ZigBee Mesh and Graphical User Interface for a Wireless Tunable Laser Spectrometer Array Onboard the International Space Station

NASA Jet Propulsion Laboratory *(NASA/JPL-Caltech)*

J. Gehant, S. Eschbach, Z. Abbas
College of Engineering and Engineering Technology
Northern Illinois Univeristy
DeKalb, Illinois
Joseph.K.Gehant@Gmail.com, Susanna.Eschbach484@Gmail.com, Zahra.Abbas1551@Gmail.com

Abstract: **Astronauts working onboard the International Space Station have reported recurring health issues that are potentially related to exposure of high levels of carbon dioxide. However, there currently is not a robust and reliable way to monitor the concentrations that astronauts are experiencing. Some studies suggest that long-term exposure to high levels of carbon dioxide can cause bone and kidney composition changes, chronic inflammation, and vision impairment, with short-term issues such as dizziness, lethargy, increased blood pressure, and headaches. Because exposure to high levels of carbon dioxide can result in these long and short-term health issues, it is important to measure carbon dioxide and other trace gases in real-time over a system-wide spatiotemporal range to provide an understanding of areas and times of high carbon dioxide accumulation.**

Keywords: ***Wireless Sensor Networks; ZigBee; Carbon Dioxide Distributions; 2.4-2.5GHz Interference Avoidance***

I. Introduction

A more extensive means of tracking carbon dioxide concentrations in real-time onboard the International Space Station (ISS) is required. Wirelessly transmitting tunable laser spectrometer sensor readings and network performance metrics in a ZigBee mesh network with six remote nodes to a base station graphical user interface (GUI) is the proposed solution.

The project features four primary objectives in the pursuit of fulfilling this task: First, interface microcontrollers attached to tunable laser spectrometers with ZigBee protocol communication modules to collect and wirelessly transmit low-layer network health metrics and spatiotemporal distributions of carbon dioxide and other trace gases to the network collector/coordinator. Second, optimize network transmissions by adjusting ZigBee networking parameters and designing a custom API in C. Third, design a graphical user interface for a base station laptop to receive, parse, visualize, and log network performance metrics and sensor readings for each remote node. Fourth, design an algorithm to actively sample the 2.4-2.5GHz spectrum to detect and actively minimize radiofrequency interference to coexist with other wireless communications operating on the same band.

II. Materials

The two main components of this project are the ZigBee RF modules provided by Digi called XBees that are used to create the mesh network, and the custom JPL printed circuit boards (PCBs) with AVR and ARM-based microcontroller units (MCUs). The ZigBee protocol was determined to be appropriate for this wireless sensor network because of its versatile mesh-networking capabilities, high-configurability, low-power transmissions (<0 dBm), low RF overhead, minimum RF interference, low power consumption, and self-healing. For this project, seven XBee modules were used: Six XBees were configured as routers and interfaced with microcontrollers to collect and transmit data through the network, and one XBee was configured as a coordinator to receive, demodulate, and transmit payload data locally via UART serial communication to the base station for processing, logging, and then displaying on the GUI.

The XBee modules are connected to the JPL MCU PCBs via UART Rx and Tx pins. The MCUs are programmed to poll and collect data from tunable laser spectrometers via analog-to-digital conversion (ADC). *Figure 1* displays the XBee module attached to the PCB.

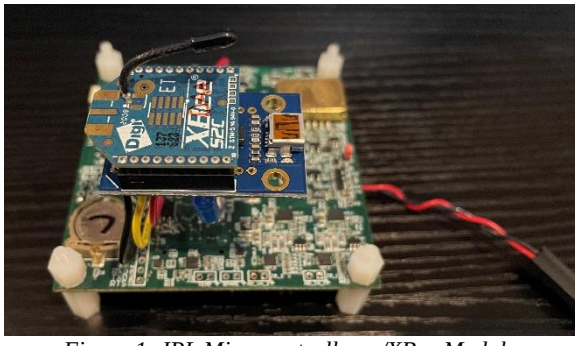

Figure 1. JPL Microcontroller w/XBee Module

III. Methods

The microcontrollers are programmed in C and their task is to collect methane, ethane, water vapor, carbon dioxide, and pressure data from their attached sensors, and poll for neighbor tables, routing tables, link quality indicator, received signal strength indicator, APS acknowledgement successes, APS acknowledgement failures, MAC acknowledgement failures, clear channel assessment failures, APS RF retries, and 2.4-2.5GHz energy spectrum analysis values from their local XBee. To obtain the network health metrics from the local XBee, the microcontroller must send out several ZDO-layer request frames for each specific cluster identification number. When the XBee receives the

frame, it replies with the requested index and data in a response frame to the microcontroller's Rx pin. Once the microcontroller receives the data from the XBee and attached sensors, it parses out the relevant bytes, formats the data into transmit request frames with ZigBee packet payloads, and then sends the frames to the attached XBee via its UART Tx pin. The XBee then processes the frame and executes the commands and addressing bytes in the preamble to transmit the payload data via ZigBee protocol to its destination – the coordinator connected to the base station via FTDI to USB.

The base station is programmed in Java and it receives frames formatted by the coordinator containing payload data from each remote node through a serial data event listener with read blocking. Once a frame has been detected, the information contained in that frame is parsed out, stored in device objects, and then displayed on the GUI. The GUI displays the data in graphs through an open-source Java graph program called Telemetry Viewer (TV). Alongside TV are two additional custom graphs that have been created: An area chart to visualize carbon dioxide distributions more effectively over time, and a network topology graph of all remote nodes with active links. Windows are also being rendered to display network performance/health metrics that are better shown as numerical values in tabular form. *Figure 2* below shows the interconnectivity of transmissions through the network's primary system components.

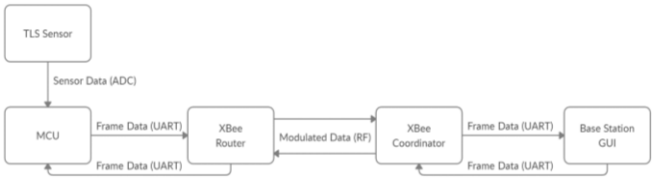

Figure 2. Network Block Diagram

IV. Features

There are two main features that have been designed and implemented into this project. The first is a timeout buffer that has been added to the microcontroller code. The purpose of the buffer is to ensure 0% data loss in the event of an RF transmission timeout, failure, and/or network reset. The buffer operates as a circular/ring buffer and can store up to 8 minutes of timestamped sensor data before it is overwritten. Once the network can operate normally in real-time after the buffer is triggered, the XBee transmits the accumulated data in optimized payloads to the base station. This ensures that when the data is reviewed by the end-user at the GUI, they will always see a consistent logging of sensor data at the time the samples were collected at the microcontroller, and thus be able to reliably monitor spatiotemporal distributions of carbon dioxide throughout the ISS.

The second feature is an energy detection and channel changing algorithm called "ZigZag". The purpose of ZigZag is to ensure that the ZigBee network is operating on the channel with the least amount of RF interference and prevent the network from interfering with other coexisting networks operating on the same 2.4-2.5GHz band. ZigZag performs an energy scan on network startup to determine the best initial channel for the network to operate on. During network operation, ZigZag periodically scans all ZigBee channels at each remote node to determine if there is a channel with significantly less noise to operate on. If certain criteria are met, ZigZag will trigger a popup on the GUI and ask the user if they want to switch the network over to the new channel. If the user agrees, then the entire network will be switched over to the new operating channel.

V. Results

The network has been tested in a controlled environment in both low and high-interference conditions. The tests have shown the network performs better in star-configuration (one-hop) than in multi-hop configuration. The network also performs optimally with <0.01% packet loss when it is not operating on the same frequency as the generated noise/interference, proving the need for ZigZag. However, by implementing the timeout buffer, data will only be delayed, not lost, in the event of a transmission failure. *Figure 3* provides results from tests in a controlled environment.

Condition	Test Configuration	Packets	Retries	BufferCnt	Failures
Noise-Free	Star	6933	172	29	0
Noise-Free	Multihop	6933	1240	50	26
High-Noise	Star	6933	1980	59	0
High-Noise	Multihop	6933	4660	259	34

Figure 3. Network Test Results

VI. Conclusion

Several network health metrics have been implemented to monitor network performance. To reduce the probability of high levels of interference/noise compromising the network's integrity, a timeout buffer has been implemented ensure 0% data loss, as well as a channel energy spectrum analysis algorithm to determine the optimal operating channel and configure the network to operate on it. Under low-noise conditions, the network can gather and transmit data with <0.01% packet loss. When higher levels of noise are introduced on the same operating frequency, latency is increased, and the timeout buffer is utilized more frequently.

Acknowledgments

As a team, we would like to thank the following individuals and institutions:

- Dr. Christensen for providing input to help improve the network, assisting with troubleshooting, and being involved with the project.
- Dr. Fonseca for all his help, guidance, and knowledge throughout this project.
- Northern Illinois University and NASA/JPL-Caltech for collaborating to make this project possible.