ZigBee Infrastructure using DIGI Gateway and TI CC2650 SensorTags

Adam Holler and **Clay McKinley**, with **Dr. Malinowski**

Senior Project 2018-2019 for Electrical Engineering

Final Report

BRADLEY University

Table of Contents

BRADLEY University

Abstract

In today's home automation market the most popular communication protocol is Bluetooth. However, a less frequently used type of communication for networks of sensors in home automation, ZigBee, has several unique benefits that Bluetooth lacks. ZigBee is based on IEEE standard 802.15.4 operating in the 2.4 GHz band and boasts low power consumption and its ability to communicate in a mesh. Mesh communication is ZigBee's main advantage over Bluetooth as it allows individual end devices to communicate between each other before forwarding data to a central coordinator. This allows for communication over longer distances as well as increased reliability. If an end device were to fail, other end devices could still communicate between each other and the coordinator. Compared to Bluetooth, ZigBee has very little documentation so research into ZigBee was critical. Using several TI 2650 Sensor Tags and a DIGI ZigBee gateway, we aimed to facilitate their connection and transmit sensor data for further use.

Problem Statement

Our goal was to research into ZigBee communication protocol as an alternative to Bluetooth. The information we looked to understand was:

- How exactly data is transmitted between ZigBee devices
- The structure of a ZigBee mesh network and the roles of each device
- What low-power capabilities were available to extend battery life of sensors
- What benefits over Bluetooth did ZigBee have and is it the better choice for our application

We kept this research in the context of a small network of sensors in a greenhouse which will collect environmental data such as temperature, humidity, sunlight and soil moisture. We needed to research sensors that would help us facilitate this as well as a ZigBee gateway that would receive the data for use by a theoretical user.

Review of Literature/Prior Work

We researched into communication protocols such as Bluetooth Low Energy & ZigBee vs Wi-Fi, 6LoWPAN. In our research, it was clear that ZigBee and Bluetooth Low Energy were top competitors for low energy systems. Below is a figure found during our research that shows average power consumption for transmitting and receiving of different communication protocols. We compared and contrasted Bluetooth Low Energy and ZigBee, finding that the former was more energy efficient for smaller wireless networks but ZigBee was more efficient for larger networks due to its mesh network capabilities. Researching ZigBee protocol gives us an opportunity to learn about a communication protocol that is not as widely used so in the future we can make decisions about which is best in what situation.

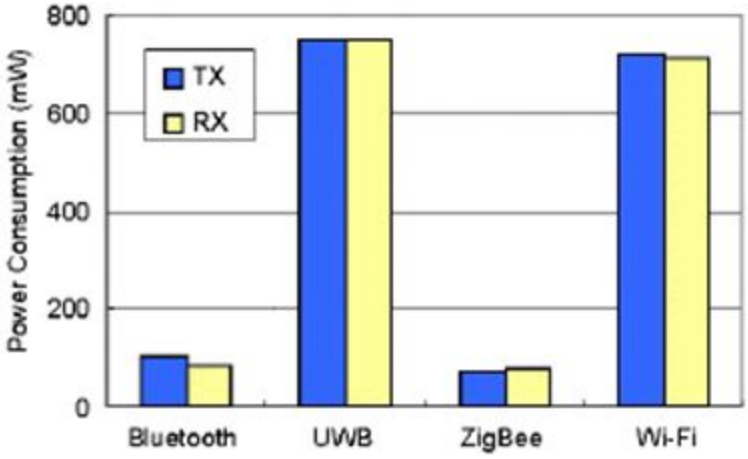


Figure1: Communication protocol power consumption

ZigBee

ZigBee was released before Bluetooth Low Energy as a way to link hundreds of devices wirelessly onto a single network. As described in [7] ZigBee has two implementation options, ZigBee and ZigBee Pro. ZigBee is for smaller networks where ZigBee Pro, the more popular version, is capable of linking up to 64,000 devices of a wider variety onto a single network.

ZigBee operates on the same 2.4 gigahertz frequency as Bluetooth which is split into 16 channels for communication. In a ZigBee system there are 3 categories of nodes: Coordinator, router, and end devices. Courtesy of [2], a possible organization of these devices is seen in figure 1.

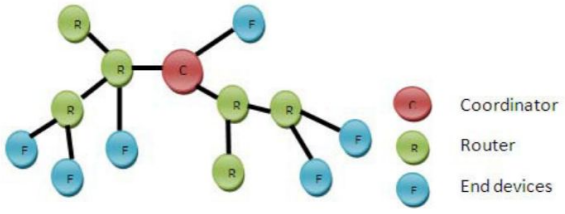


Figure1: ZigBee Network

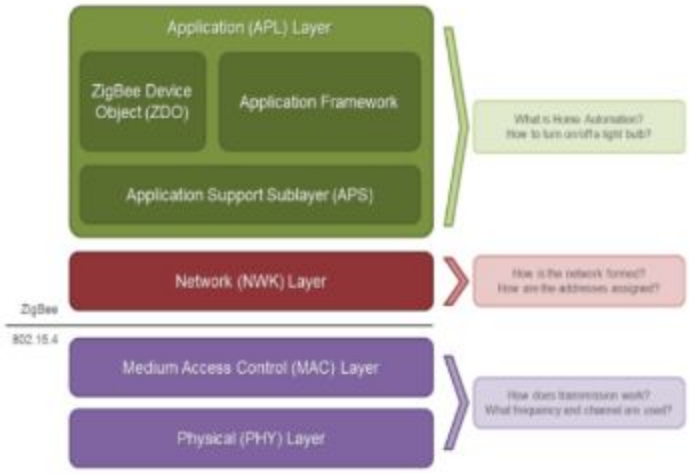
- **The coordinator** acts as the root of the network. It is responsible for determining things like the frequency channel of communication and possibly communicating with other networks. Each ZigBee network must have exactly one coordinator.

- **Routers** act as bridges that communicate that relay data between routers and end devices or to coordinators.

- **End devices** are sensors that are potentially battery powered who only have enough functionality to transmit data to either routers or coordinators and cannot receive any data. This allows for the end devices to be cheaper and have a much longer battery life as they can stay in a sleep state for significant periods of time.

There are also two physical types on devices with ZigBee detailed in [8]. There is a Full-function device (FFD) and a Reduced-function device (RFD). FFDs can talk to RFDs or other FFDs. Where a RFD can only talk to FFDs. RFDs are used in very simple applications such as a light switch where the device only has to send a simple signal.

ZigBee Stack Library



Most network protocols use the concept of layers to separate different components and functions into independent modules that can be assembled in different ways. Zigbee is built on the Physical (PHY) layer and Medium Access Control (MAC) sub-layer defined in the IEEE 802.15.4 standard. There are other standards on the IEEE 802.15.4, such as WirelessHART and MiWi. These layers handle low-level network operations such as addressing and message transmission/reception. The Zigbee specification defines the Network (NWK) layer and the framework for the application (APL) layer. The Network layer takes care of the network structure, routing, and security. The application layer framework consists of the Application Support sub-layer (APS), the Zigbee device objects (ZDO) and user-defined applications that give the device its specific functionality.

Physical Layer

Defines the physical operation of the Zigbee device including receive sensitivity, channel rejection, output power, number of channels, chip modulation, and transmission rate specifications. Most Zigbee applications operate on the 2.4 GHz ISM band at a 250 kb/s data rate. See the IEEE 802.15.4 specification for details.

MAC Layer

Manages RF data transactions between neighboring devices (point to point). The MAC includes services such as transmission retry and acknowledgment management, and collision avoidance techniques (CSMA-CA).

Network Layer

Adds routing capabilities that allows RF data packets to traverse multiple devices (multiple hops) to route data from source to destination (peer to peer).

Application Support Sub-layer (Application Framework)

Application layer that defines various addressing objects including profiles, clusters, and endpoints. We would have got to editing this.

ZigBee Device Objects

Application layer that provides device and service discovery features and advanced network management capabilities.

ZigBee PAN ID

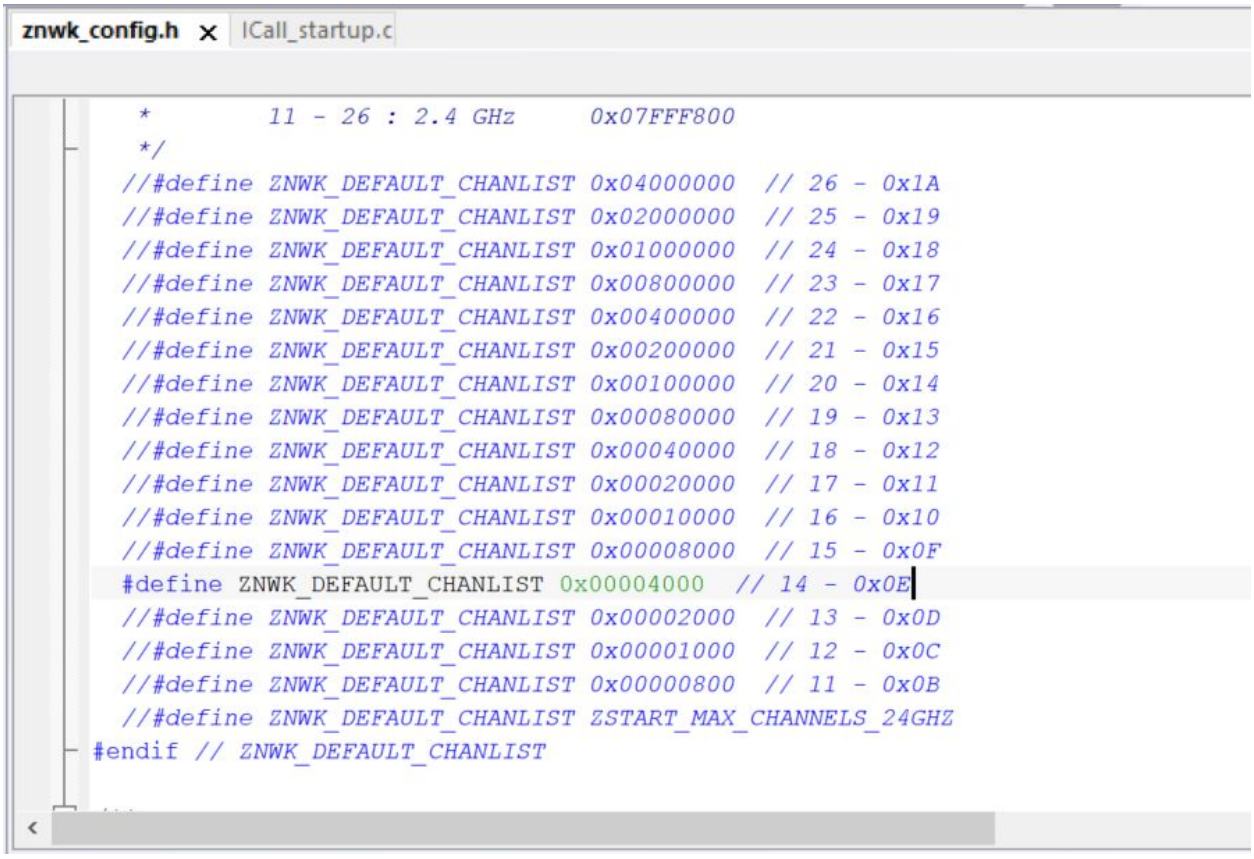
The 16-bit PAN ID is used as a MAC layer addressing field in all RF data transmissions between devices in a network. However, due to the limited addressing space of the 16-bit PAN ID (65,535 possibilities), there is a possibility that multiple Zigbee networks (within range of each other) could use the same 16-bit PAN ID. To resolve potential 16-bit PAN ID conflicts, the Zigbee Alliance created a 64-bit PAN ID.

The 64-bit PAN ID (also called the extended PAN ID), is intended to be a unique, non-duplicated value. When a coordinator starts a network, it can either start a network on a preconfigured 64-bit PAN ID, or it can select a random 64-bit PAN ID. Devices use a 64-bit PAN ID during joining; if a device has a preconfigured 64-bit PAN ID, it will only join a network with the same 64-bit PAN ID. Otherwise, a device could join any detected PAN and inherit the PAN ID from the network when it joins. All Zigbee beacons include the 64-bit PAN ID and is used in 16- bit PAN ID conflict resolution.

BRADLEY University

ZigBee Operating Channels

Zigbee uses direct-sequence spread spectrum modulation and operates on a fixed channel. The 802.15.4 PHY defines 16 operating channels (channels 11 to 26) in the 2.4 GHz frequency band. Below is the TI SensorTag file that shows the channels.



Product Research

With the application of a series of sensors for a greenhouse in mind, we began looking for suitable devices. There were more familiar microcontrollers such as the ESP286, but they did not have as many on-board sensors as well as not supporting ZigBee. The best candidate we found was the TI SensorTag which offered an impressive number of built in sensors in addition to having ZigBee support. The TI SensorTags needed a dedicated physical debugger to program directly which TI SensorTag DevPacks would be able to do. In addition to the SensorTags themselves and their debuggers, ZigBee also required a ZigBee to internet specific gateway. This gateway would act as a ZigBee coordinator and make data collected from the Sensortags available for further use. Many common IoT bridges such as Amazon's Alexa is actually a Zigbee/Ethernet gateway, though we were looking for a simpler, dedicated device. The best option at the time seemed to be a XBee gateway by DIGI. The DIGI XBee gateway was advertised as being compatible with any ZigBee Pro device. XBee modules are DIGI brand specific radios which have ZigBee capabilities. Additionally we thought having a soil moisture sensor would be a good device for data collection as they were lost cost on Amazon.

BRADLEY University

Parts Ordered

4x	BLUETOOTH SENSOR TAG	296-38831-ND
3x	DEBUGGER FOR SENSORTAG	296-42039-ND
1x	Networking Modules XBee Gateway ZigBee to Ethernet Intl	888-X2E-Z3C-E1-W
1x	Capacitive soil moisture sensor	

Specifications

TI Sensor Tag (CC2650STK wireless MCU)

- 10 low-power sensors, including ambient light, digital microphone, magnetic sensor, humidity, pressure, accelerometer, gyroscope, magnetometer, object temperature and ambient temperature.
- Ultra low power, coin cell battery, ARM Cortex-M3.
- Uses Zigbee or 6LoWPAN.

Capacitive Soil Moisture Sensor v1.2

- 5V, Analog readings 2.4-4.4V

Digi XBee® Gateway

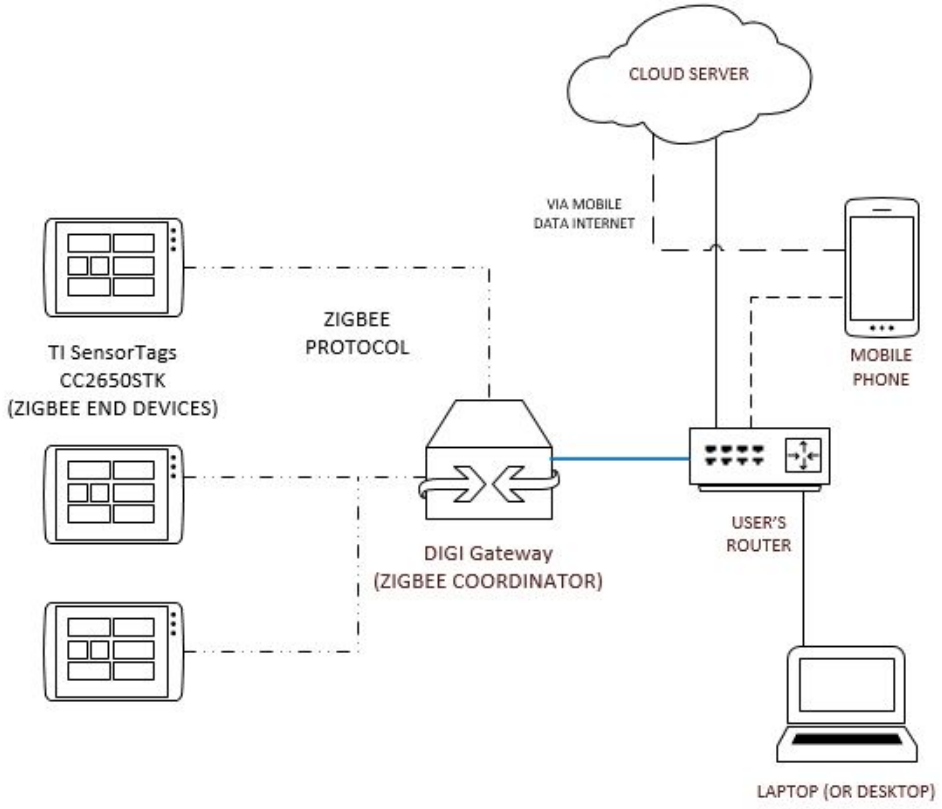
- Protocols: UDP/TCP, DHCP. Security: SSL tunnels, WEP-40, WEP-104, WPA/WPA2, Authentication with PSK and EAP
- OS: Digi Embedded Linux

SimpleLink SensorTag Debugger DevPack

- Small form-factor XDS110 debugger

Block Diagram

April 11, 2019	Zigbee Plant Monitoring Setup



Troubleshooting

The first thing we tested was the capacitive moisture sensor. It does not have a data sheet so we have hooked up the sensor to the power supply and oscilloscope. It needs a supply of around 5V and the analog output is between 2.4-4.2V.

We initially booted up the TI SensorTags we ordered and used the TI Smart Tag App to rename the devices and see how they operate. After reading through a lot of documentation, we have arrived on using Code Compiler Studio. We had some trouble installing the application. We also figured out how to use the TI DevPack and hook up to the computer via Micro USB cable. Using the BLE Device Manager, we got marginal success with connecting to the device. We still are researching how to program the device, with the ultimate goal of configuring GPIO pin for the moisture sensor.

We also have read through documentation for Digi Gateway, it did come with an ethernet cable which is nice.

In the second half of the semester we started hunting for the Zigbee firmware on the TI SensorTags. This is where the crux of the project turned into figuring out Zigbee. The Mobile Phone App for TI SensorTags had a firmware page, and it seemed simple enough. Just click on the firmware you wanted to upgrade to. There was firmware for Zigbee as an option, however, when selected the app proceeded to do nothing. No response. We figured it was bugged. Now it was on to manually flashing Zigbee

firmware to the SensorTags, which seems simple enough however finding the firmware was a hassle.

To start, there was an emphasis on Zigbee 3.0, what seemed to be Zigbee's latest development on its networking. So when we first started searching for Zigbee firmware we were directed to the Zigbee 3.0 Stack for CC2650. However, the chip on the SensorTag is a CC2640, even though the part number for the SensorTag is CC2650STK. This was a point a confusion, because the product description said the SensorTag was Zigbee compatible, however every time we tried to look up zigbee firmware we were directed to the Zigbee 3.0 Stack website.

It wasn't until a few weeks when later we happen to find an older link which took us to the ZStack 1.22a Home Automation Library where our next breakthrough was. After installing that stack, we had to dig but eventually we found there was an SensorTag example. The issue was, the example used IAR Embedded Workbench, instead of Code Compiler Studio to compile. And this caused some issues as explained below.

After temporarily solving the issues with the new compiler, it was time to start taking a look at the DIGI gateway. We had a router put in the lab where we could hook it up to the internet and we registered the device online so we could monitor it. Eventually we found we configure base settings in the Web Interface or simply a SSH session with login. Clay did find some information on the python script used to program the gateway but in the end it was easier and simpler to change network settings through this interface.

The final phase was trying to run the example program on the SensorTag and get the DIGI gateway to find it. We of course did adjust settings in the sample program. Such as the PAN ID, channel settings and EPID. From our intensive forum searching and documentation hunting these were the primary settings we had to change, since the SensorTag was already setup for Zigbee Home Automation Standard. So the next step was to configure the gateway to Zigbee Home Automation Standard. To this day we setup everything to that standard but we never could the gateway to find the SensorTag.

Complications and the Emphasis on Zigbee

Compilers and Programs

We used and experimented with several programs while trying to find the best method for utilizing ZigBee. These are listed below, all of which we installed on our own machines.

BLE Device Monitor

This was one of the first programs we found, which will detect TI brand Bluetooth devices. We hoped we would be able to alter setting of the TI sensor tags as they were Bluetooth by standard but the program did not have the functionality we were looking for.

TI SensorTag App

A mobile app developed by TI specifically for SensorTags. This app allowed us to read sensor data, change sampling rate, rename specific sensor tags and among other capabilities, it allowed us to download the ZigBee stack which would enable ZigBee capability. However, this capability does not work and no fix was released by TI.

Smart RF Studio 7

This program monitors radio frequency communication on TI SensorTags, unfortunately, its functionality does not extend to SensorTags using ZigBee.

Flash Programmer 2

Flash Programmer 2 is a programmer which could load .hex and .out files to our SensorTags which were generated by the different IDEs discussed below. The files were downloaded onto the SensorTags themselves via a USB connection to the attached debugger.

Code Composer Studio

In terms of ease of use, this IDE for the SensorTags is by far the best. Aside from creating a few small programs to test the SensorTags capabilities, its usefulness was completely negated by the fact that ZigBee functionality was exclusive to IAR Embedded Workbench described below.

IAR Embedded Workbench

This is the main compiler and debugger which is used for the Z-stack project for the TI SensorTags. This IDE specifically was required as it was the only one which could handle ZigBee protocol. The sample programs we were dealing with went as far as to check if the programmer was using IAR Embedded Workbench or it would

immediately stop running. This program does require a license which we were able to get with our student emails for 30 days at a time. Unfortunately this program was plagued with constant bugs which would require anything from restarting to reinstalling the software.

XCTU

XCTU is a program for DIGI products which would monitor ZigBee communication and allow configuration of device settings. Unfortunately, it required a serial connection which our DIGI gateway did not have.

DIGI Device Cloud

This web application is a handy tool that allowed a DIGI gateway which was connected to the internet to be remotely monitored and configured. This also allowed us to download the source python code that the DIGI Gateway was running and reupload any modified version.

PyCharm 2018.3.5

This program was our choice for modifying the python code that the DIGI gateway was running.

DIGI Gateway Web Interface

This web interface that was accessed by connecting directly to the IP of the Gateway allowed for monitoring the entire ZigBee network. The interface also allowed for changing of several settings to match with the settings we put on the TI SensorTags.

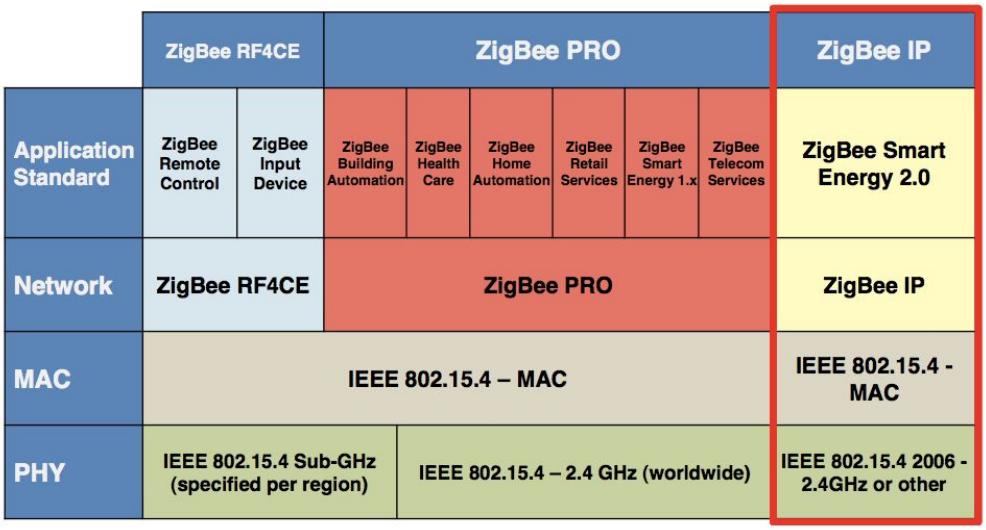
DIGI Xbee - T.I. Compatibility

There are some differences between the similar looking names to be aware of. First off, Xbee refers to a family of devices from Digi that share form factor, host interface and a group of protocols you can select from (Zigbee being one of these). Zigbee, on the other hand, is a mesh networking protocol built upon the 802.15.4 IEEE standard. So Zigbee protocol dictates how devices can communicate wirelessly and are one of the supported protocols of the Xbee products. Zigbee also has their line of goods too; however, they only support Zigbee communication protocols.

This is the main point of conflict in this project. DIGI devices have a standard for easy connection between their ZigBee devices called Xbee. When we were configuring the DIGI coordinator/router to accept non-Xbee devices, we had to set up the gateway using the Zigbee Home Automation standard.

Zigbee Home Automation Standard

There are different Zigbee standards:



The version we use is Zigbee Home Automation, which uses these standards:

5.3.1 Start Up Parameters

Short Address: 0xFFFF
E PANiD: 0x0000000000000000
PAN ID: 0xFFFF

Channel Mask
All channels in frequency band. If needed, the power transmitted by the device on channel 26 can be lowered to comply with FCC regulations.

Protocol Version
0x02 (ZigBee Specification revision 17 (2007) and later).

Stack Profile
2 (ZigBee PRO Feature Set).

Startup Control
3 (three) if un-commissioned, so it will join network by association when join command is indicated by button press sequence.

0 (Zero) if commissioned. Indicates that the device should consider itself a part of the network indicated by the *ExtendedPANId* attribute. In this case it will not perform any explicit join or rejoin operation.

Trust Center Address
0x0000000000000000

Master Key
NULL

Network Key
NULL.

Default Trust Center Link Key
0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39

Note: *The Link Key is listed in little-endian format.*

Use Default Link Key Join
0x01 (True). This flag enables the use of default link key join as a fallback case at startup time.

The SensorTag Zigbee example is set up already for Zigbee Home Automation Standard. However, the DIGI gateway was the tricky part. That had to have specific settings for this standard. Here is what DIGI has on their website:

Zigbee Home Automation

When developing a Zigbee Home Automation application or Wanting to talk with a Zigbee Home Automation Device, the XBee module you should select is the XBee ZB SMT module or XBee ZB (Zigbee) Surface Mount module.

Using our XCTU software or some other Terminal Emulator or processor, the following settings within the XBee ZB SMT module would need to be set:

ZS 2
AP 1
AO 3
EE 1
EO 2
KY 0x5A6967426565416C6C69616E63653039

Zigbee Cluster Library

We would have looked further into this library if we have had the time. This is how ZigBee is a standard when dealing with clusters and binding.

Timeline/Division of Labor

Timeline

There was always research being done. So in general are listed the major goals of each month:

- Fall Semester - Researched several products for project and ZigBee protocols and gateways.
    - October - We acquired the first set of components
- February - Focused on finding firmware (ZigBee Stack) for TI SensorTag
- March - Setup IAR Embedded Workbench, DIGI gateway, and started troubleshooting connection between gateway and SensorTag
- April - Continued ZigBee/XBee troubleshooting and documentation
- May - Finished final paper and Final presentation

Division of Labor

We typically combined effort by following what is known as agile pair programming. One of us would act as the ‘driver’ who wrote code or changed ZigBee settings.The other would be the ‘navigator’ who checked changes and kept the overall goal of these changes in mind. These roles were switched often. When we did have a true split it would typically be split into one person programming sensortag and other

programming the DIGI gateway.

Conclusions

We figured out that the use of DIGI's XBee is situational to where only DIGI devices will be interfaced with. Some companies have their own 'brand' of ZigBee with questionable cross compatibility. So beware inaccurate advertising. Going forward we recommend TI's CC1350 for a ZigBee Router or simply just use XBee devices with the DIGI gateway.

BRADLEY University

Appendix A:
Power Efficiency for Small Electronics: Comparing Bluetooth Low Energy and ZigBee

Abstract- Power efficiency for small wireless networks is vital for extending the battery life of a network. One way to extend the life time of these electronics is by altering how it communicates. There are two dominant communication protocols that have low-power in mind, Bluetooth and ZigBee. Both of these protocols will be analyzed in how they function and how they compare to other communication protocols. They will then be compared against each other in their ability to use the least amount of power possible. Each protocol's ability to reduce power without sacrificing significant performance will be key in determining which protocol is superior for low-power. These factors and more will be thoroughly discussed in the following report.

Clay McKinley
November 7 2018

Power Efficiency for Small Electronics: Comparing Bluetooth Low Energy and ZigBee

Clay McKinley, in association with Bradley University

Abstract

Power efficiency for small wireless networks is vital for extending the battery life of a network. One way to extend the life time of these electronics is by altering how it communicates. There are two dominant communication protocols that have low-power in mind, Bluetooth and ZigBee. Both of these protocols will be analyzed in how they function and how they compare to other communication protocols. They will then be compared against each other in their ability to use the least amount of power possible. Each protocol's ability to reduce power without sacrificing significant performance will be key in determining which protocol is superior for low-power. These factors and more will be thoroughly discussed in the following report.

Introduction

The two protocols which will be discussed, ZigBee and Bluetooth, dominate the market for low power wireless sensor network communication. The Bluetooth discussed will be Bluetooth 4 also known as Bluetooth Low Energy. The iteration of ZigBee discussed is IEEE 802.15.4. As explained in [5], Both of these protocols do their job well and are widely used, however they do not do exactly the same thing. Bluetooth Low Energy is designed as a one to one communication protocol as explained in [6]. This means that the layout of a wireless sensor network must change to meet the needs of a wireless sensor network. Zigbee protocol is capable of connecting far more sensors than bluetooth. So, in order to compare these protocols we will compare their effectiveness when used in small wireless sensor networks of 5 or less nodes.

Bluetooth

Bluetooth was released as a machine to machine protocol for reliable short distance communication. Today it still fits this definition but has been improved to be far more energy efficient. This has made it a strong contender against protocols like ZigBee for small wireless sensor communication. The main difference between standard Bluetooth and Bluetooth Low Energy (BLE) is that BLE makes a tradeoff in response time in order to increase power efficiency.

As described in [11], Bluetooth Low Energy has several theoretical specifications. A BLE device on a coin cell battery has a lifetime between 2 days and 14.1 years depending on what exactly the device is doing. BLE works off of what is known as a master and slave design where for each master(the device requesting data, which can be linked to several devices), BLE can support between 1 and 5,917 slaves(the devices providing data, which can only be linked to one master). The minimum time for a master to obtain a sensor reading from a slave is 676 microseconds. BLE devices are non-compatible with standard Bluetooth devices. BLE operates in 2.4 gigahertz frequency band and defines 40 radio frequency channels that are 2 megahertz apart. Three of these 40 channels are defined as advertising channels which are used to link new connections before they communicate on one of the other channels called data channels. To save energy the slaves are put in a low-power sleep mode by default and occasionally wake up to listen to requests. The master determines how often the slaves wake up to listen as well as other information such as what frequency channels to transmit on. So, every time a slave device wakes up, the master device makes a request which the slave receives, send the requested data, then re-enters sleep mode.

ZigBee

ZigBee was released before Bluetooth Low Energy as a way to link hundreds of devices wirelessly onto a single network. As described in [7] ZigBee has two implementation options, ZigBee and ZigBee Pro. ZigBee is for smaller networks where ZigBee Pro, the more popular version, is capable of linking up to 64,000 devices of a wider variety onto a single network.

ZigBee operates on the same 2.4 gigahertz frequency as Bluetooth which is split into 16 channels for communication. In a ZigBee system there are 3 categories of nodes: Coordinator, router, and end devices. Courtesy of [2], a possible organization of these devices is seen in

figure 1.

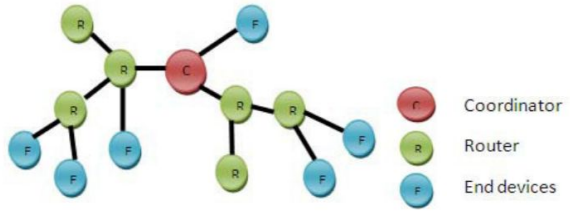


Figure1: ZigBee Network

- **The coordinator** acts as the root of the network. It is responsible for determining things like the frequency channel of communication and possibly communicating with other networks. Each ZigBee network must have exactly one coordinator.

- **Routers** act as bridges that communicate that relay data between routers and end devices or to coordinators.

- **End devices** are sensors that are potentially battery powered who only have enough functionality to transmit data to either routers or coordinators and cannot receive any data. This allows for the end devices to be cheaper and have a much longer battery life as they can stay in a sleep state for significant periods of time.

There are also two physical types on devices with ZigBee detailed in [8]. There is a Full-function device (FFD) and a Reduced-function device (RFD). FFDs can talk to RFDs or other FFDs. Where a RFD can only talk to FFDs. RFDs are used in very simple applications such as a light switch where the device only has to send a simple signal.

Power Consumption

Both ZigBee and BLE were designed with power efficiency in mind. They both far outdo other protocols for wireless sensor networks. This is illustrated in figure 2 which is seen in [3] where power consumption was averaged across several devices using ZigBee, Bluetooth, as well as WiFi and UWB, which are other popular communication protocols as comparison. These devices were measured in transmit mode(TX) as well as receive mode(RX).

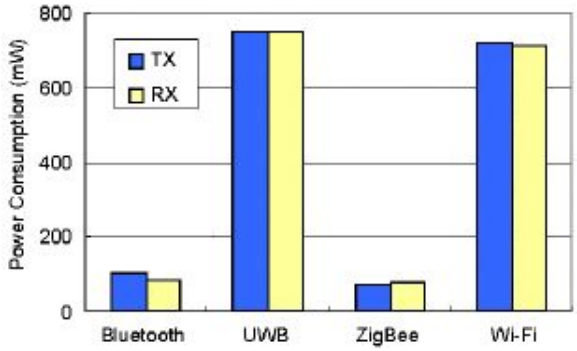


Figure2: Power Consumption

As can be seen in Figure 2 above, Bluetooth and ZigBee consume far less energy and are far superior to UWB and WiFi in terms of power efficiency. ZigBee and Bluetooth, however, were too close in power consumption and measured on too few devices to determine which one was more efficient.

In order to properly test BLE and ZigBee, they must be used equally in a state that prioritizes power efficiency. As previously mentioned, both of these protocols can utilize a sleep mode where sensors or end devices can enter a low power state. As pointed out in [10], while in sleep mode, these devices cannot transmit or receive data. In order to save energy these end devices are put on a set timer where they will wake up out of sleep mode, transmit data it has, then re-enters sleep mode. The length of this sleep interval is determined by the master in bluetooth and either by the main coordinator or individually on each end device in zigbee. So, a test was done by [4] where a single sensor node is in a sleep cycle and will transmit to some receiver. Both devices were on a 3.3 Volt power supply, which is standard for these devices, and was set to sleep for 120 seconds before waking up to transmit data. The power consumption in each was measured in microamps. The test found that Bluetooth Low Energy had the lowest power consumption of 10.1 microamps and ZigBee fell behind at 15.7 microamps.

This test is reinforced by the findings of [1] which goes more in depth by measuring the watts used in each phase of communication. The total power consumed by BLE was 234 microwatts compared to the total power consumed by ZigBee which was 356 microwatts. With each phase of communication added up, the findings were very similar to that found in [4].

Conclusion

Overall, both Bluetooth and ZigBee accomplish what they were made to do. They consume significantly less power than other protocols such as WiFi and UWB. However, when it came down to the power consumption of a single node in a sleep cycle, Bluetooth Low Energy was superior. The difference between them was large enough to be a major determining factor in which protocol to use in wireless sensor networks.

References

[1] M. Siekkinen, M. Hiienkari, J. K. Nurminen and J. Nieminen, "How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4," *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Paris, 2012, pp. 232-237.

[2] N. A. Somani and Y. Patel, "ZigBee: a low power wireless technology for industrial applications," *International Journal of Control Theory and Computer Modelling,* Vol. 2, No. 3, May 2012.

[3] J. Lee, Y. Su and C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, Taipei, 2007, pp. 46-51.

[4] A. Dementyev, S. Hodges, S. Taylor and J. Smith, "Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario," *2013 IEEE International Wireless Symposium (IWS)*, Beijing, 2013, pp. 1-4.

[5] J. Lee, M. Dong and Y. Sun, "A preliminary study of low power wireless technologies: ZigBee and Bluetooth Low Energy," *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, Auckland, 2015, pp. 135-139.

[6] N. Baker, "ZigBee and Bluetooth strengths and weaknesses for industrial applications," in *Computing & Control Engineering Journal*, vol. 16, no. 2, pp. 20-25, April-May 2005.

[7] E. Antonopoulos, K. Kosmatopoulos and T. Laopoulos, "Reducing power consumption in pseudo-ZigBee sensor networks," *2009 IEEE Instrumentation and Measurement Technology Conference*, Singapore, 2009, pp. 300-304.

[8] A. A. Essa, Xuan Zhang, Peiqiao Wu and A. Abuzneid, "ZigBee network using low power techniques and modified LEACH protocol," *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, Farmingdale, NY, 2017, pp. 1-5.

[9] Itsuki Tanabe, Hiroshi Sasaki and Li Zheng, "An ultra low power ZigBee module - AA batteries with life expectancy of 10 or more years!," *2008 5th International Conference on Networked Sensing Systems*, Kanazawa, 2008, pp. 245-245.

[10] Y. Qianjun, W. Guichu, Y. Fenqun and X. Hongyan, "Design and applications of intelligent low voltage distribution system based on ZigBee," *2011 International Conference on Electronics, Communications and Control (ICECC)*, Ningbo, 2011, pp. 791-794.

[11] C. Gomez, J. Oller and J. Paradells. "Overview and evaluation of bluetooth low energy: an emerging low-power wireless technology," *Sensors 2012,* Vol. 12, No. 9, pp. 11734-11753, August 2012.