Industrial Wireless Networks

Nesta Isakovic, Tiffany Luu, Mitchell Orsucci

Industry Mentors: Randy Sales, John Gilbert Faculty Mentor: James Peckol

ELECTRICAL ENGINEERING
UNIVERSITY of WASHINGTON

INTRODUCTION

Industrial sites are large environments with a need for reliable and responsive communication platforms. Our goal was to evaluate various wireless network topologies in order to develop a scalable and robust industrial wireless network. The system needs to be reliable in a signal hostile environment and exhibit robust communication, self-healing capabilities, scalability, and low power consumption. With these specifications, our system can be trusted to operate in a dynamically changing environment.

APPROACH

To design a network of up to 100 nodes, and given the time limitations of the project, we chose to utilize off-the-shelf components in our implementation. We initially selected three different hardware networking solutions to evaluate:

- Dust Networks from Analog Devices (IEEE 802.15)
- XBee from Digi (IEEE 802.15)
- Blue Gecko from Silicon Labs (BLE)

The Blue Gecko hardware had package types were not easily transported to a PCB without complex soldering tools. Due to this, we decided to focus on evaluating the Dust Networks and XBee hardware.

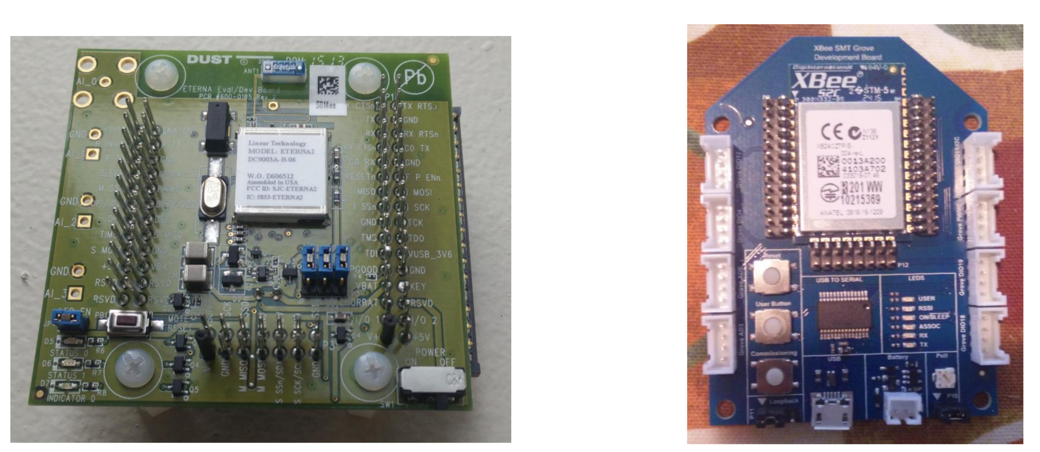


Figure 1. ***Left:*** Dust Networks mote; ***Right:*** XBee node

HARDWARE EVALUATION

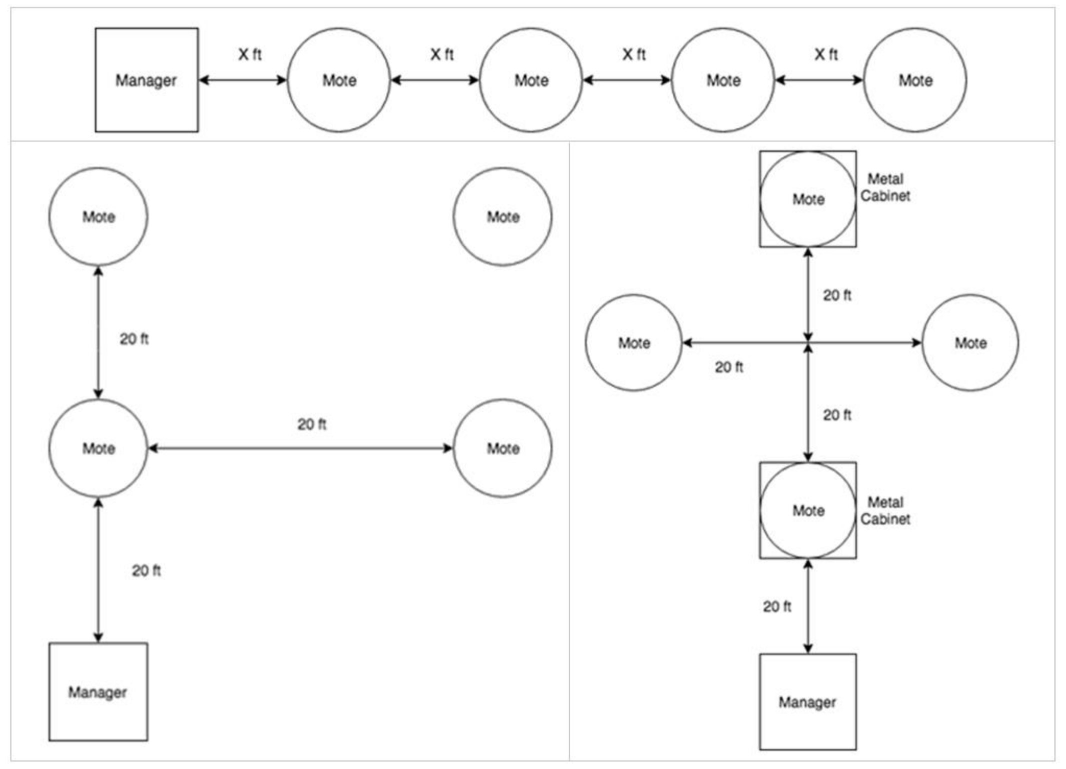


Figure 2. Different topologies for different types of tests. ***Top:*** Topology for range test; ***Left:*** Topology for ping tests; ***Right:*** Topology for interference tests

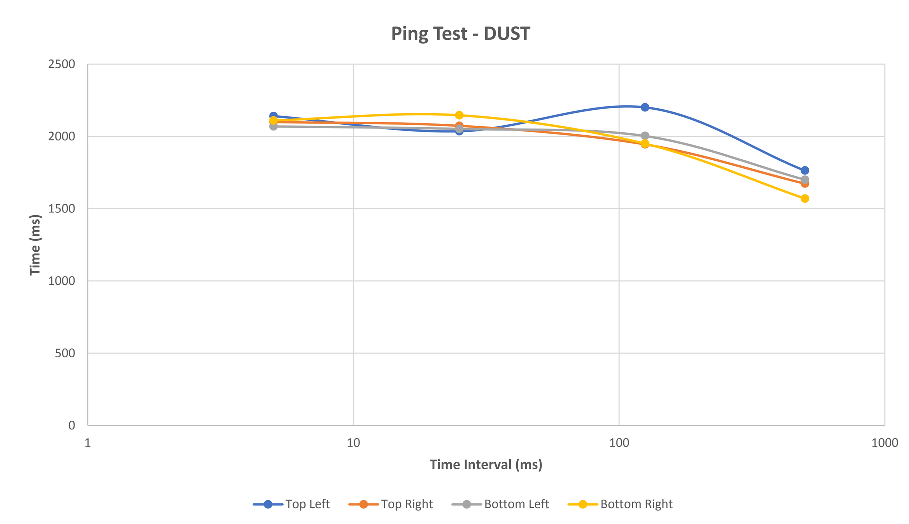
Hardware Evaluation Metrics

1. Power consumption of a node in the network during normal operation
2. Network reliability (ratio of successfully sent packets vs. dropped packets)
3. Ease of programmability and manipulation of the network
4. Available encryption schemes

TEST RESULTS

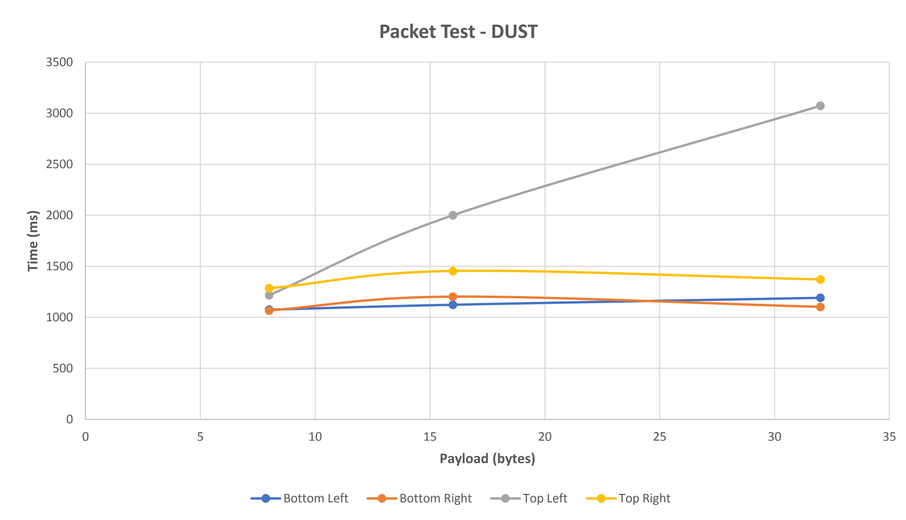
Ping Test:
Each node on the network was pinged 100 times, in the configuration shown in Figure 2. We measured the amount of time that it took for the node to respond to the gateway and noted dropped packets.

Graph 1. Dust ping test, with varying time intervals between each ping



Packet Test:
The gateway device sent packets of different sizes to each node in the network. This test was run in multiple network configurations with nodes placed in varying locations.

Graph 2. Dust packet test, with varying packet sizes (8, 16, 32 bytes)



Power Test:
We measured the current consumed during normal operation of our tests. The results were:

- XBee: ~40mA at 3.3 V
- Dust Networks: ~4mA at 3V

Given the power constraints of our design, these findings disqualified XBee as a viable option.

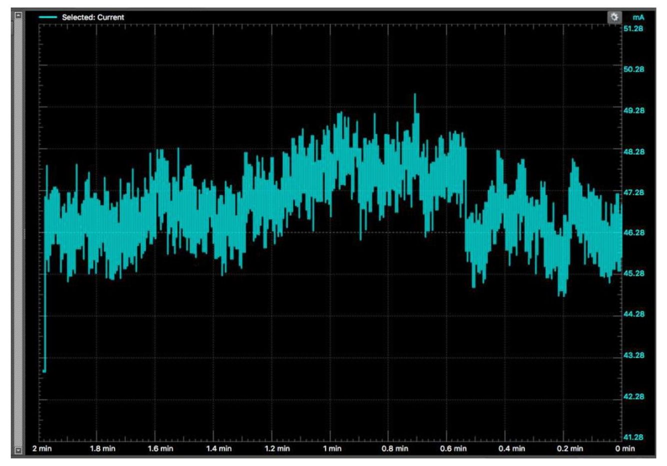


Figure 3. Graph of XBee current draw

Additional Considerations:

- Both provide software tools for network management, but Dust Networks provides more abstraction for how packets are received and acknowledge as compared to XBee.
- Both networks provide encryption methods. XBee requires more from the programmer to manage encryption than Dust does.

-> With these parameters in mind, we decided to move forward with Dust Networks.

FINAL SYSTEM

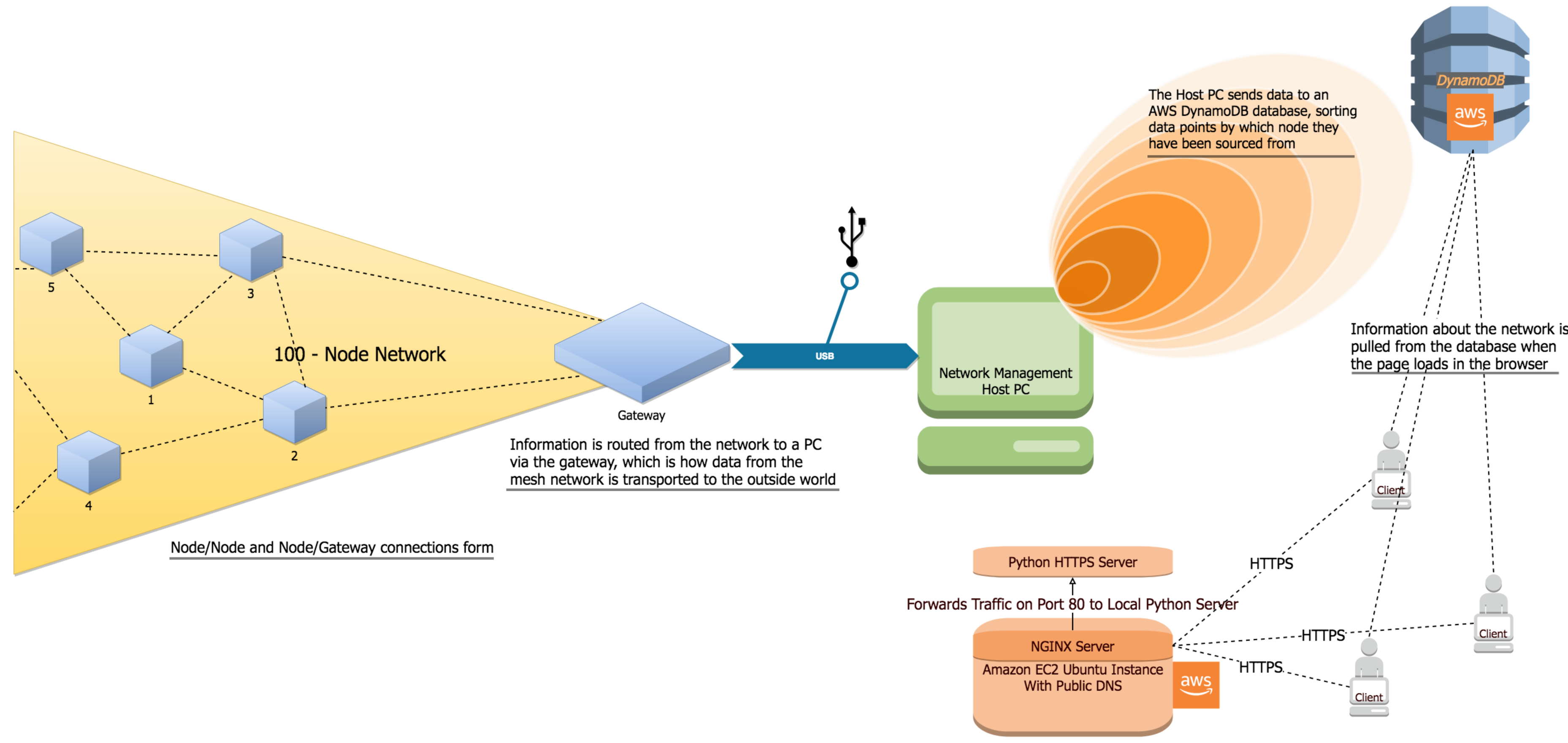


Figure 4. Overall system diagram. ***Hardware Components:*** 95 Smartmesh IP LTP5901-IPM Motes, 5 Smartmesh IP LTP5902-IPM Motes, and Manager; ***Software Components:*** Amazon AWS EC2 Server, Amazon AWS, Dynamo Database, Python HTTPS Server, NGINX Server, TauCharts, Smartmesh SDK, Temperature Recorder Python Script

SOFTWARE IMPLEMENTATION

To view the data from the network, we utilize the following servers:

- Amazon AWS EC2: runs Ubuntu and has a public DNS
- NGINX server: handles HTTP requests and forwards to local server
- Local Python server: serves static HTML, CSS, and JavaScript content

The gateway to the network is connected to a PC, which is running a Python script that utilizes the Smartmesh SDK (API for Dust Networks). Temperature data is collected from each mote and stored in an Amazon AWS DynamoDB database. We can display this data using the public DNS.

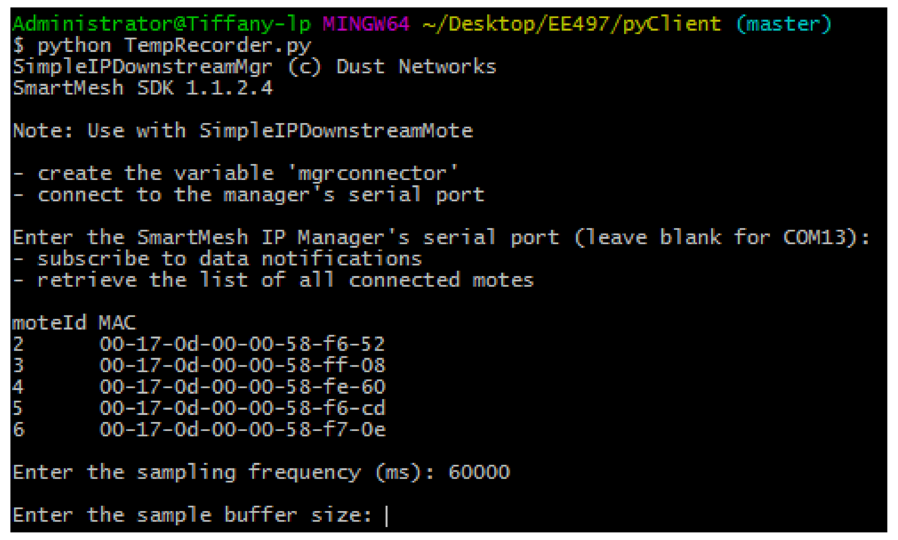


Figure 5. Example of TempRecorder.py, running in the command line interface.

The website uses TauCharts to plot the most recent data received from the network. The time range and temperature for each data point can be viewed on a graph for each mote in the network. Users can also select from the list of motes attached to the network and command it to turn on its LED.

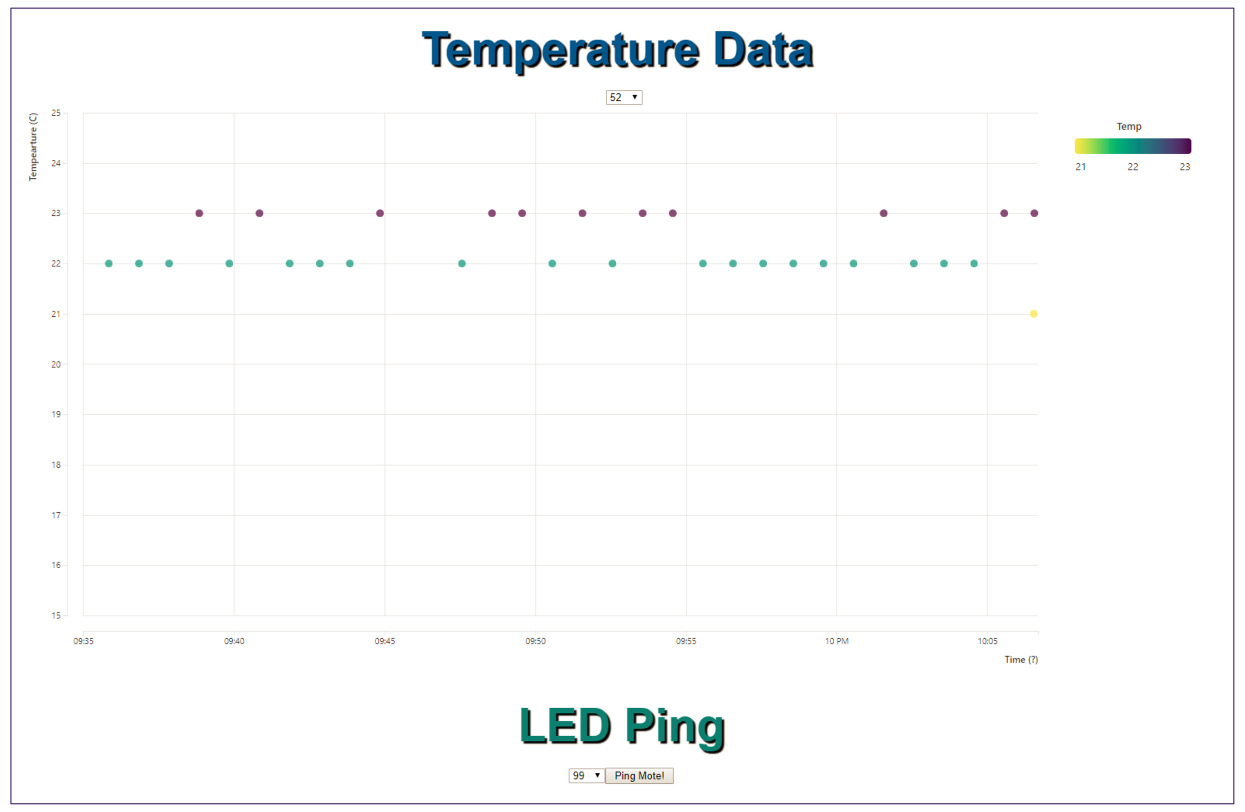


Figure 6. Screenshot of the website used to view the network data.

CONCLUSION

For this project, we accomplished the following:

- Tested and evaluated the capabilities of different hardware setups
- Established a large network with our chosen hardware
- Pulled data from the established network and sent it to an online server
- Showed capability of allowing the user to affect the network (LED ping)

Future work that could be done to improve the product:

- Vary the type of data being collected by the network
- Explore encryption and security
- Long-term permanent setup location