The Report committee for Michael A. Comai
Certifies that this is the approved version of the following report:

Developing a ZigBee Home Automation Network

APPROVED BY

SUPERVISING COMMITTEE:

Ranjit Gharpurey, Supervisor

Mark McDermott

Developing a ZigBee Home Automation Network

by

Michael A. Comai, B.S.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2011

To my wife.

Developing a ZigBee Home Automation Network

Michael A. Comai, M.S.E.
The University of Texas at Austin, 2011

Supervisor: Ranjit Gharpurey

As electronics and appliances in the home continue to evolve, wireless networks that can monitor and control these are becoming more prevalent. Development platforms for technologies like RF4CE and ZigBee are becoming more available but can still be costly for a complete platform that supports interoperable devices with public ZigBee profiles. This paper details the design and implementation of a ZigBee Home Automation network using free tools on a low cost ZigBee development platform. The platform utilizes a dedicated processor for the ZigBee stack with SPI, UART, and USB interface, allowing the application code that was developed to be integrated into a wide variety of environments.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A modern home today has tens to hundreds of control devices - switches, appliances, heating / cooling systems and more. The devices being controlled consume significant amounts of energy. While many of these ordinary household devices have become more efficient, technologies are available or becoming available that further increase this efficiency through automation. For example, automatically turning on and off lighting when a sensor detects a change in occupancy in a room or at certain times of day can reduce electricity usage. Another example is management of the heating and cooling system in a house (as shown in Figure 1.1 this is the biggest source of energy use in the average home). The programmable thermostat is a common example of a device that addresses this need. With the use of a programmable thermostat temperature setpoint adjustments can be scheduled to reduce energy costs. While this can be done without automation, doing so requires frequent attention and sacrifice in comfort (e.g. waking up to a cold house in the winter or coming home from work to a hot house in the summer) thus may not be practical for a typical family. With a programmable thermostat the degree and frequency of cost savings can be easier to realize.

The previous example illustrates one application of the evolution of automation capability not unlike the motivation behind creation of technologies

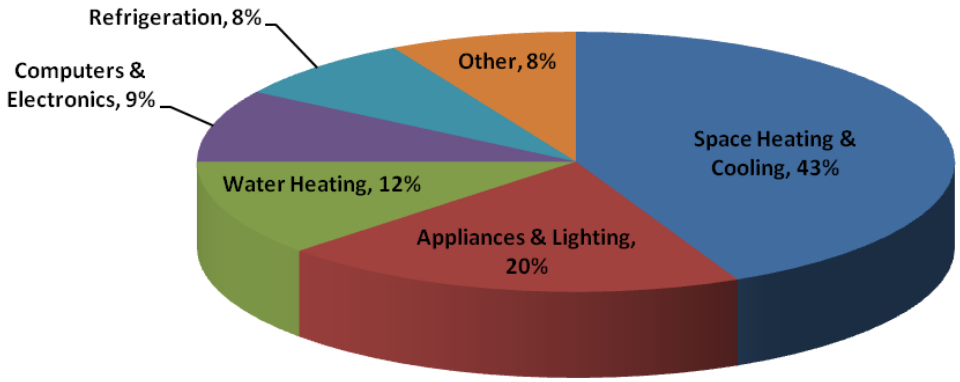

Figure 1.1: Average Home Energy Costs[3]

like ZigBee and IEEE 802.15.4. Energy efficiency provided the foundation of a market, but these advances in technology are creating broader standards, reducing costs for wider markets. With the emergence of wireless networks, the complexity for an end user to install a network has substantially decreased while capabilities have continued to increase.

Most homes today have some type of automation device in the form of a programmable thermostat or timer for a light etc., and the next generation of technology will enable integrating these types of devices into a central home network. Largely led by the consumer electronics market, shipment for these devices is expected to grow substantially in the coming years.

1.1 Wireless Personal Area Networks

The ZigBee Alliance, along with IEEE, has developed a complete protocol stack for the implementation of a low cost, low power network with a few of the target markets being consumer electronics and home automation.

This protocol stack is intended to enable a standard for interoperability among manufacturers. The Physical Layer (PHY) and Medium Access Control Layer (MAC) are defined by the IEEE 802.15.4 standard, and the Network Layer (NWK) and Application Layer (APL) are defined by the ZigBee Alliance as shown in Figure 1.2.

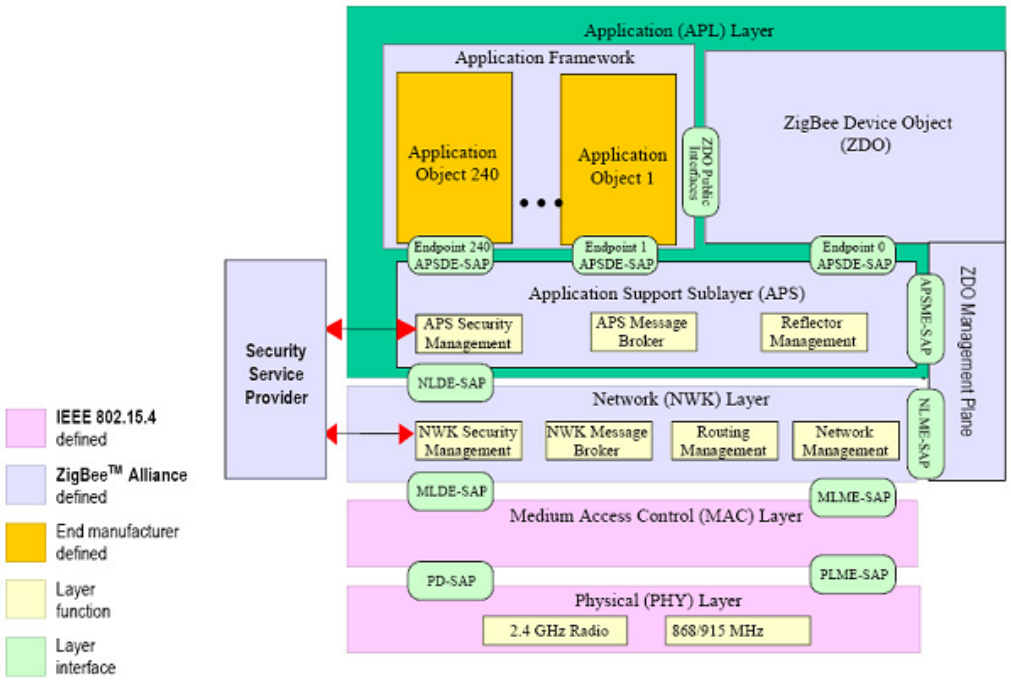

Figure 1.2: Overview of the ZigBee stack [2].

1.2 IEEE 802.15.4 and ZigBee

The IEEE 802.15.4 Specification was developed to support low-rate Wireless Personal Area Network (WPAN) involving little or no infrastructure. IEEE 802.15.4 has two PHYs that can operate in separate frequency ranges.

The lower range covers the 868 MHz European band as well as the 915 MHz band used in countries like the US and Australia while the upper range is widely used throughout the world.

The MAC layer handles all access to the physical radio channel. Resources required to implement the MAC layer alone could be quite high - some optimized implementations have required as high as 24K for a minimally featured IEEE 802.15.4 implementation. The ZigBee specification identifies IEEE 802.15.4 requirements that are optional to a ZigBee device in order to lower the resource requirements needed for compliance.

The ZigBee Application layer is comprised of three entities - the Application Support Sub-Layer (APS), the ZigBee Device Object (ZDO), and Application Objects (these are device specific Endpoints). Like the lower layers of the stack, the APS defines an interface for moving data between Endpoints and the lower layers of the stack, as well as a management interface to give the applications access to services like security and binding.

As illustrated in Figure 1.2, the APS supports up to 240 Application Objects to allow one physical device to support the functionality of multiple device profiles. For example, a device that has a light switch with an integrated motion sensor may implement one Endpoint for the switch and a second Endpoint for the motion sensor. This would allow them to both be integrated into the network as independent devices even if they reside within a single physical device.

The Application Framework describes the basic elements of a ZigBee Endpoint. Shown in Figure 1.3, these are: the Simple Descriptor, Clusters,

and Attributes. The Simple Descriptor identifies the endpoint, application profile, application device, and list of clusters within the device. A cluster defines a type of service with Attributes and a set of associated commands.

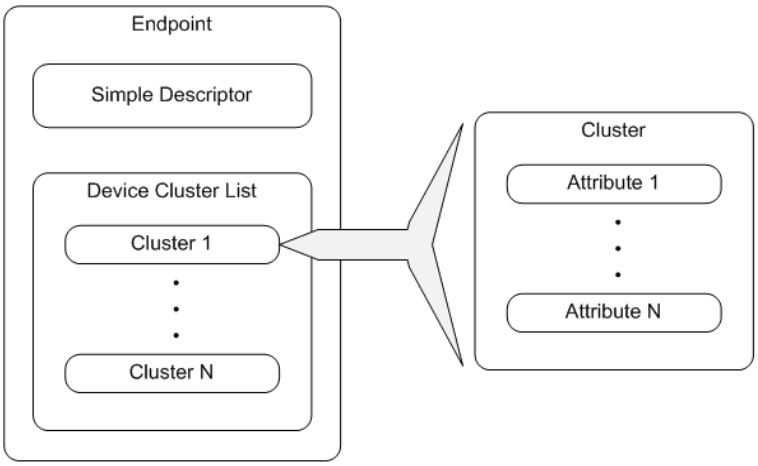

Figure 1.3: Elements of an Endpoint[1]

The ZDO is special type of Application Object that resides on Endpoint 0 of all ZigBee devices. Its purpose is to initialize the APS and NWK layers as well as assembling the device configuration to determine and implement discovery, security management, network management and binding management.

1.3 Research Objective

There are many ZigBee products on the market today and availability of new products and development tools continues to increase. Despite this, the availability of open source code and low cost development platforms is fairly limited. The intent of this research was the development of a compact frame-

work using free tools to support simple devices on the CC2530ZNP Mini Kit, a low cost development platform, that is compatible with ZigBee public profiles and the ZigBee Cluster Library (ZCL). This framework includes support for End Devices as well as a Coordinator/Concentrator for the Home Automation public profile.

Chapter 2

ZigBee Device Design

2.1 ZigBee Cluster Library and Home Automation Profile

The ZCL defines the framework for which interoperable public applications can be built. This is accomplished through definition of a set of standard clusters, cluster attributes, cluster commands, and the frame encoding that ZCL Endpoints use to relay messages. The Home Automation Profile further defines what clusters are required in an endpoint in order to implement a standard compliant device.

As with any ZigBee network, using the Home Automation profile requires a coordinator to start the network. A network may also contain a concentrator, a device that is a "system controller", enabling enhanced scheduling, configuration and other functionality.

A simple network comprised of switches and outlets may be sufficient as a standalone network with only end devices and a coordinator. However, a more complex network with climate control, lighting, and other control devices might employ a concentrator to assist with diagnostics, scheduling, and management tasks. In these more complex networks a concentrator could be implemented as a standalone device like an ethernet bridge or even part of one of the central ZigBee devices in the network like a thermostat.

While exploring options for a development platform for this research project one of the criteria was that the platform being used had to encompass both the coordinator for the network and the end device for cost reasons and to allow debug of both components in the same environment.

2.2 CC2530ZNP Mini Kit

2.2.1 Hardware overview

The hardware platform used throughout this project was Texas Instruments CC2530ZNP Mini Kit [5]. This kit includes 3 target boards, 2 battery packs intended for end devices, and an EZ430 USB Stick.

All 3 target boards are identical, containing a CC2530F256 (CC2530), MSP430F2274 (MSP430), light sensor, motion sensor, and several LEDs. The CC2530 (the Network Processor) has the IEEE 802.15.4 radio, an embedded 8051 controller with 256 kb of flash and 8 kb of RAM and handles all of the ZigBee stack except for the application endpoints. The MSP430 (the Application Processor) has 32 kb of flash and 1 kb of RAM to handle small application endpoints. Both processors have an assortment of GPIO, ADCs and other peripherals.

The EZ430 USB stick is capable of programming and debugging code on the target board's MSP430. Additionally it provides a serial link between the host PC and the target board's MSP430 through the USB interface for communicating with a terminal and/or custom applications on the host PC.

2.2.2 Software Architecture & Programming

The CC2530 comes preloaded with TI's ZigBee Network Processor (ZNP) image containing a nearly complete ZigBee stack allowing integrators and application developers to focus on the Endpoint code customized to their device. The ZNP image contains the ZDO, APL (excluding the Application Object), NWK, and MAC along with a software command interface that can be controlled via an SPI, UART or a USB interface (Note the SPI interface has additional pins that can be used to drive interrupts between the application processor and CC2530 to enable more optimized power management).

The software command interface provides the application processor a mechanism for configuration of the CC2530 specific hardware, a link to the ZDO and to the APL. It is on this foundation that a ZigBee Device can be built.

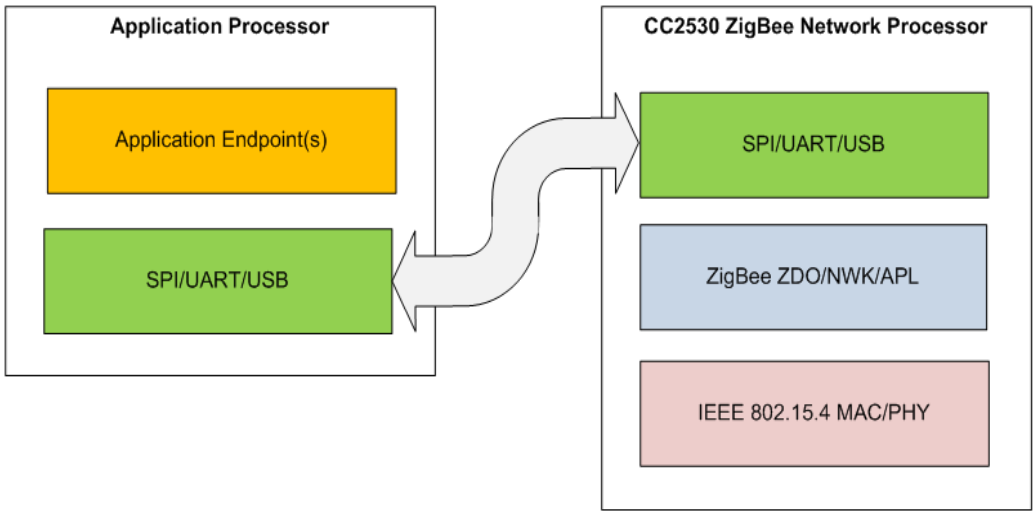

Figure 2.1: CC2530ZNP Mini Kit Component Partitioning[5]

For many applications, the CC2530 ZNP image can be used without modification - with custom applications being developed on the MSP430 application processor. There are at least two choices of compilers for the MSP430 - much of the example code that is provided by TI to demonstrate their hardware is developed in IAR Systems's Embedded Workbench. Alternatively a project called mspgcc, a GNU GCC port to the MSP430 processor, is also available. For this project we used both the IAR System's compiler in the early stages of development and the mspgcc compiler to support the code size above 4 kb.

Chapter 3

ZCL End Device Framework

3.1 ZCL Endpoint Implementation

The major components of the end device that were implemented are shown in Figure 3.1. A partial implementation was developed on the IAR Systems compiler for feasibility, followed by a more comprehensive implementation on the mspgcc compiler. The mspgcc implementation included a port of the ZCL framework from the FreakZ Open Source ZigBee Stack [6]. This final implementation yielded a small enough footprint to fit in our device (the compiled code size requires 23kb/72% of the available flash and 277b/27% of the available RAM) with further optimizations possible.

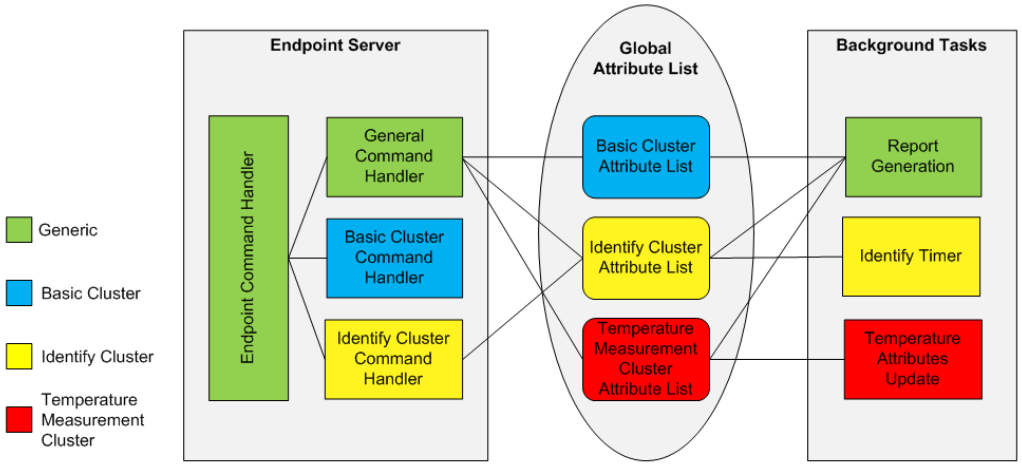

Figure 3.1: Temperature Measurement Device Endpoint.

3.2 Basic Device Setup

The basic ZigBee configuration is accomplished with the startup code in the MSP430 issuing commands through the SPI interface to the ZDO running on the CC2530. The tasks that need to be executed are: initialize the Hardware Abstraction Layer (HAL), initialize the ZNP, basic ZNP configuration, setup wireless security, register the simple descriptors, and start the application framework. These are all accomplished on the CC2530 through an API that is provided by Texas Instruments.

3.2.1 CC2530ZNP Callbacks

For low power implementations, one of the noteworthy configuration parameters on the CC2530ZNP is to set up callbacks. Callbacks configure what messages the CC2530ZNP should pass to the application processor (in our case the MSP430). There are two modes that can be used with callbacks; the default is to subscribe to individual types of messages, and the other mode is to configure the CC2530ZNP to a "direct" callback mode which passes back verbose messages without the need to subscribe. Both of our implementations use the latter configuration - this provides the application processor with more detail on incoming messages.

Since callbacks are issued from the CC2530ZNP to the MSP430 via the SPI interface, the MSP430 can be configured with an interrupt service routine to notify the MSP430 that a callback is ready to be processed. Thus the application processor can be put into a low power consumption sleep state and handle these messages with relatively low latency interrupts.

3.2.2 Message Polling

Similar to the MSP430, when configured as an End Device the CC2530 can be configured to turn off the radio and periodically wake up to check for messages. This is another big area of power savings and all that is required from an application standpoint is to configure the polling interval (as with all CC2530 parameters on this target board this is configured via the SPI interface). Since a Router or Coordinator needs to be alive to service these types of requests, they cannot take advantage of this and so typically are not battery powered.

3.3 ZCL Client/Server Model

ZigBee has setup the ZCL framework with a client/server model. In addition to the attributes that are associated with a cluster, this model defines how two Endpoints interact. Requests are initiated by a client and serviced by the server. The nature of the response, if any, is also defined by the ZCL. Commands get further broken down into two basic classes: general commands, which are common to all clusters in the library; and the cluster specific commands which are unique to the individual clusters.

General commands deal with basic interfacing to attributes - reading/writing, configuring reporting (periodic or event driven transmission of attribute data), as well as attribute discovery.

3.3.1 Command Handling

The structure of the ZCL was built in such a way to allow for the pooling of resources and command handling for efficient implementations. While it does not define an implementation, using the ZCL header can enable the code processing the incoming frames for an Endpoint to differentiate between general and cluster specific frames.

3.4 ZCL Frame Format

The ZCL defines an additional format used by ZigBee for commands passed between Endpoints (this format differs from the frames that are used by the ZDO Endpoints). The ZCL header starts the frame, followed by a command specific payload. Table 3.1 shows the ZCL header - these fields contain enough information to allow the server cluster to route the incoming message to the appropriate command handler.

Table 3.1: Format of the General ZCL Frame [1].

Bits: 8	0/16	8	8	Variable
Frame Control	Manufacturer Code	Transaction Sequence Number	Command Identifier	Frame Payload
ZCL Header				ZCL Payload

Chapter 4

ZCL Concentrator

To establish a network, we needed at minimum a coordinator to start the WPAN, and the most practical approach was to develop a concentrator that also served this role. A concentrator device can be thought of as a system controller, allowing more flexible configuration and diagnostic capabilities than a standard device. Our concentrator was developed primarily as a datalogger, but it also served to facilitate debug of the ZCL framework during the development.

4.1 Polling Datalogger

A limited ZCL implementation for the temperature sensor was developed using the kickstart version of the IAR Systems compiler for the MSP430 (this version has a 4kb code size limit). The Endpoint consisted of a command handler that only supported the read attributes command. This in turn meant that the concentrator had to periodically poll the sensor to get the updated Attribute values as illustrated by the functionality in Figure 4.1. This approach was taken to prove out the ZCL Framework using the HAL provided by TI prior to migrating the project to mspgcc.

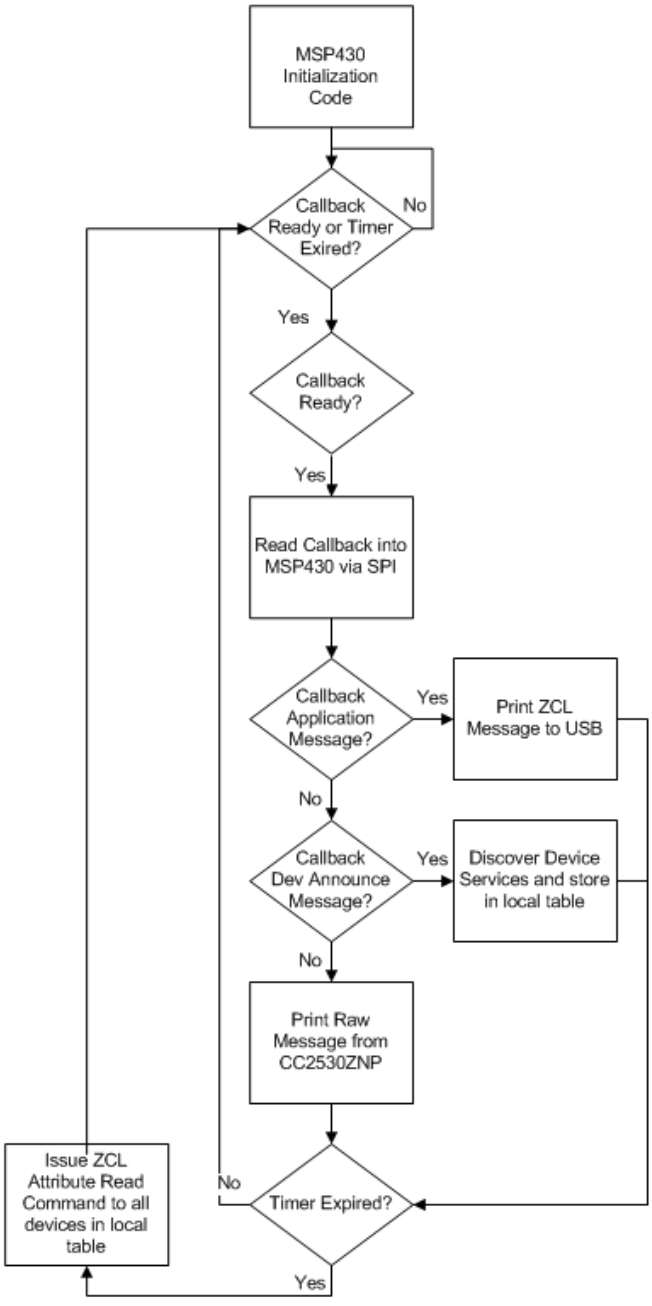

Figure 4.1: State Diagram for Polling Based Concentrator/Datalogger.

4.2 Reporting Datalogger

Once the ZCL framework was sized up and operating with a limited feature set, it was migrated to mspgcc to leverage the full 32kb of flash available to the MSP430. This final implementation of the temperature sensor supported the report functionality of the ZCL allowing the concentrator to configure the temperature sensor to send periodic updates of the sensor value. Using reporting reduces the number of messages and acknowledges exchanged with the temperature sensor and lowers power consumption.

The operation of the datalogger concentrator in this mode is shown in Figure 4.2. This configuration simplifies what the concentrator needs to do to: monitor the CC2530ZNP callbacks for a Device Announce message, discovering the services of the new device, and issuing a message to temperature sensors to send periodic reports with the temperature attribute to the datalogger.

4.3 Concentrator USB Link

To offload data from the MSP430, the concentrator is connected to a PC via the EZ430 adapter, enabling a serial link for the MSP430 to output sensor data and diagnostic information received over the ZigBee network. During trials of the datalogger Linux was used on the platform interfacing with the concentrator device. This required that the driver source code in Ubuntu Maverick for the kernel module be modified to get the driver working with the EZ430. Once this was done and the USB link was enabled, the development of a serial application was built to complete the datalogger.

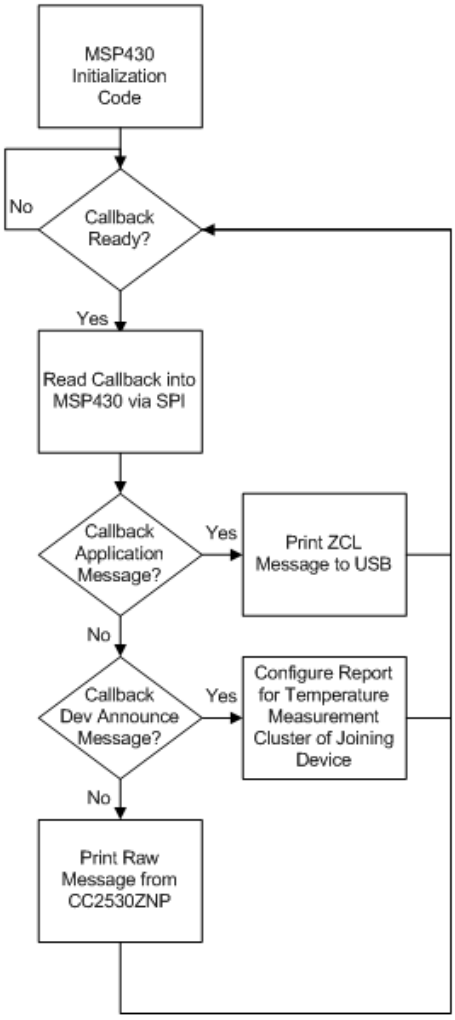

Figure 4.2: State Diagram for Report Based Concentrator/Datalogger.

4.4 Datalogger Performance

The results from the datalogger are shown in Figures 4.3 and 4.4 for the temperature sensors. Prior to this data collection intrinsic offsets were removed through a one point calibration. Noise in the temperature measurement was reduced with averaging in the temperature sensor to approximately 2 LSB of the ADC performing the measurement.

Ambient temperature variations were monitored as a forced air furnace cycled on and off to heat the space where the sensors were located. Sensor A was located near the thermostat and Sensor B was located approximately 30 feet away on an outside wall. Comparing the difference between the sensor measurements in Figure 4.3 a larger swing can be seen on Sensor B. This is expected due to its closer proximity to windows and the output register for the furnace. An unexpected spike occurred that persisted for several minutes which can be seen in Figure 4.3 shortly before 5:00pm. The source of spikes with this behavior is still being investigated.

Environmental characteristics were collected with a separate application from a the WiFi thermostat (Filtrete Model 3M-50) controlling the furnace for the space where the sensors were located, along with outdoor temperature from a local weather station via the internet. These measurements, shown in Figure 4.4, illustrate the influence that the furnace setpoint and outdoor temperature has on the indoor temperature.

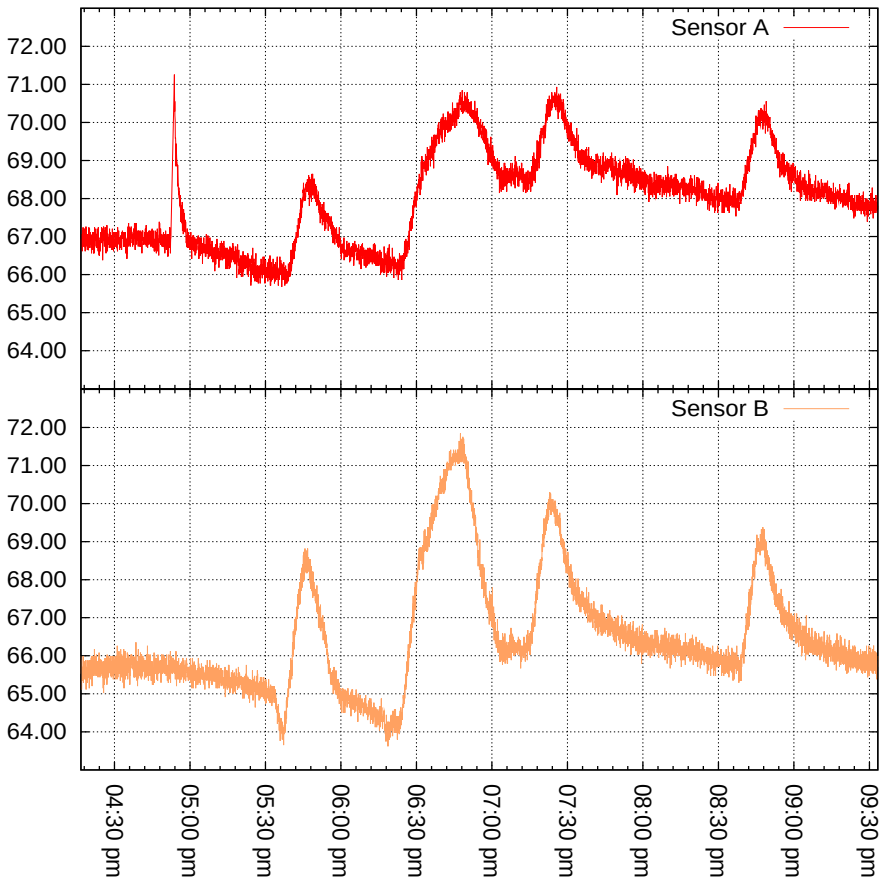

Figure 4.3: Temperature Sensor results with end device reset.

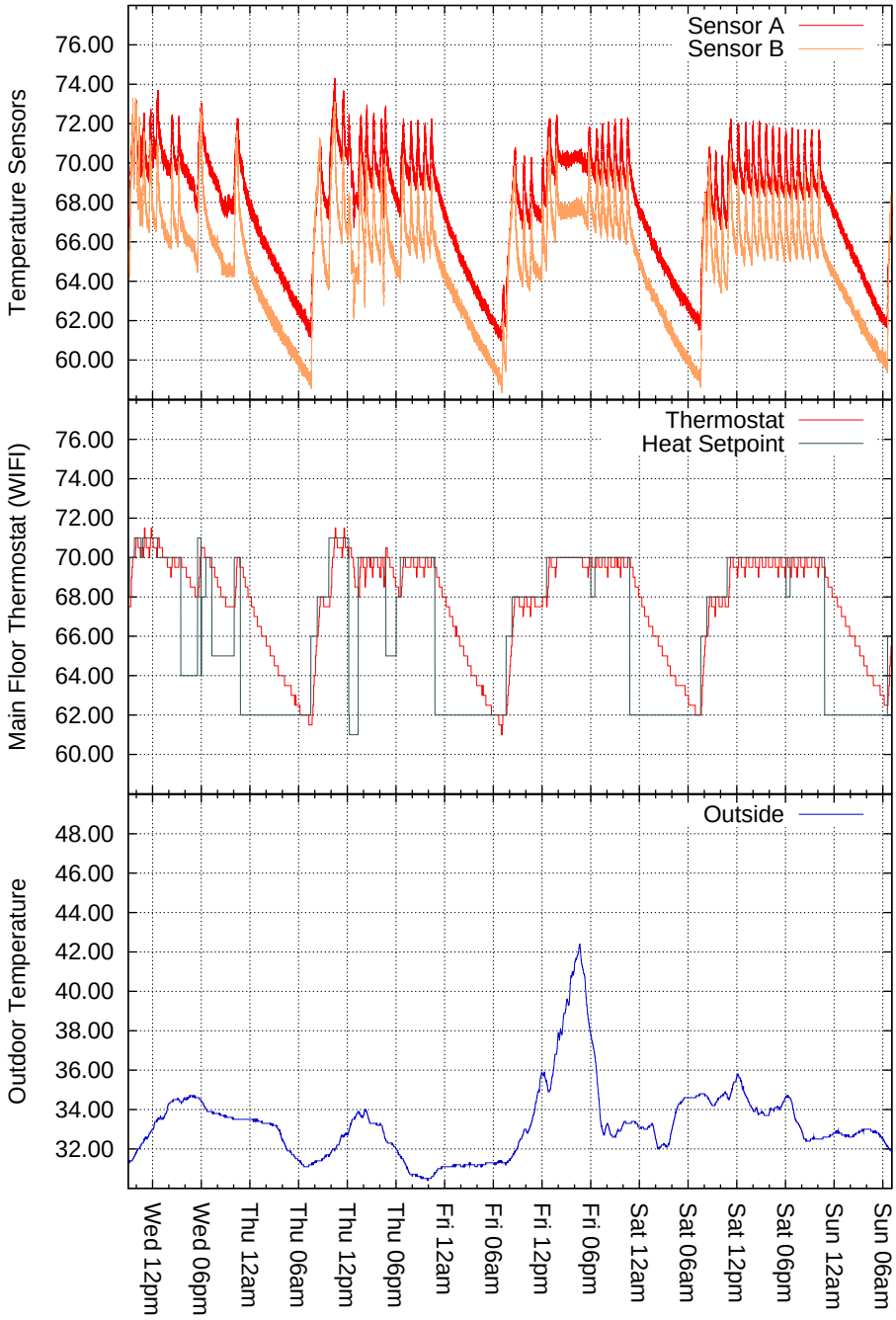

Figure 4.4: Temperature Sensor results and environment parameters.

Chapter 5

Conclusions

5.1 Summary of Key Contributions

This project has demonstrated the development and implementation of a ZigBee network supporting the Home Automation public profile on a low cost development platform using free tools. The demonstrated network contained two types of devices - a temperature sensor end device, and a datalogger concentrator/coordinator.

5.2 Future Work

While there is still space available on the MSP430 to add additional code, the implementation that was developed is too large and not structured to support multiple endpoints (i.e. a sensor and a switch) on the same device. A lot of the space consumed in the end device comes from debug capability which could be optimized to significantly reduce the code size allowing for space to support this functionality.

Bidirectional capability was implemented in the concentrator serial link; however since only a sensor end device was implemented there was only a need to log data. With support for control devices on the ZigBee network, additional functionality to control these devices via the concentrator's serial link could be developed to allow for scheduling of control tasks like turning on/off lights

or scheduling changes to a thermostat's setpoints.

Appendix

Appendix 1

Acronyms

APL Application Layer.

APS Aplication Support Sub-Layer.

HAL Hardware Abstraction Layer.

HAN Home Automation Network.

MAC Medium Access Control Layer.

NWK Network Layer.

PHY Physical Layer.

WPAN Wireless Personal Area Networks.

ZCL ZigBee Cluster Library.

ZDO ZigBee Device Object.

ZNP ZigBee Network Processor.

Bibliography

[1] Z.B. Alliance. ZigBee cluster library specification. *ZigBee Document 075123r02ZB, ZigBee Standard Organization*, 2007.

[2] Z.B. Alliance. ZigBee specification. *ZigBee Document 053474r17, ZigBee Standard Organization*, 2008.

[3] US DOE. Buildings Energy Data Book. *US Department of Energy*, 2007.

[4] IEEE. IEEE standard for information technology - Telecommunications and information exchange between systems - Local and metropolitan area networksSpecific requirements - Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). *IEEE Std 802.15.4-2006, IEEE*, 2006.

[5] Texas Instruments. Cc2530zdk-znp-mini. `http://processors.wiki.ti.com/index.php/CC2530ZDK-ZNP-MINI`, March 2011.

[6] Christopher Wang. Freakz open source zigbee stack. `http://freaklabs.org/index.php/FreakZ-Open-Source-Zigbee-Stack.html`, March 2011.

Vita

Michael A. Comai was born in Kalamazoo, Michigan on 4 April 1978, the son of Douglas Comai and Maureen F. Comai. He received the Bachelor of Science degree in Technology from Purdue University. Upon graduating, he has worked for Advanced Micro Devices in areas of design, test, and product engineering focused on high speed communication interfaces for microprocessors.

Permanent address: 528 Second Street
Traverse City, Michigan 49684

This report was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.