ZIGBEE STANDARD IMPLEMENTATION FOR A WIRELESS TEMPERATURE SYSTEM

by

Ritchinder Ritchie Singh Samrai

Bachelor of Engineering, Ryerson University, 2007

A project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2015

Author's Declaration

I hereby declare that I am the sole author of this project. This is a true copy of the project, including any required final revisions.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my project may be made electronically available to the public.

Abstract

ZigBee Standard Implementation for a Wireless Temperature System

Ritchinder Ritchie Singh Samrai
Master of Engineering, Electrical and Computer Engineering
Ryerson University, 2015

This project is concerned with the application of the ZigBee communication standard for implementing a temperature measurement system. Due to ZigBee's low-power and low data rate features, it is ideal for analog sensor systems. Digi's ZigBee devices called XBee are used in this project. The XBee devices meet all the ZigBee standard. The XBee device has the advantage of being programmed with API firmware (application programming interface). XBee's API provides fast and reliable communication between the remote stations and the base station. The remote station has three different modules: power supply, temperature sensor and XBee device. The power supply is designed to output 3.3V. The temperature sensor is designed so that the output stays within the XBee's maximum analog input voltage range of 0V to 1.2V. The XBee device is programmed as a router. The base station has three different modules: Arduino microcontroller, LCD display and XBee device. The Arduino is programmed to receive the analog readings from the XBee device and convert them into temperature readings. The temperature readings are displayed on the LCD display. The XBee device is programmed as a coordinator. The design successfully worked for 3 remote stations and 1 base station.

Acknowledgements

I would like to thank all of my family and friends for their support during my time as a Master's student at Ryerson University. I also thank my parents who have always encouraged me set my goals high and helped me achieve them along the way. Finally, I would like to especially thank my wife Satinder for her encouragement, support and patience while I completed this project.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The ZigBee Alliance is the official organization that developed and maintains the ZigBee standard. The ZigBee standard is a radio frequency or wireless communication standard. The ZigBee standard is based on the IEEE 802.15.4 specification or standard. ZigBee is also known as a wireless personal area network (WPAN). Another example of a WPAN is the Bluetooth standard, which is also based on IEEE 802.15.4. The main differences between ZigBee and Bluetooth are:

- ZigBee requires less power and energy than Bluetooth. Which also means longer battery life for ZigBee devices.
- Bluetooth has greater data rate than ZigBee, 1Mbps vs 250kbps.
- Wireless communication distance is max 100 meters for ZigBee and 10 meters for Bluetooth (with line of sight).
- ZigBee has different network topologies built in (i.e. mesh and cluster tree).

For this project, a wireless temperature sensor system was built ans tested. Essentially several remote stations with temperature sensors will communicate back to a base station the temperature reading at each location. ZigBee was chosen due to its the low power requirement and low data rate. High data rates are not required if the system is just sending analog readings across the network. The objective for this project is to have one base station receive temperature readings from 3 remote stations. The intended application of this project is for home automation. Each remote station will have a ZigBee device connected to a temperature sensor. The base station will have a ZigBee device connected to an Arduino micro-controller. The user will be able to see the readings via an LCD display and scroll through each station.

1.2 Objective

The objective of this project are as follows:

1. Research and study the ZigBee standard.
2. Research and study Digi's XBee ZigBee based device.
3. Design and implement a wireless temperature sensor system using the XBee device.

Test ZigBee communication in a [8] [10] [4] [9] [6].

1.3 Organization of Report

This project report is organized in the following way. Chapter 2 provides a detailed background information on the ZigBee standard focusing on the basics of the standard and briefly going into the more advance features such as encryption. Chapter 3 provides background information on Digi's XBee ZigBee product which is used in the implementation. Chapter 4 provides detailed information and design of the wireless temperature sensor system. Finally Chapter 5 concludes the report.

Chapter 2

Overview of ZigBee

2.1 Basics of ZigBee

The ZigBee standard is a wireless communication protocol. Its main purpose is for low-power mesh networking. The different components and functions of modern communication network standards are separated into independent modules. These modules are also known as layers [2] [5].

The entire ZigBee protocol is built on top of the IEEE 802.15.4 network layer [11]. The ZigBee standard layers are depicted in Figure 2.1.

The ZigBee protocol or layers above IEEE 802.15.4 add 3 significant features: Ad hoc network creation, routing and self-healing mesh networks. Ad hoc network creation enables ZigBee to create a full network of ZigBee devices without the need of human interaction or involvement. The routing feature characterize how a ZigBee devise communicates with its target device by passing messages through a series of other ZigBee devices in the network. The self-healing mesh feature allows ZigBee networks to automatically restructure the routing to fix broken routes created by ZigBee devices that have left the network [11] [7] [3] [5].

All networks need a minimum of two devices communicating with each other to call it a network. ZigBee networks have only three types of devices: coordinator, end device and router. Every ZigBee network must have one coordinator and either one end device or router or both. ZigBee networks may be composed of any of the following combinations [11] [10]:

1. 1 coordinator and 1 (or multiple) router
2. 1 coordinator and 1 (or multiple) end device

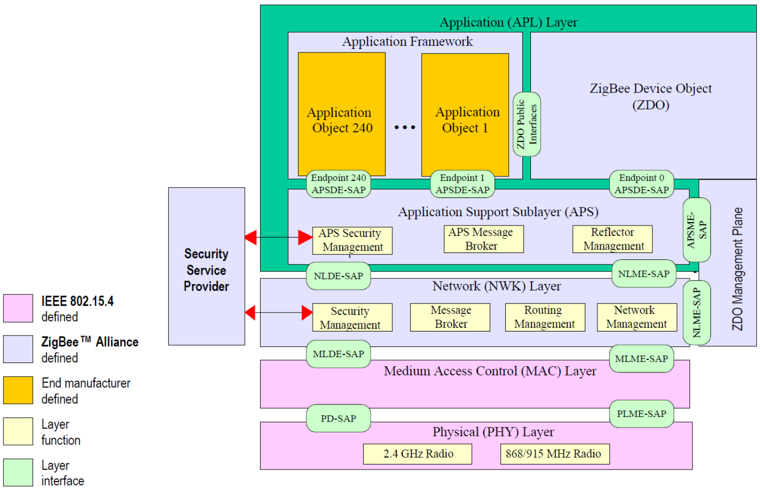


Figure 2.1: ZigBee standard layers. [11]

3. 1 coordinator, 1 (or multiple) router and 1 (or multiple) end devices

Coordinators: Zigbee networks require only one coordinator. The coordinators functions are to create the ZigBee network, manage addressing and keep the network secure [11] [4].

Routers: Routers are ZigBee network nodes capable of full functionality. They are capable of sending/receiving data and joining existing networks. These nodes also route data between nodes that are physically too far apart for direct communication. Multiple routers are allowed in ZigBee networks.

End Devices: End devices are nodes similar to routers, however they only have a few capabilities. They are capable of sending/receiving data and joining existing networks. Due to the reduced capabilities, end devices may use less expensive hardware. Power requirements are also reduced because of the reduced functions. End devices are also capable of going into sleep mode and only power on when required to send information, reducing energy requirements. These nodes need a parent device (coordinator or router) to help them join ZigBee networks. The parent device also stores any messages for end devices while they are in sleep mode [11] [7].

2.2 Network Topologies

There are 3 network topologies that ZigBee networks use: Pair, Star and Mesh. They are described below [11] [3]:

Pair Network Topology: The pair network topology consists of only 2 nodes. One node must be a coordinator the other may be either a router or end device. For practical purposes this topology will not get major benefit from using ZigBee radios.

Star Network Topology: The star topology consists of one coordinator and multiple end devices. In this network setup, all messages pass through the coordinator. There is no direct communication between end devices.

Mesh Network Topology: The mesh topology consists of one coordinator and multiple routers and end devices. Multiple end devices may have their parent as a router or coordinator (depends on which ever is better to establish communication with). Data or messages from end devices do not have to pass through coordinators to reach there destination, they pass through routers as well.

Cluster Tree Network Topology: This configuration is similar to mesh, however, the routers do not communicate with each other. Coordinators and routers create a communication back bone where end devices communicate with.

2.3 Addressing Basics

In order to send messages to ZigBee devices, you require the address of the destination device. All ZigBee devices may have up to 3 different addresses assigned. Each device has a 64-bit unique serial number which no other device will have. There is also a 16-bit address assigned to each network node by the coordinator. This address is only unique in the network. A third address called the node identifier may be assigned by the user. The node identifier is short string of text, which may or may not be unique in the network [11] [7] [3] [5].

Pan Address Multiple ZigBee network may exist in close proximity to each other. This is facilitated by the networks 16-bit Pan Address. When a ZigBee networks are created Pan Addresses are assigned to distinguish each of them. For XBee devices, Pan Addresses are manually assigned by the user. There are 65,536 Pan Addresses available to use, under every Pan Address there are 65,536 16-bit device addresses available to use.

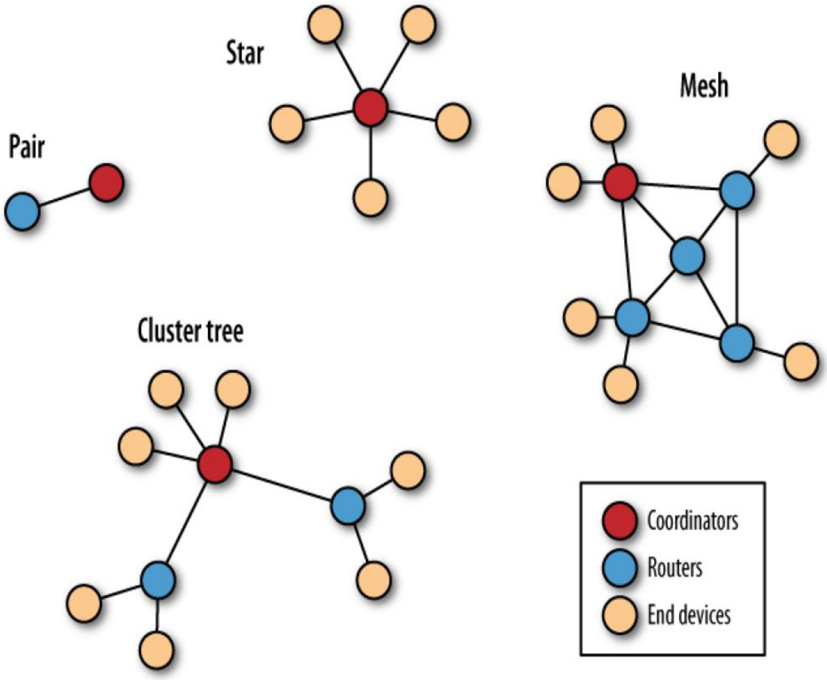


Figure 2.2: ZigBee pair, star, mesh, and cluster tree topologies [2].

Channels For ZigBee network to operate, all devices must be communicating on the same frequency (or channel). The coordinator looks at all available channels and chooses one for that network to communicate on. XBee devices automatically selects a channel for the user, so no user input is required and the user does not need to worry about channel selection [11] [7] [3] [5].

Type	Example	Unique
64-bit	0013A200403E0750	Yes, always and everywhere
16-bit	23F7	Yes, but only within a network
Node identifier	FREDS RADIO	Uniqueness not guaranteed

Table 2.1: ZigBee address examples [2].

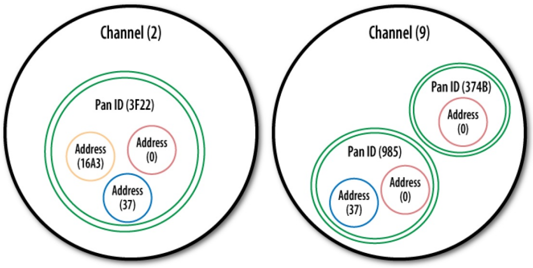


Figure 2.3: Venn diagram showing channel, PAN, and addressing [2].

2.4 ZigBee Stack Protocols

As with many communication standard, the ZigBee standard is divided into several layers. Each layer has a specific function to complete in the standard and they combine to form mesh communication structure for ZigBee. The IEEE 802.15.4 standard is base standard that the ZigBee standard is built on top of. The network layer defines the mesh network and how data is routed from device to another [11] [7] [8].

Application support sublayer: (APS) This layer defines a set of messages for typical ZigBee applications. A typical application is home automation. In essence the APS layer would allows devices for home automation from different manufacturers to communicate with each other seamlessly.

ZigBee device object: (ZDO) This layer is for working with ZigBee devices. It defines device discovery and network management.

The APS and ZDO layers are important and will help with communication between different brands of devices. It will also help with communication with ZigBee device profiles like Home Automation, Health Care and Smart Energy [11] [12] [7],

2.4.1 Application Support Layer (APS)

The APS layer defines profiles, clusters and endpoints for different applications. Profiles represents broad purpose of the application. Clusters represents the specific function or operation being performed. Endpoints represent the area within the ZigBee device where the operation will take placed. For XBee's, API (Application Specific Interface) frames send and receive APS messages [11] [12].

Application profiles may be developed by anyone or any organization and may be private or public. The ZigBee Alliance develops and maintains various public application profiles. Some of these profiles are listed below:

- Building Automation
- Health Care
- Home Automation
- Light Link
- Smart Energy
- Telecommunication Services
- Retail Services

These ZigBee public profiles may be used by anyone or any organization. Private application profiles may also be created and used by device manufactures. An example of a private application profile is Digi's drop-in networking profile for XBee radios. The is a proprietary application profile. Each application profile (public or private) has a 16-bit identifier. Each message is tagged with this identifier. Smart energy uses 0x0109. Application profiles also specify the clusters. Clusters define how ZigBee devices communicate with each other in the same application profile. Multiple clusters may be present on one endpoint [11] [12].

Endpoints allow multiple application profiles on ZigBee devices. A single ZigBee device may have smart energy and home automation profiles programmed. In essence each application profile (or endpoint) in a ZigBee device is another address in the device. For two ZigBee devices to communicate in the same application profile, APS messages are sent from the endpoint on the transmitting device to the endpoint on the receiving device. Endpoints have 8-bit identification number from 0x0 to 0xF0. Each endpoint has multiple clusters for the application profile its assigned too [2] [11].

ZigBee clusters are functions that application profiles may use. Two categories of clusters exist, client and server clusters. Client clusters transmit commands to server clusters. These commands may execute functions or change aspects of a server cluster. A device executing a service is a server cluster. Clusters are a assigned a 16-bit ID.

The ZigBee Cluster Library (ZCL) protocol is used by application profiles (but not all). The ZCL protocol specifies various commands and functions that may be used in different application profiles. The ZCL also specifies how clusters communicate with each other. The benefits of ZCL is that existing clusters may be re-used as they have already been created [2] [11].

2.5 Routing

The ZigBee standard has three different routing methods that may be used: Ad hoc On-demand Distance Vector (AODV) mesh routing, Many-to-one routing, and source routing.

Ad hoc On-demand Distance Vector (AODV) mesh routing: This is the most common default routing method for ZigBee networks. The method creates routing paths automatically from source devices to destination devices. Paths may go through (hop) multiple routers on its way to the destination device. Every step or hope automatically generating where the step will lead until the destination device is reached. This method automatically generates all paths and may be used on most network topologies. However, due to the large number of routing options the routing table may be to large for the device to store and therefore the same routing paths will need to generated every time it is needed. This will lead to low performance especially for networks with a large number of nodes [11].

Many-to-one Routing: This routing method was designed manly for networks where data or messages are transmitted from multiple nodes to one main or central node (or location). The central node transmits a routing configuration message to all nodes. All nodes within the network establish and save a path back to the central node. The routing paths only need to be calculated once for each node. This method provides superior performance for networks where one central node is receiving messages from multiple remote nodes. However, this routing method requires custom configuration. This method is also not efficient for messages between remote devices and messages from the central location to remote nodes [11] [9].

Source routing: This routing method was designed manly for networks where data or messages are transmitted from one or more central nodes (or location) to multiple destination devices. The route path from the central location to each remote device is stored either at the central node or the device thats controlling the central node (i.e. computer). This method provides superior performance for networks where central nodes are transmitting messages to multiple remote nodes. However, this routing method requires extensive custom configuration for the routing paths and may even require separate storage device for saving the routing paths [11].

2.6 Encryption and Security

Certain networks require security to protect the information that is being passed through the network. ZigBee network like all other network standards has security features. Deciding whether to add security to a network needs to be decision based on what the network is used for. Transmitting sensitive information such as banking would most likely require security features. On the other hand a remote controlled car would not require security features. It should also be noted that adding security feature will increase devices/networks resource cost and development time. When the adding security to ZigBee networks, a cost vs. benefit analysis needs to be looked at [11] [5] [6].

The ZigBee standard defines two mathematical keys to encrypt information being transmitted in the network: network keys and link keys. These two keys may be used concurrently if needed. Network keys encrypt and decrypts data from node to node until the destination of the message is reached. Essentially when a message is sent, it is encrypted first then sent. Once it reaches the next node it is decrypted and then re-encrypted and sent to the next node. Link keys encrypt data at the sender node and is only decrypted once at the receiving node. Network keys protect the network and link keys protect the data from being seen by other nodes except for the receiving node (nodes along the way will be be able to decrypt the message). Encrypting data adds bytes/overhead to every message sent and therefore decreases the packet size of a message that may be sent. More messages need to be transmitted to send the same amount of data [2] [11] [5].

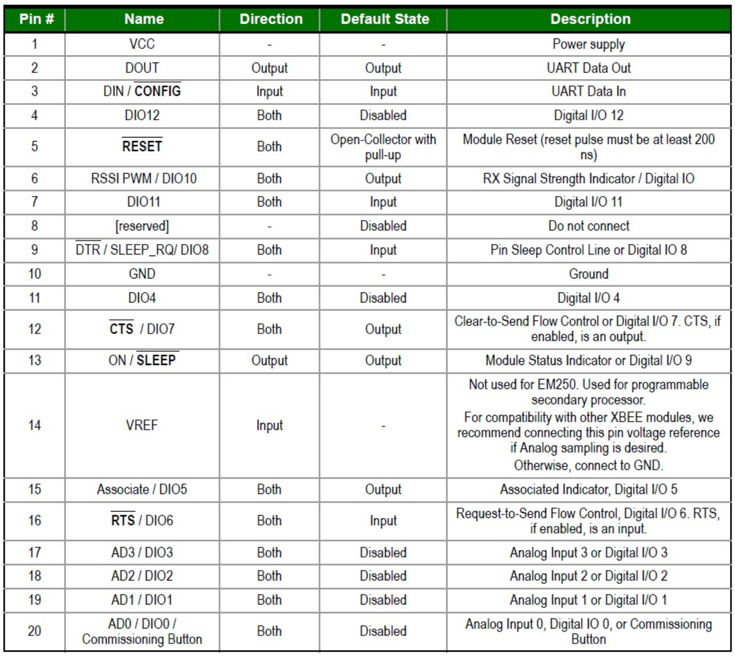
Chapter 3

Overview of XBee Devices from Digi

3.1 Basics of the XBee Device

For this project the XBee ZB (ZigBee) devices from Digi International are used to implement the wireless temperature sensor network. XBee devices (or modules) operate within the ISM 2.4GHz frequency band. Over the years Digi has released 3 versions of the XBee modules: Series 1, Series 2 and Series 2B (with Series 2B as the latest). Series 2 modules will be used for the project. The communication range for the series 2 is 133 feet (or 40 meters) indoor and 400 feet (or 120 meters) outdoors with line-of-sight. The transmit and receive operating current is 40mA at a supply voltage of 3.3V. Idle operating current is 15mA [1].

XBee modules may be ordered with several different antenna options such as: whip antenna, PCB antenna, RPSMA connector or U.FL connector. Each of these antenna options have there advantages and disadvantages. For this project all XBee modules have whip antennas. This antenna was chosen because the size of the module did not matter for the implementation and thus no extra cost was needed for a PCB antenna. Also, our implementation will not be inside a metal enclosure and thus will not need an external RPSMA or U.FL antenna. Figure 3.1 provides the pin assignments for XBee and Figure 3.2 provides the mechanical drawings of the XBee device [1].



Pin #	Name	Direction	Default State	Description
1	VCC	-	-	Power supply
2	DOUT	Output	Output	UART Data Out
3	DIN / CONFIG	Input	Input	UART Data In
4	DIO12	Both	Disabled	Digital I/O 12
5	RESET	Both	Open-Collector with pull-up	Module Reset (reset pulse must be at least 200 ns)
6	RSSI PWM / DIO10	Both	Output	RX Signal Strength Indicator / Digital IO
7	DIO11	Both	Input	Digital I/O 11
8	[reserved]	-	Disabled	Do not connect
9	DTR / SLEEP_RQ/ DIO8	Both	Input	Pin Sleep Control Line or Digital IO 8
10	GND	-	-	Ground
11	DIO4	Both	Disabled	Digital I/O 4
12	CTS / DIO7	Both	Output	Clear-to-Send Flow Control or Digital I/O 7. CTS, if enabled, is an output.
13	ON / SLEEP	Output	Output	Module Status Indicator or Digital I/O 9
14	VREF	Input	-	Not used for EM250. Used for programmable secondary processor. For compatibility with other XBEE modules, we recommend connecting this pin voltage reference if Analog sampling is desired. Otherwise, connect to GND.
15	Associate / DIO5	Both	Output	Associated Indicator, Digital I/O 5
16	RTS / DIO6	Both	Input	Request-to-Send Flow Control, Digital I/O 6. RTS, if enabled, is an input.
17	AD3 / DIO3	Both	Disabled	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Both	Disabled	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Both	Disabled	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0 / Commissioning Button	Both	Disabled	Analog Input 0, Digital IO 0, or Commissioning Button

Figure 3.1: Pin assignments for the XBee/XBee-PRO modules [1].

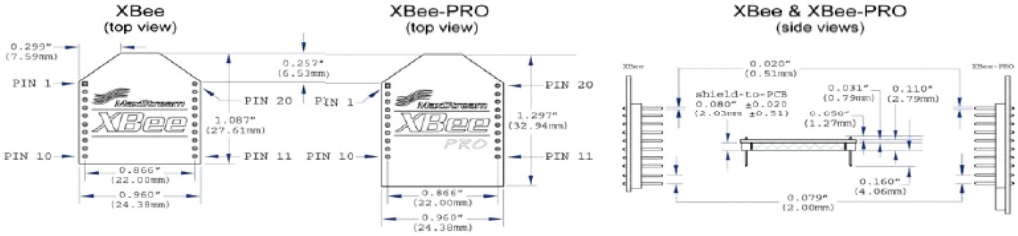


Figure 3.2: Mechanical drawings of the XBee/XBee-PRO ZB RF modules (antenna options not shown) [1].

3.2 API (Application Programming Interface) for XBee ZB Devices

XBee's may be operated in either transparent mode or API mode. XBee devices operating in transparent mode are equivalent to serial data lines. The DIN pin receives data and places the data in a queue for transmission. When the XBee devices receives the wireless transmitted data, the data is sent out of the XBee through the DOUT pin. The XBee when operated in transparent mode may be configured via AT command mode.

XBee's operated in API mode send and receive data differently. The major difference is that API mode is frame based. Data transmitted from the XBee or received by XBee is in standard frames. These frames may have additional operations as well. The DIN pin in API mode receives the data frames to be transmitted by XBee. When the XBee receives data frames, they are sent out through the DOUT pin [1].

For API all data is transmitted in a pre-defined order. Figure 3.3 provides an example data packet for API operation. Every frame starts with the start delimiter of 0x7E. The next part of the packet gives the length of the frame. This is followed by the frame data and check sum. Figure 3.4 provides an example data frame for API operation with a UART data frame. The data frame has 2 parts: API identifier and identifier specific data. Figure 3.5 gives the list of API identifiers available [1].

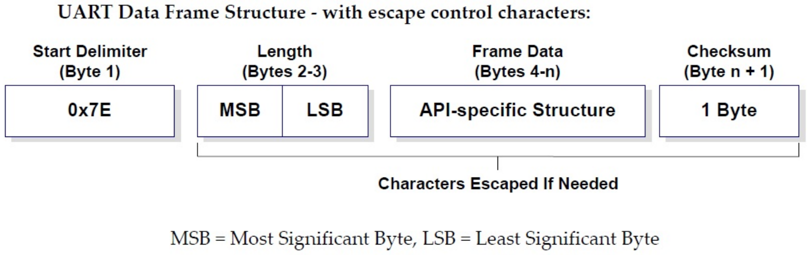

UART Data Frame Structure - with escape control characters:
Start Delimiter (Byte 1)
Length (Bytes 2-3)
Frame Data (Bytes 4-n)
Checksum (Byte n + 1)
0x7E
MSB
LSB
API-specific Structure
1 Byte
Characters Escaped If Needed
MSB = Most Significant Byte, LSB = Least Significant Byte

Figure 3.3: Data packet for API operation with escape characters (AP parameter = 2) [1].

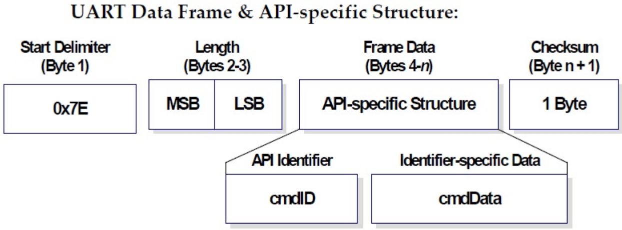
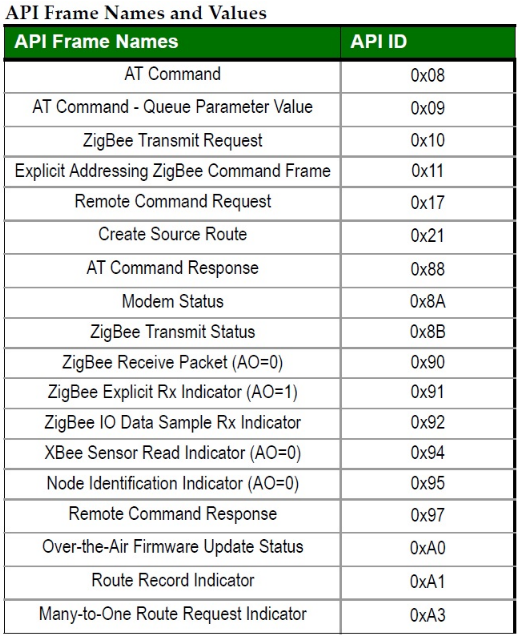


Figure 3.4: Data frame for UART & API-specific structure [1].



API Frame Names and Values

API Frame Names	API ID
AT Command	0x08
AT Command - Queue Parameter Value	0x09
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	0x92
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Route Record Indicator	0xA1
Many-to-One Route Request Indicator	0xA3

Figure 3.5: API frame names and values [1].

Chapter 4

Implementation of XBee for Wireless Temperature Sensors

4.1 Overall Structure of Temperature Sensors System

The temperature is being measured in 3 remote location (remote stations) and sending that information back to one central location (base station). Each remote station has one XBee device and temperature sensor which communicate back to the base station, the block diagram is shown in Figure 4.1. The base station is made up of a XBee device and an Arduino micro-controller with a LCD display to view the temperature at each location, the block diagram is shown in Figure 4.2.

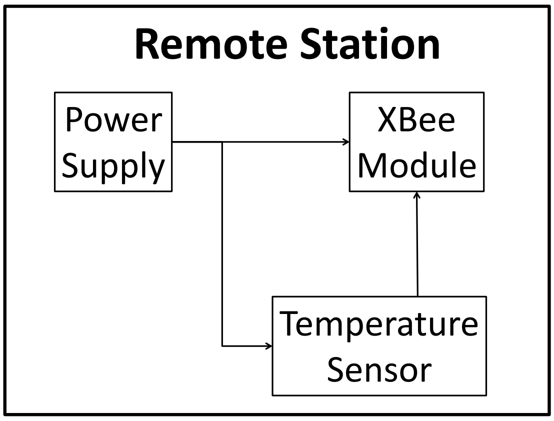


Figure 4.1: Block diagram for remote stations.

The XBee device in the remote station is programmed as a ”ZigBee Router AT”.

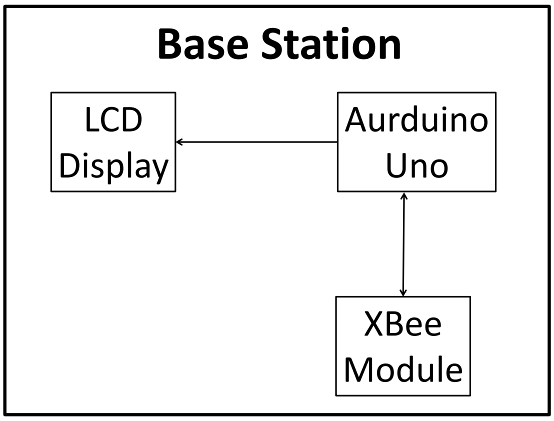


Figure 4.2: Block diagram for base station.

Each remote station is powered by a voltage regulator outputting 3.3V dc from a 5V dc input. The 5V input is supplied via an external wall wart power supply. The wall wart power supply may be substituted for a 9V battery if desired. Each remote station also has a temperature sensor outputting the temperature in Kelvin [K]. The XBee device in the remote station reads the temperature through one of the analog inputs, whose maximum input value is 1.2V (or 1200mV). The analog input takes the readings and send it to the 10-bit analog-to-digital converter (ADC). The ADC values are from 0x0000 to 0x3FF (or 0V to 1.2V), which gives 1023 possible values. The converted analog reading is sent to the base station as the readings are requested [1].

The XBee device in the base station is programmed as a "ZigBee Coordinator API". The XBee device is programmed as API at the base station to allow the Arduino to receive data packets continuously. The base stations XBee device receives the temperature readings from the remote stations XBee device. The reading received is the converted value from the XBee devices ADC. The reading is transmitted from the XBee device to the Arduino micro-controller for processing. The first step for the Arduino is to convert the ADC value to actual mV reading. The followings equation is used [1]:

$$AD_{mV} = (AD_{reading} * 1200mV)/1023$$

Once the actual mV readings are calculated it may be converted to temperature readings. The temperature sensor gives readings in Kelvin's. The Arduino converts the temperature readings in Kelvin's to degrees Celsius. The last step for the Arduino is to display the readings as the user request them.

4.2 Power Supply Design for Remote Stations

For the power supply the LD1117V33 voltage regulator was used. The voltage regulator takes a maximum input voltage of 15V dc and outputs 3.3V dc. For this project the regulator is supplied with an external 5V wall wart power supply. A 9V battery may also be used to power the voltage regulator. Decoupling capacitors of 10μF are placed at both the input and output. The input capacitor is to minimize and low-frequency noise from the 5V power supply. The output capacitor is to minimize any high-frequency noise from the output of the regulator. The data sheet for the LD1117V33 is given in the appendices.

4.3 Temperature Sensor Design for Remote Stations

For the temperature sensor the LM335 temperature sensor was used. It is a low-cost easy to use sensor, the LM335's data sheet is given in the appendices. The sensor requires an operating current of 400μA to 5mA. The sensor comes with 3 pins: V_{-}, V_{+} and ADJ (adjustment). Figure 4.3 shows the TO-92 package and its pin layout. Figure 4.4 shows the typical connection for measuring temperature. The V_{-} pin will be connected to ground of the power supply. The V_{+} is connected to a resistor, whose other lead is connected to the 3.3V power supply. The temperature reading is at the V_{+} (or output in Figure 4.4. The temperature readings is 10mV/K with the sensor calibrated at 2.982V for 25^oC. The resistor provides the sensors bias current.

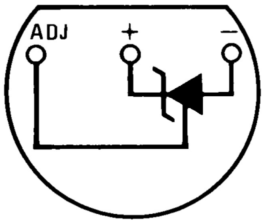


Figure 4.3: LM335 TO-92 package drawing and pin layout.

The resistor value is chosen such that the current stays within the 400μA and 5mA required operating range. To calculate the resistor value temperature range must be selected, which is -10^oC to 45^oC. From the temperature range output voltage range is calculated using $10mV/K$.

$$T_1 = -10^oC = 263.15K$$

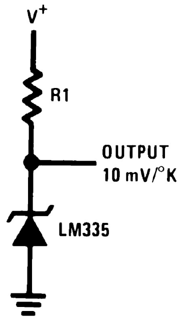


Figure 4.4: Typical LM335 implementation.

$$T_2 = 40^oC = 313.15K$$

$$263.15 \times \frac{10mV}{1K} = 2.6315V = V_{T_1}$$

$$313.15 \times \frac{10mV}{1K} = 3.1315V = V_{T_2}$$

The voltage across the resistor is calculated as follows:

$$V_{R(T_1)} = 3.3V - 2.6315V = 0.6685V$$

$$V_{R(T_2)} = 3.3V - 3.1315V = 0.1685V$$

The LM335 is operated best with the lowest current possible, this will reduce any heat affecting the temperature reading. The maximum resistance is calculated by taking the minimum 400μA current requirement and the lowest voltage across the resistor ($V_{R(T_2)}$):

$$R_{maximum(T_2)} = \frac{0.1685V}{400\mu A} = 421.25\Omega$$

For this project a 330Ω resistor is used. To make sure this resistor value keeps the operating current within range, the current is calculated for temperature range:

$$I_{T_1} = \frac{0.6685V}{330\Omega} = 2.0257mA$$

$$I_{T_2} = \frac{0.1685V}{330\Omega} = 510.6060\mu A$$

The above current calculations verify that the selected resistor value of 330Ω keeps the current within operating range. The temperature sensor can potentially output a voltage between 0V and 3.3V. The XBee devices analog input can only handle 0V to 1.2V.

In order for the temperature sensor to connect to XBee without exceeding the maximum 1.2V, a voltage divider is used to reduce the voltage [1]. This will be discussed in detail latter. In order to have the temperature readings accurate, the ADJ pin along with a 10K potentiometer is used to adjust the accuracy of the LM335. Figure 4.5 shows the configuration the temperature sensor with calibration.

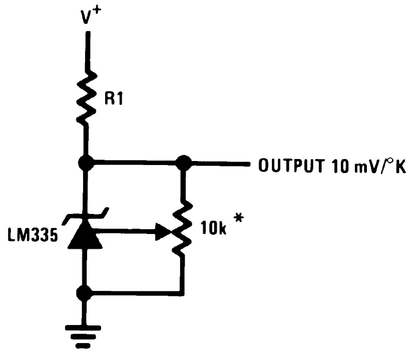


Figure 4.5: Typical LM335 implementation with calibration.

4.4 Remote Station Setup

The remote station has three components: power supply, temperature sensor, and XBee device. The XBee device and temperature sensor will be powered by the 3.3V power supply. The output of the power supply has a capacitor to minimize any noise from entering the XBee. The output of the power supply feeds inputs 1 and 10 on the XBee, with input 1 as the positive and input 10 as the negative (or ground). The power supply also feeds power to the temperature sensor via a 330Ω resistor. The temperature sensor is capable of outputting a maximum dc voltage 3.3V (also the supply voltage). However, the XBee's analog input can only handle a maximum of 1.2V. For the temperature sensor to connect to the XBee module, a simple voltage divider is used to divide the voltage by three [1]. When the readings are received at the central station, they are multiplied by 3 to obtain the correct reading. A 0.1μF capacitors is also placed between the XBee's analog input (AD0) and ground to reduce noise from interfering with the readings. Figure 4.6 shows the full schematic of the remote stations. Figure 4.7 and Figure 4.8 show the prototype created for this project.

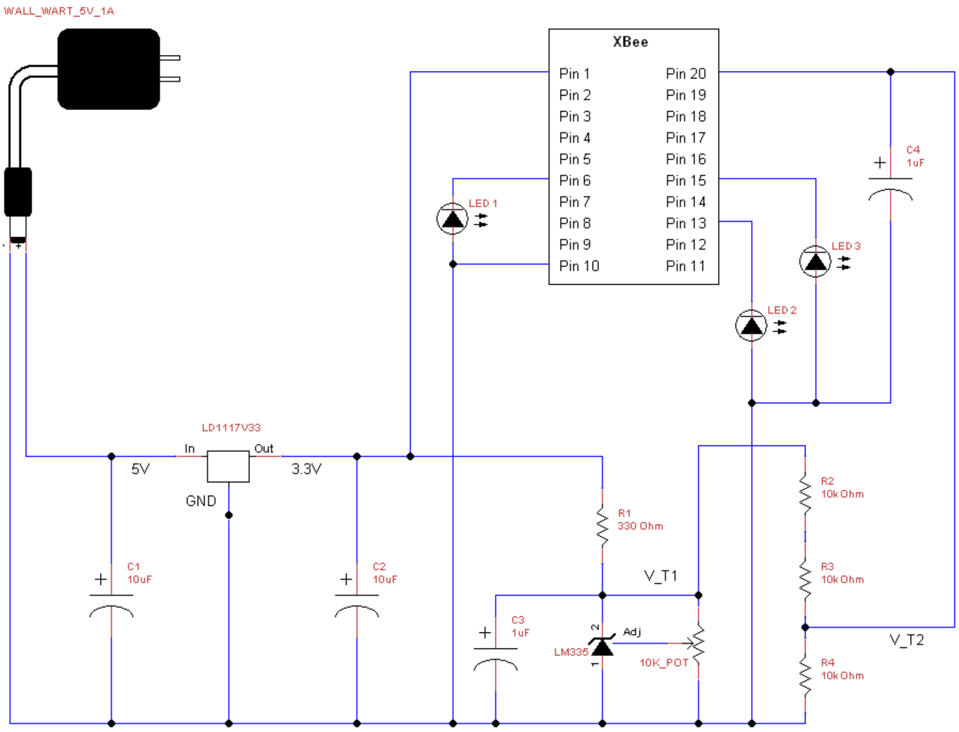


Figure 4.6: Circuit diagram for remote stations.

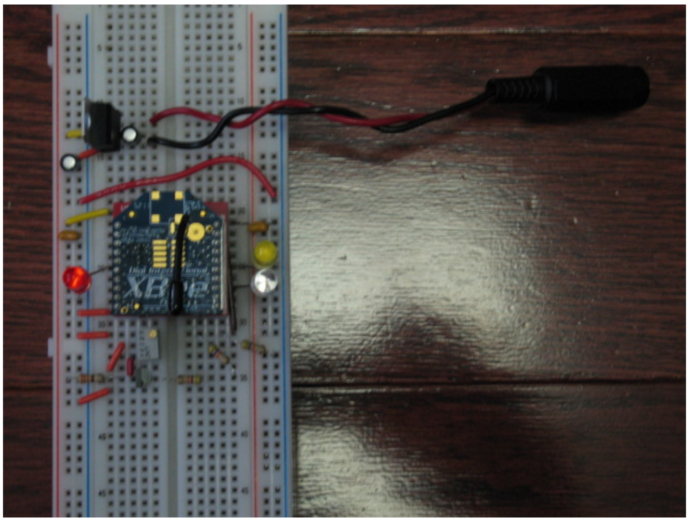


Figure 4.7: Remote station prototype top view.

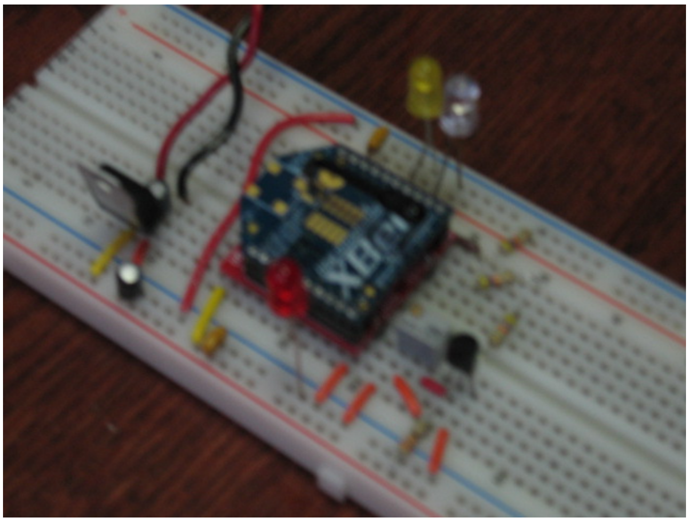


Figure 4.8: Remote station prototype side view.

4.5 Base Station Setup

For the base station we have decided to use the Arduino Uno board with the wireless prototyping shield and LCD keypad shield. The Arduino was selected for this project due to the easy to use programming interface. Figure 4.9 and Figure 4.10 show the prototype created for this project. The Arduino's programming file is given in following latter on in this chapter. For interfacing with the XBee's API we have used the "xbee.h" from Andrew Rapp. The program goes through the following steps:

1. Receive data packet from remote station.
2. Save remote stations address and analog reading for AD0.
3. Convert readings to degrees Celsius.
4. If remote stations address already exists then overwrite temperature reading with new reading.
5. If remote stations address does not exist then create save address and temperature reading.
6. If user presses down key then scroll to next remote station and display address and reading on screen.
7. Return to step 1.

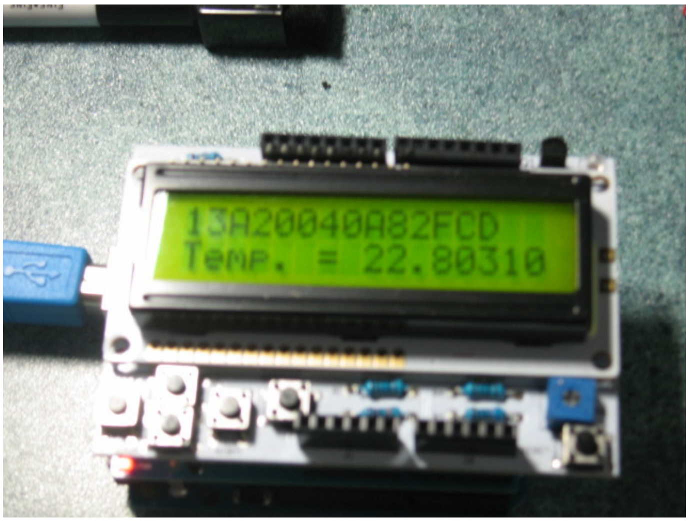


Figure 4.9: Base station prototype top view.

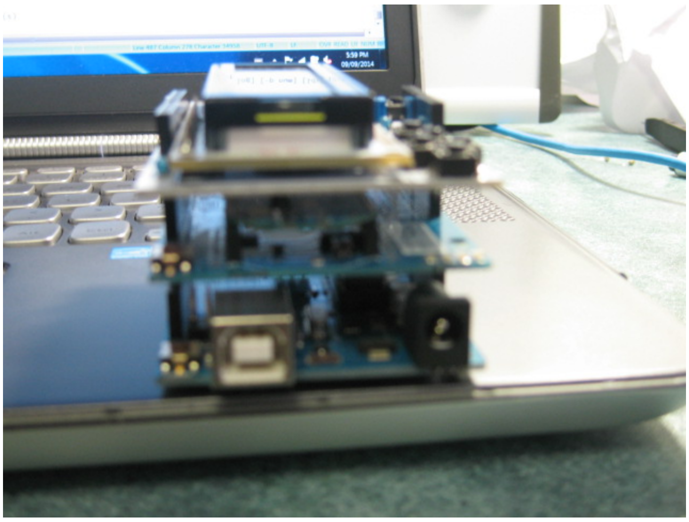


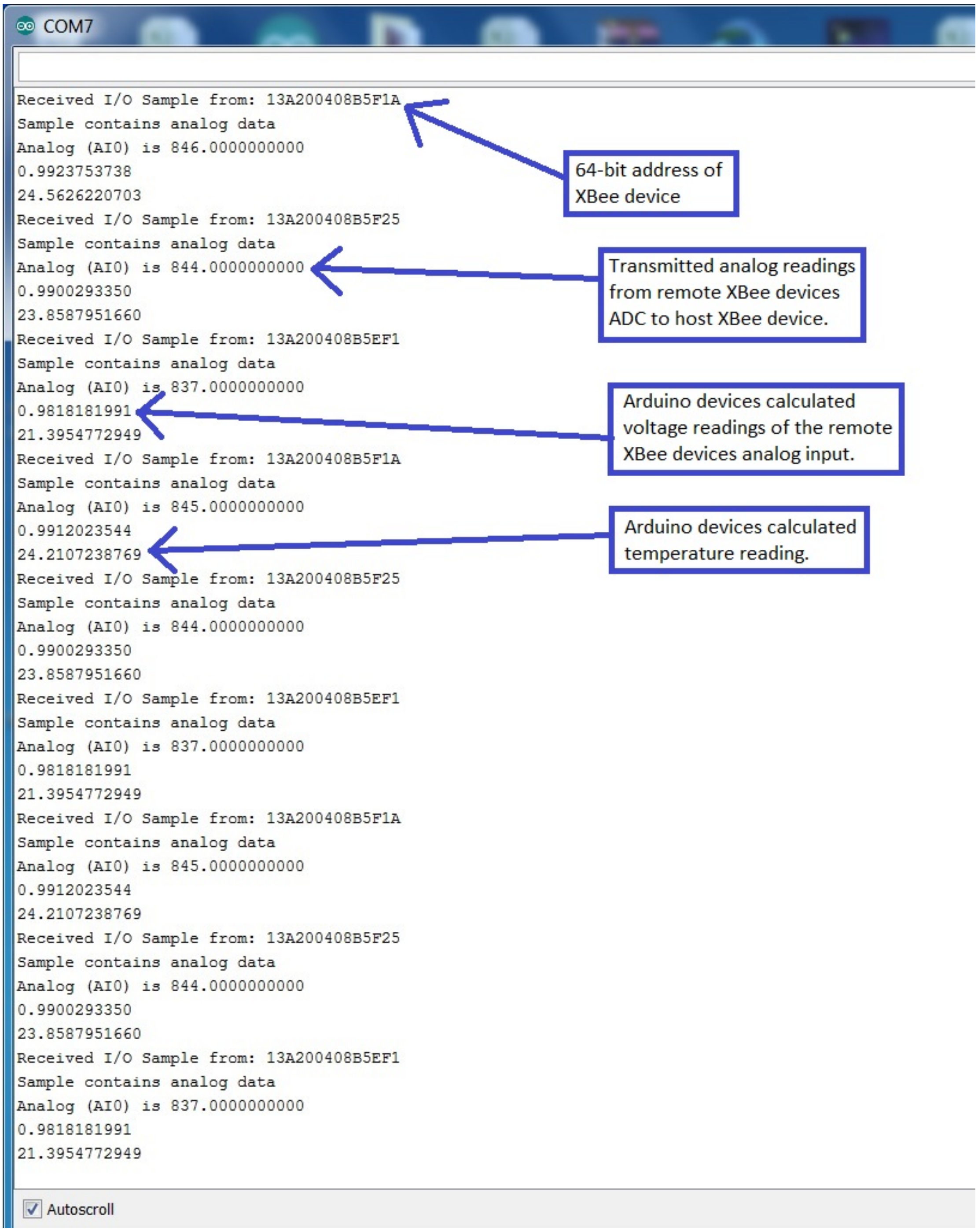
Figure 4.10: Base station prototype top view.

4.6 System Testing

With the system on all remote stations were communicating with the base stations. Figure 4.11 shows the serial port monitor output while system is on with 3 remote stations. The output is for each packet of information received by the base station from the any of the 3 remote stations. For each packet of information the following is shown:

- The 64-bit address for the remote XBee device
- The transmitted analog reading from the remote XBee devices ADC to the base stations XBee device.
- The Arduino devices calculated voltage reading of the remote XBee devices analog input.
- The Arudino devices calculated temperature readings in degrees Celsius.

The voltage readings on the serial monitor were compared to the readings taken with a volt meter from the circuit. All readings matched with very little error from the ADC. This proves that the system is successful in providing temperature readings from multiple remote locations.



```
Received I/O Sample from: 13A200408B5F1A
Sample contains analog data
Analog (AI0) is 846.0000000000
0.9923753738
24.5626220703
Received I/O Sample from: 13A200408B5F25
Sample contains analog data
Analog (AI0) is 844.0000000000
0.9900293350
23.8587951660
Received I/O Sample from: 13A200408B5EF1
Sample contains analog data
Analog (AI0) is 837.0000000000
0.9818181991
21.3954772949
Received I/O Sample from: 13A200408B5F1A
Sample contains analog data
Analog (AI0) is 845.0000000000
0.9912023544
24.2107238769
Received I/O Sample from: 13A200408B5F25
Sample contains analog data
Analog (AI0) is 844.0000000000
0.9900293350
23.8587951660
Received I/O Sample from: 13A200408B5EF1
Sample contains analog data
Analog (AI0) is 837.0000000000
0.9818181991
21.3954772949
Received I/O Sample from: 13A200408B5F1A
Sample contains analog data
Analog (AI0) is 845.0000000000
0.9912023544
24.2107238769
Received I/O Sample from: 13A200408B5F25
Sample contains analog data
Analog (AI0) is 844.0000000000
0.9900293350
23.8587951660
Received I/O Sample from: 13A200408B5EF1
Sample contains analog data
Analog (AI0) is 837.0000000000
0.9818181991
21.3954772949
```

64-bit address of XBee device

Transmitted analog readings from remote XBee devices ADC to host XBee device.

Arduino devices calculated voltage readings of the remote XBee devices analog input.

Arduino devices calculated temperature reading.

Figure 4.11: Serial monitor output while testing system.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The objective of this project was to create a temperature sensor system using ZigBee communication. Several remote stations would communicate temperature readings back to one base station. The objective was achieved using Digi's XBee ZB product. In the report the ZigBee protocol is first discussed in detail then we looked at Digi's ZigBee based product. Digi's ZigBee based product is called XBee. The XBee devices were the most critical part of the temperature system as they established communication between all remote stations and the base station. Each remote station had a power supply, temperature sensor and XBee device. The base station had an Arduino micro-controller, LCD display and XBee device. The XBee device was operated under its API mode. The API mode allowed fast and accurate data flow in the network. All remote stations communicated with the base station as they were required too. The project was successful in implementing ZigBee communication for a temperature sensor system.

5.2 Future Work

There are several future expansions that may be added on:

1. Using the XBee's sleep mode to only transmit reading less frequently or when they are requested by the user. This would save energy consumption, which is helpful if batteries are sued instead of external wall wart power supplies.

2. Added node identifiers to the XBee's for easier identification at the base station.

3. Adding an Ethernet shield to the Arduino at the base station. This would allow for the temperature to be sent up to a cloud service and therefore allow the user to view the temperature from remote locations with internet access.
4. Providing additional sensors such as humidity and pressure.

Appendix A

Arduino Code for Base Station

```
\\ ********************************************************
\\Ritchie Samrai
\\XBee Temperature Sensor System
\\ ********************************************************
\\ ******************* Included Libraries *******************
#include <LiquidCrystal.h>
#include <LCDKeypad.h>
#include <XBee.h>

\\ ******************* Variable Initialization *******************
LiquidCrystal lcd(8, 13, 9, 4, 5, 6, 7);
char msgs[5][16] = \{"Right Key OK ",
                     "Up Key OK    ",
                     "Down Key OK  ",
                     "Left Key OK  ",
                     "Select Key OK" };
int adc_key_val[5] ={ 50, 200, 400, 600, 800 };
int NUM_KEYS = 5;
int adc_key_in;
int key= -1;
int oldkey= -1;
float read1;
float read2;
uint32_t temp_MSB;
uint32_t temp_LSB;

uint32_t sensor_address_MSB[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

```
uint32_t sensor_address_LSB[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

float temperature[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int nextSensor = 0;

XBee xbee = XBee();

ZBRxIoSampleResponse ioSample = ZBRxIoSampleResponse();

XBeeAddress64 test = XBeeAddress64();

\\ ******************* SETUP *******************
Void setup() {
Serial.begin(9600);
  xbee.setSerial(Serial);
  \\start soft serial

  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temperature");
  lcd.setCursor(0,1);
  lcd.print("Reader");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0,0);
}


\\  ******************* Main Loop *******************
void loop() {
 \\ read data packet
 xbee.readPacket();

 \\look at what kind of response was receives
  if (xbee.getResponse().isAvailable()) {
   \\ received correct response
   if (xbee.getResponse().getApiId() == ZB_IO_SAMPLE_RESPONSE) {
    xbee.getResponse().getZBRxIoSampleResponse(ioSample);
```

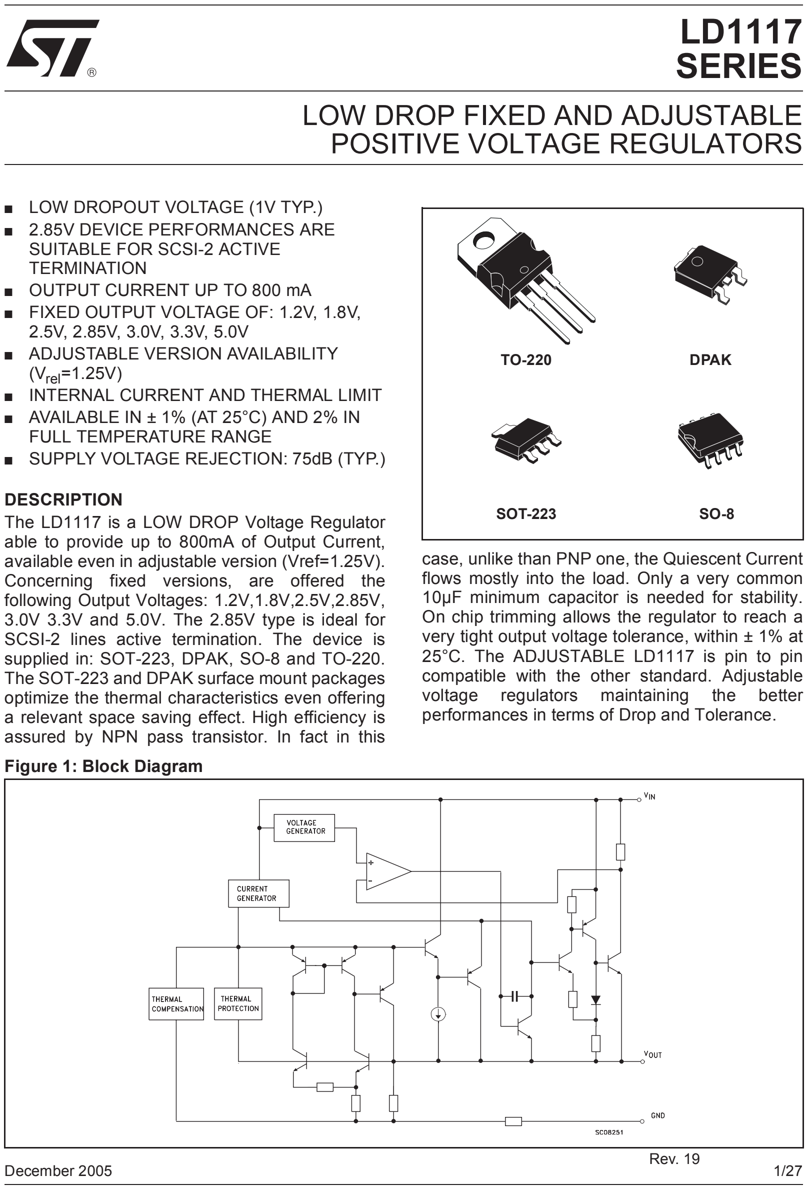
```
\\ print xbee address on serial port
Serial.print("Received I/O Sample from: ");
temp_MSB=ioSample.getRemoteAddress64().getMsb();
Serial.print(temp_MSB, HEX);
temp_LSB=ioSample.getRemoteAddress64().getLsb();
Serial.print(temp_LSB, HEX);
Serial.println("");

\\ make sure data packet has analog readings
if (ioSample.containsAnalog()) {
 Serial.println("Sample contains analog data");
}

\\ read analog inputs
for (int i = 0; i <= 4; i++) {
 if (ioSample.isAnalogEnabled(i)) {
  \\ print address and reading on serial port
  Serial.print("Analog (AI");
  Serial.print(i, DEC);
  Serial.print(") is ");
  read1 = (float) ioSample.getAnalog(i);
  Serial.println(read1, DEC);
  read2 = read1 / 1023.0 * 1.2;
  Serial.println(read2, DEC);
  read2 = (read2 * 3.0 * 100.0) - 273.15; \\ convert readings to degC
  Serial.println(read2, DEC);
 }
}

\\  see if sensor already exists
for (int g = 0; g <= 9; g++) {
 if (temp_MSB == sensor_address_MSB[g] && temp_LSB == sensor_address_LSB[g]){
  temperature[g] = read2;
  g = 10;
 }
 else if (sensor_address_MSB[g] == 0 && sensor_address_LSB[g] == 0){
  sensor_address_MSB[g] = temp_MSB;
  sensor_address_LSB[g] = temp_LSB;
  temperature[g] = read2;
  g = 10;
 }
```

```
    }
   }

   \\ got a response, but the response was not what was expected
   else {
    Serial.print("Expected I/O Sample, but got ");
    Serial.print(xbee.getResponse().getApiId(), HEX);
   }
  }

  \\ did not get any response
  else if (xbee.getResponse().isError()) {
   Serial.print("Error reading packet.  Error code: ");
   Serial.println(xbee.getResponse().getErrorCode());
  }
  adc_key_in = analogRead(0);    \\  read the value from the sensor
  key = get_key(adc_key_in);  \\  convert into key press

  \\ scroll through remote stations if down button pressed
  if (key == 2){
   delay(50);
   adc_key_in = analogRead(0); \\  read the value from the sensor
   key = get_key(adc_key_in);  \\  convert into key press
   if (key == 2){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(sensor_address_MSB[nextSensor], HEX);
    lcd.print(sensor_address_LSB[nextSensor], HEX);
    lcd.setCursor(0, 1);
    lcd.print("Temp. = ");
    lcd.print(temperature[nextSensor], DEC);
    if (nextSensor == 9){
     nextSensor = 0;
    }
    else {
     nextSensor = nextSensor + 1;
    }
   }
  }
 delay(100);
}
```

```
\\ ******************* Convert ADC value to key number *******************
int get_key(unsigned int input) {
int k;
  for (k = 0; k < NUM_KEYS; k++) {
if (input < adc_key_val[k]) {
return k;
    }
  }
  if (k >= NUM_KEYS)k = -1;  // No valid key pressed
  return k;
}
```

Appendix B

Data Sheets (front page only)



LD1117 SERIES

LOW DROP FIXED AND ADJUSTABLE POSITIVE VOLTAGE REGULATORS

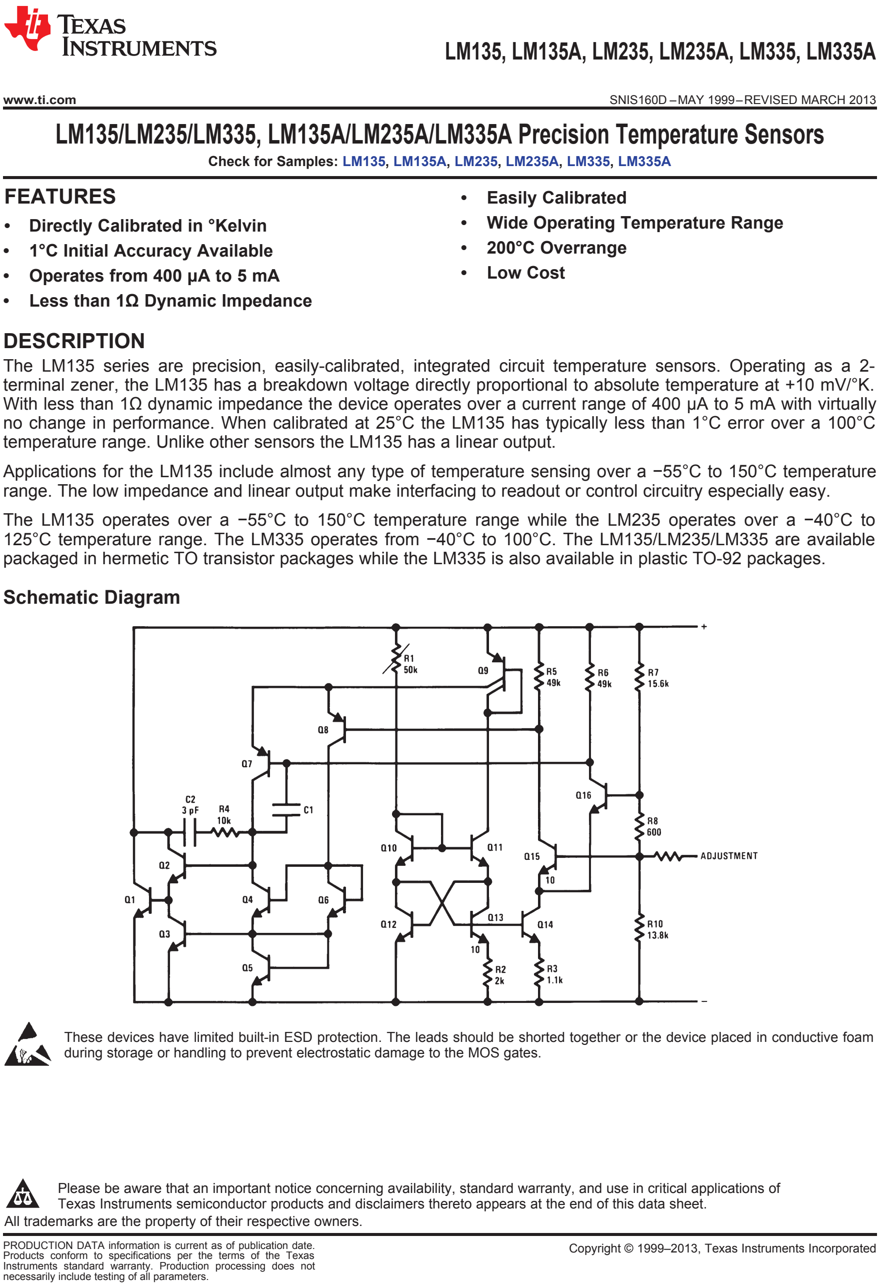
- LOW DROPOUT VOLTAGE (1V TYP.)
- 2.85V DEVICE PERFORMANCES ARE SUITABLE FOR SCSI-2 ACTIVE TERMINATION
- OUTPUT CURRENT UP TO 800 mA
- FIXED OUTPUT VOLTAGE OF: 1.2V, 1.8V, 2.5V, 2.85V, 3.0V, 3.3V, 5.0V
- ADJUSTABLE VERSION AVAILABILITY (V_{rel}=1.25V)
- INTERNAL CURRENT AND THERMAL LIMIT
- AVAILABLE IN ± 1% (AT 25°C) AND 2% IN FULL TEMPERATURE RANGE
- SUPPLY VOLTAGE REJECTION: 75dB (TYP.)

DESCRIPTION

The LD1117 is a LOW DROP Voltage Regulator able to provide up to 800mA of Output Current, available even in adjustable version (Vref=1.25V). Concerning fixed versions, are offered the following Output Voltages: 1.2V,1.8V,2.5V,2.85V, 3.0V 3.3V and 5.0V. The 2.85V type is ideal for SCSI-2 lines active termination. The device is supplied in: SOT-223, DPAK, SO-8 and TO-220. The SOT-223 and DPAK surface mount packages optimize the thermal characteristics even offering a relevant space saving effect. High efficiency is assured by NPN pass transistor. In fact in this case, unlike than PNP one, the Quiescent Current flows mostly into the load. Only a very common 10µF minimum capacitor is needed for stability. On chip trimming allows the regulator to reach a very tight output voltage tolerance, within ± 1% at 25°C. The ADJUSTABLE LD1117 is pin to pin compatible with the other standard. Adjustable voltage regulators maintaining the better performances in terms of Drop and Tolerance.

Figure 1: Block Diagram

December 2005
Rev. 19
1/27



TEXAS INSTRUMENTS

LM135, LM135A, LM235, LM235A, LM335, LM335A

www.ti.com
SNIS160D –MAY 1999–REVISED MARCH 2013

LM135/LM235/LM335, LM135A/LM235A/LM335A Precision Temperature Sensors

Check for Samples: LM135, LM135A, LM235, LM235A, LM335, LM335A

FEATURES

- Directly Calibrated in °Kelvin
- 1°C Initial Accuracy Available
- Operates from 400 µA to 5 mA
- Less than 1Ω Dynamic Impedance
- Easily Calibrated
- Wide Operating Temperature Range
- 200°C Overrange
- Low Cost

DESCRIPTION

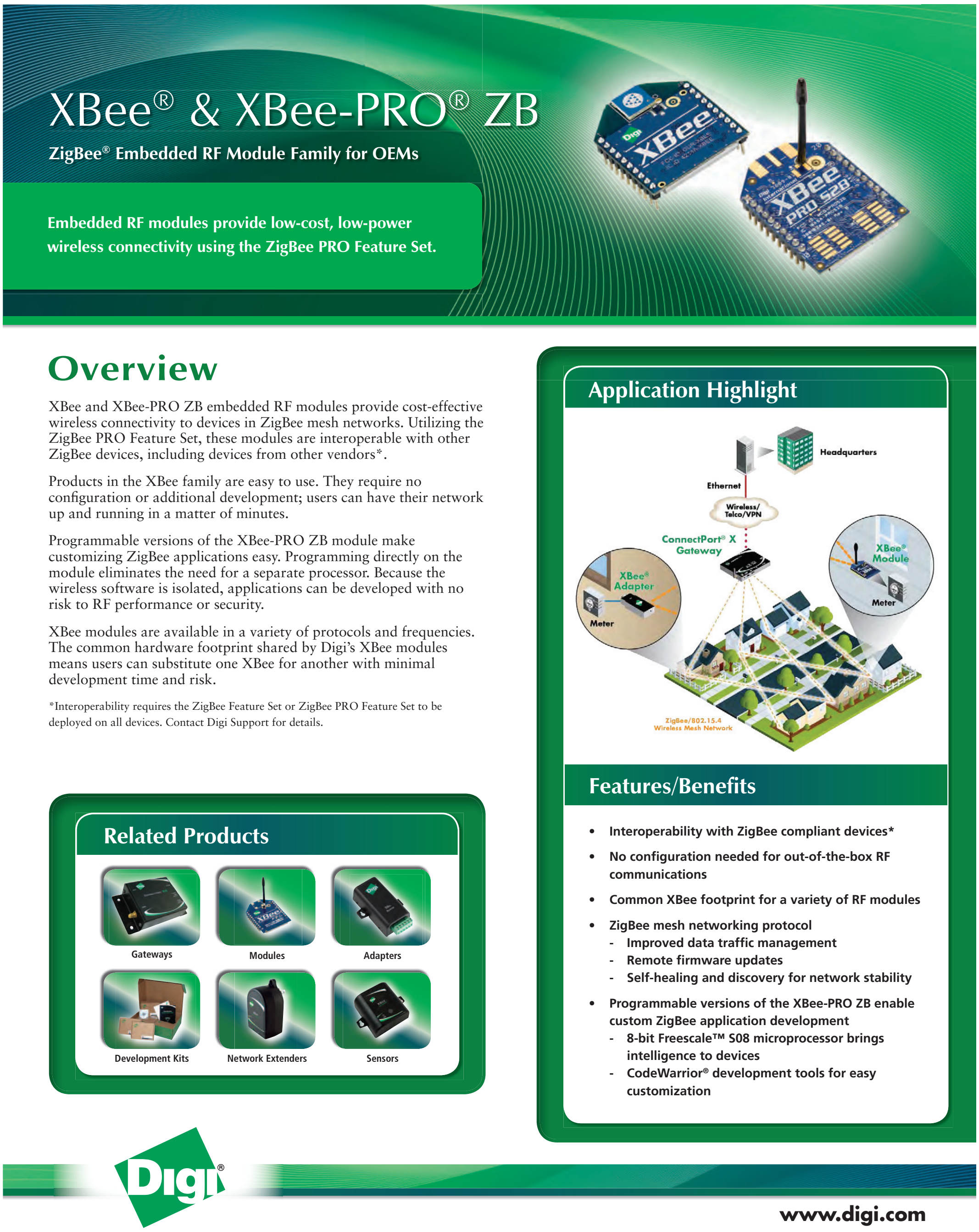
The LM135 series are precision, easily-calibrated, integrated circuit temperature sensors. Operating as a 2-terminal zener, the LM135 has a breakdown voltage directly proportional to absolute temperature at +10 mV/°K. With less than 1Ω dynamic impedance the device operates over a current range of 400 µA to 5 mA with virtually no change in performance. When calibrated at 25°C the LM135 has typically less than 1°C error over a 100°C temperature range. Unlike other sensors the LM135 has a linear output.

Applications for the LM135 include almost any type of temperature sensing over a −55°C to 150°C temperature range. The low impedance and linear output make interfacing to readout or control circuitry especially easy.

The LM135 operates over a −55°C to 150°C temperature range while the LM235 operates over a −40°C to 125°C temperature range. The LM335 operates from −40°C to 100°C. The LM135/LM235/LM335 are available packaged in hermetic TO transistor packages while the LM335 is also available in plastic TO-92 packages.

Schematic Diagram

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.



XBee® & XBee-PRO® ZB

ZigBee® Embedded RF Module Family for OEMs

Embedded RF modules provide low-cost, low-power wireless connectivity using the ZigBee PRO Feature Set.

Overview

XBee and XBee-PRO ZB embedded RF modules provide cost-effective wireless connectivity to devices in ZigBee mesh networks. Utilizing the ZigBee PRO Feature Set, these modules are interoperable with other ZigBee devices, including devices from other vendors*.

Products in the XBee family are easy to use. They require no configuration or additional development; users can have their network up and running in a matter of minutes.

Programmable versions of the XBee-PRO ZB module make customizing ZigBee applications easy. Programming directly on the module eliminates the need for a separate processor. Because the wireless software is isolated, applications can be developed with no risk to RF performance or security.

XBee modules are available in a variety of protocols and frequencies. The common hardware footprint shared by Digi's XBee modules means users can substitute one XBee for another with minimal development time and risk.

*Interoperability requires the ZigBee Feature Set or ZigBee PRO Feature Set to be deployed on all devices. Contact Digi Support for details.

Related Products

Application Highlight

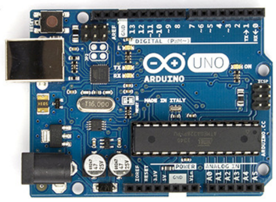
Features/Benefits

- Interoperability with ZigBee compliant devices*
- No configuration needed for out-of-the-box RF communications
- Common XBee footprint for a variety of RF modules
- ZigBee mesh networking protocol
  - Improved data traffic management
  - Remote firmware updates
  - Self-healing and discovery for network stability
- Programmable versions of the XBee-PRO ZB enable custom ZigBee application development
  - 8-bit Freescale™ S08 microprocessor brings intelligence to devices
  - CodeWarrior® development tools for easy customization

www.digi.com

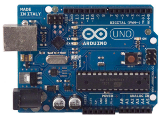
Search the Arduino Website

Arduino Uno

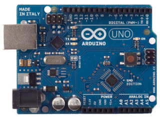


(http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg)
Arduino Uno R3 Front

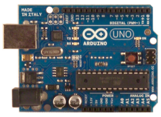
(http://arduino.cc/en/uploads/M
Arduino Uno R3 Back



(http://arduino.cc/en/uploads/Main/ArduinoUno_r2_front.jpg)
Arduino Uno R2 Front



(http://arduino.cc/en/uploads/Main/ArduinoUnoSmd.jpg)
Arduino Uno SMD



(http://arduino.cc/en/uploads/M
Arduino Uno Front

Buy From Arduino Store (http://store.arduino.cc/index.php?main_page=product_info&cPath=11&products_id=195)

Buy From Distributors (http://arduino.cc/en/Main/Buy)

Overview

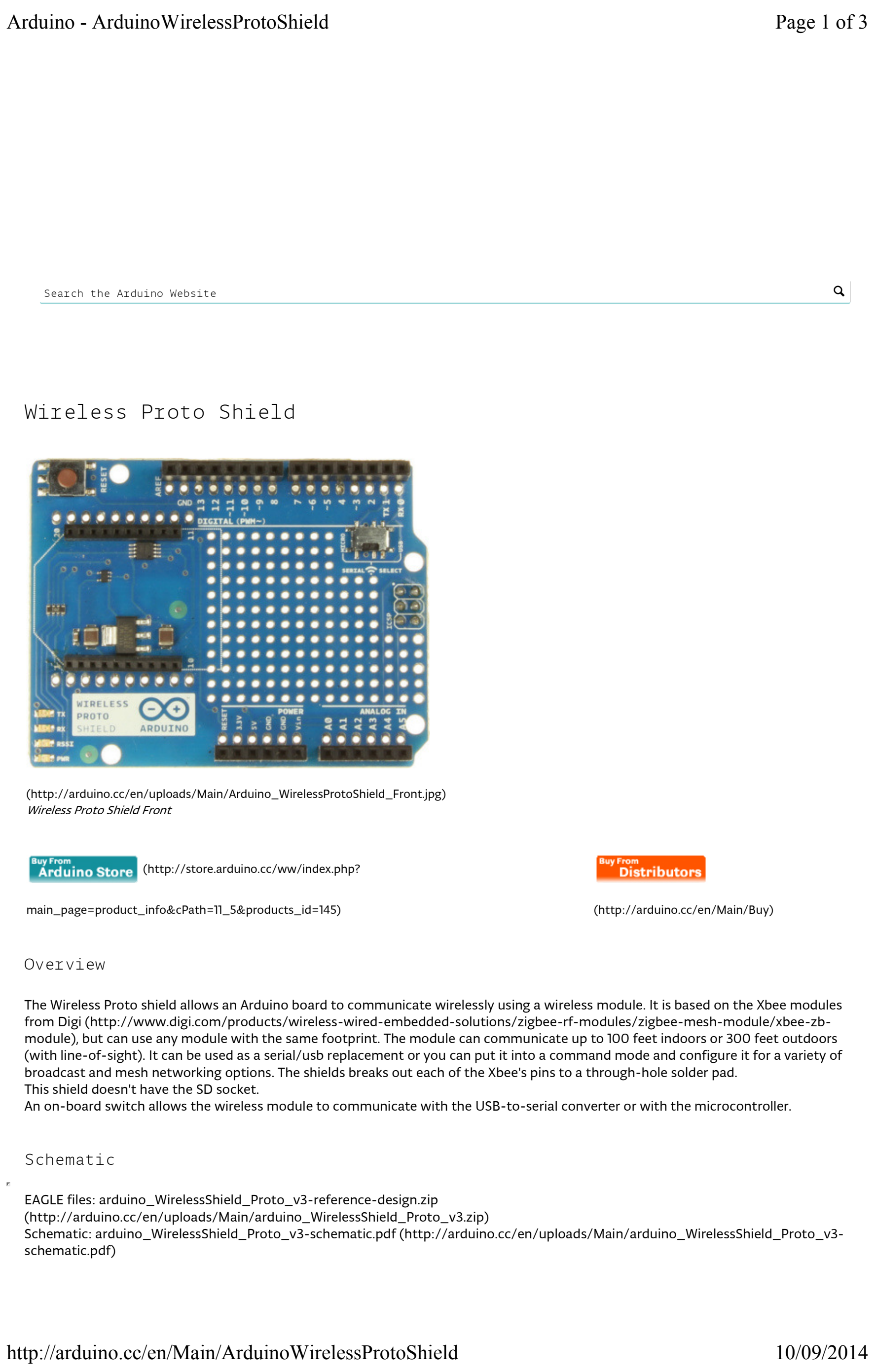
The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet (http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode (http://arduino.cc/en/Hacking/DFUProgramming8U2).

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.



Arduino - ArduinoWirelessProtoShield Page 1 of 3

Search the Arduino Website

Wireless Proto Shield

(http://arduino.cc/en/uploads/Main/Arduino_WirelessProtoShield_Front.jpg)
Wireless Proto Shield Front

Buy From Arduino Store (http://store.arduino.cc/ww/index.php?main_page=product_info&cPath=11_5&products_id=145)

Buy From Distributors (http://arduino.cc/en/Main/Buy)

Overview

The Wireless Proto shield allows an Arduino board to communicate wirelessly using a wireless module. It is based on the Xbee modules from Digi (http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module), but can use any module with the same footprint. The module can communicate up to 100 feet indoors or 300 feet outdoors (with line-of-sight). It can be used as a serial/usb replacement or you can put it into a command mode and configure it for a variety of broadcast and mesh networking options. The shields breaks out each of the Xbee's pins to a through-hole solder pad.
This shield doesn't have the SD socket.
An on-board switch allows the wireless module to communicate with the USB-to-serial converter or with the microcontroller.

Schematic

EAGLE files: arduino_WirelessShield_Proto_v3-reference-design.zip (http://arduino.cc/en/uploads/Main/arduino_WirelessShield_Proto_v3.zip)
Schematic: arduino_WirelessShield_Proto_v3-schematic.pdf (http://arduino.cc/en/uploads/Main/arduino_WirelessShield_Proto_v3-schematic.pdf)

http://arduino.cc/en/Main/ArduinoWirelessProtoShield 10/09/2014

Bibliography

[1] Digi International Inc. *XBee ® /XBee-PRO ® ZB RF Modules*, 2013.

[2] R. Faludi. *Building Wireless Sensor Networks*. OReilly Media, Inc., 2011.

[3] K. Gill, F. Y. Shuang-Hua Yang, and X. Lu. A zigbee-based home automation system. *IEEE Transactions on Consumer Electronics*, 55(2):422–430, May 2009.

[4] C. Gomez and J. Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, pages 92–101, June 2010.

[5] D.-M. Han and J.-H. Lim. Smart home energy management system using ieee 802.15.4 and zigbee. *IEEE Transactions on Consumer Electronics*, 56(3):1403–1410, August 2010.

[6] M. Idoudi, H. ELKHORCHANI, and K. GRAYAA. Performance evaluation of wireless sensor networks based on zigbee technology in smart home. *2013 International Conference on Electrical Engineering and Software Applications (ICEESA)*, pages 1–4, March 2013.

[7] J.-S. Lee, Y.-W. Su, and C.-C. Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON) Nov. 5-8, 2007, Taipei, Taiwan*, pages 46–51, 2007.

[8] M. J. LEE, J. ZHENG, Y.-B. KO, and D. M. SHRESTHA. Emerging standards for wireless mesh technology. *IEEE Wireless Communications*, pages 56–63, April 2006.

[9] A. A. Siddiqui, A. W. Ahmad, H. K. Yang, and C. Lee. Zigbee based energy efficient outdoor lighting control system. *14th International Conference on Advanced Communication Technology (ICACT)*, pages 916–919, February 2012.

[10] A. Wheeler. Commercial applications of wireless sensor networks using zigbee. *IEEE Communications Magazine*, pages 70–77, April 2007.

[11] ZigBee Alliance. *ZIGBEE SPECIFICATION*, January 17, 2008.

[12] ZigBee Alliance. *ZIGBEE HOME AUTOMATION PUBLIC APPLICATION PROFILE*, June 6, 2013.