Design and Implementation of ZigBee Based Infrastructure for Smart Environments

Azza Kamal Nabih Hanafi
Senior R&D Engineer-SECC
azkamal@itida.gov.eg

Mahmoud M. Abdallah
Senior R&D Engineer-SECC
mmabdallah@itida.gov.eg

© Copyright Software Engineering Competence Center 2013
Abstract

Wireless communication is increasingly becoming the effective technology for connecting distributed appliances in smart environments. ZigBee is characterized by its low power consumption, low cost, and practical communication range (typically, 50m), which makes it very appropriate for most smart environment applications. Moreover, ZigBee operates in the 2.4 GHz ISM band, which is publically (i.e., free) available band worldwide.

This technical report explains hardware and software design as well as the implementation details of a simplified smart environment that contains environmental nodes, light nodes, and an RFID. Environmental and light nodes are connected to the system using ZigBee technology. This simplified system that contains the common nodes existing in any environment was used as the base for an energy-aware smart home application that is developed as part of the RECOCAPE project. A demonstration for the system is configured and tested within SECC R&D lab and it showed good results.

**Keywords:** Wireless Communication, ZigBee, Smart Environment, User Identification, RFID.
Table of Contents

1. Introduction 4
2. Architecture of a Simplified Smart Environment 5
3. Packet Format 7
4. 3.1. Device Identification Packet Structure 7
5. 3.2. Device Periodic Data Packet Structure 8
6. 3.2.1. Environmental Node 9
7. 3.2.2. Light Node 9
8. 3.3. Commands Packet Structure 9
9. 4. Coordinator Node Design 10
10. 5. Environmental Node Design 12
11. 5.1. Hardware Design 12
12. 5.2. Software Design 14
13. 6. Light Node Design 16
14. 6.1. Hardware Design 16
15. 6.2. Software Design 17
16. 7. RFID User Identification Node Design 18
17. 7.1. Hardware Design 18
18. 7.2. Software Design 20
19. 8. Summary 21
20. 9. References 22
21. 10. Abbreviations 23
1. Introduction

Smart environments are the user environments that are equipped with variety of devices, sensors, and actuators that cooperate to provide the user with convenience, safety, security, and comfort [1]. One of the main enablers for smart environments is wireless technology [2-12]. Learning how to design smart environments using wireless technology is important. This technical report explains how to use ZigBee technology to build an infrastructure that wirelessly integrates nodes to be used for the development of smart environment applications. The report discusses the implementation details of a simplified smart environment that contains environmental nodes, light nodes, and an RFID. Environmental and light nodes are connected to the system using ZigBee technology. The report presents the technical details of the infrastructure including the set of packets communicated between the nodes, the hardware and software design of the individual nodes.

The infrastructure enables the user to:

- Remotely monitor and control the environment appliances at run time.
- Dynamically identify appliances to the system.
- Identify users and apply their preferences in context based scenarios.

An implementation of an energy-aware smart home application that utilized this infrastructure is done as part of the RECOCAPE project [13]. The energy-aware smart home application extended the infrastructure to integrate refrigerator, AC, and multimedia nodes. A demonstration for the system is configured and tested within SECC R&D lab and it showed good results.

The rest of the technical report is organized as follows: Section 2 presents the conceptual design of a simplified smart environment. Section 3 discusses the packets formats used to fulfill the needed communication. Section 4 to Section 7 details the design of the different nodes constituting the system. Section 8 gives a summary on the technical report. The references and abbreviations are listed in Sections 9 and 10 respectively.
2. Architecture of a Simplified Smart Environment

The proposed ZigBee based hardware infrastructure consists mainly from two parts: core hardware components used to realize the physical layer of the infrastructure and a set of defined packets that are communicated over the network to provide a standard way of communication. The hardware infrastructure at the end will constitute the lower layer providing all the needed information to higher level application scenarios. Application scenarios can be easily developed as it will manipulate the data provided in standard formats. Also extendibility will be easy as the infrastructure suggests an easy integration scenario on the level of the hardware components and the way of communicating system data.

The core hardware of the infrastructure in a brief consists of a ZigBee network; with ZigBee nodes are attached to different sensors and appliances needed to be integrated within the system. The ZigBee coordinator is the focal point of communication with the ZigBee network; it receives all status and sends required commands to the ZigBee network. This ZigBee coordinator is connected through serial interface to a personal computer. A central program is running on the PC and it receives all the ZigBee network data from the coordinator and sends commands to specific devices in the ZigBee network as needed by the application logic.

The suggested infrastructure integrates also an RFID user identification module that sends user identification data to the system central program. Hence, the system has access to environment, devices and users' data and it can provide more advanced features through manipulating all available information. The RFID module has direct interface to the PC through Ethernet connection. It is done that way and not integrated as part of the ZigBee network so that it will be portable and reusable module even if another wireless communication protocol is used to realize the system network.

For the communication that is needed to be carried out over the network, three communication scenarios are suggested to accomplish any high level application goals. The infrastructure suggests a standard identification packet, so that any node will send its identification which will enable easy integration of any additional nodes in the system. The high level application logic will be able to specify the type, supported features included in any node through proper parsing of its identification. In addition, standard packets are suggested to communicate status date of the nodes. This data packet is standardized for each device type, with different device types report different data but the data can be standardized on the level of the device category. Finally, a command packet is needed to be able to send commands to the individual nodes; again this is dependent on the device type.
By having this hardware components and defining the needed communication, a hardware infrastructure is built and can be extended in a systematic way. In addition higher level applications can be easily developed and extended as they mainly depend on a standard infrastructure and communication scenarios.

Figure 1 shows the system architecture of the proposed ZigBee based hardware infrastructure. The depicted system includes light and environmental nodes which form main components nearly in any environment. In addition; they demonstrate the different system capabilities of sensing and actuating within the environment. Similarly, ZigBee nodes can be attached to any other sensors/appliances, programmed to send the appropriate data in the standard packets, and get involved into the system rapidly and easily.

Figure 1. System architecture

Experimental implementation is done based on this proposed hardware infrastructure to verify its effectiveness. JN5148-EK010 ZigBee PRO Evaluation Kit from Jennic has been used. It provides total five sensor nodes per each kit [14].

Section 4 to Section 7 details the hardware and software design of the different nodes existing in the example architecture depicted in Figure 1 so that they can be used as a reference to similarly implement other system device nodes. Noting that, it is a single coordinator for the ZigBee network and the extension is done on the level of adding more device nodes.
Communication between system nodes is done for three purposes. First purpose is to identify the device upon the node startup so that the central program running on the PC can know the capabilities of this node. This communication is done by sending ZigBee packets from the device node to the coordinator through ZigBee protocol. Then these identifications are sent from the coordinator to the central program through the serial connectivity. Coordinator also sends its identification packet to the central program through the serial port. Second purpose is to send periodic device status; with individual devices send their status periodically to the coordinator node and finally to the central program. Third purpose is to send commands to the individual devices. This communication is initiated from the central program where appropriate actions are decided to be executed on individual device nodes. These actions are decided based on automatic manipulation of the system status or manually triggered by the environment owners. These actions are then sent to the coordinator which will pass the message through the ZigBee network to the required destination. Following is a detailed description for the messages communicated for each messaging type:

### 3.1. Device Identification Packet Structure

A standard device identification message is defined for all devices, so that the central program can parse the message appropriately and recognize the devices' capabilities. This message is realized through the standard packet format shown in Figure 2.

<table>
<thead>
<tr>
<th>Type</th>
<th>MAC</th>
<th>DT</th>
<th>CMD</th>
<th>CMDR</th>
<th>M</th>
<th>MR</th>
<th>CMDDEF</th>
</tr>
</thead>
</table>

The elements of the identification structure are explained in Table 5 in appendix A.

For the experimental implementation of the architecture in Figure 1, the Type field is set to ‘i’ to indicate that it is an identification packet.

The CMD field has a predefined size MAX_COMMANDS_PER_NODE which is set to ten. Commands can be sent to light node but not to the environmental node. Network coordinator is not expected to execute any commands by itself; it only passes the commands to their required destinations. The commands are assigned to the elements of the CMD field, NO_COMMAND which is defined to be
0 is set to the rest of entries. "set light level" command of the light node defines its range to be from zero to the maximum number of the lamps connected to the light node.

The M field has a predefined size MAX_MEASURES_PER_NODE which is set to ten. Environmental reports measurements on temperature, light, humidity, presence, and fire. RFID node reports measurements of the RFID readings, light nodes don’t report any measurements. Network coordinator is not expected to report any measures; it only passes the measurements reported by the other nodes to the central program running on the PC. The measurements are assigned to the elements of the M field; NO_MEASURE which is defined to be 0 is set to the rest of entries.

MR field determines the range within which the reported measure will be considered acceptable and valid. This information can be later used to infer specific device failure in case of reporting invalid measures by the node. For example, the environmental node should report values from zero to six for the light intensity level within the room in which the node is installed. Reporting values out of this range may indicate a failure in the light sensor.

### 3.2. Device Periodic Data Packet Structure

Not all nodes are expected to send status data. Nodes included in the environment for actuation purposes don’t report periodic status data. For example, environmental node depicted in our simple scenario shown in Figure 1 sends periodic data about its status, while light node doesn’t. The device periodic data differs as per the associated device type, with each device sends the measures determined while its identification. Figure 3 depicts the scenario for sending the periodic device data.

These periodic device data can be used to validate that nodes are live. Receiving regular device data by the central program will infer that the node is active and is performing its expected functionality. For light nodes, a dummy periodic packet will be sent. It doesn’t report actual measures in this case, but is only used to verify node existence within the network.

The following subsections presents the data packets designed for the environmental and light nodes used in our example scenario. Idea can be generalized to any other node types.
3.2.1. Environmental Node

Environmental nodes send packets periodically to report the environment status. An exceptional packet is sent whenever reporting data about presence or fire. Figure 4 shows the packet format used to report the environmental data.

<table>
<thead>
<tr>
<th>Type</th>
<th>MAC</th>
<th>DT</th>
<th>PR</th>
<th>T</th>
<th>H</th>
<th>LI</th>
<th>P</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT: Device Type</td>
<td>T: Temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H: Humidity</td>
<td>LI: LightIntensity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P: Presence</td>
<td>F: Fire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The elements of the environmental data structure are explained in Table 9 in appendix A.

3.2.2. Light Node

Light node doesn't report actual measures but it will send a periodic dummy packet that will be used only to verify its existence in the system and that it is live. Figure 5 shows the packet format used for this dummy packet.

<table>
<thead>
<tr>
<th>Type</th>
<th>MAC</th>
<th>DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT: Device Type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Light periodic dummy packet format

3.3. Commands Packet Structure

They are the set of the supported actions that can be executed by the devices connected to the individual ZigBee nodes. They are determined while device identification procedure. Hence, commands are sent in the appropriate format...
pre-specified by the attached devices. For example, the command to the light node is only a number that determines the number of the required lamps to be turned on. This number ranges from zero which means turning off all the lamps connected to the node to the maximum number of the lamps connected which means to turn on all the lamps.

The procedure to pass the required command initially from the central program and finally to the target node is that the command is sent from the central program running on the PC to the coordinator with the MAC address of the target node concatenated. Then, the coordinator passes the command to the required node identified by its MAC and sends "Done" message to the PC informing it that the command is sent to the appropriate destination. Figure 6 depicts the scenario for sending the device commands.

![Figure 6. Command processing](image)

### 4. Coordinator Node Design

The coordinator board is connected to the PC; where the environment central program will be running, through the serial interface. Figure 7 shows a simple figure to visualize this connectivity. The UART settings are: 115200 baud rate, 8 data bits, no parity, one stop bit, and none flow control.

![Figure 7. Coordinator node](image)
Figure 8 shows the tasks used to realize the required functionality of the coordinator node.

![Coordinator node design diagram]

The following is the detailed design of the major tasks depicted in Figure 8:

**APP_taskScanButtons**
This task runs periodically to scan the buttons of the controller node. Then, actions can be taken based on the current window of the application and the button pressed.

**APP_taskControllerNode**
This task implements the state machine of the controller node. It sends the coordinator identification upon ZigBee network creation.

**UART_vIsr0**
It is an ISR (Interrupt service routine) that is executed whenever a character is sent or received on UART0 of the ZigBee node.

- Receive characters sent to UART0 of the ZigBee node and add them to a queue.
- Upon receiving 'r' which is used as a pre-defined terminator for the commands sent from the PC to the coordinator (which will be later passed to their final destination node), the ISR sends a message through APP_msgCommand message queue which will activate APP_taskRecvAndProcessCommand.
**APP_taskReceiveAndProcessCommand**

- Wait to receive a message from APP_msgCommand message queue.
  - Receiving a message means that a particular command has been sent and is then required to be sent to its appropriate destination node.
- Read the first eight bytes from the UART queue; these first eight bytes are the MAC address of the required destination node, as previously explained in Figure 6.
- Read the command from the UART queue; it is constituted of the remaining packets in the UART queue.
- Send the command to the final destination node.
- Send "Done" to the central program running on the PC informing it that the command is sent to the specified destination.

**APP_taskReceiveDeviceData**

This task receives the identification and data packets from the ZigBee nodes; through ZigBee packets, then sends them to the PC through UART0 after parsing them appropriately.

- Receive ZigBee packets sent from existing system nodes through APP_msgDeviceData message queue.
- Check whether it is an identification or data packet.
  - If it was an identification packet, parse the data in the packet according to the identification data structure. Identification data structure is defined in section 3.1.
  - If it is a data packet, then parse the data according to the device type. Parsing is done based on the data structures defined in section 3.2.

5. Environmental Node Design

This section details the hardware and software design of an environmental node. Environmental node reports measurements of the different environmental parameters like ambient temperature, light, humidity, light intensity, presence, and fire.

5.1. Hardware Design

ZigBee nodes of the JN5148-EK010 ZigBee PRO Evaluation Kit have on board temperature, humidity, and light sensors that will be used to report the data of the ambient temperature, the humidity, and the light intensity level. For the
presence and fire, external sensors are connected to the node through the expansion port on the node, specifically through the digital I/O pins [15]. PIR sensor is used to detect motion and hence infer the presence information [16]. To predict the fire a PARALLAX CO2 gas sensor module is used [17]. As the DIO pins of the ZigBee nodes are configured to enable pull up, an inverter chip is used so that presence and fire status can be appropriately measured. For this purpose SN74HC04NSR inverter chip is used [18]. To improve the presence/no presence decision accuracy, two PIR sensors are connected to the ZigBee node so that there will be two sources based on which the occupancy status of the place can be determined. Table 1 and Table 2 details the hardware connections of the presence and fire sensors to the ZigBee node.

Table 1. Connections of the inverter chip

<table>
<thead>
<tr>
<th>Inverter pin</th>
<th>Sensor pin</th>
<th>ZigBee node pin</th>
<th>External connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>GND</td>
<td>GND</td>
<td>GND of an external power source that generates +5V output</td>
</tr>
<tr>
<td>VCC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1A</td>
<td>OUT pin of PIR1</td>
<td>DIO5</td>
<td></td>
</tr>
<tr>
<td>1Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>OUT pin of PIR2</td>
<td>DIO18</td>
<td></td>
</tr>
<tr>
<td>2Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3A</td>
<td>ALR pin of the CO2 gas sensor</td>
<td>DIO1</td>
<td></td>
</tr>
<tr>
<td>3Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2. Connections of PIR and fire sensors

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Sensor pin</th>
<th>Inverter pin</th>
<th>ZigBee node pin</th>
<th>External connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1</td>
<td>GND</td>
<td>GND</td>
<td>GND</td>
<td>GND of an external power source that generates +5V output</td>
</tr>
<tr>
<td></td>
<td>VCC</td>
<td></td>
<td></td>
<td>+5V from external power source</td>
</tr>
<tr>
<td></td>
<td>OUT 1A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIR2</td>
<td>GND</td>
<td>GND</td>
<td>GND</td>
<td>GND of an external power source that generates +5V output</td>
</tr>
<tr>
<td></td>
<td>VCC</td>
<td></td>
<td></td>
<td>+5V from external power source</td>
</tr>
<tr>
<td></td>
<td>OUT 2A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fire</td>
<td>GND</td>
<td>GND</td>
<td>GND</td>
<td>GND of an external power source that generates +12V output</td>
</tr>
<tr>
<td></td>
<td>VIN</td>
<td></td>
<td></td>
<td>+12V from external power source</td>
</tr>
<tr>
<td></td>
<td>CNTL</td>
<td></td>
<td></td>
<td>+12V from external power source</td>
</tr>
<tr>
<td></td>
<td>ALR 3A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2. Software Design

Figure 10 shows the tasks used to realize the required functionality of the environmental node.
The following is the detailed design of the major tasks depicted in Figure 10:

**APP_taskSendDeviceIdentification**
This task wirelessly sends the node identification packet to the coordinator through ZigBee network.
- Fill in the data of the identification data structure according to the specific node identification.
- Send the structure over the ZigBee network to the coordinator node.
- Start APP_tmrConfirmDeviceIdentification timer which will enable APP_taskConfirmDeviceIdentification task to run.

**APP_taskConfirmsDeviceIdentification**
This task confirms the reception of the device identification packet by the coordinator. This is in order not to lose this important information as it draw the way to control the device later within the system.
- Check the reception of a receive confirmation message from the coordinator node.
- If a confirmation is received; meaning that the identification packet is received successfully by the coordinator, then
  - Start APP_taskSampleSensors and APP_taskCheckUrgentEnvironmentalData tasks of the environmental node.
- If a confirmation is not received, then
  - Restart APP_taskSendDeviceIdentification task again to send the confirmation packet.
**APP_taskSampleSensors**
This task periodically sends the data of the ambient temperature, the humidity, and the light intensity level to the coordinator node.

- Sample the data of the temperature, humidity, and light from the on board sensors.
- Fill in the data structure of the environmental measures with the appropriate data. Note that the cType field of the structure should be 'd'; meaning it is a data packet.
- Send the packet wirelessly over the ZigBee network to the coordinator.

**APP_taskCheckUrgentEnvironmentalData**
This task sends an exceptional data packet whenever the state of presence or fire is changed.

- Check the DIO pins where the presence sensor is connected.
- If the system reports a change to have an existence, then send an exceptional data packet to the coordinator.
- If the previous system state was there is a presence, but the two PIRs report that there is no motion, then
  - Wait for one minute before taking a final decision that there is no presence. Upon taking a final decision an exceptional packet will be sent to the coordinator reporting the new system state.
- If a fire is detected while there was no fire, or the system reports no fire while there was previously fire, then
  - Send an exceptional data packet to the coordinator reporting the new system state.

6. Light Node Design

This section details the hardware and software design of a light node. Light nodes are used to control lights of the different places in the environment.

6.1. Hardware Design
A ZigBee node is connected to a relay board where a number of lamps are attached. The connection between the ZigBee node and the relays are done through the digital I/O pins of the ZigBee node as detailed in Table 3.
Table 3. Connections of the relay used to drive a lamp

<table>
<thead>
<tr>
<th>Relay board pin</th>
<th>ZigBee node pin</th>
<th>External connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>GND</td>
<td>+12V from external power source</td>
</tr>
<tr>
<td>+12V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTROL</td>
<td>DIO0</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td></td>
<td>Lamp</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>AC line</td>
</tr>
</tbody>
</table>

6.2. Software Design

Figure 12 shows the tasks used to realize the required functionality of the light node.
The following is the detailed design of the major tasks depicted in Figure 12:

**APP_taskSendDeviceIdentification**
It is same functionality as that reported by the environmental node but it fills the structure with its specific data related to the light node.

**APP_taskConfirmDeviceIdentification**
It is same functionality as that reported by the environmental node.

**APP_taskStopNodesReset**
This task runs periodically to restart watchdog timer of the node to prevent the node from continuous reset as it may remain idle for large periods of time. It also sends a periodic dummy packet so that system ensures that the node is live and is performing its functionality.

**APP_taskReceiveAndProcessCommand**
- Receive a command from the coordinator node through APP_msgDeviceControl message queue.
- Turn on the required number of lamps specified in the command. This is done by setting the output of the ZigBee node digital I/O pins where the relays which control the lamps are attached.

### 7. RFID User Identification Node Design

This section details the hardware and software design of an RFID node. RFID nodes are used to identify users in the different places within the environment.

#### 7.1. Hardware Design

This module consists of a PARALLAX RFID module [19] and an Arduino Ethernet board [20]. The RFID module outputs the RFID tags' readings through its serial interface. The Arduino board takes the RFID tags' readings received through its serial interface and sends them over the LAN connection so that they will be accessible over the smart home LAN. The central program running on the PC can then take these readings sent over the LAN and use them to identify the home inhabitants and apply the preferences according to the identified individuals. There are existing RFID modules that send the identification data directly over the Ethernet, but the implementation done utilizing the Arduino board and low cost RFID module, helps in providing a lower cost solution for this user identification module. This will be reflected on the overall cost of the smart home.
solution and will help in providing a cheap solution for the smart home realization.

![RFID module and Arduino board](image)

**Figure 13. RFID user identification module**

**Table 4. RFID reader connection to the Arduino board**

<table>
<thead>
<tr>
<th>Arduino board</th>
<th>RFID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>ENABLE</td>
<td>Module enable pin. Active LOW digital input. Bring this pin LOW to enable the RFID reader and activate the antenna.</td>
</tr>
<tr>
<td>0 (RX)</td>
<td>SOUT</td>
<td>Serial output to host. TTL-level interface, 2400 bps, 8 data bits, no parity, 1 stop bit.</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>5V</td>
<td>VCC</td>
<td></td>
</tr>
</tbody>
</table>

**Hardware components**

- **Parallax RFID module**
  - It is a low-cost solution to read passive RFID transponder tags up to 4 inches away.
  - It can be used in different applications like access control, and user identification.
  - The reader is activated by the passive 125 kHz RFID transponder tags and sends the value of the tag over the serial interface.

- **Arduino Ethernet**
  - Arduino is an open source hardware board with an 8-bit Atmel AVR.
  - It also includes an Ethernet interface to allow the board to be connected to LANs.
• Arduino has very comprehensive documentation and lots of open source projects which allow the developers to rabidly implement prototypes.

User interface
  ▪ Input
    • RFID card touch
  ▪ Output
    • Led is turned to green for 10 seconds to indicate disabling the input (disable sending more readings for the RFID tag) and then turned to red to indicate enabling the output

7.2. Software Design

![Diagram](image)

Figure 14. RFID module design

Software tools
  ▪ Arduino compiler: arduino-1.0.3
    • The Arduino integrated development environment (IDE) is a cross-platform development tool written in Java.
    • It incorporates a code editor, compiler and board programmer.
    • A program or code written for Arduino is called a sketch which is written in C or C++.
    • The sketch is composed of two functions to make a runnable cyclic executive program:
      o setup(): a function run once at the start of a program and is used to initialize settings
      o loop(): a function that is called repeatedly until closing the board
void setup()
- Initialize the identification message with the MAC filled with the Ethernet
  MAC address padded to be 8 bytes
- Initialize the data message
- Connect the Ethernet client with port 4444 to the server that hosts the
  central program
- Send the identification message

void loop()
- Receive the RFID reading through the serial interface and store it in the
  data message
- Send the data message over the LAN to the server that hosts the central
  program

8. Summary

The report provided the technical details of the design and implementation of a
simplified smart environment application that uses ZigBee technology to
wirelessly integrate nodes within the system. The presented infrastructure
contains environmental nodes, light nodes, and an RFID in addition to the main
control node of the ZigBee network; the coordinator node.
The structure of the different packets communicated within the system is
presented including the identification packets used to identify the system nodes,
the command packets used to send the commands to the nodes, and the data
packets used to periodically report sensor readings of the individual nodes. The
design and implementation details of the different nodes are explained.
It is worth noting that the presented infrastructure is extendable and can
support as many nodes as needed by the smart environment. The infrastructure
in this technical report was used as the base for an energy-aware smart home
application which is developed as part of the RECOCAPE project. The energy-
aware smart home application has extended the infrastructure to contain
refrigerator, AC, and multimedia nodes.
9. References


10. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>RS232</td>
<td>Recommended Standard 232</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>LED</td>
<td>Light-Emitting Diode</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver/Transmitter</td>
</tr>
<tr>
<td>AC</td>
<td>Air Conditioner</td>
</tr>
</tbody>
</table>
**Appendix A: Identification Packet Format**

Table 5. identification packet format

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>1</td>
<td>Determine whether it is an identification message or a periodic data packet.</td>
</tr>
<tr>
<td>MAC</td>
<td>8</td>
<td>The MAC address of the ZigBee node or the RFID module which is a unique identity for the device node. Note that, the MAC address of the RFID module is 6-bytes length but it makes padding to fit.</td>
</tr>
<tr>
<td>DeviceType</td>
<td>1</td>
<td>Determines whether the device is an environmental node, a light node, RFID module, the network coordinator, or any other device type. Table 6 shows an example for a list of supported devices in a home environment.</td>
</tr>
<tr>
<td>Command</td>
<td>Multiple*</td>
<td>Determine the list of supported operations that can be executed on each particular node. Table 7 shows an example for a list of supported commands in a home environment.</td>
</tr>
<tr>
<td>CommandRange</td>
<td>Multiple*</td>
<td>Determines the range of values supported by each command. For each command, a minimum and maximum value is specified to be set as the command parameter.</td>
</tr>
<tr>
<td>Measurement</td>
<td>Multiple*</td>
<td>Determine the list of supported measurements that will be reported by each particular node. Table 8 shows an example for a list of supported measurements in a home environment.</td>
</tr>
<tr>
<td>MeasurementRange</td>
<td>Multiple*</td>
<td>Determines the range of values supported by each measure.</td>
</tr>
<tr>
<td>CommandDefault</td>
<td>Multiple*</td>
<td>Determines the default values for the different settings of the nodes.</td>
</tr>
</tbody>
</table>

* Size is predetermined by the developer

Table 6. Example for a list of supported devices in a home environment

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZigBee Network Coordinator</td>
<td>'c'</td>
</tr>
<tr>
<td>Multimedia Player</td>
<td>'m'</td>
</tr>
<tr>
<td>Environmental Node</td>
<td>'e'</td>
</tr>
<tr>
<td>Refrigerator</td>
<td>'r'</td>
</tr>
<tr>
<td>Air Condition</td>
<td>'a'</td>
</tr>
<tr>
<td>Light Control Node</td>
<td>'l'</td>
</tr>
<tr>
<td>RFID Access Module</td>
<td>'i'</td>
</tr>
</tbody>
</table>
Table 7. Example for a list of supported commands in a home environment

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play multimedia player</td>
<td>1</td>
</tr>
<tr>
<td>Stop multimedia player</td>
<td>2</td>
</tr>
<tr>
<td>Adjust temperature in AC</td>
<td>3</td>
</tr>
<tr>
<td>Set grid level in AC</td>
<td>4</td>
</tr>
<tr>
<td>Set fan speed level in AC</td>
<td>5</td>
</tr>
<tr>
<td>Open fresh air in AC</td>
<td>6</td>
</tr>
<tr>
<td>Set light level</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 8. Example for a list of supported measurements in a home environment

<table>
<thead>
<tr>
<th>Measurement Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>1</td>
</tr>
<tr>
<td>LightIntensity</td>
<td>2</td>
</tr>
<tr>
<td>Humidity</td>
<td>3</td>
</tr>
<tr>
<td>Presence</td>
<td>4</td>
</tr>
<tr>
<td>Fire</td>
<td>5</td>
</tr>
<tr>
<td>RFID reading</td>
<td>6</td>
</tr>
</tbody>
</table>
### Table 9. Environmental measurement packet

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>1</td>
<td>Determine whether it is an identification message or a periodic data packet.</td>
</tr>
<tr>
<td><strong>MAC</strong></td>
<td>8</td>
<td>The MAC address of the ZigBee node or the RFID module which is a unique identity for the device node. Note that, the MAC address of the RFID module is 6-bytes length but it makes padding to fit.</td>
</tr>
<tr>
<td><strong>DeviceType</strong></td>
<td>1</td>
<td>Determines whether the device is an environmental node, a light node, RFID module, the network coordinator, or any other device type.</td>
</tr>
<tr>
<td><strong>Periodic</strong></td>
<td>1</td>
<td>Determines whether the environmental node reports a normal periodic data packet with the information about the ambient temperature, humidity, and light intensity level or it's an exceptional data packet with the information about presence and fire.</td>
</tr>
<tr>
<td><strong>Temperature</strong></td>
<td>1</td>
<td>Holds information of the current ambient temperature.</td>
</tr>
<tr>
<td><strong>Humidity</strong></td>
<td>1</td>
<td>Holds information of the current humidity in the environment.</td>
</tr>
<tr>
<td><strong>LightIntensity</strong></td>
<td>1</td>
<td>Holds information of the current light intensity level in the environment.</td>
</tr>
<tr>
<td><strong>Presence</strong></td>
<td>1</td>
<td>Specify whether the room; where the node is installed is occupied or not.</td>
</tr>
<tr>
<td><strong>Fire</strong></td>
<td>1</td>
<td>Specify whether a fire is detected in the room; where the node is installed or not.</td>
</tr>
</tbody>
</table>