

INFINETIX SmartMesh® Gateway Project

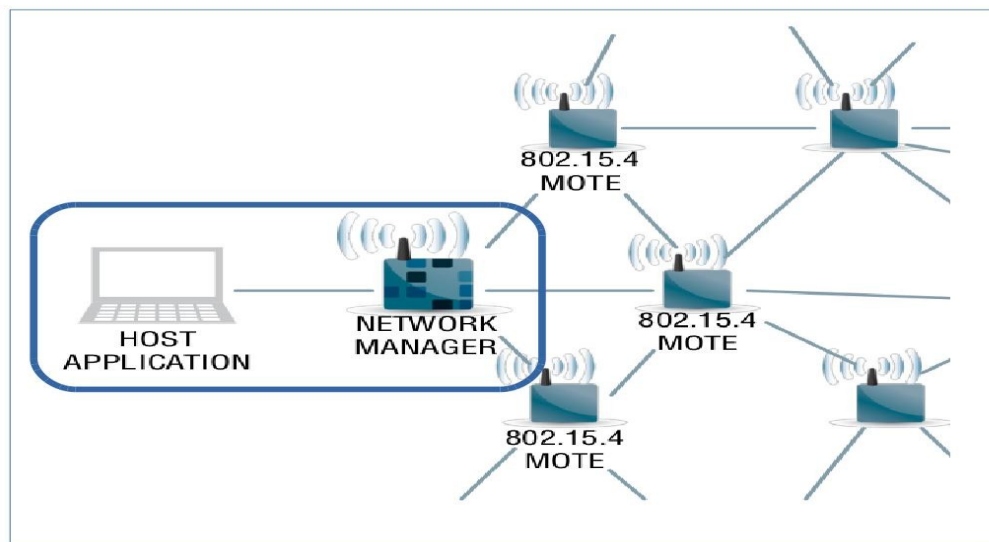
As part of our investigation into the Linear Technology's Dust Networks® SmartMesh IP™ technology, INFINETIX has designed a custom SmartMesh® Gateway as a first step in developing a complete wireless sensor network. The SmartMesh® Gateway can be used as a development platform, a reference design, or as part of a deployed system. This discussion describes both the hardware and software aspects of this development.

For general information on the Linear Technologies Dust Network, see the Reference Information section at the end of this document.

1 System Information

1. SmartMesh Gateway

The Linear Technology/Dust Networks SmartMesh IP system interacts with external systems via a gateway or host application that provides communication and control to the SmartMesh network manager as well as providing access of the data collected to a broader group of applications. The gateway application runs on a separate microprocessor and communicates with the SmartMesh manager via the application programming interface (API) on a two wire serial link.



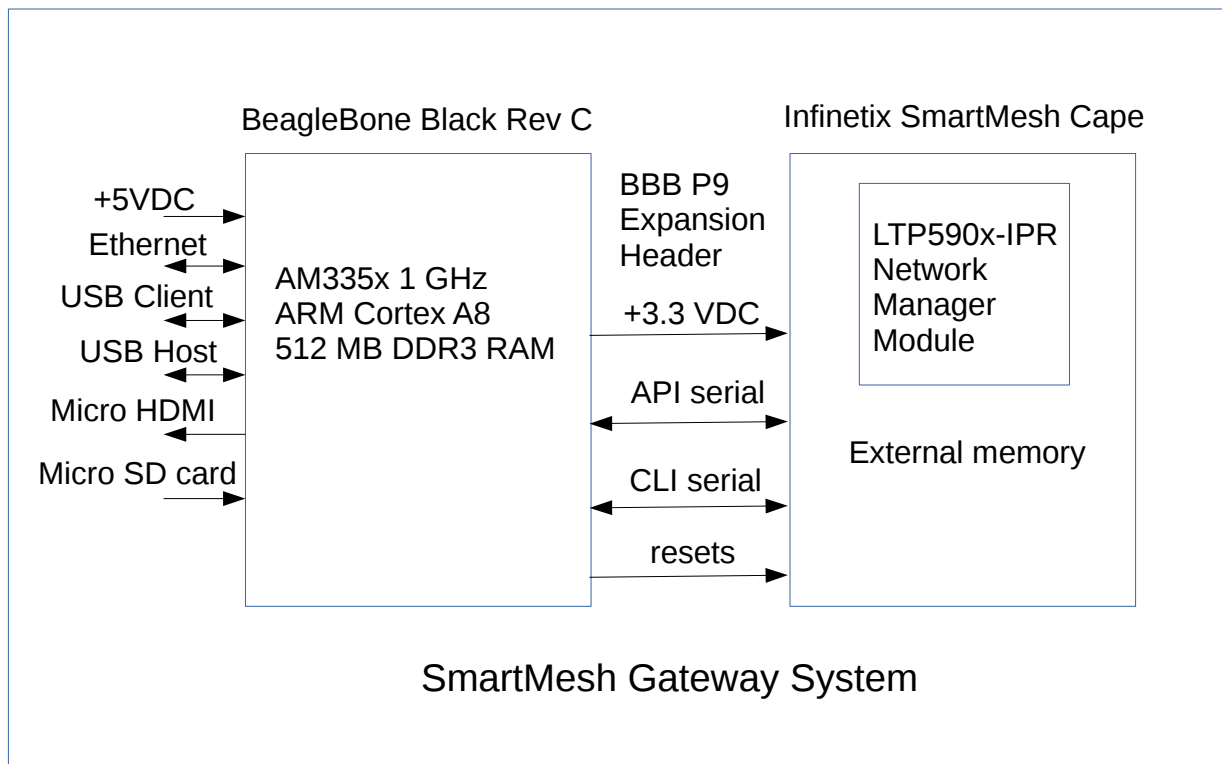
For the INFINETIX SmartMesh Gateway system, the BeagleBone Black (BBB) single board computer was selected for the host application and a custom interface board (cape) was designed for the network manager and associated supporting hardware.

The BBB, using embedded Linux as the operating system, can run either the DustNetworks DustLink gateway software or a custom C routine. These gateway applications can provide an interface to the SmartMesh system, process data collected and serve external web applications or data bases.

2 Hardware Development

1. SmartMesh Gateway Block Diagram

The hardware system block diagram, showing the BeagleBone Black, the interface board (cape) and basic interconnections is shown below. More detail on each block is included in the subsequent sections.

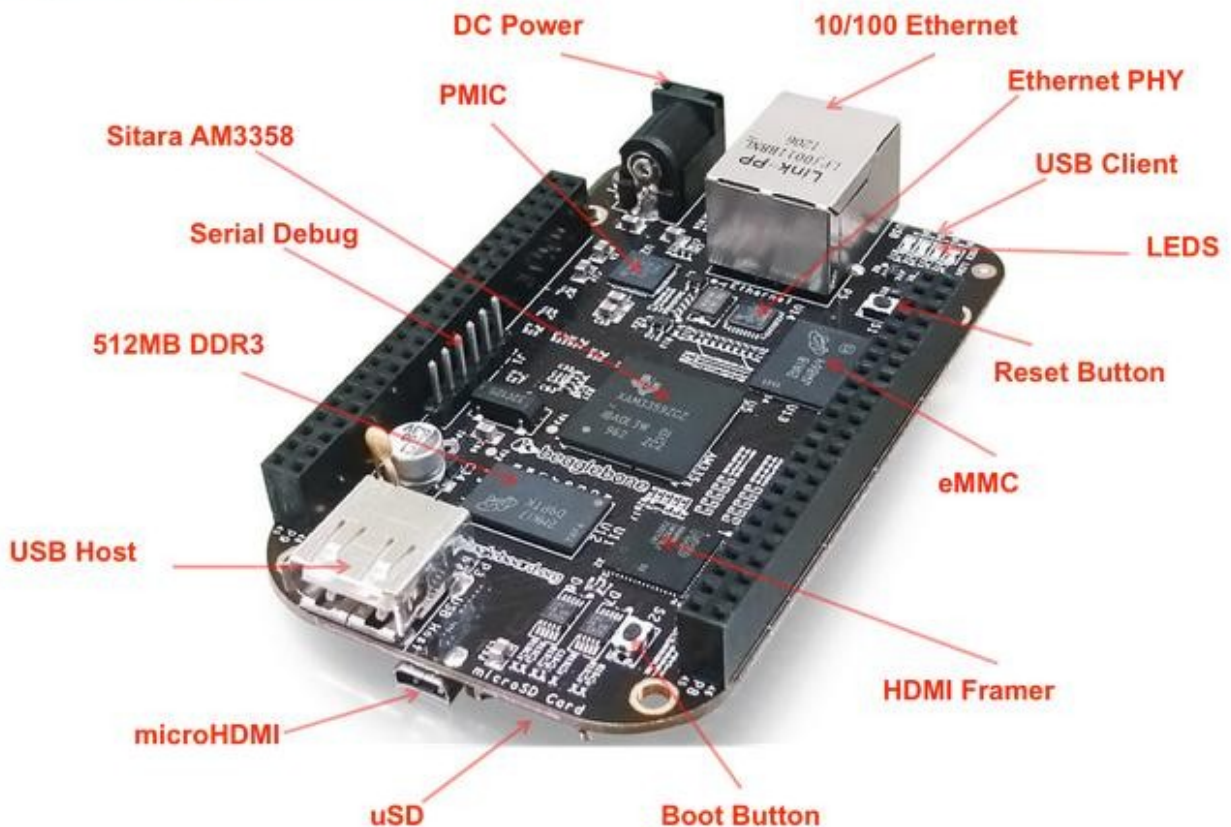


2. BeagleBone Black

This work was done and tested with Rev C of the BeagleBone Black (BBB). The BBB is an open hardware, community-supported embedded computer platform targeted for developers and hobbyists. The combination of price, performance, flexibility, and expandability make it ideal for this type of project. Some pertinent details are shown below. For more information see beaglebone.org.

- Overview of board (from beagleboard.org/black)
 - Processor AM335x 1GHz ARM® Cortex A8
 - 512 MB DDR3 RAM
 - 4 GB 8-bit eMMC on-board flash storage
 - 3D graphics accelerator,
 - microHDMI video out
 - NEON floating-point accelerator
 - 2x PRU 32-bit microcontrollers.

BeagleBone Black hardware details



3. SmartMesh Manager Cape design



The INFINETIX SmartMesh Manager Cape design provides the means to interface the SmartMesh Network Manager with the BeagleBone Black host computer system. The design was implemented as a standard BeagleBone expansion module, known as a cape the BeagleBone community. Some details of the cape design are discussed below.

- Network Manager module

The SmartMesh Manager cape is designed to use any of the LTP590x-IPR Network Manager modules. The main variations in available managers are in the type of antenna and in the number of motes supported by the manager. See the Network Manager datasheet (59012iprf) for specifics. The selected castellated module is soldered directly on to the cape board during assembly.

- External memory (including CPLD interface) (Reference: Eterna LTP5901/LTP5902 Integration Guide)

Without the addition of external SRAM, the network manager is limited to supporting networks of 32 or fewer motes. External memory is added to allow support of up to 100 motes and/or to increase throughput of the manager from 24 to 36 packets per second.

The cape design includes an Altera Max V CPLD to handle the address decoding associated with the external memory. This is different than the method

shown on the Linear Technology reference designs, which utilize RC time constants to control the address decoding registering. Please contact INFINETIX for specific information on the CPLD design.

- Interconnects with the BBB

The SmartMesh Manager cape connects to the BeagleBone Black via the P9 expansion header. This header provides power and ground reference as well as control and data communication paths. Please contact INFINETIX for specific pin and schematic information.

- Power: The cape design only uses the +3.3V supply from the BBB. The cape current draw should normally be less than 25 mA, with a peak estimate (during radio transmitting at 8 dBm with worst case memory access) at 118 mA.
- UART connections
 - The Manager API connects to /dev/ttyO1 (BBB-UART1)
 - The Manager CLI connects to /dev/ttyO4. (BBB-UART4) Note that the cape design includes zero ohm resistors to provide configuration options in connecting the BBB to the desired manager UART. The default connection will be to UARTC1 on the manager.
- Antenna layout detail: Per the recommendations of the Eterna LTP5901/LTP5902 Integration Guide, the area beneath the chip antenna as is used for the LTP5901 is kept clear of traces and dielectric material (FR4).
- Reset management: There is an active low asynchronous reset to the Manager module. From the datasheet (59012iprf):

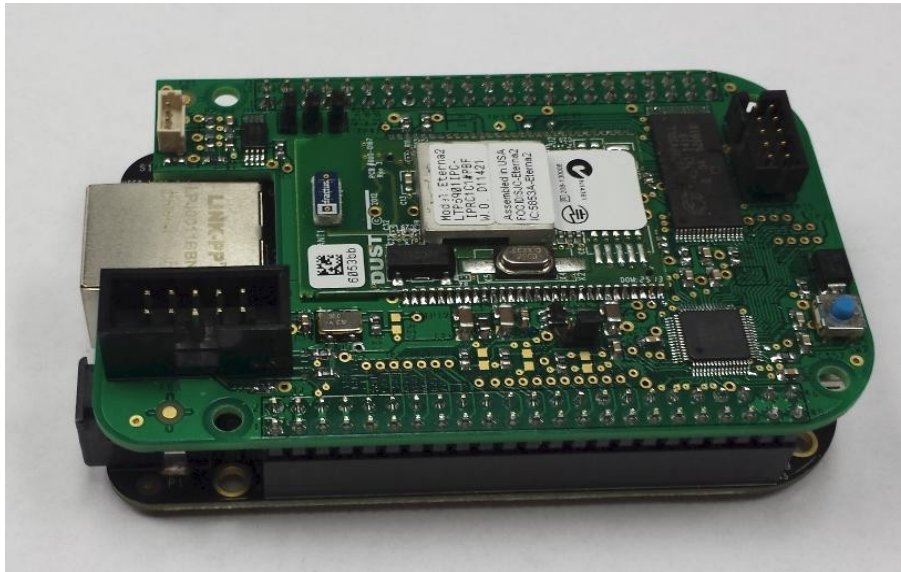
RESETn: The asynchronous reset signal is internally pulled up. Resetting Eterna will result in the ARM Cortex-M3 rebooting and loss of network connectivity. Use of this signal for resetting Eterna is not recommended, except during power-on and in-circuit programming.

 - There are four possible signals that can assert the RESETn line to the manager:
 - SYS_RESETn: This is the active low system reset signal from the BBB.
 - MOTE_RESETn: This signal comes from the BBB and is a GPIO control line that can be used to reset the manager under software control.. It is currently configured to be active high (in spite of the name).

- PUSH_RESETn: This active low signal is controlled by the push button circuit. This signal is pulled up to the +3.3v supply.
 - IPCS_RESETn: This active low signal is associated with the programming of the manager module via the Eterna Serial Programming (ESP) module.
- These four signals are logically or'd in the CPLD such that any of them being active will cause the RESETn signal to assert (active low). To ensure that false resets do not occur, internal pull-ups in the CPLD were added for the IPCS_RESETn and SYS_RESETn lines.
- Also to allow flexibility in the way resets are managed, a couple of other options were provided in case the CPLD was not loaded or configured.
 - The CPLD combination reset signal and the MOTE_RESETn signals can be selected via a couple of zero ohm resistors. The default loading would be for the CPLD output to be utilized.
 - A header was included allowing the selection between the IPCS_RESETn signal and the MOTE_RESETn/CPLD point. The default is for the header to connect to the MOTE_RESETn/CPLD connection.
- Programming headers on the Gateway cape design:
 - One (J5) is for programming the CPLD.
 - Another (J4) is for the Manager module ESP interface.
 - The third (J3) is for the serial PROM that stores configuration information to communicate cape details to the BBB. Jumpers (JP1, JP2, JP3) can be loaded to enable writing to the PROM and to set the address.
 - All three of these headers will typically be no-loads for a production board.

4. Package discussion

- SmartMesh cape installed on a BBB.



- SmartMesh Gateway demonstration model. This is a simple enclosure, done on a 3D printer, along with an etched cover, intended to display an operational gateway.



- Demonstration model with cellular back haul. This version includes an example of the BBB, the cape and a cellular radio enclosed in an example case.



3 Software Development and Usage

1. Dustlink interface

The Python based DustLink gateway interface software can run on the INFINETIX SmartMesh Gateway hardware using the following steps:

- Autostart configuration. The BBB can be set to automatically start up the DustLink application. To make that happen do the following:
 - Connect the Gateway to your computer network. This can be through the Ethernet port on the Gateway or through the USB client port. These instructions assume the USB port. If using the Ethernet port, substitute in the appropriate IP address for your system.
 - Connect via SSH (secure shell)

```
ssh ubuntu@192.168.7.2
```
- To interact with the Dustlink software, do the following from a web browser.
 - Open the server at 192.168.7.2:8080

(Note: this is assuming the USB client is the interface. Substitute the appropriate IP address if running the Gateway via the Ethernet port)

- Login as admin with a password of admin
- Click on the Managers option on under System on the side panel and this will show you the manager connections configuration page.
- The connection should be to /dev/ttyO1. If this is not the case, use the add and delete options at the Managers page to set the connection to /dev/ttyO1
- When complete, the /dev/ttyO1 connection should say it is active within 5 seconds

of completing the configuration.

- Now the system is connected and will operate as if connected to the regular development board.

2. API/CLI operation

Communication with the Network Manager occurs via the Application Programming Interface (API) or Command Line Interface (CLI). Both of these interfaces operate on serial links between the BBB Gateway processor and the Network Manager. From the SmartMesh_IP_Manager_API_Guide:

The API is intended for machine-to-machine communications (e.g. a host program talking to the manager).

In contrast, the Command Line Interface (CLI) is intended for human interaction with a manager, e.g. during development, or for interactive troubleshooting.

- API operation:

(See the SmartMesh_IP_Manager_API_Guide for more information)

For the purposes of this discussion, the API exposure is through the DustLink software or the INFINETIX Gateway application. See those sections for more detail.

The serial port for API communication is ttyO1 (capital O, not zero 0) and the link is of type 115200, 8n1.

- CLI operation:

(See the SmartMesh_IP_Manager_CLI_Guide for more information on the specific commands) What follows is a brief start-up guide.

-- log in:

```
ssh ubuntu@192.168.7.2 (or whatever is the appropriate IP address)
```

```
password: temppwd
```

-- Get the serial port running

```
minicom -s
```

-- Set up the serial port: (if not already set)

```
ttyO4 (note: not zero, but capital O)
```

```
baud 9600
```

```
Hardware/Software flow control: No
```

- log in and help for command list
 - > login user
 - > help
- a couple of CLI commands that have been useful:
 - > show status : shows overall network status
 - > sm : shows list of motes in the current network
 - > show mote <MoteID> : shows mote links and status
 - > ping <MoteID> : ping a given mote

3. INFINETIX Gateway application

The gateway application is a custom C routine running in an embedded Linux environment on the BeagleBone Black. The functionality of the application is to provide basic visibility to the local mesh network via the API as displayed through a local web server as well as to post collected data from the mesh network to an external web server. The local and external web interfaces are discussed later in this document.

- Operating system on BBB. The open source Ubuntu version of Linux is used as the operating system for this project. The development work done so far has been using Ubuntu 14.04.1 LTS (codename: trusty). To aid in ease of development and updates the operating system and application code are run from a micro SD card inserted into the BBB. The development work to date is running on an 8 GB SD card.
- Development environment. The development (editing and compiling) is done in the BBB Ubuntu environment, using gnu tools. Some specific tools installed (sudo apt-get install) were build-essential, libgnutls-dev, libmagic-dev, vim-gtk, and libcurl4-gnutls-dev.
- Gateway application code. Please contact INFINETIX for specific information on the application code and how it can be adapted for other usages.
- Autostart operation. As discussed in the DustLink software section, the BBB can automatically launch an application after booting. To have the autostart launch the Gateway application, check the /etc/rc.local file. It should include the following line:

```
/home/ubuntu/new_dust/run.sh
```

Also make sure that the following line is commented out as shown below:

```
#!/home/ubuntu/new_dust/run_dustlink.sh
```

4. Internal Gateway web server

The internal gateway web server uses JSON, jquery, and knockout javascript methodology to provide a basic web interface to the SmartMesh network. To see the dust_manager web information: Enter the following into a web browser URL line:

192.168.7.2:7878

Where 192.168.7.2 is the IP address (change as appropriate) and 7878 is the configuration server Port number, set by the run.sh shell

invoked at start-up.

You should see Sensor Setup tab as the default, with two other tab options: Ping Motes and Mote Data.

- Sensor Setup:

The screenshot shows the INFINETIX web interface. At the top left is the INFINETIX logo, and at the top right is the LINEAR TECHNOLOGY logo. Below the logos is a navigation bar with three tabs: 'Sensor Setup' (selected), 'Ping Motes', and 'Mote Data'. To the right of the navigation bar is a link for 'Configuration Wizard'. The main content area is titled 'Sensor Setup' and contains a table with the following data:

| <u>Mote Name</u> | <u>MAC</u> | <u>Access Point</u> | <u>State</u> | <u>Routing</u> |
|------------------|-------------------------|---------------------|--------------|----------------|
| Manager #4 | 00:17:0D:00:00:60:61:6B | True | Operational | True |
| Downstairs | 00:17:0D:00:00:60:21:88 | | Operational | True |
| Upstairs | 00:17:0D:00:00:60:20:18 | | Operational | True |
| Infinetix Shop | 00:17:0D:00:00:60:20:23 | | Operational | True |

Below the table is a 'Scan' button. At the bottom of the page is a footer with the text '© 2014 Infinetix Corp.'

This page displays the basic sensor information: name, MAC, access point, and operational status.

-- The Scan button does the following:

- Executes the API getMoteConfig command to gather information on all the links in the network. This information is displayed on this web page.
- Pulls the names associated with the motes in the network off of the external web server. These are displayed in the Mote Name field
- Executes the build links function which uses the API getMoteLink command to build up the network link information. This network link information is pushed to the external web server.

- Ping Motes:

| <u>Mote Name</u> | <u>Response</u> | <u>Callback ID</u> | <u>Delay</u> | <u>Voltage</u> | <u>Temperature</u> |
|------------------|-----------------|--------------------|--------------|----------------|--------------------|
| Manager #4 | OK | 401 | 23 | 3385 | 36 |
| Downstairs | OK | 402 | 3822 | 2902 | 21 |
| Upstairs | OK | 403 | 879 | 2907 | 22 |
| Infinetix Shop | OK | 404 | 2387 | 2858 | 12 |

This page updates and displays information that is gathered by the API pingMote command

-- The Restart button does the following:

-- Sends the Hello packet to the manager to initiate a new session.

-- Clears any internal flags associated with the ping loop

-- The Update button starts a loop that pings all of the motes in the network as determined in the Sensor tab

- Mote Data: Page to update and display information that is sent by the active motes in the network.

INFINETIX

LINEAR TECHNOLOGY

Sensor Setup Ping Motes **Mote Data** Configuration Wizard

Latest Mote Data (returned)

| Mote Name | Timestamp | Source Port | Destination Port | Data Payload |
|----------------|-----------|-------------|------------------|--|
| Downstairs | 3D473D6A | F0B9 | F0B9 | 00:00:05:00:FF:01:05:00:00:00:00:3D:47:3D:6A:00:04:68:DE:00:00:75:30:01:10:08:A8 |
| Upstairs | 3D473D60 | F0B9 | F0B9 | 00:0B:05:00:FF:01:05:00:00:00:00:3D:47:3D:60:00:02:10:0B:00:00:75:30:01:10:09:4A |
| Infinetix Shop | 3D473D52 | F0B9 | F0B9 | 00:00:05:00:FF:01:05:00:00:00:00:3D:47:3D:52:00:0B:31:EC:00:00:75:30:01:10:06:AB |

© 2014 Infinetix Corp.

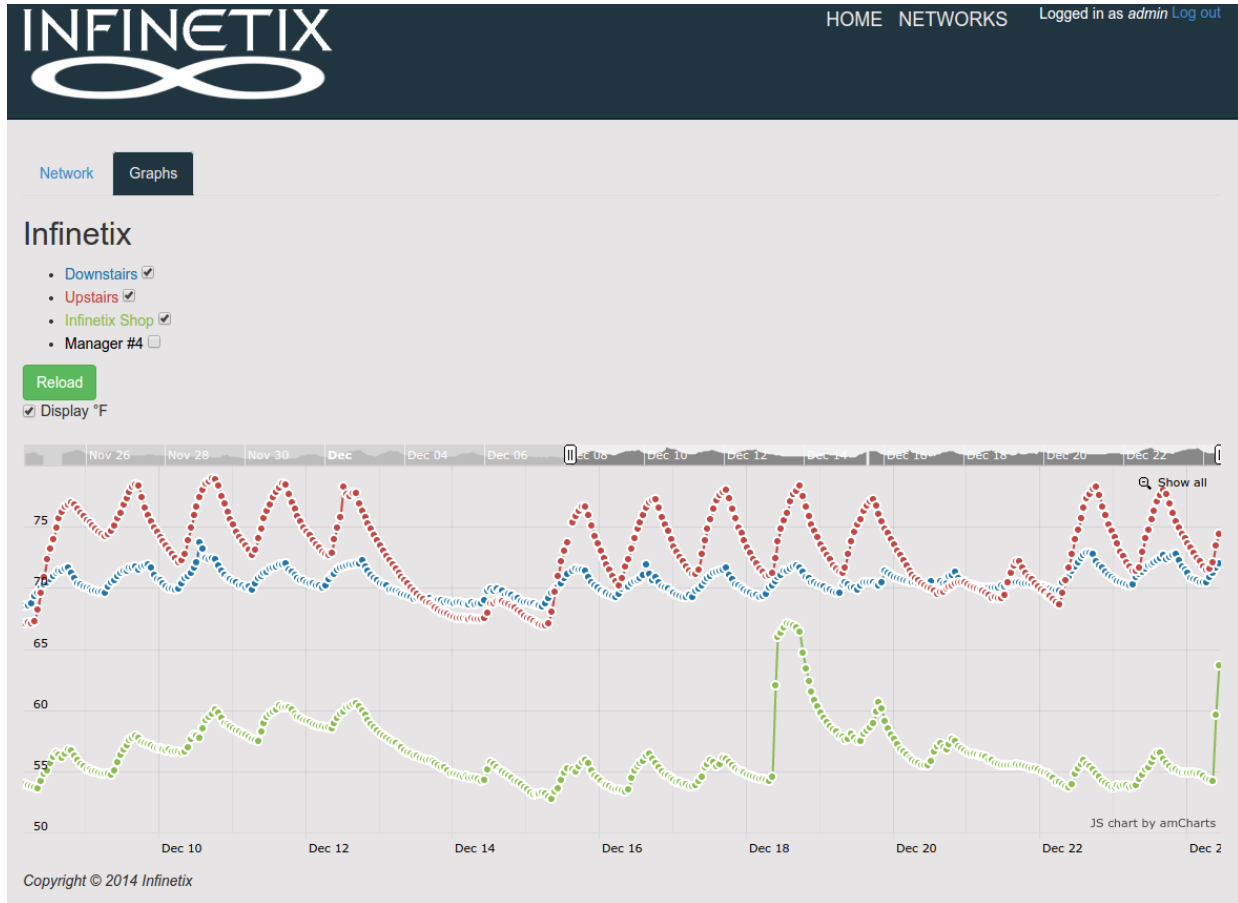
5. External Data collector web server

The external web server utilizes JS chart and amCharts to present and manage data collected and posted by the Gateway application. Below is an example of the network display, showing the interconnects between the motes and manager, with one of the motes selected to show more detail.

The screenshot displays the Infinetix web interface. At the top, the Infinetix logo is on the left and 'HOME NETWORKS' is on the right. Below the logo, there are tabs for 'Network' and 'Graphs'. The main content area is divided into two sections. On the left, under the 'Infinetix' heading, there is a list of network elements: 'Downstairs', 'Upstairs', 'Infinetix Shop', and 'Manager #4'. Below this list are two green buttons: 'Add New Mote' and 'Config Code'. A white box titled 'Infinetix Shop' contains the following information: '00:17:0D:00:00:60:20:23', 'DC9003A-B SmartMesh IP Evaluation/Development Mote', and a 'connections:' section with three entries: 'Downstairs(00:17:0D:00:00:60:21:88) RSSI:(94)', 'Upstairs(00:17:0D:00:00:60:20:18) RSSI:(87)', and 'Manager #4(00:17:0D:00:00:60:61:6B) RSSI:(84)'. On the right, a network graph shows four nodes: 'Upstairs' at the top, 'Downstairs' at the bottom, 'Infinetix Shop' on the left, and 'Manager #4' on the right. All nodes are represented by green circles, and they are interconnected by lines forming a diamond shape. A yellow starburst icon is positioned next to the 'Infinetix Shop' node, indicating it is the selected mote.

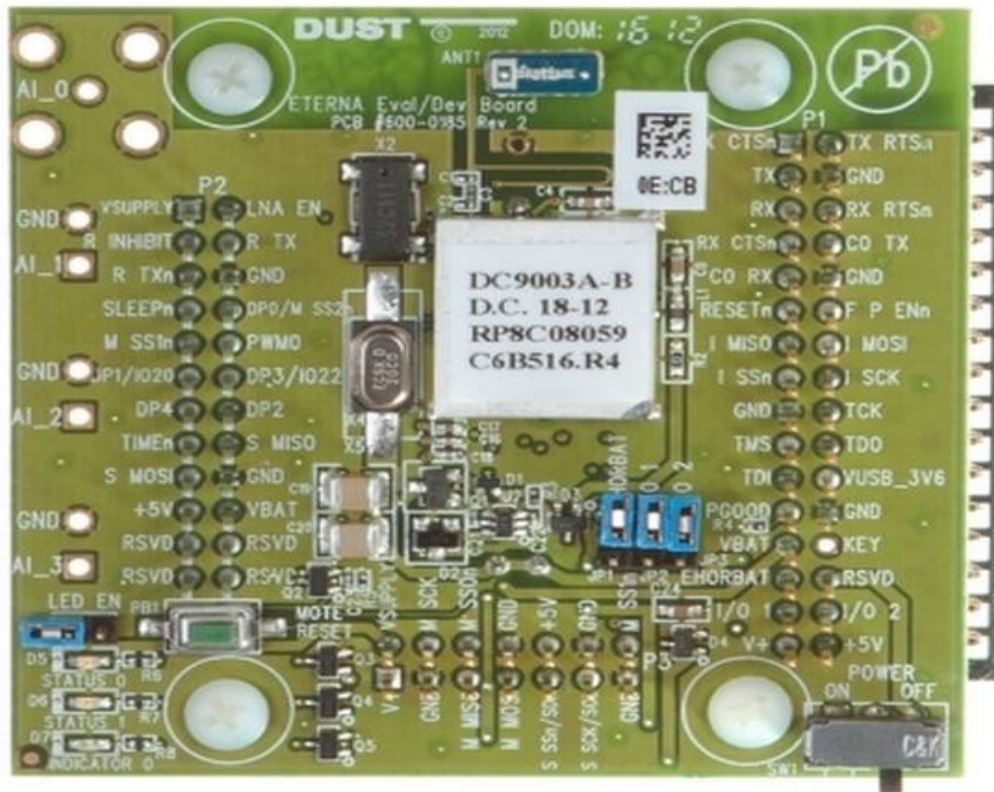
Copyright © 2014 Infinetix

And below is an example of a couple of weeks of temperature data taken using this network.



4 Mote Development

1. Linear Technology/Dust Networks Evaluation Mote



DC9003A-B SmartMesh IP Evaluation/Development Mote Module

The DC9003A-B SmartMesh IP Evaluation/Development mote was designed by Linear Technology and is a great starting point in investigation and evaluation of the technology.

INFINETIX can provide assistance in utilization of these motes including

- Evaluation Mote Code adaptations/modifications
- Other sensors connected to the Mote module.

2. INFINETIX Motes

Stay tuned for a line of INFINETIX designed SmartMesh IP motes.

5 Reference Information

1. Linear Technology/Dust Networks SmartMesh IP overview (Reference: SmartMesh IP User's guide, section 3.1.1 Network Overview)

"A SmartMesh® network consists of a self-forming multi-hop, mesh of nodes, known as motes , which collect and relay data, and a Network Manager that monitors and manages network performance and security, and exchanges data with a host application.

SmartMesh networks communicate using a Time Slotted Channel Hopping (TSCH) link layer, pioneered by Linear's Dust Networks group. In a TSCH network, all motes in the network are synchronized to within less than a millisecond. Time in the network is organized into timeslots, which enables collision-free packet exchange and per-transmission channel-hopping. In a SmartMesh network, every device has one or more parents (e.g. mote 3 has motes 1 and 2 as parents) that provide redundant paths to overcome communications interruption due to interference, physical obstruction or multi-path fading. If a packet transmission fails on one path, the next retransmission may try on a different path and different RF channel. Building networks with sufficient redundancy requires following some simple deployment guidelines - these are outlined in the application note "Planning a Deployment."

A network begins to form when the network manager instructs its on-board access point radio (AP) to begin sending advertisements - packets that contain information that enables a device to synchronize to the network and request to join. This message exchange is part of the security handshake that establishes encrypted communications between the manager or application, and mote. Once motes have joined the network, they maintain synchronization through time corrections when a packet is acknowledged.

An ongoing discovery process ensures that the network continually discovers new paths as the RF conditions change. In addition, each mote in the network tracks performance statistics (e.g. quality of used paths, and lists of potential paths) and periodically sends that information to the network manager in packets called health reports . The Network Manager uses health reports to continually optimize the network to maintain >99.999% data reliability even in the most challenging RF environments.

The use of TSCH allows SmartMesh devices to sleep in-between scheduled communications and draw very little power in this state. Motes are only active in timeslots where they are scheduled to transmit or receive, typically resulting in a duty cycle of <1%. The optimization software in the Network Manager coordinates this schedule automatically. When combined with the Eterna low-power radio, every mote in a SmartMesh network – even busy routing ones – can run on batteries for years. By default, all motes in a network are capable of routing traffic from other motes, which simplifies installation by avoiding the complexity of having distinct routers vs. non-routing end nodes. Motes may be configured as non-routing to further reduce that particular mote's power consumption and to support a wide variety of network topologies.

At the heart of SmartMesh motes and network managers is the Eterna IEEE 802.15.4e System-on-Chip (SoC), featuring our highly-integrated, low power radio design, plus an ARM® Cortex™-M3 32-bit microprocessor running SmartMesh networking software. The SmartMesh software comes fully compiled yet is configurable via a rich set of application programming interfaces (APIs) which allows a host application to interact with the network, e.g. to transfer information to a device, to configure data publishing rates on one or more motes, or to monitor network state or performance metrics. Data publishing can be uniform or different for each device, with motes being able to publish infrequently or faster than once per second as needed.”

2. Linear Technology/Dust Networks Planning a Deployment (Reference SmartMesh IP Application Notes)

“Estimating Range:

Hardware integration choices influence how well devices can communicate with each other over a distance with antenna choice being the most obvious. Post-integration, device placement can change the effective range over orders of magnitude. At one end of the spectrum, devices placed on elevated poles or towers with clear line of sight to other motes in the network may have a range of 1000 m or more. At the other end, devices placed on the ground or next to large metal objects may have an effective range of 10 m or less. So when a customer asks you “What’s the range of your radios”, in some ways that is a meaningless or unanswerable question. You can refer to the datasheet for transmit power and receive sensitivity and the resulting link budget, but the customer will determine the range with choices they make in the development of their products and an evaluation in a real environment similar to their expected deployments.

We recommend that customers at the beginning of development plan on their devices working at a spacing of 50 m. Analysis of the first several ‘typical’ deployments can guide the typical range number up or down. Deployment planning simply requires that each mote, is being placed within this range of at least three existing devices. In order to form a reliable mesh, every device must have multiple neighbors and hence numerous opportunities to connect. Placing motes within range of only one other device along a maximally spaced string will result in a fragile network prone to mote resets and data loss.

Mapping out a Deployment:

Once you have settled on a range for your environment, you can use a scale map to place motes at all the required sense points for the network. If possible, the AP should be located near the middle of the distribution of motes to reduce latency and mote power. Mark the AP location. Supposing that the range estimated above is 50 m, draw a circle with a 50 m radius around the manager. Not all motes within this circle will be able to communicate directly with the AP, but some motes outside the circle will, so on average it will balance out. The number of motes inside this circle approximates the number of 1-hop motes in the deployment.

Next draw a 100 m radius circle centered at the AP. The number of motes in the ring between 50 m and 100 m approximates the number of 2-hop motes. Repeat this process with circles of increasing radius until all motes have been encircled and note how many motes are in each hop. . . . There are two more things to check:

- Each mote, including the AP, should be within the estimated range of 3 other devices.*
- The network should be no more than 8 hops. Deeper networks are indeed possible and should just work, but they are harder to model.”*

Linear Technology, LT, LTC, LTM, Dust, Dust Networks, Eterna, SmartMesh and the Linear logo are registered trademarks and LTP, SmartMesh IP and the Dust Logo are trademarks of Linear Technology Corporation. The INFINETIX logo is a trademark of INFINETIX Corp. All other trademarks are the property of their respective owners.