Investigation into the Development of a Wireless IoT Penetration Testbed

Tellrell White
Morgan State University
Baltimore, Maryland, 21251
tewhi1@morgan.edu

Faculty Advisor: Willie L. Thompson II

ABSTRACT

IoT protocols have been proposed to replace wired systems in aircraft to support telemetry applications. They offer several advantages to wired systems due to them being wireless, low cost, and consuming less power. However, the one consideration that is often overlooked is the security of these wireless protocols. This project focused on investigating the use of open source hardware and software frameworks to create a wireless testbed to conduct penetration testing of the ZigBee protocol. To accomplish this task, the open source XBee software library was used to implement the ZigBee Network and Application Layers within the GNU Radio IDE. The XBee hardware module was leveraged for the IEEE 802.15.4 PHY and MAC Layers.

INTRODUCTION

Historically, aircraft have mostly been made up of wired systems which have required major installation, cost, power, and time. IoT protocols, on the other hand, are increasingly being adopted in aircraft due to them being low power, wireless, and low cost. An example of this is the F-35 Lightning II aircraft which uses on board sensors to monitor its performance, maintenance, and to notify the military of its surroundings [1]. Despite the benefits of IoT protocols, one vital consideration is their security. For example, what if the wireless network used on the F-35 was compromised? An attacker could potentially extract sensitive information pertaining to the performance, maintenance, etc. of the aircraft. An illustration of this scenario is shown in Figure 1. The Figure shows an aircraft containing a ZigBee network consisting of a coordinator and sensors(end users) which monitor the aircraft's engines and send collected data back to the coordinator. The role of the coordinator is to start the network and join devices while the role of end users is to send and receive data. A hub connected to the coordinator then sends this data to a ground station. In place of an attacker in the Figure is a wireless penetration test bed whose goal is to figure out the vulnerabilities of the ZigBee protocol at each layer of the protocol stack. This project focuses on an investigation into the development of this wireless IoT penetration testbed shown in the Figure using the open source XBee software library and the XBee RF module

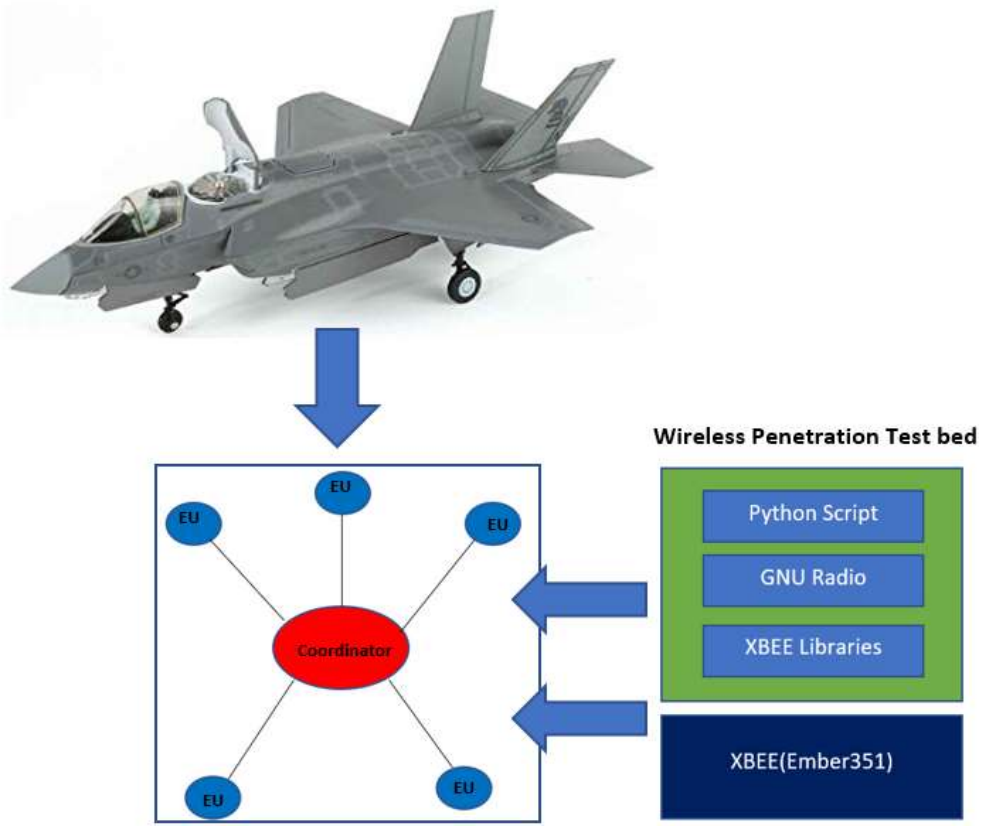

Figure 1: Use Case for IoT Wireless Testbed

The XBee API library and XBee RF module were leveraged for this research based on initial investigations into opensource ZigBee implementations. The XBee framework consists of hardware modules and a software library for the IEEE 802.15.4 standard and ZigBee protocol. To reduce development time, the XBee software library was utilized for the ZigBee Network Layer (NWK) and Application Layer (APL) and the XBee RF module was used for the IEEE 802.15.4 PHY/MAC layers.

This paper is organized into four sections. Section 1 discusses the background of this research. It includes a brief discussion on each layer of the protocol as well as some of the vulnerabilities associated with ZigBee. Section 2 explains some of the vulnerabilities associated with the ZigBee protocol. Section 3 discusses the system model for the wireless testbed. It includes a description of the XBee RF module and the XBee API library. Section 4 provides a discussion of the design implementation for this research. Section 5 includes the conclusion and a discussion of future work.

BACKGROUND

Zigbee is a low power, low cost, wireless protocol popular in the IoT community. It is commonly used in industries such as lighting, home automation, and energy. The communication stack for the Zigbee protocol is shown in Figure 2. The model shows four (4) layers: An Application Layer, a Network Layer, a Medium Access Control Layer, and a Physical Layer. The Zigbee

protocol is like the TCP/IP model. Zigbee is a layer 3 (Network Layer) and up protocol. The upper two layers of the protocol are defined by the Zigbee Alliance, while the lower two layers are defined by the IEEE 802.15.4 standard. In this research, the EM351 System-on-Chip (SoC) contained in the XBee module was used to implement the MAC and PHY layers, while the XBee software library was used to implement the upper layer functionality.

The Application Layer of the ZigBee protocol contains the Application framework, ZigBee device object (ZDO), and the Application support sub-layer (APS). The APS is responsible for filtering packets for end devices, checking for repeat packets, and maintaining binding tables, which is the connection between the endpoint on a node to one or more endpoints on other nodes. The Application framework provides two types of data services as key-value pair and generic message services. A generic message is a developer defined structure, whereas the key-value pair is used for getting attributes within the Application objects. The ZDO provides an interface between Application objects and the APS layer in Zigbee devices. The ZDO is responsible for detecting, initiating and binding other devices to the Network [2].

The Network Layer, located between the Application Layer and the MAC Layer, has several responsibilities. Namely, starting the Network, managing end devices joining and leaving the Network, and neighbor discovery. The Network Layer is also responsible for routing and establishing different network topologies such as star, mesh and tree topologies. When a coordinator attempts to establish a Network, an energy scan is completed to find the best RF channel for the Network which is typically the channel with the least interference. After a channel has been chosen, the coordinator assigns a PAN-ID, or a 16-bit number that is used as a Network identifier, which will be applied to all the devices that join the Network [2].

The IEEE 802.15.4 standard defines the lower 2 layers of the of the Zigbee protocol stack as shown in Figure 2. The MAC layer is responsible for relaying data to both the Network and Physical layers, generating beacon frames and carrier-sense multiple access with collision avoidance (CSMA/CA). The beacon frames are used to announce the presence of a Zigbee network and synchronize the devices on it. CSMA/CA is a method that prevents nodes from transmitting packets at the same time. Nodes on the Network will sense if the channel is free before transmitting data on the network and, if it is not free, back off for a pre-defined period. [3]

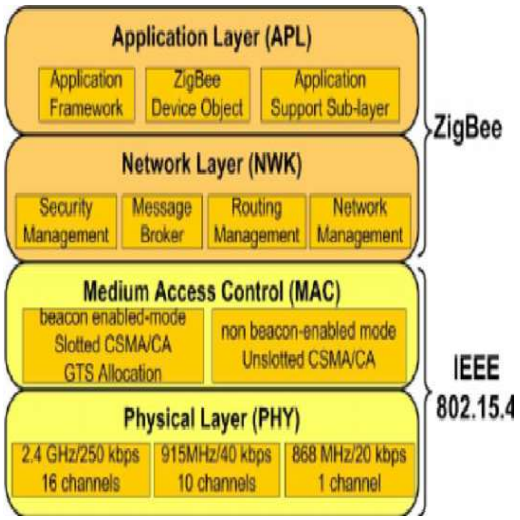

Figure 2: ZigBee Protocol Stack[4]

Several different frequency bands of operation are used by the 802.15.4 protocol as outlined in Figure 3. These bands include the 700 MHz, 800 MHz, 900 MHz, and 2.4 GHz bands. Each band of operation has its own data and spreading parameters. The PHY used in this project is the offset quadrature phase shift key (O-QPSK) used in the 2.4 GHz band. This frequency band is used worldwide containing a total of 16 channels with center frequencies spaced 5 MHz apart ranging from 2.405 GHz to 2.480 GHz. Each channel has a bandwidth of 2 MHz. The data rate for the O-QPSK PHY is 250 Kbit/s [5].

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
780	779–787	1000	O-QPSK	250	62.5	16-ary orthogonal
780	779–787	1000	MPSK	250	62.5	16-ary orthogonal
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary orthogonal
950	950–956	—	GFSK	100	100	Binary
950	950–956	300	BPSK	20	20	Binary
2450 DSSS	2400–2483.5	2000	O-QPSK	250	62.5	16-ary orthogonal
UWB sub-gigahertz (optional)	250–750	As defined in 14.4.1				
2450 CSS (optional)	2400–2483.5	As defined in 13.2		250	167 (as defined in 13.4.2)	
		As defined in 13.2		1000	167 (as defined in 13.4.2)	
UWB low band (optional)	3244–4742	As defined in 14.4.1				
UWB high band (optional)	5944–10 234	As defined in 14.4.1				

Figure 3: Chart of different PHYs, and their spreading and data parameters [5]

The Physical Layer signal for the O-QPSKY PHY is constructed according to the block diagram shown in Figure 3. Each byte of data is converted into 4-bit symbols. The symbol containing the least significant 4 bits is processed first. This 4-bit symbol is taken and mapped to a 32-bit sequence according to its hexadecimal value. The sequences used for mapping are shown in Figure 4. The resulting chip sequences are then modulated onto the carrier using O-QPSK modulation. The even chips are modulated onto the in-phase carrier, while the odd chips are

modulated onto the quadrature phase carrier. In O-QPSK modulation, the quadrature phase carrier always trails or is "offset", from the in-phase carrier by a half-symbol period. Therefore, both the in-phase and quadrature components will never change at the same time [6].

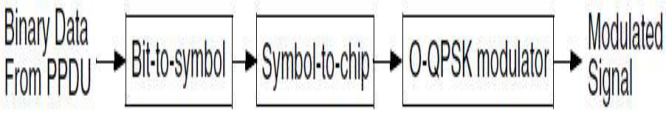

Figure 4: Block Diagram of O-QPSK PHY [5]

Data symbol	Chip values (c_0 c_1 ... c_{30} c_{31})
0	11011001111000011010100100010111 0
1	11101101100111000011010100100010
2	00101110110110011100001101010010
3	00100010111011011001110000110101
4	01010010000101110110110011100001 1
5	00110101001000101110110110011100
6	11000011010100100010111011011001
7	10011100001101010010001011101101
8	10001100100101100000011101111011
9	10111000110010010110000001110111
10	01111011100011001001011000000111
11	01110111101110001100100101100000
12	00000111011110111000110010010110
13	01100000011101111011100011001001
14	10010110000000111011110111000110 0
15	11001001011000000111011110111000

Figure 5: Chip sequences used in O-QPSK PHY [5]

VULNERABILITIES OF THE ZIGBEE PROTOCOL

The ZigBee protocol has various vulnerabilities that can be exploited at both the Network and Application Layers. ZigBee devices typically come pre-loaded with Network or link keys which mean that the keys don't have to be broadcasted over the network when devices join. However, if a node that isn't already associated with a network attempts to join, the network key is broadcasted a single time over the network unencrypted for the new device which poses a

security risk. This unencrypted transmission could be captured using a sniffer and the network would be endangered. This is seen in both home automation and lighting devices. When a new device joins the network that has an alternative key, a publicly known default key is used to transfer the network key in use by the network. This poses a risk since this default key is public information. A sniffer could be used during this unencrypted key broadcast to get the actual network key and this would in turn compromise the Network [7].

In addition to the risks discussed, due to ZigBee being a low power protocol, communication can be disrupted using noise on the RF channel that is used. Also, most end devices used in home automation such as light bulbs, and door locks don't contain some of the security features associated with other devices, such as anti-tamper resistance. So, if these nodes are physically found, the network keys, in addition to any other information contained in software could be physically removed from the device [7].

DESIGN IMPLEMENTATION

The design of the penetration test bed is shown below in Figure 8. Based on the investigation completed in this research, the MAC and PHY layers will be implemented using the XBee module. As discussed, the EM351 SoC contains a 2.4 GHz transceiver and lower MAC that adheres to the IEEE 802.15.4 standard. The XBee module containing the EM351 SoC will connect via USB interface to a laptop running the GNU Radio IDE. Custom python blocks will be created in GNU Radio for the Application and Network layers of the ZigBee protocol. These custom blocks will be contained inside of a python script or application inside GNU Radio. The open source XBee library will be utilized within GNU Radio for sending and receiving packets and for device configuration of the XBee module.

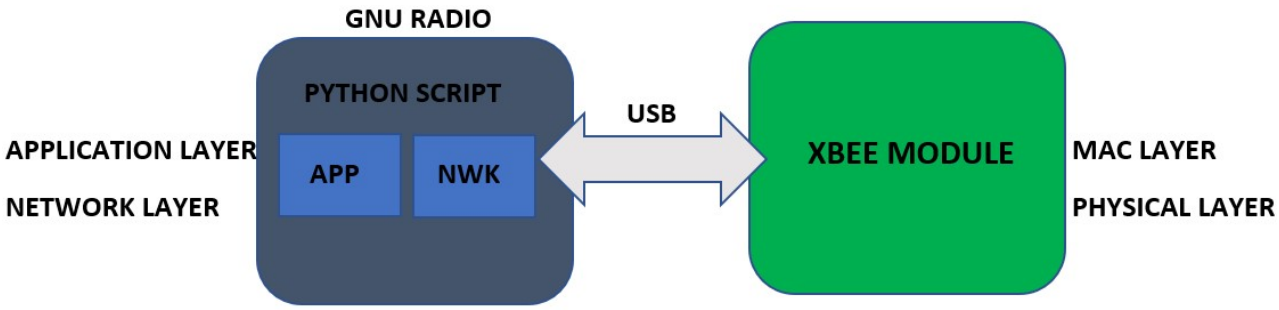

Figure 6: Design Implementation of Test Bed

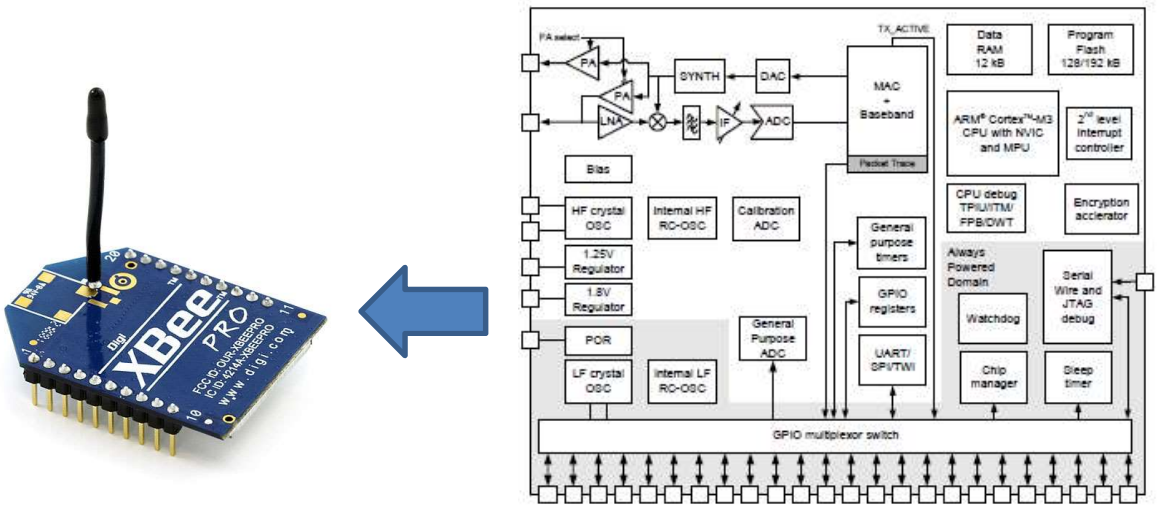

Figure 7: Diagram of EM351 and XBee Module

GNU Radio is a free and open source software framework used for the implementation of software-defined radios. It can be used with low-cost external RF hardware, or without hardware in a simulation-like environment [23]. GNU radio provides a wide range of signal processing blocks for filtering, channel coding, equalizers, etc. These processing blocks can be connected to form applications known as “flowgraphs”. GNU Radio comes with a guided user interface called GNU Radio Companion. GRC allows you to visually put together GNU Radio Applications. It also provides tools for plotting such as scope displays, FFTs, and constellation diagrams. A major strength of GNU Radio is that it allows you to implement custom functionality using C++ or Python. This custom functionality can be implemented using signal processing blocks in GRC. This feature of GNU Radio will be utilized in this research for the Application and Network Layers of the ZigBee protocol.

CONCLUSION

This research project involved the investigation into the development of a wireless IoT penetration testbed for the ZigBee protocol. Based on the investigation completed, it was concluded that the MAC and PHY layers will be implemented using the EM351 SoC contained in the XBee module. The XBee module will be connected via USB to a laptop running GNU Radio which will be used to develop custom Python blocks for the Network and Application layers. The XBee library will be used to implement Network and Application Layer functionality inside of GNU Radio that will allow for packets to be sent and received in GNU Radio and for device configuration of the XBee module. Next steps include the development of custom python blocks for the Network and Application Layers in GNU Radio, and then a test of the full stack development.

REFERENCES

[1] “The Top 3 IoT Devices in Aviation Right Now,” *Betsol*, 25-Jul-2016. [Online]. Available: https://betsol.com/2016/07/the-top-3-iot-devices-in-aviation-right-now/. [Accessed: 12-Jul-2019].

[2] L. AshritLaxmi, Electronics & Communication, and Hp, “ZIGBEE Architecture (ZIGBEE Stack) - All Layers and its Functions,” 08-Dec-2018. [Online]. Available: https://electricalfundablog.com/zigbee-architecture-zigbee-stack-layers/.

[3] “Carrier-sense multiple access with collision avoidance,” *Wikipedia*, 18-May-2019. [Online]. Available: https://en.wikipedia.org/wiki/Carrier-sense_multiple_access_with_collision_avoidance. [Accessed: 12-Jul-2019].

[4] "Open-ZB: an open-source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS - Semantic Scholar", *Semanticscholar.org*,2018.[Online].Available: https://www.semanticscholar.org/paper/Open-ZB%3A-an-open-source implementation-of-the-IEEE-CunhaKoubaa/0df134e852d5fa7ff4d5f33e27303312e7a30983. [Accessed: 11- Apr- 2018].

[5] IEEE Computer Society, “IEEE Standard 802.15.4,” 2011.

[6] “RF Wireless World,” *QPSK vs OQPSK vs pi/4QPSK-Difference between QPSK,OQPSK,pi/4QPSK*. [Online]. Available: https://www.rfwireless-world.com/Terminology/QPSK-vs-OQPSK-vs-pi-4QPSK.html. [Accessed: 12-Jul-2019].

[7] T. Zillner, “ZigBee Exploited: The good, the bad, the ugly,” *Cognesec*, 2015.