Reader Info Method

Ben Duggan

April 1, 2019

1 General notes

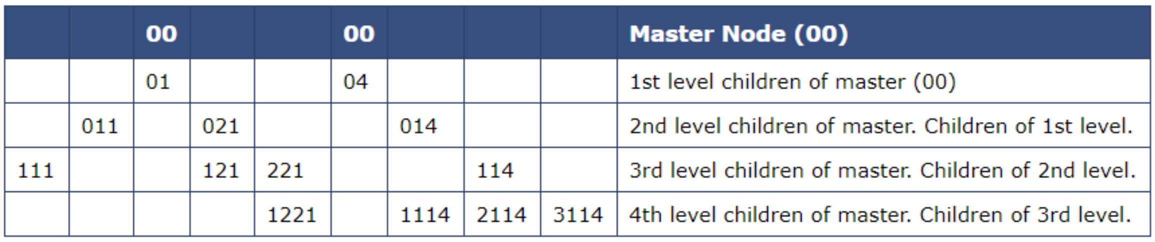
- You will need to use node.js when running this code. Download it from online. To run the website code navigate to the server folder. Then type *npm install express mysql socket.io winston* to install the server dependences. You only have to do that step the first time. Next run the application by typing *node index*. You will see some errors about connecting to MySQL DB but ignore those. Navigate to *localhost:4000* on your web browser and go to the Readers page. This is where you're working.

2 Goal

The goal is to create a network diagram of the radios used. The radios store the tree in an array with elements having a nodeID, unique id, and address, the address of the radio. The tree is a 5-arry tree with an address of 0 being the master node (Raspberry Pi). The address represents the path to get from itself to master. Addresses have the following rules (from `http://tmrh20.github.io/RF24Network/`):

- Node 00 is the base node.
- Nodes 01-05 are nodes whose parent is the base.
- Node 021 is the second child of node 01.
- Node 0321 is the third child of node 021, an so on.
- The largest node address is 05555, so up to 781 nodes are allowed on a single channel. An example topology is shown below, with 5 nodes in direct communication with the master node, and multiple leaf nodes spread out at a distance, using intermediate nodes to reach other nodes.

The main goal is to create an enhanced, dynamic and responsive table similar to this one from `http://tmrh20.github.io/RF24Network/`:



		00			00				Master Node (00)
		01			04				1st level children of master (00)
	011		021			014			2nd level children of master. Children of 1st level.
111			121	221			114		3rd level children of master. Children of 2nd level.
				1221		1114	2114	3114	4th level children of master. Children of 3rd level.

3 Implementation

All of your code will be implemented in readers.js which is found in server/dependences on the GitHub repo, `https://github.com/BenSDuggan/RFID-Network`. You will add all of your code to the createMeshTable() method in that file. The method takes an argument addrList which is an array of JSON objects where each element is a readers information. The key-value pairs that you care about are ”id” which is the nodeID and ”address” which is the address.

You will be making a HTML table to represent the network tree. The data is inserted into a table with id ”reader-MNC”. This means that you just have to write the internal HTML (append this to 'content'). You shouldn't have to touch much CSS as that is handled by BootStrap. A good reference on HTML tables is from W3 Schools.

The actual data is stored in readers.json in server/RFIDNetworkPi. The sample data in there is but you can add your own test data.

4 ToDos

1. Take addrList and add all data to HTML table, ignoring responsiveness and style, similar to the table above but without the description of each level. Here are some recommended sub-goals.

   (a) First create a new array (or JSON) variable used to store the nodes the level of each element from addrList. This can be achieved by casting ”address” to a string and getting the length.

   (b) Next iterate over each level of that array using a for loop. During each iteration organize the elements inorder, create a new row and add the id and address to the table.

   This should give you something similar to the table above.

2. Edge cases. There are two edge cases: when the address is 0 and when its ””. Both cases mean that the node is not in the tree but was once there. Add the ids of these nodes to the div with id=”reader-deadNodes”. Again you can just append to content in the later part of the method. Here are some recommended subgoals:

   (a) Before adding any data to the table (because we don't want dead nodes in the table), create a new array, iterate through addrList, add elements with address==0 —— address==”” to the new array and remove them from addrList.

   (b) Iterate through that array and append them to content.

3. Add color.