Toolkit for Building 802.15.4 and ZigBee-based Pervasive Learning Games

Imran A. Zualkernan
Computer Science and Engineering, American University of Sharjah, Sharjah, UAE
Email: izualkernan@aus.edu

***Abstract*—Ad-hoc wireless network technologies like ZigBee and 802.15.4 have been used in a variety of domains including home automation. With current developments in cross-protocol integration, sensor fusion and lower costs, these technologies also lend themselves naturally to other consumer applications. One emerging area of such consumer applications is pervasive and learning games. With the realization that children are not getting enough exercise, parents and teachers are interested in their children engaging in higher levels of physical activity. Pervasive and tangible learning games superimpose networking and sensor technologies on familiar outdoor game formats to create games that make children engage in physical activity in addition to learning. This paper presents the design and development of an open-source toolkit and the associated middleware that allow developers to build a variety of pervasive learning games using ZigBee and 802.15.4. Various sample learning games constructed by using this toolkit are also discussed.**

***Index Terms*—ZigBee, game-based learning, wearable computing, ubiquitous learning, pervasive learning**

I. Introduction

Zigbee and IEEE 802.15.4 have traditionally been used in a variety of application areas including consumer electronic devices, energy management, commercial building automation, industrial plants, and in home automation [1]. ZigBee has often also been used in conjunction with other standards. For example, [2] developed an active Radio Frequency Identification (RFID) system using a multi-hop ZigBee network as the backbone. Middleware to bridge different protocols is often used to facilitate use of ZigBee with other networking standards. In addition, [3] have a developed a hybrid wired-wireless system that utilizes ZigBee for the wireless functionality and seamlessly integrates network services using a middleware. Similarly, [4] have proposed another system that integrates a General Packet Radio Service (GPRS) data collection node with a host of ZigBee devices connected to sensors to detect flooding. In addition to being used in hybrid networking environments, there is an increasing trend towards using ZigBee in personal area networks or wearable devices. For example, [5] describe a ZigBee-based wearable system that collects various physiological parameters like heartbeat etc. to detect events like falling, and to report these in a smart home environment. Similarly, [6] proposed a system that uses a 3-axis accelerometer to detect the posture of a wheelchair user, and to transmit this information using the ZigBee network. Authors in [7] described a wearable ZigBee based system that uses optical fiber curvature to detect the change in joint angles. In another example, [8] show an application that uses Bluetooth, ZigBee and RS232 in unison to collect physiological data; this data is sent to the cloud using a mobile phone. Finally, [9] proposed a localization system for ZigBee networks that is able to localize assets within a range of 2 meters.

The use of hybrid wired and wireless networks, an ability of incorporation into wearable devices and clothing, the low-power consumption, integration with physiological and physical sensors, and a possibility of position localization make ZigBee an ideal candidate for building consumer devices that implement a new class of learning games called pervasive and tangible learning games. These games are typically modeled after playground games like Tag etc. and superimpose technology and wireless and wired networking to gain learning effects.

This paper presents a toolkit that makes it easier to develop this class of learning games that employ off-the-shelf ZigBee hardware for wireless communications. The rest of the paper is organized as follows. Next section presents a brief introduction to pervasive learning games. This is followed by a description of the architecture of the open-source toolkit. The following section presents a number of case studies where the toolkit has been used to implement pervasive learning games. The paper ends with a conclusion.

II. Pervasive Learning Games

Game design and game-based learning has been extensively explored [10], and studies show that game-based learning tends to improve student engagement and motivation [11] [12]. With the realization that children today spend a significant amount of time in front of computers and television, and do not get enough exercise, there is an emergence of learning games where children engage in physical activity by interacting with their environment while learning [13]. For example, [14] describe a pervasive game where children learn medieval history by walking around a town and using their mobile phones. Similarly, [15] describe a pervasive game for 7-

Manuscript received December 28, 2016, revised May 14, 2017.

doi: 10.12720/jait.8.2.92-99

11 year old children where small microcontrollers are embedded in children's play environments. In addition to enabling the game environment by using technology, pervasive games also use physiological features of children as a part of the game. For example, [16] implemented a game that uses a small device measuring children's heart beat and broadcasts it to other children to play an enhanced version of the tag game. Authors in [17] have also used technology-augmented tangible objects to explore open-ended play in young children. Finally, [18] have proposed a pervasive version of the hide-and-seek game to teach mathematics to children.

Fig. 1 shows a prototypical example a pervasive learning game incorporating tangible and wearable components.

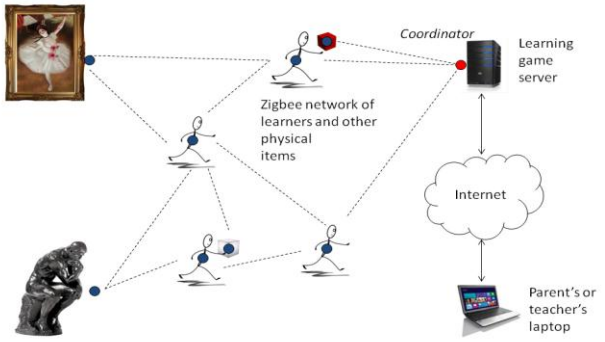

Figure 1. A prototypical example of a pervasive learning game incorporating tangible and wearable components

As Fig. 1 shows, the example learning game is played in a museum where children are required to run around the museum and to identify and answer questions about various art objects. Each student wears a ZigBee end-node device interfaced to a small microcontroller. Similarly, art objects like paintings or sculptures may also have ZigBee end-devices attached to them. Children and other items may also be interfaced to sensors. For example, an RFID reader may be available at each art object. Each child has a unique RFID tag sewn into their shirt and they may swipe the RFID tag against the reader to indicate that they have found the art object. The children may also be asked to carry other tangible devices that themselves include ZigBee end-nodes. For example, the tangible device may have a MP3 player that plays music from a specific time period and the child is required to find an art object from that period. After finding the appropriate art object, the child may leave the tangible device there for the next team member for a subsequent stage of the game.

In summary, pervasive learning games required a tight integration of sensors and various types of ZigBee nodes in addition to potentially complex game-logic.

III. Architecture of the Toolkit

The conceptual architecture for building pervasive learning games is shown in Fig. 2. As the Figure shows, there are two primary components; the Learning Game Server (LGS) and various types of client nodes like the Player node or the Tangible item nodes.

A. The Learning Game Server (LGS)

The LGS is divided into three distinct modules; the learning game logic engine (LGE), proxy objects for the various actors in the game, and the Middleware. Use of proxies with ZigBee for interface purposes is not new. For example, [19] have proposed a scheme of using Zigbee proxies to connect with the OSGI home automation networks.

In the proposed architecture shown in Fig. 2, LGE is responsible for implementing the gaming logic. For example, if the game consists of finding the paintings from a particular period, the GLE will have a database of paintings from that period available in the museum. In addition, the game logic may include scoring rules and names and teams of the students participating.

The second components of LGS are software proxy objects that encapsulate the behavior and communication for each of the physical actors including players and other tangible items participating in the game. For example, if ten paintings are included in the game, then there would be one proxy object for each painting. In addition, one proxy object is created for each student participating in the game. The purpose of the proxy objects to hide details of the ZigBee protocol from the developer. The GLE simply sends and receives messages from these proxy objects to enact the game. For example, in the beginning of each game cycle, the game engine may send the GAME_RESET message to each of the proxy objects indicating that a new game is starting. In doing so, the game engine is not concerned about the mechanics of how the message is delivered to each player or to items in the physical world.

Finally, the third component of the learning game server is the Middleware which is responsible for providing all the services required to transmit and receive the various messages from proxy objects, and to establish a bi-directional communication between the proxy objects and the physical nodes through the ZigBee network.

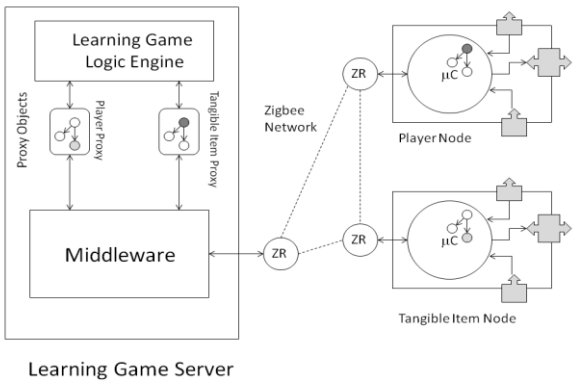

Figure 2. Architecture for building pervasive and tangible learning games where a learning game server connects to a variety of client nodes representing children or physical actors using the ZigBee network

B. Client Nodes

Client nodes are game players or tangible objects that participate in the learning game. As Fig. 2 shows, the game server is physical connected to one ZigBee radio node (ZR) on the ZigBee network which typically acts as the Coordinator to route messages to and from the game server. As Fig. 2 also shows, each game player or tangible item has an attached microcontroller which is also connected to a ZigBee node (typically an end-device). Depending on the application, each microcontroller may read various sensors, and act on actuators like turning LED's on and off, for example. In addition to communicating through the ZigBee node, the microcontroller also implements the game logic on the client side. For example, a painting may send a message to the game application indicating that a child with a particular RFID tag has just swiped their tag indicating that the child has found the painting. Like the LGS, the microcontroller communicates through sending and receiving messages using a natively compiled library provided with the framework.

C. Proxy Objects and Messaging Architecture

Fig. 3 shows the detailed structure of software proxy objects within the LGS. In addition to domain information about the object itself like name and various attributes (e.g., name of the painting and its creator), each proxy objects receives APP_MSGes from the GLE. APP_MSG are domain specific messages for each type of game which are specified by developer. For example, START_GAME and END_GAME are the two simplest messages that must be implemented. As Fig. 3 shows, the incoming messages are queued by each proxy object and transformed into ZB_MSGes to be sent over to the ZigBee network.

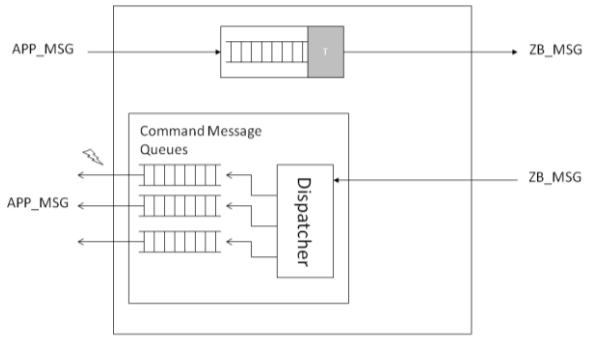

Figure 3. Structure of proxy objects within the Learning Game Server where incoming and outgoing messages are queued and transformed in addition to providing individual message queues for the various game commands

The ZB_MSG is an abstraction of a particular protocol implemented by a specific hardware/firmware implementation of the ZigBee chipset being used to establish the ZigBee network. Fig. 4. shows this mapping for one specific ZigBee chipset. As Fig. 4 shows, the Payload field of a ZB_MSG contains an APP_MSG which consists of a start code followed by COMMAND_ID. COMMAND_ID is defined by the developer and represents a message in the learning domain. For example, START_GAME is one COMMAND_ID. COMMAND_ID is followed by a 16 bit unique transaction ID which is used to uniquely identify acknowledgement indicating that a particular message has been received by the intended recipient. This filed is followed by the Payload length that can contain any arbitrary application specific data. The ZB_MSG ends with its own checksum.

The Middleware module shown in Fig. 2 asynchronously constructs ZB-MSG messages from the incoming byte stream and also assembles and sends ZB_MSG messages to the output stream to the ZigBee network. Like the outgoing messages, incoming ZB_MSGes from the ZigBee network are forwarded to each proxy object based on their unique address by the Middleware. Upon receiving the ZB_MSG intended for it, as Fig. 3 shows, the Dispatcher module within the proxy object converts the message into an APP_MSG and queues the message into an appropriate command queue defined by COMMAND_IDS. Each developer can specify a set of COMMAND_IDs for their game. For example, all GAME_STOPPED messages are automatically queued into its own queue. All these queues are made available to the LGE either through polling, or by installing callback functions that are automatically triggered when a message arrives in a particular queue. This functionality frees up the developer from worrying about the communication issues and to focus on game logic as dictated by the various commands sent or received.

A reference implementation of the middleware has been implemented using the Microsoft .NET platform. The libraries thus built are thread-safe and special care has been taken to ensure that no native-only .NET APIs has been used to ensure portability to alternative platforms like Java. In addition, the platform only uses generically available coordination primitive like threads and semaphores.

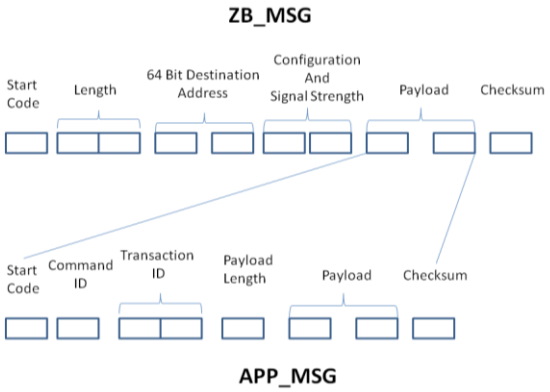

Figure 4. The structure of ZB_MSG and APP_MSG for one commercial ZigBee chipset

D. Client Side Processing

Fig. 5 shows the primary structure of how the processing is done on each client node. The grayed out

blocks are provided through a natively compiled library for each microcontroller. The developer modifies that primary command interpreter loop that consists of checking for inputs and outputs and other housekeeping work followed by a command dispatch depending on messages received. Since most small microcontrollers being used for wearable and tangible devices do not support multi-threading, a single main loop is used for all processing. As Fig. 5 shows, The ZB_MSG Receiver extracts any IEEE 802.15.14 packets received on the ZigBee radio and constructs a ZB_MSG. As mentioned earlier, this module does not assume the presence of multi-threading in the embedded operating system and hence is designed to be non-blocking. The ZB_MSG is forwarded to the Dispatcher. The APP_MSG is then made available on a queue for the Command Interpreter loop. The developer uses appropriate APIs to retrieve each APP_MSG and to take actions depending on the message received. The command loop can also send APP_MSGes using the provided API.

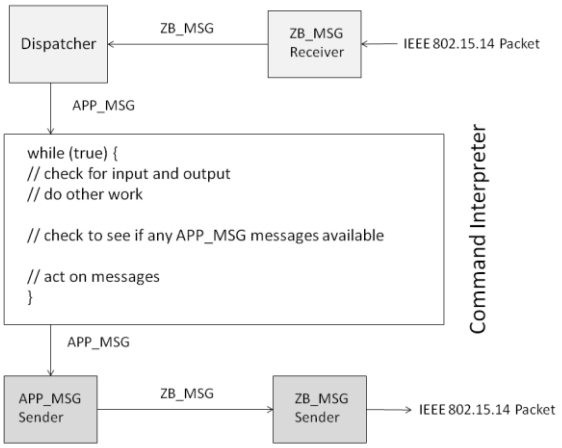

Figure 5. Structure of processing on the client side where the grayed components are provide with a microcontroller-specific library and the command interpreter is modified by the developer to implement the game logic

As Fig. 5 shows, each APP_MSG is forwarded to the APP_MSG sender that converts the message a ZB_MSG which is forwarded to the ZB_MSG sender that in turns converts it into an IEEE 802.15.14 packet for the ZigBee network.

The client-side libraries have been implemented in the C language. Special care has been taken to only use native C programming constructs in embedded implementations to ensure portability to most microcontrollers without requiring additional software libraries.

IV. Example Implementations

The toolkit and middleware described in this paper were evaluated by using it to re-implement a variety of pervasive games including those originally built using Prête-à-apprendre+ [20]. Prête-à-apprendre+ is a wearable ubiquitous platform designed for building pervasive tag learning games. In a learning tag game, children are given a topic to learn and based on the topic, they prepare three questions. Each child then wears a tag-shirt with their own questions on their shirt. As Fig. 6 shows, these three questions can be printed and stuffed into the three pockets shown in front of the shirt. An LED arrow is available next to each question indicating that the question has been remotely selected. As Fig. 6 shows, the shirt has two microcontrollers where one microcontroller communicates with the ZigBee radio module. The shirt also has a 3-axis accelerometer to detect movement. In addition, the shirt has a number of outputs including a buzzer and various other LEDs. The buzzer and the accelerometer are controlled by one microcontroller while the second microcontroller controls the LED's. The two microcontrollers are configured in a master-slave fashion through the I2C bus.

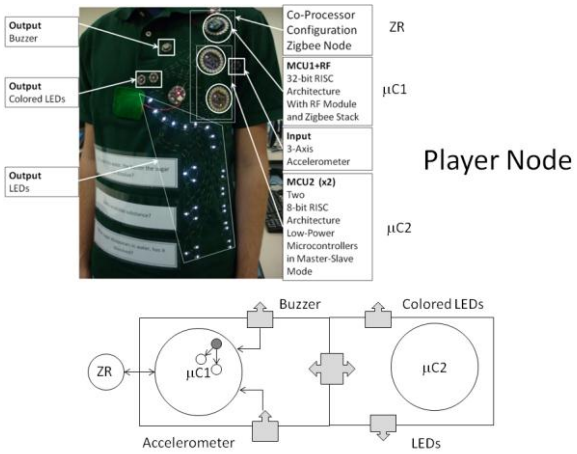

Figure 6. Player node showing a wearable Zigbee node where data from various sensors including a 3-axis accelerometer

Children play a game of tag using these shirts; the game starts by two children standing in front of each other. The game engine randomly selects a child and a question on that child's shirt and turns on the LED arrow next to their question by using the ZigBee network. Lighting up of an arrow on a shirt is a signal for the other child that the child with that lit shirt is the 'it' in the tag game.

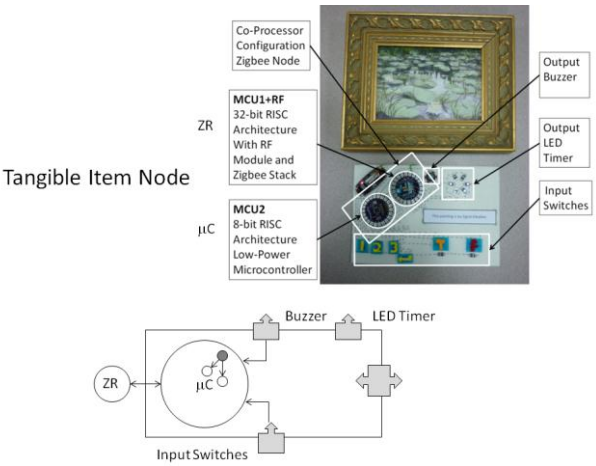

Figure 7. A tangible item Zigbee node next to an art object that allows children to enter their code upon finding the art object to indicate that they have found it.

To use the toolkit, it was assumed that there were two types of proxy objects in these games; a PAD and a SHIRT. Table I shows the APP_MSG structure (command ID's) for the SHIRT object. As Table 1 shows, for example, upon receiving QUESTION_CLEAR message, all the questions are cleared on the SHIRT node.

TABLE I. . COMMAND STRUCTURE OF APP-MSG FOR SHIRT OBJECT

COMMAND	Meaning
QUESTION_CLEAR	Clear all questions being asked on the shirt
QUESTION_ADD	Add a question to the shirt
QUESTION_SELECT	Select a question on the shirt
IO_STATE_GET	Get the state of an I/O on the shirt
IO_STATE_SET	Set the state of an I/O on the shirt
ROUND_START	Start a round of the game
ROUND_END	End a round of the game
ROUND_ABORT	Abort a round
ROUND_RESULT_GET	Retrieve the result of the round
STATE_SET	Change the state of the shirt (e.g., IDLE, IN-PLAY etc.)
STATE_GET	Get the current state of the shirt

The command structure for the PAD object was almost identical to that of the SHIRT object with some slight modifications. For example, the ROUND_RESULT_GET command, rather than returning just the question and its answer, also returned the code entered by the team that answered the question as shown in Fig. 7.

In addition to the command structures, call-backs were defined for each of the messages. For example, a QUESTION_SELECT_CALL_BACK function waited for an ACK message from the SHIRT object to ensure that the question had been selected on the shirt before proceeding on to the next stage of the game.

After the message structures were defined, the game server and the front-end were implemented using the supplied Middleware with Visual Basic .NET. The embedded code used the client-side C libraries to implement the required functionality on each of the client nodes; SHIRT and PAD.

Fig. 8 shows the first stage of the game where the students, teams and the questions for each student are entered. Fig. 9 shows the second stage of the game where a teacher or a parent can review the questions for each child. Fig. 10 shows that once the questions are finalized, the parent or the teacher can print the questions in a large font so that the labels can be printed and stuffed in front of the shirts of each of the respective children playing the tag game; each shirt has three questions on it.

Fig. 11 shows a screenshot of how a teacher or a parent can select a question on any one of the children's shirts. Once a question is selected, the system, based on the unique ID of the ZigBee node on the child's shirt (e.g., 00-13-A2-00-40-38-C9-91 in Fig. 11) as specified in the corresponding proxy object sends a message QUESTION_SELECT to the specific shirt to start the game and make the children wearing the shirt, the 'it.' The shirt receiving the message responds by lighting up the arrow LEDs. This starts a game-cycle where children in the other team chase and try to tag the child who has become the 'it' by their question lighting up. When a child tags the 'it' child, a message is sent from the shirt to indicate that the round has ended with the corresponding answer. It is up to the game engine to now determine whether the question was answered correctly or not. The result is also stored locally on the microcontroller and the game engine can query that result by using the ROUND_RESULT_GET message. At this point, the game engine also sends a ROUND_END message to all shirts to indicate that a round has ended.

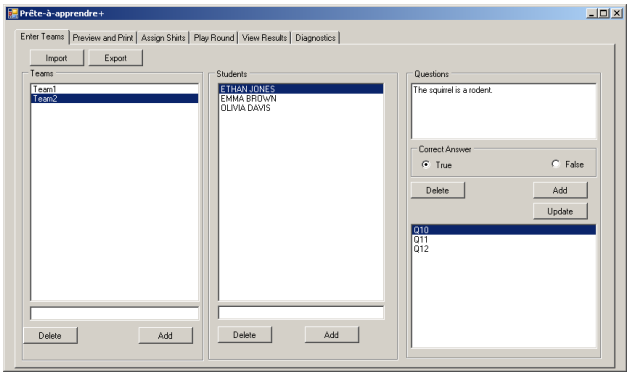

Figure 8. Setting up the team, students and questions written by each member of the team where the hardware node information has already been entered for each child's shirt

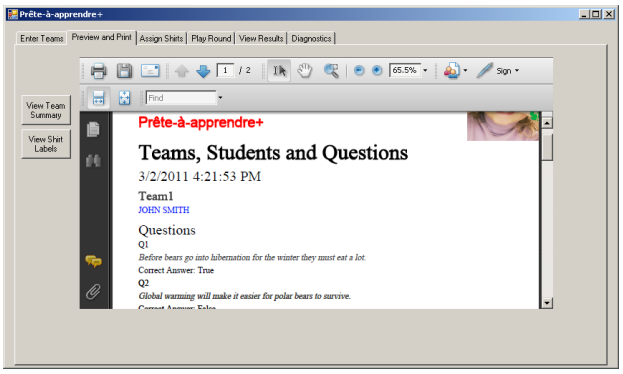

Figure 9. Showing student teams, questions written by each team and the correct answers for each question

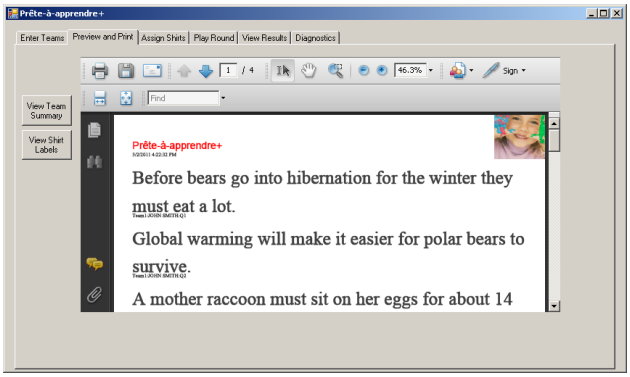

Figure 10. Application allowing teacher/parent to print the labels to be affixed to each child's shirt

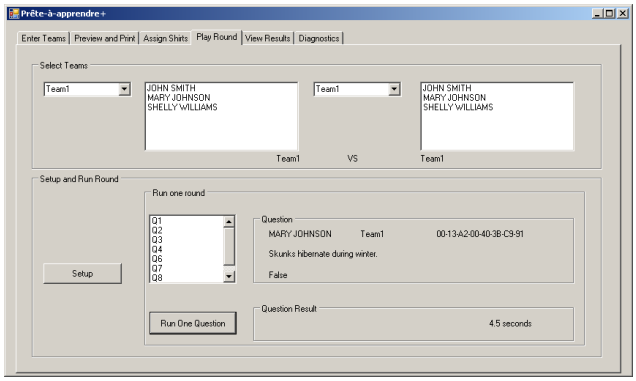

Figure 11. Teacher assigning a question and running a round of the game while the screen shows the question being answered and the amount of time that has lapsed since the question was posed

Fig. 12 shows the results of running one game session showing which questions were answered correctly or incorrectly by each child.

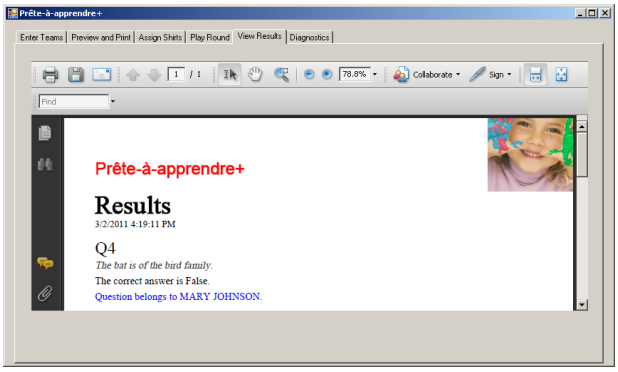

Figure 12. The system showing results at the end of one game session

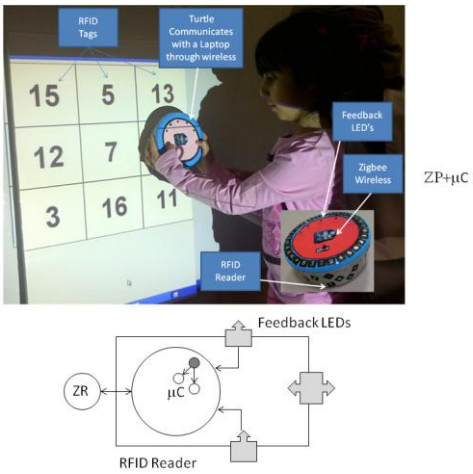

Figure 13. Turtle arithmetic addition game using ZigBee enabled node where children need to find the next number in the projected sequence by using the turtle or add or subtract two numbers given in the grid and indicate the sum or difference by using the turtle.

Fig. 13 shows an example of another pervasive game where the children learning number sequencing and addition are asked to use a ZigBee-enabled 'turtle' to select the right sequence of numbers or to indicate the sum or difference of two numbers shown on a projected grid of numbers. The game logic runs on a laptop that is connected to an overhead projector to project the number grid. Behind each cell on the grid is a unique RFID tag, and an RFID reader embedded in the turtle reads the tag as child makes a selection of the answer. The answer thus selected is transmitted to the game server using ZigBee via an appropriate message. In addition to the APP_MSGes defined earlier, this game had simple messages like SELECTED_CELL where the PayLoad of the message indicates the cell number indicating which cell of the grid was selected by the child.

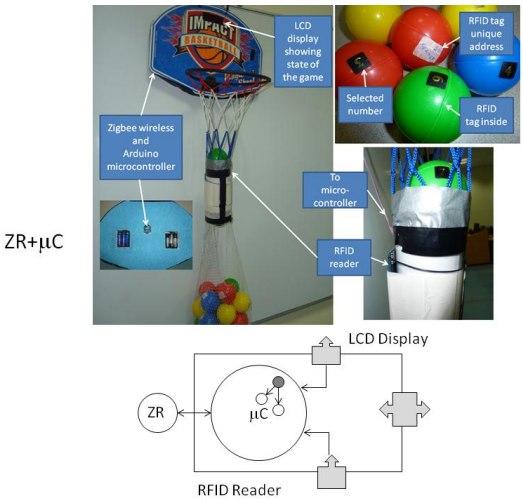

Figure 14. The learning game for number sequencing, addition and subtraction where children have to throw the ball with the right answer through the basketball hoop

Fig. 14 shows another example of a pervasive learning game where the game of basketball is used to teach sequencing of numbers or simple addition and subtraction. In this game, the basketball hoop is instrumented with a ZigBee wireless node and each ball contains a unique RFID tag. When a child throws a ball through the hoop, the RFID reader detects the unique RFID identifier of the ball and sends an appropriate message to the game server. Several different games are supported by the game engine. For example, two children throw one ball each through the hoop to make up an addition problem and a third child throws a third ball indicating the correct sum or subtraction result. The APP_MSGes for this game consist of BALL_IN_HOOP with the PayLoad including the 10 digit RFID number of the RFID tag in the ball. The setup and the game logic engine contain a map of the RFID tag numbers to actual numbers written on each ball.

Finally, Fig. 15 shows an example of a learning game that combines a ZigBee-enabled wearable shirt with a game of 'jacks.' In this game, the child is wearing a shirt that receives a question from a teacher using the ZigBee wireless network; the question is displayed on a rolling LED badge on their shirt. The child has a limited amount of time to take some stones with embedded RFID tags, throw them up like the game of jacks and pick the stone with the right answer (true or false). Upon doing so, the child now has to throw the right stone into a jar that

contains an RFID tag reader which reads the tag and sends the answer back to the system. An interesting twist in the game is that the jar is actually held by another child who is running away from the first child; the first child has to catch them and then throw their stone into the jar. In this case, shirt and the jar have different messages. Example APP_MSGes for the jar include RFID_DETECTED (the command) where the Payload is the actual 10 digit RFID code. Similarly, for the shirt, one message sent to the shirt is the actual question, QUESTION where the payload is the text of the actual question to be displayed on the LED rolling badge on the sleeve.

Figure 15. Zigbee-enabled game of Jacks combined with a wearable ZigBee-enabled shirt where children need to answer a question that appears on their shirt by throwing the right 'stone' into the jar

V. Conclusion

This paper has presented a light-weight messaging architecture and a toolkit for developing a host of pervasive games that use ZigBee. The toolkit hides the complexities of dealing with the low-level networking protocol, or a vendor-specific variation of the protocol by providing software proxy objects to the developer who can write their game logic by sending and receiving domain-specific messages to software proxy objects. The toolkit has been used to successfully implement a variety of pervasive and tangible learning games. One current limitation of the toolkit is that it uses a .NET framework making it unsuitable for Linux or Android devices. An effort is underway to port the Middleware to Java and Android which will enable developers to build applications that can use the mobile phones or the tablets as the game server and hence increasing the portability of such learning games.

Acknowledgement

This work was supported in part by a grant from the IBM Corporation and through a faculty research grant from the American University of Sharjah, UAE. I. A. Zualkernan is with the American University of Sharjah, UAE (e-mail: izualkernan@aus.edu).

References

[1] D. M. Han and J. H. Lim, "Smart home energy management system using ieee 802.15. 4 and zigbee," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 1403–1410, 2010.

[2] H. Cho, J. Kim, and Y. Baek, "Large-scale active rfid system utilizing zigbee networks," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 379–385, 2011.

[3] O. Mirabella and M. Brischetto, "A hybrid wired/wireless networking infrastructure for greenhouse management," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 2, pp. 398–407, 2011.

[4] C. H. See, K. V. Horoshenkov, R. A. Abd-Alhameed, Y. F. Hu, and S. J. Tait, "A low power wireless sensor network for gully pot monitoring in urban catchments," *Sensors Journal, IEEE*, vol. 12, no. 5, pp. 1545–1553, 2012.

[5] K. Malhi, S. C. Mukhopadhyay, J. Schnepper, M. Haefke, and H. Ewald, "A zigbee-based wearable physiological parameters monitoring system," *Sensors Journal, IEEE*, vol. 12, no. 3, pp. 423–430, 2012.

[6] O. A. Postolache, P. S. Girao, J. Mendes, E. C. Pinheiro, and G. Postolache, "Physiological parameters measurement based on wheelchair embedded sensors and advanced signal processing," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 10, pp. 2564–2574, 2010.

[7] D. Stupar, J. Bajic, L. Manojlovic, M. Slankamenac, A. Joza, and M. Zivanov, "A wearable low-cost system for human joint movements monitoring based on fiber-optic curvature sensor," *Sensors Journal, IEEE*, vol. 12, no. 12, 2012.

[8] W. T. Sung, J. H. Chen, and K. W. Chang, "Mobile physiological measurement platform with cloud and analysis functions implemented via ipso," *Sensors Journal, IEEE*, vol. 14, no. 1, 2014.

[9] H. Cho, H. Jang, and Y. Baek, "Practical localization system for consumer devices using zigbee networks," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1562–1569, 2010.

[10] K. Salen and E. Zimmerman, *Games of Play: Game Design Fundamentals*. The MIT Press, Cambridge., 2004.

[11] M. Papastergiou, "Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation," *Computers & Education*, vol. 52, no. 1, pp. 1–12, January 2009.

[12] H. Tüzün, M. Ylmaz-Soylu, T. Karakus, Y. Inal, and G. Kzlkaya, "The effects of computer games on primary school students' achievement and motivation in geography learning," *Computers & Education*, vol. 52, no. 1, pp. 68–77, 2009.

[13] I. Soute, P. Markopoulos, and R. Magielse, "Head up games: Combining the best of both worlds by merging traditional and digital play," *Personal and Ubiquitous Computing*, vol. 14, no. 5, pp. 1617–4917, July 2010.

[14] S. Akkerman, W. Admiraal, and J. Huizenga, "Storification in history education: A mobile game in and about medieval amsterdam," *Computers & Education*, vol. 52, no. 2, pp. 449–459, 2009.

[15] J. Verhaegh, I. Soute, A. Kessels, and P. Markopoulos, "On the design of camelot, an outdoor game for children," in *Proc. 2006 Conference on Interaction Design and Children*. ACM, 2006, pp. 9–16.

[16] R. Magielse and P. Markopoulos, "Heartbeat: An outdoor pervasive game for children," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 2181–2184.

[17] T. Bekker, J. Sturm, R. Wesselink, B. Groenendaal, and B. Eggen, "Interactive play objects and the effects of open-ended play on social interaction and fun," in *Proc. the 2008 International Conference on Advances in Computer Entertainment Technology*. ACM, 2008, pp. 389–392.

[18] I. Arroyo, I. Zualkernan, and B. Woolf, "Hoodies and barrels: Using a hide-and-seek ubiquitous game to teach mathematics," in *Proc. the ICALT 2011.*, 2011.

[19] Y. G. Ha, "Dynamic integration of zigbee home networks into home gateways using osgi service registry," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 470–476, 2009.
[20] I. A. Zualkernan, N. Al-Khunaizi, S. Najar, and N. Nour, "Prête-à-apprendre+: Towards ubiquitous wearable learning," in *Proc. 10th IEEE International Conference on Advanced Learning Technologies, Sousse, Tunisi*, July 2010, pp. 740–741.

Imran A. Zualkernan Dr. Zualkernan holds a B.S. (high distinction) and a Ph.D. in Computer Science from University of Minnesota, Minneapolis, USA. Dr. Zualkernan has taught at the University of Minnesota, Pennsylvania State University and the American University of Sharjah. He did post-doctoral work at the Carlson School of Management and at the Center for Research in Learning Perception and Cognition. He has been an adjunct faculty member at the Arizona State University and a visiting professor at the University of Massachusetts, Amherst. Dr. Zualkernan was a Principal Design Engineer for a high-end robotics company in Chanhassen, Minnesota. Dr. Zualkernan later served as the founding chief executive officer of a public-limited software services company and the chief technology officer of an e-Learning technologies company. His areas of interest are internet of things, ubiquitous computing, IT management and advanced learning technologies. He has published over 150 articles in refereed journals, conferences and workshops. He has been on the executive board of the IEEE technical committee on Learning Technology. He is a member of IEEE, Tau Beta Pi and Golden Key National Honor Society.