to introduce a delay between receiving a positive acknowledgement for a packet and transmitting the next packet. If there is just a single sensor node offloading data, the sensor node could easily estimate the minimum delay. But in the case of multiple sensor nodes, estimating the required delay by a sensor node becomes difficult, especially when the number of sensor nodes offloading data varies. To ensure reliable end-to-end packet delivery, the application layer acknowledgement (APP-ACK) mode was introduced.

- ***In APP-ACK mode***, an acknowledgement is from the application layer of the final destination node. The two parameters, T_{app_wait} and N_{app_retry}, are used to denote the maximum duration for waiting for a positive acknowledgement and the maximum number of retries, respectively, at the application layer.

The three acknowledgement modes are illustrated in Figure 2. It is shown that for a relay node, the packet relaying function is taken care of by the ZNP and does not involve the application processor. For each acknowledgement mode, "PULL" type schemes could be designed, in which a data sink node actively schedules and controls which sensor node should transmit and when. This would require dedicated control signalling messages, which are not desirable for the reasons mentioned before.

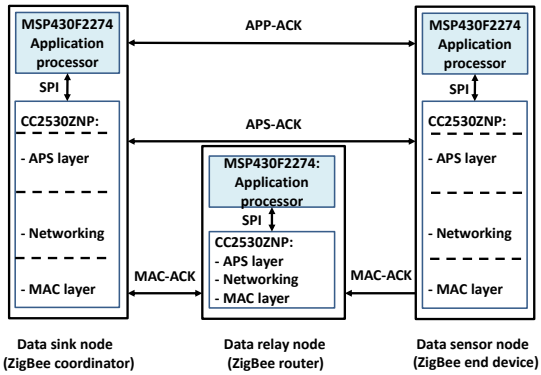

Figure 2. An illustration of three acknowledgement modes.

Therefore, simpler "PUSH" type schemes were adopted as described in the next section.

IV. DATA OFFLOADING SCHEMES

Three schemes are implemented in the system, each based on using one of the three acknowledgement modes.

A. Scheme based on MAC-ACK mode

A sensor node starts to offload its data to the data sink node as soon as it enters the data offload state. After each packet transmission, the ZNP sends the MSP430F2274 processor a positive or negative result depending on whether the MAC layer within the ZNP of the sensor node has received a MAC layer acknowledgment or not.

At the MSP430F2274 of an end device, when a positive result is received, the sensor node waits for a period of T_{app_pos} before sending the next packet. T_{app_pos} must account for, as a minimum, the time required for the data sink node to process a received packet, e.g., sending it through the serial port. When a negative result is received, the sensor node will wait for a period of T_{app_neg} before retransmitting the packet again. After N_{app_retry} retries without success, the sensor node will assume that the link to the data sink node has been lost and therefore go into the Sleep state. The data sink node does not perform any scheduling and simply passes whatever it receives to the connected laptop.

This scheme is simple and ideal for the usage scenario where the WSN consists of a data sink node and a single sensor node because acknowledgement is relatively fast in MAC-ACK mode. However, in the case of multiple sensor nodes, there is no guarantee of end-to-end packet delivery, although the reliability can be improved at the expense of increased data offload latency.

B. Scheme based on APS-ACK mode

This scheme works in the same way as Scheme A except that an acknowledgement is from the APS layer. If an APS layer acknowledgement is not received after a period of T_{aps_wait} after a packet being transmitted, the APS layer within the ZNP of the sensor node will retransmit the packet again. After N_{aps_retry} retries without success, the APS layer will

 ISBN: 978-1-61208-391-9

send a negative result to the MSP430F2274. Otherwise, a positive result will be sent to the MSP430F2274.

At the application layer, when a positive result is received, the sensor node waits for a period of T_{app_pos} before transmitting the next packet. T_{app_pos} must account for, as a minimum, the time required for the data sink node to process a received packet. When a negative acknowledgement is received, the sensor node will wait for a period of T_{app_neg} before retransmitting the same packet again. After N_{app_retry} retries without success, the sensor node will assume that the link to the data sink node has been lost and therefore go to the Sleep state.

This scheme can provide reliable end-to-end data delivery for the usage scenario where there is only one sensor node, but one or more relay nodes are required to relay packets from the sensor node to the data sink node. In the case of multiple sensor nodes, the end-to-end packet delivery reliability can be improved at the expense of increased data offloading latency, e.g., by increasing the value of T_{app_pos}, T_{app_neg} and/or N_{app_retry}.

C. Scheme based on APP-ACK mode

When this scheme is enabled, the APS-ACK mode will be disabled. A sensor node starts to transmit collected data as soon as it enters the Data Offload state. After a packet is delivered to the ZNP, the MSP430F2274 waits for an application layer acknowledgement for a maximum period of T_{app_wait}. If an acknowledgement is received within T_{app_wait}, the sensor node transmits the next packet after a delay of T_{app_pos}. If no acknowledgement is received when T_{app_wait} expires, the previous transmission is considered to have failed and the same packet is retransmitted after a delay of T_{app_neg}. After N_{app_retry} retries without success, the sensor node will assume that the link to the data sink node has been lost and therefore enters the Sleep state.

The MSP430F2274 of the data sink node issues an acknowledgement for each packet received from a sensor node in addition to sending each received packet to the connected laptop.

This scheme enables reliable end-to-end packet delivery regardless of the number of sensor nodes in the network and network topologies. Like the previous two data offloading schemes, this scheme also uses inter-packet intervals, T_{app_pos} and T_{app_neg}, to regulate traffic from the sensor nodes towards the data sink node. The key difference is that with this scheme the setting of these parameters can be dynamic as well as static. The dynamic setting is achieved by including the values of these parameters in an acknowledgement packet, making this scheme suitable for a range of usage scenarios. For example, when there is just a single sensor node, the parameter, T_{app_pos} for that node could be set to zero so that the sensor node can transmit the next packet as soon as an acknowledgement to the previous packet is received. When there are a number of sensor nodes, the parameter, T_{app_pos} for each sensor node could be set to a different value based on a user configurable priority list in the data sink node.

In summary, Table 1 lists the parameters applicable to the three data offloading schemes.

TABLE I. PARAMETERS APPLICABLE TO EACH SCHEME

Parameters		**Data offloading schemes**		
		MAC-ACK based	***APS-ACK based***	***APP-ACK based***
Parameters withinin MSP430F2274	T_{app_pos}	√	√	√
	T_{app_neg}	√	√	√
	T_{app_wait}			√
	N_{app_retry}	√	√	√
Parameters within ZNP	T_{aps_wait}		√	
	N_{aps_retry}		√	
	T_{mac_wait}	√	√	√
	N_{mac_retry}	√	√	√

Appropriate settings for these parameters depend on the data offloading scheme whilst the optimal data offloading scheme is in turn highly dependent on the usage scenarios. Considering the complex and dynamic nature of the environment in which our system is used, we have implemented our system such that most system operation parameters including the data offloading scheme, threshold level for triggering data collection, accelerometer sampling rate, etc., can be reconfigured on the fly during the system operation.

V. CONCLUSION

We have developed a low cost and adaptive WSN system for collecting raw accelerometer sensor data from railway sleepers. Our system is characterized by built-in adaptive sensor data offloading schemes and a remote control capability so that configurations can be changed after deployment. These features proved valuable for the initial field test involving collecting raw accelerometer data over a section of operational railway line where carrying out data collection reliably and quickly is crucial because accessing an operational rail track is very costly. In the initial field test, the live track was only accessible for a short time and some basic tests were performed. For example, different accelerometer configurations were used to allow the measurement of acceleration at different resolutions for various track conditions, e.g., train approaching, train passing sensor, train passing on adjacent track. The full benefits of our system will be explored in future field trials.

ACKNOWLEDGMENT

This work has been co-funded by Innovate UK.

 ISBN: 978-1-61208-391-9

REFERENCES

[1] L. Angrisani, D. Grillo, R. Moriello, and F. Filo, "Automatic detection of train arrival through an accelerometer," Instrumentation and Measurement Technology Conference (I2MTC), May 2010, pp. 898–902, ISSN: 1091-5281, E-ISBN: 978-1-4244-2833-5.

[2] E. Berlin and K. Van Laerhoven, "Sensor networks for railway monitoring: detecting trains from their distributed vibration footprints," IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), May 2013, pp.80–87, Print ISBN: 978-1-4799-0206-4.

[3] E. Berlin and K. Van Laerhoven, "Trainspotting: Combining fast features to enable detection on resource-constrained sensing devices," The Ninth International Conference on Networked Sensing Systems (INSS), June 2012, pp.1–8, E-ISBN: 978-1-4673-1784-6.

[4] C. Wang, Q. Xiao, H. Liang, and X. Chen, "On-line vibration source detection of running trains based on acceleration measurement," IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2006, pp.4411-4416, E-ISBN: 1-4244-0259-X.

[5] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar, "BriMon: A sensor network system for railway bridge monitoring," The Proceedings of the 6th International Conference on Mobile Systems, applications and services, 2008, pp. 2-14, ISBN: 978-1-60558-139-2.

[6] Y. Li and P. Gould, "Embedding Wireless Sensors in Railway Sleepers – Challenges and Choices," Multiple Access Communications Limited White Paper, available from http://macltd.com/publications. [Retrieved: January, 2015].

[7] http://processors.wiki.ti.com/index.php/CC2530ZDK-ZNP-MINI. [Retrieved: January, 2015].

 ISBN: 978-1-61208-391-9