Wireless Sensor Network Protocol for Smart Parking Application Experimental Study on the Arduino Platform

Luis Ostiz Urdiain, Carlos Pita Romero
Computer Engineering and Networking Department
Universitat Ramon Llull, La Salle
Barcelona, Spain
Email: luis.ostiz@gmail.com, carlus.pita@gmail.com

Jeroen Doggen, Tim Dams, Patrick Van Houtven
Department of Applied Engineering and Technology
Artesis University College Antwerp
Antwerp, Belgium
Email: jeroen.doggen@artesis.be, tim.dams@artesis.be, patrick.vanhoutven@artesis.be

***Abstract*—The paper presents an Arduino-based wireless sensor network to monitor parking lots using a non-standard low-power energy-balanced system. The event-driven routing protocol follows the hierarchical clustering philosophy. Energy is saved by minimising the number of transmissions needed to forward information to the base station. The smart sensor platform is build using the popular Arduino development platform, Sharp IR distance sensors and nRF24 low-power radio modules. Our practical results show that this platform is easy to use, but not the most appropriate platform to develop low-power wireless sensor network applications.**

Keywords-Wireless Sensor Network (WSN); Smart Parking Application; Clustering; Arduino; nRF24.

I. Introduction

With the rapid proliferation of vehicle availability and usage in recent years, finding a vacant car parking space is becoming more and more difficult, resulting in a number of practical conflicts (e.g., time, environmental [1]). Using Wireless Sensor Network (WSN) technology, we propose a low power solution to this parking problem.

A WSN consists of a large number of smart sensors which form a multi-hop network by radio communication in sensor fields. They measure and process information gathered from the sensing area and transmit it to the data base station. WSN can be used in many fields such as environment monitoring, intelligent transportation and smart homes [2]. In our system, we monitor a group of ten sensors, each one detecting periodically if there is a car parked a parking lot. This information is forwarded to a sink node. The sink node itself is connected to a management center, which is not yet a part of this research. The management center can help other drivers to discover areas with available parking lots.

The sensors nodes are based on the popular Arduino platform [**?**]. Arduino is an open-source electronics prototyping platform with flexible and easy-to-use hardware and software. We are developing a network protocol on top of nRF24L01 radio modules [3]. We have taken some features from other network WSN protocols, in particular clustering hierarchical protocols, such as Low Energy Adaptive Clustering Hierarchy (LEACH) [4] and Threshold sensitive Energy Efficient sensor Network (TEEN) [5].

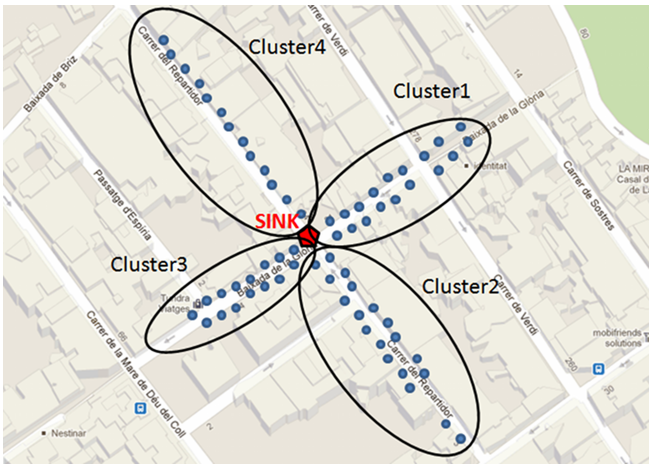

Figure 1. Proposed Single-Hop Clustering Topology

The remainder of the paper is organised as follows. In Section II, we discribe the system design the node hardware architecture and the software libraries. In Section III, we explain the design of the layer 2 and 3 WSN protocol operation. The research questions and experiments are in Section IV. Finally, we describe future work in Section V.

II. System Design

For this experimental study we developed network of smart objects based on the Arduino development platform.

A. Cluster Network

The proposed single-hop clustering topology is shown in Figure 1. The sensor network is divided in 5 groups, one for each cluster and one for the sink node. Each group uses a specific RF channel. The sensors detect if there is a car in the parking lot. The sensor will be located in a strategically fixed position on the floor and will sense periodically to detect cars. Knowing our nodes are deployed in a fixed position, we decided that the cluster members are invariable.

 ISBN: 978-1-61208-235-6

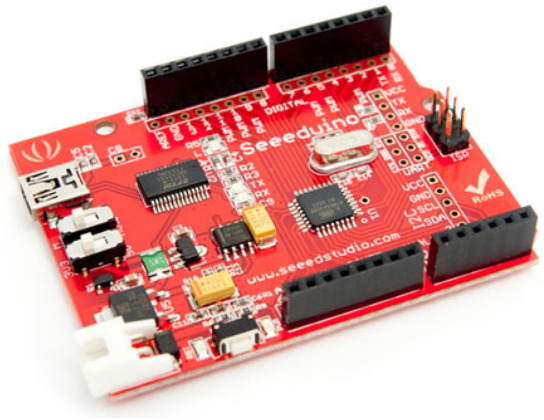

Figure 2. Seeeduino board

1) Sensor Node (SN): A SN periodically measures the distance with the sensor and compares it with the last measurement. If the result has changed it sends an advertisement message (ADV) to the cluster head.

2) Cluster Head (CH): The CH monitors a parking lot, but at the same time, it waits for advertisement messages from the other SNs. It aggregates the information and forwards it to the sink node. This role rotates between all the nodes within the same cluster.

3) Sink Node: This fixed node is continuously listening on designated channels to forward packets, coming from the cluster heads, to the management system.

B. Node Architecture

We use the Seeeduino platform [6], an Arduino-compatible development board, but with many improvements on the hardware side, e.g., energy efficient surface mount device (SMD) components and extra analog and digital I/O pins. The Atmel ATmega328P microcontroller has 32 KB integrated flash memory with read-while-write capabilities, 1 KB EEPROM and 2 KB SRAM. We use a dedicated nRF24L01 low power radio module working in the 2.4 2.5 GHz ISM band. The radio uses FSK modulation type and can operate within 125 RF channels with 250 Kb/s, 1 or 2 Mb/s of data rate. We use the GP2Y0A21YK [7] infra-red proximity sensor made by Sharp, which is a wide-angle distance measuring sensor that delivers an analog output varying from 3.1 V at 10 cm to 0.4 V at 80 cm. We use 6 external 1.2 V rechargeable NiMH AA batteries (1300 mAh) providing 7.2 V per SN and a USB powered sink node. As the energy levels of the batteries decrease, the battery voltage goes down. In practice, the module stops working properly below the 6 V cut-off voltage.

C. Software Libraries

The source code of all our libraries are publicly available under de GNU Lesser General Public License on their respective Google Code website [8].

1) Low-Power Library: This Arduino library, developed by Rocket Stream Electronics, allows us to put the microcontroller into different power saving modes [9].

2) nRF24 Module Library: This library for the RF module, developed by J. Coliz [10], provides the physical layer functions: transmitting and receiving packets. It also allows us to change the RF channel, set the retransmission options and power down the chip.

3) Distance Sensor Library: We developed this library to use Sharp IR Distance Sensors in Arduino projects. We use the sensor output to fetch the distance from a lookup table holding a sensor-specific transfer function.

4) Cluster Network Library: We developed this library that implements the layer 2 and 3 operation of the system. This library, described in detail in Section III is build on top of the nRF24 library.

5) Detecting Car Library: We assume that a car is occupying a parking lot when the distance value is between 10 cm and 60 cm for several seconds. If the distance value exceeds 60 cm, the parking lot is available. If the value is below 10 cm, the measurement is considered invalid.

6) Node Energy Library: Library which provides the current energy level depending on the time spent in the different phases. It is used to decide which node has the highest energy level during CH selection.

III. WSN Protocol Design

We are developing and testing a non-standard wireless communication protocol based on some features of clustering hierarchical protocols such as LEACH [4] and TEEN [5].

Low Energy Adaptive Clustering Hierarchy Protocol (LEACH) is a very common WSN protocol. LEACH is based on the division of the network nodes in clusters and the election of a set of temporary CHs. The CHs are in charge of their part of the network and aggregate collected data for later delivery to a sink.

Our implementation is based on the LEACH clustering philosophy but instead of using a selection procedure based on probability, we use the remaining energy level to rotate the CH role periodically [11].

In TEEN, the environment is sensed constantly, but the transmission of the information to the CH is done only when the data exceed the upper and lower threshold previously defined by the CH. Therefore, this protocol reduces the number of the transmissions to the necessary, only when the measured parameter value is outside the range of interest.

Our measurement model to detect cars is based on this concept. A sensor only transmits the parking lot status when it detects an arriving or a leaving car. We can assume a sensor will not have new information to report very often. Thereby, we lower the number of necessary transmissions to a minimum, thus increasing the system's lifetime.

 ISBN: 978-1-61208-235-6

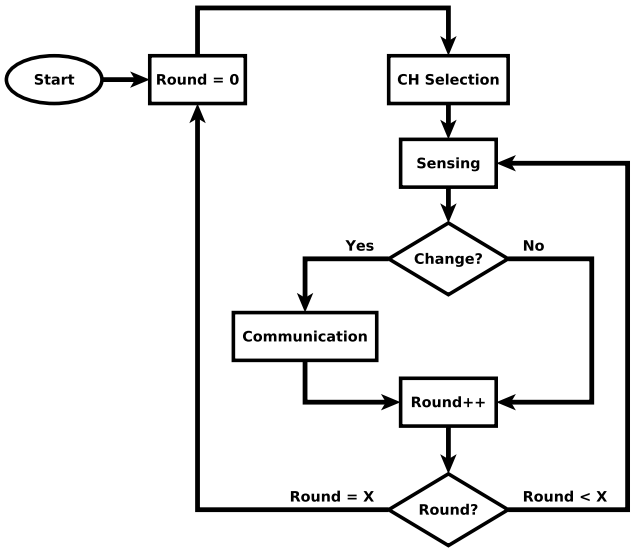

Figure 3. Flowchart of System Operation

A. Layer 2 Operation

In the collision avoidance mechanism we implemented, every node that wants to transmit must first listen to the channel. If it is free, the node transmits the packet and waits for an acknowledgement. If it is occupied, the node waits for a back-off time and listens to the channel again. After a number of unsuccessful tries, the packet is dropped. The optimal amount of tries is still being evaluated.

B. System Operation

The system operation, shown in Figure 3, is divided in three phases: CH selection, sensing, communication.

1) Cluster Head Selection Phase:

- The CH broadcasts an Energy Request message.
- The SNs measure their energy level and send it to the CH with an Energy Reply message.
- The CH collects all replies and compares the energy levels. The node with the highest energy level is selected as the new CH.
- CH broadcasts the new CH ID to all the SNs.
- The SNs update the CH ID at the same time.

2) Sensing Phase: The node detects if there is a change in the parking lot status. If there is one, it goes in to the *Communication Phase*, otherwise the node goes to sleep mode. Because the SNs are synchronised during the CH selection, they hould wake up at the same time.

3) Communication Phase: The node report the updated parking lot status to the CH by sending an ADV message. Meanwhile, the CH collects the ADV packets of all SNs. The CH sends an aggregated data packet (DATA), which contains all the collected status updates, to the sink.

IV. Experimental Study

Our goal is to develop an energy-efficient implementation of a smart parking application. Some of the research questions in this project are:

1) Is the Arduino platform a good choice for this specific WSN application?
2) How can we minimise energy consumption per node to maximise network lifetime?
3) How do we optimise the CH selection algorithm?
4) How reliable is this system?

To evaluate the performance of the protocol, we propose two different variations of the same practical scenario. The sink is positioned in the middle of a square area and 10 SNs are positioned around the sink.

1) The whole network has been divided in 2 clusters, with 5 SNs per cluster.
2) The network is formed by 10 SNs which are part of the same cluster.

A. Energy Consumption

To minimise the energy consumption, we maximise the sleep time by avoiding needless idle time and communication. To know how much energy is consumed per node, we defined different functional states depending on different energy hardware power requirements based on the low-power library specifications [9]. We measure the current consumption in each state to calculate the total energy consumption.

These defined states are: transmit, receive, idle and sleep.

First experiments on the energy consumption of the module showed a very inefficient sleep mode because the distance sensor was always powered up. An option to disable the sensor will be added to the distance sensor library.

B. Packet Loss Ratio

A set of counters were added in order to record the number of packets transmitted successfully and packets lost. These measurements were made using a 90-minute recording time with a bit rate of 1 Mb/s. In the first case the packet loss was 41 %, in the second case 72 %. Further research is needed to solve this issue. A possible cause of this very high packet loss ratio is bad synchronisation.

C. Synchronisation

The ATmega328P internal oscillator has a significant error margin when the board wakes up from sleeping mode. Because this error is different in each board, the SNs will not be able to communicate with the CH when a few rounds have passed. We implemented a non-optimal software-based solution around this hardware limitation:

1) The first round after a new CH is selected, the SNs wake up 4.8 seconds before the CH ($24 \times 0.2 = 4.8$).
2) Once the CH wakes up, it broadcasts a SYNC message.

 ISBN: 978-1-61208-235-6

3) The SNs receives the SYNC message from the CH calculated the time between waking up and the SYNC message.
4) The SN calculates its own sleeping time by taking the waiting time for the SYNC message in to account and adding one second as a safeguard.
5) In the second round after the CH selection, the procedure is similar, with a safeguard time of 100 ms.
6) From now on, the non-CH sleeps the calculated time.

V. Conclusion and Future Work

We proposed a low-power WSN solution for a smart parking application. Along its development, the work has led to new challenges and opportunities that might be interesting to address.

A. Further Development of the Application

The next step in the development of a usable application is server-side application that manages all sensor information, such as its geographical position, parking lot types and parking lot status. The user would be able to connect to the server using a website or with mobile phone application to request the location of available parking lots. Another useful idea is developing an application to facilitate monitoring of the smart parking network. Currently we are developing an embedded display module for in-car placement similar to [12]. This module sends requests over the Internet to the central server to find free parking spaces. When the car arrives in the correct street it can request up-to-date information from the CH.

B. Protocol Enhancement

In order to polish the protocol, we propose to improve its scalability and enable more complex data aggregation. In a next phase it might be useful to have multi-hop communication and location aware CH selection [13]. One important aspect that is currently not considered is security. It could be a really interesting to introduce security techniques which would not affect the protocol's and the system's behaviour.

C. Conclusion

We proposed a practical implementation of an event-driven WSN clustering protocol. Our protocol is hierarchical, all nodes are divided in clusters with one CH. We implemented an energy-aware CH selection algorithm similar to the LEACH protocol.

The system was implemented using Seeeduino development boards, nRF24L01 low-power RF modules and Sharp IR distance sensors. We implemented a software based synchronisation mechanism to solve the problems caused by the inaccuracy of the Arduino internal Timer.

These results show that the battery, sensor and radio are not the optimal hardware choices for this WSN Application. Although Arduino is easy to use as an experimental open-source platform, it is currently not the most appropriate platform to develop low-power WSN applications.

References

[1] M. Idris, Y. Leng, E. Tamil, N. Noor, and Z. Razak, "Car park system: A review of smart parking system and its technology," *Information Technology Journal*, pp. 101–113, 2009.

[2] Y. Li, M. Thai, and W. Wu, *Wireless Sensor Networks and Applications*, ser. Signals and Communication Technology. Springer, 2008, [Retrieved: August, 2012]. [Online]. Available: http://books.google.be/books?id=x1MR5Ct-rp8C

[3] Olimex Ltd., "nRF24L01 Single Chip 2.4 GHz Transceiver Datasheet," [Retrieved: August, 2012]. [Online]. Available: http://www.olimex.com/dev/mod-nrf24L.html

[4] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," in *33rd Hawaii International Conference on System Sciences*, 2000, pp. 1–10.

[5] A. Manjeshwar and D. P. Agrawal, "TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks," in *International Workshop on Parallel and Distributed Computing Issues in Wireless Sensor Networks and Mobile Computing, San Francisco, USA*, 2001, pp. 2009–2015.

[6] "Seeeduino v2.21," [Retrieved: August, 2012]. [Online]. Available: http://www.seeedstudio.com/wiki/Seeeduino_v2.21

[7] Sharp, "Sharp GP2Y0A21YK IR Distance Sensor Datasheet," 2005, [Retrieved: August, 2012]. [Online]. Available: http://www.sharpsma.com/webfm_send/1208

[8] J. Doggen, L. Ostiz, and C. Pita, "Software Libraries for the Arduino Platform," [Retrieved: August, 2012]. [Online]. Available: https://code.google.com/u/104098523773938750140

[9] Rocket Scream Electronics, "Lightweight Low Power Arduino Library," 2011, [Retrieved: August, 2012]. [Online]. Available: http://www.rocketscream.com/blog/2011/07/04/lightweight-low-power-arduino-library

[10] J. Coliz, "nRF24L01(+) 2.4 GHz Wireless Transceiver Driver," 2011, [Retrieved: August, 2012]. [Online]. Available: http://maniacbug.github.com/RF24/

[11] K. Ramesh and K. Somasundaram, "A comparative study of clusterhead selection algorithms in wireless sensor networks," *International Journal of Computer Science & Engineering Survey (IJCSES)*, pp. 153–164, 2012.

[12] V. P. Thakare and N. A. Chavan, "Performance evaluation of parking guidance and management system using wireless sensor network," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 1, pp. 96–102, 2012.

[13] T. V. Bhuvaneswari and V. Vaidehi, "Enhancement techniques incorporated in leach- a survey," *Indian Journal of Science and Technology*, vol. 2, pp. 36–44, 2009.

 ISBN: 978-1-61208-235-6